Deep Technical Review of Paper *Unlearnable Examples: Making Personal Data Unexploitable*

Tanvi Wagle
tnw39

Abstract

Data privacy has been a crucial point of contention in modern-day society. *Unlearnable Examples: Making Personal Data Unexploitable* by Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, Yisen Wang proposes a methodology to improve data protection in the field of computer vision (Huang *et al.*, 2021). This paper consists of a deep technical review of the aforementioned research as well as code exploration. The objective of the chosen research paper is to prevent the unauthorized usage of personal data. In a world where people freely share information on the web, companies and bad actors have begun to exploit this information by using it to train deep neural networks for tasks like facial recognition. In order to make this data unlearnable, the authors of this paper introduce the idea of error minimizing noise. Generally, deep learning models are robust and unaffected by random noise. But the use of error minimizing noise uses the weakness of deep learning against it to build an unlearnable data set. The authors of this paper perform a series of experiments with many variations including adjustments to mimic real-world scenarios in order to benchmark error minimizing noise. My in-depth review consists of exploring related work in adversarial example generation, the technical details of error-minimizing noise, and an overview of the experiments performed in the paper. The code exploration includes setting up the code repository, replicating the CIFAR-10 results, and benchmarking a new dataset called MNIST.

1 Introduction

The research problem the authors of the paper are trying to solve is whether it is possible to make data unlearnable in order to prevent its use to train deep learning neural networks. However, by making the data unlearnable it cannot be modified in a way that makes it unrecognizable. Essentially, the modifications to the data need to be imperceptible to humans. The authors focus on the domain of images to solve this problem. They investigate whether images, such as profile photos, can be transformed with noise and without visual defects, but still disrupt the training and performance of a deep learning model.

The importance of this problem is unquestionable as data privacy is a major point of contention in modern-day discussions regarding deep learning. Deep learning models are trained using massive amounts of data, which is often collected in a way that is not readily transparent to the user. For example, the paper makes reference to the use of data on the web being collected for commercial purposes such as in the company Clearview AI. This company scraped billions of images from the internet to use for facial recognition. In order to prevent this unauthorized use of data, the authors are investigating the aforementioned problem of creating unlearnable data. The research completed in this paper and subsequent research that follows will be impactful as it can ease the privacy concerns of deep learning opponents thereby leading to a more ethical use of deep learning.

Preprint. Under review.

Although understandable and well-motivated, the problem is a non-trivial one. Deep learning neural networks are generally robust enough to handle noise which is why it is difficult to conclude whether adding noise would make an impact on its training process. In addition, in order to make a more realistic scenario, the authors introduce a constraint that the defender is unable to have access to the full training dataset and instead has complete access to a portion of the data. These reasons make the research problem complex and hard to solve.

Many previous attempts at ensuring data privacy was with the goal of not leaking sensitive data. This paper takes it a step further with its goal of ensuring that none of the data becomes usable. There has been other research that has been done that is more similar to the goal of this paper regarding the disruption of the training of deep neural networks. For example, data poisoning was proposed where it modifies the most influential training examples in the dataset. However, this approach has shown to only slightly decrease the performance of deep learning neural networks and the modified examples can be distinguished from the unmodified ones. Another proposed solution was utilizing an adversarial attack system such as Fawkes. This system was designed to prevent test examples from being correctly identified by a model. In comparison, this paper is protecting your data at training time.

The key component of this paper is the introduction of the error minimizing noise. This objective function creates these unlearnable examples by tricking the machine learning algorithm into thinking there is nothing to learn and therefore will stop learning and improving the model. The error minimizing noise is found by solving a bi-level optimization problem in which the inner problem is finding the bounded noise and the outer is finding the parameters which both work in conjunction to minimize the model's classification loss. Another important component of the paper is the comprehensive evaluation. They perform a variety of different experiments. For example, the authors investigate two types of noise generation: class-wise and sample-wise. Within that, they perform experiments to compare the test accuracy of random noise, error-maximizing noise, and error-minimizing noise. Additionally, they test the accuracy of the error-minimizing noise on 4 different popular datasets (SVHN, CIFAR-10, CIFAR-100, ImageNet). They also create more realistic experiments where a small subset of the identities are unlearnable and test on this. In all of these experiments, the evaluations present a favorable outcome in which the test accuracy is low for the error-minimized datasets. However, the approaches in the paper do have limitations. There are practical problems with the large-scale implementation of this methodology that the authors do concede to. In the same way, the performance of error minimizing noise in adversarial training and against representational learning seems to have minimal effect. The authors do have several ideas for fixing these limitations which they claim to hope to solve in future research.

2 Related Work

2.1 Adversarial Example Generation

This paper re-imagines the concept of Adversarial Example Generation in the form of a defensive mechanism to protect the privacy of a user's image data. Due to the fact, I have limited knowledge of adversarial examples and it is the foundation for this paper, I researched the references made by the authors and gained a more comprehensive understanding which is described in this section. It was discovered by Szegedy *et al.* (2013) that deep neural networks have an interesting and exploitable flaw in that they are very susceptible to misclassifying adversarial examples. Adversarial examples can be characterized as input that undergoes a small, imperceptible transformation that results in the trained model misclassifying it with a high level of confidence. This was a surprising discovery because Deep Learning Models were thought to be very robust as they were not fooled by other types of noise such as random noise. Goodfellow *et al.* (2014) proposed that the approximate linear behavior of classifiers in high dimensional spaces makes deep learning models susceptible to adversarial examples. Other papers have published competing hypotheses. Nevertheless, the more relevant concept to this paper is the Fast Gradient Sign Method, a way of creating these adversarial examples, that Goodfellow *et al.* (2014) introduces. Naively, we could generate a random set of inputs and name it to be a possible "adversarial example". However, these randomly generated inputs would be nothing like the original and no longer considered an "imperceptible" change to the human. Instead, to achieve this imperceptible change, we can calculate the gradient on the input while keeping the model parameters unchanged. Then, we move in the same direction as the gradient by a

small factor of α. This is results in a small change in input that fools the deep neural network. The new adversarial example can be formulated more formally as follows:

$$x_{adv} = x + \alpha * sign(\nabla_x L(\theta, x, y)) \tag{1}$$

There have been works that followed that made more realistic restrictions on the adversarial generation process such as only knowing the output data and not the model parameters or loss function like in the example explained above. But the objective of all of these processes is to trick Deep Neural Networks at test time by adding maximum imperceptible error.

3 Method

3.1 Error Maximizing Noise to Error Minimizing Noise

As discussed in the previous section, the process of adversarial example generation is to find the error maximizing noise that can be added to the input data. This objective can be expressed as a min-max optimization problem where you are trying to minimize the parameters while maximizing the noise. The problem with this type of noise is that it is mostly used during test time and cannot prevent DNN from learning when applied in a sample-wise manner. Therefore, the authors of this paper formulated an error minimizing noise that aims to add perturbation in a way that minimizes the loss when training. At the most basic level, Machine learning models train by correcting mistakes, and if there are no mistakes to correct then it will not alter the model. Therefore, if each perturbed input example had a low loss then the model would train thinking that the accuracy is really high, but, in reality, it results in a very low test accuracy. This is more formally represented by:

$$\arg\min \underset{(x,y)\sim D_c}{\mathbf{E}} \left[\min_{\delta} L(f'(x+\delta), y) \right] s.t. \quad \|\delta\| \leq \epsilon \tag{2}$$

This is a bi-level optimization problem which means that there is a nested optimization present. The outer level optimization is the standard machine learning formulation of minimizing expected loss with respect to the model parameter theta and where (x,y) are sampled from the clean data, D_c. The inner level optimization is the proposed error minimizing noise where we minimize the loss with respect to the noise δ. In order to make sure, that the noise being added does not become perceptible it is bound by ϵ which through experimentation has been deemed to be 8/255.

3.2 Learn the Noise

The error-minimizing noise is learnt through projected gradient descent over the model parameters and the noise. Projected Gradient Descent is a variation of gradient descent that takes into consideration constraints and maps the gradient to the projected (constrained) space. In this case, the objective formulated in the previous section has a constraint of the noise being less than ϵ in order to make sure the perturbation is not noticeable.

Because it is a bi-level optimization problem, the algorithm optimizes theta for a number of steps, M, and then the perturbation is updated. This process is repeated for T steps. For sample-size noise generation the $T + 1^{th}$ perturbation update is as follows:

$$x'_{t+1} = \Pi_\epsilon(x'_t - \alpha * sign(\nabla_x L(f'(x'_t), y)) \tag{3}$$

Breaking down this equation, x'_t represents the perturbed sample at the t^{th} iteration and x'_{t+1} is at the $t + 1^{th}$ iteration. $alpha$ represents the step size and $\nabla_x L(f'(x'_t), y)$ is the gradient with respect to the input. Π_ϵ is a function that clips the noise and enforces the constraint in the objective. The noise can then be derived as follows:

$$\delta = x' - x \tag{4}$$

A more detailed pseudocode, taken from the paper, for the algorithm is in the appendix.

4 Experiments

4.1 Overview of Experiments in Paper

The authors of this paper do a varied number of experiments to prove the robustness and generalization capability of the error-minimizing noise proposed in the paper. In this section, I briefly summarize the purpose and findings of each experiment in the paper.

Sample vs Class-wise Noise
In this set of experiments, the authors compare the accuracy of the different types of noise applied sample-wise and class-wise. Sample-wise noise is applied to each input and class-wise noise is applied to only a chosen class of images. They compared the training accuracy over 60 epochs of random noise, error-maximizing noise, and error-minimizing noise. They found error-minimizing noise to be very effective and have lower accuracies (around 20%) when applied sample-wise in comparison to the other types of noise that eventually reached close to 100% accuracy. Even more surprising, is that when applied in a class-wise manner after a certain number of epochs (e.g. 15) all the types of noise are effective and show low accuracies. The authors postulate that class-wise noise application breaks the i.i.d assumption, therefore, has difficulty generalizing. Essentially, in the class-wise application, the model learns the noise instead of the data resulting in low accuracy. On the other hand, sample-wise noise is applied for each sample so only error-minimizing noise would be effective because a low error sample would be the only one not taken into account in the model.

Error Minimizing Noise on Different Data Sets
In this set of experiments, the authors generated sample-wise and class-wise noise on four different datasets and tested the accuracy across four deep learning neural network models. They used the ResNet18 model as the base classifier for training and generated noise on 100% of the dataset. They tested the unlearnable and clean data set on the following deep neural networks: VGG-11, RN-18, RN-50, and DN-121. Additionally, those models were tested across four data sets: SVHN, CIFAR-10, CIFAR-100, and ImageNet. The results of these experiments prove the effectiveness of error-minimizing noise as when it is applied in both a class-wise and sample-wise manner it produces very low accuracies (2-35%).

Stability Experiments
The purpose of the stability experiments was to understand the capabilities of error-minimizing noise when applied to other data protection scenarios. In the first set of experiments, they set the noise generation from 20 to 100% to observe the effectiveness of noise when applied to a proportion of the data. Unfortunately, it was found that error-minimizing noise is only fully effective when applied to 100% of the data set. For further analysis, the authors broke down the accuracy scores into accuracy on the clean portion and accuracy on the unlearnable portion. This produced interesting results. For example, in the 80-20 split (80% unlearnable and 20% clean), they found that the total accuracy scores were dominated by the 20% clean set. Essentially, for the 80% that was unlearnable, the accuracy was very low. However, for the 20% that was clean, the accuracy was high. This suggests that even 20% clean data can achieve a good performance. More analysis was done in this section by creating a single unlearnable class and applying noise filtering techniques.

Transferability Experiments
Transferability, in deep learning, refers to whether data that misleads a particular model can mislead another. Essentially, in this case, it is determining whether noise generated from one dataset can be transferred to another and thereby impact its accuracy. For example, for the sample-wise noise generation, the authors used the CIFAR-10 and CIFAR-100 datasets which both have a class of 'ship' images. They added noise to the ship example in the CIFAR-10 dataset and then added the unlearnable images to the CIFAR-100 dataset to test the accuracy. They found that the ship images were unlearnable and were predicted to be in other classes (e.g. bird, cat, dog). They also completed the experiments with class-wise noise generation on the ImageNet and CIFAR dataset and found similar effective results.

Real-world experiment
The possible impact of applying error-minimizing noise to data is to prevent unauthorized use of it to train DNNs. With this line of thought, the authors created a real-world experiment for facial recognition and applied noise to a set of data in WebFace. They wanted to emulate the idea of users uploading their perturbed images to social media which would be used by deep learning neural

networks for facial recognition software but hopefully results in inaccurate classifications. They set up the experiment in partial unlearnable, where only some of the data is perturbed, and fully unlearnable, where all of the data is perturbed. In the face recognition task, they found that in even partially unlearnable setting face recognition has low accuracies (about 16%). In the face verification, the noise was not effective in a partially unlearnable setting, but it was in the fully unlearnable.

4.2 Code & Experiment Exploration

The authors of this paper published the accompanying code with each of their experiments in a Github repository which is available at: https://github.com/HanxunH/Unlearnable-Examples. Therefore, in addition to a deep technical review of the paper, I explored the published code. First, I worked on setting up the repository and running the quick start guide which proved to be a challenge due to resources. Because this is a computer vision-based research experiment and requires more computational power, I had to search for more powerful computers to run the experiment on. I tried several options such as the iLab machines but eventually ended up toggling between using AWS and Google Colab. Google Colab does not guarantee the availability of GPU; therefore, when it was not available, I used an AWS Ec2 instance with Deep Learning Image that had the necessary dependencies. After setting up, I ran the Quickstart Jupyter notebook in the repository that had an example of adding sample-wise noise to create an unlearnable dataset. With limited knowledge of PyTorch, I spent some time understanding each section of the code. The notebook had three components: generate error-minimizing noise on data set, apply noise to data, train deep learning model on unlearnable data. The example used the base model of ResNet18 for training and the CIFAR-10 dataset. The CIFAR-10 consists of 60,000 images (50,000 for training and 10,000 for testing) in 10 classes where each image is 32 by 32. The noise was learned and represented in a 50000 by 32 by 32 by 3 array.

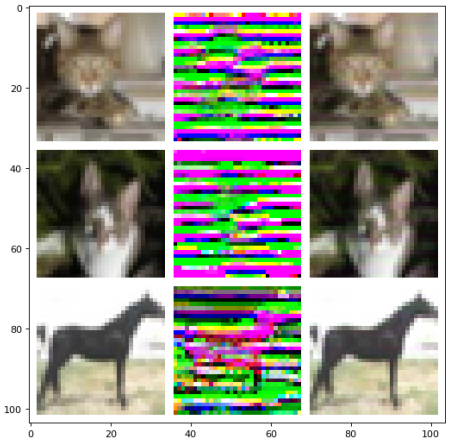
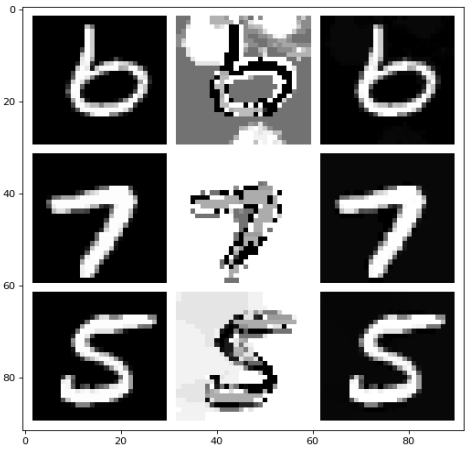


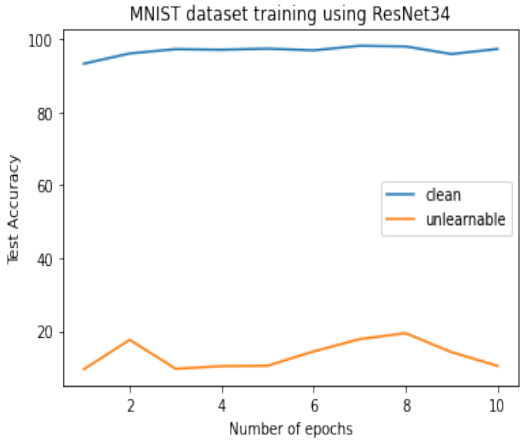
Figure 1: Sample-wise noise generated on CIFAR-10 dataset through my run of the Quickstart Notebook

Furthermore, I attempted to run some of the scripts to replicate some of the other experiments. Unfortunately, those results took too long to obtain and were infeasible for me to try and replicate in a short time. Nevertheless, in addition to running the CIFAR-10 dataset, I wanted to try another dataset that was not tested in the paper. Therefore, I chose the MNIST dataset. This is a database of handwritten black and white digits (28 by 28) with a training set of 60,000 and a test set of 10,000. I had to make several modifications to the quick start guide because the size of these images was different from CIFAR-10 and they were not RGB values. Also, I made small modifications to the ResNet18 model as well as the generation process for creating the unlearnable examples.

For the MNIST dataset, I used the base model of ResNet18 to generate the noise that was added to create the unlearnable dataset. A sample of three images is visualized in 2a. The unlearnable data set represents the original data perturbed by the error-minimizing noise. Then, I trained ResNet34 on the clean dataset and the unlearnable dataset to determine the effectiveness of error-minimizing



(a) Adding noise to the MNIST dataset



(b) Accuracy of clean data vs unlearnable data for DNN ResNet34

Figure 2: Results for MNIST dataset

noise on MNIST. The results are shown in the graph 2b. Clearly, after training for 10 epochs, the test accuracy of the unlearnable dataset is much lower than the test accuracy on the clean dataset which corroborates the experiments done in the paper.

5 Conclusions

In this deep technical review, I discuss the components of the methodology to make data unlearnable for deep neural networks. In comparison with adversarial example generation, the process consists of creating error-minimizing noise that is introduced at training time. This noise has a minimal loss resulting in the deep neural network thinking there is nothing more to learn. I empirically verified some of the results in the paper by setting up the accompanying code. In addition, I implemented a new experiment using the MNIST dataset and the low accuracies on the unlearnable data substantiated the claims made by the authors. The novel concepts presented in the research paper are impactful because it provides the ability for users to better control and protect their data through adding noise that prevents its use in DNNs. Moreover, it encourages more areas to explore in data protection.

References

Goodfellow, I., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv 1412.6572*.

Huang, H., Ma, X., Erfani, S. M., Bailey, J., and Wang, Y. (2021). Unlearnable examples: Making personal data unexploitable. In *International Conference on Learning Representations*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks.

A Appendix

Algorithm 1 Error-minimizing Perturbations

Input: Source Model θ, Perturbations $\boldsymbol{\delta}$, L_p ϵ, $(\boldsymbol{x}, y) \in D_c$, Stop Error λ, Optimization steps M
Output: δ
repeat
  for m **in** $1 \cdots M$ **do**
    $i, \boldsymbol{x}_i, y_i$ = Next$(\boldsymbol{x}, y)$
    if sample-wise **then**
      $\boldsymbol{\delta} = \boldsymbol{\delta}_i$
    else if class-wise **then**
      $\boldsymbol{\delta} = \boldsymbol{\delta}_{y_i}$
    end if
    Optimize$(\boldsymbol{x}_i + \boldsymbol{\delta}, \boldsymbol{y_i}, \theta)$
  end for
  for $\boldsymbol{x}_j, y_j$ **in** $\boldsymbol{x}, y$ **do**
    $\boldsymbol{\delta}$ = Perturbation$(\boldsymbol{x}_i, y_i, \theta, \boldsymbol{\delta})$ ▷ Follow Equation 3
    Cilp$(\boldsymbol{\delta}, -\epsilon, \epsilon)$
  end for
  $error$ = Eval$(\boldsymbol{x}, y, \boldsymbol{\delta}, \theta)$
until $error < \lambda$

This figure is taken directly from the psuedocode outline in Appendix A of the paper (Huang *et al.*, 2021). Perturbation is applied as in equation 3 in this paper. This outline is what is followed in the quickstart notebook when generating the sample wise noise.