Міністерство освіти і науки України
Київський Національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідувач кафедри кібербезпеки
та захисту інформації
_____ Н.В. Лукова-Чуйко
“_____”_____ 2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
Дипломної роботи

магістра
(назва освітньо-кваліфікаційного рівня)

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній рівень ***магістр***
(назва освітнього рівня)

кваліфікація *магістр кібербезпеки*
(код і назва кваліфікації)

на тему: Удосконалення системи багатофакторної автентифікації з використанням біометричних методів

	Прізвище, ініціали	**Оцінка**	**Підпис**
Науковий керівник	Лукова-Чуйко Н. В.		
Рецензент			
Нормоконтроль	Фесенко А.О.		

Київ
2021

Міністерство освіти і науки України
Київський Національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри
кібербезпеки та захисту інформації
_____ Н.В. Лукова-Чуйко
“_____”_____ 2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека

студенту КБм-21 (група) Савченко Кирилу Станіславовичу (прізвище ім’я по-батькові)

Тема дипломної роботи Удосконалення системи багатофакторної автентифікації з використанням біометричних методів

1. **ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ**

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол №2 від 08.10.2020 р.

2. **МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ**

Об’єкт досліджень Процес багатофакторної автентифікації з використанням біометричних методів.

Предмет досліджень Модель системи багатофакторної автентифікації з використанням мультимодального біометричного підходу

Мета роботи Визначення та реалізація оптимального біометричного методу Ідентифікації

Вихідні дані для проведення роботи Фотографії обличчя та запис голосу

3. **ОЧІКУВАНІ РЕЗУЛЬТАТИ**

Практична цінність Розробка нейромережового підходу системи багатофакторної автентифікації з використанням біометрики.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота повинна виконуватися згідно діючої законодавчої та нормативної бази в сфері автентифікації та біометрії

ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	**Строки виконання робіт (початок-кінець)**
1. Уточнення поставленої задачі.	12.10.2020 – 16.10.2020
2. Аналіз літератури.	19.10.2020 – 10.12.2020
3. Збір даних. Обґрунтування вибору рішення.	25.01.2021 – 12.02.2021
4. Дослідження процесу проведення проактивного пошуку загроз.	15.02.2021 – 26.02.2021
5. Пошук шляхів вдосконалення проактивного пошуку загроз.	05.04.2021 – 16.04.2021
6. Аналіз результатів проведеного дослідження.	19.04.2021 – 07.05.2021
7. Оформлення пояснювальної записки. Підготовка до захисту роботи	11.05.2021 – 17.05.2021

5. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ Лукова-Чуйко Н.В.

Завдання прийняв
до виконання _____ Савченко К.С.

Дата видачі завдання: 12.10.2020

Термін подання дипломної роботи до ЕК 17.05.2021

УДК 004.852+57.087

РЕФЕРАТ

Пояснювальна записка: 85 с., 21 рис., 3 табл., 2 додатки, 37 джерел.

Об’єкт дослідження: процес тренування та використання згорткової нейронної мережі.

Мета роботи: створення рекомендацій для тренування нейронної мережі для протидії атакам інверсійної моделі.

Предмет дослідження: модель системи оптимізації процесу тренування нейронної мережі з використанням технік диференційної приватності.

Методи дослідження: інформаційний аналіз, абстрагування, системний підхід, порівняння, математична статистика, емпіричні дослідження.

Новизна роботи: створення нового методу оптимізації тренування нейронної мережі.

В роботі проведено розробку нового методу оптимізації процесу тренування нейронної мережі DP-AdamW.

Запропоновано рекомендації щодо процесу тренування нейронної мережі для протидії атакам інверсійної моделі.

Практична цінність полягає у розробці нейромережевого підходу системи багатофакторної автентифікації з використанням біометрики.

Результати здійснених у дипломній роботі досліджень можуть бути використані підприємствами, які тренують нейронні мережі з використанням приватних датасетів.

Напрямки подальших досліджень: покращення точності оптимізаторів з використанням технік диференційної приватності.

Ключові слова: диференційна приватність, згорткові нейронні мережі, тренування нейронної мережі, атака інверсійної моделі, змагальна атака, глибоке навчання, інформаційна безпека.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ДПГ – Диференційна Приватність Гауса;

ШІ – Штучний інтелект;

API – Application Programming Interface;

PGD – False Rejection Rate;

TTM – Equal Error Rate;

GAN – Half-Total Error Rate;

HTTP – HyperText Transfer Protocol;

HTTPS – HyperText Transfer Protocol Secure.

ВСТУП

Актуальність теми нейронних мереж на сьогоднішний день грає велику роль. Згідно Gartner в період з 2015 до 2019 року кількість світових організацій, які почали використовувати нейронні мережі збільшилася на 270% а згідно McKinsey 44% фірм, які використовують ШІ, повідомляють про скорочення витрат бізнесу у відділах, де впроваджується ШІ. Також згідно Salesforce близько 62% користувачів готові переходити на інструменти ШІ та нейронних мереж для покращення досвіду.

Саме тому проблема безпеки нейронних мереж стає гострим питанням для багатьох організацій на сьогоднішній день, а кількість рішень з використанням приватних датасетів збільшується з кожним роком.

Метою дипломної роботи є створення рекомендацій для тренування нейронної мережі для протидії атакам інверсійної моделі.

Об’єктом дослідження є процес тренування та використання згорткової нейронної мережі.

Предметом дипломної роботи є модель системи оптимізації процесу тренування нейронної мережі з використанням технік диференційної приватності.

Практична цінність полягає у розробці нового методу оптимізації тренування згорткових нейронних мереж.

Новизна роботи полягає у створенні нового методу оптимізації тренування нейронної мережі.

При виконанні дипломної роботи були використані методи порівняння, статистичного та структурного аналізу, системного аналізу, моделювання.

Завданнями дипломної роботи є:

1. Проаналізувати сучасні способи використання згорткових нейронних мереж.
2. Проаналізувати проблеми безпеки згорткових нейронних мереж.

3. Дослідити недоліки процесу тренування згорткової нейронної мережі.
4. Використати атаку інверсійної моделі на згорткові мережі, натреновані на датасеті CIFAR-10.
5. Розробити новий метод оптимізації тренування DP-AdamW.
6. Розробити рекомендації щодо безпеки використання нейронних мереж.

РОЗДІЛ 1
ДИФЕРЕНЦІЙНА ПРИВАТНІСТЬ ЯК ПРОБЛЕМА БЕЗПЕКИ НЕЙРОННИХ МЕРЕЖ

1.1 Сучасне використання згорткових нейронних мереж

На сьогоднішній день існує безліч задач, в яких можуть використовуватися інструменти штучного інтелекту, зокрема нейронні мережі. Згідно ресурсу PapersWithCode безліч класів задач, пов’язаних з комп’ютерним баченням вирішуються завдяки згортковим нейронним мережам.

Згорткова ж нейронна мережа за рахунок застосування спеціальної операції – власне згортки – дозволяє водночас зменшити кількість інформації, що зберігається в пам'яті інформації, за рахунок чого краще справляється з картинками більш високої роздільної здатності, і виділити опорні ознаки зображення, такі як ребра, контури або грані. На наступному рівні обробки з цих ребер і граней можна розпізнати повторювані фрагменти текстур, які далі можуть скластися в фрагменти зображення.

По суті кожен шар нейронної мережі використовує власне перетворення. Якщо на перших шарах мережа оперує такими поняттями як "ребра", "грані" і т.п, то далі використовуються поняття "текстура", "частини об'єктів". В результаті такого опрацювання можна правильно класифікувати картинку або виділити на кінцевому етапі потрібний об'єкт на зображенні.

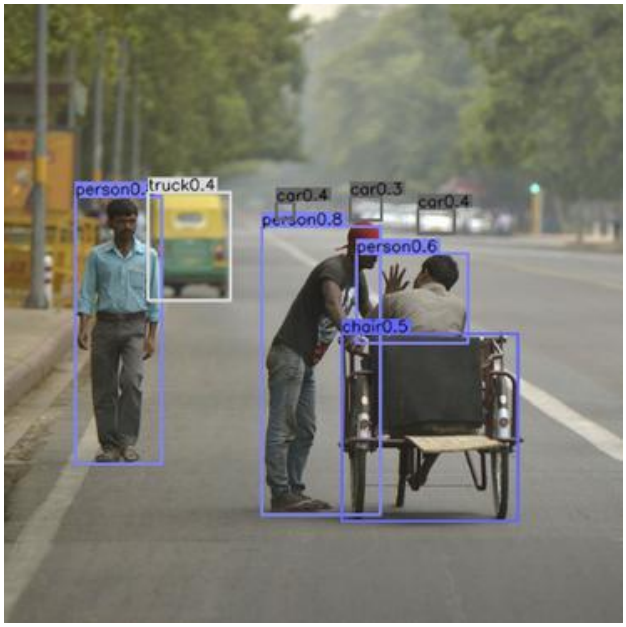

Рисунок 1.1 Приклад роботи згорткової нейронної мережі Yolo для детекції об’єктів на зображенні

У даній роботі здебільшого розглядалися задачі пов’язані з класифікацією зображень та розпізнаванню обличчя та згорткові нейронні мережі, однак дослідження проведені у ході роботи не обмежуються даними архітектурами та типами даних. Обмеження складає здебільшого процес тренування.

1.2 Поняття приватності в контексті глибокого навчання

Вирішення будь-якої задачі з використанням технологій глибокого навчання потребує наступні компоненти:

- Постановка задачі;
- Дані;
- Метрика оцінювання.

У ході даної роботи дослідження проблем безпеки нейронних мереж здебільшого розглядає домен комп’ютерного бачення, наприклад задач ідентифікації обличчя, класифікації зображень та інших. Питання приватності використання мереж стосується даних для тренування нейронної мережі.

Згідно Європейської комісії, "Персональні дані - це інформація, яка стосується ідентифікованої особи. Якщо ви не можете безпосередньо

ідентифікувати особу за цією інформацією, то вам потрібно розглянути, чи може ця особа все ще ідентифікуватися. Ви повинні брати до уваги інформацію, яку обробляєте, разом із усіма засобами, які з розумною ймовірністю будуть використані вами або будь-якою іншою особою для ідентифікації цієї особи." Точні визначення таких термінів, як "персональні дані", "обробка", "суб'єкт даних", "контролер" та "обробник" зазначені у статті 4 Регламенту.

Часто для підготовки моделей потрібні великі репрезентативні набори даних, які можуть бути об’єднані в краудсорсинг та містити конфіденційну інформацію. Моделі не повинні надавати приватну інформацію в цих наборах даних.

Диференційна приватність є міцним стандартом для гарантій конфіденційності алгоритмів у сукупних базах даних. Це визначається з точки зору специфічної для програми концепції суміжних баз даних. Наприклад, у цих експериментах кожен набір навчальних даних являє собою набір пар зображень-міток; ми говоримо, що два з цих наборів сусідні, якщо вони відрізняються в одному записі, тобто якщо одна пара міток зображень присутня в одному наборі, а відсутня в іншому.

Системи машинного навчання часто містять елементи, які сприяють захисту своїх навчальних даних. Зокрема, методи регуляризації, які мають на меті уникнути перенавчання прикладів, що використовуються для навчання, можуть приховувати деталі цих прикладів. З іншого боку, пояснити внутрішні уявлення в глибоких нейронних мережах, як відомо, складно, і їх велика потужність тягне за собою те, що ці уявлення можуть потенційно кодувати тонкі деталі принаймні деяких навчальних даних. У деяких випадках цілеспрямований зловмисник може бути в змозі витягти частини навчальних даних.

Рисунок 1.2 Зображення, відновлене за допомогою модельної інверсійної атаки (ліворуч) та тренувального зображення жертви (праворуч)

Зловмисникові надається лише ім’я людини та доступ до системи розпізнавання обличчя, яка повертає оцінку довіри класу.

1.3 Проблеми процесу тренування нейронної мережі

Глибокі нейронні мережі, які надзвичайно ефективні для багатьох завдань машинного навчання, визначають параметризовані функції від входів до виходів як композиції багатьох шарів основних будівельних блоків, таких як афінні перетворення та прості нелінійні функції. Загальновживаними прикладами останніх є сигмоїди та випрямлені лінійні одиниці (ReLU). Змінюючи параметри цих блоків, ми можемо «навчити» таку параметризовану функцію з метою підгонки будь-якого заданого кінцевого набору прикладів вводу/виводу. Точніше, ми визначаємо функцію втрат L, яка представляє штраф за невідповідність результатів нейронної мережі міткам навчальних даних. Втрата L (θ) на параметрах θ є середнім значенням втрат за навчальними прикладами {x1,. . . , xN}. Навчання полягає у знаходженні θ, яке приносить прийнятно малі втрати, ймовірно, найменші втрати (хоча на практиці ми рідко очікуємо досягнення точного глобального мінімуму).

Математично метод диференційної приватності можна сформулювати наступним чином:

Нехай $\varepsilon > 0$, A – рандомізований алгоритм, який приймає на вхід приватний набір даних D_1, а E – область значень цього алгоритму. Алгоритм A є ε - диференційно приватним, якщо для всіх записів наборів 20 даних D_1і D_2, які відрізняються єдиним записом, для всіх підмножин $S \subseteq E(A)$виконується наступна нерівність:

$p\{A(D_1) \in S\} \leq e^{\varepsilon} \cdot p\{A(D_2) \in S\}$, де p – ймовірність отримана з випадковості рандомізованого алгоритму.

1.4 Атаки на методи глибокого навчання

1.4.1 Змагальна атака

Кілька моделей машинного навчання, включаючи сучасні нейронні мережі, є вразливими до прикладів змагальності. Тобто ці моделі машинного навчання неправильно класифікують приклади, які лише трохи відрізняються від правильно класифікованих прикладів, отриманих з розподілу даних.

У багатьох випадках широкий спектр моделей з різними архітектурами, навчених на різних підмножинах навчальних даних, помилково класифікують той самий змагальний приклад. Це свідчить про те, що приклади змагальності виявляють фундаментальні сліпі місця в наших алгоритмах навчання. Причина цих змагальних прикладів була загадкою, і спекулятивні пояснення припускали, що це пояснюється надзвичайною нелінійністю глибоких нейронних мереж, можливо, в поєднанні з недостатнім усередненням моделі та недостатньою регуляризацією чисто контрольованої проблеми навчання. Ми показуємо, що ці спекулятивні гіпотези непотрібні. Лінійна поведінка у високомірних просторах достатньо для того, щоб викликати змагальні приклади. Ця точка зору дає нам змогу розробити швидкий метод формування змагальних прикладів, що робить практичне змагальне навчання. Ми показуємо, що змагальна підготовка може забезпечити додаткову вигоду регуляризації, за винятком тих, які надаються за допомогою відсіву (Srivastava et al., 2014).

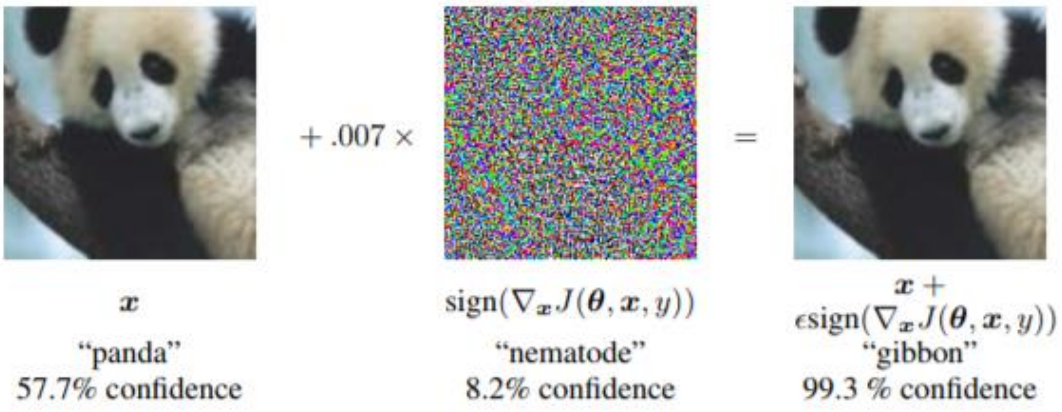

Рисунок 1.3 - Приклад використання змагальної атаки

На рисунку 1.3 приклад того, як можно ввести нейромережу в оману використанням невеликої кількості RGB-пікселей того ж розміру, як самого зображення з метою змінити результат класифікації нейромережі.

Adversarial приклад - вектор, що подається на вхід алгоритму, на якому алгоритм видає некоректний вихід.

Adversarial атака - алгоритм дій, метою якого є отримання Adversarial прикладу.

Щоб зрозуміти проблему Adversarial прикладів, давайте згадаємо одну з задач машинного навчання - навчання з учителем при класифікації. У цьому завданню у нас є пари «об'єкт-мітка», і ми повинні навчитися передбачати значення для нових об'єктів.

Якщо розглянути цю задачу з геометричної точки зору, то необхідно розділити простір таким чином, щоб на новому об'єкті передбачити «правильний» клас. Більш того, якби ми мали генеральну сукупність даних (наприклад, для набору рукописних цифр MNIST мати всілякі зображення всіх цифр), то дану гіперплощину можна було б провести ідеально за умови відокрмеленості класів. Але так як генеральної сукупності найчастіше не буває, то для вирішення даного завдання ми і використовуємо алгоритми машинного навчання - щоб максимально точно наблизити «ідеальну» гіперплощину за допомогою тих даних, що у нас є.

Будь-яке відхилення гіперплощини від ідеальної, породжує певний «зазор», потрапляючи в який, об'єкти класифікуються некоректно. Саме тому і з'являються такі приклади, як панда, класифікована як гібон. А завдання атакуючого зводиться до зміни вектора параметрів об'єкта таким чином, щоб він потрапив в цей «зазор».

Приклади Adversarial атак.

Є нейронна мережа, яка знаходить осіб на фотографії. Вона успішно справляється з поставленим завданням (зображення зліва). Але після додавання до цієї фотографії незначного шуму (зображення праворуч), алгоритм на отриманому adversarial прикладі (зображення по центру) більше не детектує особу на зображенні.

Рисунок 1.4 - Демонстрація використання змагальних атак

Даний приклад, продемонстрований в статті «Adversarial Attacks on Face Detectors using Neural Net based Constrained Optimization», цікавий тим, що безліч реальних систем розпізнавання осіб використовують для детектування осіб саме нейромережеві підходи. Людина ж не помітить різниці при погляді на обидва зображення.

Існують різні методи захисту від змагальних атак.

Розглянемо Змагальне тренування, як основний метод забезпечення захисту та протидії атаки.

Такий підхід відразу ж був запропонований в статті Intriguing properties of neural networks ще в 2013 році. Саме в цій статті і була вперше описана дана проблема і L-BFGS атака, що дозволяє отримати Adversarial приклади.

Цей метод дуже простий. Ми генеруємо Adversarial приклади за допомогою різного роду атак і додаємо їх в навчальну вибірку на кожній ітерації, тим самим підвищуючи "опірність" моделі Adversarial прикладів.

Недолік же цього методу досить очевидний: на кожній ітерації навчання, на кожен приклад ми можемо згенерувати дуже велика кількість прикладів, відповідно, і час на навчання моделі зростає багаторазово.

1.4.2 Атака інверсивних моделей

Хоча атака інверсії моделі вимагає лише доступу «чорного ящика» до навченої моделі (тобто взаємодії з моделлю через входи та виходи), ми розглядаємо зловмисників з додатковими можливостями, подібно до Шокрі та Шматикова. Підхід у даній роботі пропонує захист від сильного супротивника з повним знанням механізму навчання та доступом до параметрів моделі. Цей захист привабливий, зокрема, для програм машинного навчання на мобільних телефонах, планшетах та інших пристроях. Зберігання моделей на пристрої забезпечує енергоефективний висновок із низькою затримкою та може сприяти конфіденційності, оскільки висновок не вимагає передачі даних користувача на центральний сервер; з іншого боку, ми повинні припустити, що самі параметри моделі можуть бути піддані ворожому контролю. Крім того, коли ми стурбовані збереженням конфіденційності одного запису в навчальних даних, ми допускаємо можливість того, що супротивник контролює деякі або навіть всі інші дані про навчання. На практиці цю можливість не завжди можна виключити, наприклад, коли дані є зібраними з публічних ресурсів.

1.5 Диференційна приватність в методі SGD

Можна спробувати захистити конфіденційність навчальних даних, працюючи лише над кінцевими параметрами, які є результатом навчального процесу, розглядаючи цей процес як чорний ящик. На жаль, загалом, може не бути корисної, щільної характеристики залежності цих параметрів від навчальних даних; додавання занадто консервативного шуму до параметрів, де шум вибирається відповідно до аналізу найгіршого випадку, погіршило б корисність вивченої моделі. Тому ми віддаємо перевагу більш витонченому підходу, при якому ми прагнемо контролювати вплив навчальних даних під час навчального процесу, зокрема при обчисленні SGD. Цього підходу дотримувались у попередніх роботах інші, однак у даній роботі було внесено кілька значних модифікацій та розширень, зокрема, в питанні приватності. Алгоритм 1 окреслює основний метод навчання моделі з параметрами θ шляхом мінімізації емпіричної функції втрат L (θ). На кожному кроці SGD ми обчислюємо градієнт ∇θL (θ, xi) для випадкової підмножини прикладів, відсікаємо l2 норми кожного градієнта, обчислюємо середнє, додаємо шум для захисту конфіденційності та робимо крок у протилежному напрямку від цього середнього шумного градієнта. Врешті-решт, окрім виведення моделі, нам також потрібно буде обчислити втрату приватності механізму на основі інформації, що зберігається обчислювачам приватності.

Основними підходами є:

- Відсікання градієнту;
- Пошарові та часові параметри;
- Лоти;
- Обчислення кроку приватності;
- Обчислення кроку швидкості;

1.5.1 Обчислення кроку швидкості

Моменти, які обчислювач відстежує, пов'язані з моментами випадкової величини втрати конфіденційності, яка узагальнює стандартний підхід відстеження (ε, δ) та використовує теорему про сильний склад. Хоча таке вдосконалення раніше було відоме для складання гаусових механізмів, у ході роботи було показано, що він застосовується також для складання гаусових механізмів із випадковою вибіркою і може набагато чіткіше оцінити втрату конфіденційності. Втрата конфіденційності є випадковою величиною, що залежить від випадкового шуму, доданого до алгоритму. Те, що механізм M є (ε, δ) диференціально приватним, еквівалентно певному хвосту, обмеженому випадковою величиною втрати конфіденційності M. Хоча межовий хвіст є дуже корисною інформацією про розподіл, складання безпосередньо з нього може призвести до досить вільних меж. Натомість у роботі обчислюються моменти журналу випадкової величини втрати конфіденційності, які складаються лінійно.

1.6 Тюнінг гіперпараметрів моделі

Ми визначаємо характеристики моделей, що стосуються конфіденційності, і, зокрема, гіперпараметри, які ми можемо налаштувати, щоб збалансувати конфіденційність, точність та продуктивність. Зокрема, за допомогою експериментів ми спостерігаємо, що точність моделі більш чутлива до параметрів навчання, таких як розмір батчу та рівень шуму, ніж до структури нейронної мережі. Якщо ми спробуємо кілька налаштувань для гіперпараметрів, ми можемо тривіально скласти витрати на конфіденційність усіх налаштувань, можливо, через обчислювача моментів(швидкості). Однак, оскільки ми дбаємо лише про параметр, який дає нам найбільш точну модель, ми можемо зробити краще. Ми можемо використовувати теоретичні уявлення для зменшення кількості налаштувань гіперпараметрів, які потрібно спробувати. Хоча диференціально приватну оптимізацію опуклих цільових

функцій найкраще досягти, використовуючи розміри батчу 1, яке за своєю суттю менш стабільне, отримуємо користь від агрегування у більші батчі. У той же час теорема 1 припускає, що занадто великі батчі збільшують витрати на конфіденційність, і розумним компромісом є прийняття кількості батчей за епоху в той самий порядок, що і бажана кількість епох. Швидкість навчання у неприватному навчанні зазвичай ретельно коригується вниз, оскільки модель сходиться до локального оптимуму. На відміну від цього, нам ніколи не потрібно знижувати коефіцієнт кроку навчання до дуже малого значення, оскільки диференційоване приватне навчання ніколи не досягає режиму, коли це було б виправдано. З іншого боку, в наших експериментах ми виявляємо, що користь починати з відносно великої швидкості навчання, потім лінійно знижувати її до меншого значення за кілька епох і підтримувати її постійною після цього є більш корисним способом.

1.7 Проблема диференційної приватності в контексті машинного навчання

У статті Фредріксона автори пропонують моделі глибокого навчання з гарантіями конфіденційності в рамках f-диференціальної конфіденційності, яка нещодавно була запропонована як об'єднуюче узагальнення кількох попередніх визначень конфіденційності. Автори тренують свої приватні нейронні мережі для виконання кількох завдань, що включають різні типи даних, включаючи зображення, тексти та числові дані. Завдяки універсальності f-диференціальної системи конфіденційності експериментальні результати демонструють, що цей підхід до приватного глибокого навчання перевершує існуючі підходи з точки зору компромісу між приватністю та корисністю.

У багатьох програмах машинного навчання набори даних містять конфіденційну інформацію про людей, таких як місцезнаходження, особисті контакти, споживання засобів масової інформації та медичні записи. Використовуючи результати алгоритму машинного навчання, супротивник

може виявити деяких осіб у наборі даних, представляючи тим самим серйозні проблеми щодо конфіденційності. Ця реальність породила широкий і нагальний заклик до розробки методологій аналізу даних, що зберігають конфіденційність. Відповідно, в науковій літературі було проведено численні дослідження з питань захисту конфіденційності при аналізі даних - статистики, криптографії, машинного навчання та права.

На цьому шляху дослідницькі зусилля неодноразово вказували на необхідність суворого та різнобічного визначення приватності. Серед іншого, дослідники піддавали сумніву, чи дає використання визначення конфіденційності зрозумілі гарантії конфіденційності, і якщо так, чи це визначення конфіденційності забезпечує високу точність приватної моделі серед альтернативних визначень. Зокрема, було показано, що анонімізація як синтаксична та спеціальна концепція конфіденційності, як правило, не гарантує конфіденційність. Приклади включають ідентифікацію гомофобної особи в анонімізованому наборі даних Netflix Challenge та ідентифікацію медичних записів тодішнього губернатора штату Массачусетс у відкритих анонімних наборах медичних даних.

У цьому контексті f-диференційоване конфіденційність (DP) виникла як математично чітке визначення приватного життя. Сьогодні це визначення перетворилося на міцну основу аналізу приватних даних, а його програми розгортали Google, Apple, Microsoft та Бюро перепису населення США. Незважаючи на свою вражаючу популярність як в науковій літературі, так і в промисловості, f-DP недостатньо універсальний для обробки композицій, що є, мабуть, найбільш фундаментальним примітивом у статистичній конфіденційності. Наприклад, тренувальний процес глибоких нейронних мереж фактично є складом багатьох примітивних будівельних блоків, відомих як стохастичний градієнтний спуск (SGD). Однак за невеликого бюджету на конфіденційність у розумінні f-DP не було зрозуміло, як підтримувати високу точність прогнозування глибокого навчання. Це вимагає ретельного аналізу конфіденційності складу в рамках f-DP. Дійсно, аналіз витрат на

конфіденційність при глибокому навчанні був вдосконалений лише нещодавно, використовуючи складну техніку, яка називається бухгалтером моментів.

В ідеалі, у ході роботи основна ідея полягала у визначенні рівня конфіденційності, який дозволяє вдосконалити аналіз конфіденційності різних алгоритмів принципово не вдаючись до складних методів. Вишуканий аналіз конфіденційності не тільки підвищує надійність моделей, але також може бути використаний для підвищення точності прогнозування, розпродаючи конфіденційність для корисності. Одним із можливих кандидатів є f-диференційна приватність, розслаблення f-DP, яке нещодавно було запропоновано. Це нове визначення конфіденційності достовірно зберігає гіпотезу, що перевіряє інтерпретацію диференціальної конфіденційності, і може без втрат міркувати про загальні примітиви, пов'язані з диференціальною конфіденційністю, включаючи композицію, посилення конфіденційності за допомогою вибірки та конфіденційність груп. Крім того, f-DP включає канонічне сімейство з одним параметром, яке називають диференціальною приватністю Гауса (ДПГ). Примітно, що ДПГ є основним визначенням конфіденційності завдяки теоремі про центральну межу, яка стверджує, що гарантії конфіденційності складу приватних алгоритмів приблизно еквівалентні для розрізнення двох зрушених нормальних розподілів.

1.8 Огляд та аналіз методології f-DP

Проаналізувавши сучасні статті, пов’язані з диференційною приватністю можна зробити висновок, що f-DP пропонує сувору та універсальну структуру для розробки приватних методологій глибокого навчання. Така гарантія забезпечує захист від зловмисника знанням архітектури мережі, а також параметрів, вагів моделі. Основні виділені особливості методу f-DP:

1. Межі конфіденційності закритої форми. У рамках f-DP загальна втрата конфіденційності, спричинена навчанням нейронних мереж, допускає виразність у закритій формі. Навпаки, аналіз конфіденційності через моменти обчислювача повинен проводитись чисельним обчисленням, і неявний характер цього попереднього підходу може перешкодити розумінню того, як параметри налаштування впливають на обмеження конфіденційності.
2. Посиленіші гарантії конфіденційності. Підхід f-DP дає більші гарантії конфіденційності, ніж попередній підхід, навіть з точки зору f-DP. Це вдосконалення відбулося завдяки використанню центральної граничної теореми для f-DP, яка точно фіксує втрату конфіденційності, що виникає на кожній ітерації під час навчання моделей глибокого навчання.
3. Покращена точність прогнозування. Використовуючи потужніші гарантії конфіденційності, що надаються f-DP, можна використати певну кількість конфіденційності для покращення ефективності прогнозування. Це може бути реалізовано, наприклад, шляхом відповідного зменшення кількості шуму, доданого під час навчального процесу нейронних мереж, з тим, щоб відповідати цільовому рівню конфіденційності з точки зору f-DP.

Одним з основних питань f-DP є погіршення складу. Розвиток на цьому шляху включає основну теорему про композицію та вдосконалену теорему про композицію. У даній роботі використані та проаналізовані висновки теореми оптимального складу для f-DP, яка насправді послужила однією з мотивацій для роботи f-DP. Однак важко обчислити межі конфіденційності з теореми про їх склад. Зовсім недавно отримані чіткі композиційні межі загальної втрати конфіденційності для експоненціальних механізмів.

З іншого боку, значні останні зусилля були присвячені послабленню диференціальної приватності з використанням розбіжностей розподілу ймовірностей для подолання слабкості f-DP у складі обробки. На жаль, цим розслабленням або не вистачає посилення конфіденційності за допомогою

аргументу під вибірки, або це досить складний аргумент, який важко використовувати у різних налаштуваннях. Оскільки субдискретизація невід'ємно використовується в навчанні нейронних мереж, тому важко застосувати ці послаблення безпосередньо до аналізу конфіденційності глибокого навчання.

Щоб обійти ці технічні труднощі, пов'язані з f-DP та його послабленням, заснованими на розбіжностях, була використана методика, яку називають «обчислювачем моментів», щоб відстежувати детальну інформацію про втрату конфіденційності в процесі навчання глибоких нейронних мереж. Використовуючи обчислювач моментів, проведений аналіз значно покращує попередній аналіз конфіденційності SGD та дозволяє забезпечити значні гарантії конфіденційності для глибокого навчання, підготовленого на наборах даних з реальними розмірами. Цей прийом поширився на різні ситуації завдяки подальшій роботі. На відміну від цього, використаний підхід до приватного глибокого навчання в рамках f-DP використовує деякі потужні інструменти цього нового визначення конфіденційності, проте забезпечуючи більш чіткий аналіз конфіденційності, як теоретично, так і емпірично.

Для повноти зауважується, що були запропоновані різні підходи для включення міркувань конфіденційності до глибокого навчання, не використовуючи ітеративні та субдискретизаційні особливості навчання моделей глибокого навчання. Цей напрямок роботи включає підготовку приватної моделі ансамблем моделей "викладачів", розробку шумових об'єднаних алгоритмів усереднення та аналіз витрат на конфіденційність через аналіз процесу оптимізації нейронних мереж.

1.9 Особливості процесу диференційної приватності

У рамках диференційної приватності передбачаємо супротивника, який добре обізнаний про набір даних, за винятком окремо взятої особи, і противник прагне визначити, чи є ця особа у наборі даних на основі

результатів алгоритму. Грубо кажучи, алгоритм вважається приватним, якщо супротивникові важко визначити наявність чи відсутність будь-якої особи.

Неофіційно набір даних можна сприймати як матрицю, кожен рядок якої містить дані однієї людини. Два набори даних називаються сусідами, якщо один можна отримати, відкинувши особу від іншого. Таким чином, розміри сусідніх наборів даних відрізняються на один.

neighboring data sets differ by one.[2] Let S and S' be neighboring data sets, and $\varepsilon \geqslant 0, 0 \leqslant \delta \leqslant 1$ be two numbers, and denote by M a (randomized) algorithm that takes as input a data set.

Definition 2.1 (**Dwork, Kenthapadi, et al., 2006**; **Dwork, McSherry, et al., 2006**). A (randomized) algorithm M gives (ε, δ)-differential privacy if for any pair of neighboring data sets S, S' and any event E,

$$\mathbb{P}(M(S) \in E) \leqslant \mathrm{e}^{\varepsilon}\mathbb{P}(M(S') \in E) + \delta.$$

Для досягнення конфіденційності алгоритм обов'язково рандомізований, тоді як два набори даних є детермінованими. Це визначення конфіденційності гарантує, що, виходячи з результатів роботи алгоритму, супротивник має обмежену можливість ідентифікувати присутність чи відсутність будь-якої особи, незалежно від того, чи хтось вибирає чи відмовляється від даних.

По суті, супротивник намагається розрізнити два розподіли ймовірностей і використовуючи одну нічию. У світі цього спостереження природно інтерпретувати те, що робить супротивник, як перевірку двох простих гіпотез.

Наскільки відомо, зв’язок між різницею конфіденційності та тестуванням гіпотез вперше зазначений у роботі Фредріксона, а пізніше був розроблений в інших роботах. Інтуїтивно, конфіденційність добре гарантована, якщо проблема перевірки гіпотез важка. Виходячи з цієї інтуїції, визначення f-DP по суті використовує коефіцієнт вірогідності найгіршого

випадку розподілів і для вимірювання твердості тестування двох простих гіпотез.

Чи існує більш інформативний показник твердості? Автори пропонують використовувати компроміс між помилкою типу I (ймовірність помилкового відхилення істини) та помилкою типу II (ймовірність помилкового прийняття істини) замість кількох параметрів конфіденційності в f-DP на основі дивергенції. Визначимо функцію компромісу наступним чином:

$$T(P, Q) : [0, 1] \mapsto [0, 1]$$
$$\alpha \mapsto \inf_{\phi} \{1 - \mathbb{E}_Q[\phi] : \mathbb{E}_P[\phi] \leqslant \alpha\}.$$

Вище, і є помилки типу I та II типу правила відхилення, відповідно. У написанні цього визначення сказано, що це мінімальна помилка типу II серед усіх тестів на рівні значущості. Зверніть увагу, що мінімуму можна досягти, провівши тест коефіцієнта ймовірності, згідно з лемою Неймана – Пірсона. Як само собою зрозуміло, чим більша функція компромісу, тим складнішою є проблема перевірки гіпотез (отже, більше конфіденційності). Це мотивує наступне визначення конфіденційності.

(Рандомізований) алгоритм є -диференціально приватним, якщо

$$T\big(M(S), M(S')\big) \geqslant f$$

для всіх сусідніх наборів даних.

У цьому визначенні обидві функції є функціями, які приймають за вхідні дані, а нерівність виконується для всіх ідентифікуючих та пов'язаних з ними розподілів. Це визначення конфіденційності легко інтерпретувати через невід’ємний зв’язок із проблемою перевірки гіпотез. Шляхом адаптації результату через, f-DP є особливим визначенням f-DP в тому сенсі, що алгоритм є f-DP тоді і тільки тоді, коли це f-DP дорівнює наступному:

$$f_{\varepsilon,\delta}(\alpha) = \max\left\{0, 1-\delta-\mathrm{e}^{\varepsilon}\alpha, \mathrm{e}^{-\varepsilon}(1-\delta-\alpha)\right\}.$$

Моделі глибокого навчання навчаються, використовуючи склад багатьох оновлень SGD. Взагалі кажучи, склад стосується послідовності аналізів на одному і тому ж наборі даних, де кожен аналіз визначається результатами попередніх аналізів. Основною проблемою, з якою стикається кожне визначення конфіденційності, є визначення того, як загальна гарантія конфіденційності погіршується за складом. Формально, дозволяючи бути першим алгоритмом, а бути другим, ми визначаємо їх алгоритм композиції як. Грубо кажучи, композиція полягає в тому, щоб «випускати всю інформацію, яку вивчають алгоритми». Зокрема, другий алгоритм може взяти як вхідний сигнал вихідний сигнал на додаток до набору даних. Взагалі, склад більше двох алгоритмів слід рекурсивно.

На високому рівні цей результат конвергенції до ДПГ призводить ДПГ до центральної точки сімейства гарантій f-DP, маючи на увазі, що ДПГ дорівнює f-DP, як звичайні випадкові величини до загальних випадкових величин. Крім того, цей результат служить ефективним інструментом наближення для апроксимації гарантій конфіденційності алгоритмів композиції. Навпаки, втрата конфіденційності не може бути відстежена без втрат за складом у рамках f-DP.

Основною з властивостей f-DP є властивість субдискретизації. У тренувальних нейронних мережах градієнт на кожній ітерації обчислюється з міні-партії, яка відібрана на вибірці з навчальних прикладів. Інтуїтивно зрозуміло, що алгоритм, застосований до під вибірки, дає більші гарантії конфіденційності, ніж коли застосовується до повної вибірки. Подивившись уважно, це посилення конфіденційності пов’язано з тим, що людина користується ідеальною приватністю. Отже, конкретним і нагальним питанням є чітка характеристика того, наскільки конфіденційність посилюється піддискретизацією в рамках f-DP.

Розглянемо таку схему вибірки: для кожної людини в наборі даних включимо її або її дані в підвибірку незалежно з деякою імовірністю p, яку іноді називають під вибіркою Пуассона. Підвибірка має випадковий розмір. Враховуючи будь-який алгоритм, позначаємо його алгоритмом із вибіркою.

Шокрі та ін. 2017 рік продемонстрували, що цей метод не надто ефективний на великих наборах даних. Зміна швидкості навчання, схеми оптимізації та ініціалізації відбувалася часто, але все ще не дуже добре. Але можливі покращення при ініціалізації однорідними входами замість випадкових чисел і збереженні кількості ітерацій приблизно до 100. Ініціалізація за допомогою шумних входів призвела до більш галасливих виходів, а при ініціалізації з усіма нулями вихід вийшов менш шумним. Це можна розглядати як починаючи з чистого полотна або галасливого полотна, щоб зробити картину.

Рисунок 1.5 Приклад атаки інверсії

Основною проблемою цього типу інверсії моделі було те, що отримане зображення було дуже галасливим. Для боротьби з цими галасливими результатами була використана техніка DeepDream, яка описана у розділі 2 для оптимізації в різних масштабах. Спочатку починаємо з оптимізації зображення з просторовою роздільною здатністю 4x4, яка фокусує реконструкцію на оптимізації великомасштабних функцій, тобто які макрозміни призведуть до того, що модель подумає, що це птах. Як тільки оптимізація завершиться, ми зробимо вибірку до 8x8 та оптимізуємо для

отримання деталей з більш високою роздільною здатністю та заповнюємо їх зображенням. Це повторюється до отримання остаточної роздільної здатності зображення 32x32.

Висновок за розділом 1

Отже, у ході роботи був визначений основний клас задач використання згорткових нейронних мереж, а саме клас задач, пов'язаний з комп'ютерним баченням. У ході роботи розглядалося наступне:

1. Тип даних - зображення.
2. Типи архітектур нейронних мереж - згорткові нейронні мережі.
3. Основні метрики якості - точність, mAP.

Були проаналізовані проблеми поточного процесу тренування згорткових нейронних мереж. Коротко розглянуті основні проблеми безпеки нейронних мереж та такі види атак, як:

1. Змагальні атаки.
2. Атака інверсійної моделі, яка більш детально розглядається в розділі 2.

Була розглянута проблема диференційної приватності та проаналізовані методи, які лежать в основі f-DP підходу. Зокрема була проаналізована проблематика диференційної приватності у контексті глибокого навчання.

Таким чином, перші три поставлені завдання дипломної роботи були виконані.

РОЗДІЛ 2
АТАКА ІНВЕРСІЙНОЇ МОДЕЛІ

2.1 Вступ

Моделі глибокого навчання часто навчаються на наборах даних, що містять конфіденційну інформацію, таку як торгові операції, особисті контакти та медичні записи. Тому дедалі важливіший напрямок роботи намагався навчити нейромережі, що підпадають під обмеження конфіденційності, що визначаються диференційованою конфіденційністю або її розслабленням на основі розбіжностей. Однак ці визначення конфіденційності мають слабкі місця в обробці певних важливих примітивів (композиція та субдискретизація), тим самим даючи вільний або складний аналіз конфіденційності навчання нейронних мереж. У цій статті ми розглядаємо нещодавно запропоноване визначення конфіденційності, яке називають f-диференційованою конфіденційністю для вдосконаленого аналізу конфіденційності навчальних нейронних мереж. Використовуючи привабливі властивості f-диференціальної конфіденційності при роботі з композицією та субдискретизацією, ця стаття отримує аналітично виправлювані вирази для гарантій конфіденційності як стохастичного градієнтного спуску, так і Адама, використовуваного при навчанні глибоких нейронних мереж, без необхідності розробляти складні методи, як це робилося. Наші результати демонструють, що диференціальна система конфіденційності дозволяє провести новий аналіз конфіденційності, який покращує попередній аналіз, що, в свою чергу, пропонує налаштування певних параметрів нейронних мереж для кращої точності прогнозування без порушення бюджету конфіденційності. Ці теоретично отримані вдосконалення підтверджуються нашими експериментами з ряду завдань з класифікації зображень, класифікації тексту та систем рекомендацій. Код Python для розрахунку вартості конфіденційності

для цих експериментів є загальнодоступним у бібліотеці конфіденційності TensorFlow.

В епоху великих даних надзвичайно важливо і навіть терміново розробити алгоритми, які зберігають конфіденційність конфіденційних індивідуальних даних, зберігаючи при цьому високу корисність. Однак ці дві цілі, як правило, конкурують між собою в тому сенсі, що гарантії конфіденційності неминуче призводять до втрати якості значень обраної метрики. У даній роботі був проведений пошук оптимального балансу між якістю моделі та збереження приватності.

Однією з найпоширеніших модельних інверсійних атак є атака на основі градієнта. Основна ідея цієї атаки полягає у введенні випадкового шуму через модель, яка атакується (цільова модель), і зворотному розповсюдженні втрат від цього випадкового шуму, але замість зміни ваги ми змінюємо вхідне зображення. Тут, замість оптимізації ваг для мінімізації втрат, ми оптимізуємо зображення, щоб мінімізувати втрати, тобто генеруємо зображення, яке модель вважає найбільш вірогідним зразком класу. Це той самий алгоритм, що й атака з проектованим градієнтним спуском (PGD), знайдений у літературі про змагальні атаки. Різниця між PGD та інверсією моделі полягає в тому, що в PGD вхід є реальним зображенням, а в інверсії моделі вхід складається з випадкових чисел.

Алгоритми машинного навчання (ML) все частіше використовуються в програмах, що чутливі до конфіденційності, таких як прогнозування вибору способу життя, постановка медичних діагнозів та розпізнавання обличчя. У модельній атаці інверсії, нещодавно представленій у тематичному дослідженні лінійних класифікаторів в персоналізованій медицині Фредріксон зловживає змагальним доступом до моделі ML, щоб дізнатись конфіденційну геномну інформацію про людей. Однак, чи застосовуються атаки інверсії моделі до налаштувань поза даними прикладами, невідомо. У ході роботи було протестовано клас модельної атаки інверсії, яка використовує значення впевненості, виявлені разом із прогнозами. Дані атаки застосовуються в різних

умовах, і два вивчаються поглиблено у даній роботі: дерева рішень для опитувань способу життя, що використовуються в системах машинного навчання як послуги, і нейронні мережі для розпізнавання обличчя. В обох випадках значення достовірності виявляються тим, хто має можливість робити прогнозні запити до моделей. У дипломній роботі експериментально демонструються атаки, за допомогою яких можна оцінити, як відновити впізнавані зображення облич людей, давши лише їх ім'я та доступ до моделі ML. Також було досліджено алгоритм навчання дерева прийняття рішень, що відповідає конфіденційності, який є простим варіантом навчання CART, а також виявляючи лише округлі значення довіри. Гіпотеза на перевірку полягає в тому, чи можна уникнути такого роду атак інверсійної моделі із незначним зниженням корисності.

Обчислювальні системи все частіше включають алгоритми машинного навчання (ML) для того, щоб передбачати вибір способу життя, медичні діагнози, розпізнавання обличчя тощо. Потреба у простому кнопковому ML навіть підштовхнула ряд компаній до створення хмарних систем ML-as-a-service, де клієнти можуть завантажувати набори даних, навчати класифікатори або моделі регресії, а потім отримувати доступ для виконання запитів прогнозування використання натренованої моделі - просто використовуючи веб-додатки та використовуючи простий інтерфейс HTTP. Функції, які використовуються цими моделями та запитуються через API для прогнозування, часто представляють конфіденційну інформацію. При розпізнаванні обличчя особливостями є окремі пікселі зображення обличчя людини.

В опитуваннях про побутові звички людей функції можуть містити конфіденційну інформацію, таку як сексуальні звички респондентів. В контексті цих служб явною загрозою є те, що провайдери можуть бути поганими розпорядниками конфіденційних даних, дозволяючи навчальним даним або журналам запитів стати жертвою інсайдерських атак або викриття через компрометацію системи.

Ряд робіт зосереджено на атаках, які виникають внаслідок доступу до (навіть анонімних) даних. Можливо, більш тонке занепокоєння полягає в тому, що здатність робити запити прогнозування може дозволити клієнтам-суперникам резервувати конфіденційні дані. Недавня робота Фредріксона та співавт. в контексті конфіденційності геномів показує модель інверсії атаки, яка може використовувати чорний ящик для прогнозування моделей для оцінки аспектів чийогось генотипу. Такі атаки можуть спричинити вилучення клієнтами чутливих даних з використовуваних сервісів. У роботі Фредріксона була лише оцінка одного з можливих налаштувань даної атаки, у даній роботі був проведений більш ширший аналіз і дослідження можливих налаштувань інверсії моделей.

У цій статті автори пропонують моделі глибокого навчання з гарантіями конфіденційності в рамках f-диференціальної конфіденційності, яка нещодавно була запропонована як об'єднуюче узагальнення кількох попередніх визначень конфіденційності. Автори тренують свої приватні нейронні мережі для виконання кількох завдань, що включають різні типи даних, включаючи зображення, тексти та числові дані. Завдяки універсальності f-диференціальної системи конфіденційності експериментальні результати демонструють, що цей підхід до приватного глибокого навчання перевершує існуючі підходи з точки зору компромісу між приватністю та корисністю.

У цій роботі ми досліджуються комерційні API ML-as-a-service. У поточних налаштуваннях атака Фредріксона не є ефективною.

Тому у ході роботі були використані більш детальні атаки, які виводять чутливі функції, що використовуються як вхідні дані для моделей дерев рішень, а також атаки, що відновлюють зображення з доступу API до служб розпізнавання облич. Ключовим фактором, є те, що можна створювати алгоритми атак, які використовують значення впевненості, виставлені API.

Один із прикладів атак розпізнавання обличчя зображений на малюнку 1.2: зловмисник може створити зображення людини, яке можна впізнати,

надавши лише API-доступ до системи розпізнавання обличчя та ім’я людини, обличчя якої вона розпізнає.

2.2 ML API та інверсія моделей

Основні розглянуті налаштування атаки наступні: налаштування чорного ящика, налаштування білого ящика. У налаштуваннях чорного ящика суперницький клієнт може робити запити прогнозування щодо моделі, але фактично не завантажувати опис моделі. У налаштуванні білого ящику клієнтам дозволяється завантажувати опис моделі, а також доступ до процесу тренування. Нове покоління систем сервісів ML-as-a-service у тому числі загального призначення, таких як BigML та Microsoft Azure Learning - дозволяє власникам даних визначати, чи повинні API дозволяти доступ до білих або чорних ящиків до своїх моделей.

Розглянемо модель, що визначає функцію f, яка приймає вхід вектор ознак x1,. . . , xd для деякої розмірності ознаки d і виводить прогноз y = f (x1, ..., xd). У моделі інверсійної атаки Fredrikson et al. суперницький клієнт використовує чорний ящик для доступу до f, щоб вивести чутливу функцію, скажімо, x1, враховуючи деякі знання про інші особливості та залежне значення y, статистику помилок щодо моделі та граничні пріоритети для окремих змінних. Їх алгоритм - це максимум апостериорний (MAP) оцінювач, який вибирає значення для x1, максимізує ймовірність спостереження відомих значень (за деяких, здавалося б, обґрунтованих припущень про незалежність). Однак для цього потрібні обчислення f (x1, ..., xd) для всіх можливих значень x1 (та будь-яких інших невідомих ознак). Це обмежує його застосовність до налаштувань, де x1 приймає лише обмежений набір можливих значень.

Важливим етапом при зміні налаштувань є - оцінка оцінювача MAP у новому контексті. Було проведено тематичне дослідження, яке показує, що метрика забезпечує лише обмежену ефективність в оцінці чутливих особливостей в моделях дерев рішень, під час вирішення задач опитування, які зараз розміщені в галереї моделей BigML.

Зокрема, показник хибнопозитивних надто високий: експерименти показують, що Fredrikson et al. алгоритм міг би зробити неправильний висновок, наприклад, що людина (про яку відомо, що вона є в навчальному наборі) переглядала порнографічні відео за останній рік майже 60% часу згідно опитування. Таким чином можна припустити, що інверсія сама по собі не є значним ризиком, але насправді існують атаки, які можуть значно покращити ефективність інверсії. Наприклад, атаки на дерево білих ящиків рішень.

Досліджуючи фактичні дані, доступні через API служб BigML, можна побачити, що описи моделей включають більше інформації, ніж використано для атаки чорної скриньки. Зокрема, вони надають кількість випадків із навчального набору, які відповідають кожному шляху в дереві рішень. Поділ на загальну кількість випадків дає впевненість у класифікації. Хоча апріорі ця додаткова інформація може здатися нешкідливою, так чи інакше дана інформація насправді може бути використана.

У даній роботі розглядався новий оцінювач MAP, який використовує інформацію довіри в налаштуваннях білого поля для виведення конфіденційної інформації без помилкових спрацьовувань при тестуванні на основі двох різних моделей дерева рішень BigML. Ця висока точність дотримується для цільових суб'єктів, які, як відомо, містяться в даних тренувань, тоді як точність оцінювача значно гірша для тих, хто не входить до набору навчальних даних. Це демонструє, що публікація цих моделей створює ризик конфіденційності для тих, хто сприяє навчальним даним.

Таким чином, вони не поширюються на налаштування, де функції мають експоненційно великі домени, або коли ми хочемо інвертувати велику кількість функцій з малих доменів.

Розглянемо вилучення граней з нейронних мереж. Прикладом хитрого налаштування даних великих розмірів із великим доменом є розпізнавання обличчя: функції - вектори даних пікселів із плаваючою точкою. Теоретично рішення цієї проблеми інверсії великого домену може дозволити, наприклад,

зловмисникові використовувати API розпізнавання обличчя для відновлення зображення людини, названої лише своїм іменем (мітка класу). Звичайно, це здається неможливим у налаштуваннях чорного ящика, якщо API повертає відповіді на запити, які є лише міткою класу. Перевіряючи API розпізнавання обличчя, виявляється, що загальноприйнятим є надання мір довіри з плаваючою комою разом із міткою класу (ім'я особи). Це дозволяє нам розробляти атаки, які ставлять завдання інверсії як проблему оптимізації: знайти вхідні дані, які максимізують повернену впевненість, за умови, що класифікація також відповідає цілі. У роботі розглядається і використовується алгоритм вирішення цієї проблеми, який використовує градієнтний спуск разом із модифікаціями, характерними для цього домену. Це ефективно, незважаючи на експоненціально великий простір пошуку: у багатьох випадках реконструкція завершується лише за 1,4 секунди, а для більш складних моделей у білій скриньці - за 10–20 хвилин.

Застосовано цю атаку до ряду типових алгоритмів розпізнавання обличчя в нейронних мережах, включаючи класифікатор softmax, багатошаровий персептрон та багатоярусний автоенкодер. Як видно на малюнку 1.2, відновлене зображення не є ідеальним. Для кількісної оцінки ефективності було проведено експерименти, використовуючи Amazon's Mechanical Turk, щоб побачити, чи можуть люди використовувати використане відновлене зображення, щоб правильно вибрати цільову людину з ряду.

Кваліфіковані люди можуть правильно зробити це для класифікатора softmax з точністю майже до 95% (середня продуктивність у всіх працівників перевищує 80%) на відомих датасетах. Результати гірші для інших двох алгоритмів, але все одно перевершують випадкові вгадування.

Також досліджуються пов'язані напади в режимі розпізнавання обличчя, наприклад, використовуючи інверсію моделі, щоб допомогти ідентифікувати людину, яка має розмите зображення її обличчя.

2.3 Інверсія у контексті машинного навчання

Модель ML - це просто детермінована функція f: R → Y від d ознак до набору d відповідей Y. Коли Y є кінцевим набором, наприклад іменами людей у разі розпізнавання обличчя, називаються f класифікатором, а елементи Y класами. Якщо замість Y = R, то f - це модель регресії або просто регресія. Багато класифікаторів, і особливо розпізнавання облич спочатку обчислюють одну або кілька регресій на векторі ознак для кожного класу, з відповідними результатами, що представляють оцінки ймовірності того, що вектор ознак повинен бути пов'язаний з класом. Ці результати часто називають конфіденційними, і класифікація отримується шляхом вибору мітки класу для регресії з найбільшою впевненістю. Більш формально, f у цих випадках визначається, як склад двох функцій. Перша - це функція f ˜: Rd → [0, 1] m, де m - параметр, що визначає кількість конфіденційності.

У даному випадку m = | Y | - 1, тобто на одну менше, ніж кількість міток класів. Друга функція - це функція вибору t: [0, 1] m → Y, яка, наприклад, коли m = 1, може видавати одну мітку, якщо вхід вище 0,5, а іншу мітку - в іншому випадку. Коли m > 1, t може вивести мітку, пов'язана з якою довіра найбільша.

Зрештою f (x) = t (f˜ (x)). Серед API для таких моделей поширено, що запити класифікації повертають як f (x), так і f ˜ (x). Це забезпечує зворотний зв’язок щодо впевненості моделі в її класифікації. Якщо не зазначено інше, у цій роботі будемо вважатимемо, що обидва вони повертаються із запиту до f.

Отже, основна ідея виглядає наступним чином: генерується модель f за допомогою деякого (рандомізованого) тренувального алгоритму тренувань. В якості вхідних позначених навчальних даних db береться послідовність (d + 1) розмірних векторів (x, y) Rd × Y, де x = x1,. . . , xd - набір ознак, а y - мітка.

Таку пара є екземпляр, і, як правило, припускаємо, що екземпляри витягуються незалежно від деякого попереднього розподілу, спільного щодо

функцій та відповідей. Результатом роботи є модель f та деяка допоміжна інформація.

Приклади допоміжної інформації можуть включати статистику помилок та / або незначні пріоритети для даних навчання. Системи, що включають моделі f, будуть робити це через чітко визначені інтерфейси програмування програм (API). Недавня тенденція до систем обслуговування ML-as-a-service є прикладом цієї моделі, коли користувачі завантажують свої навчальні дані db і згодом використовують такий API для запиту моделі, навченої службою. API зазвичай надається через HTTP(S). В даний час існує низка таких служб, серед яких Microsoft Machine Learning, API прогнозування Google, BigML, Wise.io, які зосереджені на аналітиці.

Інші зосереджуються на ML для виявлення та розпізнавання обличчя. Деякі з цих послуг мають торгові площі, на яких користувачі можуть робити моделі або набори даних доступними для інших користувачів. Модель може бути білою скринькою, тобто кожен може завантажити опис f, придатної для її запуску локально, або чорною скринькою, тобто не можна завантажувати модель, а можна лише робити прогнозні запити щодо неї.

У даній роботі увага була зосереджена на налаштуваннях, в яких суперницький клієнт прагне зловживати доступом до API моделі ML. Передбачається, що супротивник має будь-яку інформацію, яку виставляє API. У налаштуваннях білого ящика це означає доступ до завантаження моделі f. У налаштуваннях чорної скриньки зловмисник може лише робити запити прогнозування щодо векторів функцій на вибір противника. Ці запити можуть бути адаптивними, однак, що означає, що запитувані функції можуть бути функцією попередньо отриманих прогнозів. В обох налаштуваннях супротивник отримує допоміжну інформацію на виході шляхом навчання, що відображає той факт, що в API ML ці дані в даний час найчастіше оприлюднюються.

У роботі також оцінювався контекст, в яких супротивник не має доступу до даних навчання db, а також можливість взяти будь-яку інформацію

про модель. Хоча в деяких випадках дані про навчання є загальнодоступними, розглядалися параметри з ризиком конфіденційності, де db можуть бути медичними даними, опитуваннями про побутові звички або зображеннями для розпізнавання обличчя. Таким чином, єдине розуміння супротивника цих даних є непрямим за допомогою API ML. Однак, не було розглянуто у даній роботі зловмисних постачальників послуг і не розглядаються суперницьких клієнти, які можуть скомпрометувати послугу, наприклад, оминувши якось аутентифікацію або іншим чином використовуючи помилку в програмному забезпеченні сервера. Такі загрози важливі для безпеки послуг ML, але вже є відомими проблемами.

Фредріксон та ін. розглядали модель лінійної регресії f, яка передбачала реальну оцінку запропонованої початкової дози препарату Варфарин, використовуючи вектор ознак, що складається з демографічної інформації про пацієнта, історії хвороби та генетичних маркерів. Чутливим атрибутом вважали генетичний маркер. Припустимо, що простота була першою ознакою x1. Вони досліджували інверсію моделі, коли зловмисник, отримуючи доступ до білого ящика до f та допоміжної інформаційної сторони (x, y) = (x2, ..., xt, y) для екземпляру пацієнта (x, y), намагається зробити висновок про генетичний маркер пацієнта x1.

Припускаємо, що aux дає емпірично розраховане стандартне відхилення σ для моделі гауссової помилки err та граничних пріоритетів p = (p1, ..., pt). Граничний попередній pi обчислюється шляхом першого розподілу дійсного рядка на непересічні сегменти (діапазони значень), а потім дозволяючи pi (v) для кожного сегмента v бути кількістю випадків, коли xi падає у v над усіма x у db, поділених на кількість навчальних векторів | db |.

Алгоритм просто заповнює цільовий вектор ознак кожним із можливих значень для x1, а потім обчислює зважену оцінку ймовірності, що це правильне значення.

Як зазначалося в їх роботі, цей алгоритм виробляє найменш упереджений максимум апостеріорної (MAP) оцінки для x1 з урахуванням

наявної інформації. Таким чином, це мінімізує рівень непередбачуваного опонента. Вони проаналізували його ефективність лише у випадку дозування варфарину, показавши, що наведений вище алгоритм IM досягає точності прогнозування генетичного маркера, наближеного до лінійної моделі, навченої безпосередньо з вихідного набору даних. Неважко помітити, що їхній алгоритм насправді є чорним ящиком і насправді є агностичним щодо способу роботи f. Це означає, що він потенційно застосовний в інших параметрах, де f - це не модель лінійної регресії, а якийсь інший алгоритм.

Очевидне розширення обробляє більший набір невідомих функцій, просто змінивши основний цикл на перебір усіх можливих комбінацій значень для невідомих функцій. Звичайно, це має сенс лише для поєднань іменних цінностей, а не справді реальних. Однак алгоритм, запропонований Fredrikson et al. має різні обмеження. Найбільш очевидно, що його не можна використовувати, коли невідомі особливості охоплюють нерозбірливо великий набір. Одним із прикладів такої установки є розпізнавання обличчя, де простір ознак величезний: у прикладах розпізнавання облич, розглянутих пізніше d ≈ 10, 304, для кожного об’єкта дійсне число в [0, 1]. Навіть якщо хтось хотів лише зробити висновок про частину функцій, це обчислювально неможливо. Більш тонке питання, полягає в тому, що навіть коли воно ефективно, воно може бути неефективним. Виявляється, можна проводити атаки, які долають обидві вищезазначені проблеми, надаючи інверсійні атаки, які використовують інформацію про довіру, розкриту ML API.

2.4 Інверсія атаки для дерев

Розглянемо інверсійні атаки на дерева рішень. Цей тип моделі використовується на широкому діапазоні даних і часто надає перевагу тому, що правила, структуровані за деревами, легкі для розуміння користувачами. У літературі є два типи дерев рішень: класифікація (де змінна класу є дискретною) та регресія (де змінна класу неперервна). Дерева класифікації розглядалися в даній роботі найбільше.

Модель дерева рішень рекурсивно розділяє простір об'єктів на непересічні області R1,. . . , Rm. Прогнози робляться для екземпляру (x, y) шляхом знаходження області, що містить x, і повернення найбільш вірогідного значення y, яке спостерігається в даних навчання в цій області.

Характеризуємо дерева математично наступним чином:

$$m\ f(x) = wi\varphi i(x),$$

де $\varphi i(x) \in \{0, 1\}$ $i = 1$, де кожна основна функція φі є показником для Ri, а wi відповідає найпоширенішій реакції, яка спостерігається в навчальному наборі в межах Ri. Для кожного шляху крізь дерево існує рівно одна базова функція, і заданий x "активує" лише один φі (і, таким чином, пройде лише один шлях через дерево), оскільки основа функції розділити функцію простору. Дерева рішень можна розширити, для зміни довірчих мір для класифікацій, змінивши форму коефіцієнтів wi. Зазвичай це досягається шляхом встановлення wi на вектор, що відповідає розподілу міток класів у даній області, як це спостерігається в навчальному наборі.

API дерев рішень. Кілька веб-API надають кінцевим користувачам процедури навчання та запитів для дерев рішень, включаючи машинне навчання Microsoft, Wise.io та BigML. Користувачі можуть завантажувати свої набори даних до цих служб, навчати дерево рішень для прогнозування вибраних змінних, а потім робити отримане дерево доступним для інших для безкоштовного використання або за плату за запит. Як запущений приклад використовується API, виставлений BigML, оскільки в даний час він має найбільший ринок навчених моделей і зосереджений на деревах рішень. Результати також переносяться на служби з подібним API. BigML дозволяє користувачам публікувати дерева в режимі чорної або білої скриньок. У режимі чорного ящика іншим користувачам дозволяється лише запитувати дерево рішень щодо прогнозів за допомогою REST API. У режимі білого поля користувачам дозволяється бачити внутрішню структуру дерева, а також завантажувати його представлення JSON для локального використання. Таким

чином, обсяг доступної інформації про навчальний набір варіюється між двома режимами. В обох налаштуваннях суперник має доступ до граничних пріоритетів для кожної ознаки навчального набору, на додаток до матриці C, для якої Ci, j дає кількість навчальних випадків з y = i, для яких модель передбачала мітку j.

У налаштуваннях білого ящику зловмисник також має доступ до числа ni кількості екземплярів навчального набору, що відповідає шляху φi у дереві. Це дозволяє обчислити достовірність певного прогнозу.

2.5 Проблема інверсії

Нехай дерево f (x) = mi = 1 wiφi (x) і нехай (x, y) буде цільовим екземпляром, який буде або з навчальних даних db, або ні. Для простоти припустимо, що існує одна чутлива функція, перша, що робить цільову функцію встановленою в даному випадку T = {1}.

Що стосується чорного ящика, звернемось до загального алгоритму та адаптуємо його до цього параметра. Основна відмінність від попередньої роботи полягає в тому, що моделі дерев рішень, які розглядаються, дають дискретні результати, а інформація про модель помилки відрізняється, будучи матрицею сплутаності на відміну від стандартного відхилення Гауса. Тоді для тут ми використовуємо матрицю сплутаності C і визначаємо err (y, y ') ∝ Pr [f (x) = y' | y - справжня мітка].

Біла скринька MI. В налаштуваннях білої скриньки ми не тільки припускаємо, що зловмисник знає кожен φi, а й кількість навчальних зразків n Wi, які відповідають m до φi. З цього він також знає N = i = 1 ni, загальну кількість зразків у навчальному наборі.

Відомі значення xK індукують набір шляхів S = {si} 1≤i≤m: '' 'S = {(φi, ni) | ∃ x ∈ Rd. xK = xK ∧ φi (x)}. Кожен шлях відповідає базовій функції φi та кількості вибірок ni. Ми дозволимо pi позначати ni / N, і зауважимо, що кожен pi дає нам деяку інформацію про спільний розподіл щодо функцій, що використовуються для побудови навчального набору. Далі ми пишемо si як

скорочення для події, коли рядок, намальований із спільного попереднього, перетинає шлях si, тобто Pr [si] відповідає ймовірності складання рядка із спільного попереднього, який перетинає si. pi - це емпірична оцінка для цієї величини, отримана з розіграшів, використаних для створення навчального набору. Нагадаємо, що основні функції розділяють простір ознак, тому x проходить точно один із шляхів у S. Нижче у роботі позначено конкретне значення для першого об'єкта через v, а конкретне значення для інших d - 1 об'єктів як vK. Ми будемо зловживати позначеннями і запишемо φi (v) як скорочення " 'для функції індикатора I (∃ x ∈ Rd. X1 = v ∧ φi (x)). Наступний оцінювач характеризує ймовірність того, що x1 = v, враховуючи, що x проходить один із шляхів s1,. . . , sm та xK = vK:

$$\begin{aligned}
&\Pr[\mathbf{x}_1 = v \mid (s_1 \vee \cdots \vee s_m) \wedge \mathbf{x}_K = \mathbf{v}_K] \\
&\quad \propto \sum_{i=1}^{m} \frac{p_i \phi_i(v) \cdot \Pr[\mathbf{x}_K = \mathbf{v}_K] \cdot \Pr[\mathbf{x}_1 = v]}{\sum_{j=1}^{m} p_j \phi_j(v)} \\
&\quad \propto \frac{1}{\sum_{j=1}^{m} p_j \phi_j(v)} \sum_{1 \le i \le m} p_i \phi_i(v) \cdot \Pr[\mathbf{x}_1 = v] \qquad (1)
\end{aligned}$$

Потім противник виводить значення для v, яке максимізує (1) як здогадка для x1. Як і Fredrikson et al. оцінювача, він повертає прогноз MAP з урахуванням додаткової інформації про кількість

У попередньому аналізі ми припустили, що зловмисник знав усі x, крім x1. Оцінювач можна поширити на загальний випадок, коли зловмисник не знає x2,. . . , xl (отже, xK = {l + 1, ..., d − 1}) шляхом підсумовування (1) за невідомими змінними.

2.6 Експерименти

У даній роботі були застововані атаки з оцінювачем описаних налаштувань із чорною скринькою та білою скринькою для дерев рішень, навчених на двох наборах даних: опитування FiveThirtyEight "Як американці люблять свій стейк" та підгрупу загального соціального опитування (GSS), що фокусується на відповідях, пов'язаних із щастям у шлюбі.

Кожен набір даних містить рядки, що відповідають окремим особам, з атрибутами, що відповідають відповідям опитування. Ці набори даних містять принаймні одну чутливу функцію і використовувались для отримання моделей, доступних на ринку BigML. Крім того, вихідні дані є загальнодоступними, що робить їх відповідними сурогатами для нашого дослідження - без вихідних даних ми не можемо оцінити ефективність наших атак.

У травні 2014 року Уолт Хікі написав статтю для розділу DataLab FiveThirtyEight, в якій робився спроба статистичного аналізу зв'язку між уподобаннями людей щодо приготування стейків та їхньою схильністю до ризикованої поведінки. Для підтримки аналізу FiveThirtyEight замовив опитування серед 553 осіб з SurveyMonkey, яке зібрало відповіді на запитання, такі як: «Ви коли-небудь палите сигарети?», «Ви коли-небудь обманювали свою другу половинку?» І, звичайно, «Як вам подобається ваш стейк, приготовлений чи ні? ". Також були зібрані демографічні характеристики, такі як вік, стать, доходи домогосподарств, освіта та регіон перепису населення.

У ході роботи були відкинуті рядки, які не містили відповідей на питання невірності чи питання приготування стейка, в результаті чого для експериментів з інверсією було отримано 332 рядки. Була використана інверсія моделі на дереві рішень, вивченому з цього набору даних, щоб зробити висновок, чи відповідав кожен учасник "Так" на питання про невірність. Дослідження подружнього щастя GSS. Загальне соціальне опитування (GSS) збирає детальну інформацію про демографічні показники, інтереси та ставлення жителів США.

У роботі була проаналізована і розглянута підмножина даних GSS, створена Джозефом Прайсом, для вивчення різних соціальних ефектів порнографії. Ця підмножина відповідає 51020 особам та 11 змінним, включаючи базову демографічну інформацію та відповіді на запитання, наприклад: "Наскільки ти щасливий у своєму шлюбі?" та "Чи дивилися ви

фільми з рейтингом X за останній рік?". Пусті рядки були відкинуті, в результаті чого для експериментів з інверсією отримано 16 127 рядків.

Підсумок результатів. Для обох наборів даних було виявлено позитивні випадки чутливої змінної (тобто, відповідь "Так" на питання "Ви коли-небудь обманювали свого друга?" Або "Ви дивились фільми з рейтингом X за останній рік?") з високою точністю.

Основними висновками є:

Отримавши доступ до білої скриньки до дерев BigML, опублікованих іншими для цих наборів даних, ми можемо прогнозувати позитивні випадки з повною точністю, тобто відсутність помилкових спрацьовувань.

Особи, відповіді на які використовуються в даних тренінгу, мають значно вищий ризик, ніж особи, не включені в дані тренінгу. Результати суворі: за результатами опитування FiveThirtyEight інверсія білої скриньки дає в середньому 593 × поліпшену точність та 371 × покращений відклик. Подібні результати мають місце для опитування GSS.

Доступ у білу скриньку до дерев рішень збільшує переваги супротивника. На дереві BigML, навченому з використанням даних GSS, супротивник білих ящиків значно підвищує точність порівняно із супротивником чорних ящиків (на 158% більше) за незначне зменшення відкликання (на 32% менше). Іншими словами, противник білої скриньки здатний визначити трохи менше відповідей "так", ніж противник чорної скриньки, але з набагато кращою (тобто досконалою) точністю.

Результати показують, що випуск дерева рішень, навченого будь-яким із цих наборів даних, суттєво збільшує ризик витоку конфіденційних даних для осіб, які надали відповіді.

Дерева, опубліковані на BigML, використовували для навчання весь набір даних. Для оцінки ефекту від включення навчальних наборів ми тренували дерева локально, побудувавши 100 дерев із використанням параметрів за замовчуванням для випадкових вибіркових стратифікованих навчальних наборів, що складаються з 50% наявних даних. Ми завантажили

моделі та провели експерименти локально, щоб уникнути навантаження на сервери BigML. Ми використовували машину з 12 ядрами Ryzen 3900X, що працюють на частоті 2,5 ГГц, і 64 ГБ пам'яті. Для встановлення базової лінії для порівняння було використано три стратегії прогнозування, що відповідають можливостям випадкового суперника, базового суперника та ідеального суперника.

Випадковий супротивник не має доступу до жодної інформації, крім домену чутливого атрибута. Це відповідає зловмиснику, який не може отримати доступ до моделі і нічого не знає про попередника. Для обох наборів даних чутлива функція є двійковою, тому найкраща стратегія супротивника - перевернути чесну монету.

Суперник базової лінії має доступ лише до граничного розподілу ймовірностей для чутливого атрибута, а не до дерева або будь-якої іншої інформації про навчальний набір. Найкраща стратегія базового суперника полягає у тому, щоб завжди вгадувати найбільш вірогідне значення відповідно до граничного розподілу (тобто його режиму, "Ні" на даних).

Ідеальний супротивник має доступ до дерева рішень, навченого з оригінального набору даних для прогнозування чутливого атрибута, і враховуючи відомі особливості для окремої людини, використовує дерево для прогнозування.

Ця стратегія для ідеального супротивника є відповідною в середовищі, оскільки успадковує обмеження класу моделі: оскільки атака інвертує дерево рішень для прогнозування, ми не очікуємо, що ці прогнози будуть більш точними, ніж ті, що приймаються рішенням дерево, навчене прогнозувати чутливу характеристику в прямому напрямку, з тих же даних.

2.7 Продуктивність

Кількість часу, необхідного для запуску атак як на чорну скриньку, так і на білу скриньку, є незначною, а атака на білу скриньку є найдорожчою. Ця атака зайняла в середньому близько 6 мілісекунд для обох наборів даних,

причому найдорожчим компонентом є перерахування шляхів через досить велике (близько 200 шляхів) дерево. Кількість викликів, необхідних для запуску атаки "чорної скриньки", невелика: 4 для набору даних FiveThirtyEight (є 2 невідомі двійкові функції) і 2 для набору даних GSS.

Для наборів даних, подібних до цих, маючи мало відомих функцій із малих доменів, атаку чорної скриньки можливо здійснити віддалено. Обговорення. На малюнку 2.1 представлені повні результати наших експериментів.

	FiveThirtyEight			GSS		
algorithm	**acc.**	**prec.**	**rec.**	**acc.**	**prec.**	**rec.**
whitebox	86.4	100.0	21.1	80.3	100.0	0.7
blackbox	85.8	85.7	21.1	80.0	38.8	1.0
random	50.0	50.0	50.0	50.0	50.0	50.0
baseline	82.9	0.0	0.0	82.0	0.0	0.0
ideal	99.8	100.0	98.6	80.3	61.5	2.3

Рисунок 2.1 - Результати експериментів на деревах рішень

Найбільша різниця між точністю чорної скриньки та білої скриньки полягає в наступному: атака білої скриньки не дала помилкових спрацьовувань, тоді як чорна скринька дала близько 15% у FiveThirtyEight та близько 60% у GSS. Це показує, що підрахунок екземплярів на деревних шляхах дає корисну інформацію про спільний пріоритет, що дозволяє зловмиснику краще ідентифікувати негативні екземпляри чутливої функції. Вимірюючись точністю, обидві атаки значно перевершують стратегію випадкового вгадування, принаймні на 30%. Однак жодна атака не досягає більшого відкликання. Це пов’язано з перекосом попереднього розподілу за чутливим атрибутом (близько 80% «Ні» в обох випадках), що призводить до атаки на користь відповідей, які не сприяють більшому відкликанню.

Результатом цього стану є набагато вища точність, що підтримує більшу впевненість у позитивних прогнозах.

Обидва алгоритми вигідно порівнюються з точки зору точності на FiveThirtyEight (принаймні на 80% покращення) та відкликання (принаймні на 20% покращення), але лише трохи кращі з точки зору точності (на 3-5% покращення). Однак на GSS точність дещо нижча, тоді як точність і відкликання більші. Малюнок 5b показує результати як відсоток від ефективності ідеальної стратегії. Досягнення 100% ефективності ідеальної стратегії означає, що атака виконується так добре, як можна було б обгрунтовано очікувати. Атака на білу скриньку досягає цього для точності та наближається (в межах 15%) для точності.

Обидві атаки різко збільшують шанси зловмисника передбачити чутливий атрибут, на 70% точність і 20% відкликання. Це демонструє, що включення до навчального набору представляє значний ризик для конфіденційності відповідей цих осіб.

2.8 Розпізнавання обличчя

Моделі розпізнавання обличчя - це функції, які позначають зображення, що містить обличчя, ідентифікатором, що відповідає індивідууму, зображеному на зображенні. Ці моделі все частіше використовуються для широкого кола завдань, таких як автентифікація, ідентифікація суб’єкта в налаштуваннях безпеки та правоохоронних органів, доповнена реальність та організація фотографій (наприклад, Facebook “DeepFace”). Зростаюча кількість веб-API підтримує розпізнавання обличчя, наприклад, пропоновані Lambda Labs, Kairos та SkyBiometry. Ряд місцевих бібліотек також надають API для розпізнавання обличчя, такі як OpenCV та OpenBR.

Спільним для всіх цих API є можливість підготовки моделі за допомогою набору зображень, позначених іменами осіб, які в них з’являються, та можливість виконувати класифікацію, враховуючи деякі

попередньо навчені моделі. Зверніть увагу, що класифікацію може проводити більший набір осіб, включаючи тих, хто не має доступу до позначеного навчального набору; використання API таким чином було запропоновано, наприклад, у додатках доповненої реальності. У цьому підрозділі розглядаються дві атаки ІМ на моделі, які використовуються цими API для порушення конфіденційності предметів у навчальному наборі. Обидві атаки передбачають, що супротивник має доступ лише до навченої моделі, але не до жодних вихідних даних навчання.

1) Під час першої атаки припускаємо, що супротивник знає мітку, вироблену моделлю, тобто ім'я особи або унікальний ідентифікатор, і бажає створити зображення особи, пов'язаної з цією етикеткою (тобто жертви). Противник використовує ІМ, щоб «реконструювати» зображення обличчя жертви за ярликом. Ця атака порушує конфіденційність особи, яка готова надати зображення себе як навчальні дані, оскільки суперник може потенційно реконструювати зображення кожної людини в навчальному наборі. Противник «виграє» екземпляр цієї атаки, якщо, показавши набір зображень обличчя, включаючи жертву, він може ідентифікувати жертву. У подальшому тексті ми називаємо це атакою відновлення.

2) Під час другої атаки припускаємо що супротивник має зображення, що містить розмите обличчя, і хоче дізнатись особу відповідної особи. Противник використовує розмите зображення як побічну інформацію в серії атак ІМ, результатом яких є розмите зображення суб'єкта. Якщо припустити, що оригінальне зображення було розмитим для захисту анонімності, ця атака порушує конфіденційність особи на зображенні.

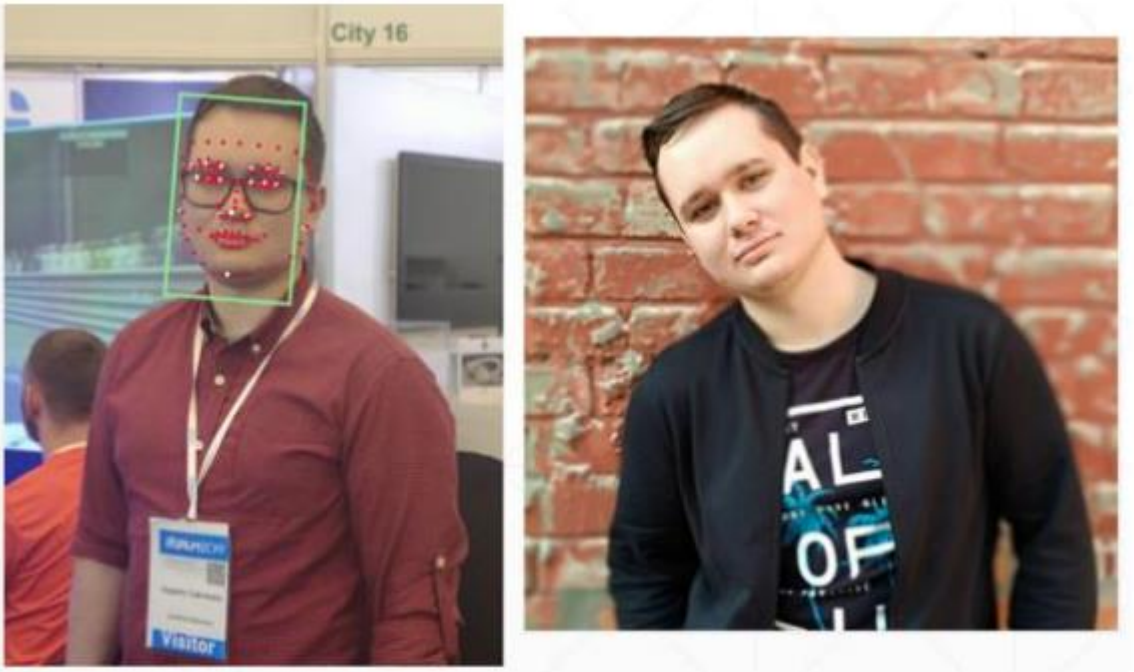

Рис. 2.2 Порівняння двох фотографій

Зображення було розмитим настільки, що модель більше не класифікує його правильно. Суперник може сподіватися на успіх лише в тому випадку, якщо людина була в навчальному наборі для моделі. Тут супротивник виграє, якщо він ідентифікує жертву за набором зображень обличчя, зроблених із тренувального набору, або якщо предмет розмитого зображення не був у навчальному наборі, а противник визначить, що зображення, створене в результаті атаки, не відповідає до будь-якого з облич. Це атакою розмивання. Ця атака будується безпосередньо на атаці відновлення. Через обмеження простору ми переносимо його детальне обговорення з експериментальними результатами до супровідного технічного звіту.

Існує багато запропонованих алгоритмів та моделей для розпізнавання обличчя. Традиційні підходи часто покладаються на складні, кодовані вручну алгоритми вилучення, вирівнювання та коригування функцій, але нещодавно ряд перспективних систем досягли кращих показників за допомогою нейронних мереж. Ці моделі швидко стають стандартом, за яким оцінюються системи розпізнавання обличчя, тому ми розглядаємо три типи нейромережевих моделей: softmax регресію, багатошарову персептронну

мережу та багатошарову мережу автоенкодерів, що відмовляє. Ці моделі різняться за складністю, причому регресія softmax є найпростішою, а мережева автоматизована кодуюча мережа є найбільш складною. Регресія Softmax. Цей класифікатор є узагальненням логістичної регресії, що дозволяє змінній класу приймати більше двох значень - у нашому випадку в наборі даних є 40 осіб, тому класифікатор повинен розрізняти 40 міток. Регресія Softmax часто використовується як остаточний шар в архітектурах глибоких нейронних мереж, тому сам по собі цей класифікатор можна розглядати як нейронну мережу без прихованих шарів. Багатошарова мережа перцептрону. У роботі використовується багатошарова мережа персептронів з одним прихованим шаром з 3000 сигмоподібних нейронів (або одиниць) та вихідним шаром softmax. Цей класифікатор можна розуміти як виконання регресії softmax після першого застосування нелінійного перетворення до вектора ознак. Суть цього перетворення, яке відповідає прихованому шару, полягає у відображенні вектора ознак у простір нижчого розміру, в якому класи розділяються вихідним шаром softmax.

Складена мережа автоматичного кодування шуму - цей класифікатор є прикладом глибокої архітектури і складається з двох прихованих шарів та одного вихідного шару softmax. Два приховані шари, які мають 1000 та 300 сигмоподібних одиниць, є екземплярами шумопоглинаючого автоенкодера.

Автоенкодер - це нейронна мережа, яка відображає свій вектор ознак у приховане представлення (як правило, у меншому просторі), а потім відображає його назад (тобто реконструює) у вихідний простір об’єктів. Автокодери навчені мінімізувати помилки реконструкції, тому вектори в прихованому просторі можна розглядати як стиснуте кодування простору функцій, яке має добре декодуватися для випадків у навчальних даних. Надія з цією архітектурою полягає в тому, що ці кодування охоплюють основні фактори зміни простору об’єктів (подібно до аналізу основних компонентів), що призводить до більшої роздільності шару softmax.

Було навчано кожен тип моделі за допомогою бази даних граней AT&T Laboratories Cambridge. Цей набір містить десять чорно-білих зображень 40 осіб у різних умовах освітлення.

Помилка моделі Softmax складає 7,5%, MLP 4,2%, DAE 3,3%.

Базова атака IM. Особливостями є повний вектор інтенсивності пікселів, що містить зображення, і кожна інтенсивність відповідає значенню з плаваючою точкою в діапазоні [0, 1]. У всіх розглянутих атаках не припускаємо, що зловмисник знає точні значення для будь-якого з пікселів у векторі, який він намагається зробити. Ці фактори поєднуються, щоб зробити цей тип інверсії суттєво відмінним від попередніх випадків, які розглядаються, тому ці атаки вимагають нових методів.

Припускаючи вектори функцій з n компонентами та m класами облич, ми моделюємо кожен класифікатор розпізнавання обличчя як функцію, f ˜: [0, 1] n → [0, 1] m. Результатом роботи моделі є вектор значень ймовірності, і-й компонент відповідає ймовірності того, що вектор ознак належить і-му класу. Запишемо f ˜i (x) як скорочення для і-ї складової результату.

Використовується градієнтний спуск (GD), щоб мінімізувати функцію витрат за участю f˜ для здійснення інверсії моделі в цьому налаштуванні. Градієнтний спуск знаходить локальні мінімуми диференційованої функції шляхом ітераційного перетворення рішення-кандидата в бік від'ємного градієнта у рішення-кандидата. Алгоритм заданий функцією MI-Face спочатку визначає функцію витрат c з точки зору моделі розпізнавання обличчя f ˜ та функцію AuxTerm, яка використовує будь-яку доступну допоміжну інформацію для інформування функції витрат. Потім MI-Face застосовує градієнтний спуск до α ітерацій, використовуючи кроки градієнта розміром λ. Після кожного кроку градієнтного спуску результуючий вектор ознак передається в процес обробки функції після обробки, який може виконувати різні маніпуляції із зображеннями, такі як зменшення шуму та різкість, як це необхідно для даної атаки. Якщо вартість кандидата не покращується за β-ітерацій або якщо вартість принаймні така ж, як γ, тоді зниження

припиняється і найкращий кандидат повертається. MI-Face потрібно обчислити градієнт функції витрат c, що, в свою чергу, вимагає обчислення градієнта моделі розпізнавання обличчя f ˜. Це означає, що для успіху атаки f ˜ має бути диференційованим. f ˜ можна обчислити вручну або автоматично, використовуючи символічні прийоми.

2.9 Атака реконструкції

Перша конкретна атака - Face-Rec, передбачає, що супротивник знає одну з міток, що виводиться моделлю, і бажає реконструювати зображення, що відображає впізнаване обличчя для людини, що відповідає мітці. Ця атака є досить прямою інстанцією MI-Face. Зловмисник не має допоміжної інформації, окрім цільової мітки, тому ми визначаємо AuxTerm (x) = 0 для всіх x. Експерименти встановили параметри для MI-Face: $\alpha = 5000$, $\beta = 100$, $\gamma = 0{,}99$ та $\lambda = 0{,}1$. У всіх випадках, крім мережевої мережі DAE, встановлюємо Process як функцію ідентифікації. Для складеної мережі DAE використовується функцію Process-DAE в алгоритмі. Оскільки супротивник може перевірити кожен із шарів моделі, він може ізолювати два шари автоенкодера.

У ході роботи було налаштовано атаку для генерації рішень-кандидатів у прихованому просторі першого шару автоенкодера, і на кожній ітерації ProcessDAE декодує кандидата у вихідний простір пікселів, застосовує фільтр шумозаглушення, за яким слідує фільтр загострення, і перекодує отримані пікселі у прихований простір.

Було виявлено, що ця обробка усуває значну кількість шуму від остаточної реконструкції, одночасно покращуючи впізнавані особливості та наближаючи відносну інтенсивність пікселів до зображень із навчального набору. Приклад показаний на рисунку 2.3.

Рис 2.3 Приклад зображень атаки

Зображення зліва було реконструйовано за допомогою мітки особи з правого зображення без використання Process-DAE, тоді як центральне зображення реконструйовано за допомогою цього кроку обробки.

Щоб оцінити ефективність атаки, вона була проведена на кожній із 40 міток у базі даних AT&T Face. Кожну партію експериментів проводили тричі, з однаковими тестовими зображеннями.

У 80% експериментів одне з п’яти зображень містило особину, що відповідає мітці, використаному в атаці. В якості контролю 10% випадків використовували звичайне зображення із набору даних, а не те, що було створено MI-Face. У всіх випадках зображення, що не відповідають мітці атаки, були вибрані випадковим чином із навчального набору.

2.10 Напади з використанням чорної скриньки

У деяких випадках можливе використання числових наближень для функції градієнта замість явного обчислення градієнта, використаного вище. Це дозволило б укласти ворога в чорну скриньку для обох типів атак на моделі розпізнавання обличчя. Був проаналізований підхід, використовуючи числове наближення градієнта scipy - він добре працює для моделей Softmax - реконструйовані зображення виглядають ідентично тим, що створені за допомогою явних градієнтів. Однак передбачувано, що продуктивність страждає. У той час як Softmax займає в середньому лише хвилину, моделі MLP та DAE займають значно більше часу. Кожне числове наближення

градієнта вимагає порядку 2-х викликів чорної скриньки функції витрат, кожна з яких займає приблизно 70 мілісекунд. За такої швидкості на проведення окремого експерименту MLP або DAE потрібно 50–80 днів. Пошук способів оптимізації атаки за допомогою приблизних градієнтів - цікава майбутня робота.

2.11 Змагальні атаки та змагальні приклади

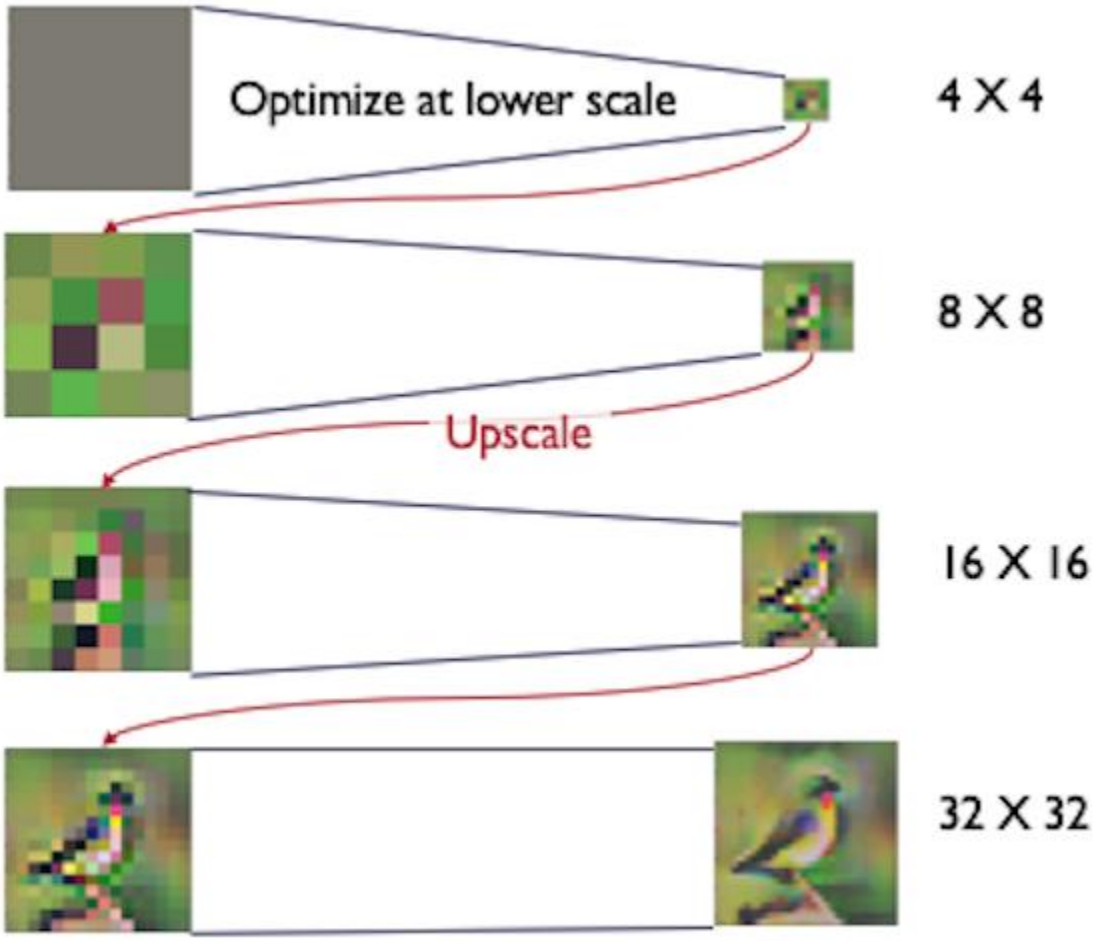

Рисунок 2.4 Покращення результату інверсійної атаки на датасеті CIFAR-10 завдяки DeepDream

Основною проблемою інверсійних атак моделей є те, що отримані рішення є змагальними вибірками з вхідного простору. Нещодавно

привертають увагу змагальні атаки, які є прикладами того, як невелика модифікація введення призводить до значних змін у результатах. Наприклад, нижче видно зображення золотої рибки з доданим шумом, яке класифікується як удав. Суперечливий шум тут також можна отримати за допомогою атак інверсії моделі.

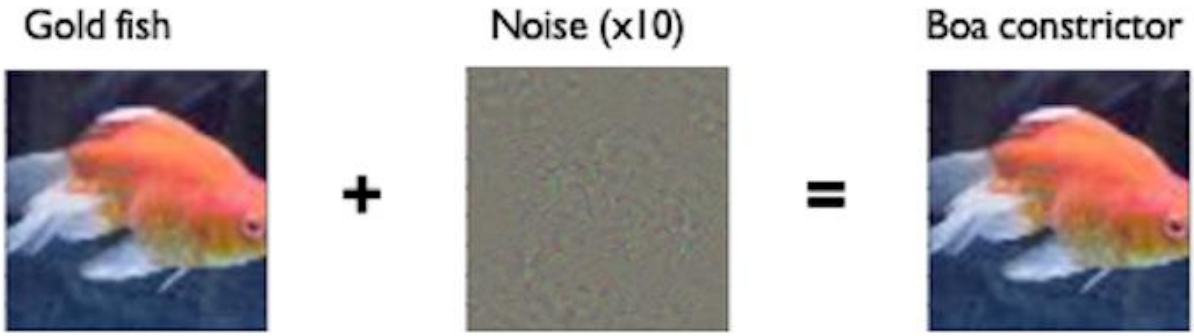

Рисунок 2.5 Додавання шуму до вхідного зображення змінює результати класифікації обраної моделі

Щоб мати змогу знайти те, що модель запам'ятала, потрібно видалити з моделі змагальні вхідні дані або, принаймні, зменшити їх. Існує велика кількість досліджень щодо того, як найкраще захищатися від змагальних нападів, але ми змагальне навчання насправді є найефективнішим захистом. Передумовою для цього захисту є атакувати модель кожної ітерації та використовувати змагальні приклади як навчальні дані. Після навчання модель навчилася правильно класифікувати змагальні приклади і більше не неправильно класифікує змагальні приклади в межах певного радіусу збурень.

У ході роботи були натреновані три різні моделі з використанням мови Python3 та фреймворку PyTorch, дві традиційно треновані (VGG16, ResNet) та одну змагальну (ResNet) на наборі даних CIFAR-10, а потім атакували їх прогнозованою атакою градієнтного спуску. Код тренування однієї з моделей наведений у додатку А.

Результати наведені нижче, але реконструкція проводиться ніч і день. Зменшуючи кількість змагальних прикладів за допомогою змагальних

тренувань, реконструкції стають більш чіткими та детальними, а зображення, що запам’ятовує модель, можна легше ідентифікувати.

Рисунок 2.6 Візуалізація результатів тренування моделей

Однією з цікавих відмінностей запам’ятовування моделей глибокого навчання порівняно з поверхневими моделями є кількість зразків, які запам’ятовуються. При глибокому навчанні модель має достатню здатність і кластеризувати, і запам’ятовувати. Це означає, що можна генерувати зображення більш високої точності, які не розмиваються через зміни пози, і можна генерувати більшу кількість зразків, як показано нижче.

Рисунок 2.7 Приклад відновлення даних класу “птиця” на датасеті CIFAR-10

Наступним питанням може бути те, які фактичні точки даних запам'ятовуються. Для цього були розглянуті подібність косинусів між кінцевим результатом згорткових шарів цільової моделі між реконструйованими зображеннями та всіма зображеннями в наборі даних CIFAR-10. У ході роботи виявилось, що дивний блакитний птах з довгою шиєю, який був створений атакою, насправді є справжнім, і назвали його казуаром, і в наборі даних є численні зразки цього птаха. Сформовані зображення з традиційно навчених моделей отримують неструктуровані подібні зразки, що припускає, що не тільки протирічні навчені моделі генерують зображення більш високої точності, але вони також більш сфокусовані на конкретних зразках або кластерах.

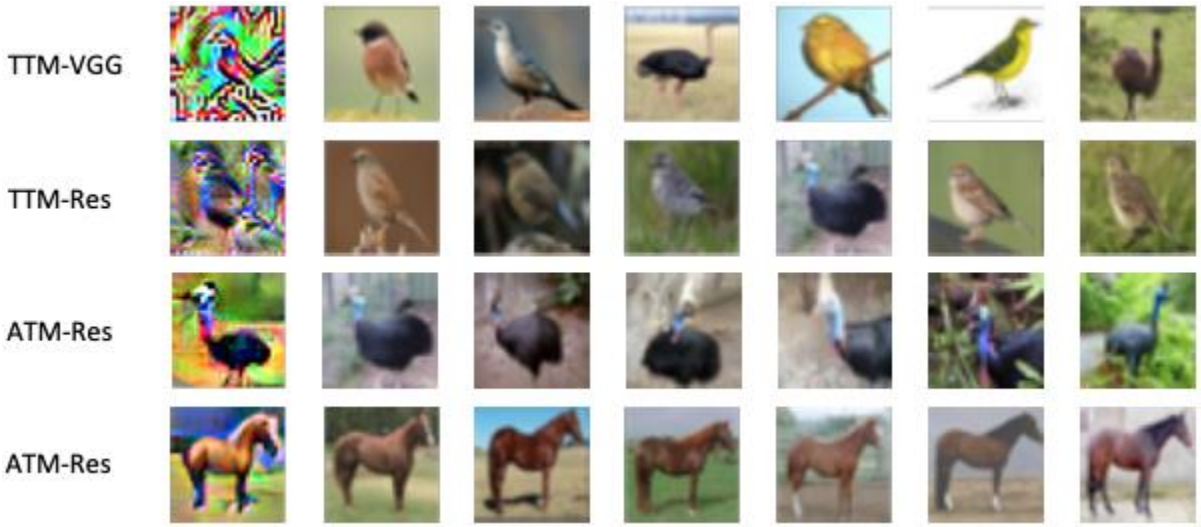

Рисунок 2.8 Результати відтворення даних, використовуючи різні налаштування тренування моделей

2.12 Інвертування міміки та метрики інверсії на практиці

Вилучення зображень із набору даних про тварин та предмети може бути не надто хвилюючим. Більшість людей із задоволенням демонструють зображення своїх собак чи котів, але як щодо окремих облич. Ці самі методи можна поширити на моделі, які тренуються на зображеннях обличчя. У роботі

це було перевірено на проблемі розпізнавання виразів обличчя Kaggle. Змагання на даній платформі складалося з зображень обличчя, згрупованих у сім різних класів емоцій (злість, огида, страх, радість, сум, здивування та нейтральність). Модель не навчена вивчати будь-яку конкретну особу в наборі даних, а натомість того, як класифікувати різні емоції, проте під час спостереження виявилось, що в деяких випадках модель запам'ятовує деяких людей більше, ніж інших. Були натреновані ті ж самі моделі, що і раніше, одну змагально та одну традиційно. Результати наведені нижче на рисунку 2.9.

Рисунок 2.9 Приклади відтворення обличчя з моделі VGG-16

Хоча реконструкції обличчя не є досконалими, вони дають інформацію про оригінальних людей. У міру вдосконалення методик навчання моделі стають все більш досконалими, і в результаті з’являються нові уразливості. Одним із прикладів цього є змагальне навчання: роблячи моделі надійнішими, їх легше інтерпретувати, а отже, їхні секрети можна краще інтерпретувати.

Успіх інверсійної атаки моделі є однією з головних проблем при розробці різних методів атаки. При оцінці атаки існує дві основні проблеми:

пошук найближчого зображення в наборі даних та кількісне визначення схожості між реконструйованим зображенням та зображенням у наборі даних. З цієї причини попередні моделі атак інверсії були проаналізовані якісно.
Був використаний двоступеневий підхід, щоб надати кількісну метрику для оцінки ефективності моделі інверсійної атаки - виявлення найближчої точки даних до реконструйованого зображення та кількісна оцінка подібності окремо.

Спочатку ідентифікується найбільш схожа точка даних, вводячи зображення в цільову модель, і витягується вектор ознак із кінцевого згорткового шару. Потім цей векторний елемент використовується для пошуку косинусної подібності найближчого зображення до отриманого реконструйованого зображення. На рисунку 2.6 показано навчальні зображення, які виявились найбільш близькими до моделі реконструкції інверсійної атаки (тут для категорії птахів). Навчальні зображення, найближчі до реконструкцій моделі TTM, мають велику мінливість у позі, типі птаха та тлі, тоді як ті, що були знайдені для реконструкції ATM, стосуються того самого типу птахів. Тут можна побачити блакитну шию та чорне тіло птахів, подібні до реконструкції. Щоб показати, що це не залежить від категорії, продемонстровані результати для коня для ATM на рисунку 2.9.

ATM-Res та TTM-Res мають середній максимальний бал схожості відповідно 0,85 та 0,78, тоді як TTM-VGG має середній бал подібності 0,99. Це свідчить про те, що TTM-VGG не може диференціювати мінливість у межах класу.

Отримавши зображення, яку модель вважає найбільш подібним, можна обчислити відстань зображення L2, щоб кількісно визначити подібність реконструйованого зображення до найближчого навчального зображення. Середній рівень L2 становить 82,4, 132,5, 97,1 для ATM-Res, TTM-VGG та TTM-Res відповідно.

Радіус змагальності обчислюється шляхом застосування змагальної атаки PGD, поки зображення не буде неправильно класифіковано. Збільшення

радіусу змагальності робить модель більш надійною, зменшує втрату конфіденційності.

У наступних підрозділах будуть розглянуті більш детально можливі підходи щодо покращення результатів відтворення даних завдяки атаки інверсії.

2.13 Інверсія моделі за допомогою PGD

У ході роботи були зроблені спостереження, що атаки PGD для інверсії моделі не реконструюють семантично значущих зображень для TTM. У кращому випадку створені зображення можуть допомогти в ідентифікації мітки певного класу, але не надають ніякої інформації про конкретну точку даних. Зображення, згенеровані за допомогою TTM-VGG, мають дуже гострі краї та лінії на всьому протязі з деяким незначним семантичним значенням. TTM-Res має трохи більше семантичного значення, але страждає від фрактальних шаблонів категорії, що генерується. Ці зображення також мають великі шаблони перевірки на всіх зображеннях. Після того, як модель навчена змагальним методом, реконструкції стають більш семантично значущими. Зображення чіткіші та орієнтовані на окремі зразки категорії.

Процес оптимізації реконструкцій також суттєво змінюється. Як видно на рисунку 2.7, TTM-Res швидко оптимізує зображення, створюючи зображення, яке класифікується як птах протягом 3 ітерацій, тоді як ATM-Res приймає 1136 ітерацій, щоб класифікувати їх як цільовий клас птахів. Це підкреслює поширеність змагальних прикладів у вхідному просторі TTM. Навіть якщо випадково ініціалізується зображення за допомогою різних входів, TTM приймає на два порядки менше ітерацій, щоб сформувати зображення, яке класифікується як клас, на який зловмисник був націлений.

Оскільки вбудовування моделі TTM не має сильного семантичного подання, широкий спектр змагальних прикладів задовольняє граничні умови цільової категорії - майже для будь-якої ініціалізації зображення існує суперечливий приклад, який знаходиться біля цього входу.

Значення активації TTM-Res та ATM-Res відрізняються майже на порядок, при цьому активація птахів становить 150 та 44 для TTM-Res та ATM-Res відповідно. Середнє значення активації за даними тренувань становить 11 та 7 для традиційно навченої моделі та змагальної моделі відповідно. Коли вводиться реконструйоване зображення з ATM-Res в TTM-Res, отримується значення активації птахів 21, все ще значно менше, ніж 150, але набагато вище середнього навчального зображення. Це демонструє, що всі реконструкції мають більш високу активацію, ніж природні зображення. Емпіричним шляхом було виявлена можливість відтворити 214 зображень за датасету CIFAR-10. Навчена змагальним способом реконструкція не є мінімумом у TTM-Res через сусідні приклади змагальності, і як тільки ці приклади видаляються, вона стає мінімальною.

Вводячи реальний навчальний образ в інверсію моделі, можливо додатково розрізнити характеристики обох моделей. Вводячи зображення кота з даних тренувань у TTM-Res, атака інверсії моделі починає генерувати кілька зайвих кішок і різко змінює зображення. Загальна інформація про зображення втрачається, і жодна з фігур не зберігається.

У той час як для ATM-Res, реконструйоване зображення більше фокусується на підкресленні рис, які визначали кота у всьому наборі даних, та на контрасті з фоном. Навчана реконструкція моделі також починає видаляти інформацію про фон, наприклад, розмиття стопи та штанів.

2.14 Інверсія моделі за допомогою DeepDream

Інверсія моделі DeepDream здатна відсунути реконструкцію зображення від прикладів змагальності завдяки використанню декількох масштабів. Це діє як фільтр, що фокусує реконструкцію спочатку на низькочастотних характеристиках, а потім додає більш високочастотні характеристики в міру масштабування реконструкції. Це значно покращує реконструкцію в TTM, як продемонстровано на рисунку 2.4. Але оскільки в ATM спостерігається зменшення поширеності змагальних прикладів, цей

метод не покращує інверсію моделі на цих моделях. Цей метод є більш успішним з TTM-VGG, ніж TTM-Res. TTM-Res, як правило, створюють реконструкції, які намагаються мати кратні значення класу, про який йде мова.

Зображення TTM-Res мають кілька коліс, рогів, котів, собак для класів вантажівок, оленів, котів та собак відповідно. Це одна з визначальних рис у методі DeepDream, але не спостерігається в TTM-VGG, що припускає, що підключення пропуску в архітектурі ResNet дозволяють моделі розглядати більше прикладів класу, як більш імовірного, що це клас. Ця характеристика ускладнює інверсію моделі, оскільки можна витягти маломасштабні елементи, але важче отримати масштабні. Завдяки змагальному навчанню моделі ResNet, ця характеристика наявності декількох прикладів класу на одному зображенні також усувається. Отримані, відтворені зображення ATM-Res сильно фокусуються на одній, конкретній, індивідуальній ознаці і створюють чіткий контраст між основним предметом / категорією та фоном зображення.

Фон вилучених зображень підкреслює те, що залишається приватним у моделі, причому фони собак переважно контрастні, а на тлі літаків, оленів, коней, човнів, вантажівок синє небо, трава, відкрите поле, вода та дороги, що відображають фони, узгоджені з усіма цими класами, тоді як собаки мають різноманітніші фони в наборі даних.

2.15 Інверсія моделі за допомогою GAN

Використовуючи GAN, фактично додатково обмежується процес генерації зображень для створення реальних зображень. Мета полягає в тому, щоб відштовхнути згенеровані зображення від змагальних або несемантично значущих зображень до реалістичних зображень. Для TTM навчальний процес швидко знаходить спосіб обдурити цільову мережу, по суті додаючи невелику кількість шуму та зосереджуючись на омані дискримінатора. Це обмежує можливість вилучення інформації з цільової моделі, оскільки згенеровані

зображення більш репрезентативні для тіньового набору даних, ніж для цільового.

Хоча кількість навчальних зображень, вразливих до інверсії моделі, обмежена, можна отримати різні реконструкції, використовуючи різні входи. Для інверсії моделі PGD можливо вводити однорідні зображення, але це обмежує реконструкцію кількома зображеннями. Також допустиме введення випадкових значень, але це шкодить загальній структурі зображення. Як видно на малюнку 2.10, реконструкції дійсно генерують різних птахів, але починають втрачати детальну інформацію. Крім того, для створення більш різноманітних птахів потрібно більше дисперсії шуму, що ще більше посилює втрату в семантичному поданні зображень.

Розмиття та врахування втрати L2 для оптимізації може допомогти у вирішенні цих проблем, але суттєво не покращить результат.

Рисунок 2.7 ілюструє шість різних птахів, отриманих за допомогою інверсії моделі DeepDream. Інверсія моделі DeepDream більше підходить для введення різних випадкових входів, оскільки вона природно розмиває і згладжує випадковий вхід у реконструйоване зображення. На малюнку ми бачимо, що колір птахів змінюється, але також змінюються деякі форми, а саме видимий страус та колібрі у середньому та лівому нижньому зображеннях відповідно.

Рисунок 2.10 Відтворення зображень завдяки GAN

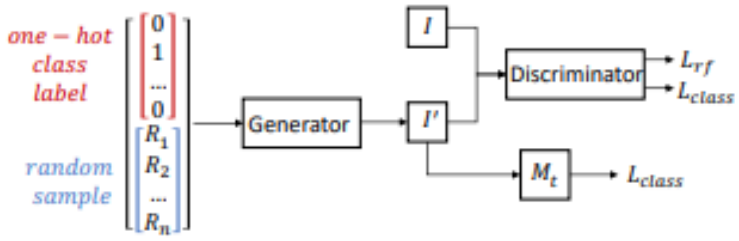

Рисунок 2.11 Діаграма генеративної змагальної мережі, що використовується для інверсії моделі

На рисунку 2.11 видно, що в генератор вводиться вектор міток одного гарячого класу та випадкових нормальних чисел. Генератор створює зображення, яке потім подається на дискримінатор із реальними зображеннями. Дискримінатор навчений класифікувати реальні зображення та розрізняти реальні зображення від сформованих зображень. Сформовані зображення також подаються до цільової моделі, а генератор навчається генерувати реальні зображення і обдурювати цільову модель.

Висновок за розділом 2

Отже у даному розділі було проаналізовано, як інформація про довіру, що повертається багатьма класифікаторами машинного навчання ML, дозволяє атакам інверсії нової моделі призводити до несподіваних проблем із конфіденційністю. Дані атаки також можна використовувати для вилучення зображень з моделей розпізнавання обличчя, які більшість кваліфікованих людей здатні послідовно повторно ідентифікувати.

Були досліджені та протестовані на датасеті CIFAR-10 змагальні атаки та атаки інверсії моделей з використанням підходів PGD, DeepDream, GAN. У ході роботи було виявлено, що використання змагального тренування покращує відновлення тренувальних даних, що є проблемою конфіденційності.

214 зображень з датасету CIFAR-10 у ході роботи були відтворені з використанням даної атаки.

Таким чином, четверте поставлене завдання дипломної роботи було виконано.

РОЗДІЛ 3
РОЗРОБКА РЕКОМЕНДАЦІЙ ЩОДО БЕЗПЕКИ ВИКОРИСТАННЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

3.1 Диференційна приватність в оптимізаторах під час тренування нейронної мережі

SGD та Adam є одними з найпопулярніших оптимізаторів глибокого навчання. Тут ми представляємо новий аналіз конфіденційності приватного варіанту SGD в рамках f-DP, а потім поширюємо дослідження на приватну версію AdamW, тобто AdamWeightDecay.

Позначаючи набір даних за S, ми розглядаємо можливість мінімізації емпіричного ризику.

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, x_i),$$

де сигма позначає ваги нейронних мереж і l є функцією втрат. На ітерації вибирається міні-партія з імовірністю субдискретизації, що має приблизний розмір. Беручи швидкість навчання та початкові ваги, ванільний SGD оновлює ваги відповідно до:

$$\theta_{t+1} = \theta_t - \eta_t \cdot \frac{1}{|I_t|} \sum_{i \in I_t} \nabla_\theta \ell(\theta_t, x_i).$$

Для збереження конфіденційності вводяться дві модифікації ванільного SGD. Спочатку до градієнта застосовується крок кліпу, щоб градієнт був фактично обмеженим. Цей крок необхідний, щоб мати граничну чутливість. Друга модифікація полягає в додаванні гаусового шуму до відсіченого градієнта, що еквівалентно застосуванню механізму Гауса до оновлених ітерацій.

Як внесок у роботу, DP-SGD використовує підвідбір Пуассона, на відміну від рівномірного відбору проб, що використовується. Однак, існує ще

один можливий метод субдискретизації: перестановка (випадкова перестановка та розподіл даних на складки в кожну епоху). Наголошується, що різні механізми субдискретизації дають різні гарантії конфіденційності. Окрім того, зазначено, що цей новий аналіз конфіденційності відрізняється від проведеного аналізу.

Корисно порівняти моменти обчислювача з аналізом конфіденційності, використовуючи фреймворк f-DP. Обчислювач моменту дає чітке індивідуальне відображення для визначення загальної втрати конфіденційності в термінах f-DP під композицію, що виходить за рамки теореми про склад. Дещо докладніше, обчислювач моментів використовує функцію генерування моментів випадкової змінної втрати конфіденційності для відстеження втрати конфіденційності за складом. Як зловживання нотацією ця стаття використовує функції і для позначення відображення, викликаного моментами бухгалтера в обох напрямках. Для замкнутості додаток містить офіційний опис двох функцій.

У цій роботі продемонстровано використання f-DP, нещодавно запропонованого визначення конфіденційності, для навчання приватних моделей глибокого навчання за допомогою SGD або Adam. Завдяки своїй міцності в обробці композиції та піддискретизації та потужній теоремі про центральну межу конфіденційності, f-DP дозволяє обмежувати конфіденційність закритої форми, яка є гострішою, ніж та, що надається обчислювачем моментів f-DP.

Застосовуючи чисельні експерименти, показано, що навчені нейронні мережі можуть бути досить приватними з точки зору f-DP. Це, в свою чергу, свідчить про те, що можна додати менше шуму під час навчального процесу, маючи ті самі гарантії конфіденційності, що і використання обчислювача моментів, покращуючи тим самим корисність моделі.

Існують також кілька напрямків для подальших досліджень. В якості першого напрямку можливо розглянути використання шкал шуму та швидкості навчання, залежних від часу, в DP-SGD та DP-AdamW для кращого

компромісу між втратою конфіденційності та корисністю в рамках f-DP. Вдалося досягти значного прогресу завдяки концентрованій диференційній приватності в цьому напрямку. Загалом, прямолінійною, але цікавою проблемою є поширення цієї роботи на складні архітектури нейронних мереж з різноманітними стратегіями оптимізації. Наприклад, чи можна розробити деякі рекомендації щодо вибору оптимізатора серед DP-SGD, DP-AdamW та інших для даної проблеми класифікації за певних обмежень конфіденційності? Емпірично, моделі глибокого навчання дуже чутливі до гіперпараметрів, таких як розмір міні-партії, з точки зору точності тесту. Тому з практичної точки зору було б дуже важливим включити налаштування гіперпараметрів у структуру f-DP. Натхненний ще одним цікавим напрямком - дослідити можливий взаємозв'язок між гарантіями f-DP та суперечливою стійкістю нейронних мереж. Враховуючи хорошу інтерпретованість та потужний набір інструментів f-DP, варто дослідити, чи, з широкої точки зору, його перевага над попередніми різними послабленнями конфіденційності матиме загальні приватні статистичні завдання та завдання машинного навчання.

3.2 Реалізація алгоритму

Основна ідея цього алгоритму полягає в тому, що ми можемо захистити конфіденційність навчального набору даних, втручаючись у градієнти параметрів, які модель використовує для оновлення своїх ваг, а не безпосередньо на дані. Додаючи шум до градієнтів на кожній ітерації, ми запобігаємо моделі запам'ятовувати свої навчальні приклади, одночасно дозволяючи навчатися в сукупності. (Непрямий) шум, як правило, має тенденцію до зниження протягом багатьох партій, побачених під час навчання.

Однак додавання шуму вимагає тонкого балансу: занадто багато шуму руйнує сигнал і занадто мало не гарантує конфіденційності. Щоб визначити правильну шкалу, розглянемо норму градієнтів. Важливо обмежити,

наскільки кожна проба може внести свій вклад у градієнт, оскільки викиди мають більші градієнти, ніж більшість зразків.

Потрібно забезпечити конфіденційність цих вибіжників, особливо тому, що вони мають найбільший ризик запам'ятовування моделлю. Для цього обчислюється градієнт для кожного окремого зразка в міні-партії. Окремо відсікаються градієнти, накопичуючи їх назад в єдиний тензор градієнта, а потім додаємо шум до загальної суми.

Це обчислення за вибіркою було однією з найбільших перешкод у побудові методу. Це більш складне завдання порівняно із типовою операцією з PyTorch, коли Autograd обчислює тензор градієнта для всієї партії, оскільки це має сенс для всіх інших випадків використання ML, і це оптимізує продуктивність. Щоб подолати це, була використана ефективна техніка для отримання всіх бажаних векторів градієнта при навчанні стандартної нейронної мережі.

Ідея алгоритму - поєднання трьох основних ідей:

1. Adam Weight Decay оптимізатор для тренування згорткових нейронних мереж.
2. Техніки диференційної приватності.
3. Регуляризація моделі.

Регуляризація, в математиці і статистиці, а також в задачах машинного навчання і обернених задачах, означає додавання деякої додаткової інформації, щоб знайти рішення некоректно поставленої задачі, або щоб уникнути перенавчання.

Регуляризацію використовують у задачах класифікації. Емпіричне навчання класифікаторів на скінченному набору даних завжди є недостатньо визначеною задачею.

Підходи до регуляризації моделі існують наступні:

1. Обмеження вагів лос функції.
2. Кліпінг градієнту функції втрат.
3. Підходи, пов'язані з батч нормалізацією.

4. Аугментація даних, зокрема підходи для test time augmentation.

У роботі були використані декілька ідей, які покращили результат, а саме кліпінг, тобто нормалізація значень градієнту під час оновлення вагів нейромережі під час ітерації тренування.

$$v_t^{(i)} \leftarrow \nabla_\theta l(\theta_t, x_i)$$

$$\underline{v}_t^{(i)} \leftarrow v_t^{(i)} / max\left\{1, \left|\left|v_t^{(i)}\right|\right|_2 / R\right\}$$

Регуляризація методу Adam Weight Decay відрізняється від звичайного Adam наступним виразом:

$$\omega_t \leftarrow m_t / \left(\sqrt{\quad}\right)$$

$$\theta_{t+1} \leftarrow \theta_t - \eta_t \omega_t$$

Саме дана частина функції з довільним гіперпараметром коефіцієнту регуляризації дозволяє отримати кращий результат сходимості алгоритму тренування. Особливо, принципи регуляризації допомагають обійти проблему перенавчання.

Перенавчання в питанні згорткових нейронних мереж - надлишкова ємність нейронної мережі, що під час тренування призводить до “вивчення” тренувальних даних, однак до зниження результатів метрики на валідаційних/тестових даних.

Проблема відтворення даних за допомогою атаки інверсійної моделі частково заснований на факторі запам’ятовування згортковою мережою текстури зображень або ключових кадрів відео. Саме опираючись на дану інтуїцію в ході роботи в основному використовувалися регуляризаційні підходи.

Нормалізація батчу як один з підходів не розглядався окремо, оскільки даний шар є частиною основних мереж, які використовувались в задачах класифікації та інших. Популярні архітектури використовують шари батч нормалізації постійно. Формула шару нормалізації батчу.

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Повний алгоритм методу наведений нижче.

Алгоритм DP-AdamW

Вхід	датасет S$\{x_1,\ldots,x_n\}$, функція втрат $l(\theta,x)$		
	Параметри	початкові ваги θ_0, швидкість навчання η_t, ймовірність субдискретизації p, кількість ітерацій T, шкала шуму σ, границя норми градієнта R, параметри імпульсу (β_1,β_2), початковий імпульсm_0, початковий градієнт минулого квадрата u_0 та мала константа $\xi > 0$.	
for	t = 0,...,$T-1$ **do** Візьмемо пуасонівську підвибірку $I_t \subseteq \{1,\ldots,n\}$з ймовірністю p **for** i ∈ I_t **do**		
		$v_t^{(i)} \leftarrow \nabla_\theta l(\theta_t, x_i)$ $\underline{v}_t^{(i)} \leftarrow v_t^{(i)} / max\left\{1, \left\|v_t^{(i)}\right\|_2 / R\right\}$	Кліп градієнта
		$\hat{v}_t \leftarrow \frac{1}{\lvert I_t\rvert}\left(\sum \underline{v}_t^{(i)}\right)$ $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1)\hat{v}_t$ $u_t \leftarrow \beta_2 u_{t-1} + (1-\beta_2)(\hat{v}_t \odot \hat{v}_t)$ $\omega_t \leftarrow m_t / \left(\sqrt{\ }\right)$ $\theta_{t+1} \leftarrow \theta_t - \eta_t \omega_t$	Агрегація градієнтів Крок оновлення моменту ⊙- добуток Адамара Крок додавання шуму Гаусса Оновлення вагів
Вихід	θ_{t+1}		

Код додавання операцій технік диференційної приватності до оптимізатору наведений у додатку Б.

3.3 Аугментація даних

Аугментація зображення - це процес створення нових навчальних прикладів із існуючих. Щоб зробити новий зразок, потрібно трохи змінити оригінальне зображення. Наприклад, можна зробити новий образ трохи яскравішим; можна вирізати шматок із вихідного зображення; можна зробити нове зображення, віддзеркаливши оригінальне тощо. Ось декілька прикладів трансформації вихідного образу, що створить новий навчальний зразок.

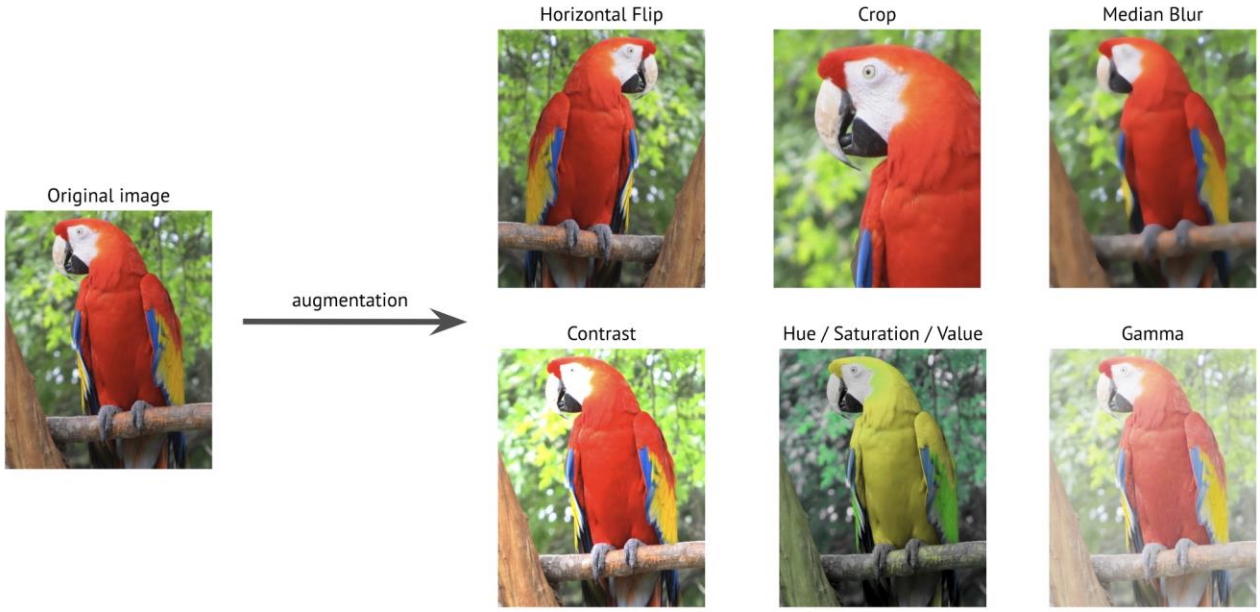

Рисунок 3.1 Приклад аугментації CIFAR-10 класу Bird.

Базові прийоми збільшення були використані майже у всіх статтях, що описують найсучасніші моделі розпізнавання зображень. AlexNet була першою моделлю, яка продемонструвала виняткові можливості використання глибоких нейронних мереж для розпізнавання зображень. Для навчання автори використовували набір основних прийомів збільшення зображень. Вони змінили розмір оригінальних зображень до фіксованого розміру 256 на 256 пікселів, а потім обрізали плями розміром 224 на 224 пікселі, а також їх

горизонтальні відбиття від цих зображень із розміром. Крім того, вони змінили інтенсивність RGB-каналів на зображеннях.

Послідовні найсучасніші моделі, такі як Inception, ResNet та EfficientNet, також використовували методи збільшення зображень для навчання.

У 2018 році Google опублікував статтю про AutoAugment - алгоритм, який автоматично виявляє найкращий набір доповнень для набору даних. Вони показали, що спеціальний набір збільшення покращує продуктивність моделі.

Ось порівняння між моделлю, яка використовувала лише базовий набір збільшення та моделлю, яка використовувала певний набір збільшення, виявлену AutoAugment. У таблиці наведено точність перших-1 (%) набору для перевірки ImageNet; чим вище, тим краще.

Model	Base augmentations	AutoAugment augmentations
ResNet-50	76.3	77.6
ResNet-200	78.5	80.0
AmoebaNet-B (6,190)	82.2	82.8
AmoebaNet-C (6,228)	83.1	83.5

Таблиця демонструє, що різноманітний набір збільшення зображень покращує продуктивність нейронних мереж порівняно з базовим набором лише з кількома найпопулярнішими методами перетворення.

Збільшення допомагають боротися із надмірним оснащенням та покращують ефективність роботи глибоких нейронних мереж для таких завдань комп’ютерного зору, як класифікація, сегментація та виявлення об’єктів. Найкраще те, що бібліотеки збільшення зображень, такі як Albumentations, дозволяють додати збільшення зображень до будь-якого конвеєра комп'ютерного зору з мінімальними зусиллями.

У ході роботи використовувалася бібліотека Albumentations. Код аугментацій наведений у додатку Б.

Головними аугментаціями, які використовувались під час тренування:

1. Горизонтальний фліп.
2. Вертикальний фліп.
3. Афінні перетворення.
4. 19-20 градусів оберт.
5. Зміна контрасту
6. RGB-shift.

3.4 Тестування та результати

Для тестування результатів нового методу оптимізації тренування згорткової мережі тестування було вирішено порівняти результати згорткової нейромережі, навченої за допомогою класичного оптимізатора Adam, та нового оптимізатору DP-AdamW.

Тестування складалося з наступних компонентів:

1. Тренувальний датасет.
2. Модель.
3. Процес тренування.
4. Метрика якості.
5. Кількість прикладів, які можливо відновити атакою інверсійної моделі.

Як тренувальний датасет було обрано CIFAR-10.

Моделлю для тестування було використано VGG-16. Дана модель фактично складається з шарів згортки, шарів нелінійності(ReLU), шарів батч нормалізації, maxpool, а також щільний шар з нелінійною функцією активації softmax на 10 класів згідно датасету CIFAR-10.

Для даної моделі використовувалися претреновані ваги з датасету ImageNet.

Нижче наведена архітектура моделі VGG-16 на рисунку 3.2.

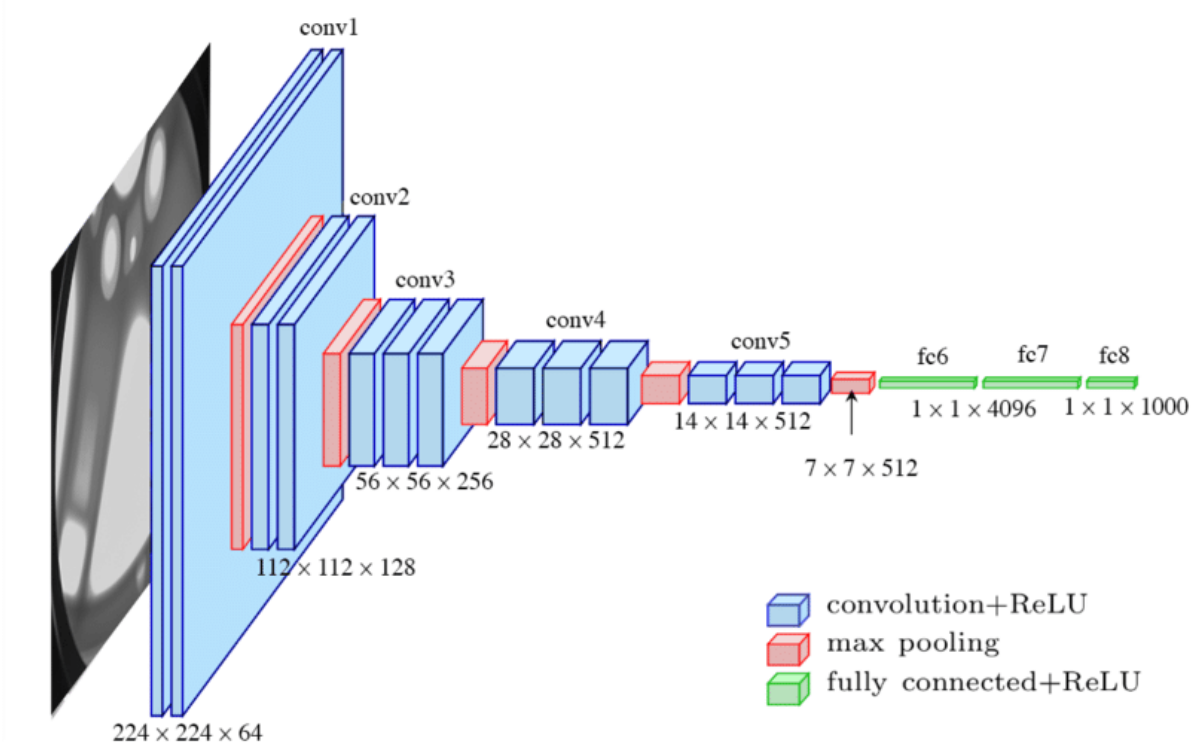

Рисунок 3.2 Архітектура моделі VGG-16

На вхід подається зображення розміром (224, 224, 3), на виході отримуємо розподіл ймовірностей віднесення до одного з десяти класів.

Для метрики якості використовувалась точність, тобто відношення кількості правильних передбачень до загальної кількості передбачень. Була обрана дана метрика через легкість інтерпретації, а також через те, що датасет CIFAR-10 мав збалансований набір класів. В інших випадках при дисбалансі класів рекомендується використання метрики f1. Кількість відновлених прикладів оцінювався емпірично.

У наступній таблиці наведено результати тренування з використанням різних оптимізаторів.

Датасет	Модель	Оптимізатор	Точність	Кількість відновлених прикладів
CIFAR-10	VGG-16	Adam	94%	214
		DP-AdamW	**87%**	**6**

Аналізуючи результати, можна дійти висновку, що відновлення прикладів завдяки атаки інверсійної моделі впливає на результуючу точність моделі. Покращення стійкості до атаки погіршує якість.

Нижче представлений приклад спроби відновити зображення з датасету на рисунку 3.3.

Рисунок 3.3 Спроба атаки інверсійної моделі натреноваї VGG-16 за допомогою DP-AdamW.

3.5 Оцінка методу DP-AdamW

Оцінка методу спиралася на такі обрані критерії, як:

1. Вартість рішення(якісна оцінка).
2. Точність на обраному датасеті(кількісна оцінка, у відсотках).
3. Стійкість до атаки інверсійної моделі(порівняльна оцінка згідно використання оптимізатору Adam, у відсотках).

Нижче наведена таблиця з оцінкою методу згідно перерахованих критеріїв.

Оптимізатор DP-AdamW	
Вартість	низька
Точність	87%
Стійкість	97%

3.6 Покращення результатів

Після тестування даного методу наступним завданням було покращення поточної метрики. Оскільки модель була натренована з використанням аугментацій фліпу, обертів, зміни контрасту та інших, у даній роботі була використана техніка “test time augmentation”.

Подібно до того, як аугментація даних робить із навчальним датасетом, метою тестових аугментацій є виконання випадкових модифікацій тестових зображень. Таким чином, замість того, щоб показувати звичайні, «чисті» зображення лише один раз натренованій моделі, кілька разів буде показано їй доповнені зображення. Прогнози кожного відповідного зображення будуть усереднені і будуь приймати це як остаточне здогадування.

Рисунок 3.4 Змінені зображення машини, яка подавалась на вхід

Використання даного підходу дозволило збільшити якість на тестовому датасеті на 2%.

Наступним покращенням було вибір порогу для отримання оптимальних збалансованих значень метрик precision, recall для кожного з 10 класів.

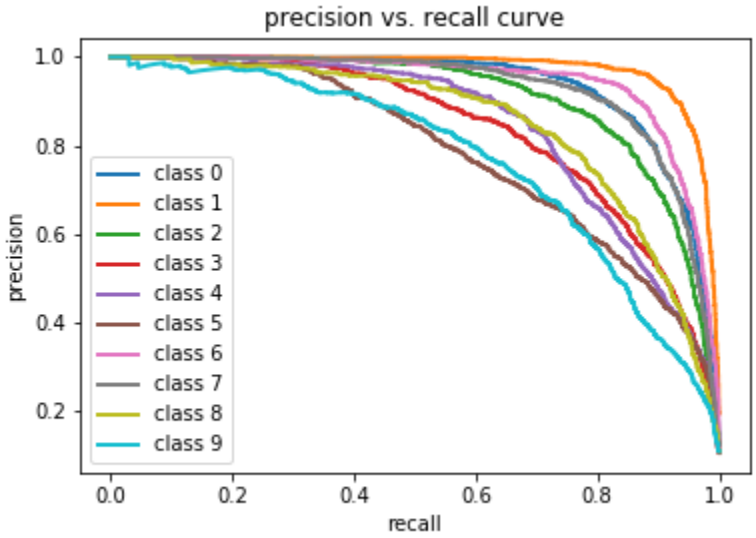

Рисунок 3.5 Візуалізація значень precision, recall для кожного з 10 класів датасету CIFAR-10

У ході експериментальної перевірки найкращим порогом було виявлено 0.6, що дозволило покращити результати моделі на 1% на тестовій вибірці.

Загалом, результати моделі покращилися з 87% до 90%.

3.7 Розробка рекомендацій щодо безпеки згорткової нейронної мережі

Таким чином, рекомендації для забезпечення безпеки використання нейронних мереж та збереження приватності даних через високий рівень стійкості обраної натренованої моделі до атак інверсійної моделі та змагальних атак наступні:

1. Використання нового оптимізатору тренування DP-AdamW через низьку вартість, високу точність та високу стійкість до атак інверсійної моделі.
2. Використання аугментацій даних для кращої регуляризації моделі.
3. Використання шару батч нормалізації в архітектурах згорткових нейронних мереж.

4. Використання тестових аугментацій для покращення результатів передбачення моделі згідно обраної метрики.

Висновок за розділом 3

Отже, у цьому розділі була проведена наступна робота:

1. Проаналізовані алгоритми диференційної приватності у оптимізаторах SGD, Adam.
2. Було розроблено новий метод оптимізації тренування згорткової нейронної мережі диференційно приватний DP-AdamW.
3. Був представлений новий алгоритм тренування згідно даного оптимізатору.
4. Метод оптимізації був оцінений згідно 3 основних критеріїв - вартості, точності, стійкості до атак. Хоча точність була трохи гірше та зменшилася на 4%, однак стійкість до атак інверсії збільшилася на 97%.
5. Були розроблені кроки щодо покращення результату тренування моделей з використанням аугментації даних та тестових аугментацій.

Таким чином, п’яте та шосте поставлені завдання дипломної роботи були виконані.

ВИСНОВКИ

Таким чином, у ході дипломної роботи було досягнуто мету роботи, проаналізовано сучасні методи диференційної приватності, проаналізовані та протестовані атаки інверсійної моделі на датасеті Cifar-10.

Було проаналізовано та визначено, що атаки інверсійної моделі створюють проблему конфіденційності даних в приватних наборах даних.

Був створений новий метод оптимізації згорткових нейронних мереж DP-AdamW, який дозволив збільшити стійкість до атак інверсійної моделі на 97%.

Після цього були розроблені конкретні заходи по удосконаленню точності отриманої моделі завдяки аугментаціям даних та підбору порогу, що дозволило покращити модель на 3%.

Практичне значення роботи полягає у розробці нового методу оптимізації процесу тренування згорткових нейронних мереж.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Постанова К. М. У. Про затвердження Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах» від 29 березня 2006 р //Офіційний вісник України. – 2006. – №. 13.

2. ТЗІ Н.Д.ТЗІ 1.1-002-99. Загальні положення щодо захисту інформації в комп'ютерних системах від несанкціонованого доступу //К.: ДСТСЗІ СБ України. – 1999. – Т. 16.

3. ДСТУ 2938-94. Системи оброблення інформації. Основні поняття. Терміни та визначення. – 1995. – с. 36.

4. А. І. Семенченка, В. М. Дрешпака. Захист інформації в системах електронного урядування. – 2017. – с. 72.

5. ISO/IEC 9798-5:2009 - Information technology — Security techniques — Entity authentication — Part 5: Mechanisms using zero-knowledge technique.

6. Franks J. et al. HTTP authentication: Basic and digest access authentication. – 1999. –№. RFC 2617.

7. Nan, X., & Guang, Z. CPK credibility authorization system. – 2006.

8. Multi-Factor Authentication. // PCI Security Standards Council. – 2017.

9. ISO/IEC TR 29156:2015 - Information technology – Guidance for specifying

performance requirements to meet security and usability needs in applications using

biometrics.

10. Das M. L. Two-factor user authentication in wireless sensor networks //IEEE

transactions on wireless communications. – 2009. – Т. 8. – №. 3. – С. 1086-1090.

11. ISO/IEC JTC 1/SC 37 [Електронний ресурс]. – Режим доступу:

https://www.iso.org/committee/313770.html

12. Face Anti-Spoofing или технологично узнаём обманщика из тысячи по лицу [Електронний ресурс]. – Режим доступу:https://habr.com/ru/company/ods/blog/452894/

13. ISO/IEC TC JTC1 SC37 Biometrics. ISO/IEC 2382-37:2017 IT – Vocabulary – Part 37: Biometrics// ISO and IEC – 2017.

14. Мельник Л. С. Засіб автентифікації за клавіатурним почерком на основі нейромережевої моделі. Магістерська дипломна робота.// – 2015. – 76 с. Українською мовою.

15. Kupin A. I., Kumchenko Y. O. Usage of training methods to parameterization of multilayer neural computing structures for technological processes //Радіоелектронні і комп'ютерні системи. – 2014. – №. 5. – С. 100-104.

16. Вежневец В., Дегтярева А. Обнаружение и локализация лица на изображении //CGM Journal. – 2003.

17. Yang M. H., Roth D., Ahuja N. A SNoW-based face detector //Advances in neural information processing systems. – 2000. – С. 862-868.

18. Купін А. І., Кумченко Ю. О. Развитие существующих методов создания эталона изображения для IT-биометрической идентификации //Наукові праці [Чорноморського державного університету імені Петра Могили комплексу Києво-Могилянська академія]. Серія: Комп'ютерні технології. – 2014. – №. 250, Вип. 238. – С. 85-89.

19. Хитров М. Мультимодальная система доступа с использованием голосовой биометрии [Електронний ресурс]. – Режим доступу: http://www.sdirector.ru/magazine/magdocs/view/94.html

20. Попов М. Биометрические системы безопасности [Електронний ресурс]. – Режим доступу: http://www.bre.ru/security/12571.html

21. VeriFinger SDK [Електронний ресурс]. – Режим доступу: http://www.neurotechnology.com/verifinger.html.

22. VeriLook SDK [Електронний ресурс]. – Режим доступу: http://www.neurotechnology.com/verilook.html.

23. VeriEye SDK [Електронний ресурс]. – Режим доступу: http://www.neurotechnology.com/verieye.html.

24. Русин Б. Біометрична автентифікація та криптографічний захист: монографія/Русин Б //НАН України, Фіз.-мех. ін.-т ім.. ГВ Карпенка.–Львів: Коло.–2007.–287 с.

25. Verbitskiy E. et al. Reliable biometric authentication with privacy protection //Proc. 24th Benelux Symposium on Information theory. – 2003. – С. 19.

26. Болл Р. М. и др. Руководство по биометрии //М.: техносфера. – 2007. – Т. 368.

27. Безопасность алгоритмов машинного обучения. Защита и тестирование моделей с использованием Python [Електронний ресурс]. – Режим доступу:

https://habr.com/ru/company/dsec/blog/438644/

28. Современные биометрические методы идентификации [Електронний ресурс]. –Режим доступу https://habr.com/ru/post/126144/

29. Обман нейронной сети для начинающих [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/dsec/blog/443164/

30. Standard F. 1037C: Telecommunications: Glossary of Telecommunication Terms. National Communication System. Technology and Standards Division. Washington, DC: General Services Administration //Information Technology Service. – 1996.

31. Безопасность алгоритмов машинного обучения. Атаки с использованием Python[Електронний ресурс]. – Режим доступу:

https://habr.com/ru/company/dsec/blog/437092/

32. Goodfellow I. J., Shlens J., Szegedy C. Explaining and harnessing adversarial examples //arXiv preprint arXiv:1412.6572. – 2014.

33. Szegedy C. et al. Intriguing properties of neural networks //arXiv preprint

arXiv:1312.6199. – 2013..

34. Song Q., Wu Y., Yang L. Attacks on State-of-the-Art Face Recognition using

Attentional Adversarial Attack Generative Network //arXiv preprint arXiv:1811.12026. – 2018.

35.Wang S. et al. Defensive dropout for hardening deep neural networks under adversarial attacks //Proceedings of the International Conference on Computer-Aided Design. –ACM, 2018. – С. 71.

36. Xu W., Evans D., Qi Y. Feature squeezing: Detecting adversarial examples in deep neural networks //arXiv preprint arXiv:1704.01155. – 2017.

37. Zantedeschi V., Nicolae M. I., Rawat A. Efficient defenses against adversarial attacks //Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. –ACM, 2017. – С. 39-49.

ДОДАТКИ

ДОДАТОК А

Код тренування нейронної мережі:

```
import os
import cv2
from PIL import Image, ImageOps, ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True

import pandas as pd
import numpy as np
import torch
import copy
from tqdm import tqdm

import albumentations as A
from albumentations.pytorch import ToTensorV2
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, precision_score, recall_score

from torchvision.models import resnet50, resnet34, resnet18, resnet101, resnet152,
densenet121
from torch import nn, optim
from torchvision import transforms, utils
from torch.utils.data import Dataset, DataLoader, SubsetRandomSampler
import time

import warnings
warnings.simplefilter("ignore")

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print('Device: {}'.format(device))


np.random.seed(42)
```

```
torch.manual_seed(42)


ALLOWED_EXTENSIONS = ['jpg','JPG','jpeg','JPEG','png','PNG']

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

class Dataset(Dataset):
    def __init__(self, split_df_path, transform=None):
        self.split_df = pd.read_csv(split_df_path)
        self.transform = transform

    def __len__(self):
        return self.split_df.shape[0]

    def __getitem__(self, index):
        row = self.split_df.loc[index]
        #print(os.path.join(row['folder_name'], row['filename']))
        img =
np.array(ImageOps.exif_transpose(Image.open(os.path.join(row['folder_name'],
row['filename']))).convert('RGB')))

        if self.transform:
            img = self.transform(image=img)['image']

        return torch.tensor(img).float(), torch.tensor(row.target).float()



def save_train_validation_df(NOT_GOOD_IMG_FOLDER,
GOOD_IMG_FOLDER, model_name, test_size=0.4):

    filenames_good = os.listdir(GOOD_IMG_FOLDER)
    filenames_good = pd.DataFrame(list(filter(lambda x: allowed_file(x),
filenames_good)), columns=['filename'])
    filenames_good['target'] = 0
    filenames_good['folder_name'] = GOOD_IMG_FOLDER
```

```
    filenames_not_good = os.listdir(NOT_GOOD_IMG_FOLDER)
    filenames_not_good = pd.DataFrame(list(filter(lambda x: allowed_file(x),
filenames_not_good)), columns=['filename'])
    filenames_not_good['target'] = 1
    filenames_not_good['folder_name'] = NOT_GOOD_IMG_FOLDER

    df = pd.concat([filenames_not_good, filenames_good]).sample(frac=1)

    train, validation = train_test_split(df, stratify=df['target'], test_size=test_size)
    validation, test = train_test_split(validation, stratify=validation['target'],
test_size=0.25)

    train = train.reset_index(drop=True)
    validation = validation.reset_index(drop=True)
    test = test.reset_index(drop=True)

    train.to_csv(f'{model_name}_train.csv', index=False)
    validation.to_csv(f'{model_name}_validation.csv', index=False)
    test.to_csv(f'{model_name}_test.csv', index=False)

    return train, train.shape[0], validation.shape[0], test.shape[0]


def train_model(model, dataloaders, dataset_sizes, criterion, optimizer, scheduler,
model_name, BATCH_SIZE, num_epochs=25):
    since = time.time()

    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    for epoch in range(num_epochs):

        # Each epoch has a training and validation phase
        for phase in ['train', 'val', 'test']:
            if phase == 'train':
                model.train()  # Set model to training mode
            else:
                model.eval()   # Set model to evaluate mode
```

```
running_loss = 0.0
running_corrects = 0

losses = []
accuracies = []

predicted_values = []
real_values  = []

# Iterate over data.
bar = tqdm(dataloaders[phase])
for inputs, labels in bar:
    inputs = inputs.to(device)
    labels = labels.to(device)

    # forward
    # track history if only in train
    with torch.set_grad_enabled(phase == 'train'):
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        loss = criterion(outputs, labels.long())

        # backward + optimize only if in training phase
        if phase == 'train':
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

    with torch.no_grad():

        # statistics
        running_loss += loss.item() * inputs.size(0)
        running_corrects += torch.sum(preds == labels.data)

        losses.append(loss.item())
        smooth_loss = np.mean(losses[-30:])

        accuracies.append(torch.sum(preds == labels.data))
```

```
                smooth_acc = np.sum(accuracies[-30:]) / (len(accuracies[-30:]) *
BATCH_SIZE)

                predicted_values.extend(preds.cpu().detach().numpy() )
                real_values.extend(labels.data.cpu().detach().numpy().astype(int))


            bar.set_description(f'epoch: {epoch}, phase: {phase}, smth_loss:
{smooth_loss}, smth_acc: {smooth_acc}')

        if phase == 'train':
            scheduler.step()

        with torch.no_grad():
            epoch_loss = running_loss / dataset_sizes[phase]
            epoch_acc = running_corrects.double() / dataset_sizes[phase]

        print('{} Loss: {:.4f} Acc: {:.4f} F1: {:.4f} Precision: {:.4f} Recall:
{:.4f}'.format(
            phase, epoch_loss, epoch_acc, f1_score(real_values, predicted_values),
precision_score(real_values, predicted_values), recall_score(real_values,
predicted_values)))

        # deep copy the model
        if phase == 'val':
            best_acc = epoch_acc
            best_model_wts = copy.deepcopy(model.state_dict())
            torch.save(model.state_dict(),
f'{model_name}_epoch{epoch}_acc{round(best_acc.item(),3)}.pt')

        if phase == 'test':
            print('='*15)

    time_elapsed = time.time() - since
    print('Training complete in {:.0f}m {:.0f}s'.format(
        time_elapsed // 60, time_elapsed % 60))
    print('Best val Acc: {:4f}'.format(best_acc))

    # load best model weights
```

```
    model.load_state_dict(best_model_wts)
    return model

def generic_train(model, NOT_GOOD_IMG_FOLDER, GOOD_IMG_FOLDER,
model_name, train_transform, validation_transform, BATCH_SIZE=4,
test_size=0.3, lr=3e-4, n_epochs=25):

    train_df, train_lenght, val_lenght, test_lenght =
save_train_validation_df(NOT_GOOD_IMG_FOLDER, GOOD_IMG_FOLDER,
model_name, test_size=0.3)



    train_dataset = Dataset(split_df_path=f'./{model_name}_train.csv',
transform=train_transform)

    validation_dataset = Dataset(split_df_path=f'./{model_name}_validation.csv',
transform=validation_transform)

    test_dataset = Dataset(split_df_path=f'./{model_name}_test.csv',
transform=validation_transform)

    train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE,
shuffle=True)
    val_loader = DataLoader(validation_dataset, batch_size=BATCH_SIZE)
    test_loader = DataLoader(test_dataset, batch_size=BATCH_SIZE)

    dataloaders = {'train': train_loader, 'val': val_loader, 'test': test_loader}
    dataset_sizes = {'train': train_lenght, 'val': val_lenght, 'test': test_lenght}
    #print(train_df.head())
    #print(train_df.target.value_counts())
    nSamples = [train_df.target.value_counts()[0], train_df.target.value_counts()[1]]
    normedWeights = [1 - (x / sum(nSamples)) for x in nSamples]
    normedWeights = torch.FloatTensor(normedWeights).to(device)
    criterion = nn.CrossEntropyLoss(weight=normedWeights)

    # Observe that all parameters are being optimized
    optimizer = optim.Adam(model.parameters(), lr=lr)
```

```
    # StepLR Decays the learning rate of each parameter group by gamma every step_size epochs
    step_lr_scheduler = optim.lr_scheduler.CosineAnnealingLR(optimizer, n_epochs)

    model = train_model(model, dataloaders, dataset_sizes, criterion, optimizer, step_lr_scheduler, model_name, BATCH_SIZE, num_epochs=n_epochs)
```

ДОДАТОК Б

Код додавання операцій технік диференційної приватності до оптимізатору:

```
def attach(self, optimizer: torch.optim.Optimizer):
    self.validator.validate(self.module)
    norm_clipper = (
        clipping.ConstantFlatClipper(self.max_grad_norm)
        if not isinstance(self.max_grad_norm, list)
        else clipping.ConstantPerLayerClipper(self.max_grad_norm)
    )

    if self.misc_settings.get("experimental", False):
        norm_clipper = clipping._Dynamic_Clipper_(
            [self.max_grad_norm],
            self.misc_settings.get("clip_per_layer", False),
            self.misc_settings.get(
                "clipping_method", clipping.ClippingMethod.STATIC
            ),
            self.misc_settings.get("clipping_ratio", 0.0),
            self.misc_settings.get("clipping_momentum", 0.0),
        )

    self.clipper = PerSampleGradientClipper(
        self.module,
        norm_clipper,
        self.batch_first,
        self.loss_reduction,
    )

    def dp_zero_grad(self):
        self.privacy_engine.zero_grad()
        self.original_zero_grad()
```