УДК 004.056.5 **DOI:10.30837/rt.2025.4.223.09**

О.І. ФЕДЮШИН, канд. техн. наук, П.В. ШУЛІК, канд. техн. наук, В.В. ПРОСОЛОВ, Д.О. В'ЮХІН, О.В. ЧЕЧУЙ, канд. техн. наук

ВИКОРИСТАННЯ МЕТОДІВ ГЛИБОКОГО НАВЧАННЯ ДЛЯ ВІЗУАЛІЗАЦІЇ ТА ВИЯВЛЕННЯ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вступ

Шкідливе програмне забезпечення (Malware) залишається однією з найбільш стійких та серйозних загроз у сфері кібербезпеки. Воно постійно еволюціонує, атакуючи широкий спектр цифрових систем починаючи від традиційних настільних і мобільних платформ до новітніх пристроїв Інтернету речей [1]. Обсяги та складність сучасних кібератак невпинно зростають, що створює значні виклики для існуючих захисних механізмів.

Традиційні підходи до виявлення шкідливого програмного забезпечення (ШПЗ), які здебільшого покладаються на сигнатурний аналіз, демонструють дедалі нижчу ефективність. Їхня проблема полягає у нездатності виявляти нові та невідомі загрози. Сучасні зловмисники широко використовують техніки обфускації (заплутування коду), а також поліморфні та метаморфні механізми [2]. Ці методи дозволяють шкідливому ПЗ змінювати свою структуру у кожній новій ітерації, зберігаючи при цьому свою шкідливу поведінку. Як наслідок, сигнатурний аналіз, що шукає точні збіги у коді, стає практично марним проти нових загроз.

З великим обсягом даних, які генеруються кожної секунди, важливо мати ефективні методи для автоматичного виявлення шкідливих програм серед цього потоку інформації. Глибоке навчання (Deep Learning, DL) [3] може допомогти автоматизувати процес виявлення шкідливих програм та відфільтрувати їх з великої кількості даних. Моделі DL здатні вивчати складні, приховані закономірності у даних, що дозволяє їм ідентифікувати шкідливу поведінку навіть у раніше невідомих зразках.

Одним з найбільш перспективних напрямків став підхід, що поєднує кібербезпеку та комп'ютерний зір [1]. Суть методу полягає у візуалізації шкідливого ПЗ: бінарний код виконуваного файлу (наприклад, PE-файлу) конвертується у графічне зображення, наприклад, у відтінках сірого [2]. Це дозволяє трактувати задачу класифікації шкідливого ПЗ як задачу класифікації зображень. Такий підхід відкриває можливість застосування потужних архітектур згорткових нейронних мереж (CNN), які домінують у сфері аналізу візуальних даних. Дослідники активно порівнюють ефективність різних архітектур CNN, а також новітніх моделей, таких як Vision Transformers (ViT) [4], та розробляють гібридні системи, що поєднують аналіз зображень (статичні ознаки) з аналізом поведінкових патернів, наприклад, послідовностей викликів API (динамічні ознаки) [5].

Незважаючи на значний прогрес, питання вибору оптимальної архітектури CNN для задачі класифікації візуалізованого шкідливого ПЗ залишається відкритим, оскільки різні моделі мають різну обчислювальну складність та ефективність у виявленні тонких текстурних відмінностей у зображеннях.

Метою даного дослідження є проведення детального порівняльного аналізу ефективності низки поширених архітектур згорткових нейронних мереж (зокрема, DenseNet169, EfficientNetB0, InceptionV3, MobileNetV2, ResNetV2, VGG16 та XceptionNet) для задачі класифікації сімейств ШПЗ на основі його попередньої візуалізації.

Об'єктом дослідження є процес класифікації ШПЗ на основі методів глибокого навчання та візуалізації даних.

Предметом дослідження є порівняльний аналіз ефективності та результативності (точність, повнота, F1-score) обраних архітектур згорткових нейронних мереж при класифікації зображень шкідливого ПЗ.

В якості концептуальної основи для дослідження запропоновано обрати методи та моделі з використанням Transfer Learning. Це дозволяє виявляти атаки на ранніх стадіях. На їх основі побудована класифікаційна модель та проведено її тестування.

Використання нейронних мереж та методів комп'ютерного зору для аналізу Malware

Зловмисники постійно вдосконалюють свої атаки, і PE-файли залишаються одними з основних векторів розповсюдження шкідливого програмного забезпечення. Розуміння та аналіз цих файлів є важливим для виявлення і нових загроз.

Формат PE використовується в операційних системах Windows для збереження виконуваних файлів і бібліотек. Зловмисники часто маскують свої віруси і троянці, змінюючи PE-файли, і дослідження цього формату дозволяє виявляти такі зміни. Частіше за все аналіз включає в себе аналіз імпортів, секцій, реєстрації, підписів і багато іншого і може використовувати різні методи, наприклад, такі як статичний аналіз, динамічний аналіз, емуляція та дешифрування шкідливого коду. Ці методи дозволяють отримувати більше інформації про загрози. У розробників зловмисного програмного забезпечення стало звичайною практикою використовувати кілька методів обфускації за допомогою метаморфічних та поліморфних методів для створення варіантів існуючого сімейства зловмисних програм, щоб уникнути їх швидкого виявлення. Ці методи дозволяють змінювати бінарний код шкідливого ПЗ при кожному новому зараженні, роблячи неможливим його виявлення за допомогою заздалегідь відомих, статичних сигнатур (хеш-сум).

Одна із технік, яка запропонована сучасними дослідниками для детектування Malware, полягає у візуалізації бінарного коду виконуваних файлів (PE, ELF тощо), і подальшому його аналізі на основі нейронних мереж. Процес візуалізації зазвичай включає наступні кроки: зчитування байтів, трансформацію у 2D уявлення і формування кінцевого зображення. При цьому бінарний файл читається як одновимірний потік байтів (значення від 0 до 255), потім перетворюється на двовимірну матрицю пікселів. Фінальна матриця зберігається як зображення у відтінках сірого (grayscale), де кожне значення байта відповідає яскравості пікселя.

Ця трансформація виявилася надзвичайно ефективною, оскільки програми, що належать до однієї родини шкідливого ПЗ, часто мають схожі структури коду. Навіть після обфускації, ці структури призводять до схожих візуальних текстур та патернів на зображеннях, які людське око може не розпізнати, але які легко "вловлюються" нейронними мережами.

Після перетворення файлів у зображення задача виявлення шкідливого ПЗ стає задачею класифікації зображень. Тут на перше місце виходять згорткові нейронні мережі, які є золотим стандартом у комп'ютерному зорі.

Архітектура CNN характеризується наявністю згорткових шарів, шарів об'єднання та повністю зв'язаних шарів (рис. 1).

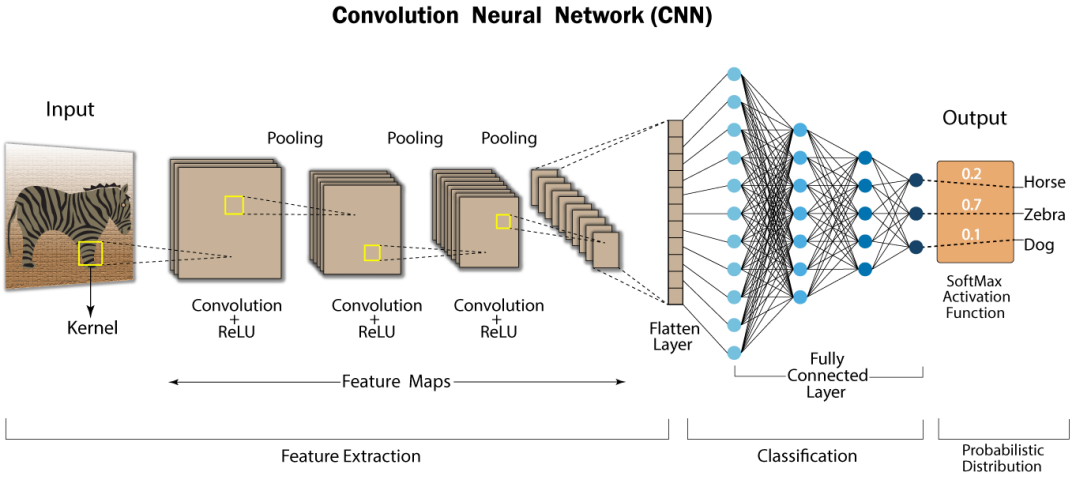

Рис. 1. Базова архітектура CNN

Ключові переваги CNN у цій задачі: автоматичне виділення ознак, ієрархічне навчання, стійкість до зсувів, універсальність. На відміну від традиційного машинного навчання CNN не потребують ручного "конструювання ознак" (feature engineering). Їхні згорткові фільтри автоматично навчаються виявляти релевантні візуальні патерни (текстури), що відповідають шкідливим фрагментам коду. Глибокі шари мережі вчаться розпізнавати все складніші патерни, комбінуючи простіші, виявлені на попередніх шарах. Завдяки механізмам згортки моделі стійкі до того, де саме у файлі (і, відповідно, на зображенні) знаходиться шкідливий патерн. Універсальність полягає в тому, що цей метод не залежить від конкретної архітектури процесора чи операційної системи, оскільки він аналізує лише сирий бінарний потік.

Навчання глибоких CNN (таких як, VGG, ResNet, Inception, XceptionNet) "з нуля" вимагає величезних наборів даних та обчислювальних потужностей. У сфері кібербезпеки домінуючим підходом стало використання трансферного навчання [4].

Підхід полягає у використанні моделей, що вже були навчені на мільйонах загальних зображень (наприклад, з набору даних ImageNet). Виявляється, що базові фільтри, які ці моделі вивчили (детектори кутів, градієнтів, текстур), є універсальними. Вони ефективно "бачать" текстури і в зображеннях шкідливого ПЗ, що дозволяє: 1) використати потужні, перевірені архітектури; 2) значно скоротити час навчання та обсяг даних; 3) досягти дуже високих показників точності (часто понад 98–99 %).

Найпоширенішим втіленням трансферного навчання в контексті глибинного навчання є наступний робочий процес (рис. 2):

- взяття шарів з попередньо навченої моделі;
- їх замороження, щоб уникнути знищення будь-якої інформації, яка в них міститься, під час майбутніх тренувань;
- додавання нових тренувальних шарів поверх заморожених шарів. Вони навчаться перетворювати старі функції на прогнози на новому наборі даних;
- навчання нових шарів на вашому наборі даних.

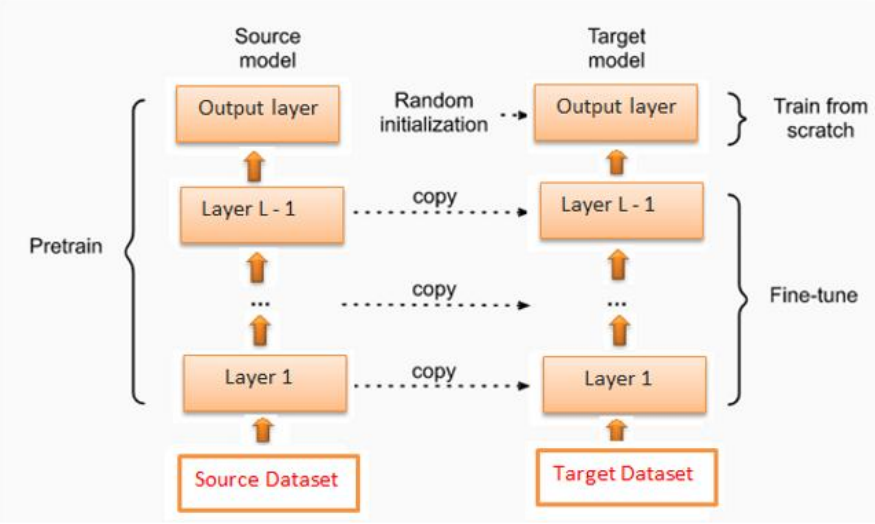

Рис. 2. Ілюстрація процесу Transfer Learning

Останнім необов’язковим кроком є тонке налаштування, яке складається з розморожування всієї моделі, отриманої вище (або її частини), і повторного навчання на нових даних із дуже низькою швидкістю навчання. Як результат – можна досягти значних покращень шляхом поступової адаптації попередньо навчених функцій до нових даних.

Найбільшим викликом для всіх методів глибокого навчання є їхня непрозорість. Модель може з 99 % точністю сказати, що файл є шкідливим, але не може пояснити чому. Це створює попит на методи пояснювального штучного інтелекту (Explainable AI, XAI), такі як

Grad-CAM, які можуть візуалізувати, на які "частини" зображення програмного коду модель звернула увагу при ухваленні рішення [3].

Для оцінки ефективності детектування елементів ШПЗ було обрано та проаналізовано сім відомих архітектур згорткових нейронних мереж: DenseNet169, EfficientNetB0, InceptionV3, MobileNetV2, ResNetV2, VGG16 та XceptionNet. Для кожної архітектури базові згорткові шари були "заморожені" (або частково розморожені), а верхні шари класифікатора були замінені новими, адаптованими до кількості класів у нашому наборі даних.

DenseNet169 [6] – це архітектура згорткової нейронної мережі, яка належить до сімейства щільно зв’язаних згорткових мереж (DenseNets).

Ключовою інновацією DenseNet є щільна схема з’єднання. На відміну від традиційних згорткових нейронних мереж, де кожен шар пов’язаний лише з сусідніми шарами, DenseNet з’єднує кожен шар із кожним іншим у прямому зв’язку. Цей щільний зв’язок досягається шляхом об’єднання карт функцій усіх попередніх шарів як входів до поточного шару. Ця конструкція має кілька переваг, включаючи покращене повторне використання функцій, градієнтний потік і ефективність параметрів.

Мережа VGG16 [7] складається з малих згорткових фільтрів. VGG16 має три повністю зв'язані шари та 13 згорткових шарів. У неї проста, глибока архітектура. Слугує чудовим базовим рівнем (baseline). Вхідними даними для будь-якої мережевої конфігурації вважається зображення фіксованого розміру 224 x 224 із трьома каналами – R, G і B. Єдина виконана попередня обробка – нормалізація значень RGB для кожного пікселя. Це досягається шляхом віднімання середнього значення від кожного пікселя.

ResNetV2 [8] використовує "залишкові з'єднання" (residual connections), що дозволяє навчати надзвичайно глибокі мережі без втрати градієнта. Ідеально для виявлення дуже складних, ледь помітних патернів.

InceptionV3/XceptionNet [9] мають як ключову ідею "багатомасштабний аналіз" (рис. 3). Вони одночасно застосовують фільтри різного розміру (1x1, 3x3, 5x5), що дозволяє їм бачити і дрібні, і великі патерни одночасно. Це може бути критично важливим, оскільки шкідливий код може бути як короткою інструкцією (дрібний патерн), так і цілим вбудованим блоком (великий патерн).

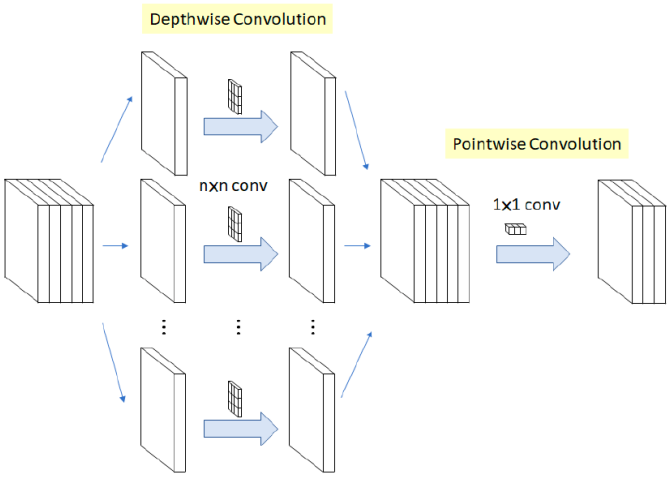

Рис. 3. Архітектура XceptionNet

MobileNetV2/EfficientNetB0 являють собою "ефективні" архітектури. Вони розроблені для досягнення максимальної точності при мінімальних обчислювальних витратах. Їх аналіз важливий для практичного застосування, бо така модель могла б працювати навіть на мобільному пристрої або в антивірусному сканері реального часу без значного навантаження на систему.

Експериментальні дослідження та метрики оцінки

З метою оцінки ефективності виявлення різними CNN ШПЗ через бінарні файли було обрано набір даних зловмисного програмного забезпечення MaleVis [10]. Він містить 14226 зображень зловмисного програмного забезпечення, що охоплює 25 класів, а також 1 клас звичайного програмного забезпечення. З набору даних було використано всі 26 класів і 7280 зображень додатково відібрано з цих класів з метою навчання. Для цілей тестування (5126 зображень) та валідації (1820 зображень) було відібрано 6946 зображень із всіх класів. Зображення в наборі даних MaleVis були отримані шляхом вилучення бінарних зображень із файлів шкідливих програм у 3-канальному форматі RGB. Потім розміри зображень змінювалися до роздільної здатності квадратів 224x224.

Ключовим етапом дослідження, що передував навчанню моделей, було перетворення виконуваних PE-файлів у графічні зображення. Для цього завдання було обрано Python-скрипт Bin2png [11], здатний без втрат конвертувати бінарні дані у формат PNG.

Процес генерації зображень мав два етапи:

1) Конвертація. За допомогою адаптованого скрипту вся колекція бінарних файлів з набору даних була перетворена на PNG-зображення. На цьому етапі встановлювалася фіксована ширина (224 або 300 пікселів), що призводило до генерації зображень з різною висотою, яка залежала від розміру вихідного файлу.

2) Масштабування (стандартизація). Оскільки архітектури згорткових нейронних мереж вимагають на вході зображення фіксованого розміру, усі отримані PNG-файли були стандартизовані. Їх було приведено до квадратних розмірів 224×224 та 300×300 пікселів.

Для мінімізації втрат інформації та артефактів під час масштабування обрано метод інтерполяції Ланцоша. Цей метод, реалізований у бібліотеці OpenCV, аналізує розширений набір пікселів (8×8) і забезпечує високу якість зображення у порівнянні з простішими методами (наприклад, білінійною інтерполяцією).

Для навчання та оцінки моделей було використано пропорції поділу оригінального набору даних Malevis. Загальний набір даних був розділений на дві основні частини: 1) тестувальний набір: 36 % від усіх даних; 2) набір для навчання та валідації: 64 % від усіх даних. Особливістю даної роботи було те, що оригінальний 64 %-й набір був додатково розділений для створення окремого валідаційного набору. Цей набір є критично необхідним для моніторингу перенавчання моделі в процесі тренування.

Поділ 64 %-го набору проводився у пропорції 4:1 (тобто, 80 % даних для навчання та 20 % – для валідації). Таким чином, фінальний розподіл усього набору даних виглядає так: 1) тренувальний набір: 51.2 % (80 % від 64 %); 2) валідаційний набір: 12.8 % (20 % від 64 %); 3) тестувальний набір: 36 % (збережено з оригінального поділу).

Такий підхід дозволив одночасно виділити дані для валідації, не порушуючи при цьому цілісність оригінального тестового набору, що забезпечує об'єктивність та порівнянність результатів.

В табл. 1 представлено дані щодо класів зловмисного ПЗ і їх кількості (в тому числі відношення кількості тренувальних і тестувальних даних). На рис. 4 наведено приклади зображень представників різних видів зловмисного ПЗ.

В якості додаткових інструментів для роботи з датасетом, навчання моделей та оцінки метрик ефективності було обрано TensorFlow, Matplotlib та Seaborn. TensorFlow надає повний набір інструментів, бібліотек і ресурсів спільноти, які полегшують дослідникам і розробникам впровадження та розгортання моделей машинного навчання. Matplotlib та Seaborn будуть використані для візуалізації даних та статистичної обробки.

Після підготовки набору даних (конвертації файлів у зображення та їх стандартизації) ми розробили єдиний алгоритм для навчання кожної з обраних нейронних мереж. Цей процес, проілюстрований на блок-схемі (рис. 5), використовує поширений та ефективний підхід трансферного навчання (transfer learning).

Таблиця 1

Номер	Назва	Клас	Тренувальні/Тестувальні/ Валідаційні дані
1	Adposhel	Adware	280/144/70
2	Agent-fyi	Trojan	280/120/70
3	Akorfn	Ransomware	280/146/70
4	Allaple	Worm	280/128/70
5	Amonetize	Adware	280/147/70
6	Androm	Backdoor	280/150/70
7	AutoRun-PU	Worm	280/146/70
8	BrowseFox	Adware	280/143/70
9	CVEap	Ransomware	280/149/70
10	Dinwod!rfn	Trojan	280/149/70
11	DoejoCrypt	Ransomware	280/135/70
12	Elex	Trojan	280/150/70
13	Expiro-H	Virus	280/151/70
14	Fasong	Worm	280/150/70
15	Hlux!IK	Worm	280/150/70
16	Injector	Trojan	280/145/70
17	InstallCore.C	Adware	280/150/70
18	MultiPlug	Adware	280/149/70
19	Neoreklami	Adware	280/150/70
20	Neshta	Virus	280/147/70
21	Goodware	Not a Malware	280/1482/70
22	Sality	Virus	280/149/70
23	Snarasite.D!tr	Trojan	280/150/70
24	Stantinko	Backdoor	280/150/70
25	VBKrypt	Trojan	280/150/70
26	Vilsel	Trojan	280/146/70

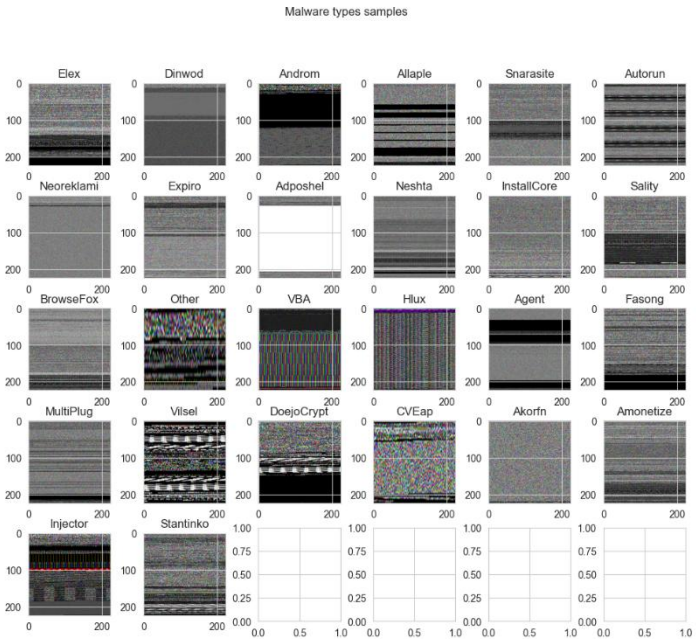

Рис. 4. Приклад графічних представлень кожного з класів ШПЗ в датасеті

Ми не будували нейронну мережу "з нуля", а брали потужні, вже навчені моделі (як VGG16, ResNet та ін.) і "перекваліфіковували" їх для нашого конкретного завдання розпізнавання зображень шкідливого ПЗ.

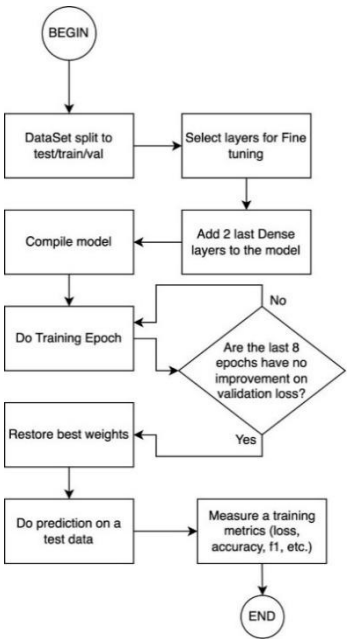

Рис. 5. Блок-схема алгоритму тренування нейронної мережі

Основні кроки проведення експерименту [12].

К р о к 1. Підготовка даних. Весь наш набір зображень був розділений на дві ключові групи:

- набір для навчання та валідації (64 %). Ці дані використовувалися для самого процесу "навчання". Модель використовувала їх, щоб знайти закономірності (тренування), а ми періодично перевіряли її успішність на частині цих даних (валідація);

- тестовий набір (36 %). Ці дані модель ніколи не бачила під час навчання. Вони були "відкладені" для фінального, об'єктивного тестування.

К р о к 2. Адаптація архітектури (трансферне навчання). Для кожної з моделей ми виконали "тонке налаштування":

1) "Заморозка" шарів. Ми припустили, що перші шари нейронної мережі (наприклад, перші 100) вже чудово навчені розпізнавати базові речі (лінії, кути, прості текстури) завдяки їх навчанню на мільйонах інших зображень. Ми "заморозили" їхні ваги, заборонивши їм змінюватися.

2) "Розморозка" шарів. Ми дозволили до-навчання лише для більш глибоких шарів (усіх, що йдуть після 100-го). Це дало змогу мережі адаптувати свої знання про складні патерни до специфіки наших зображень ШПЗ.

3) Встановлення нового "класифікатора". Ми "відрізали" оригінальний вихідний шар моделі (який міг розпізнавати, наприклад, 1000 класів типу "кіт", "собака", "автомобіль") і додали замість нього два нових щільно зв'язаних (Dense) шари:

- прихований шар (1000 нейронів) як проміжний, що допомагає моделі агрегувати вивчені ознаки;

- вихідний шар (26 нейронів) як кінцевий шар, де кожен нейрон відповідає одному з 26 класів шкідливого ПЗ у нашому датасеті.

К р о к 3. Процес тренування та компіляції.

Після адаптації модель була скомпільована (підготовлена до навчання) та запущена. Ключовою особливістю був механізм "ранньої зупинки" (Early Stopping), який допоміг нам заощадити час та уникнути перенавчання (ситуації, коли модель просто "зазубрює" тренувальні дані, але погано працює на нових).

Ось як він працював:

- модель тренувалася циклами (епохами);

- встановили "терпіння" (patience) у 8 епох;

- після кожної епохи модель перевіряла свою ефективність (значення функції втрат) на валідаційних даних;

- якщо протягом 8 епох поспіль модель не показувала покращення (не встановлювала новий "рекорд" ефективності), навчання автоматично зупинялося;

- після зупинки система автоматично відновлювала ваги моделі з тієї епохи, яка була найкращою.

Встановлено максимальну межу у 80 епох, але завдяки цьому розумному механізму жодній з наших моделей не знадобилося більше 25 епох, щоб досягти свого піку ефективності.

К р о к 4. Фінальна оцінка.

Після того, як навчання було завершено і ми отримали найкращу версію кожної моделі, настав час тестування. Ми взяли тестовий набір (36 % даних, які модель ніколи не бачила) і попросили моделі зробити на ньому прогнози.

Далі порівняли ці прогнози з реальними, правильними відповідями і розрахували матрицю невідповідностей (візуальну таблицю, що показує, які класи модель плутала між собою) і ключові метрики Precision, Recall та F1-Score, щоб об'єктивно оцінити точність кожної моделі.

Після навчання всіх 7 нейронних мереж ми оцінили їхню ефективність. Для цього ми використали стандартні метрики: Accuracy (загальна точність), Precision (точність прогнозів), Recall (повнота виявлення) та F1-Score (збалансований показник).

Щоб отримати узагальнені бали (табл. 2) ми використали «зважений» підхід. Це означає, що точність у класі, де було 1000 тестових файлів, мала більшу «вагу» для загальної оцінки, ніж у класі, де було лише 100 файлів. Це робить оцінку більш справедливою та об’єктивною, особливо коли класи нерівномірні.

Таблиця 2

Нейронна мережа	**Precision**	**Recall**	**F1**	**Accuracy**
XceptionNet	0.89	0.86	0.86	0.86
EfficientNetB0	0.90	0.85	0.86	0.85
MobileNetV2	0.90	0.85	0.86	0.85
DenseNet169	0.90	0.86	0.86	0.86
InceptionV3	0.89	0.83	0.85	0.83
VGG16	0.88	0.82	0.82	0.82
ResNet50V2	0.88	0.83	0.83	0.83

Як видно з табл. 2, усі моделі показали досить високі та схожі результати. По показнику Precision найкращий бал (0.90) у EfficientNetB0, MobileNetV2 та DenseNet169. Це означає, що якщо ці моделі казали "Це вірус!", вони мали рацію у 90 % випадків. По показниках Recall та Accuracy найкращий бал (0.86) у XceptionNet та DenseNet169. Це означає, що вони змогли правильно детектувати та класифікувати 86 % усіх файлів у тестовій вибірці.

За суто чистими балами DenseNet169 та XceptionNet виглядають лідерами. Але висока точність не основний показник. Нам також важлива ефективність моделі. DenseNet169 "легша" бо має 14 мільйонів параметрів (умовних "налаштувань"), тоді як XceptionNet має 27 мільйонів. Але під час нашого експерименту ми "до-навчали" 69 шарів у DenseNet169, і лише 32 – у XceptionNet. Через це один цикл (епоха) навчання у DenseNet169 тривав у 1.5 рази довше. Найважливіше, що XceptionNet досягла свого найкращого результату всього за 7 епох, тоді як DenseNet169 знадобилося 13 епох.

В и с н о в о к : хоча обидві моделі точні, XceptionNet виявилася більш перспективною. Вона вчиться набагато швидше і вимагає менше циклів навчання для досягнення пікової ефективності. Ми детально проаналізували, де саме помилялася наша найкраща модель XceptionNet.

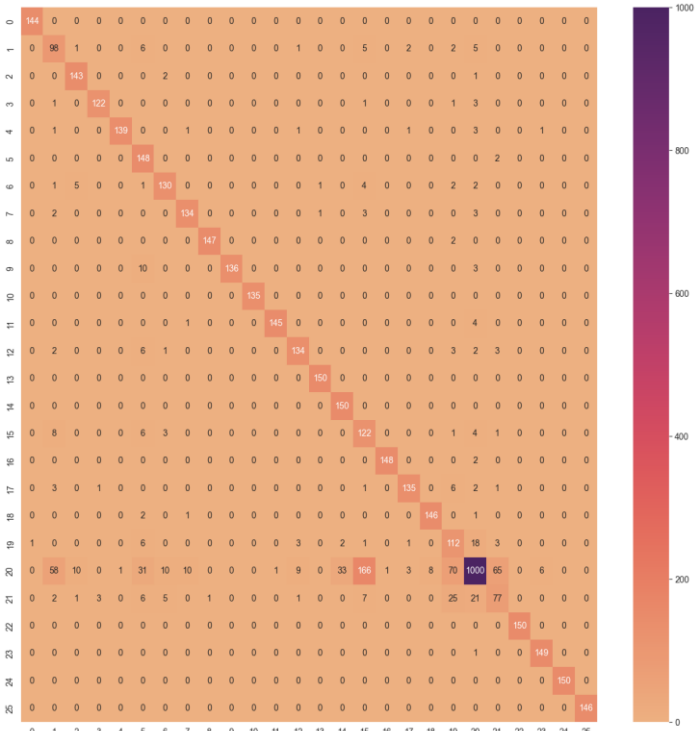

Рис. 6. Матриця невідповідностей для XceptionNet

Матриця невідповідностей (рис. 6) показала кілька цікавих речей. Найбільше помилок модель робила з класом "Goodware" (безпечні файли). Вона часто плутала їх з вірусами. Це очікувано, оскільки у нас було в 10 разів більше безпечних файлів, ніж будь-якого іншого класу шкідливого ПЗ. Найпроблемніший вірус "Injector" (Клас 16). Модель найчастіше плутала цей тип трояна з безпечними файлами. Це може свідчити про те, що його візуальні патерни дуже "хаотичні" або схожі на звичайні програми. Деякі класи, наприклад, "Snarasite.D!tr", "DoejoCrypt" та "Vilsel", модель розпізнавала зі 100 % точністю. Їхні візуальні патерни виявилися дуже унікальними та легко впізнаваними.

Висновки

Традиційні антивіруси здебільшого покладаються на сигнатурний аналіз. Вони шукають у файлах унікальні "цифрові відбитки" (сигнатури) вже відомих вірусів. Цей підхід став на практиці мало ефективним через масове використання зловмисниками технік поліморфізму та обфускації. Після перетворення бінарних файлів у зображення задача виявлення шкідливого ПЗ може бути перекваліфікована в задачу класифікації зображень.

У ході дослідження успішно вирішено актуальну задачу класифікації шкідливого програмного забезпечення за допомогою методів глибокого навчання, застосованих до візуальних представлень виконуваних PE-файлів.

Було сформовано збалансований набір даних, що включав різні сімейства шкідливого ПЗ та "чисті" файли (Goodware) і проведено порівняльний експериментальний аналіз семи поширених архітектур згорткових нейронних мереж, які навчалися за методологією трансферного навчання. Ключовим етапом роботи стала конвертація цих файлів у зображення у відтінках сірого за допомогою адаптованого скрипту Bin2png.

Експерименти засвідчили, що класифікація шкідливого ПЗ за його графічним образом є високоточним підходом, який не поступається іншим сучасним методам статичного аналізу. Моделі XceptionNet та DenseNet169 продемонстрували найкращі узагальнені показники, досягнувши ідентичної фінальної точності (Accuracy) у 86 %. Попри однакову точність глибший аналіз показав, що XceptionNet є більш перспективною. Вона досягла свого пікового результату всього за 7 епох навчання, тоді як DenseNet169 знадобилося 13. Це свідчить про

вищу ефективність та швидкість адаптації архітектури XceptionNet до даної специфічної задачі. Практичне значення роботи полягає в тому, що розроблений підхід може бути інтегрований у сучасні антивірусні програмні комплекси. Він здатен слугувати потужним модулем статичного аналізу для виявлення нових та поліморфних загроз, які ефективно обходять традиційні сигнатурні сканери.

Список літератури:

1. Brosolo M. et al. Security through the Eyes of AI: How Visualization is Shaping Malware Detection. (2025). arXiv:2505.07574v4. Available: https://doi.org/10.48550/arXiv.2505.07574.
2. Shahnawaz M. et al. Dynamic Malware Classification of Windows PE Files using CNNs and Greyscale Images Derived from Runtime API Call Argument Conversion. arXiv:2505.24231v1. (2025). Available: https://doi.org/10.48550/arXiv.2505.24231.
3. Bensaoud A. et al. A Survey of Malware Detection Using Deep Learning. (2024). arXiv:2407.19153v1. Available: https://doi.org/10.48550/arXiv.2407.19153.
4. Ashawa M., Owoh N., Hosseinzadeh S., & Osamor J. Enhanced Image-Based Malware Classification Using Transformer-Based Convolutional Neural Networks (CNNs) // Electronics. 2024. Vol. 13(20). P. 4081. Available: https://doi.org/10.3390/electronics13204081.
5. Kim H. & Kim M. Malware Detection and Classification System Based on CNN-BiLSTM // Electronics. 2024. Vol. 13. P. 2539. Available: https://doi.org/10.3390/electronics13132539.
6. Huang G., Liu Z., Van Der Maaten L. and Weinberger K. Q. Densely Connected Convolutional Networks // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017. P. 2261–2269. Available: doi: 10.1109/CVPR.2017.243.
7. Simonyan K., & Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // Proceedings of the 3rd International Conference on Learning Representations (ICLR). (2015). Available: https://doi.org/10.48550/arXiv.1409.1556.
8. He K., Zhang X., Ren S., & Sun J. Identity Mappings in Deep Residual Networks // Proceedings of the European Conference on Computer Vision (ECCV). (2016). Available: https://doi.org/10.48550/arXiv.1603.05027.
9. Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA. 2017. P. 1800–1807. [Online]. Available: doi: 10.1109/CVPR.2017.195.
10. MaleVis Dataset Home Page. Режим доступу: https://web.cs.hacettepe.edu.tr/~selman/malevis/.
11. GitHub – ESultanik/bin2png: A simple cross-platform script for encoding any binary file into a lossless PNG. GitHub. Режим доступу: https://github.com/ESultanik/bin2png.
12. Федюшин О. І., Хижняк К. М. CNN та їх використання для класифікації MALWARE // Матеріали Одинадцятої міжнар. наук.-техн. конф. «Проблеми інформатизації», Харків, 16–17 листопада 2023 р. С. 49.

Надійшла до редколегії 04.10.2025

Відомості про авторів:

Федюшин Олександр Іванович – канд. техн. наук, доцент, Харківський національний університет радіоелектроніки, доцент кафедри безпеки інформаційних технологій, факультет комп'ютерної інженерії та управління, Україна; e-mail: oleksandr.fediushyn@nure.ua; ORCID: http://orcid.org/0000-0002-3600-405X

Шулік Павло Вікторович – канд. техн. наук, Харківський національний університет радіоелектроніки, ст. викладач кафедри безпеки інформаційних технологій, факультет комп'ютерної інженерії та управління, Україна; e-mail: pavlo.shulik@nure.ua; ORCID: https://orcid.org/0009-0004-6200-2172

Просолов Владислав Валерійович – Харківський національний університет радіоелектроніки, аспірант кафедри безпеки інформаційних технологій; Україна; e–mail: vladyslav.prosolov@nure.ua; ORCID: https://orcid.org/0009-0002-1276-4828

В'юхін Даніїл Олександрович – Харківський національний університет радіоелектроніки, ст. викладач кафедри безпеки інформаційних технологій, факультет комп'ютерної інженерії та управління, Україна; e-mail: daniil.viukhin@nure.ua; ORCID: https://orcid.org/0009-0009-8442-9587

Чечуй Олександр Вікторович – канд. техн. наук, доцент, Харківський національний університет Повітряних Сил імені Івана Кожедуба, доцент кафедри радіоелектронних систем пунктів управління Повітряних Сил, Україна; e-mail: alche1972@ukr.net; ORCID: https://orcid.org/0000-0002-7584-4457