Міністерство освіти і науки, молоді та спорту України
Львівський національний університет імені Івана Франка
Факультет прикладної математики та інформатики
Кафедра обчислювальної математики

# Курсова робота

на тему:

## *"Розробка алгоритмів захисту від атак на глибокі нейронні мережі"*

Виконав:
студент IV курсу групи ПМп-41
напрямку підготовки (спеціальності)
113 – "Прикладна математика"
Бугрій Б.О.

Науковий керівник:
доц. Музичук Ю.А.

Львів - 2021

# Зміст

**Вступ** **3**

**1 Опис проблеми** **4**
1.1 Постановка задачі . . . . . 4
1.2 Типологія захисту . . . . . 5

**2 Глибокі нейронні мережі** **7**
2.1 Будова штучної нейронної мережі . . . . . 7

**3 Визначення стійкості** **9**

**4 Огляд атак на моделі машинного навчання** **11**
4.1 Швидкий градієнтний спуск . . . . . 11
4.2 Оптимізація нульового порядку . . . . . 12

**5 Захисна дистиляція та її модифікації** **13**
5.1 Дистиляція нейронних мереж . . . . . 13
5.2 Процес тренування та основні параметри . . . . . 14
5.3 Оцінка захисту . . . . . 16

**6 Методи з гарантованою стійкістю** **19**
6.1 Захист PixelDP . . . . . 19
6.2 Застосування та основні параметри . . . . . 20
6.3 Аналіз захисту . . . . . 21

**Висновок** **22**

**Додатки** **24**

**Література** **26**

# Вступ

Сьогодні розумні системи штучного інтелекту є невіддільною частиною життєдіяльності суспільства. Завдяки таким системам людству вдалося досягти значних результатів у багатьох сферах та галузях, таких як, наприклад, медицина, програмна інженерія, машинобудування та робототехніка.

Зараз найбільш поширеним типом алгоритмів машинного навчання є глибокі нейронні мережі, які здатні знаходити закономірності у великих масивах даних, а результати їх роботи часто перевершують людей. Проте, як показують численні дослідження [1], такі алгоритми часто використовують антиінтуїтивні, в порівнянні зі смисловим значенням, закономірності стосовно певних характеристик. Через це нейронні мережі є вразливими до різного роду зловмисних втручань, які можуть призвести до невірних результатів.

Моделі, які показують відмінні результати на звичайних даних, можуть бути легко ошуканими зразками, які лише трохи відрізняються від правильно класифікованих прикладів. Це розкриває фундаментальні недоліки в алгоритмах машинного навчання, які можуть бути використані зловмисниками для заподіяння шкоди, що може вплинути на безпеку технологічних процесів людської життєдіяльності і призвести до катастроф великого масштабу.

Природним є бажання розробити архітектури та алгоритми тренування нейронних мереж, які зменшують ризики таких атак та є стійкими до зловмисних збурень. Постає питання, які саме фактори впливають на стійкість моделі, що робить її більш вразливою до атак, а що ні. Щоб ефективно захиститись від можливих загроз, потрібно знати слабкі місця глибоких нейронних мереж та стратегії нападу, що використовують зловмисники.

Ця робота є логічним продовженням нашої попередньої роботи [14] про алгоритми нападу на моделі машинного навчання. Знаючи підходи до атак на нейронні мережі, ми розглянемо деякі стратегії захисту від них та введемо відповідну класифікацію. Ми спробуємо покращити системи машинного навчання, зробити їх більш стійким до зловмисних втручань, при цьому залишивши їх придатними до використання. Також необхідним буде ввести метрики стійкості мереж, щоб мати можливість об'єктивно порівняти отримані результати та обрати оптимальні з них.

# 1 Опис проблеми

Для повного розуміння поставленої мети, у цьому розділі ми формалізуємо постановку задачі побудови нейронних мереж, стійких до ошукуючих атак. Також ми опишемо пов'язані терміни та поняття, типи захисту та іншу інформацію, яку потім використаємо в наступних розділах.

## 1.1 Постановка задачі

Нехай система машинного навчання $M$ на основі зразків $x \in S$ робить передбачення $y$. Тут $S$ – множина всеможливих зразків-зображень з предметної області, які допустимі для використання моделлю $M$.

**Означення 1.1** *Зразок* $x^{adv} = x + \tau$, $x^{adv} \in S$ *називається ошукуючим якщо для достатньо малих збурень* $\tau$ *виконуються такі умови:*

$$M(x) = y_{true} \tag{1.1}$$

$$M(x^{adv}) \neq y_{true} \tag{1.2}$$

*де* $y_{true}$ *- правильне передбачення.*

Нашою метою є побудова максимально ефективної моделі машинного навчання, яка буде менш вразливою до такого типу ошукуючих зразків. Задачу захисту моделі від ошукуючої атаки можна формалізувати наступним чином.

Нехай $S^{adv}(M) \subset S$ – множина ошукуючих зразків для моделі $M$. Необхідно знайти модель $M'$ яка є, в певному сенсі, модифікацією оригінальної моделі, таку, що

$$S^{adv}(M') = \emptyset. \tag{1.3}$$

Мається на увазі, що для моделі $M'$ неможливо створити ошукуючі зразки, тобто вона є невразливою до атак.

Такий "ідеальний" випадок є практично неможливим, тому ми будемо використовувати пом'якшене формулювання, а саме вимагатимемо, щоб для моделі-образа задовільнялася умова

$$n(S^{adv}(M')) < n(S^{adv}(M)) \tag{1.4}$$

де $n(S)$ –кількість елементів в множині $S$. Іншими словами, нашою метою є побудова моделі, більш стійкої до атак, ніж оригінал.

## 1.2 Типологія захисту

Зважаючи на те, що існує багато абсолютно різних варіантів та підходів до захисту нейронних мереж від атак, доцільним є ввести певну класифікацію.

Першим спадає на думку поділ *за типом атак*, проти яких ми намагаємось побудувати захист. Тоді всі стратегії захисту можемо розподілити на три групи.

- **Захист від ошукуючих атак.** В основу ошукуючих атак покладена модифікація вхідних даних таким чином, щоб модель машинного навчання не могла розпізнати їх вірно. Така вразливість спричинена тим, що більшість методів машинного навчання розроблені для роботи над конкретними проблемами, набори даних для навчання та тестування яких генеруються з одного і того ж статистичного розподілу. Коли ці моделі застосовуються в реальному світі, супротивники можуть надавати зразки, які порушують це статистичне припущення, що призводить до компрометації результатів. Існує багато різноманітних алгоритмів і підходів до побудови захисту від такого типу атак і деякі з них ми розглянемо у цій роботі.

- **Захист від викрадення.** Розробка високопродуктивних глибоких нейронних мереж зазвичай потребує багато зусиль, часу та грошей. Саме тому зловмисники шукають шляхи як заволодіти цінною інформацією, яка допоможе відтворити модель за умов обмеженого доступу. Наприклад, вони можуть використати результати її передбачень для генерації власного тренувального набору та аналізу градієнтів. Завдання захисника у такому випадку – зберегти якомога більше інформації в таємниці, щоб не надати супротивнику можливість побудувати аналогічну модель.

- **Захист від отруєння.** Отруєння моделі зазвичай передбачає забруднення навчальних даних, на яких тренується модель. Воно вважається атакою на цілісність, оскільки втручання в навчальні дані впливає на здатність моделі видавати правильні прогнози через зсув границі рішень. Найпоширеніші заходи захисту від таких атак це попередня обробка даних та обмеження доступу до них.

Оскільки нам доступні різні підходи до побудови захисту, варто також виокремити ще одну вертикаль класифікації - *за стратегією захисту*. Тут можемо виділити такі стратегії, як:

- **Модифікація архітектури моделі та процесу тренування.** Ця стратегія включає в себе зміну кількості шарів та нейронів у них або додавання до вже наявної моделі нових шарів специфічної природи,

які, наприклад, можуть відсіювати зайвий шум, зменшувати розмірність зразка чи додавати до результатів деякі випадкові величини, щоб зробити важчим завдання пошуку ошукуючих градієнтів. Процес і алгоритм тренування також може бути зміненим, щоб отримати модель, стійкішу до атак.

- **Генерація специфічного тренувального набору.** Додавання спеціально підготованих зразків до тренувального набору також може покращити стійкість моделі до атак. Одним із очевидних підходів є розширення множини вхідних даних власноруч згенерованими ошукуючими зразками, щоб охопити більшу предметну область. Деякі науковці також розглядають підходи до модифікації міток для вже існуючих екземплярів з метою кодування в них додаткової інформації.

- **Створення захисної оболонки.** Уже готова модель машинного навчання може бути поміщена в обмежене середовище, яке містить в собі засоби для захисту від атаки. Тобто оболонка моделі – це проксі-середовище, яке може по різному обмежувати комунікацію з самою моделлю. Наприклад, додаткова нейронна мережа може бути застосована для визначення шкідливих вхідних даних та недопущення таких даних до основної моделі. Захисник також може створити обмеження на кількість запитів, допустимих для одного користувача, та часовий інтервал між ними. Інформація про надмірну активність окремих користувачів може бути використана для ідентифікації атак.

Важливим фактором є те, що до одної нейронної мережі можна одночасно застосовувати різні типи захисту, що в деяких випадках може сприяти покращенню стійкості мережі до атак. Але потрібно зважати на те, що деякі стратегії захисту незначно погіршують результати моделі або збільшують затрати на її тренування та підтримку. Тому застосування надмірної кількості стратегій захисту може зробити модель непридатною для використання. У даній роботі ми сконцентруємося на дослідженні захисту від ошукуючих атак застосувавши різні підходи та стратегії, щоб спробувати досягти оптимальних результатів.

# 2 Глибокі нейронні мережі

## 2.1 Будова штучної нейронної мережі

*Штучні нейронні мережі (Artificial Neural Networks)* – це набір алгоритмів машинного навчання, натхненних роботою людського мозку. Вони ґрунтуються на сукупності з'єднаних один з одним вузлів, які називають штучними нейронами, що можуть передавати між собою інформацію. Нейрон, отримавши сигнал від інших нейронів, обробляє його і передає далі по нейронній мережі.

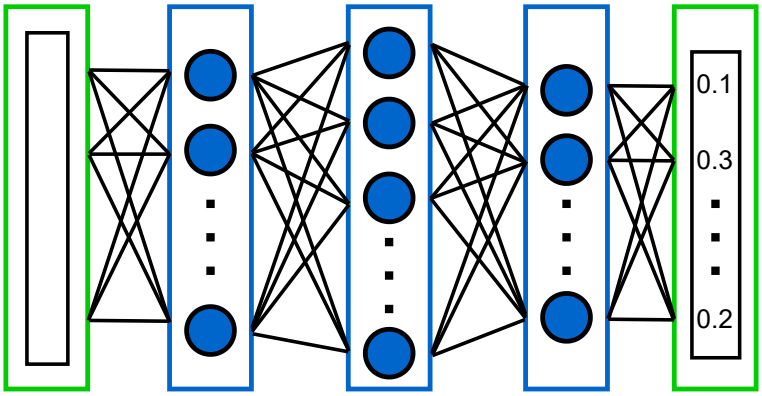



*Рисунок 1.* Штучна нейронна мережа.

На практиці, дані, які передаються між нейронами – це дійсні числа, а вихід кожного нейрона обчислюється деякою функцією від суми його входів, яку називають *функцією активації*. До лінійної частини також додають ще один параметр – *зміщення (bias)*, щоб краще регулювати вихід нейрона. Тож роботу нейрона можна описати формулами

$$\begin{aligned} z &= w^T x + b \\ \hat{y} &= g(z), \end{aligned} \tag{2.1}$$

де $z$ називають *лінійною частиною*, $x \in \mathbb{R}^n$ – сукупність входів, $w \in \mathbb{R}^n$ – ваговий вектор, $b \in \mathbb{R}$ – зміщення, $g(z)$ - функція активації.

Всі нейрони в межах мережі можна поділити на окремі групи – *шари*. Нейрони в межах одного шару є незалежними між собою, їх виходи обчислюються лише на основі виходів попередніх шарів. Набір початкових даних (наприклад набір зображень і т.д.) називають *вхідним шаром* і при обчисленні кількості шарів його не враховують. Шар, який повертає кінцевий результат (наприклад вектор з ймовірностями приналежності зображення до певних класів), називають *вихідним шаром*. Решту шарів називають *прихованими*, а нейронні мережі, які мають хоча б один прихований шар – *глибокими (deep)*.

Для нейронних мереж особливо важливо, щоб функція активації була *нелінійною*, що дозволяє моделі відображати значно складніші та більш комплексні залежності між вхідними параметрами та шуканим значенням. В протилежному випадку ми зможемо отримати лише лінійну границю рішень. Без

нелінійних функцій активації, незалежно від кількості шарів та нейронів в них, нейронна мережа буде працювати так само, як нейронна мережа з одного нейрона.

Нейронні мережі для багатокласової класифікації займаються задачами, в яких невідома величина може належати до одного з двох або більше класів. Кількість класів позначатимемо літерою $C$. Вхідні екземпляри - це вектори $x \in \mathbb{R}^n$. Кожному зразку $x$ поставлено у відповідність вектор $y$, у якому один елемент – одиниця, відповідає класу зразка $x$, а решта значень – нулі. З пар значень $(x, y)$ складаються тренувальний та тестувальний набори моделі.

Для задач багатокласової класифікації як функцію активації останнього шару НМ використовують функцію *Softmax*:

$$\hat{y}(x) = \left[ \frac{e^{z_i(x)}}{\sum_{i=0}^{C-1} e^{z_i(x)}} \right]_{i \in 0 \ldots C-1} \tag{2.2}$$

$\hat{y}$ – вектор-передбачення для вхідних даних. Елементи векторів є ймовірностями, що екземпляр $x$ належить до відповідних класів. Після отримання $\hat{y}$ для зразків $x$ роботу мережі оцінюють за допомогою *функції витрат*. У випадку багатокласової класифікації це функція крос-ентропії:

$$\xi(Y, X) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{C} y_j^{(i)} \log\left(\hat{y}_j^{(i)}\right) = J(W, b), \tag{2.3}$$

де верхній індекс $(i)$ відповідає $i-$ му рядку в матриці, а $W$ та $b$ – сукупність всіх ваг і зміщень в НМ.

# 3 Визначення стійкості

Під час проведення досліджень важливою є можливість порівняти отримані результати, щоб визначити ефективність того чи іншого методу та вибрати підхід до тренування та застосування моделі, який робить її більш стійкою до ошукуючих зразків.

Отже, потрібно якимсь чином виміряти стійкість моделі. Тут варто згадати, якими метриками користуються зловмисники. У нашій попередній роботі [14] ми вже згадували, що під час атак основною метою є знайти зразок $x^{adv}$ який є найближчим до оригінального зразка $x$. Щоб визначити "близькість" ошукуючих зразків зловмисники часто використовують норму $L_p$, де $1 \leq p < \infty$. Вона має вигляд

$$\|x - \tilde{x}\|_p = \left( \sum_{i=1}^{n} (x_i - \tilde{x}_i)^p \right)^{\frac{1}{p}} \tag{3.1}$$

Природнім є обмежувати розмір шуму, який зловмисники накладають на оригінальні зображення. Ці обмеження також часто є залежними від вищезгаданої норми. Ба більше, як ми вже згадували в нашій попередній роботі, деякі дослідники розглядають проблему мінімізації шуму як частину задачі створення ошукуючих зразків [13].

Значення (3.1) буде дуже малим, якщо шум складається з невеликих змін для багатьох пікселів, що, загалом, свідчить про те що людині складно буде відрізнити ошукуючий зразок від оригінального. З цього припущення випливає, що якщо шум $\tau$ є достатньо великим, то ошукуючий зразок було важче створити і його легше можна виявити. Далі розглянемо можливі варіанти визначення стійкості проти такого ошукуючого зразка.

Нехай $B_p(r) := \{\tau \in \mathbb{R}^n : \|\tau\|_p \leq r\}$ це куля, утворена на основі $p$ -норми з радіусом $r$. Для заданої моделі класифікації $M$ та фіксованого вхідного зразка $x \in \mathbb{R}^n$ зловмисники можуть успішно створити ошукуючий зразок $x^{adv}$ з розміром шуму $L$ для обраної $p$-норми якщо їм вдасться знайти $\tau \in B_p(L)$ таке, що $M(x + \tau) \neq M(x)$. Узагальнивши, можна сказати що супротивники намагаються знайти якнайменше $\tau$ яке змінить результат класифікації.

Інтуїтивно зрозуміло, що модель можна розглядати як стійку, в певному сенсі, до ошукуючих зразків, якщо результати її передбачень є нечутливими до малих змін будь-якого прийнятного вхідного зразка. Важко формально визначити, які зразки можна вважати прийнятними. Оскільки перевірка продуктивності моделі проводиться на тестувальних даних, які не використовують під час тренування моделі, то такий набір можна застосувати і як початкові дані для оцінки стійкості.

**Означення 3.1** *Модель $M$ називається стійкою до атак за нормою $L_p$ на заданому зразку $x$ коли $M(x) = M(x + \tau)$ $\forall \tau \in B_p(L)$. Якщо $M$ – це НМ для багатокласової класифікації, то рівняння еквівалентне до:*

$$\forall \tau \in B_p(L) : \hat{y}_k(x + \tau) > \max_{i:i \neq k} \hat{y}_i(x + \tau) \tag{3.2}$$

*де $k := M(x)$, $\hat{y}$ – вектор ймовірностей, вихідний результат останнього шару НМ.*

Тобто якщо модель є певною мірою стійкою, то малі зміни вхідного зразка не змінюють оцінки настільки, щоб змінити результат передбачення. Можемо узагальнити даний принцип на весь тренувальний набір, для визначення стійкості моделі використавши середню відстань від оригінальних зразків до ошукуючих.

**Означення 3.2** *Нехай $M$ – нейронна мережа для багатокласової класифікації, $x_i \in X_{test}$ – тестувальний набір, $x_i^{adv} \in S^{adv}(M)$ – ошукуючі зразки, які відповідають прикладам з тестового набору. Тоді стійкість моделі на обраному наборі даних та з використанням відповідної норми визначимо як*

$$r_p(M, X_{test}) := \frac{\sum_{i=1}^{n_{test}} \|x_i - x_i^{adv}\|_p}{n_{test}} \tag{3.3}$$

*де $n_{test} = n(X_{test})$. Також можемо визначити, що $r_p(M) := r_p(M, X_{test})$ при $n(X_{test}) \to \infty$.*

Метрику $r_p(M, X_{test})$ зручно використовувати для оцінки стійкості моделі. Вона дає можливість чисельно порівняти роботу до та після застосування одної з стратегій захисту та при застосуванні різних алгоритмів.

# 4 Огляд атак на моделі машинного навчання

Перш ніж приступити до побудови захисту, варто розглянути деякі з алгоритмів генерації ошукуючих зразків, які широко використовуються для оцінок стійкості моделей. Серед них виділяють атаки на білу скриньку (коли зловмисник володіє усією інформацією про модель, такою як, наприклад, архітектура, функції активації, ваги моделі і т. д.) та на чорну скриньку (зловмиснику нічого не відомо про модель, може бути можливість застосувати модель для класифікації власних зразків). Обидва типи атак мають власну специфіку та підходи до вирішення поставленої задачі, але базуються на додаванні до оригінальних зразків штучно створеного шуму.

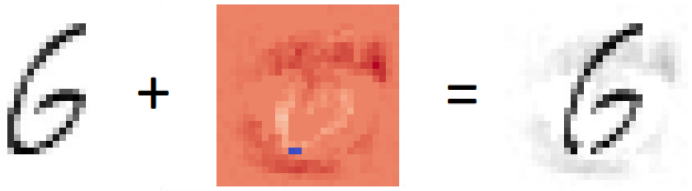


*Рисунок 2.* Початкове зображення класифікується нейронною мережею як “шість”. Після додавання до нього певного шуму НМ класифікує його як “п’ять”.

Нейронні мережі є особливо чутливими до атак на білу скриньку. Отримавши доступ до всіх параметрів, у зловмисників з’являється можливість обчислити точні похідні для функцій (у випадку диференційовності моделі), за допомогою яких виконується класифікація. Тоді під час атаки можна застосувати різні чисельні методи для вирішення конкретної проблеми, що дозволяє досягти високої ефективності згенерованих ошукуючих зразків. Окрім того, складно створити дієві механізми захисту моделі від таких атак, бо зловмисник може врахувати їх при створенні ошукуючих зразків.

## 4.1 Швидкий градієнтний спуск

Метод швидкого градієнтного спуску є одним з найпростіших методів генерації ошукуючих прикладів. Його застосовують для атак на різні системи машинного навчання, зокрема й на нейронні мережі [8].

Основною метою методу є максимізація функції витрат $J$, розглядаючи її як $J(x, y_{true})$, зафіксувавши ваги нейронної мережі. В такому випадку потрібно знайти $x^{adv}$ таке, що виконується нерівність:

$$J\left(x^{adv}, y_{true}\right) > J\left(x, y_{true}\right) \tag{4.1}$$

Тоді, згідно з [9], ми зможемо знайти розв’язок поставленої задачі. Пошук ошукуючих зразків здійснюємо за формулою:

$$x^{adv} = x + \epsilon sign\left(\nabla_x J\left(x, y_{true}\right)\right) \tag{4.2}$$

де $\epsilon$ - додатний параметр, який називатимемо *розміром кроку*. Напрямок $sign\left(\nabla_x J\left(x, y_{true}\right)\right)$ є *напрямком зростання* для функції $J$.

Варто зауважити, що метод не гарантує знаходження ошукуючого зразка, бо нерівність (4.1) свідчить лише про збільшення функції правдоподібності. Тому в загальному випадку доцільно використовувати ітераційну форму методу. Також можна застосовувати інші методи, такі як [13], для того, щоб зменшити розмір шуму $\tau$.

## 4.2 Оптимізація нульового порядку

Більшість реальних систем, які використовують нейронні мережі, не розголошують своїх конфігурацій (як от структура мережі і її ваги), тому атаки, на кшталт швидкого градієнтного спуску не так просто застосувати на практиці. В деяких сценаріях зловмисники мають можливість використовувати модель для отримання передбачень на основі власних зразків (як от, наприклад, при використанні програмних інтерфейсів від Google AI чи Amazon). Запити до мережі дозволено повторювати необмежену кількість разів і на основі результатів покращувати ошукуючі зразки.

Тоді є можливість обчислити градієнт наближено, для чого достатньо лише результату класифікації $f(x)$ та відповідного вхідного параметра $x$, як показано в [11]. Якщо обчислимо часткові похідні наближено, за формулою

$$\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}_i} \approx \lim_{h \to 0} \frac{J\left(\mathbf{x} + he_i\right) - J\left(\mathbf{x} - he_i\right)}{2h}, \tag{4.3}$$

де $e_i$ - вектор, в якому $i$-тий елемент дорівнює одиниці, а всі решта – нулю, то зможемо застосувати для атаки на чорну скриньку алгоритми, що використовують похідні для розв'язання задачі оптимізації, такі як швидкий градієнтний спуск.

При використанні наближених значень похідних в методі швидкого градієнта (4.2) вдається досягти результативності, близької до тої, яку отримуємо під час атак на білу скриньку. Це зумовлено тим, що для ефективної роботи методів достатньо лише знаку градієнту. Те, що значення обчислені з певною похибкою, значним чином не впливає на результат. Недоліком такого сценарію є необхідність виконувати класифікацію значну кількість разів, що займає багато часу та ресурсів і може бути легко виявлено.

# 5 Захисна дистиляція та її модифікації

В цьому розділі ми розглянемо підхід до оптимізації нейронної мережі або групи нейронних мереж, метою якого є зменшення обчислювальних затрат та ресурсів під час функціонування моделі. Це підхід був розроблений інженерами компанії Google Хілтоном на ін. і описаний в [3]. Не зважаючи на те, що першочерговою метою дистиляції є зменшення розміру нейронної мережі щоб зробити можливим розгортання на пристроях з обмеженими ресурсами, згідно з [2], її побічні ефекти можуть зробити мережу стійкішою до ошукуючих атак.

## 5.1 Дистиляція нейронних мереж

Основна ідея методу дистиляції полягає у перенесенні знань з мережі прообразу за допомогою векторів імовірності приналежності зразка до деякого класу, отриманих використовуючи вже існуючу нейронну мережу. На основі цих даних тренується нова нейронна мережа з меншою розмірністю, при чому без значних втрат точності. Ця ідея базується на тому, що знання, здобуті моделлю під час тренування, можна отримати не лише безпосередньо з ваг НМ, вони також "закодовані" в її передбаченнях.

Як ми уже згадували раніше, кожен результат передбачення містить у собі деяку інформацію про модель. Таким чином вектори імовірності $\hat{y}$, отримані з одної моделі містять у собі достатньо інформації, щоб з їх допомогою можна було відтворити модель. Дистиляція "витягує" знання з цих векторів і переносить їх на модель з менш комплексною архітектурою.

Використовуючи уже натреновану та готову до використання модель, кожному зразку тренувальних даних ставиться у відповідність вектор ймовірностей приналежності зразка до кожного з класів, які належать до предметної області моделі. Перевага використання таких прогнозів полягає у тому, що тепер кожна мітка містить не просто жорсткі оцінки, а інформацію про приналежність зразка до кожного з класів.

Процедура тренування прообразу не має строгих обмежень і нічим не відрізняється від звичних підходів. Наприклад, можна застосувати градієнтний спуск на даних з тренувального набору. Тепер можемо ввести поняття передавального набору.

**Означення 5.1** *Передавальним набором даних моделі $M$ називається тренувальний набір $X_{train}$, кожному з елементів якого поставлено у відповідність вектор ймовірностей $\hat{y}$, такий, що $\sum_{i=0}^{C-1} \hat{y} = 1$ і $M(x) = argmax(\hat{y})$ $\forall x \in X_{train}$, де $C$ – кількість класів з предметної області моделі.*

На основі векторів ймовірності, створених першою НМ, формується передавальний набір. Після цього нова нейронна мережа з такою ж архітектурою

та гіперпараметрами тренується на передавальному наборі. Очікується, що за рахунок передачі знань від моделі-образа дистильована модель зможе краще узагальнювати зразки, передані для передбачень, що і буде причиною кращої стійкості до ошукуючих атак.

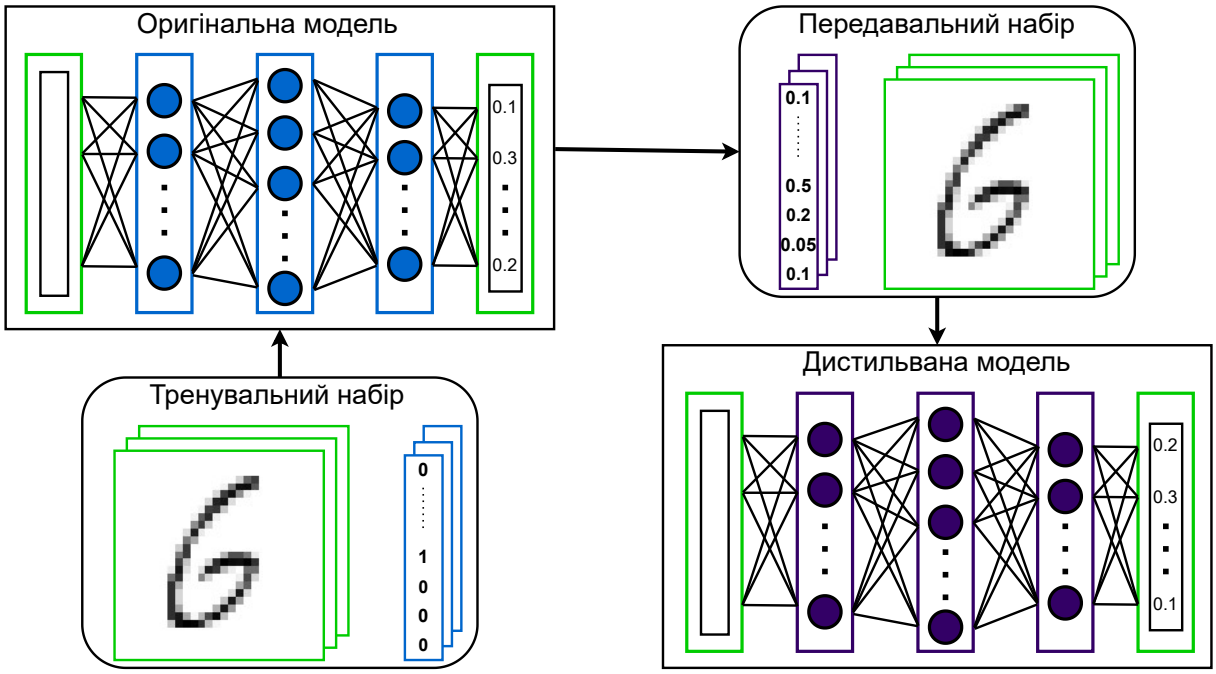



*Рисунок 3.* Ідея тренування моделі з використанням захисної дистиляції.

## 5.2 Процес тренування та основні параметри

Процес дистиляції розпочинається з підготовки мережі-прообразу, вихідний шар якої побудований на основі функції – Softmax з врахуванням параметра $T$.

$$\hat{y}(x) = \left[ \frac{e^{z_i(x)/T}}{\sum_{i=0}^{C-1} e^{z_i(x)/T}} \right]_{i \in 0 \ldots C-1} \tag{5.1}$$

У межах Softmax шару кожен нейрон відповідає класу за індексом $i \in 0 \ldots C-1$ (де $C$ - кількість класів) і обчислює відповідний компонент за формулою 5.1, де $z(x) = z_0(x), \ldots, z_{C-1}(x)$ – $C$ логітів, лінійної частини останнього шару нейронної мережі, що відповідають виходу прихованого шару для кожного з $C$ класів у наборі даних.

Параметр $T$ також називають температурою дистиляції. Він впливає на те, наскільки елементи вектора ймовірностей $\hat{y}$ будуть відрізнятись один від одного. Температура дистиляції відіграє центральну роль у основних явищах дистиляції, що ми покажемо пізніше у цьому розділі.

Так при малих $T$ різниця між ймовірностями буде значною, тобто один із елементів вектора буде сильно виділятись з-поміж інших. В свою чергу, висока температура змушує НМ створювати ймовірнісні вектори з відносно великими значеннями для кожного класу.

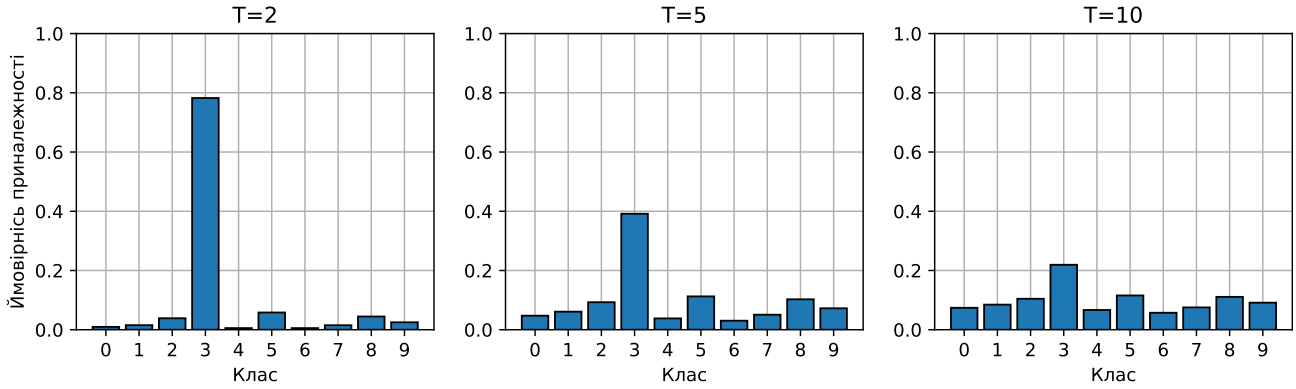



*Рисунок 4.* На даному рисунку розглянуто розподіл ймовірностей приналежності зразків класу "трійка" з тренувального набору до інших класів на основі класифікації . Легко бачити що при збільшенні температури розподіл прямує до рівномірного. Варто звернути увагу на те, що ймовірності для класів "двійка", "п'ятірка" та "вісімка" є істотно більшими, що свідчить про візуальну близькість між ними і класом "трійка".

Дійсно, при високій температурі логіти у векторі $z(x)$ стають незначними у порівнянні з температурою $T$. Тому всі компоненти векторів імовірності $\hat{y}(x)$, виражені у рівнянні (5.1) прямують до $1/C$ при $T \to \infty$. Чим вища температура, тим більш неоднозначним буде розподіл ймовірностей (тобто всі значення виходу $\hat{y}(x)$ близькі до $1/C$), тоді як при малій температурі розподіл буде більш дискретним (тобто одна з ймовірностей у виході $\hat{y}(x)$ буде близькою до 1, а решта – близькі до 0).

В найпростішій формі дистиляції знання переносяться до дистильованої моделі через тренування на вище згаданому передавальному наборі даних, використовуючи м'які мітки отримані при високих температурах вихідного шару моделі-оригіналу. Така ж температура використовується під час тренування дистильованої моделі, але після завершення процесу її повертають до стандартного значення, рівного одиниці, так як після завершення процесу тренування зміни температури не мають впливу на кінцевий результат класифікації $M(x)$, а лише на виходи останнього шару моделі.

Проблема цього підходу криється в тому, що точність моделі-прообраза завжди є менша за 100%. Це означає, що передавальний набір буде містити деякі зразки з неправильними мітками і тренування на ньому нової моделі призведе до втрат точності на тестувальному наборі. Якщо відомі точні класи об'єктів, приклади з неправильними мітками варто виключити з передавального набору або замінити їх на правильні жорсткі мітки. Таким чином

процес дистиляції може бути значно покращеним і точність моделі-образу залишиться на високому рівні.

Ще один підхід до розв'язання проблем з передавальним набором наводять автори в [3]. Під час тренування вони пропонують використовувати середнє арифметичне двох різних функцій оцінки. Перша з них – функція крос ентропії , обчислена з використанням такої ж високої температури $T$ в вихідному шарі НМ, яка використовується під час генерації м'яких міток. Друга функція оцінки – функція крос ентропії з правильними жорсткими мітками при температурі $T = 1$.

Оскільки величини градієнтів масштабуються з коефіцієнтом $\frac{1}{T^2}$, то варто домножити їх на $T^2$ при використанні як м'яких, так і жорстких міток. Це гарантує, що відносні частки вкладу м'якої та жорсткої міток залишатимуться приблизно незмінними, якщо температура буде змінюватися під час експериментів.

## 5.3 Оцінка захисту

Для оцінки і аналізу захисту ми застосуємо метод захисної дистиляції, використавши як прообраз власноруч натреновану модель для класифікації цифр на основі набору даних *MNIST*. Гіперпараметри для моделі наведені в додатку 1.

Спершу потрібно дослідити, яким чином гіперпараметри методу вплинуть на стійкість моделі. Як ми вже згадували раніше, ключовий параметр, який використовується під час побудови даного захисту – це температура дистиляції $T$. Спершу дослідимо, як він впливає на стійкість моделі, виміряну за допомогою метрики (3.3).

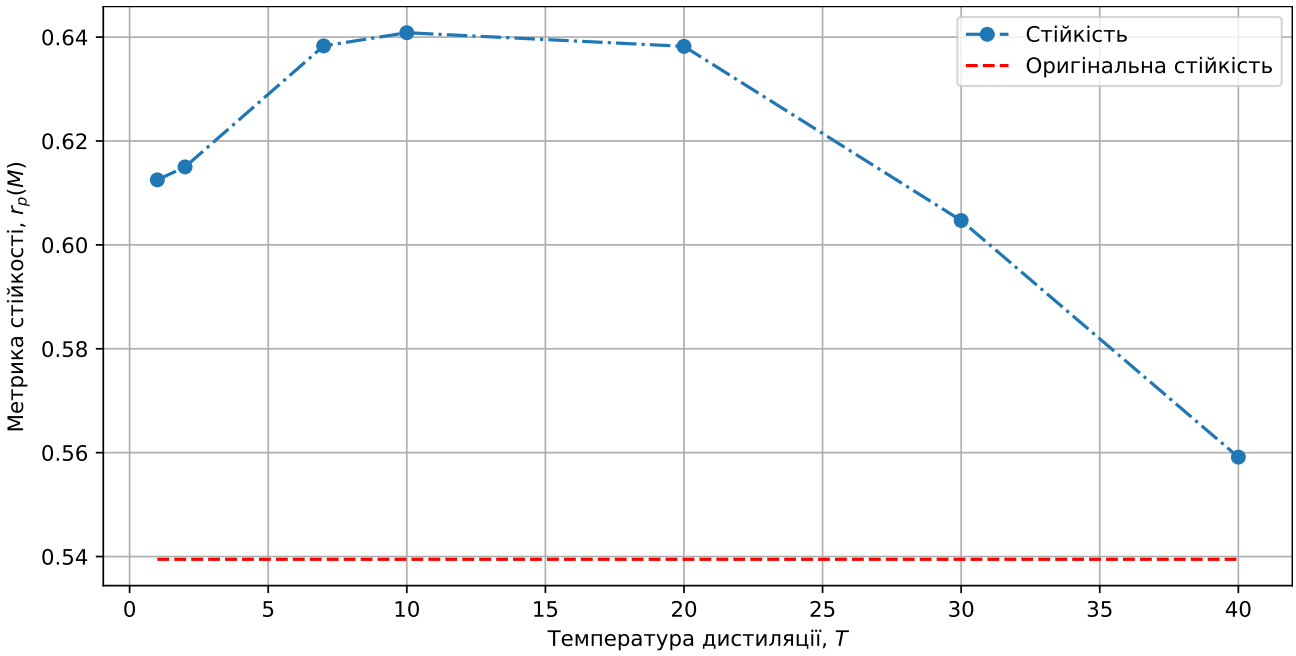


*Рисунок 5.* Залежність стійкості НМ від температури дистиляції. Червоним зображений рівень стійкості моделі-прообразу до застосування захисту.

З рисунка 5 бачимо, що температура дистиляції справді має позитивний вплив на здатність моделі протистояти атакам. Для $T < 20$ помітна тенденція на зростання стійкості до ошукуючих зразків, але потім вона починає падати.

Така поведінка зумовлена тим, що дистильована модель починає втрачати точність при високих температурах (див. Додаток 3). Щоб підвищити її доводиться збільшувати кількість ітерацій тренування, що, в свою чергу, веде до перетренування та зниження стійкості до атак.

Дослідимо діапазон з найвищою стійкістю. Щоб вибрати оптимальну модель, потрібно також звернути увагу на точність дистильованої моделі на зразках з предметної області. Відповідні результати наведені в таблиці нижче.

| $T$ | Точність на $X_{train}$ | Точність на $X_{test}$ | $r_p(M)$ | Приріст стійкості |
|---|---|---|---|---|
| 2 | 92% | 92% | 0.615 | 14% |
| 7 | 90% | 89% | 0.638 | 18.3% |
| 10 | 88% | 87% | 0.641 | 18.8% |
| 20 | 78% | 77% | 0.639 | 18.5% |

Можемо зробити висновок, що для даної моделі оптимальним значенням температури дистиляції є $T = 7$, тому що воно дозволяє отримати відносно стійку модель з мінімальними втратами точності.

Тепер детальніше проаналізуємо поведінку обраної оптимальної моделі під час атаки. Нас цікавить, як буде змінюватись точність моделі під час атаки. Якщо брати до уваги лише ошукуючі зразки, тобто зразки, які до атаки класифікувалися правильно, то побачимо наступну картину.

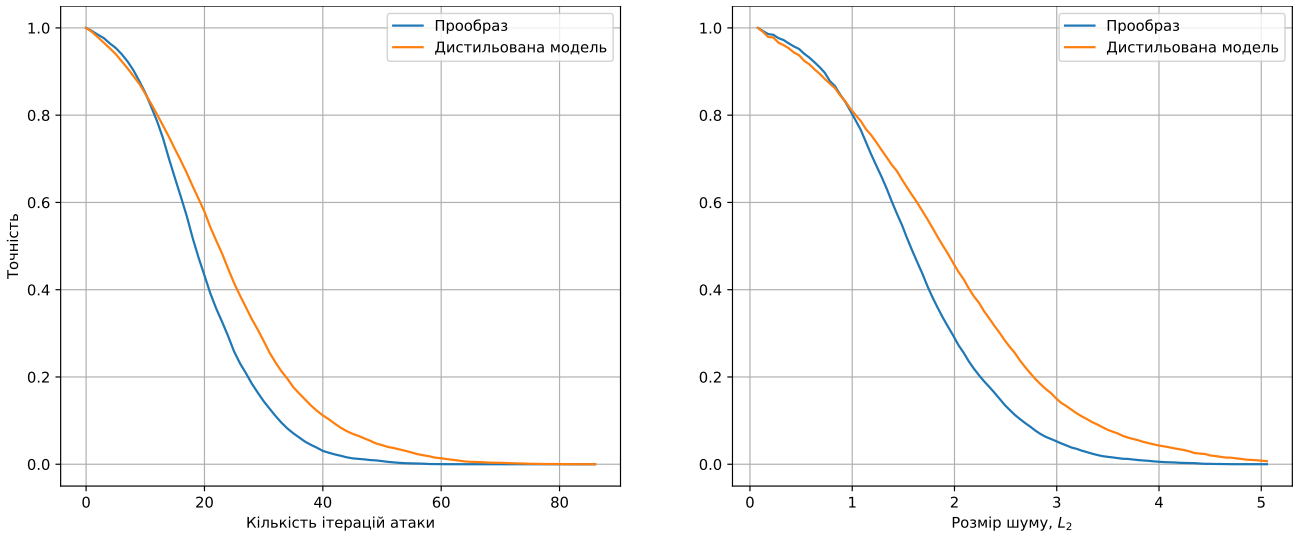


*Рисунок 6.* Точність дистильованої моделі та її прообразу під час атаки. Початкові дані – зразки з тестувального набору $X_{test}$, які правильно класифікувалися моделлю $M$. Температура дистиляції $T = 7$.

Річ у тому, що попередні графіки на рисунку 6 є дещо оманливими, бо не враховують зразків, які одразу класифікувалися неправильно. Взявши їх до уваги, тобто проводячи експерименти на цілому тестувальному наборі, ситуація трохи зміниться.

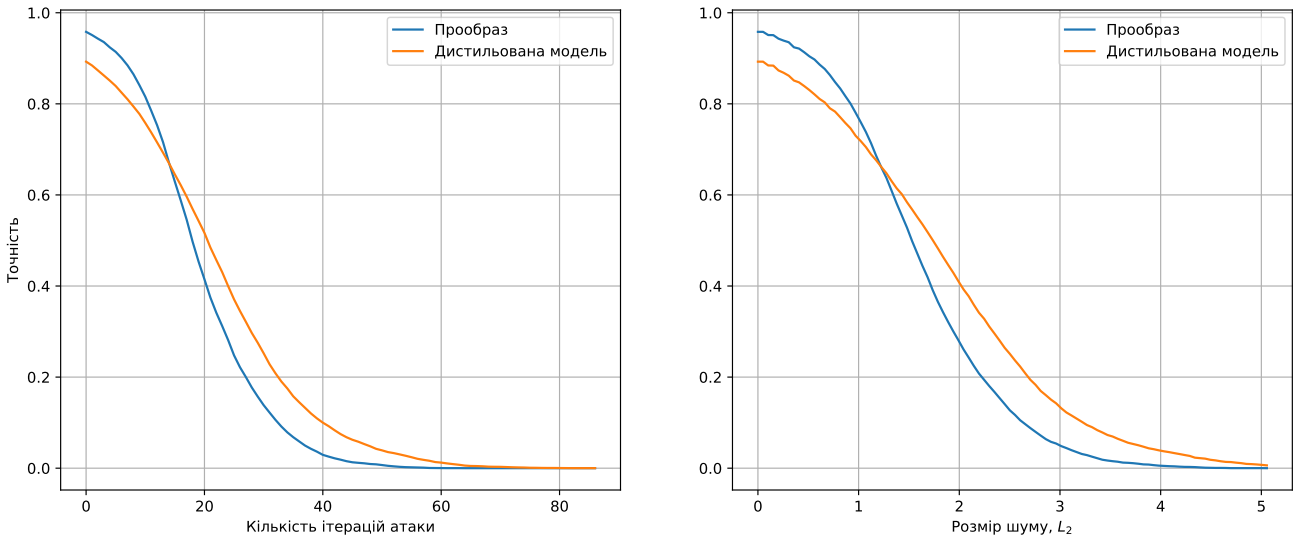



*Рисунок 7.* Точність дистильованої моделі та її прообразу під час атаки. Початкові дані – всі зразки з тестувального набору $X_{test}$. Температура дистиляції $T = 7$.

Опираючись на отримані результати очевидним є те, що захисна дистиляція в основному позитивно впливає на ті елементи тестувального набору, ошукуючі зразки для яких було важче знайти і для початкової моделі.

# 6 Методи з гарантованою стійкістю

Основна проблема багатьох методів захисту, зокрема і розглянутої раніше захисної дистиляції полягає в тому, що їх ефективність важко, або навіть неможливо довести теоретично. Через це часто виникають випадки, коли захист моделі є ефективним лише для конкретних задач і проти конкретних методів атаки, як, наприклад, показано в [12]. Тому цінними є методи, при використанні яких можна гарантувати певний рівень стійкості. Такі методи захисту також називають *сертифікованими*. Один з цих методів ми і розглянемо в даному розділі.

## 6.1 Захист PixelDP

*PixelDP* - один з сертифікованих методів захисту моделей машинного навчання від ошукуючих атак. Він базується на додаванні випадкового шуму до обчислень, які виконує модель. Цей метод був запропонований Матіасом Лекуєром та ін. в [4]. В основі цього підходу лежить принцип *диференційованої конфіденційності (differential privacy, DP)*, який застосовують для запобігання викриття інформації про окремі елементи під час виконання обчислень на великих базах даних [7].

Суть диференційованої конфіденційності – гарантувати, що незначні зміни в базі даних, такі як видалення або заміна декількох рядків, гарантовано призведуть до *обмеженої* зміни в розподілі виходів алгоритму. Аналогічно, PixelDP гарантує, що невеликі зміни вхідного зразка $x$ призведуть до обмеженої зміни вектора ймовірностей $\hat{y}$, яка не вплине на результат класифікації $M(x)$. В попередніх розділах ми визначили цю властивість як стійкість моделі до ошукуючих атак. Таким чином зловмисникам буде важче знайти ошукуючі зразки і їх можна буде легше виявити.

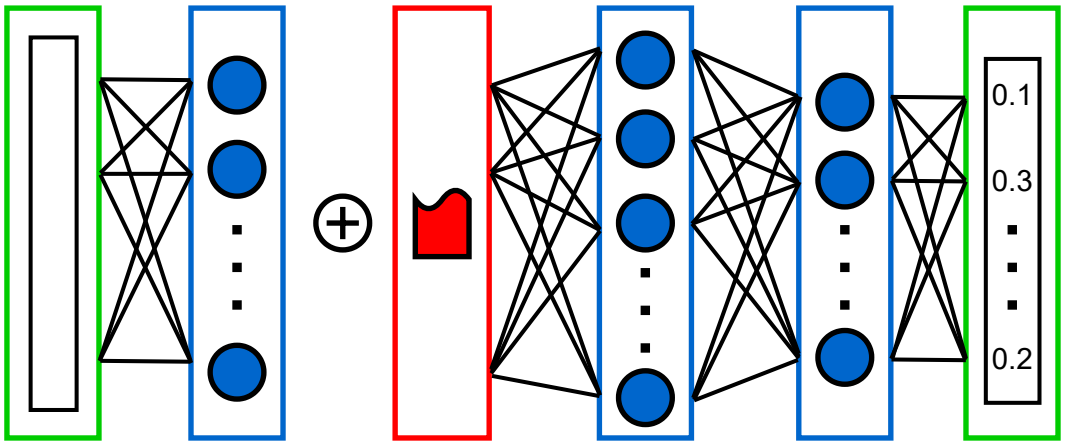



*Рисунок 8.* Архітектура нейронної мережі після застосування захисту PixelDP. Червоним позначено новий шар для генерації захисного шуму.

Як зображено на рисунку 8, в архітектуру мережі додають шар генерації шуму, що рандомізує передбачення НМ. За рахунок цього вдається досягти обмеженої чутливості очікуваного результату до змін вхідного зразка [4].

## 6.2 Застосування та основні параметри

Процедура тренування моделі, до якої застосовуємо даний захист, є схожою до звичайної процедури тренування: можна використовувати оригінальну функцію витрат та метод оптимізації. Основна відмінність полягає у вимогах до шару НМ, після якого ми накладатимемо шум – він повинен мати обмежену чутливість до зміни вхідних зразків.

**Означення 6.1** *Чутливість шару нейронної мережі – це максимальна зміна виходів шару, яка може бути спричинена зміною вхідного зразка, визначена за допомогою метрик відстані ($L_p$ і $L_q$ норм для входу та виходу відповідно). Позначатимемо її як $\Delta_{p,q}$.*

$$\Delta_{p,q} = \Delta^g_{p,q} = \max_{x \neq \tilde{x}} \frac{\|g(x) - g(\tilde{x})\|_q}{\|x - \tilde{x}\|_p} \tag{6.1}$$

*де $x$, $\tilde{x}$ – два різні вхідні зразки, $g(x)$ – функція, яка відображає роботу відповідного шару.*

Обчислення чутливості функції $g$ залежить від того, де ми розмістимо шар з захисним шумом. Наприклад, якщо додавати збурення одразу до класифікованого зображення, то очевидно, що чутливість буде рівна одиниці. В інших випадках може бути необхідним внести додаткові модифікації, такі як нормалізацію входів чи заміну функції активації, щоб чутливість шару можна було обчислити.

Шум, який генерується під час захисту, позначатимемо як $noise(\Delta, L, \epsilon, \delta)$, де $\Delta$ – чутливість попереднього шару, $L$ - очікуваний розмір атаки. Є кілька варіантів вибору розподілу для захисного шуму:

- *Розподіл Лапласа* з математичним сподіванням $\mu = 0$ та стандартним відхиленням $\sigma = \sqrt{2}\Delta_{p,1}L/\epsilon$.
- *Розподіл Гауса* з математичним сподіванням $\mu = 0$ та стандартним відхиленням $\sigma = \sqrt{2\ln\left(\frac{1.25}{\delta}\right)}\Delta_{p,2}L/\epsilon,\ \epsilon \leq 1$

де $\epsilon$ та $\delta$ – параметри диференційованої приватності. Під час тренування модель використовує фіксоване значення шуму, яке обчислюється один раз, а під час проведення передбачень щоразу використовуються інші значення шуму.

Варто зауважити, що результати роботи нейронної мережі будуть відрізнятися для кожного з експериментів, тому що вони залежать від випадкової величини. На основі одного вектора ймовірностей $\hat{y}$ неможливо визначити правильне передбачення, тому потрібно оцінити очікуване передбачення рандомізованої нейронної мережі. Таку оцінку можна отримати виконуючи передбачення для конкретного зразка $x$ велику кількість разів. Позначатимемо оцінку передбачення НМ як $M_n(x)$, де $n$ – кількість виконаних передбачень, тобто

$$M_n(x) = argmax\left(\frac{1}{n}\sum_{i=0}^{n-1}\hat{y}(x)\right) \tag{6.2}$$

За рахунок використання $M_n(x)$ при великих $n$ результати захищеної мережі прямують до результатів мережі прообразу, що означає відсутність втрат точності в порівнянні з оригіналом. Основний недолік цього методу якраз і полягає у процедурі обчислення $M_n(x)$, тому що для отримання коректного результату захищена модель витрачає в $n$ разів більше часу, ніж модель-прообраз. Чим більший розмір захисного шуму, тим більше $n$ необхідно обрати для отримання точного результату.

## 6.3 Аналіз захисту

Створення ошукуючих зразків для моделі, захищеної за допомогою алгоритму PixelDP містить у собі ряд перешкод. Спершу, варто зазначити що оцінювати цей захист з застосуванням звичайного аналізу ошукуючих градієнтів не дає жодного результату, так як за рахунок рандомізації обчислень на основі лише одного передбачення практично неможливо визначити, в якому напрямку потрібно рухатись від оригінального зображення до ошукуючого. Тому для обчислення ошукуючих градієнтів варто застосувати такий самий підхід, як і для визначення передбачення. Мається на увазі, що ошукуючі градієнти слід обчислювати певну кількість разів, а потім знаходити їх середнє значення і опираючись на нього додавати ошукуючий шум.

Ще одна особливість PixelDP-мережі – під час одної класифікації зразок може бути ошукуючим, а під час іншої – ні. Це додає атакуючим ще більше труднощів. Якщо вони використовують ітераційну атаку в якій знаходження ошукуючого зразка є умовою зупинки, то їхній алгоритм буде зупинятись після деякої обмеженої кількості ітерацій думаючи, що ошукуючий зразок знайдено, але насправді в більшості випадків мережа класифікуватиме його правильно.

# Висновки

Під час виконання цієї роботи ми розглянули низку методів та стратегій захисту глибоких нейронних мереж від ошукуючих атак. Дослідження відбувались у кілька послідовних етапів, що дало нам можливість всебічно проаналізувати причини вразливості та поведінку захищеної моделі під час атаки.

Спершу ми формалізували поставлену задачу та ввели класифікацію алгоритмів захисту моделей машинного навчання за їх природою та призначенням. Це дало нам можливість локалізувати проблему та приступити до її вирішення.

Ми визначили надійність глибоких нейронних мереж як їх стійкість до обмежених збурень вхідних даних. Іншими словами, надійна модель повинна показувати високу точність не лише всередині навчального набору даних, а й за його межами, тобто моделювати функцію, яка "інтуїтивно" розподіляє віхідні дані між запропонованими категоріями в околі даного зразка. Цей окіл може бути визначений нормою у відповідному просторі.

Щоб зрозуміти природу та походження ошукуючих зразків, ми дослідили поширені алгоритми нападу, які використовують зловмисники. Для атак, які базуються на створенні ошукуючих зразків, характерним є використання градієнтів, обчислених для оцінки чутливості мережі до зміни вхідних даних. Очевидно, що при великих значеннях ошукуючих градієнтів зловмиснику значно простіше знайти відповідний ошукуючий зразок для деякого вхідного зображення, бо невеликі збурення вхідного зразка призведуть до значної зміни результату передбачення.

На основі цих диних ми зробили висновок, що, щоб збільшити стійкість мережі до ошукуючих атак, необхідно зменшити різку залежність від вхідних даних, а отже і амплітуду коливань ошукуючих градієнтів. Це означає що нам потрібно "згладити" модель, отриману під час стандартного процесу навчання, щоб допомогти їй краще узагальнювати зразки, які не є частиною тренувального набору. З цим завдання непогано справляється алгоритм захисної дистиляції, в основі якого лежить передача знань від оригінальної моделі за допомогою передавального набору. З її допомогою нам вдалося значно покращити стійкість моделі до ошукуючих зразків з шумом середніх розмірів, при чому без значних втрат точності та додаткових обчислювальних затрат.

Якщо ж необхідно гарантувати стійкість моделі для достатньо малих значень шуму – в пригоді стане захист PixelDP, що базується на принципі диференційованої приватності і рандомізує обчислення нейронної мережі. Через труднощі в обчисленні ошукуючих градієнтів зловмисникам буде важче знайти ошукуючі зразки хорошої якості. Основний недолік даного алгоритму полягає в тому, що для оцінки правильного результату потрібно виконувати передбачення значну кількість разів, що займає багато часу та обчислювальних ресурсів.

Отож, як показали наші дослідження, побудова захисту від ошукуючих

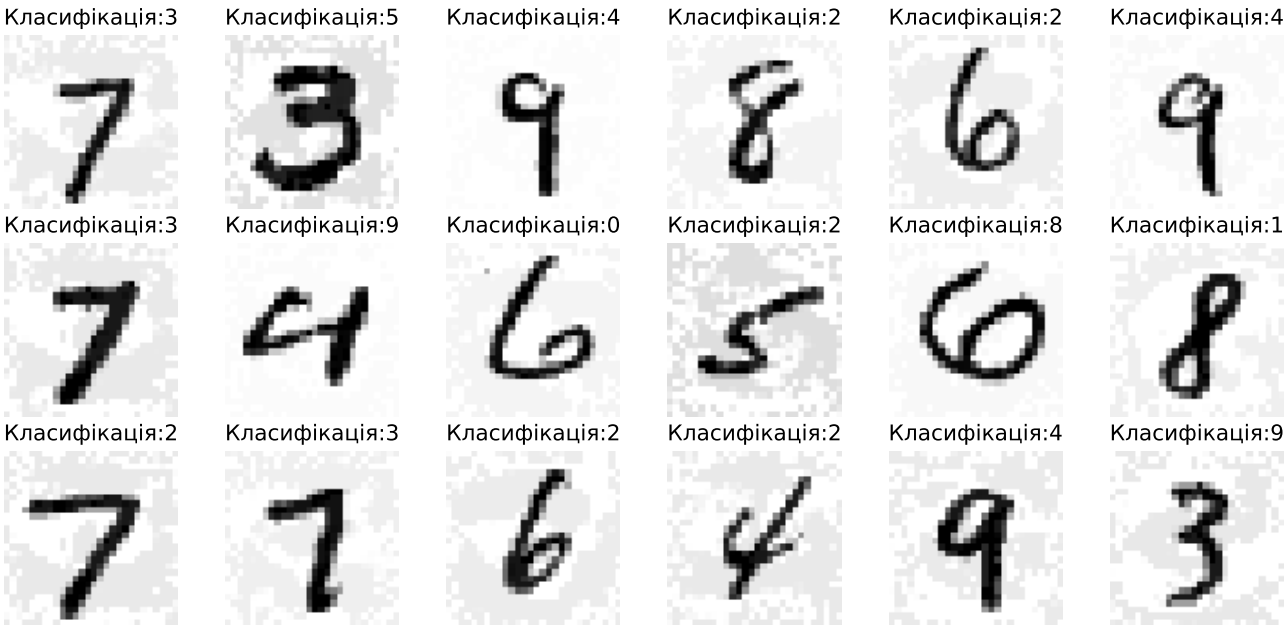
атак – це нетривіальна задача, яка на даний момент не має однозначного та абсолютно ефективного вирішення. Відомі стратегії захисту мають багато недоліків та не гарантують 100%-го захисту від атаки. Тому можна зробити висновок, що в цій області ще багато роботи та ідей для майбутніх досліджень.

# Додатки

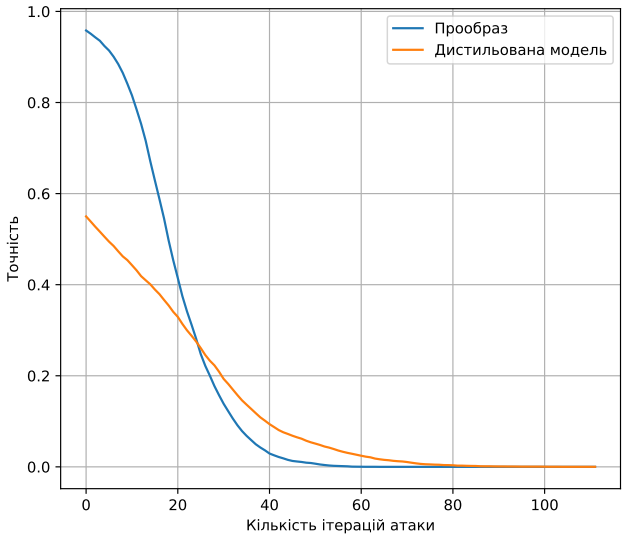
*Додаток 1.* Гіперпараметри щільної (dense) нейронної мережі для класифікації цифр, яка виступала прообразом під час досліджень.

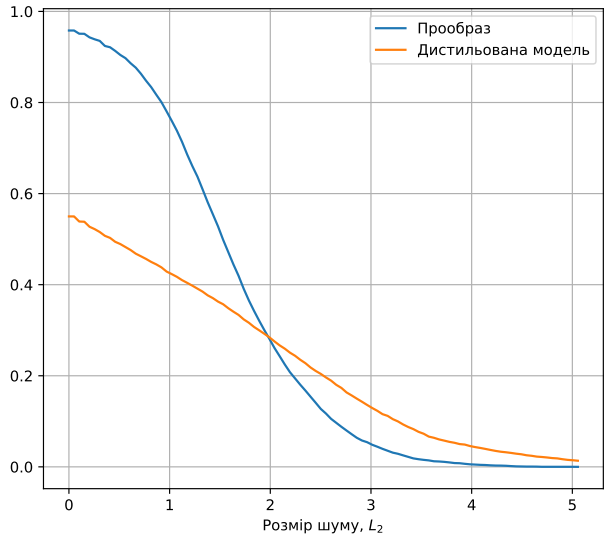
| Функція активації для $l$—го шару $(l < L)$ | Приховані шари | Швидкість навчання $\alpha$ | Кількість ітерацій | Розмір міні-частини |
|---|---|---|---|---|
| $tanh(x)$ | один прихований шар з 60-ти нейронів | 0.1 | 300 | 2048 |

*Додаток 2.* Деякі приклади ошукуючих зразків, отриманих за допомогою ітераційного методу швидкого градієнту.



*Додаток 3.* Для моделі, дистильваної під впливом високої температури (наприклад, при $T = 40$) спостерігається значна втрата точності, зокрема і під час атаки.



# Література


[1] *Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, Rob Fergus* / Intriguing properties of neural networks / arXiv preprint arXiv:1312.6199 (2014)

[2] *Nicolas Papernot, Patrick McDaniel, Xi Wu Somesh Jha,Ananthram Swami* / Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks / arXiv preprint arXiv:1511.04508 (2016)

[3] *Geoffrey Hinton, Oriol Vinyals, Jeff Dean* / Distilling the Knowledge in a Neural Network / arXiv preprint arXiv:1503.02531 (2015)

[4] *Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, Suman Jana* / Certified Robustness to Adversarial Examples with Differential Privacy/ arXiv preprint arXiv:1802.03471 (2019)

[5] *Alhussein Fawzi, Omar Fawzi, Pascal Frossard* / Analysis of classifiers' robustness to adversarial perturbations / arXiv preprint arXiv:1502.02590 (2016)

[7] *Cynthia Dwork* / Differential Privacy / Automata, Languages and Programming. ICALP (2006)

[8] *Ian Goodfellow, Jonathon Shlens, Christian Szegedy* / Explaining and Harnessing Adversarial Examples / arXiv preprint arXiv:1412.6572 (2014)

[9] *Alfio Quarteroni, Riccardo Sacco, Fausto Saleri* / Numerical Mathematics / – Springer, 2000. –300 p.

[10] *Nicolas Papernot, Patrick McDaniel, Ian Goodfellow* / Transferability in machine learning: from phenomena to black-box attacks using adversarial samples / arXiv preprint arXiv:1605.07277 (2016)

[11] *Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, Cho-Jui Hsieh* / ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models / arXiv preprint arXiv:1708.03999 (2017)

[12] *Nicholas Carlini, David Wagner* / Defensive Distillation is Not Robust to Adversarial Examples / arXiv preprint arXiv:1607.04311 (2016)

[13] *Nicholas Carlini, David Wagner* / Towards Evaluating the Robustness of Neural Networks / arXiv preprint arXiv:1608.04644 (2017)

[14] *Богдан Бугрій* / Атаки на глибокі нейронні мережі / Львів (2020)