

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики  
Кафедра обчислювальної математики

## Курсова робота

АНАЛІЗ АТАК НА ЛІНІЙНІ МОДЕЛІ МАШИННОГО НАВЧАННЯ

Виконав: студент III курсу групи ПМп-31  
напрямку підготовки (спеціальності)  
123 — "Прикладна математика"

Середович В.В.  
(прізвище та ініціали)

Керівник: Музичук Ю.А.  
(прізвище та ініціали)

# Зміст

<b>1</b>	<b>Вступ</b>	<b>2</b>
1.1	Постановка задачі . . . . .	2
1.2	Основні поняття . . . . .	3
<b>2</b>	<b>Модель і датасет</b>	<b>4</b>
2.1	Мультикласова логістична регресія . . . . .	4
2.2	Датасет . . . . .	5
2.3	Тренування моделі . . . . .	5
<b>3</b>	<b>Методи атак</b>	<b>6</b>
3.1	FGSM . . . . .	7
3.2	I-FGSM . . . . .	8
3.3	TI-FGSM . . . . .	9
3.4	MI-FGSM . . . . .	9
3.5	DeepFool . . . . .	11
<b>4</b>	<b>Методи захисту</b>	<b>13</b>
4.1	Randomization . . . . .	13
4.2	Denoising . . . . .	13
<b>5</b>	<b>Аналіз атак</b>	<b>14</b>
<b>6</b>	<b>Висновок</b>	<b>15</b>

# Розділ 1

## Вступ

Алгоритми машинного навчання активно використовується у різних областях нашого життя та дуже добре демонструють себе у задачах класифікації. Однак, виявляється, що ці алгоритми є досить вразливими навіть до незначних, правильно підібраних, пертурбацій вхідної інформації. Такі модифіковані приклади, які навмисно змінюють для того, щоб вони були хибно передбаченні називають змагальними прикладами або *adversarial samples*. В деяких випадках ці зміни можуть бути настільки малі, що для людини вони зовсім не будуть помітні, однак класифікатор моделі буде робити хибне передбачення. Така вразливість може бути дуже небезпечною, коли алгоритми машинного навчання застосовуються в критичних для здоров'я людини середовищах. Саме тому ця область привертає до себе багато уваги і порбеює досліджень.

В межах теми цієї роботи будуть розглядатись різні методи атак на лінійні моделі машинного навчання які будуть аналізуватись і порівнюватись між собою. Також будуть запропоновані методи захисту моделі.

### 1.1 Постановка задачі

*Мета* даної роботи полягає у тому, щоб дослідити ефективність атак на лінійні моделі машинного навчання, та визначити методи захисту від них.

Виходячи з мети, визначені завдання роботи:

- Реалізувати лінійну модель машинного навчання
- Дослідити різні методи генерування змагальних прикладів
- Проаналізувати ефективність атак
- Визначити методів захисту

## 1.2 Основні поняття

В цій секції будуть розглянуті основні поняття, які будуть використовуватись в ході роботи.

### **Змагальний приклад (Adversarial sample)**

Нехай існує класифікатор  $f(x) : x \rightarrow y$ , де  $x \in X, y \in Y$ , який передбачає значення  $y$  для вхідного  $x$ . Метою змагального прикладу є знайти такий  $x^*$ , який знаходиться в околі  $x$ , але хибно визначається класифікатором. Зазвичай максимальний рівень шуму який може бути в змагальному прикладі може бути не більше за певну  $L_p$  норму  $\|x^* - x\|_p < \varepsilon$ , де  $p = 1, 2, \infty$ . В межах даної роботи для визначення рівня пертурбації буде використовуватись саме  $L_\infty$  норма.

### **Націлені атаки (Untargeted Attacks)**

Націлені атаками є атаки метою яких є внести в приклад  $x$ , який правильно передбачається як  $f(x) = y$  незначний шум так, щоб класифікатор зробив хибне передбачення  $f(x^*) \neq y$ .

### **Ненацілені атаки (Targeted Attacks)**

Ненацілені атаками називають такі атаки, метою яких є внести у вхідний приклад  $x$  такі пертурбації щоб класифікатор передбачив якийсь конкретний клас  $f(x^*) = y^*$ , де  $y^*$  є заданою ціллю  $y^* \neq y$ .

### **Атаки на закриту модель (White Box attacks)**

Атаки на закриту модель описують такий сценарій, коли в нападника є повний доступ до параметрів і градієнтів моделі.

### **Атаки на відкриту модель (Black Box attacks)**

До атак на відкриту модель відносять атаки, при яких в нападника нема доступу до параметрів моделі.

## Розділ 2

# Модель і датасет

### 2.1 Мультикласова логістична регресія

В якості лінійного методу машинного навчання, на котрий будуть здійснюватись атаки буде використовуватись модифікований алгоритм логістичної регресії для мультикласової класифікації.

Нехай маємо набір тренувальних даних:

$$(x^{(1)}, y^{(1)}), \quad (x^{(2)}, y^{(2)}), \quad \dots, \quad (x^{(m)}, y^{(m)})$$
$$X = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad x \in \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix}$$
$$Y = \begin{bmatrix} \vdots & \vdots & & \vdots \\ y^{(1)} & y^{(2)} & \dots & y^{(m)} \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad y \in \begin{bmatrix} y_1 \\ \dots \\ y_C \end{bmatrix} \quad y_1, \dots, y_C$$

$X$  - матриця в якій кожен стовпець є набором характеристик  $i$ -ого прикладу,  $i = 1, \dots, m$

$Y$  - матриця класів в якій кожен стовпець це масив розмірності  $C$ , з одиницею на місті справжнього класу

$m$  - кількість характеристик

$n$  - кількість характеристик в кожному прикладі

$C$  - кількість класів

Задача прикладу  $x \in R^n$ , знайти  $\hat{y} = P(y = 1 \mid x)$ ,  $0 \leq \hat{y} \leq 1$

$$\hat{y} = \text{softmax}(\omega^T x + b), \quad \text{де } \omega \in R^n, b \in R - \text{невідомі параметри}$$

Функція активації матиме вигляд:

$$\text{softmax}(z) = \frac{e^z}{\sum_{k=1}^C e_k^z} \quad (2.1)$$

Для кожного прикладу з тренувального датасету потрібно обчислити:

$$z^{(i)} = \omega^T x^{(i)} + b, \text{ де } y^{(i)} = \text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{k=1}^C e_k^z}, \text{ так щоб } \hat{y}^{(i)} \approx y^{(i)}$$

В якості функції втрати буде використовуватись функція кросс-ентропії:

$$\xi(y, x) = - \sum_{i=1}^C y_i \log(\hat{y}_i) \quad (2.2)$$

Задача полягає в тому щоб знайти параметри  $w \in R_x^n, b \in R$  що мінімізують функцію  $J(\omega, b)$ . Для цього будемо використовувати алгоритм градієнтного спуску.

## 2.2 Датасет

Для аналізу атак на лінійні методи машинного навчання буде використовуватись MNIST база даних яка складається з 60тис. тренувальних зображень та 10 тис. тестувальних рукописних цифр.



Рис. 2.1: датасет рукописних цифр MNIST

На її основі буде здійснюватись тренування моделі та аналіз атак.

## 2.3 Тренування моделі

TODO короткий опис реалізованої моделі

## Розділ 3

### Методи атак

В цьому розділі будуть розглянуті деякі методи генерування змагальних зразків, які були реалізовані і протестовані на лінійній моделі натренований в розділі 2.3.

Далі, під час опису будуть використовуватись такі позначення:

- $\mathbf{X}$  - це зображення, зазвичай це 3-D тензор (ширина  $\times$  висота  $\times$  глибина). В нашому випадку має розміри  $(28 \times 28 \times 1)$  пікселей, яке складається з додатніх цілих чисел в межах  $[0, 255]$ .
- $\mathbf{X}^*$  - згенерований змагальний приклад для зображення  $\mathbf{X}$
- $y_{true}$  - правильний клас для зображення  $\mathbf{X}$ .
- $J(\mathbf{X}, y)$  - штрафна функція моделі. Ваги моделі  $\omega$  навмисно не будемо вказувати, вважатимемо їх сталими, як значення які будуть отримані з моделі машинного навчання. В нашому випадку штрафною функцією є функція кросс-ентропії 2.2.
- $Clip_{\mathbf{X}, \varepsilon}\{\mathbf{X}^*\}$  - функція яка виконує по-піксельне обрізання зображення  $\mathbf{X}^*$  так, щоб результат був в  $L_\infty$   $\varepsilon$  - околі вхідного зображення  $\mathbf{X}$ . Рівняння для цього виглядає наступним чином:

$$Clip_{\mathbf{X}, \varepsilon}\{\mathbf{X}^*\}(x, y, z) = \min\left\{255, \mathbf{X}(x, y, z) + \varepsilon, \max\{0, \mathbf{X}(x, y, z) - \varepsilon, \mathbf{X}^*(x, y, z)\}\right\}, \quad (3.1)$$

де  $\mathbf{X}(x, y, z)$  - це значення каналу  $z$  зображення  $\mathbf{X}$  з координатами  $(x, y)$ .

### 3.1 FGSM

Одним з найпоширеніших методів для генерування змагальних зразків машинного навчання є Fast Gradient Sign Method. В перше він був представлений в роботі Goodfellow, Shlens та Szegedy [1]. Ідея цього методу полягає в тому, щоб знайти такий adversarial приклад  $x^*$  який максимізує функцію втрати  $J(x^*, y)$  до певного  $L_\infty$  обмеження. Цього можна досягти один раз використавши використавши зворотне поширення:

$$X^* = X + \varepsilon \cdot \text{sign}(\Delta_x J(x, y)), \quad (3.2)$$

де  $\varepsilon$  - деякий гіперпараметер.

Для того щоб застосувати цей метод для логістичної регресії необхідно обчислити градієнт від функції кросс-ентропії 2.2. Для цього спочатку знайдемо похідну від *softmax* функції:

$$z_i = \omega^T x^{(i)} + b; \quad \hat{y}_i = a_i = \text{softmax}(z_i); \quad \text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{k=1}^C e^{z_k}};$$

Якщо  $i = j$ ,

$$\frac{\partial \hat{y}_i}{\partial z_j} = \frac{\partial \frac{e^{z_i}}{\sum_{k=1}^C e^{z_k}}}{\partial z_j} = \frac{e^{z_i} \sum_{k=1}^C e^{z_k} - e^{z_j} e^{z_i}}{(\sum_{k=1}^C e^{z_k})^2} = \quad (3.3)$$

$$= \frac{e^{z_j}}{\sum_{k=1}^C e^{z_k}} \times \frac{(\sum_{k=1}^C e^{z_k} - e^{z_j})}{\sum_{k=1}^C e^{z_k}} = y_i(1 - y_j) \quad (3.4)$$

Якщо  $i \neq j$ ,

$$\frac{\partial \hat{y}_i}{\partial z_j} = \frac{\partial \frac{e^{z_i}}{\sum_{k=1}^C e^{z_k}}}{\partial z_j} = \frac{0 - e^{z_j} e^{z_i}}{\sum_{k=1}^C e^{z_k}} \times \frac{e^{z_i}}{\sum_{k=1}^C e^{z_k}} = -y_i y_j \quad (3.5)$$

Далі обчислюємо похідну від функції кросс-ентропії:

$$\xi(y, x) = - \sum_{i=1}^C y_i \log(\hat{y}_i)$$



$$\begin{aligned}
\frac{\partial \xi}{\partial z_i} &= - \sum_{j=1}^C \frac{\partial y_j \log(\hat{y}_j)}{\partial z_i} = - \sum_{j=1}^C y_j \frac{\partial \log(\hat{y}_j)}{\partial z_i} = - \sum_{j=1}^C y_j \frac{1}{\hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial z_i} = \\
&= - \frac{y_i}{\hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_i} - \sum_{j \neq i} \frac{y_j}{\hat{y}_j} \frac{\partial \hat{y}_j}{\partial z_i} = - \frac{y_i}{\hat{y}_i} \hat{y}_i (1 - \hat{y}_i) - \sum_{j \neq i} \frac{y_j}{\hat{y}_j} (-\hat{y}_i \hat{y}_j) = \\
&= -y_i + y_i \hat{y}_i + \sum_{j \neq i} y_j \hat{y}_i = -y_i + \sum_{j=1}^C y_j \hat{y}_i = \hat{y}_i - y_i,
\end{aligned}$$

де  $i = 1, \dots, C$

Тепер можна обчислити градієнт від штрафної функції по  $x$ :

$$\frac{\partial \xi}{\partial x_i} = \frac{\partial \xi}{\partial z_i} \frac{\partial z_i}{\partial x_i} = (\hat{y}_i - y_i) \frac{\partial (\omega^T x_i + b)}{\partial x_i} = (\hat{y}_i - y_i) \omega^T = dz \cdot \omega^T \quad (3.6)$$

Маючи всі наведені обчислення, можна записати алгоритм знаходження змагального прикладу [1].

---

#### Algorithm 1 FGSM

---

- 1: **Input:** Приклад  $x$ , клас  $y_{true}$ , класифікатор  $f$ , параметри:  $\omega, b$ .
  - 2: **Input:** значення пертурбації  $\varepsilon$ .
  - 3: **Output:** Adversarial  $x^*$  з нормою  $\|x^* - x\|_\infty \leq \varepsilon$ ;
  - 4:  $z = \omega^T x + b$ ;
  - 5:  $\hat{y} = softmax(z)$ ;
  - 6:  $\delta = (\hat{y} - y_{true}) \cdot \omega^T$ ;
  - 7:  $x^* = Clip_{X, \varepsilon}\{x + \alpha \cdot sign(\delta)\}$ ;
  - 8: **return**  $x^* = x_T^*$ .
- 

## 3.2 I-FGSM

Даний метод є модифікацією попереднього, особливість якого полягає в тому, що для знаходження змагального прикладу буде використовуватись ітераційний підхід. Ми застосовуємо звичайний метод декілька разів з певним невеликим кроком  $a$  і після кожного кроку використовуємо функцію *clip* 3.1, для того щоб переконатись що значення знаходяться в  $\varepsilon$ -околі оригінального зображення.

Ітераційної алгоритм швидкого градієнтного спуску буде мати вигляд [2].

---

**Algorithm 2** *I – FGSM*

---

```
1: Input: Приклад  $x$ , значення класу  $y_{true}$ , класифікатор  $f$ 
2: Input: значення пертурбації  $\varepsilon$ , кількість ітерації  $T$ .
3: Output: Adversarial  $x^*$  з нормою  $\|x^* - x\|_\infty \leq \varepsilon$ ;
4:  $a = \varepsilon/T$ ;
5:  $x^* = x$ ;
6: for  $t = 0$  to  $T - 1$  do
7:    $x^* = Clip_{X,\varepsilon}\{x^* + \alpha \cdot sign(\Delta_x J(x, y))\}$ ;
8: end for
9: return  $x^* = x_T^*$ .
```

---

### 3.3 TI-FGSM

TI-FGSM або як його ще часто називають Projected Gradient Descent є націленою версією попереднього алгоритму, а отже він дає можливість обрати клас який має передбачити модель. Два попередні методи описаних вище намагалися максимізувати функцію втрати для правильного класу, не вказуючи яким буде неправильний клас. Цей метод - навпаки, мінімізує функцію втрати для деякого заданого неправильного класу. На сам алгоритм це вплине так, що на етапі додавання пертурбації у зображення знак буде змінений на протилежний.

---

**Algorithm 3** *TI – FGSM*

---

```
1: Input: Приклад  $x$ , значення класу  $y_{true}$ , класифікатор  $f$ 
2: Input: значення пертурбації  $\varepsilon$ , кількість ітерації  $T$ .
3: Output: Adversarial  $x^*$  з нормою  $\|x^* - x\|_\infty \leq \varepsilon$ ;
4:  $a = \varepsilon/T$ ;
5:  $x^* = x$ ;
6: for  $t = 0$  to  $T - 1$  do
7:    $x^* = Clip_{X,\varepsilon}\{x^* - \alpha \cdot sign(\Delta_x J(x, y))\}$ ;
8: end for
9: return  $x^* = x_T^*$ .
```

---

### 3.4 MI-FGSM

Попередні розглянуті методи хоч і є ефективними, але мають певні недоліки. Для прикладу однокроковий (FGSM), вираховує градієнт тільки один раз, припускаючи, що межа рішень є лінійною. На практиці це зазвичай не так і тому для змагального прикладу, згенерованого таким чином, характерне "недотренування", що обмежує ефективність атаки.

З іншого боку ітеративний метод (I-FGSM), з кожною ітерацією жадібно рухає змагальний приклад в напрямку градієнту, що може призвести до зупинки в локальній точці оптимума та "перетренування" моделі. Щоб вирішити цю проблему в роботі Dong та ін. [4] була представлена momentum оптимізація ітеративного методу. Він допомагає стабілізувати напрямки зміни, прикладу що допомагає уникнути локального оптимума.

### TODO опис momentum метода

Алгоритм даної модифікації має наступний вигляд 4

---

#### Algorithm 4 *MI – FGSM*

---

```

1: Input: Приклад  $x$ , значення цілі  $y$ , класифікатор  $f$ .;
2: Input: значення пертурбації  $\varepsilon$ , значення цілі  $y$ , кількість ітерації  $T$ .;
3: Output: Adversarial  $x^*$  з нормою  $\|x^* - x\|_{inf} \leq \varepsilon$ ;
4:  $a = \varepsilon/T$ ;
5:  $g_0 = 0$ ;  $x_0^* = x$ ;
6: for  $t = 0$  to  $T - 1$  do
7:    $g_{t+1} = \mu \cdot g_t + \frac{\Delta_x J(x^*, y)}{\|\Delta_x J(x^*, y)\|_1}$ ;
8:    $x^* = Clip_{X, \varepsilon} \{x_t^* + a \cdot sign(g_{t+1})\}$ ;
9: end for
10: return  $x^* = x_T^*$ .

```

---

### 3.5 DeepFool

DeepFool була вперше представлена в роботі Moosavi-Dezfooli, Fawzi та Frossard [2]. На відмінно від FGSM методу цей алгоритм не можна використовуватись для націлених атак, де є можливість вибрати клас який має передбачити модель. Отже, його можна використовувати тільки для генерування прикладів які будуть класифікуватись відмінно від правильно класу. Основною метою цього методу є знаходження мінімального необхідного значення пертурбація, для того щоб модель хибно передбачила клас прикладу.

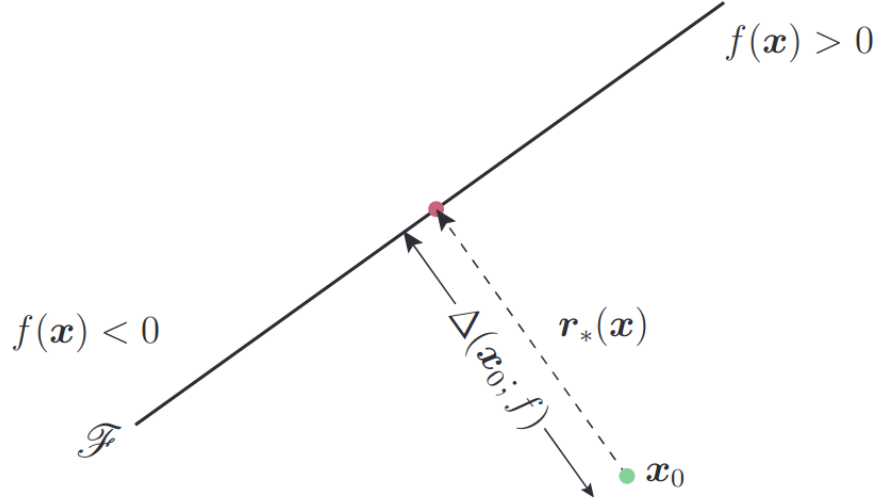


Рис. 3.1: Приклад для лінійного бінарного класифікатора [2]

Для випадку бінарної класифікації легко бачити, що надійність моделі  $f$  в точці  $x_0$ ,  $\Delta(x_0; f)$ , дорівнює відстанні від  $x_0$  до площини гіперпараметра  $\mathcal{F} = \{x : \omega^T x + b = 0\}$ , яка розділяє два класи (Рис. 3.1). Таким чином, мінімальний зсув, необхідний для зміни рішення класифікатора відповідає ортогональній проекції  $x_0$  на  $\mathcal{F}$ . Це можна записати у вигляді формули:

$$r_*(x_0) := -\frac{f(x_0)}{\|\omega\|_2^2} \omega. \quad (3.7)$$

Для випадку  $l_2$  норми DeepFool алгоритм для мультикласового випадку буде мати наступний вигляд [5].

---

**Algorithm 5** DeepFool: мультикласовий випадок

---

```
1: Input: Приклад  $x$ , значення цілі  $y$ , класифікатор  $f$ .;  
2: Input: значення цілі  $y$ , кількість ітерацій  $T$ .;  
3: Output: Adversarial  $x^*$  з нормою  $\|x^* - x\|_2 \leq \varepsilon$ ;  
4:  $x_0 = x$ ;  $i = 0$ ;  
5: while  $\hat{f}(x_i) = \hat{f}(x_0)$  do  
6:   for  $k \neq \hat{f}(x_0)$  do  
7:      $w'_k = \Delta f_k(x_i) - \Delta f_{\hat{k}(x_0)}(x_i)$ ;  
8:      $f'_k = \Delta f_k(x_i) - \Delta f_{\hat{k}(x_0)}(x_i)$ ;  
9:   end for  
10:   $\hat{l} = \arg \min_{k \neq \hat{f}(x_0)} \frac{|f'_k|}{\|w'_k\|_2}$ ;  
11:   $r_i = \frac{|f'_l|}{\|w'_l\|_2} w'_l$ ;  
12:   $x_{i+1} = x_i + r_i$ ;  
13:   $i = i + 1$ ;  
14: end while  
15: return  $\hat{r} = \sum_i r_i$ .
```

---

Для того що реалізувати цей алгоритм необхідно спочатку знайти градієнт від функції кросс-ентропії 2.2 по  $x$ .

Маючи:

$$\frac{\partial \hat{y}_i}{\partial x_k} = \begin{cases} y_i(1 - y_j) & , \text{ якщо } i = j \\ -y_i y_j & , \text{ якщо } i \neq j \end{cases}, \text{ де } i = 1, \dots, C; j = 1, \dots, C; \quad (3.8)$$

При знаходженні похідної нас цікавить випадок  $i = j$ , тому отримуємо:

$$\frac{\partial \hat{y}_i}{\partial x_k} = \frac{\partial \hat{y}_i}{\partial z_j} \frac{\partial z_j}{\partial x_k} = y_i(1 - y_i) \frac{\partial(\omega^T x_k + b)}{\partial x_k} = y_i(1 - y_i) \omega^T = dz \cdot \omega^T$$

## Розділ 4

# Методи захисту

### 4.1 Randomization

Одним з ефективних методів захисту, який добре себе продемонстрував в ході змагань за найкращих метод атак та захисту 2018 року [6] є рандомізація вхідних даних.

Зазвичай пертурбації, які генеруються в наслідок ітеративних атак, можуть легко опинитись перетренованими для деяких параметрів моделі і є унікальними для кожного прикладу. Ця властивість дає можливість зробити висновок, що трансформації додані до вхідних даних, такі як зміна розміру, стискання, або додавання відступів до зображення може зруйнувати структуру певної змагальної пертурбації і таким чином стати непоганим захистом. Цей метод є ефективним і для атак на відкриту модель тек, якщо трансформації відбуваються випадково, а нападник не знає про характер цих трансформацій.

### 4.2 Denoising

Ще одним з методів захисту є позбавлення від шуму вхідних даних.

**TODO опис метода**

## Розділ 5

### Аналіз атак

TODO аналіз методів та захисту, графіки ефективності атак/захисту (Вже частково реалізовані)

## Розділ 6

## Висновок

### Висновок



# Бібліографія

- [1] Ian J. Goodfellow, Jonathon Shlens та Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2014. arXiv: 1412.6572 [stat.ML].
- [2] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi та Pascal Frossard. *DeepFool: a simple and accurate method to fool deep neural networks*. 2015. arXiv: 1511.04599 [cs.LG].
- [3] Alexey Kurakin, Ian Goodfellow та Samy Bengio. *Adversarial examples in the physical world*. 2016. arXiv: 1607.02533 [cs.CV].
- [4] Yinpeng Dong та ін. *Boosting Adversarial Attacks with Momentum*. 2017. arXiv: 1710.06081 [cs.LG].
- [5] Xiaoyong Yuan та ін. *Adversarial Examples: Attacks and Defenses for Deep Learning*. 2017. arXiv: 1712.07107 [cs.LG].
- [6] Alexey Kurakin та ін. *Adversarial Attacks and Defences Competition*. 2018. arXiv: 1804.00097 [cs.CV].
- [7] Yinpeng Dong та ін. *Benchmarking Adversarial Robustness*. 2019. arXiv: 1912.11852 [cs.CV].