МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

"Інформаційна технологія тестування робастності моделей штучного інтелекту до змагальних атак"

Завідувач випускаючої кафедри	Довбиш А.С.
Керівник роботи	Москаленко В.В.
Студент	Хитров О.Б.

СУМИ 2021

Реферат

Записка 51 стор., 24 рис., 1 додаток, 27 джерел

Об'єкт дослідження - математичний апарат моделей машинного навчання на основі глибоких нейронних мереж, їх вразливості до змагальних атак

Мета роботи - розробка програмного забезпечення для генерації змагальних зразків та тестування робастності моделей машинного навчання

Методи дослідження математичне моделювання, порівняння, аналізу, тестування.

Результати було проведено аналіз алгоритмів змагальних атак та методів захисту від них. Представлено програму для проведення тестів і оцінки робастності моделей машинного навчання для розпізнавання зображень з різною топологією та формую вхідних даних. Для реалізації проєкту була використана бібліотека tensorflow.js. Система розгорнута на платформі github у вигляді web-додатку. Були проведені тести та оцінка робастності моделей MNIST, GTSRB, CIFAR до найпоширеніших змагальних атак, Fast Gradient Sign, Basic Iterative Method, Jacobian Saliency Map Attack, Carlini & Wagner attack.

МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА,

ЗМАГАЛЬНІ АТАКИ, TENSORFLOW

Сумський державний університет

(назва вузу)

Факультет ***ЦЗДВН*** Кафедра ***Комп'ютерних наук***

Спеціальність ***122-Комп'ютерні науки***

Затверджую:

зав. кафедри _____

"_____"_____ 20____р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ)
СТУДЕНТОВІ

Хитрову Олександру Борисовичу

(прізвище, ім'я, по баткові)

1. Тема проєкту (роботи) ***Інформаційна технологія тестування робастності моделей штучного інтелекту до змагальних атак***

   затверджую наказом по інституту від "_____"_____ 20_____р. №_____

2. Термін здачі студентом закінченого проєкту (роботи) _____

3. Вхідні дані до проєкту (роботи) _____

   _____

   _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

   1) Аналіз проблеми та постановка задачі

   2) Проєктування системи.

   3) Проведення тестів

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

   _____

   _____

6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник

_____
(підпис)

Завдання прийняв до виконання

_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проєкту (роботи)	Термін виконання проєкту (роботи)	Примітка
1	*Аналіз проблеми та постановка задачі*		
2	*Проєктування системи*		
3	*Інформаційне та програмне забезпечення*		
4	*Оформлення кваліфікаційної магістерської роботи*		

Студент дипломник _____
(підпис)

Керівник проєкту _____
(підпис)

Зміст

Вступ

Глибокі нейронні мережі демонструють високу ефективність в таких областях як розпізнавання зображень, розпізнавання мови, майже з точністю людини. Але було помічено, що вони є вразливими до змагальних атак. Якщо існує клас x, можливо знайти x' який буде подібним до x але буде класифікований як z. Це ускладнює застосування нейронних мереж в областях з високими вимогами до безпеки. Вперше це було помічено в області розпізнавання зображень[23]. Можливо внести незначні зміни у вхідне зображення, щоб результат класифікації був хибним. Як правило, величина зміни є настільки не значними, що вони непомітні для людини.

Ці вразливості привернули увагу багатьох дослідників, які намагалися підвищити стійкість нейронних мереж до подібних атак. Більшість спроб були невдалими або мали суттєві обмеження. Машинне навчання використовується в багатьох прикладних областях, в яких на основі результатів моделі приймаються критично важливі рішення, таких як керування безпілотними транспортними засобами, дронам, роботам, пошук небезпечного програмного забезпечення, розпізнавання голосових команд. Розуміння особливостей безпеки моделей глибинного навчання стає ключовим питанням області. Було доведено, що при розпізнаванні голосу можливо згенерувати такі зразки даних, які будуть сприйматись моделлю як команда, але не будуть звучати як мова для людини. Це може використано для контролю над пристроями без відома їх користувачів. Наприклад при відтворенні відео з прихованою голосовою командою можливо змусити мобільний пристрій відкрити зловмисний сайт. В області розпізнавання шкідливого ПЗ, вразливості моделей до змагальних атак дають змогу створювати шкідливе ПЗ, яке при незначних модифікаціях може уникати класифікації. В області безпілотного транспорту також проводились експерименти зі змагальними атаками, демонструють потенційні загрози у використанні моделей. [11] Отже, необхідно вивчити та порівняти надійність різних класифікаторів до змагальних модифікацій. Це могло бути ключем до кращого розуміння обмежень сучасної архітектури моделей ма-

шинного навчання.

В цій роботі

- розглядаються механізми атак на нейронні мережі та можливі заходи захисту.
- пропонується простий та надійний спосіб для обчислення робастності різних класифікаторів

Метою даної є розробка підходів та підготовка інструментарію для оцінки робастності моделей, що здатні навчатися.

Розділ 1

Аналіз предметної області

1.1 Про машинне навчання

Машинне навчання це галузь комп'ютерних наук що досліджує алгоритми які виконують задачі не слідуючи статичним інструкціям, а поступово свої параметри до досягнення максимально точного результату прогнозу або класифікації. Процес оптимізації параметрів моделі називають навчанням або тренуванням. Машинне навчання можна поділити на дві категорії в залежності від того як відбувається навчання на[20]:

- Навчання під наглядом (*Supervised learning*)
- Навчання без нагляду (*Unsupervised learning*)

При навчанні під наглядом йдеться про моделі, які класифікують не відомі їм зразки використовуючи функцію, сформовану з використанням розмічених зразків. Навчання без нагляду займається дослідженням не розмічених даних, мережі. В даній роботі досліджуються моделі які навчаються під наглядом та класифікують зразки.

1.2 Математичний апарат нейронних мереж

1.2.1 Структура

Штучні нейронні мережі — це обчислювальні системи, натхнені біологічними нейронними мережами, які здатні навчатися, поступово покращуючи свої результати в процесі багаторазового розгляду прикладів, що використовуються для

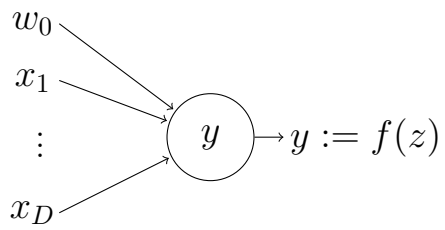

Рис. 1.1: Окремий нейрон та його компоненти. Функція активації позначена як f і застосовується до вхідних даних нейрона z формуючи його вихідний параметр $y = f(z)$. $x_1, \ldots, x_D$ представляє вхідні параметри інших вузлів мережі

розв'язання задач, які складно вирішити статичним набором інструкцій, таких як розпізнавання зображень, голосу, машинний переклад та інших. З математичної точки зору нейромережу можна розглядати як функцію $f(x)$, яка визначається як композиція інших функцій $g_i(x)$, які можна розкласти на інші функції.

Нейронна мережа складається з нейронів, які утворюють декілька шарів. Нейрон - це структурна одиниця, яка приймає один або декілька вхідних сигналів, опрацьовує їх, та формує вихідний сигнал, застосовує до *зваженої суми* вхідних сигналів певну функцію, яка посилює або послаблює його, і видає отриманий результат. В залежности від структури мережі, результат може видаватись назовні або іншому нейрону. Мережі, які складаються з багатьох шарів називають глибокими мережами. Як правило, вони мають вхідний та вхідний шари, та один або декілька проміжних (прихованих) шарів. В штучних нейронних мережах сигналами є числа. Функція яка виконується до сигналу називають функцією активації.

Розглянемо більше детально математичний апарат систем що здатні навчатися на основі штучних нейронних мереж. Такі нейронні мережі отримують вхідні дані X, та видають вектор ймовірностей Y. Ключовим елементом нейронної мережі є зв'язки між нейронами *weights*, кожен зв'язок має певну вагу, виражену числовим значенням. Саме у зв'язках зосереджуються "знання" мережі, які вона отримує з навчального матеріалу. Перед навчанням вагові коефіцієнти обираються випадково. Під час навчання, на кожній ітерації, вагові коефіцієнти коригуються в напрямку, при якому результат мережі є точнішим.

1.2.2 Поширення сигналів

Зазвичай нейронна мережа представляється у вигляді графу, в якому нейрони розміщуються на вузлах, а ваги зв'язків між ними розміщуються на ребрах.

В тренуванні нейронної мережі виділяють два процеси

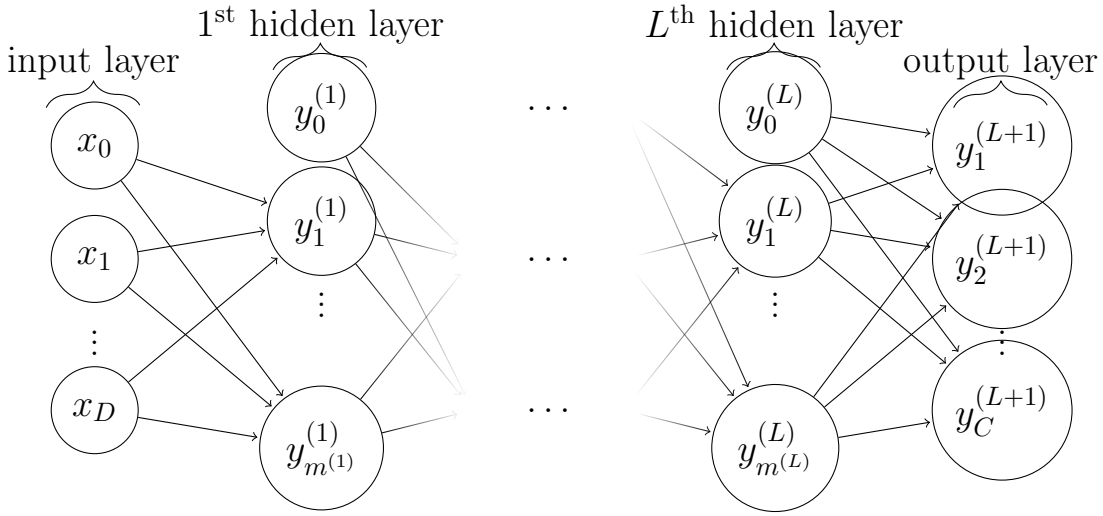

Рис. 1.2: Граф мережі з $(L+1)$ шарами, яка має D вхідних та C вихідних вузлів. $l^{\text{й}}$ прихований шар містить $m^{(l)}$ прихованих вузлів.

- *Feedforward* - Пряме поширення сигналу в напрямку від вхідного шару до вихідних. Мережі "показують" певний зразок, і оцінюють її результат. На цьому етапі сигнали проходять від вхідного шару, крізь приховані шари на вихідний шар.

- *Backpropagation* - Зворотне розповсюдження помилки. Результат мережі оцінюється у порівнянні з очікуваним результатом та обчислюється помилка. Зв'язки між нейронами, "відповідальними" за утворення помилки, коригуються відповідно до величини їх впливу на формування помилки.

Процес проходження сигналу через нейрон можна представити у вигляді функції, яка приймає вхідні параметри (x), ваги (w), похибку bias (b), та видає певний результат y. Ця функція складається з багатьох менших функцій - знаходження добутків вхідних параметрів та вагових коефіцієнтів, знаходження суми цих значень та похибки, застосування функції активації до отриманого результату. Формулу можна представити так

$$Atcivation(\sum[inputs * weights] + bias)$$

Таким чином, вхідним параметром функції активації є *Зважена сума* вхідних сигналів.

1.2.3 Функції активації

Функції активації застосовуються до вхідних параметрів нейрону або шару нейронів, та підсилюють або пригнічують вихідний сигнал. Надають мережі властивости нелінійності, забезпечують плавний перехід при зміні значень входу, тобто невелика зміна входу призводить до невеликої зміни виходу.

Розглянемо найбільш поширені функції активації.

Тотожна Функція яка переводить кожний елемент множини (області) визначення в себе.

$$f(x) = x$$

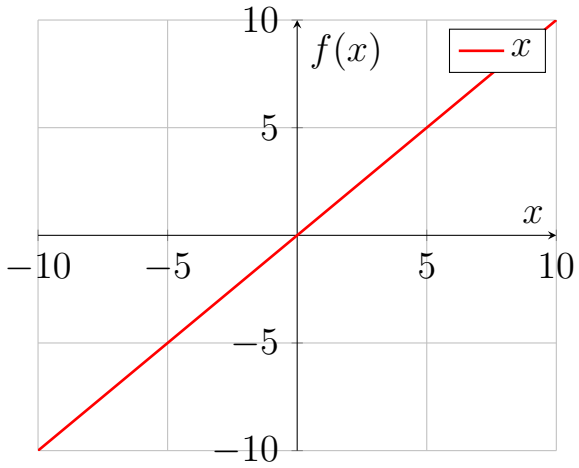

Рис. 1.3: Тотожність

Двійковий крок (функція Гевісайда) Розривна функція дійсної змінної, значення якої дорівнює 0 для від'ємних значень аргументу і рівне 1 для додатніх значень аргументу.

$$f(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x \geq 0 \end{cases} \tag{1.1}$$

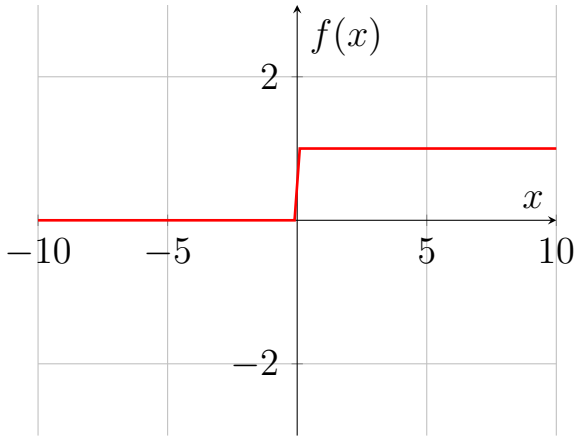

Рис. 1.4: Двійковий крок

Tanh

$$f(x) = \tanh x = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

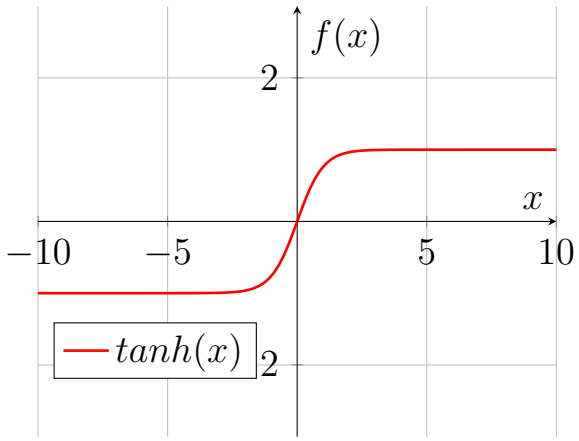

Рис. 1.5: Tanh

Сигмоїда - неперервно диференційована монотонна нелінійна S-подібна функція, яка часто застосовується для "згладжування" значень деякої величини.

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

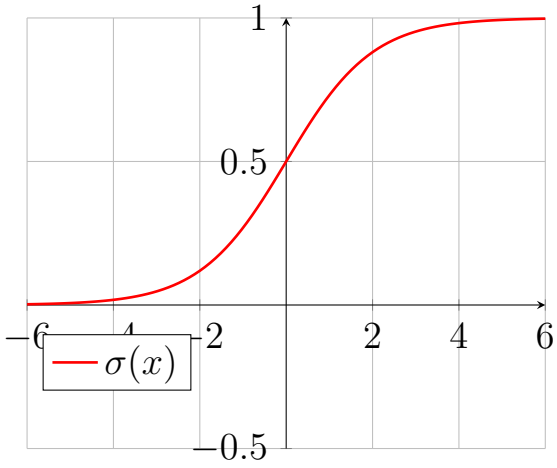

Рис. 1.6: Сигмоїда

Сигмоїда застосовується в нейронних мережах для того, щоб ввести деяку нелінійність в роботу мережі, але при цьому не дуже сильно змінити результат її роботи. [19]

ReLU (випрямлений лінійний вузол)

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x & x \geq 0 \end{cases} \tag{1.2}$$

Також цю функцію можна представити як

$$f(x) = max(x, 0)$$

Де x - вихідний результат нейрона.

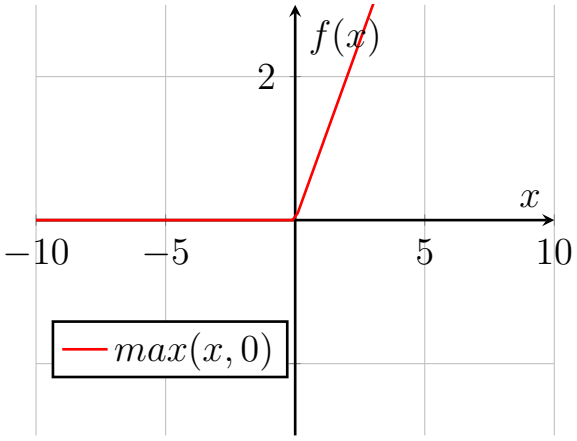

Рис. 1.7: Випрямлений лінійний вузол (ReLU)

Переваги ReLU:

- Біологічна правдоподібність: Одностороння на відміну від центрально симетричного гіперболічного тангенса.
- Розріджена активація: наприклад, у випадково ініціалізованій мережі, тільки близько 50% прихованих елементів активуються (мають не нульове значення).
- Краще градієнтне поширення: рідше виникає проблема зникання градієнту у порівнянні з сигмоїдальною передавальною функцією, яка може виникнути в обох напрямках.[7]
- Швидкість обчислення: тільки порівняння, додавання та множення.

Soft-max (нормована експоненційна функція) - приймає L-вимірний вектор z із довільними значеннями елементів та видає L-вимірний вектор $\delta(z)$ з дійсними значеннями елементів в області [0, 1] що в сумі дають одиницю. В нейронних мережах - використовується як функція активації на останньому шарі для категорійного розподілу, тобто розподілу ймовірностей при L різних можливих варіантах. [2]

$$S_{i,j} = \frac{e^{z_{i,j}}}{\sum_{l=1}^{L} e^{z_{i,j}}} \tag{1.3}$$

Результатом є вектор, що містить показники *впевненості* моделі про належність зразка щодо кожного класу від 0 до 1. Клас, до якого модель з найбільшою ймовірністю відносить зразок буде мати найбільший рівень впевненості.

1.2.4 Функції втрат

Функція втрат оцінює помилку моделі, втрата - це величина яка характеризує точність. В ідеалі вона повинна дорівнювати 0. Нейронні мережі які використовують функцію активації soft-max видають розподіл ймовірностей. Для порівняння істинної ймовірності та тієї, що видала нейронна мережа використовується поняття *Перехресної ентропії*

Втрата перехресної ентропії

$$L_i = -\sum y_{i,j} log(\widehat{y}_{i,j})$$

L величина втрати
i індекс класу
j індекс мітки
y очікуваний розподіл ймовірностей
$\widehat{y}$ розподіл ймовірностей виданий моделлю

Наприклад, якщо розглянути мережу, яка класифікує дані на три класи, результатом буде вектор з трьох значень $[0.6, 0.3, 0.1]$. Кожен елемент даного вектору показує впевненість моделі в тому класі за індексом цього елементу. Якщо зразок належить до першого класу, то очікуваним результатом є вектор $[1, 0, 0]$, тоді втрата перехресної ентропії в цьому випадку дорівнює

$$L_i = -\sum y_{i,j} log(\widehat{y}_{i,j})$$

$$= -(1 * log(0.6) + 0 * log(0.3) + 0 * log(0.1))$$

$$= -(-0.51082562377 + 0 + 0) = 0.51082562377$$

Таким чином, якщо впевненість моделі в тому що зразок належить до певного класу дорівнює 1, тобто модель на 100% впевнена в своїх результатах, втрата дорівнює 0. Величина втрати зростає у міру наближення впевненості моделі до 0.

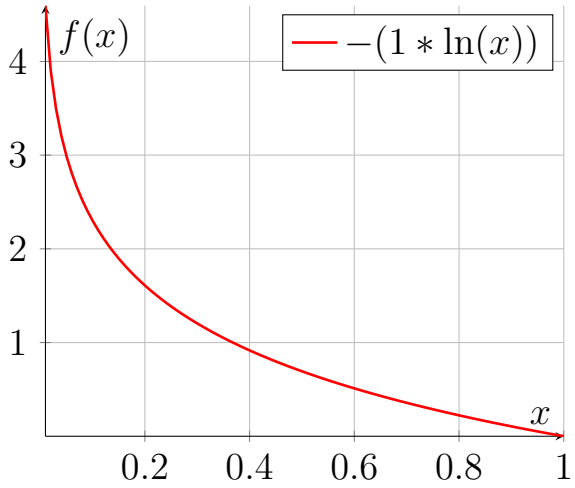

Рис. 1.8: Втрата перехресної ентропії. При абсолютно правильному результаті моделі (1) функція втрат дорівнює 0

1.2.5 Зворотне розповсюдження помилки

Коли величина втрат обчислена, необхідно змінити вагові коефіцієнти зв'язків між нейронами таким чином, щоб величина втрати зменшувалась. Окремі зв'язки

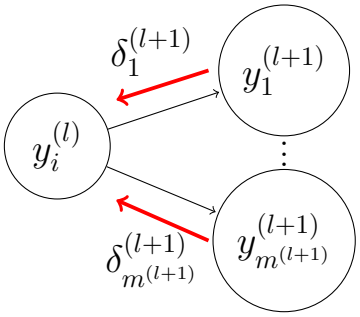

Рис. 1.9: Після оцінки помилки всіх вузлів поширюються в зворотньому напрямку

мають різний рівень впливу на величину втрати, тому цей рівень слід враховувати під час коригування вагових коефіцієнтів. Цей етап називається Зворотне розповсюдження помилки (*Backpropagation*) На цьому етапі оцінюється точність результату який видала нейромережа за допомогою функції втрат, яка порівнює наскільки точним був результат і як він відрізняється від бажаного результату. Формується градієнт - вектор, який містить часткові похідні функції втрат. Спершу обраховується часткові похідні функції втрат з врахуванням параметрів які впливали на результат.

1.2.6 Градієнт

Градієнтом скалярної функції f є вектор, компонентами якого є похідні всіх аргументів даної функії.

$$\vec{\nabla} f(x,y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Розглянемо обчислення градієнтного вектора на прикладі квадратичної функції.

$$f(x) = x^2$$

Похідна квадратичної функції дорівнює $\frac{\partial f}{\partial x} = 2x$. При вхідних даних: $x = [2, 3, 6]$, градієнтний вектор буде дорівнювати

$$\nabla_{f(x)} = [4, 6, 12]$$

В контексті нейронних мереж, градієнт - це напрямок, при якому невелика зміна вагових коефіцієнтів призводить до найшвидшого зростання функії втрат всім напрямкам. Кількість елементів градієнтного вектора дорівнює кількості вагових

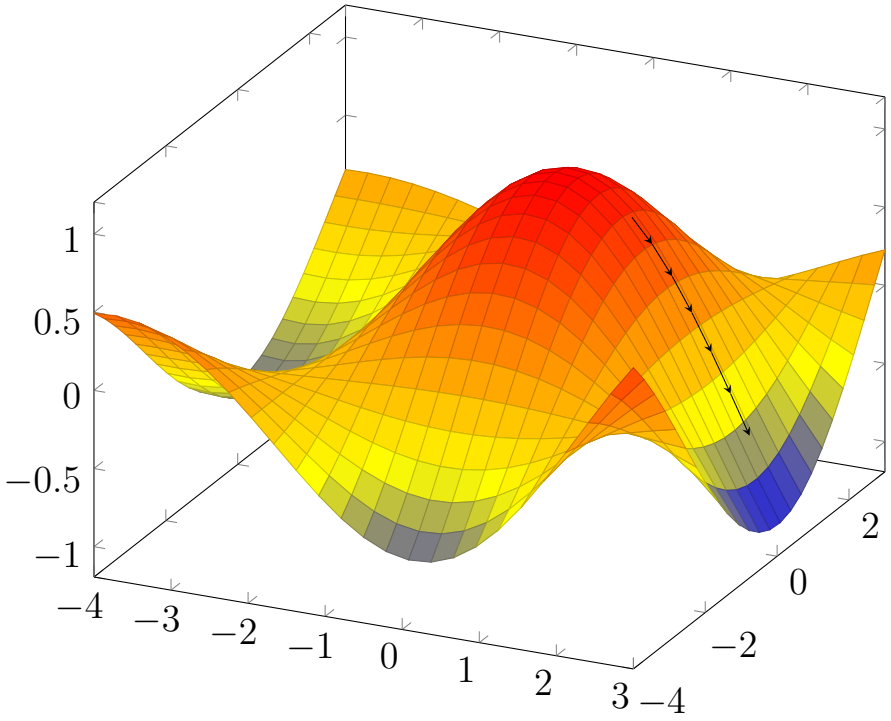

Рис. 1.10: Градієнтний спуск. Пошук області в якій значення функції втрат є найменшим

коефіцієнтів. Градієнт залежить від поточних значень вагових коефіцієнтів Під час процесу зворотного розповсюдження помилки вагові коефіцієнти невеликими кроками, розмір яких називають *learning rate* змінюються в напрямку, протилежному градієнту[3]. Даний спосіб оптимізації називається *градієнтним спуском*, він полягає в тому, що ваги зв'язків коригуються відповідно до величини свого впливу на функцію втрат. Кожен з вагових коефіцієнтів нейронної мережі отримує уточнення пропорційно до часткової похідної функції втрат по відношенню до поточної ваги на кожній ітерації навчання відповідно до правила диференціювання складної функції (ланцюгового правила):

$$\frac{d}{dx}f(g(x)) = \frac{d}{dg(x)}f(g(x)) * \frac{d}{d(x)}g(x) = f'(g(x)) * g'(x)$$

1.3 Змагальні атаки

Змагальні атаки - техніки направлені на порушення роботи моделі машинного навчання за допомогою модифікованих зразків [10]. Більшість методів машинного навчання були розроблені для роботи з конкретними наборами проблем, у яких тренувальні та тестові дані генеруються з одного статистичного розподілу. Коли ці моделі застосовуються до реального світу, ініціатори атак можуть підготувати дані, які порушують це статистичне припущення. Ці зразки можуть бути організовані, щоб використовувати конкретні вразливості та скомпрометувати результати моделі.[12]. Хоча змагальні атаки є специфічними до класифікатора, певним чином, вони узагальнюються на різні моделі[24].

Ініціатор атаки намагається знайти такі вхідні дані X^* які будуть призведуть до хибної класифікації. Серед дослідників даної проблематики, щодо таких даних вживається термін *змагальний зразок*.

Автори статті *Adversarial Examples Are Not Bugs, They Are Features* висловлюють думку що механізм змагальних атак полягає в тому, що під час класифікації нейромережі оцінюють мікроскопічні ознаки, які людина не здатна сприймати. Змагальні зразки маніпулюють цими ознаками, збільшуючи помилку класифікатора, залишаючи образ не змінними з точки зору людини. Робастна модель повинна не повинна опиратися на такі ознаки. [9]

Змагальні атаки можуть переслідувати такі цілі:

1. **Зниження впевненості** - знизити коефіцієнти впевненості моделі в результатах класифікації, понизити довіру до моделі.

2. **Похибка класифікації** - змінити результат моделі таким чином, щоб вхідне зображення було класифіковано як будь-який інший клас відмінний від класу, до якого належить оригінальне зображення.

3. **Направлена похибка класифікації** - створити зразки вхідних даних, щоб вхідне зображення було класифіковано як певний клас. Прикладом може бути створення набору цяток або зображення шуму, класифікованих як цифра. [15]

4. **Направлена похибка класифікації вхідних даних** - модифікувати зображення, що належить до класу відомого моделі таким чином, щоб в ре-

зультаті модель розпізнавала його як певний інший клас. Наприклад змінити зображення цифри таким чином, щоб воно розпізнавалось як інша цифра.

1.3.1 Механізми змагальних атак

Загальна ідея змагальних атак полягає в максимізації функції втрат моделі з урахування вхідного зображення. У випадку направленої атаки, мінімізується функція втрат по цільовій мітці. В більшості атак певною мірою виконується процес протилежний градієнтному спуску під час якого вхідне зображення модифікується так щоб результати мережі були хибними. Це схоже на процес зворотного розповсюдження помилки, коли модифікуються вагові коефіцієнти моделі, з різницею що в випадку змагальної атаки модифікуються вхідні дані. В залежно від обсягу модифікацій виділяють такі категорії атак

- L_0 Атаки Змінюється мінімальна кількість ознак (пікселів)
- L_2 Атаки Змінюється значна кількість ознак (пікселів)
- L_∞ Атаки Змінюється велика кількість пікселів, може бути модифікований кожен піксель

1.3.2 White box атаки

White box атаки - вид змагальних атак при якому ініціатор має певний обсяг інформації про повну інформацію про модель та її будову. В даній роботі розглядаються атаки, що проводяться під час фази тестування мережі. Втручання в процес навчання не досліджується. В залежності від обсягу інформації якої володіє ініціатор атаки, можна виділити такі ситуації:[16]

- **Ініціатор володіє тренувальними даними та архітектурою моделі** В цій ситуації ініціатор має повну інформацію про нейронну мережу F та тренувальні дані T, що включає кількість шарів, функції активації, матриці вагових коефіцієнтів та похибок, функії втрат.
- **Ініціатор володіє архітектурою моделі** В цій ситуації ініціатор має інформацію про архітектуру мережі F та її параметри. В такому випадку ініціатору відомо, скільки шарів має мережа, які функції активації та втрат використовуються.

- **Ініціатор володіє тренувальними даними** В цьому випадку ініціатор здатний зібрати сурогатний набір даних із того самого розподілу, що й оригінал набір даних, що використовується для навчання мережі. Однак ініціатор не має знань про архітектуру, яка використовується для проєктування нейронної мережі. Таким чином, типові атаки, проведені в цій моделі, швидше за все, включають навчальні архітектури глибокого навчання, які зазвичай застосовуються використання сурогатного набору даних для апроксимації вивченої моделі за законним класифікатором.

- **Оракул** В цьому випадку ініціатор має здатність використовувати нейронну мережу (або її проксі-сервер) як «оракул». Ініціатор може отримати класифікацію вихідних даних із наданих вхідних даних (подібно атак на основі підібраного відкритого тексту). Це дозволяє диференційні атаки, коли ініціатор може відстежити кореляції між змінами у вхідних даними та вихідним результатом.

- **Зразки результатів моделі** В цій ситуації ініціатор атаки має пари вхідних даних та результатів їх класифікації. Але він не має змоги модифікувати їх та спостерігати відмінності в результатах моделі

В основі більшості атак цього виду використовується градієнтний спуск, під час якого ініціатор атаки намагається мінімізувати певну функцію, яка виражає дистанцію між валідними вхідними даними x та змагальним зразком x', в результаті модель класифікує зразок x' як клас x

Fast gradient sign

Fast Gradient Sign - L_∞ Атака яку запропонував Christian Szegedy[23], найшвидший, але досить ефективний метод змагальних модифікацій.

При цьому методі атаки обраховується функція втрат для вхідних даних. Градієнт розповсюджується з вихідного шару на вхідні дані, які модифікуються в напрямку протилежному градієнту що збільшує помилку моделі.

$$x_{adv} = x + \epsilon * sign(\nabla_x \ell(f(x), y_{true}))$$

Fast Gradient Sign Method, де

f нейронна мережа
ℓ втрата перехресної ентропії

Розглянемо детальніше процес формування змагального зображення. Спочатку виконується класифікація оригінального зразка, отримується результат sofmaxt функції, та обчислюється функція втрат від оригінальної мітки $\ell(f(x), y_{true})$. Далі отримується градієнтний вектор функції втрат $\nabla_x \ell(f(x), y_{true})$ і зображення змінюється в напрямку, в якому функція втрат росте. До градієнтного вектора застосовується операцію $sign$, яка поелементно видає знак числа у вхідному векторі. Якщо значення елементу від'ємне, то елемент вектору буде -1, якщо додатне до 1, якщо 0, то 0. Отриманий вектор поелементно перемножується на певну величину ϵ. Таким чином отримується вектор з шумом, який додається до оригінального зразка x. Існує варіант направленої атаки, коли замість мітки оригінального класу береться мітка іншого цільового класу, за який ми хочемо видати зображення, і береться напрямок не довільний напрямок в якому максимізується функція втрат відповідно до оригінального класу, як раніше, а той, в якому мінімізується функція втрат відносно цільової мітки.

Basic Iterative Method

Iterative Gradient Sign або Basic Iterative Method (BIM) це Покращена версія Fast gradient sign була запропонована [11], яка виконується ітеративно. Полягає в тому, що замість одного кроку в напрямку градієнту розміру ϵ здійснюється багато кроків розміру α. На кожній ітерації перевіряється, що модифіковане зображення отримане в результаті атаки все ще знаходиться в межах дистанції ϵ від оригінального.

$$x_{adv_{t+1}} = clip_{\epsilon,x}(x_{adv_t} + \alpha * sign(\nabla_x \ell(f(x_{adv_t}), y_{true})))$$

Basic Iterative Method, де

f нейронна мережа
ℓ втрата перехресної ентропії
x_{adv_0} x - оригінальний зразок

Метод Iterative Gradient Sign показав вищу ефективність в порівнянні з Fast Gradient Sign[11]

Jacobian-based Saliency Map Attack (JSMA)

L_0 Атака, яку запропонував Nicolas Papernot [16]. Використовує матрицю Якобі - матрицю яка містить комбінації похідних функції по відношенню до її аргументів

$$f(x,y) = \begin{bmatrix} f_1(x,y) \\ f_2(x,y) \end{bmatrix}$$

$$J_{f(x,y)} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix}$$

Через матрицю похідних можна визначити, ознаки (пікселі) які мають найбільший вплив на вибір класу мережею. Обчислюється так звана "карта помітності" *(saliency map)*, яка характеризує наскільки зміна значення даного пікселя збільшує впевненість мережі щодо цільового класу та зменшує ймовірності всіх інших класів і визначається як:

$$S(x,t)[i] = \begin{cases} 0, & \text{if } \frac{\partial F_t(x)}{\partial x_i} < 0 \text{ або } \sum_{j \neq t} \frac{\partial F_j(x)}{\partial x_i} > 0 \\ \frac{\partial F_t(x)}{\partial x_i} \left| \sum_{j \neq t} \frac{\partial F_j(x)}{\partial x_i} \right|, & \text{інакше} \end{cases}$$

$\frac{\partial F_t}{\partial x_i}$ та $\sum_{j \neq t} \frac{\partial F_j(x)}{\partial x_i}$ вказують на скільки функція $F_t(x)$ зростає і зменшується $\sum_{j \neq t} F_j(x)$ при модифікації ознаки x_i. Атака здійснюється ітеративно, на кожній ітерації модифікуються максимально помітні пікселі.

Ця атака працює досить ефективно на зображення невеликого розміру, але вимагає значних ресурсів пам'яті для роботи з великими зображеннями.

Алгоритм 1 Алгоритм атаки JSMA [16] де $\nabla F(X)$ - матриця Якобі, Γ - ознаки в просторі пошуку, t - цільовий клас

Input $\nabla F(X), \Gamma, t$

for each pair$(p, q) \in \Gamma$ **do**
  $\alpha = \sum i = p, q \frac{\partial F_t(X)}{\partial X_i}$
  $\beta = \sum i = p, q \sum j \neq t \frac{\partial F_j(X)}{\partial X_i}$
  if $\alpha > 0$ and $\beta < 0$ and $-\alpha \times \beta > max$ **then**
    $p_1, p_2 \leftarrow p, q$
    $max \leftarrow -\alpha \times \beta$
  end if
end for
return p_1, p_2

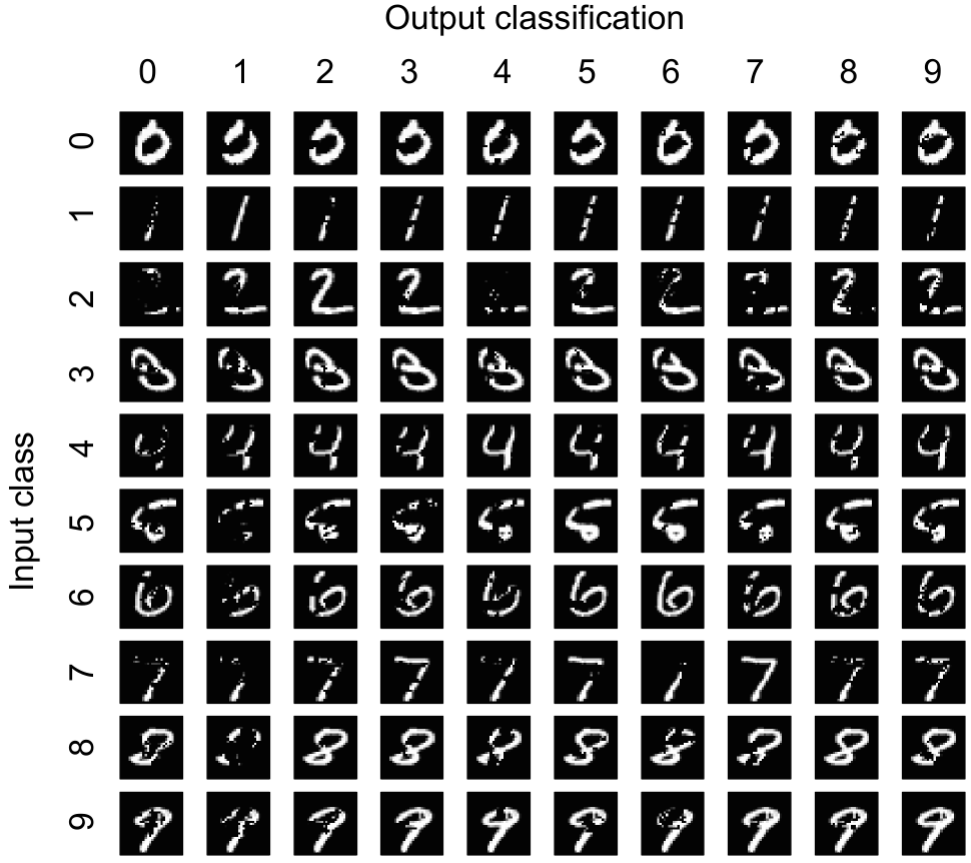

Рис. 1.11: Результат атаки JSMA [16]

Атака Carlini Wagner

Дана атака розширює з оригінальну задачі оптимізації з обмеженням: [23]

$$\min ||\delta||_2^2$$

$$s.t. f(x+\delta) = T$$

$$x + \delta \in |0,1|^n$$

де, T - цільовий клас.

Функція яка оптимізується під час атаки:

$$\min_{\delta} ||\delta||_2^2 + c * \max\{\max\{Z(x+\delta)_i | i \neq T\} - Z(x+\delta)_T, -k\}$$

Де Z - це вхідні дані softmax шару та k - гіперпараметри. З урахуванням обмеження $x + \delta \in |0,1|^n$ відбувається заміна змінної та оптимізується w

$$\min_{\delta} ||\delta(w)||_2^2 + c * \max\{\max\{Z(x+\delta(w))_i | i \neq T\} - Z(x+\delta(w))_T, -k\}$$

де

$$\delta(w) = \frac{1}{2}(\tanh(w) + 1) - x$$

Дана функція мінімізується алгоритмом оптимізації Адам та отримується змагальна модифікація

$$x_{adv} = x + \delta * (w_{min})$$

На кожній ітерації алгоритм атаки обирає пікселі, які не мають значного впливу на результат моделі, і "запам'ятовує" їх, щоб надалі їх значення не змінювалось. Набір таких пікселів збільшується до тих пір, поки не залишається мінімальна підмножина пікселів, які можна змінити для створення змагального зразка. δ - модифікація зразка x, при тому, що $x + \delta$ - змагальний зразок.

Це найефективніша атака, спрацьовує в більшості випадків з прийнятними затратами по часу та обчислювальних ресурсах навіть з великими зображеннями.

Deepfool

Deepfool - [14] - не направлена L_2 атака. Є більш ефективною ніж Fast Gradient Sign, та генерує змагальні зразки ближчі до оригінальних.

Розглянемо детальніше алгоритм даної атаки:

1. Алгоритм використовує вхідне зображення x та модель $f(x)$

2. Результатом алгоритму є змагальний зразок $\widehat{r}$

Алгоритм 2 Алгоритм атаки Depfool

Input Зразок x, нейромережа f
Otput Змагальний зразок $\widehat{r}$
Initialize $x_0 \leftarrow x, i \leftarrow 0$
while $\widehat{k}(x_i) = \widehat{k}(x_0)$ **do**
    for $k \neq k(x_0)$ **do**
        $w'_k \leftarrow \nabla f_k(x_i) - \nabla f_{\widehat{k}(x_0)}(x_i)$
        $f'_k \leftarrow f_k(x_i) - f_{\widehat{k}(x_0)}(x_i)$
    end for
    $\widehat{l} \leftarrow \arg\min_{k \neq \widehat{k}(x_0)} \frac{|f'_k|}{||w'_k||_2}$
    $r_i \leftarrow \frac{|f'_{\widehat{l}}|}{||w'_{\widehat{l}}||_2^2} w'_{\widehat{l}}$
    $x_{i+1} \leftarrow x_i + r_i$
    $i \leftarrow i + 1$
end while
return $\widehat{r} = \sum_i r_i$

3. Змагальний зразок ініціалізується оригінальним зображенням, ініціалізується змінна циклу i

4. Запускається цикл з умовою зупинки, коли передбачення моделі не дорівнює оригінальній мітці

5. Беруться n класів, які мають найбільшу ймовірність після оригінального класу і зберігається мінімальна різниця між градієнтами функції втрат по оригінальній мітці та мітками найближчих класів w_k.

6. У внутрішньому циклі обчислюється мінімальні значення w_k та f_k.

7. Використовуючи w_k та f_k. знаходиться найближча до x гіперплощина

$$\widehat{l} \leftarrow \arg \min_{k \neq \widehat{k}(x_0)} \frac{|f'_k|}{||w'_k||_2}$$

де змінні що починаються з f - мітки класів, змінні що починаються w градієнти, k - вказує на класи з найбільшою ймовірністю після оригінального класу $\widehat{k}(x_0)$

8. Обчислюється вектор модифікацій, який переводить x до гіперплощини, обчисленої на попередньому кроці

9. Модифікація додається до зображення і перевіряється результат моделі по ньому
10. Збільшується лічильник циклу
11. Повертається остаточна модифікація, яка є сумою всіх згенерованих модифікацій

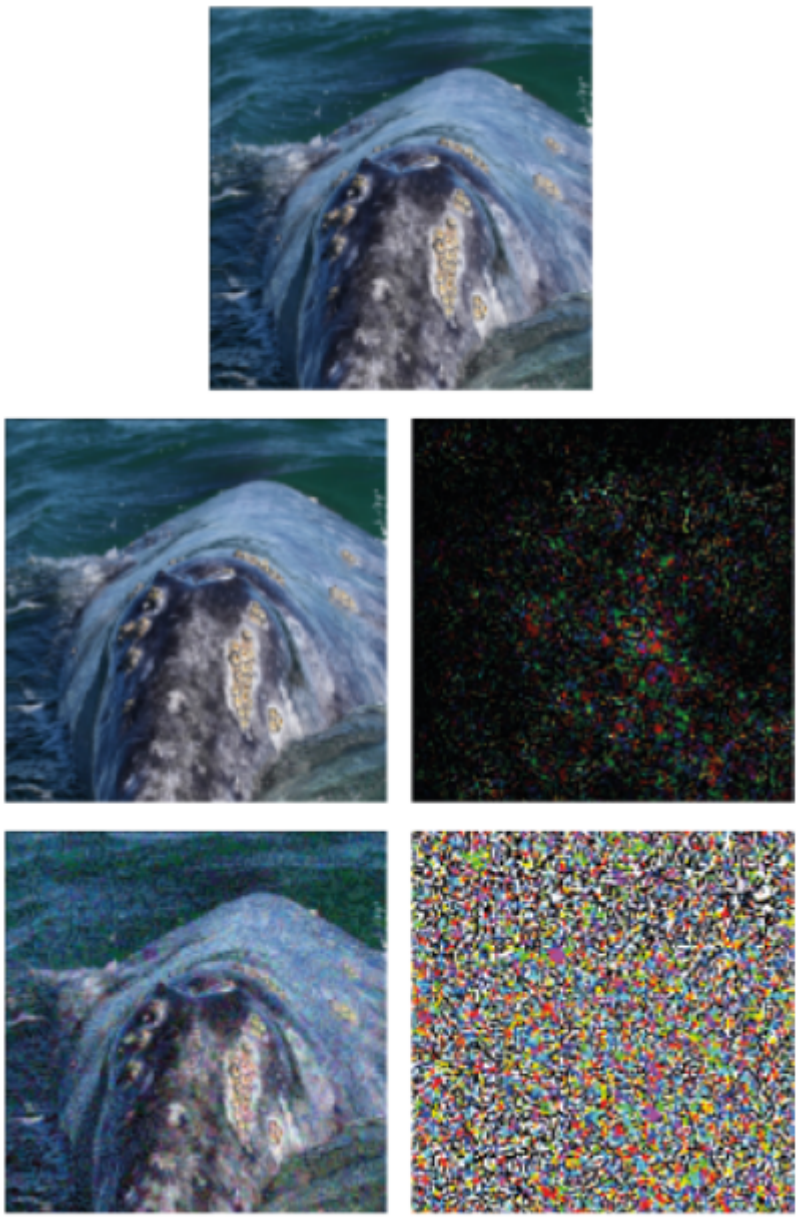

Рис. 1.12: Приклад змагальних модифікацій. Перший ряд: оригінальне зображення x, яке класифікується моделлю як "кит". Другий ряд: зображення $x + r$ класифікується як "черепаха", модифікація r згенерована алгоритмом Deepfool, Третій ряд: зображення $x + r$ класифікується як "черепаха", модифікація r згенерована алгоритмом Fast gradient sign. В першому випадку помилка досягається з меншим обсягом модифікацій[14]

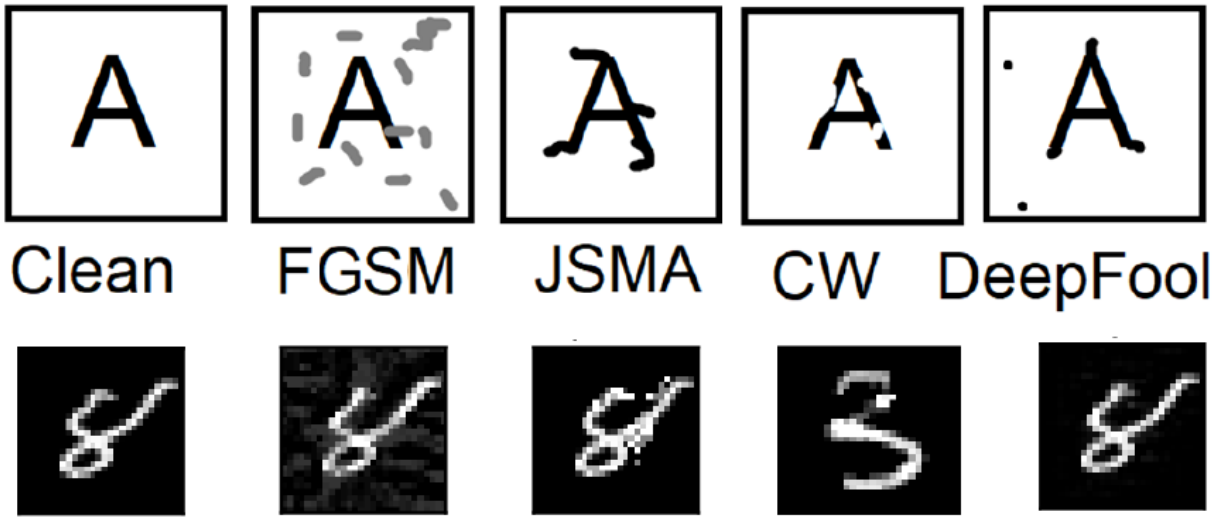

Рис. 1.13: [8]Верхній ряд демонструє модифікації оригінального зображення (для більшої наочності ефект посилений). Цей ряд є відтворенням спостережень і не є реальним зразком. Нижній ряд демонструє реальні атаки на зображення з каталогу MNIST, при яких цифра 8 класифікується як інша

1.3.3 Black box атаки

При цьому виді атаки, ініціатор не має доступу до внутрішньої структури моделі.

Атака одного пікселя

Атака одного пікселя це атака заснована на методі Диференціальної еволюції[22]. Диференціальна еволюція - алгоритм оптимізації, при якому спочатку генерується множина векторів, так зване покоління. Під векторами розуміються точки n-вимірного простору, в якому визначена цільова функція $f(x)$, яку потрібно мінімізувати. На кожній ітерації алгоритм генерує нове покоління векторів, випадковим чином комбінуючи вектори з попереднього покоління. Число векторів в кожному поколінні одне і те саме і є одним з параметрів методу.

Нове покоління векторів генерується в такий спосіб. Для кожного вектора x_i зі старого покоління обираються три різні випадкових вектори v_1, v_2,v_3, за вийнятком самого вектора x_i, і генерується так званий *мутантний* вектор за формулою:

$$v = v_1 + F * (v_2 - v_3)$$

Над мутантним вектором v виконується операція "схрещування" (*crossover*), яка полягає в тому, що деякі його координати заміщуються відповідними координатами з початкового вектора x_i (кожна координата заміщається з деякою ймовірністю, яка також є ще одним з параметрів цього методу). Отриманий після схрещування вектор називається *пробним вектором* (*trial vector*). Якщо він виявляється кращім за вектор x_i (тобто значення цільової функції стало меншим), то в новому поколінні вектор x_i замінюється на пробний вектор, а в іншому разі - залишається x_i.

Вхідний зразок представляється як вектор, кожен елемент якого представляє один піксель. Нейронна мережа f отримує n-вимірний вектор вхідних даних - $x = (x_1, ..., x_n)$, який коректно класифікується як клас t. Ймовірність того, що x належить до класу t - $f_t(x)$. Вектор $e(x) = (e_1, ..., e_n)$ - змагальний зразок на основі x, цільовий клас adv і обмеження на максимальну модифікацію L. Таким чином задача ініціатора атаки максимізувати функцію $f_{adv}(x+e(x))$ при $e(x) \leq L$ При проведенні атаки генеруються покоління так званих кандидатів - структури що містять певну кількість модифікацій. Кожна модифікація містить координати пікселя та RGB значення. На кожній ітерації атаки обчислюється

$$x_i(g+1) = x_{r1}(g) + F(x_{r2}(g) - x_{r3}(g)),$$

$$r1 \neq r2 \neq r3$$

де x_i елемент вектору кандидатів, $r1, r2, r3$ - випадкові числа, F - параметр масштабу, g - індекс поточного покоління. Після генерації кандидати порівнюються з попередниками, відповідно до індексу і той при якому ймовірність цільового класу зростає, відбираються до наступного покоління. Еволюція триває 100 поколінь, або припиняється якщо ймовірність цільового класу зростає до 90%.

Рис. 1.14: Результат атаки з застосуванням методу диференціальної еволюції [22]

1.4 Робастність

Робастність це здатність мережі правильно класифікувати змагальні зразки. Універсальної метрики для оцінки робастності моделі не існує. Як правило, в дослідженнях проводиться ряд атак на мережу і визначається процент успішних атак при дотриманні певної дистанції.

Автори атаки Deepfoll [14] пропонують метрику для оцінки робастності до змагальних модифікацій класифікатору f, при якій обчислюється мінімальна кількість модифікацій вхідного зразка яка призводить до помилки класифікації $\widehat{P}_{adv}(f)$, яка визначається як

$$\widehat{P}_{adv}(f) = \frac{1}{|y|} \sum_{x \in y} \frac{||\widehat{r}(x)||_2}{||x||_2}$$

де $\widehat{r}(x)$ - мінімальна кількість модифікацій, y - тестовий набір даних.

Задача оцінки робастності ускладнюється відсутністю метрики, впливу модифікацій на зміст вхідних даних, та їх сприйняття людиною.

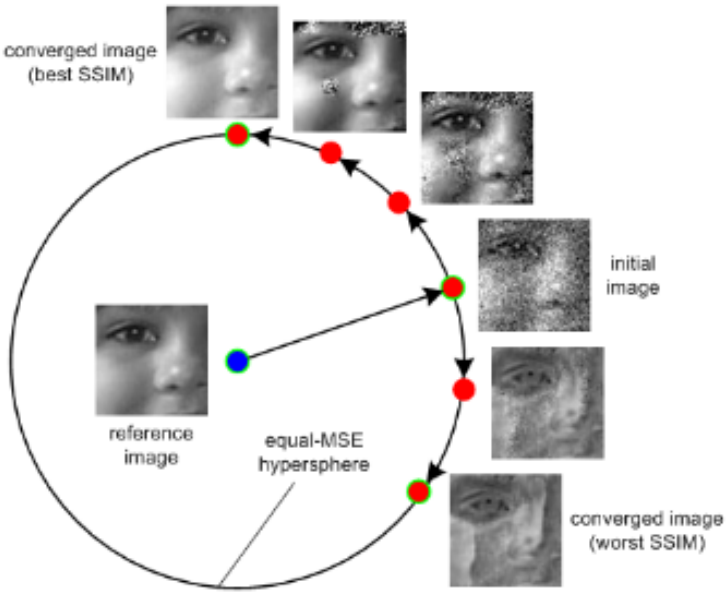

Рис. 1.15: Зображення, які однаково віддалені від еталонного зображення за l_2 метрикою, можуть бути дуже віддалені за рівнем сприйняття людиною. Ілюстрація Wang та Bovik[26]

1.5 Методи захисту

Було запропоновано декілька методів захисту від змагальних атак:

- **Adversarial training**- додавання змагальних зразків до набору тренувальних даних. Таким чином, модель може коректно класифікувати змагальні зразки. Це найпростіший підхід, його недоліком є суттєве зниження точності моделі. Дослідження на наборі CIFAR показали зниження точності моделі з 95% до 87% при застосуванні техніки Adversarial training. Ще одним недоліком даного методу є необхідність повторювати тренування при появі нових видів атак, це значно ускладнює застосування цього методу.

- **Попередня обробка зображень**- набір статичних функцій обробки зображень перед класифікацією (видалення шуму)[17].

- **Defensive distillation** - полягає в комбінації двох моделей. Друга модель використовує знання першої моделі для покращення точності. Black-box атаки з використанням генетичних алгоритмів та атаки Carlini Wagner довели неефективність даного методу захисту.

- **Застосування еволюційних алгоритмів** для виявлення оптимальних технік попередньої обробки зображення[17]

Дані методи захисту мають деякі недоліки та обмеження, та з легкістю обходяться останніми атаками.[8]

1.6 Підсумки

В цьому розділі було розглянуто математичний апарат моделей машинного навчання, їх функціональні особливості, які обумовлюють вразливості до атак, основні види змагальних атак та методи захисту від них. Дослідження показують, що для класифікації об'єкту на зображенні, глибокі нейронні мережі, зазвичай, не ідентифікують макроознаки об'єкта. Наприклад, різниця між літаком та машиною не залежить від фону зображення, або наявності крил чи вікон. Результат класифікації більш залежить від величин кольорових складових пікселів в певних позиціях. Цей факт обумовлює ненадійність результатів моделі при незначних модифікаціях вхідних даних. Незважаючи на важливість цього явища, на даний час не існує ефективних методів захисту від змагальних атак, що вимагає подальшого дослідження цієї теми.

Розділ 2

Постановка задачі та методи дослідження

2.1 Мета роботи

В даній роботі пропонується розробка інтерактивної системи для демонстрації змагальних атак на моделі машинного навчання для розпізнавання зображень.

Розроблена система повинна виконувати такі функції:

- Проведення серії атак на існуючу модель згідно з налаштуваннями користувача
- Формування звіту на основі проведених тестів

Розроблена система має відповідати таким вимогам:

- Універсальність
  - у користувачів системи має бути можливість проводити тестування на існуючих моделях з різною топологією
  - система повинна зберігати результати тестування в рамках проєкту
  - у користувача повинна бути можливість проводити серії тестів на моделі в рамках проєкту, змінюючи параметри моделі при проведенні тестів
- Інтерактивність
  - система має надати можливість користувачу спостерігати модифікації вхідних зразків

- у користувача повинна бути можливість візуалізації атак
- у користувача повинна бути можливість змінювати параметри атак (інтенсивність модифікації, кількість ітерацій)

Проведення подібних тестів потребує написання додаткового коду, універсальної системи здатної оцінити робастність моделі машинного навчання на даний час не існує. Тому дана система буде корисною для розробників та користувачів нейронних мереж з використанням tensorflow, з високими вимогами до робастності.

2.2 Огляд існуючих праць та проєктів

Існує декілька проєктів пов'язаних з тематикою роботі, які демонструють проведення змагальних атак.

advis.js [13] - інтерактивна демонстрація атаки методом Fast Gradient Sign з можливістю коригування параметру інтенсивності модифікації. Демонстрація проводиться на моделі Mobilnet, у користувача є можливість завантажити зображення, застосувати змагальну атаку та спостерігати зміну результатів моделі. Є візуалізація ділянок зображення які мають вирішальне значення для розпізнавання класу у вигляді теплокарти.

https://krasch.io/adversarial-tensorflowjs/ [18] - дана робота присвячена дослідженню методу Targeted Fast Gradient Sign. Містить код даної атаки, та її демонстрацію на моделі Mobilnet

adversarial.js [21] - web додаток, який містить API tensorflow для проведення деяких видів атак та сторінку з демонстрацію поширених атак.

- Fast Gradient Sign Method L_∞
- Basic Iterative Method L_∞
- Jacobian-based Saliency Map Attack L_0
- Jacobian-based Saliency Map Attack, One-Pixel L_0
- Carlini & Wagner L_2

На сайті **adversarial.js** демонструються робота бібліотеки з такими моделями

- MNIST (розпізнавання рукописних цифр)

- GTSRB (розпізнавання дорожних знаків)
- CIFAR-10 (розпізнавання об'єктів на зображеннях невеликого розміру)
- ImageNet (розпізнавання об'єктів на зображеннях великого розміру)

В даних роботах проводиться візуалізація атак, але відсутня можливість проведення тестів на користувацьких моделях, розроблених для специфічних задач та обчислення середньої робастності. Тому, розробка універсальної інтерактивної системи тестування робастності моделей машинного навчання є доцільною.

Розділ 3

Моделювання системи оцінки робастності

3.1 Вибір інструментів для розробки

При моделюванні планується розробити систему яка дозволяє перевірити нейронну мережу для розпізнавання зображень на стійкість до відомих змагальних атак, найбільш зручним та потужним інструментом для цього є бібліотека tensorflow.js.

3.2 Основні відомості про Tensorflow.js

TensorFlow.js — це бібліотека JavaScript з відкритим вихідним кодом з апаратним прискоренням для навчання та розгортання моделей машинного навчання. TensorFlow.js надає гнучкі та інтуїтивно зрозумілі API для створення моделей з нуля за допомогою низькорівневої бібліотеки лінійної алгебри JavaScript або API шарів високого рівня. Моделі створені в TensorFlow.js можуть працювати в браузері або в середовищі виконання Node.js. В TensorFlow.js є можливість трансформувати існуючі моделі TensorFlow написані на інших мовах програмування, та використовувати їх в браузерах, мобільних пристроях або Node.js, перенавчати їх за допомогою даних датчиків, підключених до браузера, або інших даних на стороні клієнта.

Отже, для розробки системи оцінки робастності бібліотека tensorflow.js є найбільш вдалим вибором через ряд причин:

- Дозволяє проводити тренування та тестування нейронних мереж
- Працює в браузерах та Node.js
- Використовує GPU-прискорення (WebGL в браузері та ядра CUDA в Node.js)
- Підтримує архітектуру моделей нейронних мереж на JavaScript
- Забезпечує серіалізацію та десеріалізацію моделей
- Підтримує конвертацію моделей з фреймворків машинного навчання на Python
- Містить вбудовану систему вводу/обробки даних та API візуалізації
- Має значну схожість з Tensorflow та Keras, код досить легко переноситься на інші мови програмування

Назва *Tensorflow* відображає процес який відбувається всередині програми, написаної з використанням бібліотеки: дані вигляді так званих *тензорів*, проходять через шари нейронної мережі та інші вузли обробки, внаслідок чого відбувається навчання моделей.[1].

3.2.1 Тензори

Тензор - це багатовимірний масив в термінології спеціалістів по обчислювальній техніці. Тензор - ключова структура даних в tensorflow, яка є генералізацією векторів, матриць, та багатомірних масивів. В глибоких нейронних мережах всі дані та результати обчислень представляються у вигляді тензорів. Наприклад зображення у відтінках сірого можна представити у вигляді двомірного тензора, кольорове зображення зазвичай представляється у вигляді тримірного тензора. Звук, відео текст та інші види даних також можна представити у вигляді тензорів. Будь-який тензор має дві характеристики: тип даних (наприклад **int32** або **float32**) та форму. Форма являє собою розмір тензору по кожному виміру. Наприклад форма двомірного тензора може бути [128, 256]. Тензори виконують роль контейнерів для організації даних в структури, зручні для паралельної обробки на графічних процесорах, що значно прискорює роботу з ними. [3]

3.3 Збір вхідних даних для тестів

Для проведення тестів моделі, в систему необхідно буде внести такі дані: для не направлених атак: модель, зразок, оригінальні мітки. Для направлених атак: модель, зразок, оригінальні мітки, цільова мітка. Модель має бути типу `tf.LayersModel`, останній шар має мати тип `tf.layers.softmax`

Метод `tf.loadLayersModel()` завантажує модель, десеріалзує збережену топологію моделі з файлу в `model.json`. Потім цей метод зчитує двійкові дані значень вагових коефіцієнтів з файлу `weights.bin` за допомогою їх опису з файлу `model.json`. За бажанням можна вивести стислу інформацію про топологію моделі.

Отже, для проведення тестування робастності моделі нейронної мережі користувачу необхідно буде завантажити в систему наступні файли:

1. файл моделі `model.json`
2. файл вагових коефіцієнтів `weights.bin`
3. файл з даними для тестування `sample_x.json`
4. файл з мітками `sample_y.json` містить мітки у вигляді розподілу
5. файл з назвами класів `class_names.json`

3.4 Метрики

Метрики за якими будуть оцінюватись змагальні зображення

- Відстань Чебишова

$$l_\infty(\vec{x}, \vec{y}) = \max_{i=1,\ldots,n} |x_i - y_i|$$

  Метрика максимуму або l_∞ - метрика на векторному просторі, яка визначає відстань між двома векторами як найбільшу різницю їхніх координат.[4] Названа на честь російського математика Пафнутія Чебишова.

- Евклідова відстань

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

- Пікове співвідношення сигналу до шуму. (*Peak signal-to-noise ratio)*

  Позначається абревіатурою PSNR і є інженерним терміном, що означає співвідношення між максимальною силою сигналу та шуму, що спотворює значення сигналу. PSNR зазвичай вимірюється логарифмічною шкалою в децибелах. PSNR часто використовують для оцінки спотворень при стисканні зображень. PSNR визначається через середньоквадратичне відхилення (СКВ або MSE (англ. mean square error)), яке для двох монохромних зображень I та K розміру m×n, одне з яких вважається зашумленими наближенням іншого, обчислюється наступним чином:

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

Розділ 4

Реалізація інформаційної технології

4.1 Розробка проекту

В якості фреймворку для побудови інтерфейсу програми обрано Angular - фреймворк на мові TypeScript з відкритим кодом, який розвивається за підтримки компанії Google та спільноти. Angular дуже потужний фреймворк, який дозволяє розробляти складні web програми, він має підтримку великих компаній та популярний серед спільноти розробників. До його переваг можна віднести: повторне застосування коду за рахунок компонентної архітектури розмежування логіки програми від користувацького інтерфейсу, можливість впровадження модульних тестів, легке сприйняття коду.

В програмі реалізовані наступні класи:

- `Attacks` - статичний клас що містить алгоритми атак. Кожна атака представлена функцією яка приймає об'єкт `LayersModel`, вхідний зразок та конфігурацію атаки. Для варіанту націленої атаки, додається мітка цільового класу. Результатом атаки є об'єкт `AttackResult` якій містить два тензори - модифіковане зображення та самі модифікації.

- `TestContext` - містить налаштування тестової сесії, конфігурації атак, зберігає результати тестів

- `TestCase` - результат окремої атаки, містить вхідний та модифікований зразки, результати моделі по ним, та дистанції між ними

Рис. 4.1: Діаграма класів проєкту

- `AttackSummary` - містить метрики по окремому методу атаки, (рівень успішності атаки, середні відстані між оригінальними та змагальними зразками)
- `AttackStatus` - статус атаки
  - `Failed` - атака не успішна. Модель розпізнала зразок як правильний клас.
  - `Succeeded` - атака успішна, помилка моделі.
  - `PartialySucceeded` - атака вдалась частково. Варіант націленої атаки, при якому модель розпізнала вхідний зразок не як правильний клас, але і не як цільовий клас.

4.2 Інтерфейс системи

У взаємодію користувача з системою можна виділити такі етапи

- Імпорт моделі та тестових даних в систему

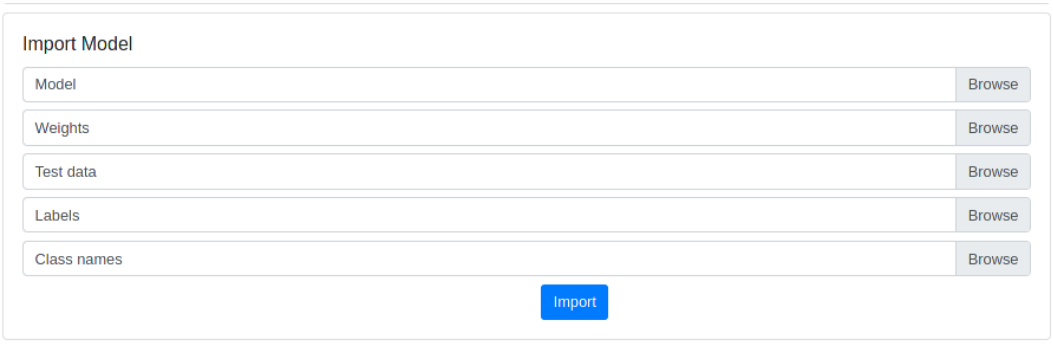

Рис. 4.2: Панель імпорту моделі

- Налаштування конфігурації тестів
- Перегляд результатів тестів

4.3 Тестування програмного забезпечення

Роботу даного додатку перевірено на таких мережах:

- MNIST (розпізнавання рукописних цифр)

- GTSRB (розпізнавання дорожніх знаків)
- CIFAR-10 (розпізнавання об'єктів на зображеннях невеликого розміру)
- ImageNet (розпізнавання об'єктів на зображеннях великого розміру)

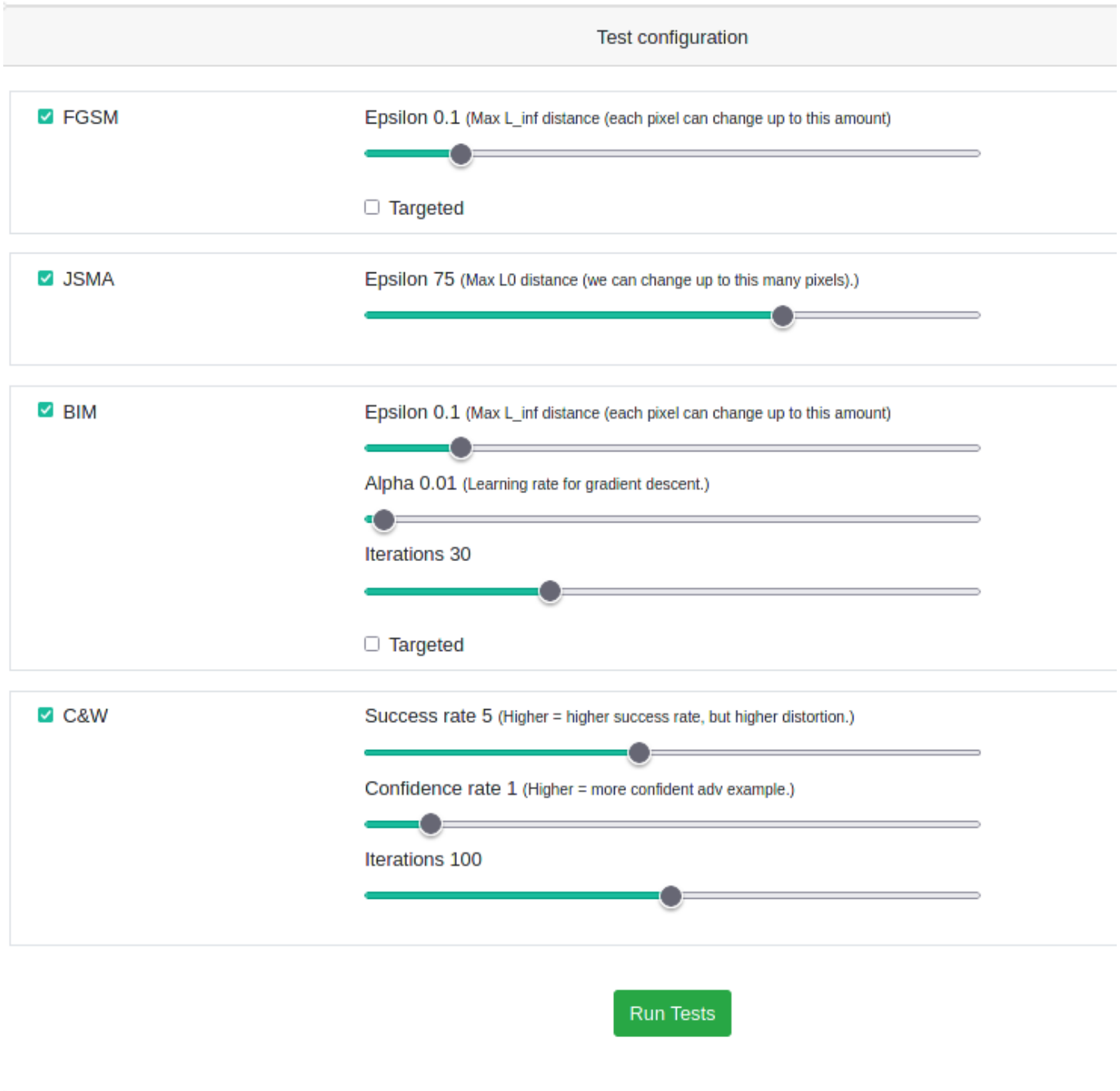

Рис. 4.3: Панель налаштування тестів. Користувач може обрати методи генерації змагальних зразків, та налаштувати їх параметри. Для методів Fast gradient sign та Basic Iterative method доступний вибір режиму таргетованої атаки

Рис. 4.4: Результат атаки FGSM на мережу MNIST.

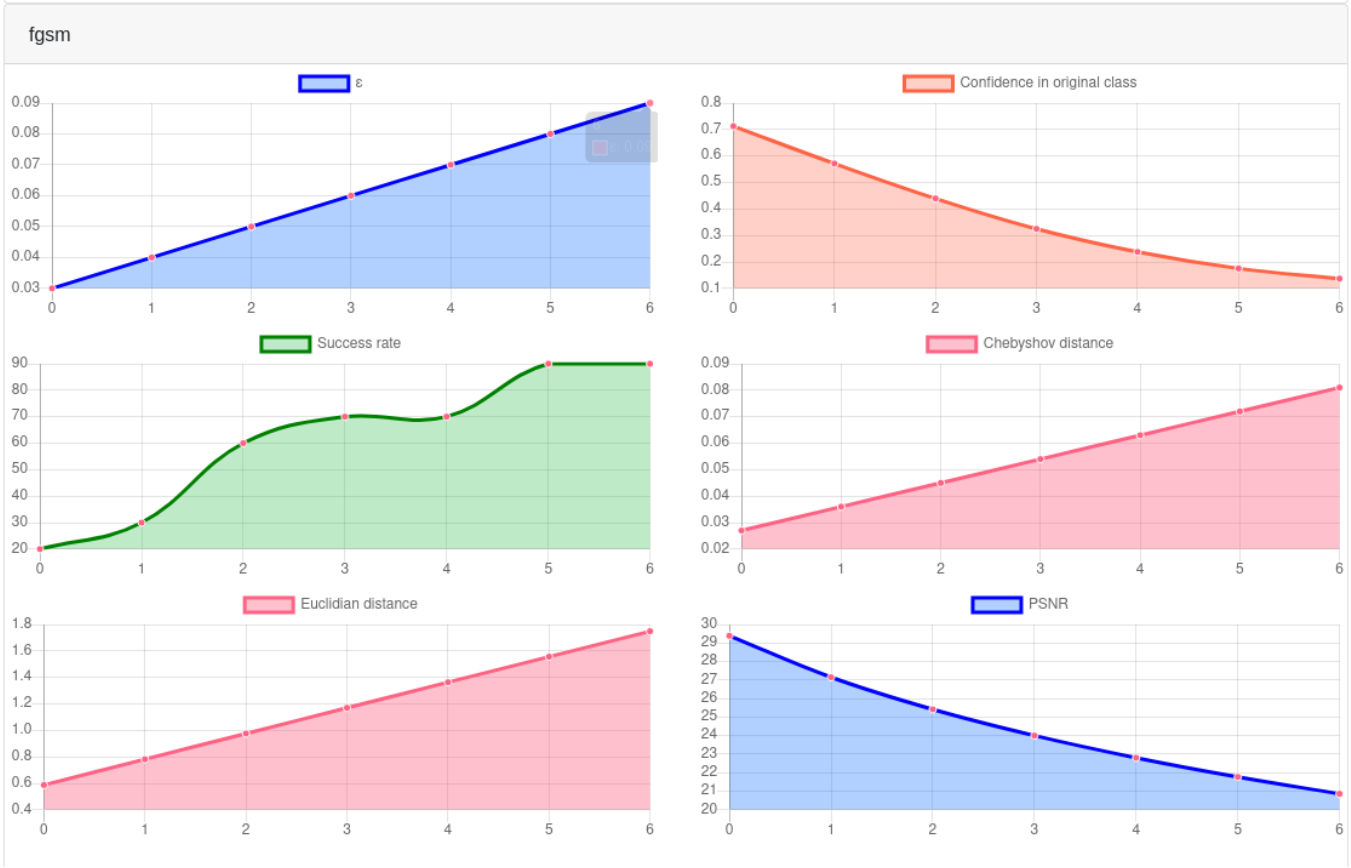

Рис. 4.5: Метрики атаки FGSM на мережу MNIST:

- ϵ - ступінь модифікацій
- Confidence in original class - середнє значення ймовірності моделі щодо правильного класу
- Success rate - процент успішних атак
- Відстань Чебишова
- Евклідова відстань
- Пікове співвідношення сигналу до шуму

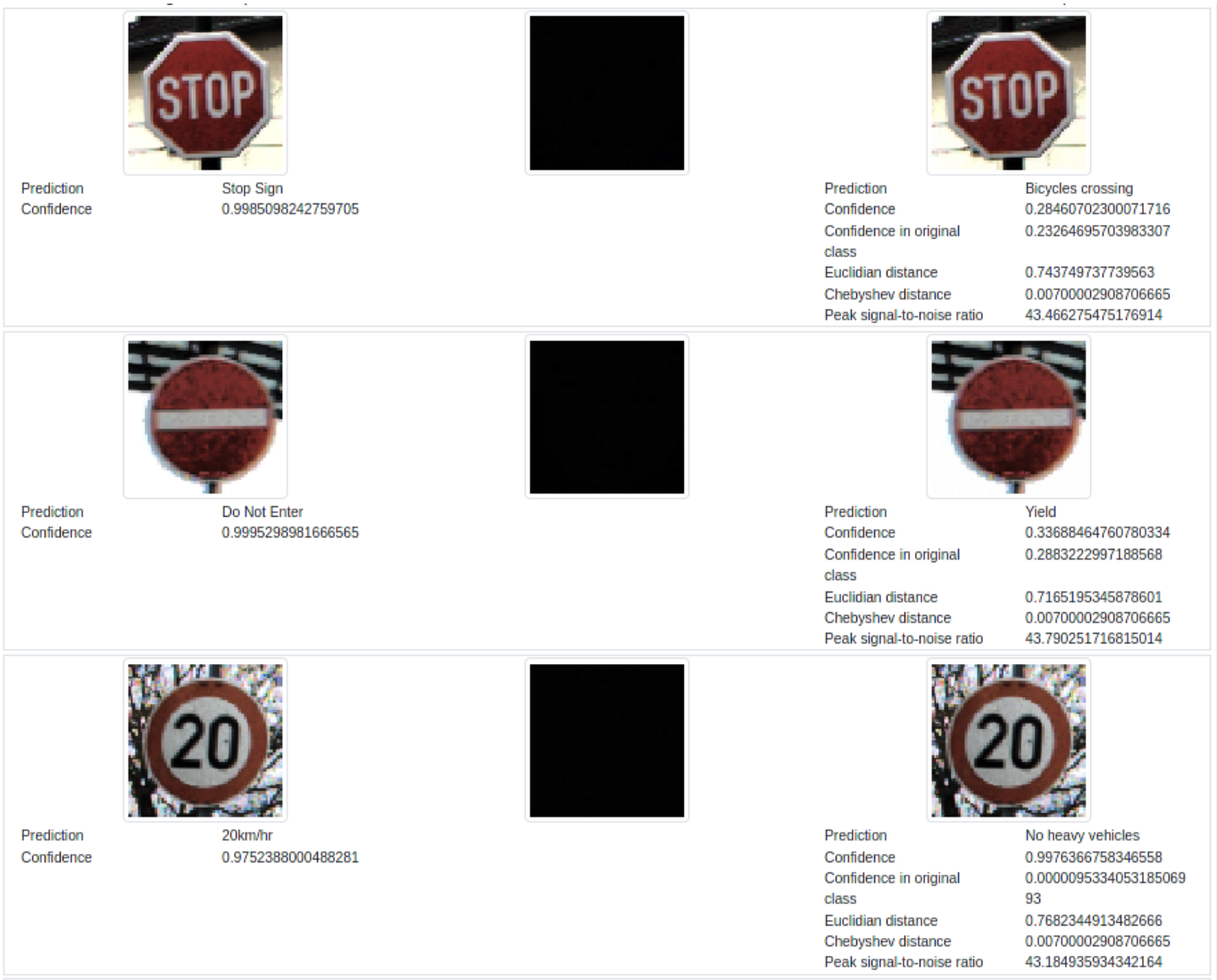

Рис. 4.6: Результат атаки FGSM на мережу GTSRB

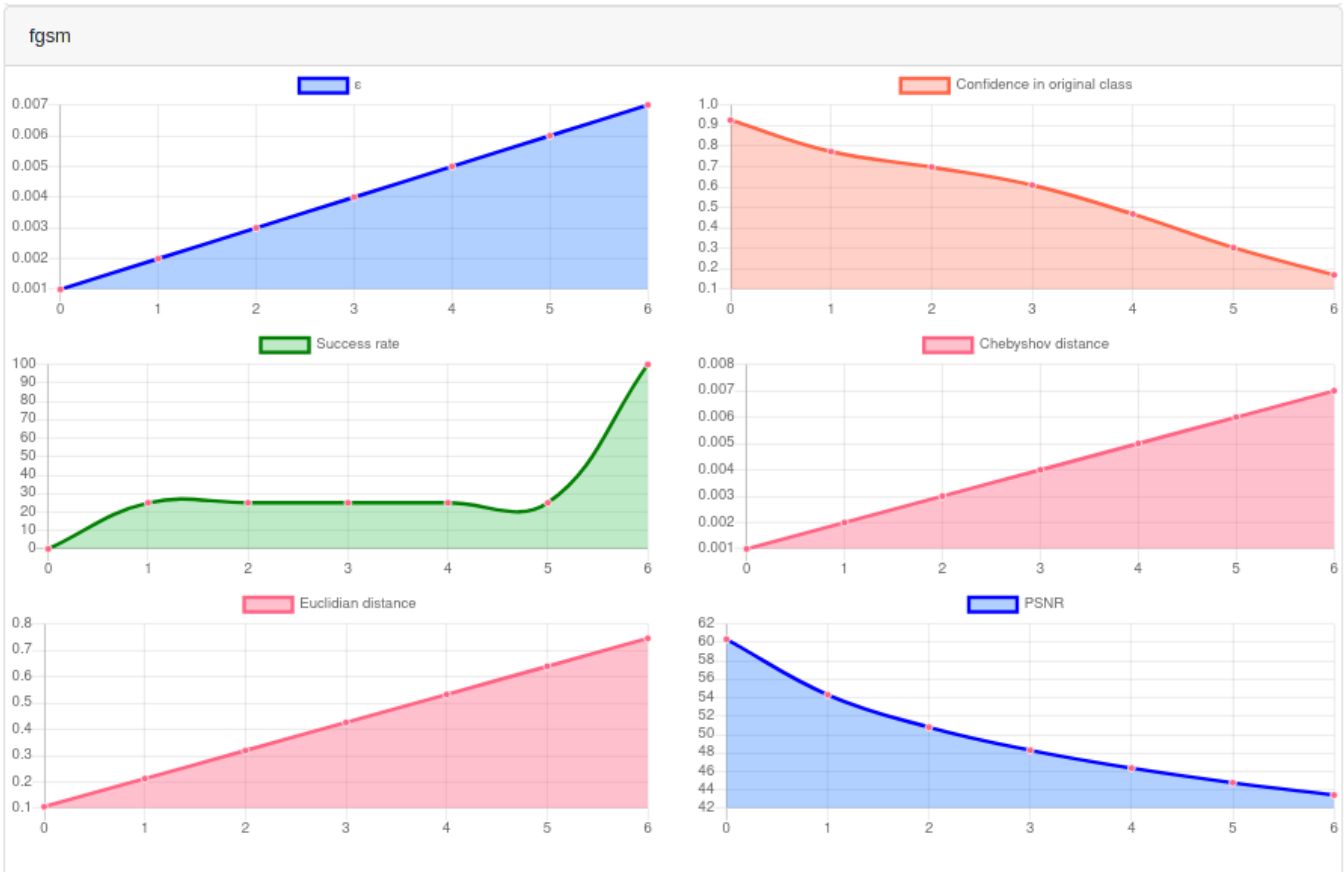

Рис. 4.7: Метрики атаки FGSM на мережу GTSRB

Рис. 4.8: Результат атаки FGSM на мережу CIFAR10

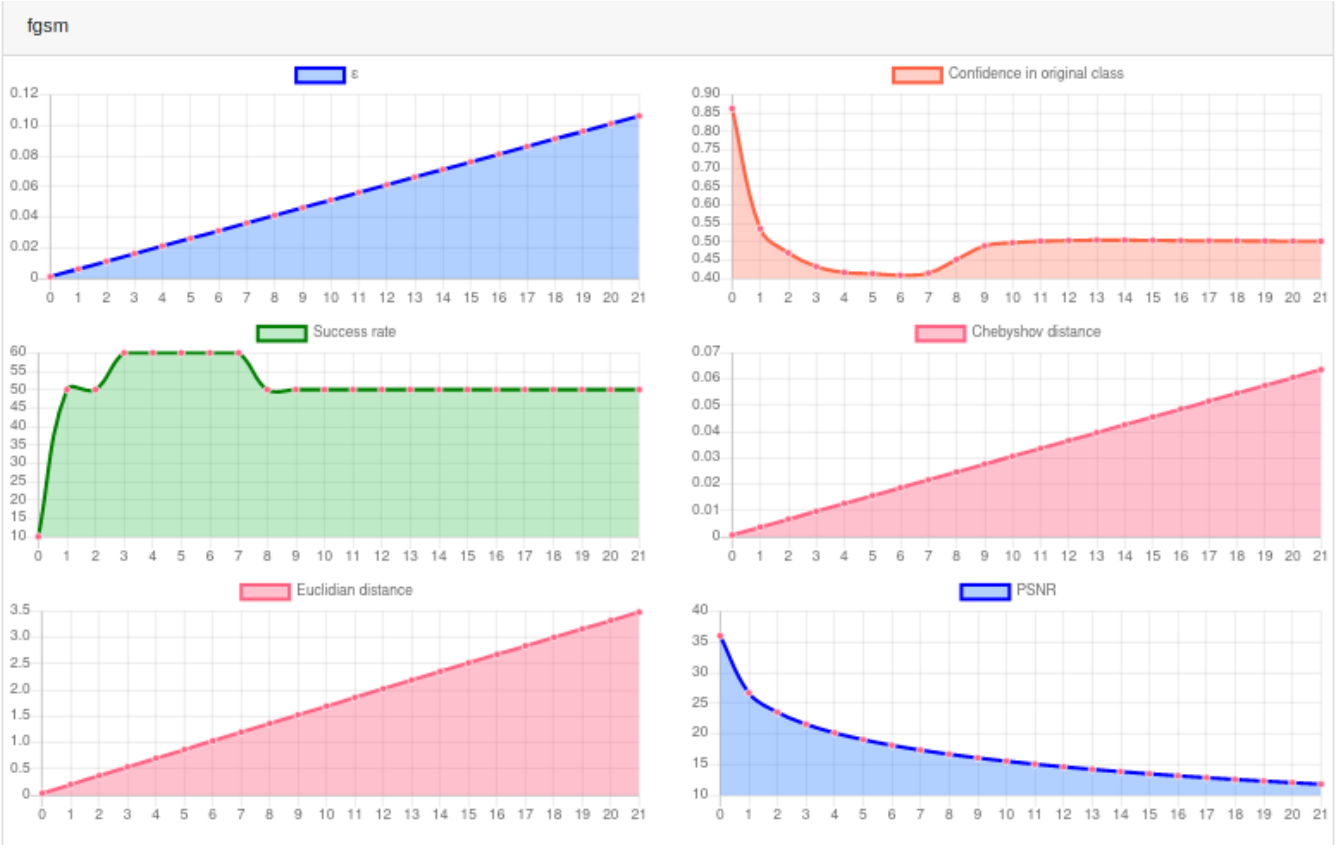

Рис. 4.9: Метрики атаки FGSM на мережу CIFAR10

Висновок

В даній роботі були досліджений математичний апарат моделей машинного навчання, та основні проблеми які виникають при розробці та використання цих моделей. Проаналізовані алгоритми генерації змагальних зразків, розроблено web додаток для випробування робастності, здатний генерувати змагальні зразки для моделей з різною топологією. При розробці використана бібліотека tensorflow.js та UI фреймворк Angular. Проведено тестування декількох нейронних мереж. Всі моделі, які досліджувались в даній роботі мають вразливості до змагальних атак.

Глибокі мережі досягли високої продуктивності в класифікації зображень, але вони не стійкі до незначних модифікацій, і мають тенденцію помилково класифікувати дані з мінімальними порушеннями які візуально не відрізняються від оригінальних зразків. Це викликає серйозні занепокоєння точки зору безпеки та вимагає подальшого дослідження.

Список літератури

[1] *A WebGL accelerated JavaScript library for training and deploying ML models.* вер. 3.11.0. 2021. URL: https://github.com/tensorflow/tfjs.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[3] S. Cai, S. Bileschi та E. Nielsen. *Deep Learning with JavaScript: Neural networks in TensorFlow.js*. Manning, 2020. ISBN: 9781617296178. URL: https://books.google.com.ua/books?id=N2dswgEACAAJ.

[4] C. D. Cantrell. *Modern Mathematical Methods for Physicists and Engineers*. USA: Cambridge University Press, 2000. ISBN: 0521598273.

[5] Nicholas Carlini та David Wagner. "Towards Evaluating the Robustness of Neural Networks". в: 2017. DOI: 10.1109/SP.2017.49.

[6] Justin Gilmer та ін. "Motivating the Rules of the Game for Adversarial Example Research". в: *CoRR* abs/1807.06732 (2018). arXiv: 1807.06732. URL: http://arxiv.org/abs/1807.06732.

[7] Xavier Glorot, Antoine Bordes та Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". в: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. за ред. Geoffrey Gordon, David Dunson та Miroslav Dudík. т. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, листоп. 2011, с. 315—323. URL: https://proceedings.mlr.press/v15/glorot11a.html.

[8] Kishor Datta Gupta, Dipankar Dasgupta та Zahid Akhtar. "Determining Sequence of Image Processing Technique (IPT) to Detect Adversarial Attacks". в: *SN Computer Science* 2 (5 2021). ISSN: 2662-995X. DOI: 10.1007/s42979-021-00773-8.

[9] Andrew Ilyas та ін. “Adversarial examples are not bugs, they are features”. в: т. 32. 2019.

[10] Mazaher Kianpour та Shao-Fang Wen. “Timing Attacks on Machine Learning: State of the Art”. в: січ. 2020, с. 111—125. ISBN: 978-3-030-29515-8. DOI: 10.1007/978-3-030-29516-5_10.

[11] Alexey Kurakin, Ian J. Goodfellow та Samy Bengio. “Adversarial examples in the physical world”. в: 2019. DOI: 10.1201/9781351251389-8.

[12] Hazel Si Min Lim та Araz Taeihagh. “Algorithmic Decision-Making in AVs: Understanding Ethical and Technical Concerns for Smart Cities”. в: *Sustainability* 11.20 (2019). ISSN: 2071-1050. DOI: 10.3390/su11205791. URL: https://www.mdpi.com/2071-1050/11/20/5791.

[13] Jason Lin та Dilara Soylu. *AdVis: Visualizing and Attributing ML Attacks to Adversarial Examples*. тех. звіт. Accessed: 2021. Georgia Institute of Technology, 2018.

[14] Seyed Mohsen Moosavi-Dezfooli, Alhussein Fawzi та Pascal. “DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks”. в: т. 2016-December. 2016. DOI: 10.1109/CVPR.2016.282.

[15] Anh Nguyen, Jason Yosinski та Jeff Clune. “Deep Neural Networks are Easily Fooled”. в: *Computer Vision and Pattern Recognition, 2015 IEEE Conference on* (2015). ISSN: 1875-7855.

[16] Nicolas Papernot та ін. “The limitations of deep learning in adversarial settings”. в: 2016. DOI: 10.1109/EuroSP.2016.36.

[17] Aaditya Prakash та ін. “Deflecting Adversarial Attacks with Pixel Deflection”. в: 2018. DOI: 10.1109/CVPR.2018.00894.

[18] Dr. Katharina Rasch. *Generating adversarial images in the browser with tensorflow.js*. https://krasch.io/adversarial-tensorflowjs/. Accessed: 2021. 2021.

[19] Tariq Rashid. *Make Your Own Neural Network*. 1st. North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2016. ISBN: 1530826608.

[20] Christian Robert. “Machine Learning, a Probabilistic Perspective”. в: *CHANCE* 27 (2 2014). ISSN: 0933-2480. DOI: 10.1080/09332480.2014.914768.

[21] Avatar Kenny Song. *Break neural networks in your browser*. `https://github.com/kennysong/adversarial.js`. 2021.

[22] Jiawei Su, Danilo Vasconcellos Vargas та Kouichi Sakurai. "One Pixel Attack for Fooling Deep Neural Networks". в: *IEEE Transactions on Evolutionary Computation* 23 (5 2019). ISSN: 19410026. DOI: 10.1109/TEVC.2019.2890858.

[23] Christian Szegedy та ін. "Intriguing properties of neural networks". в: 2014.

[24] C.-Y. Tsai та D. Cox. *Are Deep Learning Algorithms Easily Hackable?* `http://coxlab.github.io/ostrichinator`. 2015.

[25] Danilo Vasconcellos Vargas та Jiawei Su. "Understanding the one-pixel attack: Propagation maps and locality analysis". в: т. 2640. 2020.

[26] Zhou Wang та Alan C. Bovik. *Mean Squared Error: Love it or leave it?* 2009.

Додаток А

Посилання

Код проєкту https://github.com/o-khytrov/advtest

Демо https://o-khytrov.github.io/advtest/