Exit Through the Training Data: A Look into Instance-Attribution Explanations and Efficient Data Deletion in Machine Learning

Jonathan Brophy
Department of Computer Science
University of Oregon
jbrophy@uoregon.edu

Abstract

The widespread use of machine learning models, coupled with large datasets and increasingly complex models have led to a general lack of understanding for how individual predictions are made. The GDPR has even stated that any individual has a "right to an explanation" from an automated decision if that decision can significantly impact their life. It is perhaps unsurprising then that explainable AI (XAI) has become a very popular research topic over the past several years. This survey looks at one aspect of model interpretability: *instance-attribution* explanations; these techniques are able to trace the prediction of a test instance back to the training instances that contributed most to its prediction. This technique is used as a means of debugging and improving models and datasets.

Another issue confronting many institutions today is that of data privacy and ownership. Debates surrounding these topics have resulted in legal action; for example, the GDPR states that companies must comply with removing user data upon request. Removing user information from databases is straightforward; however, machine learning models are not inherently designed to accommodate such a request. Practitioners can always retrain a model from scratch with the requested samples removed, but this quickly becomes prohibitively expensive. Thus, the second part of this review focuses on the task of *efficient data deletion* from machine learning models. In other words, these works are able to remove the effect of one or multiple training instances without having to retrain the model from scratch.

For both research topics, the current landscape of works is provided, including the advantages and disadvantages of each, open research questions, and promising research opportunities. Then, a discussion of how these research subfields are related is presented, and a practical use case of identifying and mitigating bias/misinformation from universal language models is provided; realizing this use case will henceforth require innovation in both of these areas.

1 Introduction

The growth and expansion of machine learning in our society has brought predictive modeling to almost every facet of our lives, but with its power and ubiquity comes unprecedented and challenging problems. One of those problems is related to the increasing size and complexity of many machine learning models (e.g. deep learning architectures, model ensembles, etc.). Although impressive from a predictive standpoint, these models are not easily understood, both from a global perspective of the model (e.g. looking at the weights of a model) or from a local perspective (e.g. how did my model make *this* particular decision?). For many decades, predictive performance was the main attributing factor to the success of a resultant model. This is certainly still an important factor for any predictive model created today, but more emphasis is now being put on *how* these models are making decisions, not just what they are predicting.

It may not be critical to know exactly how an auto-correct model fixes your text message, or how one's Google search results came about, but there are many scenarios where automated decision-making processes can have a significant impact on an individual's life. For example, a person may be denied a bank loan by an automated system run by a bank. In a bad (but not unrealistic) scenario, the person may be denied any explanation as to why they were not approved for a loan. Even worse, the employees using the system may not understand how or why their model came to that specific decision. The model may have even been trained on biased demographic data, where the most predictive attributes may be the person's ethnicity or living address. These problems of transparency (not given an explanation as to how the decision is made) and ignorance (not knowing how the decision is made) are some of the problems that can occur when using predictive modeling; this is especially important when these decisions can significantly impact people's lives. For a more detailed account of this and other scenarios, please consult the book *Weapons of Math Destruction* (O'Neil 2016) for a great primer.

The lack of transparency and interpretability in machine learning models can decrease user understanding of these models, which may ultimately lead to their decreased use. Europe has taken initial steps to address some of these issues in their GDPR bill[1], where one article in the bill states that consumers have a "right to an explanation" (Goodman and Flaxman 2017) for any automated decision that can significantly impact that person's life (e.g. bank loan, medical diagnosis, etc.). Thus, it is not surprising to see the subfield of interpretability in machine learning rapidly expanding; this

[1] https://www.privacy-regulation.eu/en/index.htm

is evident from the increasing number of papers published in this area, the creation of new workshops and conferences dedicated to explainable AI (XAI), and the increasing number of industry-backed tools[2] created for the purpose of understanding and improving predictive models learned from data.

There are many ways to view interpretability in machine learning, as the term is ill-defined (Lipton 2018), but interpretability is generally described as a method of analyzing the model (i.e. analyzing the components of the model or seeing how changes to the input can change the outputs of the model) and/or data used to train the model to better understand the model's behavior; understanding the model at a deeper level can then help to debug, improve, and increase trust in the model, among others (Miller 2018). To get a more comprehensive background of model interpretability, please review any of the increasing number of introductions[3] to machine learning interpretability and the many methods already created for virtually every subfield of AI (Gunning 2017).

This paper studies a relatively new and particularly interesting angle of model interpretability, that of understanding model predictions through the lens of the training data. Since every machine learning algorithm is dependent upon the data it is trained, one avenue for understanding how a model makes a decision for a new instance is to trace that prediction back through the model to the specific training instances most responsible for that decision. For example, if your image classifier labels a picture of a dog with cats in the background as a cat, then it would be beneficial to have a system that could trace that prediction back to examples in the training data that may also have a dog with cats in the background labeled as cats. This ability to find noisy or problematic instances in the training data not only helps improve the dataset, but ultimately improves the resulting model, and any other models that train on this dataset. This small example helps emphasize the need to answer the question: "what is it about my training data that led my model to make this particular prediction?" This is especially important as the size of datasets continue to increase, where it is simply not possible for one to manually inspect all data points.

These types of explanations are called *instance-attribution* methods, as each training example attributes more or less to the model's prediction of the target instance. Thus, the first part of this paper introduces and analyzes the different methods for generating instance-attribution explanations developed for different machine learning models. Then, the second part of this paper looks at the burgeoning area of *efficient data deletion* in machine learning.

Imagine a scenario where one has found a number of problematic training instances in the data after they have trained their complex neural network model for weeks. They subsequently remove the offending training examples from their database and would like to update their model to reflect these changes. Unfortunately, most current machine learning models are not well-equipped to handle these types of requests. The only option for most machine learning practitioners is to retrain their model on the newly revised dataset. If one can "remove" the effect of a training sample from a learned model, then the practitioner may be able to avoid retraining their model from scratch, saving time and computational resources.

The ability to effectively remove training samples from a learned model not only has time and money saving potential, but may also become a more necessary option as data privacy and ownership become more mainstream (Veale, Binns, and Edwards 2018). Once again, the GDPR has attempted to address this issue by mandating that companies remove a user's data if that user so desires (Villaronga, Kieseberg, and Li 2018). Currently, the details of what a company must do to satisfy this request is under debate (Kwak et al. 2017), but it may soon be a reality that all traces of that user's data must be removed from the company, including all machine learning models.

Deleting someone's personal data from a database is a relatively simple task, as databases are designed to efficiently handle this type of action, but most machine learning models are not setup to accommodate this kind of request. The resultant model from a machine learning algorithm is generally a condensed view of the training data, summarizing the statistics of the data well enough to make useful predictions on future unseen data. Thus, the second part of this paper analyzes the current progress of enabling efficient data deletion in machine learning models, and outlines promising future directions in this area.

For each of these two broader topics: *instance-attribution explanations* and *efficient data deletion*, a more in-depth background is provided to give the reader a better sense of what these approaches are, and why they are useful. Then the current landscape of work is shown as well as the limitations of each approach, and the gaps in knowledge that need filling. Finally, a realistic use case is presented and a proposal for future study is outlined.

2 Instance-Attribution Explanations

2.1 Background

The complexity and ubiquity of machine learning models has boosted the subfield of Explainable AI (XAI) into the awareness of researchers around the world. This is evident from the newly emerging workshops and conferences dedicated to XAI, and existing conferences that are putting more of an emphasis on model interpretability; there are also a number of academic and industrial-supported libraries to help users better understand their model's decision-making processes.

From a high-level perspective, model interpretability can impact all levels of the machine learning pipeline, from the machine learning practitioner, to a high-level executive, to the consumer of the model's decisions. As a machine learning practitioner, knowing how your model makes decisions is crucial for being able to improve and/or debug it. As an executive of a company, one may be less well-versed in the details of machine learning algorithms than the practitioner,

[2] https://github.com/microsoft/interpret

[3] https://xaitutorial2019.github.io

but knowing how a model makes predictions may impact how business decisions are made and how the model is deployed, if at all. From a user perspective, an accompanying explanation of how a model makes a particular decision can engender trust from the user. This trust may then allow models to be used in more and more diverse situations. However, one must be wary of the possibility of misleading explanations, whether it be purposeful or accidental; this is discussed further in §2.9.

This paper reviews all work related to instance-attribution methods. Before analyzing these approaches, it is necessary to introduce a closely related form of interpretability: *feature-attribution* methods. Feature-attribution techniques, a form of interpreting model predictions through the attribution of individual features, have garnered much attention in the past few years.

These methods allow one to answer the question, "What part of my input caused my model to make this prediction?" Many models offer a global perspective about what features are predictive; such as the coefficients in a linear regression model or the feature importances in a tree ensemble, but feature-attribution methods allow one to see which features are most important for *this* particular input, and quantify how much each feature is contributing to the final prediction, whether it be a single classification or a ranking (Hoeve et al. 2018).

Seminal work by Ribeiro et al. (Ribeiro, Singh, and Guestrin 2016) called LIME explains the prediction of a complex model (e.g. a neural network (Hecht-Nielsen 1992) or tree ensemble (Banfield et al. 2006)) using a surrogate linear model. LIME takes the input instance and slightly perturbs that input many times, then fits a linear model to those perturbations and produces an explanation in terms of the linear model's coefficients. This process of explaining a complex model with a simpler, arguably more understandable model such as a linear model is popular among feature-attribution methods (i.e. it is easier to reason about a model with less internal components such as a linear regression model than one with more such as a deep neural network).

More recent work by Lundberg et al. (Lundberg and Lee 2017) recognized the relations between LIME and several other works, and tied them together under one framework called Shapley values. Efficient implementations of this framework for different models allow anyone using complex machine learning models to analyze their model's predictions through the lens of the input features (Lundberg, Erion, and Lee 2018).

Analagous to feature-attribution techniques, instance-attribution methods allow one to answer the question, "Which part of my training data is most responsible for my model's prediction?" As all machine learning models are built using data, it seems only natural to better understand a model's behavior by finding the training instances it deems most important. Taking this perspective on model interpretability brings unique advantages when compared to feature-attribution methods; such as the ability to debug noisy or problematic data, increase user trust by showing similarly predicted or highly influential instances to the test instance (Zhou et al. 2019), increase model understanding, and create adversarial examples that expose flaws in the model and may lead to more robust models.

These techniques can and likely should be used in conjunction with other interpretability techniques like feature-attribution methods to get the most comprehensive view of how one's machine learning model behaves. Next, the current landscape of techniques providing instance-attribution explanations are shown, starting with the simplest and most naive, leave-one-out retraining.

2.2 Preliminaries

As previously mentioned, the term *interpretability* is ill-defined (Lipton 2018); thus, in this paper we define interpretability in the context of instance-attribution explanations as any beneficial actionable insights resulting from an instance-explanation of a test instance prediction for a given model. For example, an instance-attribution explanation of an incorrectly predicted test instance can identify noisy or problematic training instances, prompting the beneficiary of the explanation to remove those instances, improving the dataset and resulting models trained on the newly cleaned training data.

This paper focuses on instance-attribution methods designed for classification models. In classification, given a dataset $D = \{x_i, y_i\}_{i=1}^{n}$ where n is the number of training instances, $x_i = \{x_i^1, x_i^2, \cdots, x_i^k\}$ denotes instance i with k attributes, and y_i is the label of instance i taking on one of c labels; the goal is to predict the label of a new test instance x_t, denoted $\hat{y_t}$. The following instance attribution techniques attempt to estimate the effect of each training instance on $\hat{y_t}$. Finally, we denote Θ as the parameters of a neural network model.

2.3 Leave-One-Out Retraining

The most straightforward and naive approach to quantifying the "effect" of a training instance on a model's prediction of a test instance is to take out that training instance, retrain the model, and make a new prediction on the same test instance; the "effect" of that training instance is the resulting difference between the two predictions.

The benefit of this approach is the simplicity and ease of implementation, as this method can work with virtually every machine learning algorithm in existence. This may even be favorable if the model being trained is very simple the data they use for training is very small. This approach gives one perspective of how much each training instance influences the model's prediction.

Unfortunately, most real-world datasets are too big for a model to retrain for every instance; models are also increasingly complex, taking hours, days, or even weeks to retrain for even a single example. Thus, several new approaches designed primarily for more complex models (e.g. neural networks, tree ensembles) aim to approximate the results you would get from performing leave-one-out retraining without having to explicitly retrain the model from scratch; the most notable of these methods is called *influence functions*.

2.4 Influence Functions

Influence functions (Cook and Weisberg 1980) were first introduced in the field of robust statistics, and allows one to answer the question, "How much does training instance x_i affect the prediction $\hat{y}$ if x_i is upweighted by an infinitesimal amount?" Influence functions were first developed to work with ordinary least squares (OLS) regression, where it can measure the effect of a training instance on a test prediction without having to retrain the model. Wojnowicz et al. (2016) introduced influence sketching, which extends Generalized Cook's Distance (Pregibon 1981) (the generalized version of Cook's Distance (Cook 1977)) to work for larger scale regression datasets by injecting random projections into its construction. They are able to apply this methodology to generalized linear models (GLMs) and demonstrate the ability to detect highly influential samples on a cybersecurity data set.

Koh and Liang (2017) were the first to adapt influence functions to a broad range of neural network architectures, where they define the influence of a particular training instance on a test instance prediction as:

$$\mathcal{I}_{up,loss}(x, x_t) = -\nabla_\theta L(x_t, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(x, \hat{\theta}) \qquad (1)$$

where $\mathcal{I}_{up,loss}(x, x_t)$ represents the influence of the training instance x on the loss of the test instance x_t if x were upweighted by an infinitesimal amount. $\hat{\theta}$ is the empirical risk minimizer given by:

$$\hat{\theta} \coloneqq \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L(x_i, \theta) \qquad (2)$$

assuming the empirical risk is twice differentiable and strictly convex in θ. The rest of equation (1) contains the derivative of the test instance loss $\nabla_\theta L(x_t, \hat{\theta})^\top$, the derivative of the training instance loss $\nabla_\theta L(x, \hat{\theta})$, and the inverse Hessian matrix $H_{\hat{\theta}}^{-1}$. The inverse Hessian matrix is an expensive task, but Koh and Liang use the LiSSA (Agarwal, Bullins, and Hazan 2017) algorithm to efficiently compute Hessian vector products instead of explicitly computing the entire matrix.

This definition of influence can be interpreted in several ways: first, this is the change in test loss as the training instance is infinitesimally upweighted. This can also be viewed as the proportion between the derivative of the training loss and the derivative of the test loss; if both are big and in the same direction, this represents a greater influence of the training instance on the test instance prediction. From the model's perspective, influence functions can be seen as a similarity measure between the training instance and the test instance. Finally, this is roughly equivalent to a second order Taylor expansion, where the expansion occurs around the test loss, and moves in the direction of the training loss derivative.

Applying influence functions results in an approximation to leave-one-out retraining, and its ability to be efficiently computed on more complex models opens up new insights to these types of models previously unattainable. For example, Figure 1 shows the most influential training samples on the prediction of a test image (Figure 4 from (Koh and Liang 2017)). The samples are shown for two different types of image classifiers, and the highly influential samples highlight some of the differences between these models. The SVM concentrates more on color, while the Inception network is able to better capture the characteristics of the clown fish. The most helpful samples for the SVM are fish images, while dog images generally hurt the prediction; this is in constrast to the Inception network, where dog images generally do not hurt the prediction and in one case actually helps (Figure 1: top-right). Also, influence is not necessarily correlated with euclidean distance.

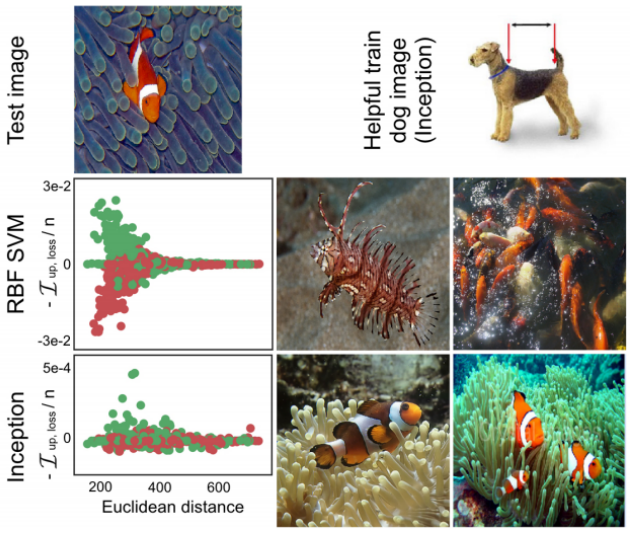

Figure 1: The top two most influential samples on the prediction of "fish" for a test image from the SVM and Inception image classifiers. Also, the influence of each sample (green for fish images and red for dog images) are plotted against the Euclidean distance to the test image in terms of pixel space (Figure 4 from (Koh and Liang 2017)).

This approach by Koh and Liang provides the ability to compute the influence of training examples on a test prediction from any model with a convex and twice differentiable loss function, which covers many neural network-type models and many other differentiable models such as SVMs; however, these are not the only high-performing and ubiquitous models in use today. Tree-based methods, in particular tree-ensemble approaches such as random forest (Breiman 2001) and gradient boosted decision trees (GBDT) (Friedman 2001) are still very active areas of research, with new implementations of the GBDT framework (Chen and Guestrin 2016; Ke et al. 2017; Prokhorenkova et al. 2018) being consistently released for the past several years, many of which are backed by industry support.

Influence functions as proposed by Koh and Liang cannot be directly applied to GBDT, as the step-wise nature of attribute-based splits in a decision tree make the

Figure 2: A misclassified test instance (left) and the four most inhibitory samples to the test instance's true label (Figure 5 from (Yeh et al. 2018)).

entire model non-differentiable. Fortunately, Sharchilev et al. (2018) recently proposed an extension of influence functions to tree ensembles, focusing mainly on GBDT.

They developed two algorithms, *LeafRefit* and *LeafInfluence*, as well as faster versions of each one. *LeafRefit* makes the assumption that the effect of removing a training instance can be estimated assuming the tree structure remains fixed. With this in hand, they are able to update the leaf values and measure the effect of the target training instance. For *LeafInfluence*, they no longer need this assumption but rely on the intuition that a small enough perturbation of a target training instance will not disrupt the tree structure. This allows the leaf values to change smoothly with the weight of the training instances when computing the derivative of the test loss with respect to the training weights, resulting in an approximation of leave-one-out retraining for GBDT models.

They then show that their methods are able to accurately approximate the effect of a training instance on a test instance in an information retrieval setting using the metric normalized discounted cumulative gain (NDCG) (Wang et al. 2013). They also demonstrate how they are able to find problematic training instances for their model, and how removing those instances from their model improves performance. Finally, they are able to detect some domain mismatch in a hospital dataset; however, it should be noted that this dataset was intentionally modified to introduce this mismatch.

Their method hinges upon one configuration, the update set, which trades off computational complexity for accuracy. This update set tells their algorithm which training-instance-prediction changes to pay attention to; paying attention to fewer and fewer instances speeds up computation, but sacrifices approximation quality, while larger update sets increase quality, but significantly increase the runtime. In their runtime experiments, it takes an average of two seconds to compute the influence of each training instance using the largest update set. If the training set is large, this may be a limiting factor when choosing to use this approach or not, as explanations may not be readily available in realtime.

Even with Koh and Liang's tractable adaptation of the influence functions framework to differentiable models, and any extensions of this methodology to inherently non-differentiable models, these approaches still require some potentially expensive derivative computations. This problem is exacerbated when the number of parameters in the model increases to extreme levels (e.g. neural networks with billions of parameters). Additionally, deep learning models typically need large amounts of training data to work well, increasing the time it takes to compute the influence of each and every training instance for one test instance. *Representer point* approaches are a response to these issues.

2.5 Representer Points

The Representer Point framework introduced by Yeh et al. (2018) is an alternative to influence functions. Their work hinges on the representer theorem (Schölkopf, Herbrich, and Smola 2001), which states that the optimal solution to many learning problems, specifically those applied to empirical risk minimization within a reproducing kernel Hilbert space (RKHS), can be represented in terms of the training examples. Yeh et al. show that the pre-activation predictions from the second-to-last layer of a neural network can be decomposed into a linear combination of the training point activations.

They show that if a model is optimized with an L2 regularizer:

$$\Theta^* = \arg\min_{\Theta} \frac{1}{n} \sum_{i}^{n} L(y_i, \phi(x_i, \Theta)) + \lambda |\Theta_1|^2 \quad (3)$$

where $\phi(x_i, \Theta)$ is the pre-activation prediction of training sample x_i and Θ_1 represents the parameters of the last intermediate layer of the network, then any pre-activation prediction $\phi(x_t, \Theta)$ of any arbitrary test point x_t can be decomposed as follows:

$$\phi(x_t, \Theta) = \sum_{i}^{n} k(x_t, x_i, \alpha_i) \quad (4)$$

where $k(x_t, x_i, \alpha_i)$ represents the contribution of training instance x_i on x_t's pre-activation prediction. $k(x_t, x_i, \alpha_i) = \alpha_i f_i f_t$ is a weighted similarity measure (in this case the dot product between the last layer feature values of x_i and x_t) where α_i is the "representer value" of training sample x_i defined by:

$$\alpha_i = \frac{1}{-2\lambda n} \frac{\partial L(x_i, y_i, \Theta)}{\partial \phi(x_i, \Theta)} \tag{5}$$

In practice, if one wishes to compute the representer values of an already trained network, one can retrain the last layer of the network, using the feature representation of the second-to-last layer as the input to this new model; then, train this last layer with an L2 regularizer and compute the representer values. Once these are computed, the pre-activation prediction of any arbitrary point can be decomposed as a sum of the training instance contributions using equation (4).

Figure 2 gives an insightful example of problematic training instances found using representer points. There is an incorrectly predicted test instance where some antelope are predicted as deer. In the negatively influential training samples, there are antelope in the picture, along with other animals such as elephants and zebras. These types of training images can confuse image classifiers into making incorrect predictions. The authors suggest this tool can be used to find and remove examples like these, or give these samples multiple labels and turn this into a multi-label classification problem.

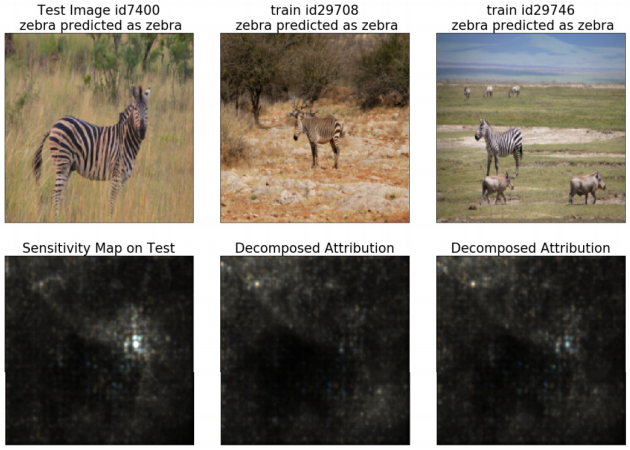

Figure 3: A test instance (left) and two positively influential training samples, including the saliency maps for each one (Figure 6 from (Yeh et al. 2018)).

They also provide an example of combining two types of explanations, instance and feature attribution methods (Figure 3). A test image with its prediction and two positively influential training samples are shown, along with their corresponding sensitivity maps generated by SmoothGrad (Smilkov et al. 2017). These sensitivity maps essentially show the user what parts of the image the model is using the most by computing the gradient of the class function with respect to the input image. Thus, this example not only tells us which training samples are important for this test instance, but also what parts of these samples it believes are important.

Finally, Yeh et al. demonstrate the efficiency of their approach. The representer point framework incurs a one-time fine-tuning cost (an average of $\sim$7s and $\sim$12s on the CIFAR-10 and AWA datasets) to compute the representer values, but then computing the contribution of all training samples is much faster ($\sim$0.1s and $\sim$0.2s to explain a single test example). Contrast this with influence functions, which incur no setup or fine-tuning costs, but took $\sim$267s and $\sim$172s to generate an explanation for a single test example on the CIFAR-10 and AWA datasets, respectively. Clearly, if generating real-time instance-based explanations is a priority, the representer point framework is a more suitable alternative to influnce functions.

Similar to influence functions, this representer point framework cannot be directly applied to non-differentiable models such as decision trees. In recent work, we have developed a new method called Tree-ensemble Representer-point EXplanations (TREX), a way to provide instance-attribution explanations for tree ensembles (mainly random forest and GBDT) based on the representer theorem. TREX works by approximating the behavior of a tree ensemble with a surrogate kernel model that fits the representer theorem criteria and enables its predictions to be decomposed as a sum of its training samples.

TREX uses tree ensemble kernels (Davies and Ghahramani 2014) to capture the structure of the trees and measure the similarity between data points. This kernel is used to train a kernel model (e.g. kernel logistic regression (KLR) (Yu, Huang, and Lin 2011) or a support vector machine (SVM) (Cortes and Vapnik 1995)) to approximate the original tree ensemble. The tree ensemble kernels are defined as dot products in an alternate feature representation defined by the feature mapping ϕ $k(x_i, x_j; T) = \phi(x_i; T) \cdot \phi(x_j; T)$. Note that the kernel is parametrized by T, since the computation necessarily depends on the structure of the tree ensemble. Figure 4 shows the different choices for $\phi(x, T)$: LeafPath (Bloniarz et al. 2016; Plumb, Molitor, and Talwalkar 2018; He et al. 2014), FeaturePath (Davies and Ghahramani 2014), and a new tree ensemble kernel called LeafOutput.

Recent work has also used tree-ensemble kernels to generate explanations. MAPLE is a model-agnostic explainer that computes the similarity of the test instance to the training instances, and uses this similarity (which they call the "local training distribution") as an instance-based explanation (Plumb, Molitor, and Talwalkar 2018). However, their approach *only* looks at the similarity between the test instance and the training instances, whereas TREX decomposes the test instance prediction as a weighted sum of the training instance similarities to the test instance.

Our results show that TREX is able to accurately approximate the predictive behavior of a tree ensemble, better debug and identify the most problematic training instances (Figure 5), and generate explanations faster than alternative methods such as LeafInfluence.

The approaches discussed thus far have all found some measure of similarity between data points through the model's perspective. This is similar to K nearest neighbor (KNN) (Altman 1992) whose test predictions are directly computed from its nearest neighbors in some feature space. Showing hese neighbors is a perfectly valid instance-based

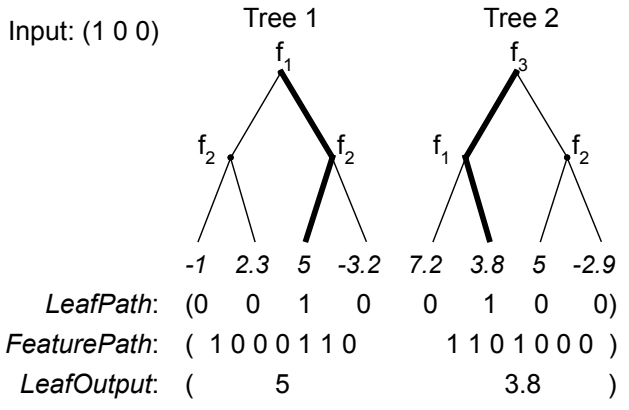

Figure 4: Three different transformations of a single data instance from a two-tree gradient boosted ensemble. Each f_i is a feature, and the numbers at the leaves represent leaf values. The lines in bold represent the paths taken through each tree in the ensemble given the input instance.

explanation of the test instance prediction, but KNNs in the original feature space are generally not as predictive as more complex models such as an ensemble of decision trees. TREX uses tree ensemble kernels to enter a transformed feature space where data points are more semantically clustered, a form of nearest neighbor for decision trees (Lin and Jeon 2006); TREX additionally computes representer values to weight those similarities to generate instance-attribution explanations. The work by Yeh et al. can also be framed this way; they use all intermediate layers as a feature transformation, and compute the similarity between data points using this new representation.

2.6 Data Shapley

Ghorbani and Zou inroduce data Shapley (Ghorbani and Zou 2019) to evaluate the value of individual training points (Raskar et al. 2019). Their method is based on Shapley values (Shapley 1953), a game theoretic approach to evaluating participant contribution to a shared coalition.

Data Shapley defines the value of a data point in a dataset D = $\{x_i, y_i\}_{i=1}^n$ as follows:

$$\phi_i = C \sum_{S \subseteq D - \{i\}} \frac{V(S \cup \{i\}) - V(S)}{\binom{n-1}{|S|}} \tag{6}$$

where ϕ_i represents the valuation of data point i, S is a subset of D not including training point i, and $V(S)$ represents the predictor performance when training the model on the subset S. Thus, this equation can be interpreted as the average change in predictor performance when training with and without training point i using all possible subsets of D not including training point i. This is close to leave-one-out (LOO) evaluation where they evaluate ϕ_i as $V(D) - V(D - \{i\})$. However, they argue this simpler alternative only uses one possible subset, and does not satisfy the equitable valuation properties outlined in Ghorbani and Zou's work (2019).

Since equation (6) is typically intractable to compute exactly, Ghorbani and Zou introduce an approximation to this measure called truncated Monte Carlo (TMC-Shapley), in which they sample random permutations of the training data, and add a new data point only if adding it improves performance over using only the previous data points. They also find that increasing the training size decreases the marginal contribution of each sample to the point where each marginal contribution is approximately zero, at which point they stop the computations of additional data points.

Using TMC-Shapley, Ghorbani and Zou showcase a number of use cases for different datasets. For example, they modify a 3000 instance spam dataset by flipping half of its labels; they then order the training instances to be checked by their valuations, and fix them when inspected. In most cases, their approach orders the training data such that checking and fixing the noisy instances is more efficient than checking the training data ordered randomly or by LOO. They also take an image dataset and add varying levels of white noise to 10% of the images, and find that TMC-Shapley assigns lower value to these noisy images as they degrade the model's performance.

They also show how TMC-Shapley can be used for domain adaptation by removing training instances with negative valuations when evaluating on the target data. Overall, their approach of data valuation has solid theoretical grounding under the foundation of Shapley values, but their experiments lack any time comparisons with alternatives such as LOO; without this, it is hard to get a sense of how much faster their approach is, and this can have a big impact on its adoption and widespread use.

Jia et al. (2019b) also use Shapley values to approximate the worth of individual data points using various approximations to the exact Shapley computations. In related work, Jia et al. (2019a) compute the approximate and exact valuation of data points for a K-nearest neighbor classifier. Their approach is able to guarantee bounds on the approximation error, unlike the approach from Ghorbani and Zou.

2.7 Prototypes and Criticisms

Prototype selection attempts to select the minimum number of training samples that most comprehensively capture the distribution of the training set. The main motivation behind these approaches is to efficiently train classifiers on a small subset of a much larger dataset. However, recent methods have put an additional emphasis on interpretability, allowing one to inspect the prototypical examples on which these models are trained (Bien, Tibshirani, and others 2011).

A Bayesian case model has been developed using Bayesian optimization to generate case-based reasoning explanations and provide prototype selections (Kim, Rudin, and Shah 2014). Building on this work, Kim et al. (2016) introduce the idea of criticisms, which show training examples from areas in the training data that cannot be cleanly represented by prototypical samples; they show that this improves interpretability mainly in more ambiguous segments of the training data. Finally, Anirudh et al. (2017) developed a generic graph-based approach to finding the most influential samples in the training data.

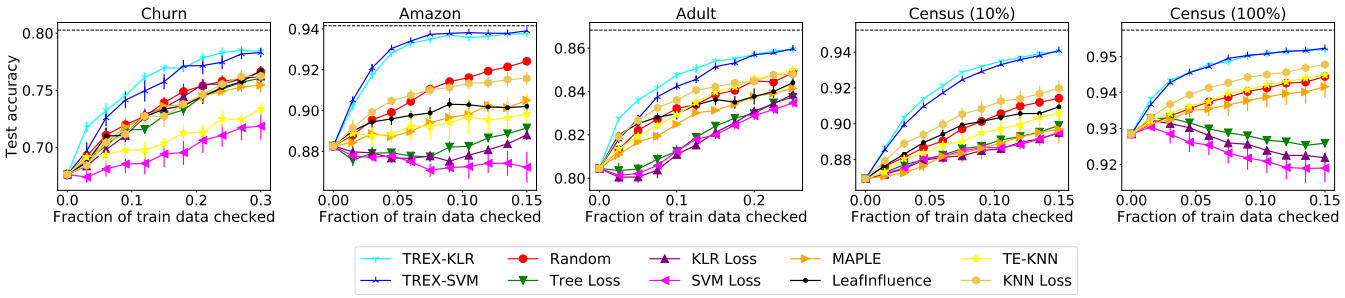

Figure 5: Change in test accuracy as training points are checked and fixed; the dashed line represents test accuracy before label corruption.

Prototype and criticism selection approaches are certainly related to model interpretability via the training data. However, these are not instance-attribution explanations; they provide a condensed global view of the training data, but cannot show how much each training instance contributes to a *specific* test prediction.

2.8 Additional Applications

There are a number of different beneficial use cases that arise from analyzing the impact of individual training instances on one or a group of target instance predictions, many of which take a form of dataset cleaning or debugging. For example, Nelson et al. (2008) use a form of leave-one-out retraining to find adversarially created email attacks on a spam filter training dataset. Sullivan et al. (2013) identify noisy data by finding the training samples most responsible for incorrectly classifying near ground-truth "demonstration" data points supplied by a demonstrator; they find these noisy points through model-specific heuristics (e.g. the training samples present in the parent node of the leaf that incorrectly classified the "demonstration" instance).

In contrast, some methods can even help create poisoning attacks, instead of defend against them. Influence functions can tell one exactly how to change a particular training instance to increase the loss on a target test instance. Koh et al. (2017) use this to generate noise that will alter specific training instances in a way that they are indistinguishable from the original, but cause the predicted labels to flip for the target instances (Figure 6). They demonstrate this approach on a model which correctly classifies 591/600 test instances. If they perturb just 10 training instances for each test instance, they can flip the prediction of all but one of the 591 predictions, making the classifier essentially useless. This highlights the ease at which new adversarial examples can be created, and may ultimately lead to more robust models as classifiers can use these techniques to generate adversarial examples and then come up with defenses against these attacks.

Brunet et al. (2019) use influence functions to measure the bias of word embeddings for systems such as GloVE (Pennington, Socher, and Manning 2014), where they simplify the computation of the influence function to the learned embedding of the GloVE model, resulting in efficient computation of differential bias for large corpora of text. Alternatively, Pezeshkpour et al. (2019) use influence functions to explain a model's link prediction by finding the most influential facts surrounding that relation; they also use this tool to study the robustness of link prediction models.

Schulam and Saria (2019) build on influence functions to approximate the uncertainty surrounding a target test prediction, and they use this method to reliably audit models after training. Ting and Brochu (2018) use influence funtions to come up with an optimal subsampling strategy for datasets that are too large.

2.9 Limitations and Research Opportunities

Analyzing model behavior via the training data has led to some intriguing and useful applications, but this one perspective of model interpretability has a number of limitations.

Scalability. One of the main problems going forward is scalability. Influence functions are still too slow to be used for very large models with millions or billions of parameters, and this problem is only exacerbated with the increase in data sizes. Representer points is an effective solution to this problem, though these approaches need to introduce a separate interpretable surrogate model to explain the behavior of the original complex model, and even these methods will likely face scalability issues as dataset sizes increase.

Multiple instances. Current approaches only show the effect of *individual* training instances on one or multiple target test predictions, though it may be beneficial to identify and compute the aggregate effect from groups of training instances. Some preliminary results for this line of work are underway (Koh et al. 2019; Basu, You, and Feizi 2019).

Robustness. Just as inputs to machine learning models are susceptible to adversarial manipulations (Narodytska and Kasiviswanathan 2017), model explanations are vulnerable to similar events. For example, Ghorbani et al. have shown that two inputs with visually indistinguishable differences can result in two very different feature-based explanations (in this case saliency maps), questioning the robustness of these explanations (Ghorbani, Abid, and Zou 2019). If feature-based explanation systems are vulnerable to these types of manipulations, than instance-attribution techniques are very likely to as well. Thus, new methods should be developed that strive towards robust instance-attribution explanations in the face of small changes to target test instances.

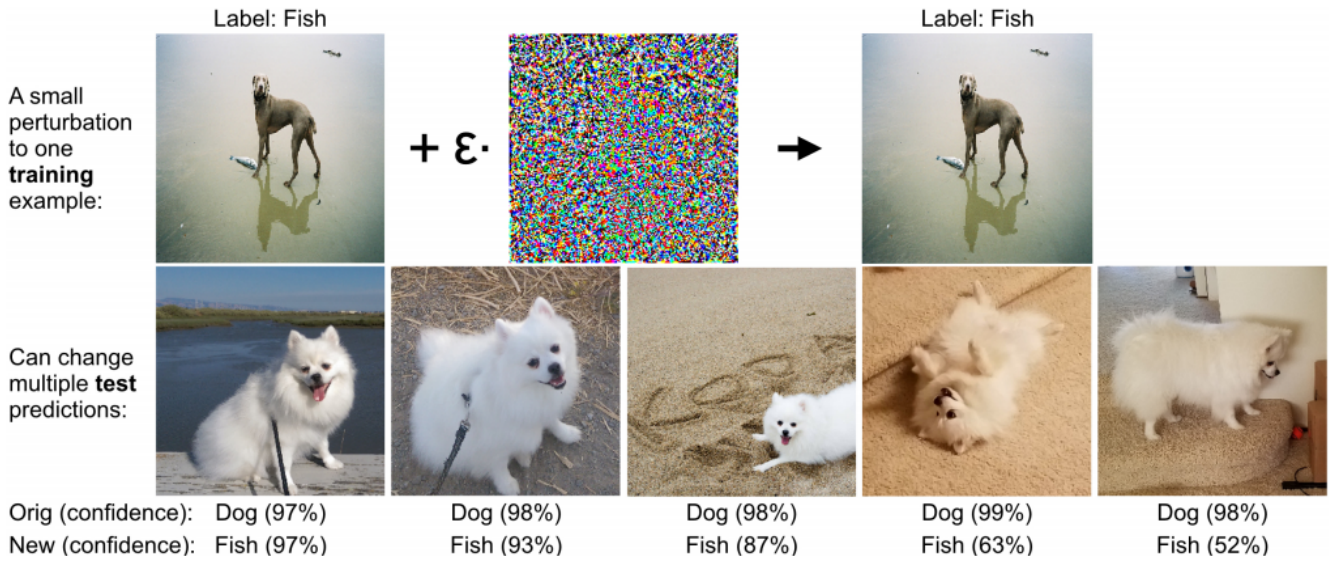

Figure 6: Example of adding visually imperceptible noise to a training image can flip the prediction of a number of target test instances (Figure 5 from (Koh and Liang 2017)).

Evaluation. Part of the problem with developing new instance-attribution techniques (as well as feature-attribution methods) is the lack of clear and consistent evaluation metrics (Doshi-Velez and Kim 2017). While there are ongoing efforts to standardize the evaluation of explanations for some models and problem domains (Yeh et al. 2018; Hooker et al. 2019), others believe post-hoc explanation systems should not be used at all, and instead researchers should devote their time to developing more inherently interpretable models that still perform well (Rudin 2019).

Generative Models There is plenty of opportunity to explain the behavior of generative models such as GANs (Goodfellow et al. 2014) and universal language models like BERT (Devlin et al. 2019) and GPT-2 (Radford et al. 2019). The approaches discussed thus far typically have well-defined training instances and a straightforward model architecture. In the case of neural language models, each training instance is not well-defined, as the input to the model are finite sequences of tokens; thus, explaining how a language model produced a specific output in terms of the training data is not immediately obvious. GANs are also not easy to explain, as the training procedure of a GAN is dependent not only on input training images, but also on images from the generative model. It is not clear how one would apply any of the methods discussed so far on these kinds of models.

Applications. Finally, although the number of different applications for instance-attribution explanation systems is growing, there is still much room for new and insightful applications of these techniques, both to new models as well as differing problem domains; this is especially important for problem domains where the training instances are not as visually digestible (i.e. text / image data vs tabular data).

3 Efficient Data Deletion

3.1 Background

The current debate about data privacy and ownership has sparked new discussions about whether or not someone has the right to remove their data from companies that use it (Shintre, Roundy, and Dhaliwal 2019). The GDPR has taken an initial stance in this debate and has mandated that companies remove user data upon request. This has led to subsequent discussions about the extent to which companies must fulfill such a request, and if they even have the capacity to do so (Kwak et al. 2017).

Removing data from standard databases, such as relational database management systems (RDBMS), is a simple task since these systems store data in the form of linked tabular data, making deletion from these tables straightforward and efficient. However, machine learning models can also be viewed as a type of database, designed to summarize the statistics of the training data well enough to provide useful predictions on future unseen data. This poses a problem, if machine learning models have a "memory" of the data they are trained on, they can then leak information about that data. Even if this data has been deleted from a companies' RDBMS system, the data continues to exist in their learned models.

In addition to legal obligations that motivate the need for models to update themselves upon request, there are also financial and interpretable reasons for efficient model updating. The complexity of many machine learning models today, coupled with increasing dataset sizes, have increased training times significantly; it can sometimes take days or even weeks to train a large model (Shoeybi et al. 2019)! If a company must remove data from its models, it may become prohibitively expensive to retrain these models with the specified data removed, especially if deletions

were to happen frequently. If a model can be efficiently updated without having to retrain, the company may save valuable resources. From an interpretable standpoint, generating an instance-attribution explanation via leave-one-retraining may be a viable option through efficient deletion.

There are two broad categories of model unlearning: exact and approximate. Definitions and examples of each are presented in the following sections. Then, challenges and promising research directions are discussed.

3.2 Exact Model Unlearning

The term *exact unlearning* describes methods that are able to remove the effect of training sample i from a machine learning model, resulting in a new or updated model that is exactly the same as if that model had been trained without training sample i.

Leave-One-Out Retraining The simplest way to implement exact unlearning is by leave-one-out retraining; this happens by removing one or multiple instances from the training data and retraining the model on the remaining instances. This is not only the simplest method to understand and implement, but it also works for essentially every machine learning algorithm in existence. Unfortunately, depending on the size and complexity of the dataset and model, retraining can often be a prohibitively expensive option. Thus, *decremental learning* techniques are a response to this problem.

Decremental Learning Decremental learning removes the effect of a training sample on a learned model by updating it, resulting in an exact unlearning model. Some models can already support this operation efficiently without modification; consider a simple naive Bayes model (Rish and others 2001) model with two binary attributes and a binary class label. This model can be represented by 5 parameters: $P(y = 1)$, $P(X_1 = 1|y = 1)$, $P(X_1 = 1|y = 0)$, $P(X_2 = 1|y = 1)$, $P(X_2 = 1|y = 0)$. These probabilities are simply counts from the training data (i.e. $P(y = 1) = \frac{\text{\# instances where y=1}}{\text{\# total instances}}$). Thus, deleting an instance from the learned model simply means updating these counts and probabilities, an O(1) operation; this can be done without retraining the model from scratch. K-nearest neighbor (Dudani 1976) is another model that can handle decremental learning, though it does not actually have a training phase. However, the majority of models, do not naturally support decremental learning; thus, the works described next typically alter the training algorithms in one way or another to produce a model capable of efficient data deletion.

Decremental learning has strong analogs to incremental/online learning, which was invented to handle scenarios of out of-core-learning (i.e. not all the data fits in memory) and streaming data (i.e. not all the data is available at one time). Incremental learning has been around for a long time (Schlimmer and Granger 1986), and some approaches even support both incremental and decremental learning. For example, it has been shown for SVMs that incremental (Diehl and Cauwenberghs 2003) and decremental versions can be made by keeping track of the computations that satisfy the Kaursh-Kuhn Tucker (KKT) conditions; if a deletion is requested, those statistics can be easily updated to support that operation while maintaining the KKT conditions (Cauwenberghs and Poggio 2001; Karasuyama and Takeuchi 2010; Romero, Barrio, and Belanche 2007; Chen et al. 2018). Cauwenberghs and Poggio use the decremental part of their SVM to perform leave-one-out evaluation efficiently. Incremental and decremental learning also exist for proximal SVMs (Mangasarian and Wild 2001) for both binary (Fung and Mangasarian 2002) and multiclass cases (Tveit, Hetland, and Engum 2003; Tveit and Hetland 2003). Finally, Duan et al. developed decremental learning algorithms for lagrangian and least squares SVMs using similar principles (Duan et al. 2007).

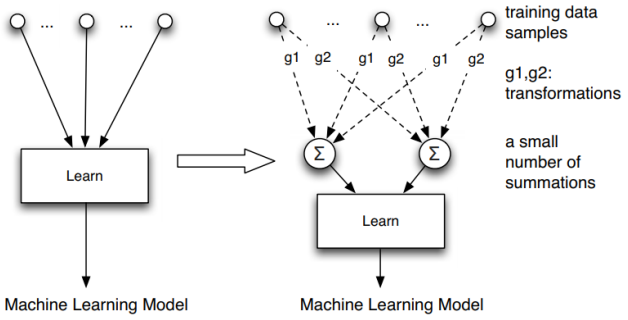

Figure 7: Adaptation of a learning algorithm to support SQ-learning (Figure 1 from (Cao and Yang 2015)).

More recent work by Cao and Wang (2015) have shown that a number of machine learning algorithms can be slightly modified to support SQ-learning (Kearns 1998), a form of learning where the algorithm does not depend on the individual training points directly, but rather on statistical summations of the data (Figure 7). They explain that the non-adaptive form of SQ-learning results in exact model unlearning, which they demonstrate with a naive bayes model and an item-item collaborative filtering model (Sarwar et al. 2001). On several experiments for different problem domains, they pollute the training data and compare their removal approach against retraining with a set of target samples removed; they find that their approach is significantly faster than retraining from scratch.

Cao et al. (2018) have developed a method called KARMA which identifies polluted training samples from a learning system and uses the approach by Cao and Wang (2015) to efficiently unlearn those samples from the system.

Quantization Stability is an idea from statistical learning theory which states that small changes to the dataset should result in small changes to the resulting model (Mukherjee et al. 2006; Shalev-Shwartz et al. 2010). Ginart et al. (2019) leverage this concept by developing a learning algorithm which results in a quantized model (their algorithm is specifically designed for K-Means). Their method uses additional data structures to store metadata for each quantized centroid from the initial learning process. This extra metadata

Dataset	k-means Runtime (s)	Q-k-means Runtime (s)	Q-k-means Speedup	DC-k-means Runtime (s)	DC-k-means Speedup
Celltype	4.241±0.248	0.026±0.011	163.286×	0.272±0.007	15.6×
Covtype	6.114±0.216	0.454±0.276	13.464×	0.469±0.021	13.048×
MNIST	65.038±1.528	29.386±0.728	2.213×	2.562±0.056	25.381×
Postures	26.616±1.222	0.413±0.305	64.441×	1.17±0.398	22.757×
Gaussian	206.631±67.285	0.393±0.104	525.63×	5.992±0.269	34.483×
Botnet	607.784±64.687	1.04±0.368	584.416×	8.568±0.652	70.939×

Figure 8: Amortized runtime of 1000 sequential deletions (Table 4 from (Ginart et al. 2019)).

Table 1: Dataset Statistics (Ginart et al. 2019)

Dataset	Samples	Dimension	K
Celltype	12,009	10	4
Covtype	15,120	52	7
MNIST	60,000	784	10
Postures	74,975	15	5
Gaussian	100,000	25	5
Botnet	1,018,298	115	11

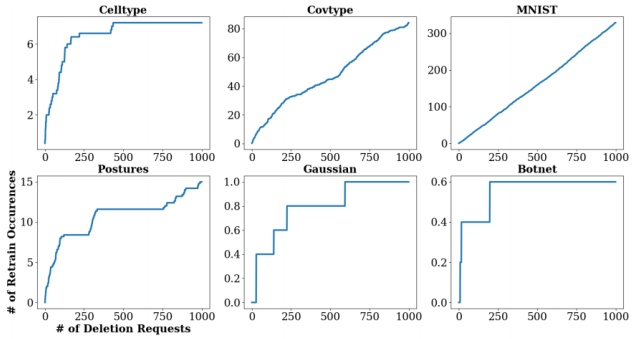

Figure 9: Average number of retrains on each dataset for Q-k-means (Figure 2 from (Ginart et al. 2019)).

allows them to quickly calculate whether deleting a specific instance would result in a different centroid. If a different centroid is optimal or the chosen sample to delete is one of the randomly initialized centroids, then the model is retained from scratch. The core idea of their approach is that a small number of deletions does not result in a large change in the resulting model. Additionally, they find that the amortized complexity of their algorithm for a single deletion operation is low.

They also define a model as deletion efficient if it can meet a lower bound of $\Omega(\frac{n}{m})$ where n is the number of training samples and $m \leq n$ is the number of sequential deletions. In order to empirically test their method, they analyze the amortized runtime of deleting a sequence of 1000 training samples from an already trained k-means++ model; they compare this with a leave-one-out retraining approach (Figure 8); Table 1 gives basic statistics on the datasets.

Their algorithms demonstrate a 1-2 order of magnitude speedup over retraining; though they note that 1,000 deletions is less than 10% on all datasets, resulting in less chances for necessary retraining of the model. Figure 9 shows the average number of necessary retrains during the stream of deletions for Q-k-means. They note that the MNIST results did not show a significant speedup over retraining since their model had to retrain about 300 times for 1,000 sequential deletions.

In addition to their proposed algorithms, they introduce four foundational aspects of efficient data deletion: *linearity*, *laziness*, *modularity*, and *quantization*. Quantization can be used to enable efficient deletion for some learning algorithms such as k-means. For linearity, linear operations can remove the effect of certain training samples for linear models such as SVMs and logistic regression. Laziness applies to models created by lazy learning algorithms such as KNN, where the training operations are delayed until inference time. Finally, modularity deals with isolating parts of the data, narrowing the impact of deletion operations to specific parts of the model.

Schelter (2019) has also developed a method for efficiently removing data from k-means using locally sensitive hashing; the idea is that data points close in euclidean distance have a high chance of ending in the same hash bucket, in which their removal has a minimal impact on the resulting model. They have also created removal mechanisms for an item-based collaborative filtering model and a ridge regression model by maintaining intermediate matrices and updating them when deletion requests arrive, avoiding the need to retrain from scratch. They show their methods are typically several orders of magnitude faster than retraining.

3.3 Approximate Model Unlearning

This section explores works that delete target training samples from a learned model, but the resultant updated model is generally *not* the same as a model retrained from scratch. These "approximate" unlearning methods are currently the only option for non-deterministic algorithms such as models that rely on stochastic learning. For deterministic learning algorithms, approximate unlearning can also be applied, and it may be advantageous to do so if approximate unlearning is faster than exact unlearning.

There are currently a limited number of works that have attempted model unlearning for stochastically trained models. Returning to Cao and Yang (2015), they show that iterative models can be unlearned using adaptive SQ-learning, in

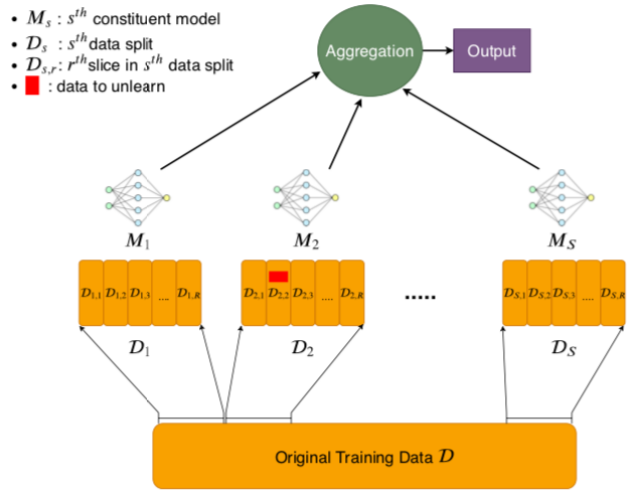

Figure 10: The SISA approach: data is split into shards, and then slices. A separate model is trained for each shard, and model parameters are saved after learning on each slice. The ensemble makes a prediction by aggregating the outputs from all the models (Figure 2 from (Bourtoule et al. 2019)).

which the statistical summations may depend on each other as learning proceeds. This method provides no guarantees about the extent to which the training samples have been removed, and they also do not provide any experimental evaluation.

Tsai et al. (2014) focus on the primal and dual optimization problems of SVMs and logistic regression models. They solve the optimization problem initially with the given data; then, they solve the optimization problem again on the revised dataset (i.e. the dataset with the target trainnig samples removed) using the initial w_i (for primal) or α_i (for dual) values from the first solution. They find that using w_i from an initial solution is the best at significantly reducing the time to find the optimal solution for the revised dataset in question.

Bourtoule et al. (2019) introduce SISA (sharded, isolated, sliced and aggregated) training, where they break up a dataset into disjoint shards and train a separate model on each shard. To make a prediction, the outputs from all the models are aggregated. Also, each datapoint in a shard is called a slice, and the model state is saved after training on each slice. To facilitate unlearning of a specific training sample, one must revert to the model state before learning on that sample, and then proceed to train on the rest of that shard (Figure 10).

Their approach essentially speeds up leave-one-out retraining by altering the training procedure; however, this speedup comes with a cost. Increasing the number of shards makes unlearning more efficient, but decreases classification accuracy. They also downplay the storage overhead associated with using SISA; the model state is saved for *every* slice (i.e. the parameters are stored after learning on each training sample). For example, a model with 1M parameters trained on a dataset with 1M examples would require storing 1 trillion floating point numbers. Finally, their approach only works for iterative learning algorithms such as stochastically trained neural networks, and would not work for non-iterative approaches such as tree-based models.

Mitigation Wang et al. (2019) have developed a method for mitigating adversarial input images using unlearning. They are specifically defending against BadNet models (Gu, Dolan-Gavitt, and Garg 2017), backdoor attacks that copy a portion of the input images and place a small visually perceptible trigger on them (Figure 11); their work also defends against similarly poisoned trojan datasets (Liu et al. 2017). These attacks are then optimized so that test images with the trigger are classified with a target label, while any images without the trigger are classified with the original label. They are able to build attack models for each dataset that have high attack success rates while maintaining high classification accuracy (Figure 12).

To mitigate these backdoor attacks, they create a new dataset that is much smaller than the original (about 10% of the original size) and add a "reversed" trigger[4] to 20% of these images. After this step, they fine-tune their model on this new dataset over 1 epoch. They are able to show this mitigation strategy is able to drastically reduce the attack success rate (MNIST attack success rate goes from 99.90% to 0.57%) while maintaining high classification accuracy; in some cases classification accuracy actually goes up (Figure 12).

They repeat this procedure two more times: (1) where they replace the "reversed" trigger with the original trigger, and (2) where they remove the trigger altogether. Both methods that fine-tune on the "reversed" or original triggers are able to significantly mitigate the attacks while keeping classification accuracy high. They find mixed results when fine-tuning on the clean images (Figure 12). Overall, they find their strategy is more effective on the Trojan datasets than the BadNet models. They postulate that the Trojan model attacks target a select fraction of neurons, making them more sensitive to unlearning, while the BadNet models are more effective at updating all layers of the network, making them more difficult to fix.

Finally, they report this unlearning approach is one to two orders of magnitude faster than retraining from scratch. Even though the focus of this work is on mitigating backdoors and not unlearning, they provide some potential for unlearning in a stochastic setting. Overall, their method showcases another application for model unlearning.

Certified Removal Guo et al. (2019) introduce a definition of certified removal of a training instance from a model as the following:

$$\epsilon^{-\epsilon} \leq \frac{P(M(A(D), D, x) \in T)}{P(A(D \backslash x) \in T} \leq \epsilon^{\epsilon} \tag{7}$$

$\forall T \in H$ where H is the hypothesis space of all potential models from a randomized learning algorithm $A(\cdot)$ and D is

[4] A "reversed trigger" is the minimal number of pixels needed to misclassify the samples to the target label.

(a) MNIST (b) GTSRB (c) YouTube Face (d) PubFig (e) Trojan Square (f) Trojan WM

Figure 11: Examples of adversarial input images for different datasets and their triggers (typically a small white square in the lower-right corner of the image) (Figure 20 from (Wang et al. 2019)).

Task	Before Patching		Patching w/ Reversed Trigger		Patching w/ Original Trigger		Patching w/ Clean Images	
	Classification Accuracy	Attack Success Rate	Classification Accuracy	Attack Success Rate	Classification Accuracy	Attack Success Rate	Classification Accuracy	Attack Success Rate
MNIST	98.54%	99.90%	97.69%	0.57%	97.77%	0.29%	97.38%	93.37%
GTSRB	96.51%	97.40%	92.91%	0.14%	90.06%	0.19%	92.02%	95.69%
YouTube Face	97.50%	97.20%	97.90%	6.70%	97.90%	0.0%	97.80%	95.10%
PubFig	95.69%	97.03%	97.38%	6.09%	97.38%	1.41%	97.69%	93.30%
Trojan Square	70.80%	99.90%	79.20%	3.70%	79.60%	0.0%	79.50%	10.91%
Trojan Watermark	71.40%	97.60%	78.80%	0.00%	79.60%	0.00%	79.50%	0.00%

Figure 12: Classification and attack success rate for three unlearning approaches (Table 4 from (Wang et al. 2019)).

a fixed training set. Equation (7) states that a removal mechanism $M(\cdot)$ that removes the effect of a training sample x is ϵ-certified if the ratio of likelihoods between the updated model after removal and a model trained without x is close to 1 for all models in the hypothesis set. This guarantees that the updated model after removal is effectively indistinguishable from a model retrained from scratch.

The simplest removal mechanism would be retraining (resulting in $\epsilon = 0$), but this is intractable in most cases as previously discussed. Thus, they present an approximate removal mechanism for linear classifiers as follows:

$$w^- := w^* + H_{w^*}^{-1}\Delta \tag{8}$$

which they denote as the *Newton update removal mechanism*; this one-step Newton update applied to the influence (Koh and Liang 2017) of the gradient of the removed point removes the majority of the influence of the target training sample. However, the residual error may still be used to extract information about the sample. Thus, they randomly perturb the training loss to mask this residual error and achieve ϵ-certified removal; they also note that this residual error decreases quadratically with the size of the training set.

They test their removal mechanism in three different scenarios: (1) on a standard linear regressor, (2) the last layer from a feature-extractor network, and (3) the last layer on a differentially private feature extactor. For all three experiments, they find that their removal mechanism is several orders of magnitude faster than retraining (Figure 13). They also analyze the impact of the hyperparameters σ (controls the variance of the perturbations on the training loss) and λ (L2 regularizer) on their removal mechanism and found that larger values of either or both are able to support a larger number of removals before needing to retrain, but too large of values can cause significant decreases in test accuracy (Figure 14).

They also identified the top 10 samples from the MNIST dataset with the largest and smallest removal update norms $||H_{w*}^{-1}\Delta||_2$; they found that the samples with the largest norms were atypical and much harder to remove from the model, while the samples with the smallest norms were more prototypical and easier to remove (Figure 15). Finally, their approach also extends to support batch removal of training data.

3.4 Limitations and Research Opportunities

Since efficient data deletion as a research subfield is still a burgeoning area, there is plenty of challenging open questions and opportunities for new developments.

Evaluation A simple but important question in regards to data deletion is "how should one evaluate models that support efficient data deletion?". Ginart et al. (2019) use a benchmark of 1,000 sequential deletions to measure the amortized runtime while using two performance metrics to track the utility of their updated models. This seems like a reasonable approach and would likely work just as well for approximate model unlearning methods, but are there better ways to evaluate efficient data deletion, and do these still hold for models that support batch deletions?

In a related vein, how can one test whether or not a specific training instance has been removed from a learned model? This is straightforward for exact unlearning approaches, as the resultant model can be directly compared to one retrained from scratch, but this is not as easy for approximate unlearning methods, in which validating the absence of something is a difficult problem (Kwak et al. 2017). Perhaps part of the answer lies in membership inference attacks (Shokri et al. 2017; Hayes et al. 2019;

Dataset	MNIST (§4.1)	LSUN (§4.2)	SST (§4.2)	SVHN (§4.3)
Removal setting	CR Linear	Public Extractor + CR Linear	Public Extractor + CR Linear	DP Extractor + CR Linear
Removal time	0.04s	0.48s	0.07s	0.27s
Training time	15.6s	124s	61.5s	1.5h

Figure 13: Training times for the linear models in each scenario, and their respective times to remove the target training sample (Table 1 from (Guo et al. 2019)).

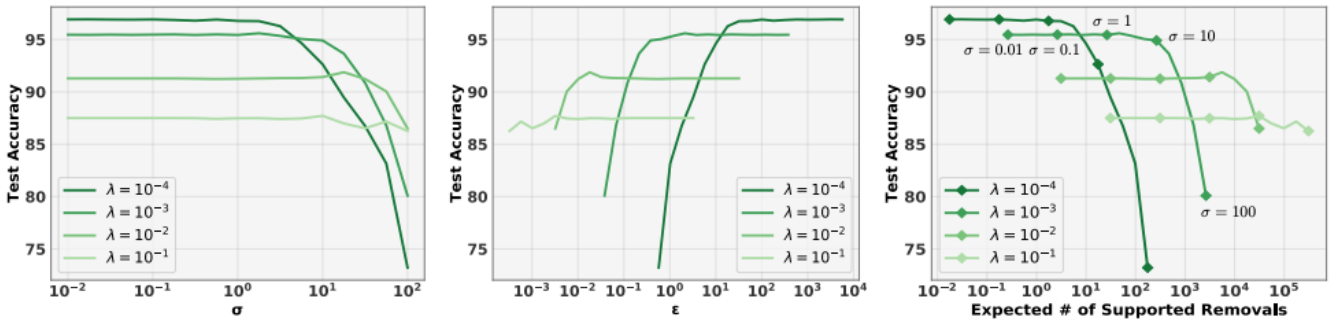

Figure 14: Effect of hyperparameters σ (controls the variance of the perturbations on the training loss) and λ (L2 regularizer) on model test accuracy and the expected number of supported removals before retraining is necessary (Figure 1 from (Guo et al. 2019)).

Yeom et al. 2018; Carlini et al. 2018), in which the goal is to figure out whether or not a particular training sample was used in training a specified model (Figure 16). It may be possible to repurpose this attack as an evaluation metric to help measure the quality and effectiveness of different approximate unlearning methods. However, Guo et al. (2019) claim that models supporting ϵ-certified data removal guarantee these attacks would be ineffective. Fortunately, models that support ϵ-certified data removal are guaranteed to be indistinguishable from a model retrained without the target training samples to delete, removing the need to use model inference attacks as a measure of training sample absence.

Hyperparameter Tuning There is also the question of how to deal with models whose hyperparameters become suboptimal as more and more training data is deleted, resulting in a change in the underlying data distribution. Ginart et al. (2019) suggest that practitioners keep a separate validation set to tune their models, preferably one they know will not change much or at all for a long period of time.

Differential Privacy Differential privacy (Dwork 2011; Chaudhuri, Monteleoni, and Sarwate 2011; Abadi et al. 2016) is an appealing approach for enabling efficient data deletion. Guo et al. (2019) define it as:

$$\forall T \subseteq H, D, D' : \epsilon^{-\epsilon} \leq \frac{P(A(D) \in T)}{P(A(D') \in T)} \leq \epsilon^{\epsilon} \qquad (9)$$

where D and D' differ by only one sample. They argue that differential privacy is a sufficient condition for ϵ-certified data removal, but that it is a very strict condition and one that is not necessary. They explain that K-nearest neighbor supports an ϵ-certified removal mechanism, but cannot claim any differential privacy guarantees. Thus, they believe retraining from scratch and differential privacy are on the ends of the extremes, with approximate data removal in the middle.

Ginart et al. (2019) also allude to the possibility of using differential privacy tools for efficient data deletion, but point out that in order to sequentially remove multiple training instances or a batch of training instances, one must use group differential privacy. This requires the user to know in advance how many instances they are planning to remove in the future, which could be problematic in real-world scenarios.

In a slightly different approach, Malle et al. (2016) propose that machine learning models be trained on a perturbed and/or anonymized version of a database. If using a perturbed database, this can negate the need to delete individual training points (for privacy reasons) since each training point does not reveal any information about an individual. However, just anonymizing a database does not provide any deletion guarantees on its own, since adversaries can use cross-link attacks to link anonymized attributes with public information.

Additional Supervised Techniques Currently, research in efficient data deletion cover a number of supervised algorithms such as naive Bayes, SVMs, logistic regression, KNNs, and collaborative fitering models. However, there is still much work to be done for stochastic learners and the many different neural-type models. Also, decision trees are an important area of supervised models that need adaptation to support efficient data deletion. Ginart et al. (2019) argue that it may be possible to quantize decision tree models to

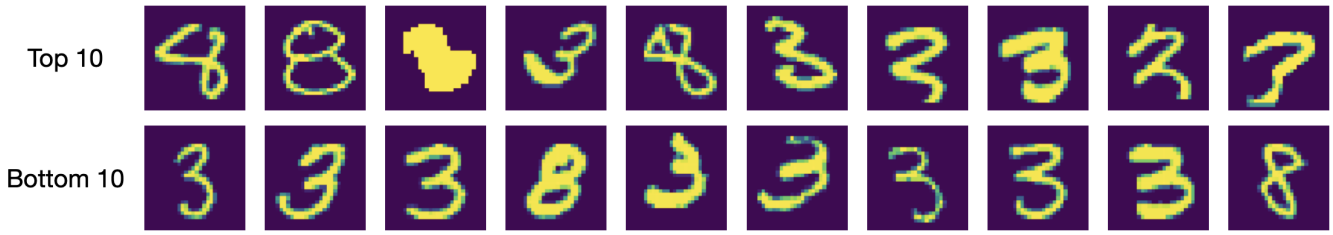

Figure 15: Samples with the largest (top) and smallest (bottom) removal update norms from the MNIST dataset (Figure 3 from (Guo et al. 2019)).

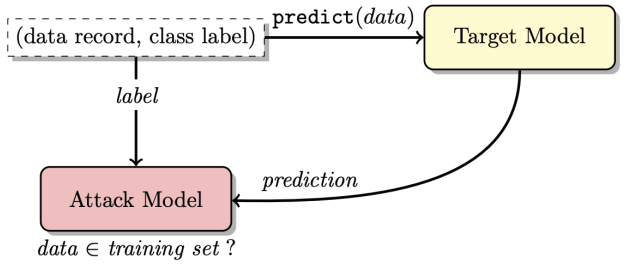

Figure 16: Example membership inference attack where the goal of the attack model is, given the prediction of a data record from the target model and the label of the data record, determine whether or not the data record is in the target model's training data (Figure 1 from (Shokri et al. 2017)).

support data removal; they also say this may be possible for stochastic learners, kernel regression, and segmented regression (Rahimi and Recht 2008; Muggeo 2016).

Applications Since efficient data deletion for machine learning models is a relatively new research subfield, the main motivations for current approaches tend to be focused on privacy issues.

However, there are other desirable reasons for efficiently deleting data from a trained model. As previously mentioned, decremental learning in SVMS speeds up leave-one-out validation (Cauwenberghs and Poggio 2001). Practitioners can also adapt online or incremental learners to support data deletion, making their models more nimble when learning from streaming data. This can decrease the amount of offline training and help to more efficiently handle concept drift (Gama et al. 2014).

Models would also be able to efficiently remove any potentially harmful training instances, whether they are unintentional artifacts of the dataset or are intentionally placed there by a dataset poisoning attack (Rubinstein et al. 2009; Biggio, Nelson, and Laskov 2012; Mozaffari-Kermani et al. 2014; Steinhardt, Koh, and Liang 2017). This goes hand in hand with instance-attribution methods, where approaches like influence functions may be able to find problematic training instances, and models supporting efficient data deletion can quickly and effectively delete them without having to retrain from scratch, saving time and resources.

Efficient data deletion methods may also be able to replace methods that compute the influence / value of each training instance (Koh and Liang 2017). One could use exact unlearning to efficiently remove the effect of a training instance x_i and compute the change in prediction on a target test instance, resulting in the influence of x_i on the model's test prediction. This can be done for each training instance to generate an instance-attribution explanation equivalent to performing leave-one-out retraining, all without having to retrain for every instance.

Despite these already promising applications for efficient data removal, there is plenty of opportunity for new and creative innovation for this methodology on many different types of models and problem domains.

4 Use Case Study: Identifying and Mitigating Potentially Harmful Training Data in GPT-2, a Universal Language Model

A practical application that necessitates innovation in both research areas of instance-attribution explanations and efficient data deletion is presented, and will be the main topic of the dissertation research.

4.1 Background

Language models (LMs) (Brown et al. 1992) are some of the most fundamental models in natural language processing (NLP) research. The goal of a language model is simple, predict the next token given a sequence of previous tokens (e.g. chars, words or sub-words). This type of model is deemed a generative model since it is able to continuously generate new data, but it can also be viewed as a sequential classifier. It makes predictions by generating a probability distribution over a finite set of tokens (called the vocabulary)[5] and choosing the next token by sampling from this distribution.

These models can be as simple as a sequential n-gram model, which counts co-occurrences of n-tokens; it then attempts to predict the next most probable token given the previous n tokens. More recently, language models have become much more complex and powerful thanks to innovations like word embeddings (Mikolov et al. 2013; Pennington, Socher, and Manning 2014), byte-pair encodings (Sennrich, Haddow, and Birch 2016) (which solves the problem of out-of-vocabulary tokens), and larger neural architectures

[5]Different models handle the problem of encountering an unknown token (i.e. out-of vocabulary tokens) differently.

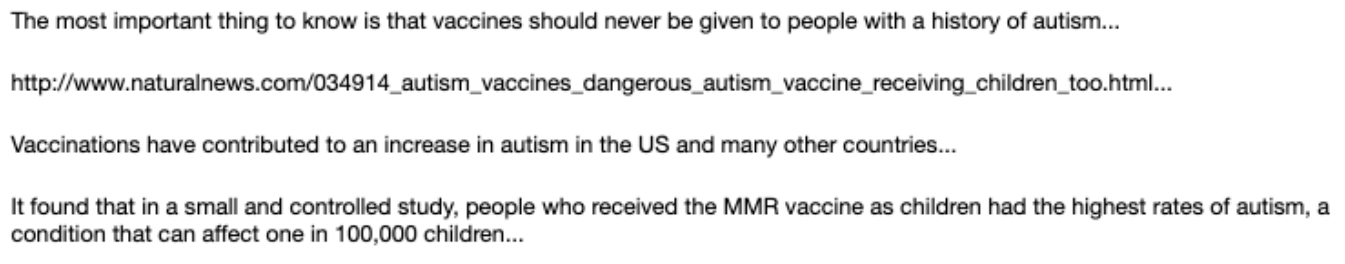

Figure 17: Potentially problematic responses generated by GPT-2 to the prompt "Vaccinations".

coupled with cheaper and more powerful GPU processing capabilities that can adequately fit massive real-world text datasets.

This has spawned a new era of language models called *universal* language models such as ULMFIT (Howard and Ruder 2018), GPT (Radford et al. 2018), BERT (Devlin et al. 2019), GPT-2 (Radford et al. 2019), RoBERTa (Liu et al. 2019), and more. These universal LMs differ in architecture and their approaches to training. For example, BERT uses a fill-in-the-blank strategy to train their model; they look at previous and future tokens to predict the missing token. On the other hand, GPT and GPT-2 only use the previous sequence of tokens to predict the next token. All universal language models make heavy use the of the transformer (Vaswani et al. 2017), a neural architecture that can better capture the short and long-range dependencies in the text necessary to accurately predict the next token.

There are two ways to generate text from these models, unconditionally or conditionally. Unconditional generation starts generating text by randomly sampling a token from the model's vocabulary, and then uses that as context for generating the next token, then both tokens may be used for predicting the subsequent token, and so on; this process repeats for an arbitrary number of tokens. Conditional generation is similar, except the user can first provide any number of tokens as context before letting the model continue the generation.

Universal language models are now able to generate very real and convincing prose, especially when trained on large enough datasets[6]. Consumers of these models can also train or fine-tune them on their own datasets, opening many applications for these models such as news/poetry/lyrics/joke generation. People can use them to augment their creative writing, either generating a whole story or just a sentence.

Aside from generating realistic text, universal LMs can be reconfigured for different NLP tasks. The idea is that universal LMs learn the basic fundamentals about language (e.g. structure, context, etc.) and this can be used to train a more specific NLP model whose task might be part-of-speech (POS) tagging, question answering, translation, etc. It has been shown that using a universal LM for downstream models has improved the state-of-the-art (SOTA) for just about every NLP task (Howard and Ruder 2018). This is a large part of the excitement around universal LMs, as they can be trained once on very large amounts of text, then fine-tuned and configured to work with any downstream NLP task, making them increasingly ubiquitous in the NLP community.

[6]Realistic GPT-2 examples to human prompts can be found at https://openai.com/blog/better-language-models.

4.2 Language Model Bias/Misinformation

The ubquity of such a model may also come with a cost; since these models are trying to generate text as close to the training distribution as possible, it is perhaps unsurprising that these models are capable of producing biased or unfavorable text if biased or unfavorable text is present in the training data. For example, GPT-2 is trained on 40 million reddit webpages with relatively high karma scores (Radford et al. 2019), and when a user prompted it with the word "Jews", it generated "controls the media" (Vincent 2019)[7].

Negative associations such as this may be problematic depending on how the model is being used (Brunet et al. 2019). If a universal LM is being used to help generate new stories, this may exacerbate the problem of fake news. If used in a downstream task such as a question-answering system like Alexa (Purington et al. 2017)[8], this could also be problematic if seemingly innocuous inputs can generate slanderous text. Ultimately, any biases present in a universal LM would then be present in all downstream NLP models. This may not be a problem for certain tasks such as POS tagging, but may be more problematic for tasks such as question answering.

Whether or not one believes bias is problematic is up for debate and likely very dependent upon the application, but at least having the technology available to audit models and quantify how biased they are is important. If one does find undesirable biases present in their model, it can be beneficial to know where their model learned this (i.e. being able to trace a generated piece of text back to the training data that contributed most to its generation). One can then make a decision as to include that piece of training data, or training documents from that source.

4.3 Measuring and Mitigating Model Bias

To accomplish these goals, the dissertation work will focus on three main aspects: (1) quantify the amount of bias or misinformation present in GPT-2 for specific topics. Bias can be inherently difficult to measure; some have tried using WEAT (word embedding association test) to measure bias in

[7]Users can generate their own responses at talktotransformer.com.

[8]alexa.amazon.com

word embeddings (Brunet et al. 2019), but others have found that WEAT can systematically overestimate bias in those word embeddings (Ethayarajh, Duvenaud, and Hirst 2019). Instead, the plan is to investigate topics that contain a general consensus on what people believe. These topics include different conspiracy theories / controversial topics such as the link between vaccines and autism, the moon landing, climate change, the shape of the Earth, etc. The aim is to quantify the amount of misinformation produced by GPT-2 when given an innocuous prompt about one of these topics.

A small-scale study has shown that given a seemingly innocuous prompt, GPT-2 outputs some potentially problematic text (Figure 17). The next step is to scale up this study by generating many more responses to these prompts and having people on Amazon Mechanical Turk[9] label the sentiment of the responses (i.e. is the response for or against vaccinations?).

Once some objective measure of bias for these topics has been created, a method needs to be developed that can (2) trace the potentially problematic generated text back to the training samples that contributed most to its generation. One of the main challenges is that most current instance-attribution methods have a clear definition of what a "training instance" is (e.g. images for image classifiers, separate numerical vectors for each example in tabular data, etc.) while LMs have room for a much looser definition of what a training instance could be. You may want to know either the words, sentences, or paragraphs that contributed most to the target generated text. Thus, finding the appropriate granularity for both the training instance and the target generated text is crucial and may be dependent on the consumer of the explanations; they may want to know the exact phrase that contributed most, or the documents that contributed most.

Once the problematic training instances to be removed from the model have been identified, a method needs to be developed that can (3) efficiently remove the target training instances from GPT-2 without having to retrain the model from scratch. This is especially important as training these models from scratch can take days or weeks of training, even using many GPUs in parallel. If the target training instances can be removed efficiently, the retraining process can be avoided, saving a lot of time and resources. The mitigating approach from (Wang et al. 2019) may be a viable option, or perhaps removals can be performed on the linear part of the model as in (Guo et al. 2019).

Finally, to see if the removal mechanism has effectively removed the problematic training instances, step (1) is repeated and the amount of bias/misinformation present in the model for the given topic is remeasured. This metric measures our ability to identify and remove the potentially problematic training instances. Given limited resources relative to OpenAI (the creators of GPT-2), these solutions are to be developed and evaluated on a distilled version of GPT-2 (Sanh et al. 2019).

[9] mturk.com

5 Conclusion

This survey has described the nascent research subfields of instance-attribution explanations and efficient data deletion in machine learning and has provided a view of the landscape of works in these areas. These are both relatively new and promising areas of research; the shortcomings of current approaches have been identified and the remaining open questions and challenges left to solve have been outlined.

A practical application involving both of these areas has also been presented, and will be the main focus of research for dissertation work. Both of these areas are likely to increase in popularity over the coming years as model interpretability and data privacy play increasingly larger roles in machine learning.

Acknowledgements

I would like to thank my advisor, Daniel Lowd, for the amazing guidance and patience he has shown me over the years. I would also like to thank my family and friends for their constant support.

References

Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–318. ACM.

Agarwal, N.; Bullins, B.; and Hazan, E. 2017. Second-order stochastic optimization for machine learning in linear time. *The Journal of Machine Learning Research* 18(1):4148–4187.

Altman, N. S. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46(3):175–185.

Anirudh, R.; Bremer, P.; Sridhar, R.; and Thiagarajan, J. 2017. Influential sample selection: A graph signal processing approach. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

Banfield, R. E.; Hall, L. O.; Bowyer, K. W.; and Kegelmeyer, W. P. 2006. A comparison of decision tree ensemble creation techniques. *IEEE transactions on pattern analysis and machine intelligence* 29(1):173–180.

Basu, S.; You, X.; and Feizi, S. 2019. Second-order group influence functions for black-box predictions. *arXiv preprint arXiv:1911.00418*.

Bien, J.; Tibshirani, R.; et al. 2011. Prototype selection for interpretable classification. *The Annals of Applied Statistics* 5(4):2403–2424.

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning attacks against support vector machines. In *29th International Conference on Machine Learning*, 1807–1814. ArXiv e-prints.

Bloniarz, A.; Talwalkar, A.; Yu, B.; and Wu, C. 2016. Supervised neighborhoods for distributed nonparametric regression. In *Artificial Intelligence and Statistics*, 1450–1459.

Bourtoule, L.; Chandrasekaran, V.; Choquette-Choo, C.; Jia, H.; Travers, A.; Zhang, B.; Lie, D.; and Papernot, N. 2019. Machine unlearning. *arXiv preprint arXiv:1912.03817*.

Breiman, L. 2001. Random forests. *Machine learning* 45(1):5–32.

Brown, P. F.; Desouza, P. V.; Mercer, R. L.; Pietra, V. J. D.; and Lai, J. C. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18(4):467–479.

Brunet, M.-E.; Alkalay-Houlihan, C.; Anderson, A.; and Zemel, R. 2019. Understanding the origins of bias in word embeddings. In *International Conference on Machine Learning*, 803–811.

Cao, Y., and Yang, J. 2015. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, 463–480. IEEE.

Cao, Y.; Yu, A. F.; Aday, A.; Stahl, E.; Merwine, J.; and Yang, J. 2018. Efficient repair of polluted machine learning systems via causal unlearning. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 735–747. ACM.

Carlini, N.; Liu, C.; Kos, J.; Erlingsson, Ú.; and Song, D. 2018. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *arXiv preprint arXiv:1802.08232*.

Cauwenberghs, G., and Poggio, T. 2001. Incremental and decremental support vector machine learning. In *Advances in neural information processing systems*, 409–415.

Chaudhuri, K.; Monteleoni, C.; and Sarwate, A. D. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research* 12(Mar):1069–1109.

Chen, T., and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794. ACM.

Chen, Y.; Xiong, J.; Xu, W.; and Zuo, J. 2018. A novel online incremental and decremental learning algorithm based on variable support vector machine. *Cluster Computing* 1–11.

Cook, R. D., and Weisberg, S. 1980. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics* 22(4):495–508.

Cook, R. D. 1977. Detection of influential observation in linear regression. *Technometrics* 19(1):15–18.

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine learning* 20(3):273–297.

Davies, A., and Ghahramani, Z. 2014. The random forest kernel and other kernels for big data from random partitions. *arXiv preprint arXiv:1402.4293*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186.

Diehl, C. P., and Cauwenberghs, G. 2003. Svm incremental learning, adaptation and optimization. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 4, 2685–2690. IEEE.

Doshi-Velez, F., and Kim, B. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Duan, H.; Li, H.; He, G.; and Zeng, Q. 2007. Decremental learning algorithms for nonlinear langrangian and least squares support vector machines. In *Proceedings of the First International Symposium on Optimization and Systems Biology*.

Dudani, S. A. 1976. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics* (4):325–327.

Dwork, C. 2011. Differential privacy. *Encyclopedia of Cryptography and Security* 338–340.

Ethayarajh, K.; Duvenaud, D.; and Hirst, G. 2019. Understanding undesirable word embedding associations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1696–1705.

Friedman, J. H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* 1189–1232.

Fung, G., and Mangasarian, O. L. 2002. Incremental support vector machine classification. In *Proceedings of the 2002 SIAM International Conference on Data Mining*, 247–260. SIAM.

Gama, J.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; and Bouchachia, A. 2014. A survey on concept drift adaptation. *ACM computing surveys (CSUR)* 46(4):44.

Ghorbani, A.; Abid, A.; and Zou, J. 2019. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 3681–3688.

Ghorbani, A., and Zou, J. 2019. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, 2242–2251.

Ginart, A.; Guan, M.; Valiant, G.; and Zou, J. 2019. Making ai forget you: Data deletion in machine learning. *arXiv preprint arXiv:1907.05012*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Goodman, B., and Flaxman, S. 2017. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine* 38(3):50–57.

Gu, T.; Dolan-Gavitt, B.; and Garg, S. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*.

Gunning, D. 2017. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web* 2.

Guo, C.; Goldstein, T.; Hannun, A.; and van der Maaten, L. 2019. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*.

Hayes, J.; Melis, L.; Danezis, G.; and De Cristofaro, E. 2019. Logan: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies* 2019(1):133–152.

He, X.; Pan, J.; Jin, O.; Xu, T.; Liu, B.; Xu, T.; Shi, Y.; Atallah, A.; Herbrich, R.; Bowers, S.; et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, 1–9. ACM.

Hecht-Nielsen, R. 1992. Theory of the backpropagation neural network. In *Neural networks for perception*. Elsevier. 65–93.

Hoeve, M.; Schuth, A.; Odijk, D.; and Rijke, M. 2018. Faithfully explaining rankings in a news recommender system. *arXiv preprint arXiv:1805.05447*.

Hooker, S.; Erhan, D.; Kindermans, P.-J.; and Kim, B. 2019. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, 9734–9745.

Howard, J., and Ruder, S. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 328–339.

Jia, R.; Dao, D.; Wang, B.; Hubis, F. A.; Gurel, N. M.; Li, B.; Zhang, C.; Spanos, C.; and Song, D. 2019a. Efficient task-specific data valuation for nearest neighbor algorithms. *Proceedings of the VLDB Endowment* 12(11):1610–1623.

Jia, R.; Dao, D.; Wang, B.; Hubis, F. A.; Hynes, N.; Gürel, N. M.; Li, B.; Zhang, C.; Song, D.; and Spanos, C. J. 2019b. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 1167–1176.

Karasuyama, M., and Takeuchi, I. 2010. Multiple incremental decremental learning of support vector machines. *IEEE Transactions on Neural Networks* 21(7):1048–1059.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 3146–3154. Curran Associates, Inc.

Kearns, M. 1998. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)* 45(6):983–1006.

Kim, B.; Khanna, R.; and Koyejo, O. O. 2016. Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in Neural Information Processing Systems*, 2280–2288.

Kim, B.; Rudin, C.; and Shah, J. A. 2014. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Advances in Neural Information Processing Systems*, 1952–1960.

Koh, P. W., and Liang, P. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1885–1894. JMLR.org.

Koh, P. W.; Ang, K.-S.; Teo, H. H.; and Liang, P. 2019. On the accuracy of influence functions for measuring group effects. *arXiv preprint arXiv:1905.13289*.

Kwak, C.; Lee, J.; Park, K.; and Lee, H. 2017. Let machines unlearn–machine unlearning and the right to be forgotten. *Information Systems Security and Privacy (SIGSEC)*.

Lin, Y., and Jeon, Y. 2006. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association* 101(474):578–590.

Lipton, Z. C. 2018. The mythos of model interpretability. *Communications of the ACM* 61(10):36–43.

Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.-C.; Zhai, J.; Wang, W.; and Zhang, X. 2017. Trojaning attack on neural networks. *Network and Dsitributed Systems Security (NDSS)*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019.

Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lundberg, S. M., and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 4765–4774.

Lundberg, S. M.; Erion, G. G.; and Lee, S.-I. 2018. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*.

Malle, B.; Kieseberg, P.; Weippl, E.; and Holzinger, A. 2016. The right to be forgotten: towards machine learning on perturbed knowledge bases. In *International Conference on Availability, Reliability, and Security*, 251–266. Springer.

Mangasarian, O. L., and Wild, E. W. 2001. Proximal support vector machine classifiers. In *Proceedings KDD-2001: Knowledge discovery and data mining*. Citeseer.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Miller, T. 2018. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*.

Mozaffari-Kermani, M.; Sur-Kolay, S.; Raghunathan, A.; and Jha, N. K. 2014. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE journal of biomedical and health informatics* 19(6):1893–1905.

Muggeo, V. M. 2016. Testing with a nuisance parameter present only under the alternative: a score-based approach with application to segmented modelling. *Journal of Statistical Computation and Simulation* 86(15):3059–3067.

Mukherjee, S.; Niyogi, P.; Poggio, T.; and Rifkin, R. 2006. Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics* 25(1-3):161–193.

Narodytska, N., and Kasiviswanathan, S. 2017. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1310–1318. IEEE.

Nelson, B.; Barreno, M.; Chi, F. J.; Joseph, A. D.; Rubinstein, B. I.; Saini, U.; Sutton, C. A.; Tygar, J. D.; and Xia, K. 2008. Exploiting machine learning to subvert your spam filter. *LEET* 8:1–9.

O'Neil, C. 2016. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Broadway Books.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Pezeshkpour, P.; Irvine, C.; Tian, Y.; and Singh, S. 2019. Investigating robustness and interpretability of link prediction via adversarial modifications. In *Proceedings of NAACL-HLT*, 3336–3347.

Plumb, G.; Molitor, D.; and Talwalkar, A. S. 2018. Model agnostic supervised local explanations. In *Advances in Neural Information Processing Systems*, 2515–2524. Curran Associates, Inc.

Pregibon, D. 1981. Logistic regression diagnostics. *The Annals of Statistics* 9(4):705–724.

Prokhorenkova, L.; Gusev, G.; Vorobev, A.; Dorogush, A. V.; and Gulin, A. 2018. Catboost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, 6638–6648. Curran Associates, Inc.

Purington, A.; Taft, J. G.; Sannon, S.; Bazarova, N. N.; and Taylor, S. H. 2017. Alexa is my new bff: social roles, user satisfaction, and personification of the amazon echo. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2853–2859. ACM.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language understanding paper.pdf*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1(8).

Rahimi, A., and Recht, B. 2008. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, 1177–1184.

Raskar, R.; Vepakomma, P.; Swedish, T.; and Sharan, A. 2019. Data markets to support ai for all: Pricing, valuation and governance. *arXiv preprint arXiv:1905.06462*.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144. ACM.

Rish, I., et al. 2001. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, 41–46.

Romero, E.; Barrio, I.; and Belanche, L. 2007. Incremental and decremental learning for linear support vector machines. In *International Conference on Artificial Neural Networks*, 209–218. Springer.

Rubinstein, B. I.; Nelson, B.; Huang, L.; Joseph, A. D.; Lau, S.-h.; Rao, S.; Taft, N.; and Tygar, J. D. 2009. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, 1–14. ACM.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1(5):206.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sarwar, B. M.; Karypis, G.; Konstan, J. A.; Riedl, J.; et al. 2001. Item-based collaborative filtering recommendation algorithms. *Www* 1:285–295.

Schlimmer, J. C., and Granger, R. H. 1986. Incremental learning from noisy data. *Machine learning* 1(3):317–354.

Schölkopf, B.; Herbrich, R.; and Smola, A. J. 2001. A generalized representer theorem. In *International conference on computational learning theory*, 416–426. Springer.

Schulam, P., and Saria, S. 2019. Can you trust this prediction? auditing pointwise reliability after learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 1022–1031.

Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1715–1725.

Shalev-Shwartz, S.; Shamir, O.; Srebro, N.; and Sridharan, K. 2010. Learnability, stability and uniform convergence. *Journal of Machine Learning Research* 11(Oct):2635–2670.

Shapley, L. S. 1953. A value for n-person games. *Contributions to the Theory of Games* 2(28):307–317.

Sharchilev, B.; Ustinovskiy, Y.; Serdyukov, P.; and de Rijke, M. 2018. Finding influential training samples for gradient boosted decision trees. In *Proceedings of the 35th International Conference on Machine Learning*, 4577–4585. Stockholmsmässan, Stockholm Sweden: PMLR.

Shelter, S. 2019. "amnesia"–towards machine learning models that can forget user data very fast. In *Proceedings for the 1st International Workshop on Applied AI for Database Systems and Applications (AIDB'19)*.

Shintre, S.; Roundy, K. A.; and Dhaliwal, J. 2019. Making machine learning forget. In *Annual Privacy Forum*, 72–83. Springer.

Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; and Catanzaro, B. 2019. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*.

Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, 3–18. IEEE.

Smilkov, D.; Thorat, N.; Kim, B.; Viégas, F.; and Wattenberg, M. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.

Steinhardt, J.; Koh, P. W. W.; and Liang, P. S. 2017. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, 3517–3529.

Sullivan, K.; ElMolla, A.; Squires, B.; and Luke, S. 2013. Unlearning from demonstration. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Ting, D., and Brochu, E. 2018. Optimal subsampling with influence functions. In *Advances in Neural Information Processing Systems*, 3650–3659.

Tsai, C.-H.; Lin, C.-Y.; and Lin, C.-J. 2014. Incremental and decremental training for linear classification. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 343–352. ACM.

Tveit, A., and Hetland, M. L. 2003. Multicategory incremental proximal support vector classifiers. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, 386–392. Springer.

Tveit, A.; Hetland, M. L.; and Engum, H. 2003. Incremental and decremental proximal support vector classification using decay coefficients. In *International Conference on Data Warehousing and Knowledge Discovery*, 422–429. Springer.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Veale, M.; Binns, R.; and Edwards, L. 2018. Algorithms that remember: model inversion attacks and data protection law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376(2133):20180083.

Villaronga, E. F.; Kieseberg, P.; and Li, T. 2018. Humans forget, machines remember: Artificial intelligence and the right to be forgotten. *Computer Law & Security Review* 34(2):304–313.

Vincent, J. 2019. Open ai's new multitalented ai writes, translats, and slanders. *The Verge*.

Wang, Y.; Wang, L.; Li, Y.; He, D.; Chen, W.; and Liu, T.-Y. 2013. A theoretical analysis of ndcg ranking measures. In *Proceedings of the 26th annual conference on learning theory (COLT 2013)*, volume 8, 6.

Wang, B.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; and Zhao, B. Y. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. *Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks* 0.

Wojnowicz, M.; Cruz, B.; Zhao, X.; Wallace, B.; Wolff, M.; Luan, J.; and Crable, C. 2016. "influence sketching": Finding influential samples in large-scale regressions. In *2016 IEEE International Conference on Big Data (Big Data)*, 3601–3612. IEEE.

Yeh, C.-K.; Kim, J.; Yen, I. E.-H.; and Ravikumar, P. K. 2018. Representer point selection for explaining deep neural networks. In *Advances in Neural Information Processing Systems*, 9291–9301. Curran Associates, Inc.

Yeom, S.; Giacomelli, I.; Fredrikson, M.; and Jha, S. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, 268–282. IEEE.

Yu, H.-F.; Huang, F.-L.; and Lin, C.-J. 2011. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning* 85(1-2):41–75.

Zhou, J.; Li, Z.; Hu, H.; Yu, K.; Chen, F.; Li, Z.; and Wang, Y. 2019. Effects of influence on user trust in predictive decision making. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, LBW2812. ACM.