НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

Погорелов Володимир Володимирович

УДК 004.056.5:004.8

ДИСЕРТАЦІЯ
НЕЙРОМЕРЕЖЕВІ МОДЕЛІ ТА МЕТОДИ РОЗПІЗНАВАННЯ КОМП'ЮТЕРНИХ ВІРУСІВ

Спеціальність 05.13.21 – «Системи захисту інформації»

Подається на здобуття наукового ступеня кандидата технічних наук

Дисертація містить результати власних проваджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.


Погорелов В.В.

Науковий керівник:
доктор технічних наук, професор
Терейковський Ігор Анатолійович,
професор кафедри безпеки інформаційних технологій Національного авіаційного університету

Київ – 2020

АНОТАЦІЯ

Погорелов В.В. Нейромережеві моделі та методи розпізнавання комп'ютерних вірусів. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук (доктора філософії) за спеціальністю 05.13.21 «Системи захисту інформації». – Національний авіаційний університет, Київ, 2020.

У роботі вирішено актуальну науково-прикладну задачу підвищення ефективності протидії комп'ютерним вірусам, за рахунок розробки і дослідження нових нейромережевих моделей, методів і засобів розпізнавання комп'ютерних вірусів, здатних оперативно пристосовуватись до умов використання і реагувати на виникнення нових видів вірусів. У дисертаційній роботі проведено аналіз сучасних нейромережевих моделей та методів розпізнавання комп'ютерних вірусів, що показав наявність низки недоліків, пов'язаних з високою потребою в обчислювальних ресурсах, низькою адаптованістю до проведення аналізу обфускованого програмного коду та недостатньою ефективністю розпізнавання. В результаті виконаного аналізу обґрунтована перспективність застосування в контурі розпізнавання систем антивірусного захисту нейромережевих засобів. При цьому показано можливість застосування нейромережевих моделей як в поведінкових аналізаторів, так і при використанні сигнатурного аналізу. Вперше розроблено концептуальну модель оцінювання глибоких нейронних мереж, яка за рахунок взаємопов'язаних принципів допустимості використання, визначення множини ефективних видів та оцінювання ефективності виду глибокої нейронної мережі дозволяє визначити множину сучасних нейромережевих моделей для побудови ефективних антивірусних засобів. Вперше розроблено модель формування параметрів навчальних прикладів глибокої нейронної мережі, яка за рахунок формального представлення закодованих значень викликів API-функцій, байт-послідовності N-грамів, опкодів, основних регістрів процесора, а також результатів статичного аналізу зразків шкідливих та безпечних програм, двомірної інтерпретації бінарного коду програми і

параметрів графу залежностей значень та станів дозволяє будувати засоби нейромережевого аналізу обфускованого програмного коду. Вперше розроблено метод визначення архітектурних параметрів глибокої нейронної мережі, призначеної для розпізнавання вірусів, який за рахунок використання запропонованої концептуальної моделі оцінювання глибоких нейронних мереж та моделі формування параметрів навчальних прикладів, що використовуються для реалізації етапів визначення основних умов застосування, доцільності використання нейромережевої моделі та найбільш ефективної архітектури, а також формування параметрів навчальних прикладів та визначення параметрів архітектури найбільш ефективного виду глибокої нейронної мережі, дозволяє сформувати набір величин, які забезпечують пристосованість такої мережі до визначених умов застосування. Отримав подальший розвиток метод нейромережевого розпізнавання комп'ютерних вірусів, який, за рахунок визначення умов створення та застосування нейромережевих засобів, процесів формування портретів вірусів та безпечних програм, а також визначення архітектурних параметрів глибокої нейронної мережі та верифікації і оцінки ефективності нейромережевих засобів, забезпечує достатню похибку розпізнавання при різних умовах застосування з урахуванням обмежень щодо створення навчальної вибірки та обмежень щодо обчислювальних ресурсів системи антивірусного захисту. Практичне значення одержаних результатів дисертаційного дослідження полягає у наступному: розроблене алгоритмічне та програмне забезпечення, що базується на створених нейромережевих методах та моделях, дозволило забезпечити достатню точність розпізнавання комп'ютерних вірусів та приблизно в 1,5 рази зменшити обчислювальні витрати, пов'язані з визначенням значень архітектурних параметрів ГНМ; розроблені програми, що реалізують запропоновані моделі та методи. Результати проведених розрахунків вказують на те, що ефективність розробленого НМЗ приблизно в 1,14 рази вища ніж у подібних відомих засобів. Таким чином, результати досліджень підтверджують можливість підвищення ефективності

розпізнавання комп’ютерних вірусів за рахунок застосування розроблених нейромережевих моделей та нейромережевих засобів. На базі запропонованого методу нейромережевого розпізнавання комп’ютерних вірусів розроблено алгоритмічне забезпечення та відповідна програмна модель системи, що дозволила з достатньою точністю розпізнавати комп’ютерні віруси та забезпечити оперативність створення алгоритмів функціонування апаратно-програмних засобів захисту інформації. Експериментальне дослідження програмної моделі системи, а також впровадження та успішне практичне використання відповідних розробок підтвердило достовірність теоретичних положень та гіпотез, практичних розробок і висновків дисертаційної роботи.

Ключові слова: захист інформації, нейронна мережа, комп’ютерний вірус, антивірусний захист, шкідливе програмне забезпечення.

ABSTRACT

Pogorelov V. Neural models and methods for computer viruses recognition. – Qualifying research paper made as manuscript.

Thesis for scientific degree of candidate of technical sciences, specialty 05.13.21 – Information Security Systems, National Aviation University, Kyiv, 2020.

The current scientific and practical task of an anti-virus detection efficiency improvement by developing and researching new neural network models, methods and tools for recognizing computer viruses, able to quickly adapt to conditions of use and respond to new types of viruses was developed in terms of this Thesis. The Thesis describes the review of modern neural network models and computer viruses detection methods, which has demonstrated a number of shortcomings associated with high demand for computing performance, low adaptability to the analysis of obfuscated software code and insufficient detection efficiency. Based on the performed review the perspective of application in the environment of detection of anti-virus protection systems of neural network means are substantiated. At the same time, the possibility of using neural network models both in behavioral analyzers

and when using signature analysis is shown. For the first time, a conceptual deep neural networks evaluation has been developed, which allows to identify a lot of modern neural network models in order to build an effective anti-virus tools due to the interrelated principles of acceptability, determining the set of effective types and evaluating the efficiency of deep neural network. It was a first time when a model of formation of training samples parameters of deep neural network was developed which due to the formal representation of encoded values of API-functions call requests, byte-series of N-grams, operation codes, processor main registers, as well as the results of static analysis of templates of malicious and safety programs, 2D interpretation of binary code and parameters of the value state dependence graph allows to build the tools of neural network analysis of obfuscated program code. It was a first time when a method for determining the architectural parameters of a deep neural network designed for virus detection was developed. Due to the use of the proposed conceptual deep neural networks evaluation and the model of formation of training templates parameters used to implement the basic application conditions determination stages, the feasibility of using a neural network model and the most efficient architecture as well as the formation of the training templates parameters and determination of the parameters of the most effective deep neural network architecture, allows to form a set of values that ensure the adaptability of such a network to certain conditions of use. The method of neural network detection of computer viruses has been work out, which, by determining the neural networks creation and usage conditions, processes of forming software virus images and safety programs, as well as by determining the deep neural network architectural parameters and verification and evaluation of neural networks, provides sufficient detection error under different conditions of use subject to restrictions on the training sample creation and restrictions on the computing performance of the anti-virus protection system.

The practical outcome of the thesis research is as follows: the developed algorithm and software, based on the created neural network methods and models, allowed to provide sufficient accuracy of computer virus detection and to reduce

approximately 1.5 times the computational costs associated with determining values of the deep neural network architectural parameters; programs implementing the proposed models and methods have been developed. The calculations show that the efficiency of the developed neural network tool is approximately 1.14 times higher versus the known tools. Thus, the research confirms the possibility of increasing the efficiency of detection of computer viruses through the use of developed neural network models and tools. Based on the proposed method of neural network detection of computer virus, algorithms and the corresponding program model of the system were developed, which allowed to detect computer viruses with sufficient accuracy and to ensure the creation of algorithms for hardware and software information protection means. Experimental research of the software model of the system, as well as the implementation and successful practical use of relevant inventions confirmed the validity of theoretical thesis and hypotheses, practical inventions and conclusions of this Thesis.

Keywords: information protection, neural network, computer virus, anti-virus protection, malicious software.

Список публікацій здобувача:

1. I. Dychka, D. Chernyshev, I. Tereikovskyi, L. Tereikovska , V. Pogorelov, «Malware Detection Using Artificial Neural Networks», *Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing*, Vol. 938. Springer, Cham, pp.3-12, 2019. (SCOPUS) DOI: *https://doi.org/10.1007/978-3-030-16621-2_1*.

2. I. Dychka, I. Tereikovskyi, L. Tereikovska, V. Pogorelov, S. Mussiraliyeva, «Deobfuscation of Computer Virus Malware Code with Value State Dependence Graph», *Advances in Computer Science for Engineering and Education. ICCSEEA 2018. Advances in Intelligent Systems and Computing*, Vol. 754. Springer, Cham, pp.370-379, 2018. (SCOPUS) DOI: *https://doi.org/10.1007/978-3-319-91008-6_37*.

3. Hu. Zhengbing, I. Tereykovskiy, L. Tereykovska, V. Pogorelov,

«Determination of Structural Parameters of Multilayer Perceptron Designed to Estimate Parameters of Technical Systems», *International Journal of Intelligent Systems and Applications(IJISA)*, Vol. 9, No.10, pp. 57-62*)*, 2017. (SCOPUS)
DOI: *10.5815/ijisa.2017.10.07, ISSN 2074-9058*.

4. І. Терейковський, О. Заріцький, Л. Терейковська, В. Погорелов, «Метод розробки архітектури глибокої нейронної мережі, призначеної для розпізнавання комп'ютерних вірусів», *Захист інформації*, Т. 20, № 3, С. 188-199, 2018.

5. Л. Терейковська, Є. Іванченко, В. Погорелов «Метод адаптації глибокої нейронної мережі до розпізнавання комп'ютерних вірусів», *Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво»*, Луцьк, Випуск № 35, С. 198-205, 2019.

6. В. Погорелов, «Дослідження нейромережевих засобів розпізнавання кібератак на мережеві ресурси інформаційних систем», *Системні технлогії,* №5, С. 61–69, 2017.

7. В. Погорелов, «Проблематика використання нейромережевих систем розпізнавання кібератак», *Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво»*, №27, С. 67–74, 2017.

8. O. Tereykovskiy, V. Pogorelov «Cyberattack recognition with radial basis function neural network», *Projekt interdyscyplinarny projektem XXI wieku*, Tom 2, pp. 255-262, 2017. (Колективна монографія)

9. Б. Айтчанов, И. Бапиев, А. Корченко, В. Погорелов, Л. Терейковская, «Концептуальная модель обеспечения эффективности нейросетевого распознавания кибератак», *Труды международной научно-практической конференции «Математические методы и информационные технологии макроэкономического анализа и экономической политики», посвященной празднованию 80-летнего юбилея академика НАН РК Абдыкаппара Ашимовича Ашимова*, *11-12*, С. 321–325, 2017.

10. B. Aitchanov, I. Bapiev, I. Terejkowski, L. Terejkowska, V. Pogorelov, «Calculation of expected output signal of neural network model for detecting of

cyber-attack on network resources», *Information Technologies, Management and Society, The 15th International Scientific Conference Information Technologies and Management,* pp. 59–62, 2017.

11. V. Pogorelov, M. Karpinski, E. Ivanchenko, «Method of neural networks utilization for malware recognition», *The 10 th International Scientific Conference «ITSec» March*, pp. 58, 2020.

12. В. Погорєлов, «Використання графу залежностей значень і станів у задачі розпізнавання поліморфних комп'ютерних вірусів», *Стан та удосконалення безпеки інформаційно-телекомунікаційних систем (SITS' 2020)*, Миколаїв, 2020, С. 34-35.

13. В. Погорелов, «Нейромережевий метод розпізнавання комп'ютерних вірусів», *VI Міжнародна науково-практична конференція «Актуальні питання забезпечення кібербезпеки та захисту інформації»*, 2020, С. 88-93.

14. I. Tereikovskyi, V. Pogorelov, O. Tereikovskyi, «Determination of structural parameters of a multilayer cyber threat detection perceptron», *Aviation in the XXI-st Century,* 2018, pp. 3.3.1 – 3.3.4.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

API – application programming interface (прикладний програмний інтерфейс)

CPU – central processing unit (центральний процесор)

БШП – багатошаровий персептрон

ВШ – вхідний шар

ГНМ – глибока нейронна мережа

ГС – голосовий сигнал

ДШП – двохшаровий персептрон

ЗНМ – згорткова нейронна мережа

ІС – інформаційна система

МЗР – мережа зустрічного розповсюдження

НМ – нейронна мережа

НМЗ – нейромережевий засіб

НММ – нейромережева модель

НМС – нейромережева система

ОС – операційна система

ПЗ – програмне забезпечення

САЗ – системи антивірусного захисту

ВСТУП

Актуальність теми. В теперішній час системи антивірусного захисту (САЗ) є одним з основних засобів захисту інформації більшості комп'ютерних систем і мереж. Не зважаючи на те, що такі системи використовуються вже не одне десятиліття і їх розробкою та створенням методологічної бази займаються висококваліфіковані фахівці, практичний досвід і результати багатьох науково-практичних досліджень вказують на наявність в сучасних антивірусах розпізнавання суттєвих недоліків. Основним з яких є недостатня точність розпізнавання всієї номенклатури комп’ютерних вірусів, що підтверджується відомими випадками успішних вірусних кібератак на вітчизняні та закордонні комп'ютерні системи і мережі. Однак впровадження відомих засобів розпізнавання комп’ютерних вірусів в вітчизняні системи захисту інформації викликає необхідність їх складної адаптації до очікуваних умов використання. Також недоліками відомих засобів розпізнавання є висока вартість і відсутність докладної науково-технічної документації.

Важливим напрямком підвищення точності розпізнавання є «інтелектуалізація» методів розпізнавання за рахунок використання теорії штучних нейронних мереж (НМ). Перспективність вказаного напрямку підтверджується окремими вдалими застосуваннями НМ в засобах розпізнавання комп’ютерних вірусів (антивірус з відкритим програмним кодом ClamAV, стартап Deep Instinct) та великою кількістю відповідних теоретико-практичних робіт.

Разом з тим, недостатня точність розпізнавання та недостатня адаптованість до умов експлуатації, закритість використаних рішень, значно обмежують сферу їх застосування. При цьому постійний прогрес в області теорії НМ вказує на можливість значного вдосконалення апробованих засобів розпізнавання.

В такій постановці проблеми є актуальною науково-прикладна задача розробки ефективних нейромережевих моделей та методів розпізнавання комп’ютерних вірусів, адаптованих до умов вітчизняних систем антивірусного захисту (САЗ).

Дослідження вітчизняних та зарубіжних вчених, зокрема И. Бенджио (Yoshua Bengio), Бодянського Є. В., Я. Лекуна (Yann LeCun), Різника О.М., Руденка О.Г., Д. Хінтона (Geoffrey Hinton), З. Хохрайтера (Sepp Hochreiter) вказують на те, що перспективним шляхом підвищення ефекивності антивірусного захисту є застосування апарату НМ для розпізнавання комп'ютерних вірусів. Це пояснюється тим, що задача розпізнавання комп'ютерних вірусів є однією із основних при розробці САЗ, що підтверджується ефективністю використання НМ для вирішення подібних задач оцінки параметрів безпеки інформаційних систем у відомих засобах захисту інформації (AVZ, продукція компаній Cisco, Symantec) та є доведеною адаптивністю нейромережевих засобів (НМЗ) розпізнавання до умов застосування в САЗ.

Методологічну і теоретичну основу для ефективного впровадження НМЗ розпізнавання комп'ютерних вірусів складають теоретичні розробки та досвід створення систем захисту інформації таких науковців як, Б. Ахметова, Є. Бодянського, Д. Деннінга, О. Додонова, В. Лахна, А. Лукацького, О. Корченка, О. Петрова, О. Резніка, О. Руденка, С. Форестера, І. Терейковського, В. Харченка, В. Хорошка.

Таким чином, задача розробки ефективних нейромережевих моделей та методів розпізнавання комп'ютерних вірусів зумовила актуальність наукових досліджень і розробок, яким присвячена дисертаційна робота.

Зв'язок роботи з науковими програмами, планами, темами. Тематика дисертаційної роботи та одержані результати безпосередньо пов'язані з «Основними науковими напрямами та найважливішими проблемами фундаментальних досліджень у галузі природничих, технічних і гуманітарних наук НАН України на 2019-2023 роки», зі Стратегією кібербезпеки України від 15 березня 2016 року №96/2016. Результати роботи відображені у звітах держбюджетних науково-дослідних робіт Національного авіаційного університету «Квантово-криптографічні методи захисту критичної інформаційної структури держави» (д.р. № 0117U006770), «Системи мультирівневого розмежування доступу до інформаційних

ресурсів» (№19/14.01.05) та «Дослідження ризиків інформаційної безпеки об’єктів критичної інфраструктури ГТС України та розробка методології поводження з ними» (д.р. № 0118U002371).

Мета і задачі дослідження. Метою дисертаційної роботи є підвищення ефективності протидії комп’ютерним вірусам за рахунок розробки і дослідження нових нейромережевих моделей, методів і засобів розпізнавання комп’ютерних вірусів, здатних оперативно пристосовуватись до умов використання і реагувати на виникнення нових видів вірусів.

Відповідно до поставленої мети визначено такі основні завдання дослідження:

– проведення аналізу можливостей нейромережевих засобів (НМЗ) розпізнавання комп’ютерних вірусів;

– розробка моделі формування параметрів та концептуальної моделі оцінювання глибоких нейронних мереж (ГНМ);

– розробка методу визначення архітектурних параметрів глибокої нейронної мережі та розвиток методу розпізнавання комп’ютерних вірусів;

– проведення експериментальних досліджень, спрямованих на верифікацію запропонованих рішень.

Об'єктом дослідження є процеси розпізнавання комп’ютерних вірусів.

Предметом дослідження є нейромережеві моделі та методи розпізнавання комп’ютерних вірусів.

Методи дослідження. Використано методи теорії захисту інформації, НМ, комп’ютерного моделювання, експертного і статистичного аналізу та оптимізації.

Наукова новизна отриманих результатів. Проведені у дисертаційній роботі дослідження дозволили розробити й науково обґрунтувати принципи, моделі та методи нейромережевого розпізнавання комп’ютерних вірусів. Зокрема отримані такі наукові результати.

Вперше розроблено:

– концептуальну модель оцінювання глибоких нейронних мереж, яка за рахунок взаємопов’язаних принципів допустимості використання, визначення

множини ефективних видів та оцінювання ефективності виду глибокої нейронної мережі дозволяє визначити множину сучасних нейромережевих моделей для побудови ефективних антивірусних засобів;

– модель формування параметрів навчальних прикладів глибокої нейронної мережі, яка за рахунок формального представлення закодованих значень викликів API-функцій, байт-послідовності N-грамів, опкодів, основних регістрів процесора, а також результатів статичного аналізу зразків шкідливих та безпечних програм, двомірної інтерпретації бінарного коду програми і параметрів графу залежностей значень та станів дозволяє будувати засоби нейромережевого аналізу обфускованого програмного коду;

– метод визначення архітектурних параметрів глибокої нейронної мережі, призначеної для розпізнавання вірусів, який за рахунок використання запропонованої концептуальної моделі оцінювання глибоких нейронних мереж та моделі формування параметрів навчальних прикладів, що використовуються для реалізації етапів визначення основних умов застосування, доцільності використання нейромережевої моделі та найбільш ефективної архітектури, а також формування параметрів навчальних прикладів та визначення параметрів архітектури найбільш ефективного виду глибокої нейронної мережі, дозволяє сформувати набір величин, які забезпечують пристосованість такої мережі до визначених умов застосування;

Отримав подальший розвиток:

– метод нейромережевого розпізнавання комп’ютерних вірусів, який, за рахунок визначення умов створення та застосування нейромережевих засобів, процесів формування портретів вірусів та безпечних програм, а також визначення архітектурних параметрів глибокої нейронної мережі та верифікації і оцінки ефективності нейромережевих засобів, забезпечує достатню похибку розпізнавання при різних умовах застосування з урахуванням обмежень щодо створення навчальної вибірки та обмежень щодо обчислювальних ресурсів системи антивірусного захисту.

Практичне значення одержаних результатів дисертаційного дослідження полягає у наступному:

– розроблене алгоритмічне та програмне забезпечення, що базується на створених нейромережевих методах та моделях, дозволило забезпечити достатню точність розпізнавання комп’ютерних вірусів та приблизно в 1,5 рази зменшити обчислювальні витрати, пов’язані з визначенням значень архітектурних параметрів ГНМ, що підтверджується актом впровадження в діяльність ТОВ «Сайфер ПРО» (акт впровадження від 17.08.2020);

– розроблені програми, що реалізують запропоновані моделі та методи, впроваджені в навчальний процес на кафедрі безпеки інформаційних технологій Національного авіаційного університету (акт впровадження від 25.02.2020).

– результати проведених розрахунків вказують на те, що ефективність розробленого НМЗ приблизно в 1,14 рази вища ніж у подібних відомих засобів. Таким чином, результати досліджень підтверджують можливість підвищення ефективності розпізнавання комп’ютерних вірусів за рахунок застосування розроблених нейромережевих моделей (НММ) та НМЗ, що підтверджується актом впровадження в діяльність ТОВ «Сайфер ПРО» (акт впровадження від 17.08.2020)

Особистий внесок здобувача. Всі основні результати дисертаційної роботи отримані здобувачем самостійно. У роботах, опублікованих із співавторами, здобувачу належить: [1] – дослідження підходів до виявлення шкідливих програм за допомогою штучних НМ; [2] – розроблено нейромережеву модель призначену для розпізнавання комп'ютерних вірусів; [3] – розроблено модель деобфускації програмного коду для визначення вхідних параметрів нейромережевої моделі призначеної для розпізнавання комп'ютерних вірусів; [4, 5] – розроблено метод застосування ГНМ для розпізнавання комп'ютерних вірусів; [8] – розроблено нейромережеву модель для розпізнавання комп’ютерних вірусів; [9] – розроблено підхід до визначення ефективності НМЗ розпізнавання шкідливого програмного забезпечення; [10] – розроблено підхід до визначення вихідних сигналів нейромережевої моделі призначеної для розпізнавання шкідливого програмного забезпечення; [11] – розроблено метод розпізнавання шкідливого

ПЗ; [14] – визначено перелік параметрів, що використовуються для розпізнавання комп'ютерних вірусів.

З робіт, що опубліковані у співавторстві, у дисертаційній роботі використовуються виключно результати, отримані особисто здобувачем.

Апробація результатів дисертації. Основні результати дисертації доповідались, обговорювались та отримали позитивні оцінки на наступних конференціях: Information Technologies, Management and Society, The 15th International Scientific Conference Information Technologies and Management (2017); Международной научно-практической конференции «Математические методы и информационные технологии макроэкономического анализа и экономической политики», посвященной празднованию 80-летнего юбилея академика НАН РК Абдыкаппара Ашимовича Ашимова (2017); Aviation in the XXI-st Century (2018); Advances in Computer Science for Engineering and Education. ICCSEEA (2018, 2019); The 10 th International Scientific Conference «ITSec» (2020); Стан та удосконалення безпеки інформаційно-телекомунікаційних систем (SITS' 2020); VI Міжнародна науково-практична конференція «Актуальні питання забезпечення кібербезпеки та захисту інформації» (2020).

Публікації. Основні положення дисертації опубліковано у 14 наукових працях, серед яких 7 наукових статей (4 статті у фахових наукових виданнях України, 3 – у міжнародних рецензованих виданнях, які входять до бази наукометричної бази даних SCOPUS), 1 закордонна колективна монографія, а також 6 матеріалів і тез доповідей на конференціях.

Структура та обсяг роботи. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків та списку використаних джерел (106 найменувань) на 11 сторінках, 2 додатки на 11 сторінках. Загальний обсяг дисертації становить 166 сторінок, у тому числі 144 сторінки основного тексту, ілюстрацій – 38, таблиць – 9.

РОЗДІЛ 1
АНАЛІЗ НЕЙРОМЕРЕЖЕВИХ ЗАСОБІВ РОЗПІЗНАВАННЯ КОМП'ЮТЕРНИХ ВІРУСІВ

1.1. Науково-практична задача розпізнавання комп'ютерних вірусів

В теперішній час в умовах глобалізації інформаційних процесів, входження України до світового інформаційного простору та потужної інформаційної експансії з боку інших держав, особливої гостроти набувають задачі, пов'язані із забезпеченням інформаційної безпеки держави [79-81]. Одним із основних напрямків вирішення цих задач є створення методів та засобів протидії комп'ютерним вірусам, покликаних попереджувати пошкодження державних інформаційних ресурсів [84]. Хоча використовуються такі методи та засоби вже не одне десятиліття, їх розробкою та створенням методологічної бази займаються висококваліфіковані фахівці, проте практичний досвід і результати багатьох науково-практичних досліджень вказують на наявність в сучасних антивірусах суттєвих недоліків [84] . Основним з них є недостатня точність розпізнавання всієї номенклатури комп'ютерних вірусів, що підтверджується відомими випадками успішних вірусних кібератак на інформаційні ресурси вітчизняних державних установ. Разом з тим впровадження відомих засобів розпізнавання комп'ютерних вірусів в систему інформаційної безпеки держави викликає необхідність їх складної адаптації до очікуваних умов використання [77]. Зазначимо, що відповідно до вітчизняних та міжнародних нормативних документів в області захисту інформації під поняттям комп'ютерного вірусу розуміють програму, що володіє здатністю до самовідтворення і, як правило, здатна здійснювати дії, які можуть порушити функціонування КС і/або зумовити порушення політики безпеки [81]. При цьому вважається, що комп'ютерний вірус може порушувати цілісність інформації, програмне забезпечення та (чи) режим роботи обчислювальної техніки. Також більшість дослджень вказують на те,

що основним завданням розробки систем протидії комп'ютерним вірусам є розпізнавання.

Методи розпізнавання комп'ютерних вірусів можуть бути розділені на 2 класи [1-2, 4]:

1. Методи, засновані на сигнатурах.

2. Методи, засновані на виявленні аномалій.

У більшості сучасних антивірусах центральне місце займає сигнатурний підхід. Він дає 100% точність виявлення вже відомих віруси, але не дозволяє розпізнати ті віруси сигнатури яких невідомі. Аномальний підхід навпаки дозволяє виявляти віруси з невідомими сигнатурами, однак його негативною рисою є висока ймовірність хибних спрацювань. Найчастіше аномальні методи базуються на використанні еврістичних правил. Існує чимало таких методів [3], але останнім часом найбільший потенціал представляють методи, які використовують машинне навчання. Аномальні методи виявляють вірус, використовуючи параметри, що характеризують відмінності функціонування програми від специфікації, що характерна для її нормальної поведінки. Переваги та недоліки сигнатурного аналізу:

1. Дозволяє визначати конкретний вірус з високою точністю і малою часткою помилкових спрацьовувань.

2. Малоефективний для розпізнавання поліморфних вірусів.

3. Вимагає регулярного і оперативного оновлення бази сигнатур.

4. Для реакції на невідомі віруси потрібні експертні правила, створення яких вимагає ручного аналізу вірусів і виділення сигнатур.

5. Нездатний виявити нові типи вірусів.

6. Для розпізнавання різних версії одного і того ж вірусу необхідні різні сигнатури.

7. З урахуванням великого обсягу бази сигнатур величезна, сигнатурний аналіз є достатньо ресурсоємною операцією.

Переваги та недоліки аномального методу виявлення комп'ютерних вірусів:

1. Можливість виявлення раніше невідомих вірусів (вірусів нульового дня (zero-day viruses)).

2. Висока ймовірність помилкових спрацьовувань, тобто таких при яких легітимне програмне забезпечення буде розпізнане, як вірус.

3. Висока складність навчання.

4. Для вже навченої системи аналіз виконується порівняно швидко

І сигнатурні і аномальні методи можуть використовувати три різних підходи для виявлення вірусів:

1. Статичний підхід. Використовуючи цей підхід, підозріла програма аналізується статично (тобто без запуску самої програми), як звичайний файл.

2. Динамічний підхід. При цьому підході, підозріла програма аналізується динамічно, тобто під час її виконання в реальному часі.

3. Гібридний підхід. Об'єднання статичного і динамічного підходів в різних частинах аналізу шкідливої програми.

Створення класифікатора для виявлення вірусів умовно можна розділити на три стадії:

1. Формування простору ознак опису файлу (features extraction). Результатом цієї стадії є вектор, що містить прізнаковие характеристики даного об'єкту. У задачі побудови класифікатора статичного виявлення вірусів ознаками можуть виступати наступні об'єкти:

- Рядки - виконуваний файл розглядається як звичайна рядок або послідовність рядків. Ознаки - числові характеристики рядків (наприклад, частота нулів в підрядку);

- Структурні елементи файлу, що виконується. Для Windows-орієнтованих систем, це стосується PE-файлів (Portable Executable). Ознаки, витягнуті з структурної інформації можуть бути наступними: сертифікат, date/time stamp, інформація компонувальника, тип CPU, логічна інформація (вирівнювання секцій, розмір, секції коду, налагоджувальні прапори), інформація про імпорт (список тих DLL, які використовує виконуваний файл), інформація про експорт (функції які надає PE-файл іншим програмам),

таблиця релокацій (переміщень) директорії ресурсів.

- N-грами на рівні байт. Сегменти послідовних байт з різних місць всередині виконуваного файлу довжини N. Кожна N-грами розглядається як ознака.

- N-грами на рівні опкодів. Мається на увазі, що опкод (opcode - operation code) - специфічний для CPU операційний код, який виконує спеціальну машинну команду (наприклад, mov, push, add).

2. Вибір ознак (feature selection). Протягом цієї фази вектор, створений на першій стаді обчислюється, а надлишкові і нерелевантні ознаки викидаються з розгляду. Реалізація цієї стадії дозволяє збільшити ефективність процесу навчання моделі за рахунок скорочення кількості необхідних операцій і, як наслідок, збільшення швидкості навчання, підвищити узагальнюючі здатності за рахунок скорочення розмірності простору ознак, підвищити якість інтерпретації навчання. Завдання цієї стадії полягає в тому, щоб з вже виявлених ознак вибрати найбільш значимі (інформативні). Існує декілька підходів до виділення інформативності ознак. Найбільш популярними є кореляційні методи [10, 19].

3. Побудова математичної моделі, класифікатора, який використовує вектор ознак, отриманий на попердній стадії. Для побудови класифікатора можуть використовуватися такі підходи: дерева рішень, випадковий ліс, метод найшвидшого бустінгу, логістична регресія, метод опорних векторів, метод k-найближчих сусідів, метод адаптивного бустингу, наївний Байес, нейронні мережі. Враховуючи останні досягнення в області теорії НМ найбільшу перспективу мають класифікатори на їх основі.

Практичний досвід та результати [30] вказують на те, що в сучасних умовах однією із найбільш актуальних є задача розпізнавання пліморфних та зашифрованих вірусів при створенні яких використовується технологія обфускації програмного коду. Зазначимо, що обфускація це приведення виконуваного коду або вихідного тексту програми до виду, який зберігає її функціональність, але ускладнює розуміння, аналіз алгоритмів роботи, а

також модифікацію при декомпіляції. Легітимною підставою для використання процедури обфускації є те, що ця технологія використовується і для безпечних програм з метою забезпечення дотримання авторських прав.

Обфускований програмний код мало придатний для аналізу в антивірусних засобах, а тому виникає необхідність реалізації процедури деобфускації. Тобто необхідно перевести обфускований програмний код до вигляду, що придатний для аналізу. Методів деобфускації достатньо багато. Базуються вони на різних підходах і мають різні можливості і різну ступінь ефективності. Одним із найбільш сучасних є підхід на основі аналізу графу виконання програм. До переваг цього підходу слід віднести його універсальність та інтерпретованість результатів виконання.

1.2. Характеристика сучасних типів нейромережевих моделей

Більшість сучасних типів НММ в певному сенсі є розвитком двохшарового персептрону, що представляє собою НМ з прямим поширенням сигналу, для навчання якої використовується методи котрі базуються на алгоритмі зворотнього поширення помилки [104-106]. Структура двохшарового персептрону показана на рис. 1.2.1.

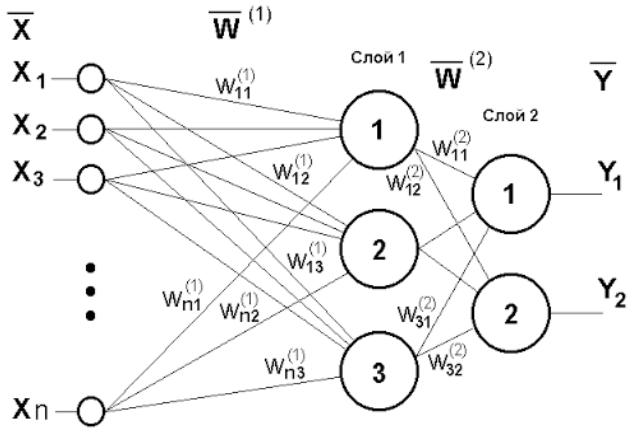

Рис. 1.2.1. Структура двохшарового персептрону

На рис. 1.2.1 символом X позначено вектор вхідних параметрів, символом W матриця вагових коефіцієнтів, а символом Y вектор вихідних

параметрів. Вхідним нейронам двохшарового персептрону притаманна лінійна функція активації виду (1.2.1), а схованим та вихідним нейронам – сигмоїд або гіперболічний тангенс, що задаються виразами (1.2.2, 1.2.3). Для розрахунку у виразах (1.2.1-1.2.3) сумарного вхідного сигналу використовується вираз (1.2.4)

$$F(\text{NET}) = \text{NET} \times a, \tag{1.2.1}$$

$$F(\text{NET}) = \frac{1}{1+e^{-a\times\text{NET}}}, \tag{1.2.2}$$

$$F(\text{NET}) = \frac{e^{a\times\text{NET}}-e^{-a\times\text{NET}}}{e^{a\times\text{NET}}+e^{-a\times\text{NET}}}, \tag{1.2.3}$$

$$\text{NET} = \sum_{i=0}^{K} x_i w_i, \tag{1.2.4}$$

де K – кількість вхідних зв'язків нейрону, x_i – величина і-го зв'язку, w_i – вага і-го зв'язку, NET - вхідний сигнал нейрону.

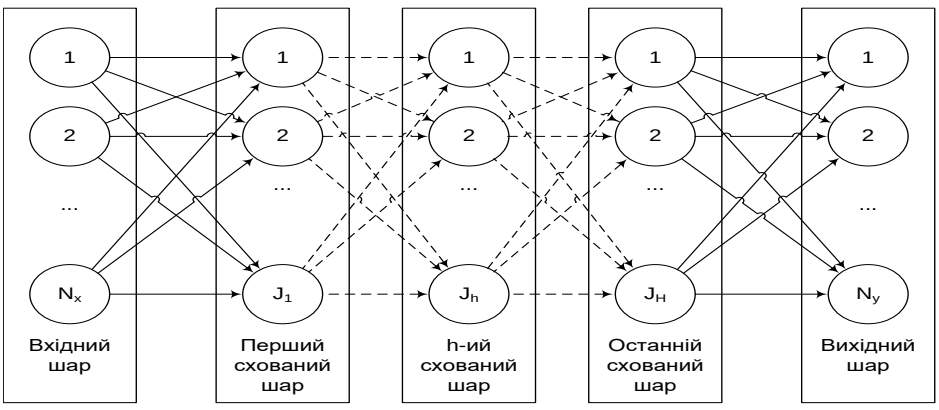

Рис. 1.2.2. Структура глибокої нейронної мережі з прямим розповсюдженням сигналу

Основою сучасних НММ є так звана глибока нейронна мережа (DNN), класична структура якої наведена на рис. 1.2.2. [101].

На відміну від двохшарового персептрону в DNN кількість схованих шарів більша за 1. Крім того, особливістю DNN є використання в схованих та вихідних нейронах функції активації типу ReLU:

$$Y(X)=\max(0,X), \tag{1.2.5}$$

де Y – вихідний сигнал нейрону, X – сумарний вхідний сигнал нейрону.

При цьому сумарний вхідний сигнал нейрону розраховується за допомогою (1.2.4).

В деяких випадках у вихідних нейронах використовується функція активації типу Softmax:

$$Y(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}, \tag{1.2.6}$$

Для визначення ефективності ГНМ традиційно використовуються показники точності та втрат, що визначаються виразами:

$$A = N_{\text{right}}/N, \tag{1.2.7}$$

$$L = N^{-1} \sum_{t=1}^{N} e^{T}(t,Q)\, W(\theta) e(t,Q), \tag{1.2.8}$$

де N_{right} - кількість прикладів, що розпізнані правильно, N - загальна кількість прикладів, $e(t, Q)$ - вектор помилки в момент часу t, n_y - кількість виходів нейроної мережі, $W(Q)$ – матриця вагових коефіцієнтів.

Однією із найбільш апробованих модифікацій DNN є згорткова нейронна мережа (ЗНМ), базова структура якої показана на рис. 1.2.3. На відміну від класичної DNN ЗНМ пристосована до розпізнавання кольорових зображень, що дозволяє застосовувати її у випадках для класифікації образів, коли важливою є топологія даних. Найбільш відомими різновидами ЗНМ

являються LeNet-5, MobileNet, SqueezeNet, VGG16, AlexNet, GoogleNet та інші. Крім врахування топології до основних переваг ЗНМ щодо повнозв'язних ГНМ відносять знано меншу кількість вагових коефіцієнтів НМ, що позитивно відображається на ресурсоємності НМЗ.

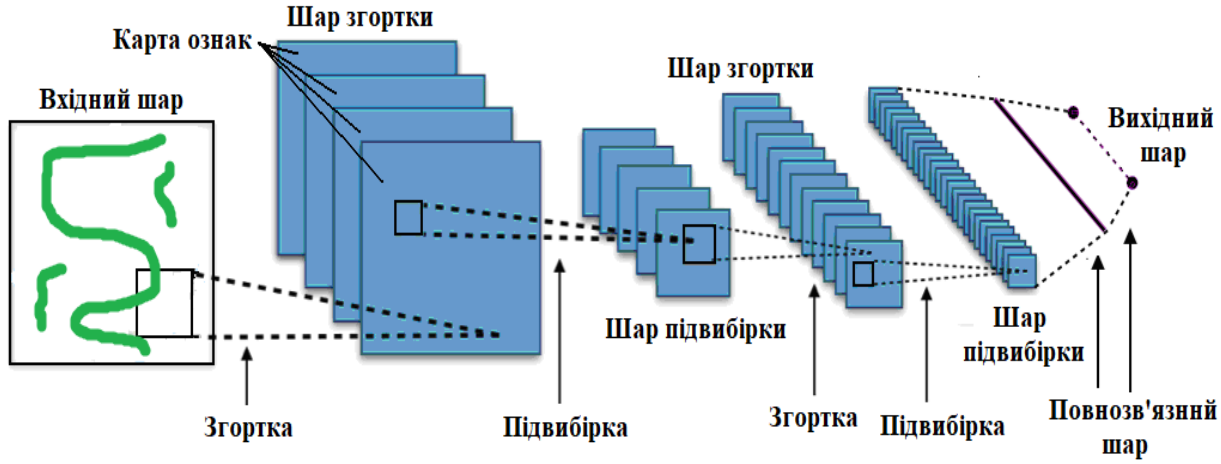

Рис. 1.2.3. Структурна схема згорткової нейронної мережі з прямим розповсюдженням сигналу типу LeNet-5

В класичних ЗНМ типу LeNet-5 кількість вихідних нейронів дорівнює кількості розпізнаваних образів. Структура прихованих нейронних шарів підбирається емпірично. Сумарний вхідний сигнал деякого нейрона шару згортки розраховується так:

$$x_k^{(i,j)} = f\left(x_{0,k} + \sum_{s=1}^{K}\sum_{t=1}^{K} w_{k,s,t}\, x^{((i-1)+s,(j+t))} \right) \qquad (1.2.9)$$

де $x_k^{(i,j)}$ - вхідний сигнал (i, j) -го нейрону k-ої карти ознак, $x_{0,k}$ - зміщення кожного з нейронів k-ої карти ознак, K - розмір рецептивної області нейрону (розмір ядра згортки), $w_{k,s,t}$ - ваговий коефіцієнт (s, t)-го синаптичного зв'язку нейрона k-ої карти ознак, x - вихід нейрону попереднього шару.

Вихідний сигнал нейрона карти ознак розраховується шляхом

підстановки вхідного сигналу в функцію активації:

$$y=f(x). \tag{1.2.10}$$

У сучасних варіантах ЗНМ в якості функції активації схованих нейронів використовується функція ReLU. В цьому випадку вираз (1.2.3) замінюється виразом (1.2.5). В нейронах вихідного шару, як правило, використовується функція активації типу Softmax, що задається виразом (1.2.6).

Важливим недоліком класичних ЗНМ є складність аналізу динамічних рядів даних, адже кількість вхідних параметрів ЗНМ так само, як і ГНМ є фіксованою величиною. Для виправлення останнього недоліку використовують рекурентну ЗНМ, структура якої показана на рис. 1.2.4. Особливістю наведеної структури є наявність рекурентного модулю, що дозволяє опрацьовувати серію подій в часі. В якості означеного рекурентного модулю як правило використовують блок LSTM-пам'яті (long short-term memory network), структура якого показана на рис. 1.2.5.

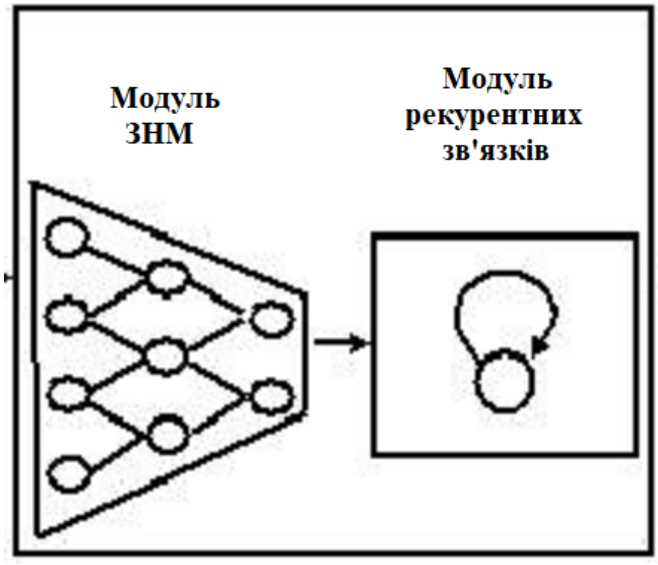

Рис. 1.2.4. Структурна схема рекурентної згорткової нейронної мережі

На відміну від класичних рекурентних НММ блок LSTM-пам'яті містить в собі так звані вентилі. Вхідний вентиль спрацьовує, коли на вхід блоку

LSTM-пам'яті надходять новий образ. В цьому випадку блок повинен визначити, чи несе даний образ нову корисну інформацію, яку варто додати в довготривалу пам'ять. Таким чином, при надходженні нового вхідного образу блок LSTM-пам'яті спочатку забуває всю довготривалу інформацію.

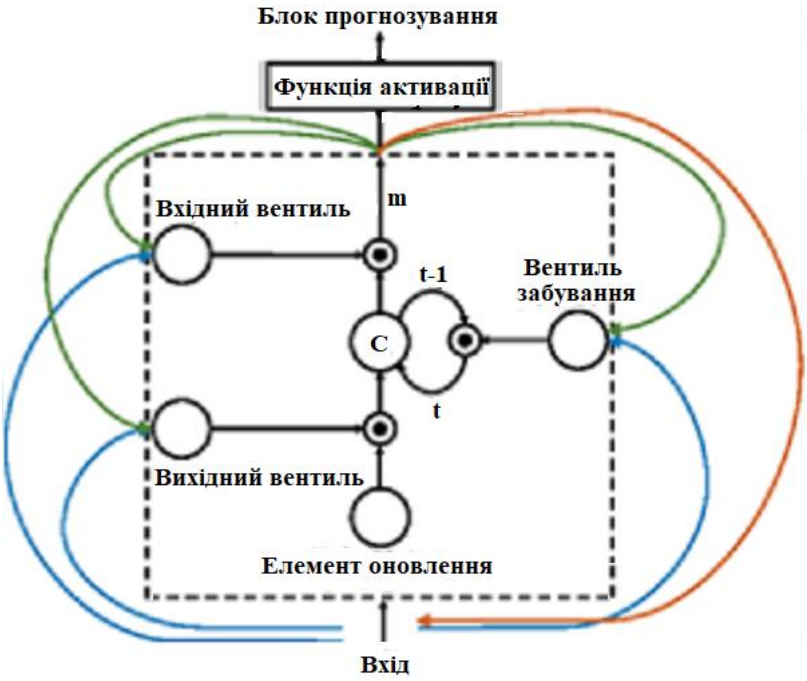

Рис. 1.2.5. Структурна схема блоку LSTM-пам'яті

Після цього визначається, які елементи вхідного образу можуть бути корисні і додаються в довгострокову пам'ять. Вентиль забування призначений для того, щоб в процесі навчання сформувати спеціальний механізм забування: коли приходить нова вхідна інформація, блок LSTM-пам'яті повинен знати, які знання слід продовжувати пам'ятати, а які слід забути. Вихідний вентиль визначає, які елементи довгострокової пам'яті можуть стати в нагоді в самий найближчий час. Наприклад, певна сигнатура k-го комп'ютерного вірусу є корисною інформацією, яку має сенс зберігати в довготривалій пам'яті, але якщо k-ий комп'ютерний вірус не підлягає розпізнаванню, ця інформація, швидше за все, не знадобиться.

Таким чином, замість того, щоб завжди використовувати всю довготривалу пам'ять, мережа навчається фокусувати увагу на певних її

елементах. За допомогою описаних вище механізмів навчання блок LSTM-пам'яті визначає, які елементи нового навчального образу необхідно забути, які додати, а на яких необхідно сфокусувати увагу. Такий підхід дозволяє відстежувати інформацію протягом більш тривалих періодів часу.

В багатьох випадках важливою проблемою побудови НММ є недостатня кількість маркованих навчальних прикладів. Для вирішення вказаної проблеми використовується автоендкодер, структура якого показана на рис. 1.2.6.

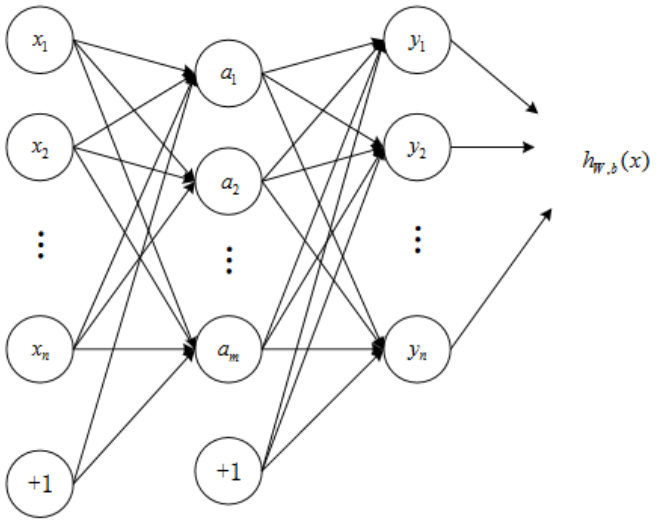

Рис. 1.2.6. Архітектура автоендкодера

На рис 1.3.8 вихідний сигнал автокодувальника відповідає вектору ($y_1, y_1,$ ... y_n). Навчання автокодувальника реалізується за допомогою алгоритму зворотнього розповсюдження помилки. При цьому цільова функція навчання визначається за допомогою виразу:

$$f_{W,b}(x) \approx x. \qquad (1.2.11)$$

Використання даної цільової функції передбачає те, що вихідний сигнал автокодувальника повинен дорівнювати вхідному сигналу. Таким чином, навчання класичного автокодувальника зводиться до того, що б за допомогою алгоритму зворотнього поширення помилки знайти такі значення вагових

коефіцієнтів при яких вихідний сигнал буде дорівнює вхідному [42]. При цьому навчальні приклади можуть бути немаркованими, тобто не містити очікуваний вхідний сигнал. Пошук оптимального значення вагових коефіцієнтів проводиться за допомогою градієнтного спуску шляхом мінімізації функції втрат.

1.3. Аналіз нейромережевих моделей та методів розпізнавання комп'ютерних вірусів

В процесі аналізу робіт, присвячених використанню НМ в САЗ для розпізнавання комп'ютерних вірусів, виявлено, що в багатьох випадках помітна невідповідність назви та опису наведеної розробки. Тому аналіз вказаних робіт здійснено з точки зору визначення основних характеристик нейромережевих методів та моделей. Крім того, додатково проаналізовані нейромережеві засоби розпізнавання кібератак, функціональність яких схожа з функціональністю засобів розпізнавання комп'ютерних вірусів.

В роботах [103] показано, що основною задачею застосування НМ в САЗ являється розпізнавання ШПЗ на основі узагальнення контрольованих параметрів, відображених в навчальних прикладах [100]. При цьому процес нейромережевого розпізнавання ШПЗ, як правило полягає в нейромережевій оцінці величин множини контрольованих параметрів. Якщо виставлена за допомогою НМ оцінка знаходиться в певному діапазоні, то вважається, що ШПЗ розпізнано, а у випадку виходу за межі цього діапазону вважається, що в комп’ютерній системі ШПЗ відсутнє.

Відповідно описаній в [90] методології розробки НМЗ захисту інформації основні напрямки підвищення ефективності таких засобів пов’язані з адаптацією виду та параметрів НММ до очікуваних умов застосування, які в першу чергу визначаються використаною множиною вхідних параметрів.

В статті [30] розглянуто проблеми детектування складних поліморфних і метаморфних комп'ютерних вірусів. Запропоновано і детально розглянуто

новий метод побудови функціональної сигнатури вірусу (**МПФС**) на основі послідовності системних викликів. Показано, що наведена функціональна сигнатура на основі послідовності системних викликів (трас виконання) може бути використана для детектування складних поліморфних і метаморфних вірусів, для яких побудова класичної сигнатури ускладненою або і неможливою. Зазначено, що методи автоматичного виділення сигнатур на стороні кінцевого користувача на основі даних евристичного або поведінкового аналізу можуть стати основою антивірусного ПЗ нового покоління.

В роботі [9] розглянуто підхід кодової нормалізації (**ПКД**) - застосування методів деобфускації/оптимізації для виділення мінімальної функціонально еквівалентної форми послідовності інструкцій. Основним недоліком запропонованого методу є припущення про коректне дизассемблювання нормалізованого коду, зроблене відповідно до того, що вірус при побудові нової копії повинен сам дизасемблювати.

У роботах [1, 12] розглянуто підхід до розпізнавання метаморфних вірусів (**ПРМВ**), яким не потрібно проводити самостійне дизасемблювання при поширенні і генерації нових копій.

В роботі [82], розроблено класифікатор для статичного виявлення комп'ютерних вірусів (**КСВВ**), що базується на машинному навчанні. Показано недоліки відомих підходів до формування множини вхідних параметрів. У підходу з виділенням рядків недолік полягає в тому, що різні класи вірусів мають різні рядкові описи. Відповідно підхід ефективний тільки при розпізнаванні конкретного тип у вірусів. У підходу на основі N-грам байт погана інтерпретація результатів. У підходу на основі N-грам опкодів недостатня точність виявлення. Підхід на основі структурних ознак вразливий з точки зору підробки. Розроблений класифікатор використовує змішаний підхід.

Стаття [17] присвячена розробці евристичного сканера САЗ (**ЕССАЗ**) на базі НМ АРТ-1, архітектура якої наведена на рис. 1.3.1. В якості джерела

вхідних параметрів НММ використано параметри мережевої активності, а саме: num_failed_logins – кількість невдалих спроб входу; num_root – число спроб входу root-користувача, або адміністратора; src_bytes – кількість даних (в байтах) від дже- рела до адресата; dst_bytes – кількість даних (в байтах) від адре- сата до джерела; num_file_creations – число операцій створення файлу; num_shells – число підказок оболонки; num_access_files – число операцій з файлами доступу.

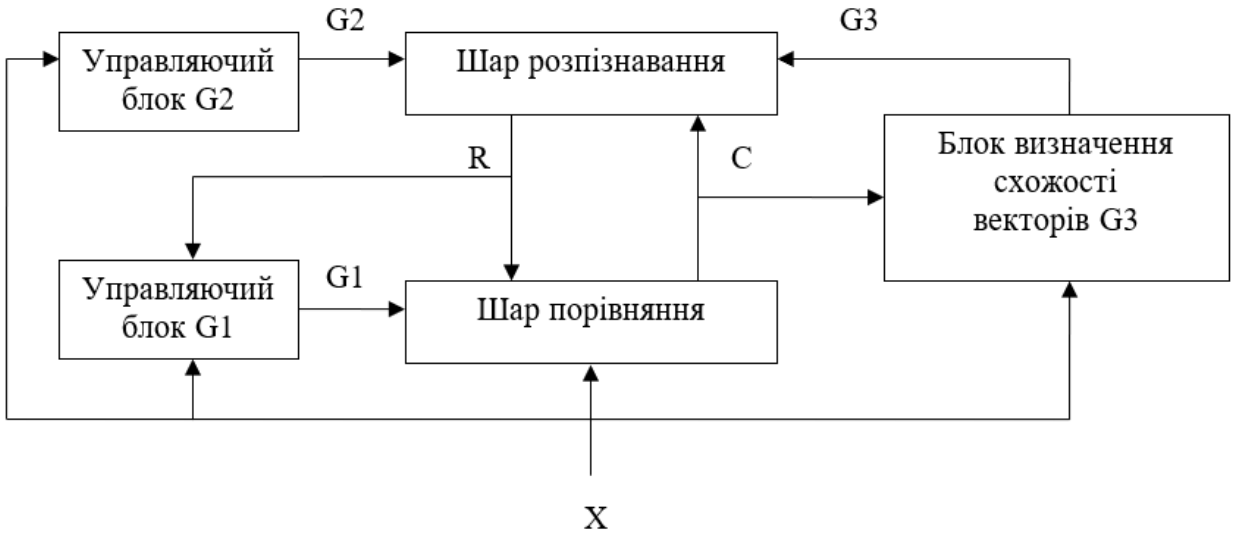

Рис. 1.3.1 Архітектура нейронної мережі типу АРТ-1

В статті [13] розглянуті особливості програмної реалізації алгоритмів методики формування навчальної множини (**АМФНМ**) для бінарних класифікаторів, що використовуються при евристичному статистичному аналізі в антивірусах. Зазначається важливість етапу підготовки навчальної вибірки. В якості джерела вхідних даних запропоновано використовувати N-грами байт програмного коду.

В роботах [32, 57] наведено опис методів визначення фрагментів програмного коду (**МВФПК**), призначених для визначення переліку та оцінки значень вхідних параметрів НММ, що використовуються в системах детектування ШПЗ та в системах антивірусного захисту. Для підвищення інформативності контрольованих параметрів в методі застосована процедура

їх попередньої обробки. Також описано підхід до застосування НММ на базі топографічної карти Кохонена, структура якої показана на рис. 1.3.2. Задекларовано, що вибір виду НММ відбувався за рахунок проведення порівняльних числових експериментів. В якості критерію порівняння використано термін навчання.

В роботі [28] запропоновано підхід до визначення переліку вхідних параметрів НММ (**ПВПВП**), призначеної для розпізнавання ШПЗ на основі аналізу програмного коду. Підхід передбачає використання для деобфусикації теоретичних рішень, що використовуються для оптимізації програмного коду. Також розроблена процедура деобфускації програмного коду з використанням графа залежності значень і станів. Встановлено, що використання розробленої процедури дозволяє представити функціональну семантику тестованих програм у вигляді графа. В результаті стало можливим нейромережеве виявлення ШПЗ на основі семантики його виконання.

В [12] запропоновано систему розпізнавання комп'ютерних вірусів на основі методу нейромережевого аналізу нормалізованих сигнатур (**МНАНС**). Декларується точність розпізнавання в межах 80-91%. Вказується на можливість розпізнавання поліморфних вірусів. В якості базової НММ використано двохшаровий персептрон, структура якого показана на рис. 1.2.1.

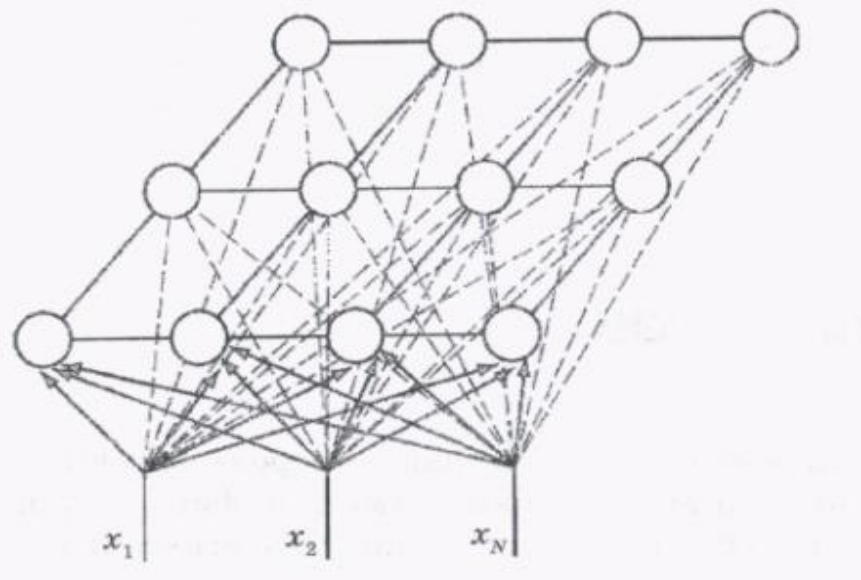

Рис. 1.3.2 Архітектура нейронної мережі типу карти Кохонена

Схожі результати отримані і в [22, 39, 41, 46] де також використано систему нейромережевого розпізнавання комп'ютерних вірусів (**СНРКВ**) на базі двохшарового персептрону з одним або двома вихідними нейронами.

При цьому вхідні нейрони співвідносяться із параметрами, що характеризують структуру PE-файлів. Основною відмінністю між результатами [22, 39, 41, 46] є використання різних підходів до попередньої обробки вхідних параметрів НММ.

В роботі [98] описані експерименти, що свідчать про точність розпізнавання комп'ютерних вірусів на рівні 91%. Використано сигнатури комп'ютерних вірусів представлені в базі даних malwr. Для розпізнавання використано підхід до відображення динаміки передачі ШПЗ (**ПВДП**).

Відзначимо, що в [22, 39, 41, 46] механізму оптимізації структури двохшарового персептрону та механізму формування навчальної вибірки не наведено. Також викликають певні сумніви у доцільності використання НММ на базі достатньо застарілих НММ типу двохшарового персептрону та топографічної карти Кохонена.

В [45] розроблена нейромережева система розпізнавання з переднавчанням на основі автоендкодера (**НСРА**). Використано DNN, що складається із 8 шарів, кожен із яких містить 30 нейронів. Для отримання множини вхідних даних DNN використано спеціально розроблений метод автоматичної генерації сигнатур комп'ютерних вірусів. Наведено результати числових експериментів в яких точність розпізнавання досягає 98%. Разом з тим в [45] вказано, що DNN навчається тільки за допомогою механізму автоендкодера на не маркованих даних. Це дещо знижує достовірність отриманих результатів, оскільки вважається, що для забезпечення високої точності розпізнавання також необхідно передбачити навчання DNN за допомогою алгоритму зворотного поширення помилок на маркерованих навчальних даних.

В [55] наведено опис методу застосування згорткових нейронних мереж (CNN) для розпізнавання ВПО (**МЗЗНМ**). Зазначимо, що CNN є одним із

різновидів DNN, який традиційно використовується для розпізнавання зображень. Для цього в [55] розроблено спосіб перетворення сигнатури програмного коду у сіре масштабоване зображення розміром 32x32 пікселів. Наведено описи проведених експериментів, в яких досліджувався вплив структурних параметрів CNN на точність розпізнавання. Розглянуто три варіанти структурних рішень CNN типу LeNet-5. Для найкращого варіанту точність розпізнавання становить 93.86%. Сформульовано пропозицію про необхідність розробки теоретичних положень щодо адаптації структурних параметрів CNN до умов задачі розпізнавання ВПО. В роботах [39, 40] в якості бази алгоритму пошуку ШПЗ використаний бустінг на вирішальних деревах (XGBoost). Робота [13] присвячена аналізу методів машинного навчання для виявлення ШПЗ. Обгрунтовано висновок про доцільність застосування нейромережевих аналізаторів вдосконалених за рахунок впровадження сучасних нейромережевих рішень.

В дослідженні [46] розроблено спосіб нейромережевого аналізу (**СНМА**) PE-файлу представленого у вигляді чорно-білого зображення. Завдання стояло у визначенні типу (malware family) ШПЗ. Сукупність електронних даних представляли у вигляді матриці (0: чорний, 1: білий). Ширина зображення фіксувалася в залежності від розміру файлу (32, <10kB, 64, 128, ..., 1024, > 1000kB). На основі зображень були визначені загальні патерни, текстури для 25 типів шкідливих ПО. Для аналізу текстури застосовувався фільтр Габора. Ознаки, отримані із зображень використовували в класифікаторі метод k-найближчих сусідів з метрика Евкліда. Хоч дослідження і представляє інтерес, але через тривалість нейромережевого аналізу має суттєві обмеження.

В статтях [90, 91] та дисертації [91] розроблені нейромережеві методи та моделі протидії атакам на ресурси інформаційних систем (**НММПА**). Роботи акцентовані на вирішення питань, що виникають при розпізнаванні мережевих кібератак за допомогою сучасних типів НММ. Відзначається, що основні перспективи розпізнавання пов'язані з застосуванням ГНМ з прямим

розповсюдженням сигналу. Розглянута задача розробки такої НММ при недостатньому обсягу навчальних даних. Запропоновано методи навчання НММ за допомогою експертних знань та автоендкодера, архітектура якого показана на рис. 1.2.6.

В роботах [85-89] представлена нейромережева методологія оцінки параметрів безпеки Інтернет-орієнтованих інформаційних систем (**НМОПБ**), що призначена для розпізнавання кібератак. Зазначимо, що поняття кібератаки є достатньо широким і включає в себе в тому числі і атаки реалізовані за рахунок ШПЗ.

Серед проаналізованих, дана розробка є найбільш фундаментальною. У ній отримали подальший розвиток теоретичні положення побудови НМЗ оцінки ПБ, які полягають в розроблених підходах до розпізнавання поступових і несподіваних кібератак, визначенні оптимального виду НММ, доцільності застосування НМЗ, класифікації статистично подібних кібератак, застосуванні продукційних правил для представлення експертних знань, параметрах оцінки ефективності НМЗ. Також розроблені моделі створення і використання НМЗ оцінки ПБ, які за рахунок застосування розроблених теоретичних положень дозволяють: визначити перелік оцінюваних ПБ, а також зменшити ресурсомісткість створення НМЗ. На основі зазначених моделей розроблений ряд методів, що дозволяють підвищити ефективність використання НМЗ Так, метод уявлення експертних знань для НМЗ оцінки ПБ, дозволяє забезпечити оперативність розпізнавання і розширити множину типів кібератак, для яких відсутні статистичні дані. Метод визначення тимчасових характеристик використання НМЗ оцінки ПБ завдяки використанню розроблених аналітичних залежностей між очікуваними і допустимими термінами розробки забезпечує можливість визначення доцільності застосування зазначених засобів.

Метод визначення ефективності розробки НМЗ оцінки ПБ, що за рахунок застосування запропонованих параметрів оцінки ефективності та сформованого інтегрального показника ефективності дозволяє вибрати

найбільш ефективний засіб. Застосування методу дозволило визначити, що у відомих НМЗ розпізнавання кібератак недостатньо повно розроблений механізм формування навчальної вибірки при кодуванні очікуваного вихідного сигналу, не враховується близькість еталонів класів, котрі мають бути розпізнані. Крім цього, такі засоби недостатньо адаптовані до застосування сучасних типів НСМ. На основі взаємопов'язаного використання розроблених підходів, моделей і методів розроблено методологію нейромережевої оцінки ПБ, що дозволяє значно розширити функціональні можливості НСР і вибрати з них найбільш ефективне. Також наведено опис та результати експериментів в яких для розпізнавання ШПЗ було використано НММ на базі двохшарового персептрону. При цьому оптимізація параметрів та оптимізація процедури навчання НММ не проводилась.

Разом з тим можна відзначити, що прогрес в області теорії НМ та захисту інформації зумовлює можливість доповнення вказаних рішень за рахунок застосування нових типів НММ та нових підходів до розпізнавання ШПЗ. З позицій мети представленого дослідження викликає інтерес визначений в роботах [85-91] перелік параметрів, що характеризують ефективність НМЗ розпізнавання кібератак. Однак при оцінці ефективності НМЗ розпізнавання комп'ютерних вірусів даний перелік параметрів слід модифікувати через його надмірність та необхідність врахування особливостей сучасного стану рішень в області теорії НМ. Також в переліку доцільно відобразити особливості представленого дослідження.

1.4. Шляхи вдосконалення нейромережевих засобів розпізнавання комп’ютерних вірусів

В результаті проведеного аналізу встановлено, що підвищення ефективності сучасних НМЗ розпізнавання комп'ютерних вірусів йде шляхом забезпечення в них певних можливостей, які характеризуються за допомогою параметрів, представлених в табл. 1.1.

Таблиця 1.1

Параметри оцінки ефективності нейромережевих засобів

Назва параметру	**Опис параметру**
R_1	Наявність процедури кодування вхідних параметрів
R_2	Наявність процедури нормалізації вхідних параметрів
R_3	Визначення найбільш ефективного типу глибокої нейронної мережі
R_4	Визначення параметрів глибокої нейронної мережі
R_5	Оптимізація методу навчання
R_6	Можливість навчання за допомогою експертних правил
R_7	Визначення доцільності використання типу глибокої нейронної мережі для вирішення поставленої задачі
R_8	Наявність процедури деобфускації
R_9	Наявність процедури формування навчальної вибірки з різнорідних статистичних даних

Також зроблено висновок про те, що ефективність НМЗ розпізнавання комп'ютерних вірусів в значній мірі залежить від пристосованості до застосування ГНМ та проведення деобфускації програмного коду, що принципово забезпечує можливість нейромережевого розпізнавання поліморфних вірусів. Даний висновок сформульований на підставі результатів робіт [26], в яких обґрунтовано підходи до визначення параметрів нейромережевих моделей та деобфускації програмного коду комп'ютерних вірусів. За рахунок цього, запропоновано використання параметрів R_3, R_4, R_7 та R_8, опис яких також представлено в табл. 1.1. Надалі представлений в табл.1.1 перелік параметрів може бути розширено.

Наявність вказаних процедур в проаналізованих нейромережевих моделях та методах наведено в табл. 1.2.

Таблиця 1.2

Наявність базових процедур, що визначають ефективність нейромережевих засобів розпізнавання комп'ютерних вірусів

№	Назва засобу	Параметр								
		R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
1	МПФС	+	+	-	-	-	+	-	-	+
2	ПКД	+	-	-	-	-	-	-	+	-
3	ПРМВ	+	+	-	-	-	-	-	-	-
4	КСВВ	+	+	-	-	-	-	-	-	+
5	ЕССАЗ	+	+	-	-	-	-	-	-	-
6	АМФНМ	+	+	-	-	-	-	-	-	+
7	МВФПК	+	+	-	-	-	-	-	-	-
8	ПВПВП	+	+	-	-	-	-	-	+	-
9	МНАНС	+	+	-	-	-	-	-	-	-
10	ПВДП	+	+	-	-	-	-	-	-	-
11	НСРА	+	+	-	+	+	-	-	-	+
12	МЗЗНМ	+	+	-	+	+	-	-	-	+
13	СНМА	+	+	-	+	+	-	-	-	+
14	НММПА	+	+	-	-	+	+	-	-	-
15	НМОПБ	+	+	-	+	+	+	-	-	+
16	СНРКВ	+	+	-	-	-	-	-	-	-

Аналіз даних табл. 1.2 вказує на недостатню адаптацію відомих нейромережевих НМЗ до застосування сучасних рішень в області НМ та до розпізнавання обфускованих комп'ютерних вірусів. В підсумку проведений аналіз дозволяє стверджувати, що для підвищення ефективності розпізнавання комп'ютерних вірусів необхідно розробити відповідний нейромережевий метод, в якому вказані недоліки будуть виправлені. Розробка такого методу призводить до необхідності вдосконалення методологічних засад застосування НММ.

В першому наближенні передбачено оцінювання величин запропонованих параметрів по бінарній шкалі: 0 або 1. Деякий параметр дорівнює 0, якщо відповідна йому процедура в НМЗ не забезпечується і величина параметру дорівнює 1 в протилежному випадку.

При цьому для всіх проаналізованих методів $R_3=R_7=0$. Тобто в більшості

з відомих методів нейромережевого розпізнавання комп'ютерних вірусів не реалізована процедура визначення доцільності використання типу глибокої нейронної мережі для вирішення поставленої задачі та процедура визначення найбільш ефективного типу глибокої нейронної мережі.

Крім того, використання запропонованих критеріїв дозволяє за аналогією з [26] визначити інтегральний показник ефективності НМЗ (R_Σ) за допомогою наступного виразу:

$$R_\Sigma = \sum_{i=1}^{9}(\alpha_i R_i), \tag{1.4.1}$$

де α_i– ваговий коефіцієнт і-го критерію ефективності.

Визначити найбільш ефективний НМЗ можна, скориставшись виразом:

$$\max_{E_i} = \{E_1, E_2, .. E_I\}, \tag{1.4.2}$$

де I – кількість типів НМЗ, R_Σ – інтегральний показник ефективності i-го НМЗ.

У загальному випадку визначення вагових коефіцієнтів вимагає окремого дослідження, а в базовому варіанті можна припустити, що $\alpha_i = 1$.

1.5. Висновки до першого розділу

В результаті проведеного аналізу можливо зробити висновок про те, що важливим та актуальним напрямком підвищення ефективності систем розпізнавання комп'ютерних вірусів є застосування нейромережевих моделей та методів.

В той же час аналіз науково-практичних досліджень, присвячених нейромережевим технологіям, вказує на необхідність їх подальшого вдосконалення в напрямку адаптації до очікуваних умов експлуатації. При цьому очікувані умови впровадження нейромережевих засобів характеризуються варіативністю обмежень на термін розробки САЗ,

обмеженістю обчислювальних ресурсів, що можуть бути використані при побудові НММ, типом антивірусної системи, відсутністю достатньо повних баз даних прикладів комп'ютерних вірусів, необхідних для проведення навчання НММ та забезпеченням можливості аналізу обфускованого програмного коду.

В підсумку, на підставі проведеного аналізу визначено, що для створення ефективних нейромережевих засобів розпізнавання комп'ютерних вірусів, адаптованих до умов вітчизняних систем антивірусного захисту, необхідно вирішити ряд завдань:

- розробити принципи та правила визначення ефективних видів глибоких нейронних мереж, призначених для розпізнавання комп'ютерних вірусів;

- розробити принципи та модель формування параметрів навчальних прикладів глибокої нейронної мережі, що забезпечать можливість нейромережевого аналізу обфускованого програмного коду;

- розробити метод проектування архітектури глибокої нейронної мережі, призначеної для розпізнавання вірусів, що забезпечить пристосованість такої мережі до очікуваних умов застосування;

- дослідити та розвинути метод нейромережевого розпізнавання комп'ютерних вірусів, який забезпечить достатню похибку розпізнавання при різних умовах застосування з врахуванням обмежень щодо створення навчальної вибірки та обмежень щодо обчислювальних ресурсів системи антивірусного захисту.

РОЗДІЛ 2. ЕЛЕМЕНТИ МЕТОДОЛОГІЧНОЇ БАЗИ НЕЙРОМЕРЕЖЕВОГО РОЗПІЗНАВАННЯ КОМП'ЮТЕРНИХ ВІРУСІВ

2.1. Концептуальна модель забезпечення ефективності нейромережевого розпізнавання комп'ютерних вірусів

В результаті досліджень, проведених в першому розділі дисертаційної роботи визначено, що одним із найбільш важливих напрямків розвитку антивірусних засобів співвідноситься з задачею розробки НМЗ розпізнавання комп'ютерних вірусів. Важливою складовою вказаної задачі є необхідність визначення параметрів НММ, адаптованих до умов розпізнавання комп'ютерних вірусів, як у випадку деобфускованого, так і у випадку обфускованого програмного коду. Базуючись на результатах [28] можна стверджувати, що таких умов відносяться:

- Допустимий термін розробки;

- Обсяг матеріальних ресурсів, які можливо використати при створенні засобів розпізнавання.

- Доступність баз даних параметрів комп'ютерних вірусів, необхідних для навчання НММ.

- Доступний обсяг обчислювальних ресурсів антивірусних засобів.

При цьому питання реєстрації та збереження параметрів комп'ютерних вірусів вважаються вирішеними [15, 24, 50, 62]. Також в даній дисертаційній роботі не розглядаються питання пов'язані з сигналізацією про розпізнані віруси.

Відповідно до рекомендацій [8, 9, 68], перший етап вирішення сформульованого завдання асоційовано з розробкою концептуальної моделі забезпечення ефективного застосування НММ для розпізнавання комп'ютерних вірусів. В даному випадку під поняттям концептуальної моделі будемо розуміти змістовну модель, при формулюванні якої використовуються

поняття і уявлення в області захисту інформації. Слід відзначити, що саме створення концептуальної (базової) моделі є першим етапом розвитку методологічної бази, що репрезентує собою систему принципів і способів організації та побудови теоретичної і практичної діяльності, а також вчення про цю систему [58, 95].

В зв'язку з тим, що очікуваним практичним результатом дисертаційної роботи являється програмно-апаратний комплекс розпізнавання комп'ютерних вірусів, то при побудові концептуальної моделі використано термінологію не тільки із області захисту інформації, але й із області теорії нейронних мереж, програмної та комп'ютерної інженерії.

Також проведено гармонізацію термінології, що використовується в області нейромережевого розпізнавання комп'ютерних вірусів. Запропоновано визначення наступних термінів:

- Діагностичний параметр - параметр, що використовується для розпізнавання комп'ютерного вірусу.

- Портрет комп’ютерного вірусу (ПКС) – множина діагностичних параметрів, що описують комп'ютерний вірус.

- Портрет сигнатури (ПС) – множина параметрів, що характеризують сигнатуру комп’ютерного вірусу.

- Портрет поведінки операційної системи (ППС) – множина параметрів, що характеризують події в операційній системі під час функціонування комп’ютерного вірусу.

- Портрет діагностичних параметрів - множина діагностичних параметрів, що використовується для розпізнавання виду ПЗ.

- Нейромережева модель (НММ) – опис нейронної мережі, що включає в себе метод навчання, структуру зв'язків, параметри штучного нейрону, спосіб розповсюдження сигналу.

- Глибока нейронна мережа (ГНМ) – нейромережева модель в якій кількість схованих шарів синаптичних зв'язків більша ніж два.

- Навчання НММ - процес настройки параметрів такої моделі, що реалізується на експериментальних даних (навчальних прикладах).

- Навчальний приклад (НП) - запис навчального набору даних, значення якого використовуються для обчислення корекції параметрів моделі.

У випадку маркованих даних НП представляє собою:

$$НП=[\boldsymbol{X},\boldsymbol{Y}], \tag{2.1}$$

де $\boldsymbol{X}=(x_1,x_2,\ldots,x_n)$ – заданий вектор вхідних значень НММ; $\boldsymbol{Y}=(y_1,y_2,\ldots,y_n)$ – вихідний вектор, який повинна сформувати навчена НММ в якості відгука при подачі на її вхід вектора $\boldsymbol{X}$.

При немаркованих даних до складу НП входить тільки вектор $\boldsymbol{X}$.

- Навчальна вибірка (НВ) - множина навальних прикладів НММ.

В контексті завдання даного дисертаційного дослідження концептуальна модель, призначена, перш за все, для формалізації причинно-наслідкових зв'язків, які властиві процесу розпізнавання комп'ютерних вірусів. Ще одним призначенням концептуальної моделі є визначення аспектів розпізнавання, що впливають на рівень захищеності сучасних ІС. Крім цього, в концептуальної моделі враховано:

- Умови функціонування НМЗ розпізнавання комп'ютерних вірусів, що визначаються характером взаємодії окремих частин засобів розпізнавання і компонентами ІС.

- Необхідність реалізації ефективного використання НММ для розпізнавання комп'ютерних вірусів і основні напрямки підвищення ефективності їх функціонування.

- Можливість управління НМЗ і визначення їх налаштувань.

Також в результаті досліджень [41] визначено множину термінів, що використовуються для опису параметрів, котрі можуть використовуватись в НМЗ розпізнавання комп'ютерних вірусів:

− API (application programming interface/інтерфейс прикладного програмування) ОС – множина класів, функцій, процедур, структур та констант ОС, що є доступними для використання ШПЗ.

− Виклик API-функції - сигнатура виклику ШПЗ API-функції.

− Сигнатура функції - це частина загального оголошення функції, яка дозволяє засобам трансляції виконувати ідентифікацію цієї самої функції серед інших. До сигнатури виклику API-функції ШПЗ в першу чергу відносяться ім'я функції, що використовується ШПЗ та послідовність типів її аргументів.

− Сигнатура вірусу/ШПЗ - характерні формалізовані ознаки коду комп'ютерного вірусу/ШПЗ, що можуть бути використані для його розпізнавання. В якості сигнатур скриптового ШПЗ можуть бути використані множини операторів програмного коду. В якості сигнатур файлів, що виконуються використовуються послідовності байт (хеш-сканів цих послідовностей), що властиві певному типу ШПЗ. Також в якості сигнатур можуть бути використані виклики потенційно небезпечних API-функції, знайдені в коді ПЗ.

− Байт-послідовність N-грамів – неперервна послідовність кількістю в N байт. Розрізняють байт-послідовності N-грамів, характерних для ШПЗ та для безпечного ПЗ. Байт-послідовності N-грамів в основному використовуються для розпізнавання ШПЗ на основі статистичного аналізу коду піддослідного ПЗ.

− Опкод (operation code) – інструкція машинного коду, що ідентифікує операцію, яка має бути виконана.

− Бінарне двовимірне представлення програмного коду - представлення програмного коду у вигляді двовимірно чорно-білого зображення;

− Параметри PE-заголовку файлу – службова інформація, що міститься в PE-заголовку та описує різні властивості файлу і його структуру. Структура заголовків складається із наступних частин: заголовок DOS, заглушка DOS, заголовок PE, заголовки секцій.

На наступному етапі побудови концептуальної моделі з урахуванням загальноприйнятої технології використання НММ визначено, що в узагальненому вигляді процес забезпечення нейромережевого розпізнавання комп'ютерних вірусів повинен передбачати:

- Визначення параметрів навчальних прикладів.
- Формування навчальної вибірки.
- Визначення виду і параметрів НММ.
- Використання НММ для розпізнавання.

Діаграма декомпозиції застосування НММ для розпізнавання комп'ютерних вірусів показана на рис.2.1.

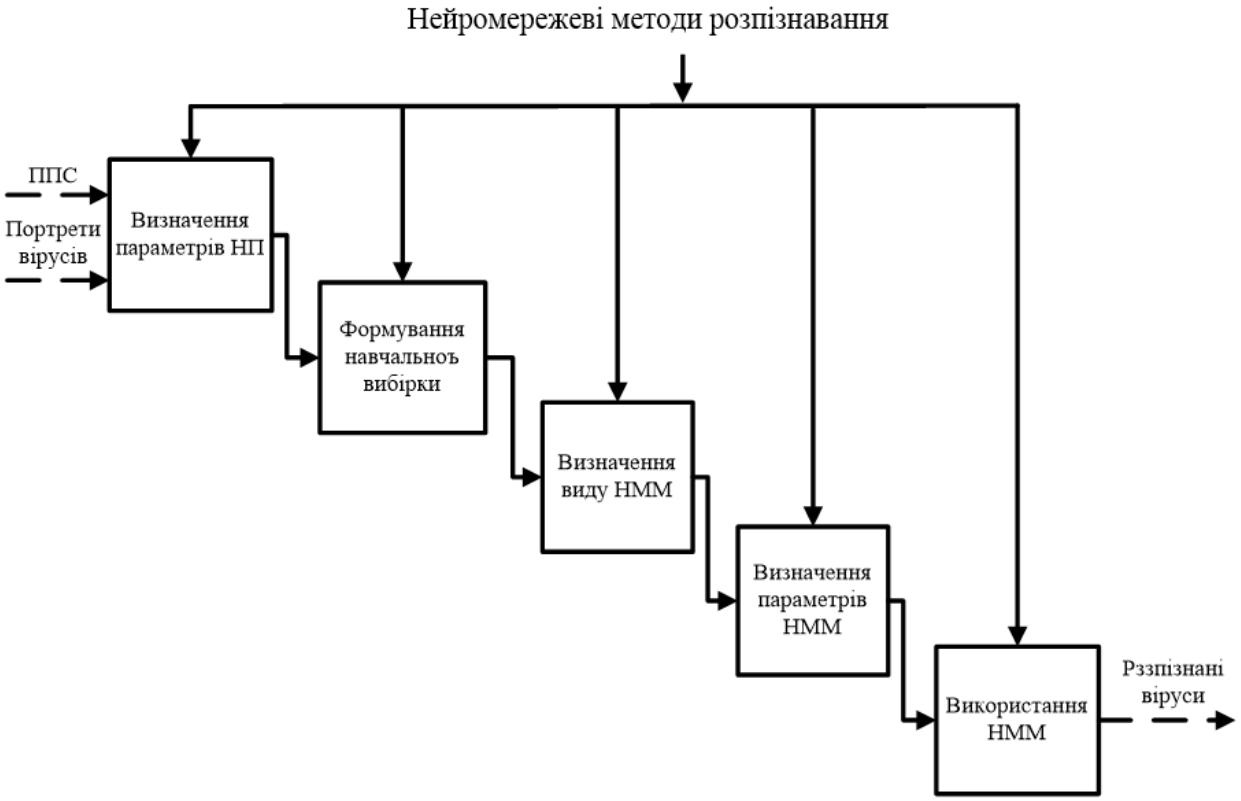

Рис. 2.1. Діаграма декомпозиції нейромережевого розпізнавання комп'ютерних вірусів

Коротка характеристика складових даної діаграми:

- Формування параметрів навчальних прикладів - визначення для кожного виду комп'ютерних вірусів множини вхідних і вихідних параметрів і способу їх кодування.

- Формування навчальної вибірки - визначення множини навчальних прикладів, що достатньо якісно описують портрети комп'ютерних вірусів. Кількість, якість і номенклатура прикладів повинні бути достатніми для навчання НММ.

- Визначення виду і параметрів НММ - визначення виду і параметрів НММ, при яких модель найбільш повно відповідає умовам поставленої задачі розпізнавання комп'ютерних вірусів.

- Використання НММ - розпізнавання комп'ютерних вірусів.

Наступним етапом створення концептуальної моделі стала розробка показаної на рис. 2.2 схеми компонентів НМС розпізнавання комп'ютерних вірусів.

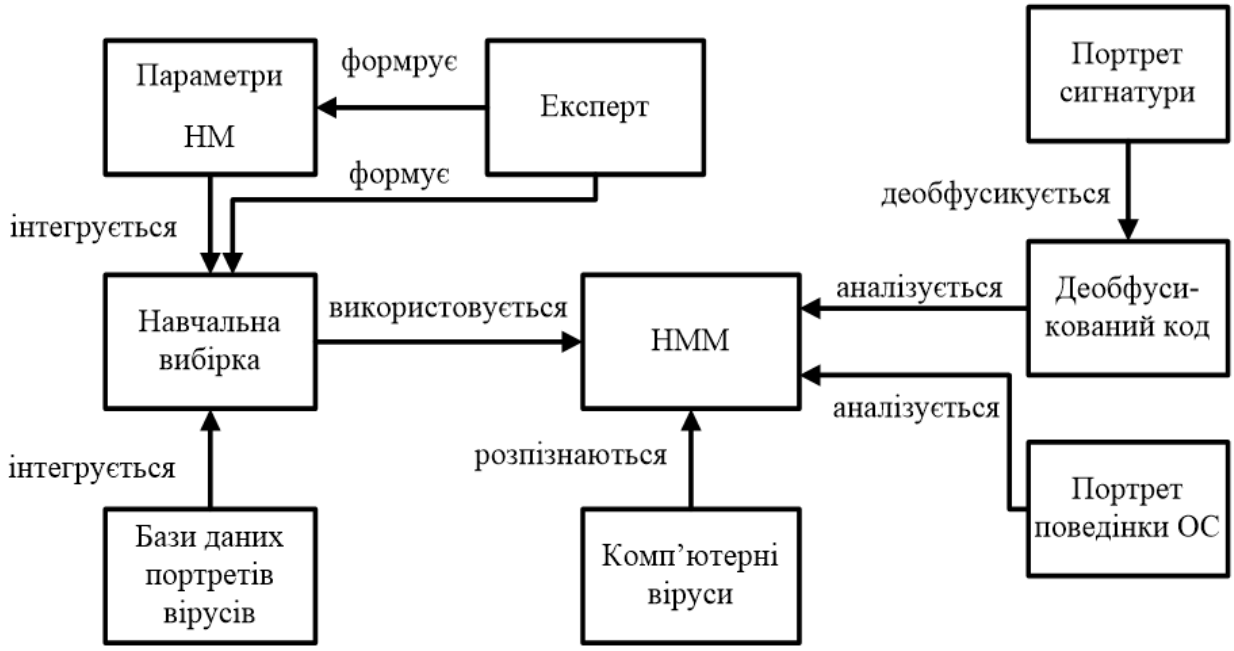

Рис. 2.2. Схема взаємодії компонентів нейромережевої системи розпізнавання комп'ютерних вірусів

У схемі враховані особливості реалізації НМС, призначені для

розпізнавання обфускованих комп'ютерних вірусів, і результати розділу 1, які стосуються недоліків відомих НМС аналогічного призначення.

Таким чином, в процесі розробки враховано:

- Недосконаліть методів формування параметрів навчальних прикладів для НММ, призначених для розпізнавання кібератак на мережеві РІС.

- Тривалий період формування навчальної вибірки для НММ в разі обмеженого доступу до баз даних портретів комп'ютерних вірусів.

- Складність доступу до існуючих баз даних портретів комп'ютерних вірусів.

- Необхідність деобфускації програмного коду.

Тому в схемі передбачена можливість формування параметрів навчальних прикладів і навчальної вибірки за допомогою експертних даних. Аналіз даних, показаних на рис. 2.1 і рис. 2.2, дозволяє стверджувати, що на ефективність нейромережевого розпізнавання комп'ютерних вірусів впливають ряд факторів, показаних на рис. 2.3.

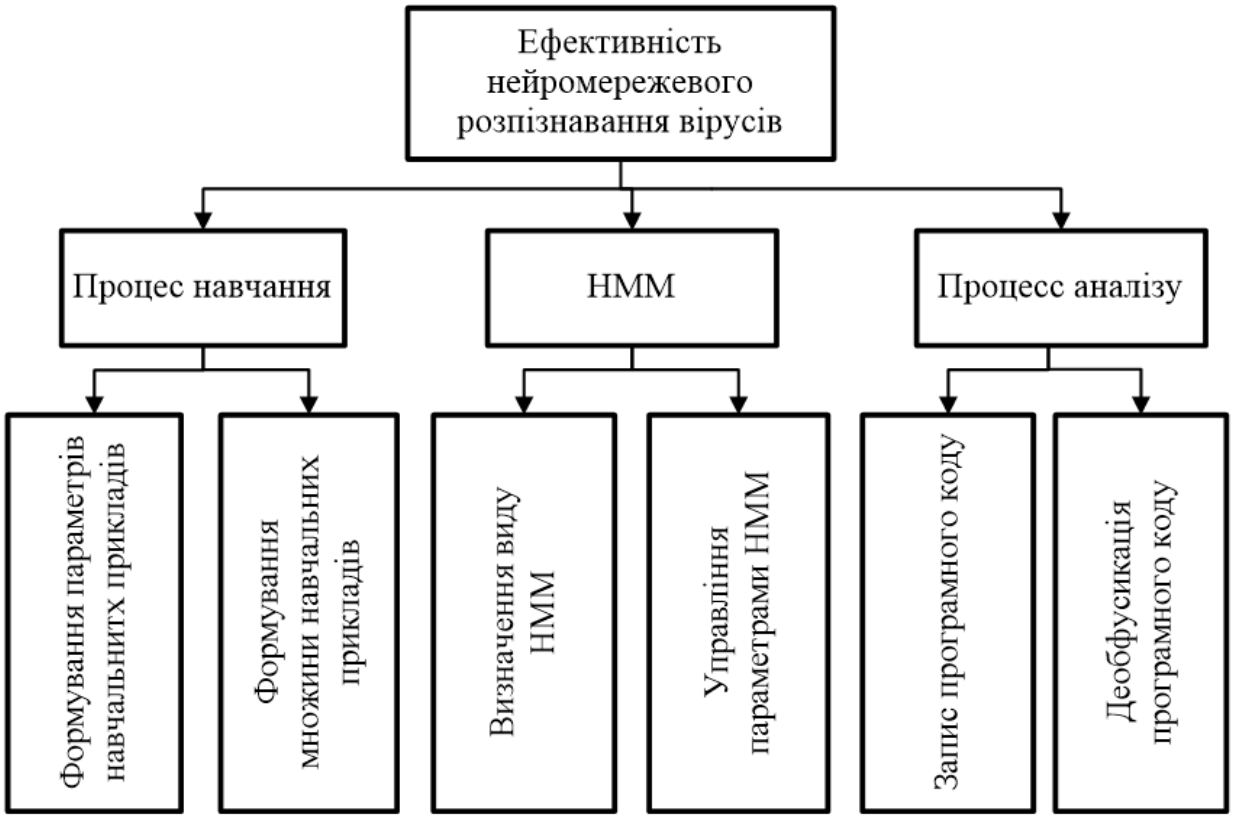

Рис. 2.3. Фактори, що впливають на ефективність розпізнавання

Крім цього, можна стверджувати, що ефективність нейромережевого розпізнавання доцільно оцінювати як з точки зору ефективності процесу використання НМЗ, так і з точки зору ефективності процесу навчання НММ. При цьому показники ефективності повинні відображати тривалість, ресурсомісткість і точність названих процесів. Таким чином, обґрунтовані показання на рис. 2.4 показники оцінки ефективності нейромережевого розпізнавання комп'ютерних вірусів.

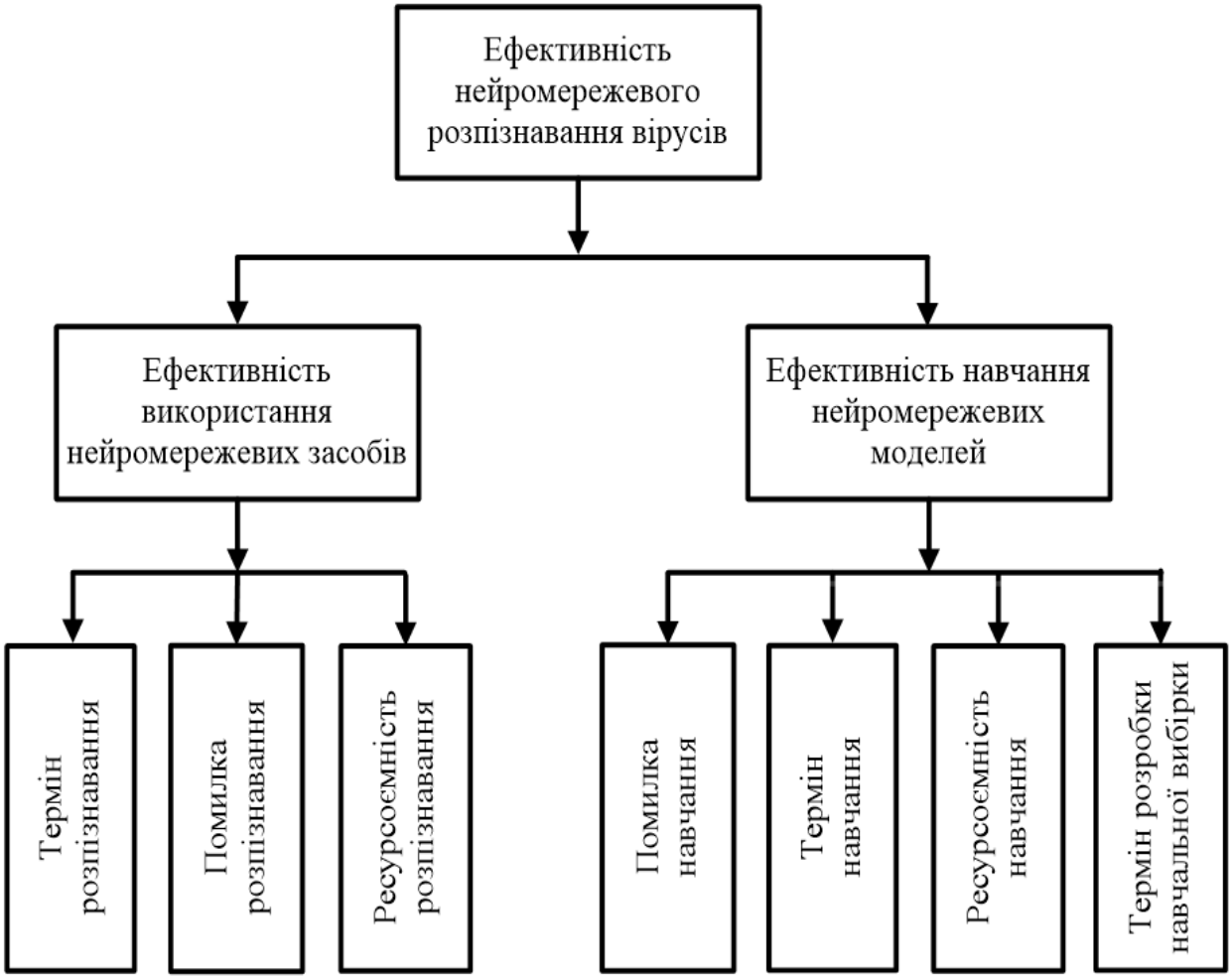

Рис. 2.4. Показники оцінки ефективності нейромережевого розпізнавання

У формалізованій постановці інтегральну ефективність процесу нейромережевого розпізнавання комп'ютерних вірусів можливо описати за допомогою наступних параметрів:

-$E_{НСР}$– ефективність розробки та використання НМЗ;

-$E_{\text{ОВ}}$– ефективність розробки навчальної вибірки;

- e_1– визначення ефективних видів НММ;

- e_2– визначення параметрів НММ;

- e_3– ресурсоємність використання НМЗ;

- e_4– визначення параметрів навчальних прикладів;

- e_5– деобфускація програмного коду.

В результаті визначено, що в аналітичному вигляді концептуальну модель забезпечення ефективності процесу нейромережевого розпізнавання комп'ютерних вірусів можна відобразити за допомогою виразів:

$$E_{\Sigma} = f(E_{\text{НСР}}, E_{\text{ОВ}}), \tag{2.2}$$

$$E_{\text{НСР}} = f\left(e_1, e_2, e_3\right), \tag{2.3}$$

$$E_{\text{ОВ}} = f(e_4, e_5), \tag{2.4}$$

де E_{Σ}– інтегральна ефективність процесу.

Аналіз розробленої концептуальної моделі дозволяє стверджувати, що для ефективного нейромережевого розпізнавання комп'ютерних вірусів необхідно доповнити методологічну базу рядом принципів і моделей процесів використання НМЗ. Оскільки в результаті аналізу сучасних НММ визначено перспективність використання ГНМ, то в методологічній базі слід акцентувати увагу саме на цей вид моделей.

Зазначимо, що особливості ГНМ в значній мірі залежать від структури та математичного забезпечення НММ, що в свою чергу визначається видом цієї моделі. Надалі необхідно застосувати отримані елементи методологічної бази для розробки нейромережевих моделей і методів розпізнавання комп'ютерних вірусів.

2.2. Принципи використання глибоких нейронних мереж

В основу розробки принципів використання ГНМ для розпізнавання комп'ютерних вірусів покладені результати роботи [88], присвяченої використанню НММ для розпізнавання мережевих кібератак та роботи [76] в якій НММ застосовані для аналізу голосового сигналу. Базуючись на вказаних дослідженнях передбачено доповнення методологічної бази принципами, що відображають:

- Допустимість використання виду ГНМ для розпізнавання комп’ютерних вірусів.

- Визначення множини ефективних видів ГНМ для розпізнавання комп’ютерних вірусів.

- Оцінювання ефективності виду ГНМ.

- Представлення процесу функціонування програмного забезпечення у вигляді графа залежностей значень та станів.

- Оцінювання безпечності програмного забезпечення за допомогою графів залежностей значень та станів.

Принцип допустимості використання виду ГНМ для розпізнавання комп’ютерних вірусів, полягає в тому, що серед множини доступних деякий вид ГНМ входить до множини допустимих видів, якщо його основні характеристики задовольняють вимогам щодо допустимого терміну і допустимої ресурсоємності побудови НМЗ.

У формалізованому вигляді вказаний принцип передбачає використання наступних компонент:

- **DNN** – множина доступних видів ГНМ.

- DNN_i - *i*-ий доступний вид ГНМ.

-$\mathrm{DNN}_{\mathrm{avl}}$- множина допустимих видів ГНМ.

- τ_{avl} - допустимий термін розробки НМЗ.

- Q_{avl} - допустима ресурсоємність побудови НМЗ.

- Q(**DNN$_i$**) – ресурсоємність побудови НМЗ на основі *i*-го доступного виду ГНМ.

- τ(**DNN$_i$**) – термін розробки НМЗ на основі *i*-го доступного виду ГНМ.

В результаті проведеної формалізації запропонований принцип допустимості використання виду ГНМ можливо записати у вигляді наступного виразу:

$$\text{if}(Q(\text{DNN}_i) \le Q_{\text{avl}}) \wedge (\tau(\text{DNN}_i) \le \tau_{\text{avl}}) \rightarrow \text{DNN}_i \in \text{DNN}_{\text{avl}}, \quad (2.5)$$

Принцип визначення множини ефективних видів ГНМ для розпізнавання комп'ютерних вірусів полягає у тому, що і-ий доступний вид ГНМ входить до множини ефективних видів, якщо для нього значення функції ефективності не менше допустимого значення.

Формалізація принципу визначення множини ефективних видів ГНМ передбачає використання наступних компонент:

- $V(\text{DNN}_i)$ - значення функції ефективності для *i*-го доступного виду ГНМ.

- V_d – мінімально допустиме значення функції ефективності.

- DNN_{eff} - множина ефективних видів ГНМ.

Таким чином, запропонований принцип визначення множини ефективних видів ГНМ можливо записати у вигляді наступного виразу:

$$\text{if}\, V(\text{DNN}_i) \le V_d \rightarrow \text{DNN}_i \in \text{DNN}_{e\text{ff}}, \quad (2.6)$$

При цьому в першому наближенні для розрахунку функції ефективності можливо використовувати вираз виду:

$$V(\text{DNN}_i) = \sum_{k=1}^{K} \alpha_k H_k(\text{DNN}_i), \quad (2.7)$$

де $H_k(\text{DNN}_i)$ – значення *k*-го критерію для ГНМ з *i*-ою архітектурою, $\alpha_k = [0.., 1]$– ваговий коефіцієнт *k*-го критерію ефективності, *K* – кількість критеріїв.

Зазначимо, що вираз (2.7) використовується в базовому випадку, в подальшому процедура розрахунку функції ефективності може бути вдосконалена з урахуванням результатів [9].

Принцип оцінювання ефективності виду ГНМ, призначеної для розпізнавання комп'ютерних вірусів, полягає в тому, що серед множини допустимих i-ий вид ГНМ ($\mathbf{DNN}_i$) є найбільш ефективним, якщо для нього функція ефективності (V_i) має максимальне значення. Як і попередні, принцип оцінювання ефективності виду ГНМ можливо записати у вигляді наступного виразу:

$$\max_{V_i} = \{V_1, V_2, .. V_I\}. \quad (2.8)$$

Принцип представлення процесу функціонування програмного забезпечення у вигляді графа залежностей значень та станів полягає в тому, що поведінка програмного забезпечення представляється за допомогою спеціалізованого графа залежностей значень та станів.

У формалізованому вигляді вказаний принцип передбачає використання наступних компонент:

- F – множина, що описує дії програмного забезпечення.

- T – множина переходів.

- S – множина станів графу, що описує поведінку програмного забезпечення.

В результаті формалізації означений принцип представлення процесу функціонування програмного забезпечення у вигляді графа залежностей значень та станів набуває такого вигляду:

$$\mathbf{F} = < \mathbf{T}, \mathbf{S} > . \quad (2.9)$$

Принцип оцінювання безпечності програмного забезпечення за допомогою графів залежностей значень та станів полягає в тому, що

безпечність програмного забезпечення можливо оцінити за рахунок порівняння графа залежностей значень та станів піддослідної програми з відповідними графами безпечних програм та графами комп'ютерних вірусів.

У формалізованому вигляді принцип оцінювання безпечності програмного забезпечення за допомогою графів залежностей значень та станів можливо записати у вигляді наступних виразів:

$$\text{if} F_x = F_s \rightarrow x \in S, \tag{2.10}$$

$$\text{if} F_x = F_v \rightarrow x \in V, \tag{2.11}$$

де F_x– граф піддослідного програмного забезпечення, F_s – множина графів безпечних програм, F_v – множина графів комп'ютерних вірусів, x – піддослідне програмне забезпечення, S – множина безпечних програм, V– множина комп'ютерних вірусів.

2.3. Правила визначення ефективних видів глибоких нейронних мереж

Запропоновані правила визначення ефективних видів ГНМ базуються на розробленому принципі допустимості застосування виду та на принципі визначення множини ефективних видів ГНМ.

Основними складовими вказаної моделі є

- DNN_{ent} – множина доступних видів ГНМ.
- DNN_{avl}– множина допустимих видів ГНМ.
- DNN_{eff}– множина ефективних видів ГНМ.

Процес визначення множини ефективних видів ГНМ можливо представити у вигляді:

$$DNN_{ent} \rightarrow DNN_{avl} \rightarrow DNN_{eff}. \tag{2.12}$$

Як показують результати [59] до складу множини доступних видів ГНМ входять:

– dnn_1 - повнозв’язні ГНМ, при навчанні яких процедура переднавчання не передбачена;

– dnn_2 - повнозв’язні ГНМ, при навчанні яких використовується процедура переднавчання;

– dnn_3 - ГНМ типу ЗНМ з прямим поширенням сигналу;

– dnn_4 - ГНМ типу рекурентних ЗНМ.

При цьому множина доступних видів ГНМ записується так:

$$DNN_{ent} = \{dnn_1, dnn_2, dnn_3, dnn_4\}. \quad (2.13)$$

Особливості структурних рішень доступних видів ГНМ показані на рис. 1.3.4-1.3.8. Так на рис. 1.3.4 показана узагальнена структурна схема повнозв’язної ГНМ, при навчанні яких процедура переднавчання не передбачена. По суті вказані ГНМ являються різновидом класичного багатошарового персептрону у якого кількість схованих шарів нейронів більша за одиницю. Вхідні нейрони такої нейронної мережі асоціюються з параметрами, що використовуються для розпізнавання комп'ютерних вірусів. Це можуть бути назви потенційно небезпечних API-функцій або характерні фрагменти програмного коду. На рис. 1.3.4 кількість таких параметрів дорівнює N_x. Вихідні нейрони мережі асоціюються з безпечним програмним забезпеченням та з назвами вірусів, що мають бути розпізнаними. Так, показана на рис. 1.3.4 ГНМ, призначена для розпізнавання (N_y-1) вірусів.

В задачах розпізнавання ШПЗ перевагами повнозв'язних ГНМ в яких процедура переднавчання не передбачена є відносна простота, апробованість та надійність. До недоліків відносять:

- Неможливість аналізу динамічних рядів даних.

- Неможливість врахувати топологію даних, що призводить до зменшення обчислювальної потужності мережі.

- Необхідність використання великого обсягу маркованих навчальних прикладів, що призводить до значного ускладнення навчальної вибірки.

Одним із шляхів усунення останнього недоліку є використання повнозв'язних ГНМ, при навчанні яких використовується процедура переднавчання. Структура та математичне забезпечення таких НММ не відрізняється від dnn$_1$. Відмінності полягають лише у процедурі навчання, що розділяється на два етапи. На першому етапі, що носить назву переднавчання ГНМ навчається на не маркованих прикладах.

Для цього, як правило використовується так званий автоендкодер, структура якого показана на рис. 1.3.8.

При розпізнаванні ШПЗ вхідними даними автоендкодера є немарковані навчальні приклади, параметри яких були визначені при формуванні концептуальної моделі. Таким чином, навчальні приклади автоендкодера, що призначений для використання в антивірусних засобах, визначається виразом виду:

$$\boldsymbol{x}=(x_1, x_2,\dots x_n), \tag{2.17}$$

де n – кількість вхідних нейронів, $x_1, x_2,\dots x_n$ – діагностичні параметри.

В цьому випадку вихідний сигнал автокодувальника в якому кількість схованих шарів нейронів дорівнює q розраховується так:

$$f_{W,b}(x) = a^{(q)}, \tag{2.18}$$

де W – масив вагових коефіцієнтів, b – масив зсувів, $a^{(q)}$ – масив вихідних значень нейронів в останньому q-му шарі.

На рис 1.3.8 вихідний сигнал автокодувальника відповідає вектору ($y_1, y_1, \dots y_n$), тобто вектору назв ШПЗ, що має бути розпізнане. Таким чином, використання автоендкодера дозволяє проводити навчання ГНМ на немаркованих прикладах, що відповідають портретам діагностичних

параметрів.

На другому етапі навчання ГНМ типу dnn_2 навчається на маркованих даних. За рахунок двоетапного навчання необхідна для якісного навчання кількість маркованих прикладів значно зменшується. Так відповідно результатів [49] у випадку розпізнавання комп'ютерних вірусів необхідна кількість маркованих навчальних прикладів для ГНМ типу dnn_1 становить:

$$P \approx 10 \times N. \tag{2.20}$$

де P – кількість маркованих навчальних прикладів; N – кількість вхідних сигналів НММ.

Разом з тим для ГНМ типу dnn_2 необхідну кількість навчальних прикладів можливо оцінити так:

$$P_1 \approx 10 \times N, \tag{2.21}$$

$$P_2 \approx 0{,}1 \times P_1, \tag{2.22}$$

де P_1 – кількість не маркованих навчальних прикладів; P_2 – кількість маркованих навчальних прикладів.

Таким чином, аналіз виразів (2.20-2.22) дозволяє стверджувати про те, що використання автоендкодера дозволяє до 10 разів зменшити кількість маркованих навчальних прикладів, що необхідні для навчання. Разом з тим, в джерелах [74-75] вказується, що використання автоендкодера може значно ускладнити процес навчання ГНМ. Використання ЗНМ, спричинене можливістю врахування НММ топології вхідних параметрів, що дозволяє в значній мірі зменшити кількість синаптичних зв'язків моделі, а відповідно і зменшити її ресурсоємність. Базовою конструкцією ЗНМ є НММ типу LeNet-5, структурна схема якої показана на рис. 1.3.5. Особливістю застосування ЗНМ в антивірусних засобах є те, що діагностичні параметри, котрі

використовуються в якості вхідних параметрів моделі, повинні бути представлені у вигляді квадратного чорно-білого, сірого або кольорового зображення. Як правило, вхідні параметри ЗНМ відповідають окремим пікселям. Тому кількість вхідних параметрів дорівнює розміру зображення. Стосовно розпізнавання комп'ютерних вірусів у випадку використання ЗНМ у антивірусному моніторі окремий піксель може відповідати одній із API-функцій операційної системи. При цьому вісь абцис буде відповідати переліку API-функцій, що підлягають аналізу. Вісь ординат може бути співвіднесена з часовим інтервалом розпізнавання.

В загальному випадку кількість вихідних нейронів дорівнює кількості типів ПЗ, що мають бути розпізнані.

Дещо спростивши задачу розпізнавання комп'ютерних вірусів можливо стверджувати, що кількість вихідних нейронів на одиницю більша від кількості типів комп'ютерних вірусів, що підлягають розпізнаванню.

Зазначимо, що адаптація ЗНМ до умов задачі розпізнавання комп'ютерних вірусів може бути реалізована за рахунок основних конструктивних НММ даного типу. На підставі теоретичних робіт, присвячених ЗНМ [77-78] можна стверджувати, що такими конструктивними параметрами є:

- Розмір вхідного поля, що відповідає вхідному полю діагностичних параметрів.

- Кількість вхідних нейронів, що відповідають кількості діагностичних параметрів.

- Кількість вихідних нейронів, що дорівнює кількості типів ПЗ (легітимного та шкідливого), яке має бути розпізнане.

- Кількість нейронів в повнозв'язному шарі.

- Кількість шарів згортки.

- Кількість карт ознак в кожному шарі згортки.

- Кількість шарів підвибірки.

- Масштабний коефіцієнт для кожного шару підвибірки.

- Розмір ядра згортки для кожного шару згортки.

- Зміщення рецептивного поля при виконанні кожної процедури згортки.

- Розмір карти ознак для кожного шару згортки.

- Структура зв'язків між сусідніми шарами згортки/підвибірки.

З урахуванням необхідності мінімізації помилки розпізнавання комп'ютерних вірусів модель оптимізації структурних параметрів ЗНМ можна записати за допомогою виразу:

$$\begin{cases} \Delta\left(L_{in}, L_{ls}, L_{out}, K_{h,k}, b_k, K_{ls}, \left|\mathbf{Q_{i,i+1}}\right|_{K_{ls}}\right) \to \min \\ R \le R_{\max} \end{cases} \qquad (2.25)$$

де Δ - помилка розпізнавання, $\left|\mathbf{Q_{i,i+1}}\right|_{K_{ls}}$ - вектор, що складається з матриць які визначають зв'язки між сусідніми прихованими шарами нейронів, R - ресурсоємність мережі, R_{max} - максимально допустима ресурсоємність мережі, L_{in} - кількість вхідних нейронів, L_{ls} - кількість нейронів в повнозв'язному шарі, L_{out} - кількість вихідних нейронів, $K_{h,k}$ – кількість карт ознак в кожному із шарів згортки, b_k – розмір кожного із ядер згортки, K_{ls} – кількість шарів згортки.

До труднощів застосування ЗНМ в поставленій дисертаційній задачі слід віднести складність визначення наявності топології у вхідних даних. Ще одним недоліком ЗНМ є складність аналізу динамічних рядів даних, адже кількість вхідних параметрів ЗНМ так само, як і ГНМ, є фіксованою величиною. При цьому зміну значень діагностичних параметрів в часі логічно представити у вигляді динамічного ряду даних. Для виправлення останнього недоліку доцільно використати рекурентну ЗНМ, структурні рішення якої показані на рис. 1.3.6, 1.3.7.

Проведені дослідження особливостей доступних видів ГНМ дозволяють стверджувати, що в першому наближенні допустимість їх використання можливо оцінити з позицій допустимого терміну розробки відповідної НММ. При цьому термін розробки НММ можливо визначити за допомогою виразу:

$$t_{dnn}=t_{ds}+t_m, \quad (2.26)$$

де t_{dnn} - термін розробки ГНМ; t_{ds}- термін формування навчальної вибірки; t_m - термін розробки параметрів ГНМ.

Значення t_{dnn}, t_{ds} та t_m вимірюються в секундах. Також секундах вимірюються і складові цих термінів, що будуть показані в подальших розрахунках.

Базуючись на результатах [16] визначено, що розробка параметрів ГНМ в основному зводиться до навчання НММ. Тому в базовому варіанті термін розробки параметрів ГНМ співвідноситься із терміном навчання мережі.

Разом з тим при оціночних розрахунках вважається, що формування навчальної вибірки зводиться до розробки навчальних прикладів кількість яких є мінімально достатньою для навчання ГНМ. Вважається, що ця кількість залежить від кількості вхідних і вихідних параметрів НММ і визначається виразом:

$$L=20N_x N_y, \quad (2.27)$$

де L – кількість навчальних прикладів, N_x – кількість вхідних нейронів ГНМ, N_y – кількість вихідних нейронів ГНМ.

Враховуючи, що термін формування одного навчального прикладу є відносно сталою величиною, термін формування навчальної вибірки можна розрахувати так:

$$t_{ds} = \vartheta L, \quad (2.28)$$

де ϑ - термін формування одного навчального прикладу.

Враховуючи (2.27) отримаємо:

$$t_{ds} = 20\vartheta N_x N_y. \quad (2.29)$$

Як свідчать результати теоретичних робіт [36] в загальному випадку для оцінки тривалості навчання НММ доцільно використовувати вираз виду:

$$t_s = \lambda \times N \times W \times I, \tag{2.30}$$

де t_s - тривалість навчання; λ - тривалість однієї навчальної ітерації; N – кількість нейронів; W - кількість синаптичних зв'язків; I - кількість навчальних ітерацій.

Зазначимо, що при використанні загальнопоширених методів навчання, що базуються на алгоритмі оберненого розповсюдження помилки достатню кількість навчальних ітерацій можливо оцінити так:

$$t_{s,dnn1} = k_{i,dnn} e^{-\varepsilon} L^2, \tag{2.31}$$

де ε - допустима похибка навчання; $k_{i,dnn}$ – коефіцієнт пропорційності.

Як показують дані [32] для повнозв'язних ГНМ кількість синаптичних зв'язків можливо оцінити так:

$$W_{dnn1,2} \approx k_{dnn1,2}\left(N_x + N_y\right)^2, \tag{2.32}$$

де $k_{dnn1,2}$ - коефіцієнт пропорційності, $W_{dnn1,2}$ - кількість синаптичних зв'язків для ГНМ типу dnn_1 та dnn_2.

За рахунок застосування (2.31, 2.30) для повнозв'язних ГНМ в яких процедура переднавчання не передбачена вираз (2.30) видозмінюється так:

$$t_{s,dnn1} = k_{dnn1} \lambda e^{-\varepsilon} L^2 \left(N_x + N_y\right)^2, \tag{2.33}$$

де $t_{s,dnn1}$- тривалість навчання для ГНМ типу dnn_1; k_{dnn1}- коефіцієнт пропорційності для ГНМ типу dnn_1.

Для ГНМ типу dnn_2 змінюється лише коефіцієнт пропорційності. Тобто:

$$t_{s,dnn2} = k_{dnn2}\lambda e^{-\varepsilon} L^2 \left(N_x + N_y\right)^2, \quad (2.34)$$

де $t_{s,dnn2}$- тривалість навчання для ГНМ типу dnn_1; k_{dnn2}- коефіцієнт пропорційності для ГНМ типу dnn_2.

Зазначимо, що відповідно даних [38] тривалість навчання ГНМ на одному маркованому навчальному прикладі можливо вважати рівною тривалості навчання на одному немаркованому навчальному прикладі.

При визначенні орієнтовного терміну навчання ЗНМ можливо відштовхуватись від того, що по суті ЗНМ є модифікацією ГНМ адаптованою до розпізнавання образів в яких важливо враховувати топологію даних. Ця відмінність відображається наприклад на основі порівняння рис. 2.6 та рис. 2.7 на яких відображені структурні схеми dnn_1 та dnn_3.

Разом з тим порівняльний аналіз математичного забезпечення цих типів НММ [102] дозволяє стверджувати, що урахування відмінностей в топології основна особливість процесу навчання dnn_3 полягає у необхідності проходу ядра згортки по вхідному полю ЗНМ, по всім картам згортки та по всім картам ознак. В першому наближенні тривалість таких проходів можна розрахувати так:

$$t_{CNN,dnn2} = k_{CNN,dnn2} N_x^2, \quad (2.35)$$

де $k_{CNN,dnn2}$ – коефіцієнт пропорційності.

Зазначимо, що врахування топології даних дозволяє значно зменшити кількість вагових коефіцієнтів мережі. За рахунок цього можливо вважати, що кількість вагових коефіцієнтів для ЗНМ прямо пропорційна сумі кількості вхідних та вихідних нейронів. Тобто для ЗНМ з прямим розповсюдженням сигналу кількість синаптичних зв'язків можливо оцінити так:

$$W_{dnn3} \approx k_{i,dnn3}\left(N_x + N_y\right), \quad (2.36)$$

де $k_{i,dnn3}$ - коефіцієнт пропорційності, W_{dnn32} - кількість синаптичних зв'язків для ГНМ типу dnn_3.

Інтегрувавши вирази (2.30-2.32, 2.35, 2.36) отримаємо:

$$t_{s,dnn3} = k_{i,dnn3}\lambda e^{-\varepsilon}L^2\left(N_x + N_y\right)k_{CNN,dnn2}N_x^2, \quad (2.37)$$

де $t_{s,dnn3}$- тривалість навчання для ГНМ типу dnn_3; - допустима похибка навчання.

Після відповідних спрощень отримаємо:

$$t_{s,dnn3} = k_{dnn3}\lambda e^{-\varepsilon}L^2\left(N_x + N_y\right)N_x^2, \quad (2.38)$$

де k_{dnn3}- коефіцієнт пропорційності для ГНМ типу dnn_3.

При оцінці терміну навчання рекурентних ЗНМ використано два постулати:

- Як свідчать дані [1-10] та порівняння структурних схем, показаних на рис. 2.7-2.9, рекурентну ЗНМ можливо подати у вигляді розгорнутої в часі ЗНМ.

- Глибину розгортання можливо оцінити за допомогою емпіричного коефіцієнту.

Використання вказаних постулатів дозволило для визначення терміну навчання рекурентних ЗНМ використати вираз (2.38), видозмінивши його так:

$$t_{s,dnn4} = k_{dnn3}k_{CNN,dnn4}\lambda e^{-\varepsilon}L^2\left(N_x + N_y\right)N_x^2, \quad (2.39)$$

де $t_{s,dnn4}$- тривалість навчання для ГНМ типу dnn_4; $k_{CNN,dnn4}$ – коефіцієнт глибини рекурсії.

Після деяких спрощень отримаємо:

$$t_{s,dnn4} = k_{dnn4}\lambda e^{-\varepsilon}L^2\left(N_x + N_y\right)N_x^2, \quad (2.40)$$

де k_{dnn4}- коефіцієнт пропорційності для ГНМ типу dnn_4.

Таким чином, вирази (2.33, 2.34, 2.38, 2.40) дозволяють оцінити орієнтовний термін навчання для доступних видів ГНМ, що дозволило перейти до визначення загального терміну їх побудови.

Для визначення загального терміну побудови повнозв'язних ГНМ, при навчанні яких процедура переднавчання не передбачена в вираз (2.26) підставлені вирази (2.29) та (2.33). Отримано:

$$t_{dnn1} = 20N_xN_y + k_{dnn1}\lambda e^{-\varepsilon}L^2\left(N_x + N_y\right)^2, \quad (2.41)$$

де t_{dnn1} – термін побудови ГНМ типу dnn_1.

Після тривіальних спрощень отримано:

$$t_{dnn1} = k_{dnn1}\lambda e^{-\varepsilon}L^2\left(N_x^2 + N_y^2\right) + 2N_xN_y(10\vartheta_m + k_{dnn1}\lambda e^{-\varepsilon}L^2), \quad (2.42)$$

де ϑ_m – термін створення одного маркованого прикладу.

Для визначення загального терміну побудови повнозв'язних ГНМ, при навчанні яких використовується процедура переднавчання врахована необхідність створення як маркованих, так і немаркованих прикладів.

При цьому:

$$\vartheta_{nm} \approx 0{,}01\vartheta_m, \quad (2.43)$$

$$L_{nm} \approx 10L_m, \quad (2.44)$$

$$L_{dnn2} = L_{nm} + L_m, \quad (2.45)$$

Тобто термін створення одного немаркованого прикладу в 100 разів

коротший ніж термін створення одного маркованого прикладу. Кількість немаркованих прикладів як правило мінімум в 10 разів перевищує кількість маркованих прикладів. Підставивши (2.44) в (2.45) отримано:

$$L_{dnn2} = 11L_m. \tag{2.46}$$

Після підстановки (2.46) в (2.28) визначено, що термін створення навчальної вибірки для ГНМ типу dnn_2 становить:

$$t_{dnn2,ds} = 1100\vartheta_{nm}L_m. \tag{2.47}$$

Зазначимо, що для гібридної навчальної вибірки мінімально достатня кількість навчальних прикладів розраховується так:

$$L_{dnn2} = 50N_xN_y. \tag{2.48}$$

Враховуючи (2.45) отримано:

$$L_m \approx 4{,}54N_xN_y. \tag{2.49}$$

Після підстановки (2.49) в (2.47) отримано вираз (2.50), що може бути використаний для остаточного розрахунку терміну формування навчальної вибірки для повнозв'язних ГНМ, при навчанні яких використовується процедура переднавчання.

$$t_{dnn2,ds} = 5000\vartheta_{nm}N_xN_y. \tag{2.50}$$

Для визначення загального терміну побудови повнозв'язних ГНМ, при навчанні яких використовується процедура переднавчання не передбачена в вираз (2.26) підставлені вирази (2.50) та (2.34). Отримано:

$$t_{dnn1} = 5000\vartheta_m N_x N_y + k_{dnn2}\lambda e^{-\varepsilon} L^2 \left(N_x + N_y\right)^2, \quad (2.51)$$

де t_{dnn2} – термін побудови ГНМ типу dnn_2.

Після тривіальних спрощень (2.51) визначено:

$$t_{dnn2} = k_{dnn2}\lambda e^{-\varepsilon} L^2 \left(N_x^2 + N_y^2\right) + 2N_x N_y (25000\vartheta_{nm} + k_{dnn2}\lambda e^{-\varepsilon} L^2), \quad (2.52)$$

де ϑ_{nm} – термін створення одного не маркованого прикладу.

Для визначення загального терміну побудови ГНМ типу ЗНМ з прямим поширенням сигналу в вираз (2.26) підставлені вирази (2.29) та (2.38). В результаті підстановки визначено:

$$t_{dnn3} = 20N_x N_y + k_{dnn3}\lambda e^{-\varepsilon} L^2 \left(N_x + N_y\right) N_x^2, \quad (2.53)$$

де t_{dnn3} – термін побудови ГНМ типу dnn_3.

Після перетворень на спрощень отримано:

$$t_{dnn3} = 20\vartheta_m N_x N_y + k_{dnn3}\lambda e^{-\varepsilon} L^2 \left(N_x^3 + N_y N_x^2\right). \quad (2.54)$$

По аналогії з (2.54) отримано вираз для розрахунку загального терміну побудови рекурентних ЗНМ:

$$t_{dnn4} = 20\vartheta_m N_x N_y + k_{dnn4}\lambda e^{-\varepsilon} L^2 \left(N_x^3 + N_y N_x^2\right), \quad (2.55)$$

де t_{dnn4} – термін побудови ГНМ типу dnn_4.

Відзначимо, що вирази (2.54) та (2.55) відрізняються лише коефіцієнтами пропорційності k_{dnn}. Таким чином вирази (2.42, 2.52, 2.54, 2.55) визначають загальний термін побудови ГНМ типу dnn_1, dnn_2, dnn_3 та dnn_4 відповідно.

Зазначимо, що для dnn_1, dnn_3 та dnn_4 необхідна мінімальна кількість навчальних прикладів розраховується за допомогою виразу (2.27). В результаті підстановки (2.27) в (2.42, 2.54, 2.55) отримано:

$$t_{dnn1} = 400k_{dnn1}\lambda e^{-\varepsilon}N_x^2N_y^2\left(N_x + N_y\right)^2 + 20\vartheta_m N_x N_y, \quad (2.56)$$

$$t_{dnn3} = 400k_{dnn3}\lambda e^{-\varepsilon}N_x^2N_y^2\left(N_x^3 + N_yN_x^2\right) + 20\vartheta_m N_x N_y, \quad (2.57)$$

$$t_{dnn4} = 400k_{dnn4}\lambda e^{-\varepsilon}N_x^2N_y^2\left(N_x^3 + N_yN_x^2\right) + 20\vartheta_m N_x N_y. \quad (2.58)$$

Для ГНМ типу dnn_2 необхідна мінімальна кількість навчальних прикладів визначається виразами (2.48). В результаті підстановки (2.48) в (2.52) та деяких спрощень отримано:

$$t_{dnn2} = 2500\left(k_{dnn2}\lambda e^{-\varepsilon}N_x^2N_y^2\left(N_x + N_y\right)^2 + 2\vartheta_{nm}N_x N_y\right), \quad (2.59)$$

Базуючись на теоретичних роботах в області НМ [14] визначено, що в першому наближенні:

$$k_{dnn2} = k_{dnn2} = k_{dnn3} = k_{dnn4} = 0.001. \quad (2.60)$$

$$k_{dnn4} = 0.002. \quad (2.61)$$

Підставивши (2.60) в (2.56-2.58), а (2.61) в (2.59) отримано:

$$t_{dnn1} = 0.4\lambda e^{-\varepsilon}N_x^2N_y^2\left(N_x + N_y\right)^2 + 0.02\vartheta_m N_x N_y, \quad (2.62)$$

$$t_{dnn2} = 2.5\left(\lambda e^{-\varepsilon}N_x^2N_y^2\left(N_x + N_y\right)^2 + 2\vartheta_{nm}N_x N_y\right), \quad (2.63)$$

$$t_{dnn3} = 0.4\lambda e^{-\varepsilon} N_x^2 N_y^2 \left(N_x^3 + N_y N_x^2\right) + 0.02\vartheta_m N_x N_y, \quad (2.64)$$

$$t_{dnn4} = 0.8\lambda e^{-\varepsilon} N_x^2 N_y^2 \left(N_x^3 + N_y N_x^2\right) + 0.02\vartheta_m N_x N_y. \quad (2.65)$$

Також шляхом експертного оцінювання визначено, що в системах антивірусного захисту максимально допустима помилка навчання ГНМ на тестових даних не повинна перевищувати 0.98. При цьому:

$$e^{-0.98} \approx 1. \quad (2.66)$$

Враховуючи (2.65) вирази (2.61-64) спрощені так:

$$t_{dnn1} = 0.4\lambda N_x^2 N_y^2 \left(N_x + N_y\right)^2 + 0.02\vartheta_m N_x N_y, \quad (2.67)$$

$$t_{dnn2} = 2.5\left(\lambda N_x^2 N_y^2 \left(N_x + N_y\right)^2 + 2\vartheta_{nm} N_x N_y\right), \quad (2.68)$$

$$t_{dnn3} = 0.4\lambda N_x^2 N_y^2 \left(N_x^3 + N_y N_x^2\right) + 0.02\vartheta_m N_x N_y, \quad (2.69)$$

$$t_{dnn4} = 0.8\lambda N_x^2 N_y^2 \left(N_x^3 + N_y N_x^2\right) + 0.02\vartheta_m N_x N_y. \quad (2.70)$$

Сучасні інструментальні засоби розробки НММ, орієнтовані на хмарні обчислення та дозволяють розпаралелити процес навчання. Врахувати вказану можливість в виразах (2.67-2.70) можливо шляхом введення в ці вирази відповідного коефіцієнта, що зумовлює зміну запису цих виразів:

$$t_{dnn1} = 0.4\frac{\lambda}{q} N_x^2 N_y^2 \left(N_x + N_y\right)^2 + 0.02\vartheta_m N_x N_y, \quad (2.71)$$

$$t_{dnn2} = 2.5\left(\frac{\lambda}{q} N_x^2 N_y^2 \left(N_x + N_y\right)^2 + 2\vartheta_{nm} N_x N_y\right), \quad (2.72)$$

$$t_{dnn3} = 0.4\frac{\lambda}{q}N_x^2 N_y^2\left(N_x^3 + N_y N_x^2\right) + 0.02\vartheta_m N_x N_y, \quad (2.73)$$

$$t_{dnn4} = 0.8\frac{\lambda}{q}N_x^2 N_y^2\left(N_x^3 + N_y N_x^2\right) + 0.02\vartheta_m N_x N_y. \quad (2.74)$$

де q – коефіцієнт розпаралелювання процесу навчання (кількість обчислювальних процесів, що реалізують навчання НММ).

При цьому коефіцієнт q в основному залежить від доступного апаратного забезпечення процесу розробки НММ. Так на основі даних [6] сформовано рекомендації по використанню різних апаратних засобів, що забезпечують ефективне навчання ГНМ. Основні параметри вказаних апаратних засобів наведено в табл. 2.1. Зазначимо, що кількість потоків навчання ГНМ в значній мірі залежить також і від параметрів апаратного забезпечення. Так, наприклад, у випадку застосування сучасних відеокарт сімейства nVidia та технології CUDA кількість потоків навчання може досягати 512.

Таблиця 2.1

Параметри розповсюджених апаратних засобів, що використовуються для навчання нейромережевих моделей

Тип апаратного забезпечення	Перелік вимог			
	Обсяг оперативної пам'яті, ГБ	Кількість ядер	Кількість відеокарт	Обсяг пам'яті окремої відеокарти, ГБ
Локальний комп'ютер (робоча станція)	64	10	1	8
Хмарний комп'ютер	100	24	2	12
Кластер	5	1000	16	24

Для більш компактного запису виразів (2.71-2.74) введемо позначення:

$$\acute{\lambda} = \frac{\lambda}{q}. \quad (2.75)$$

де $\acute{\lambda}$ – приведена тривалість однієї навчальної ітерації.

Використавши (2.75) в (2.71-2.74) отримано:

$$t_{dnn1} = 0.4\acute{\lambda}N_x^2N_y^2\left(N_x + N_y\right)^2 + 0.02\vartheta_m N_x N_y, \quad (2.76)$$

$$t_{dnn2} = 2.5\left(\acute{\lambda}N_x^2N_y^2\left(N_x + N_y\right)^2 + 2\vartheta_{nm}N_x N_y\right), \quad (2.77)$$

$$t_{dnn3} = 0.4\acute{\lambda}N_x^2N_y^2\left(N_x^3 + N_yN_x^2\right) + 0.02\vartheta_m N_x N_y, \quad (2.78)$$

$$t_{dnn4} = 0.8\acute{\lambda}N_x^2N_y^2\left(N_x^3 + N_yN_x^2\right) + 0.02\vartheta_m N_x N_y. \quad (2.79)$$

Інтеграція виразів (2.76-2.79) та принципу формування множини допустимих видів ГНМ (2.5) дозволила записати правило для визначення множини допустимих видів ГНМ за допомогою виразів (2.80-2.83).

$$if 0.4\acute{\lambda}N_x^2N_y^2\left(N_x + N_y\right)^2 + 0.02\vartheta_m N_x N_y \le t_{max} \rightarrow dnn_1 \in DNN_{avl}, \quad (2.80)$$

$$if 2.5\left(\acute{\lambda}N_x^2N_y^2\left(N_x + N_y\right)^2 + 2\vartheta_{nm}N_x N_y\right) \le t_{max} \rightarrow dnn_2 \in DNN_{avl}, \quad (2.81)$$

$$if 0.4\acute{\lambda}N_x^2N_y^2\left(N_x^3 + N_yN_x^2\right) + 0.02\vartheta_m N_x N_y \le t_{max} \rightarrow dnn_3 \in DNN_{avl}, \quad (2.82)$$

$$if 0.8\acute{\lambda}N_x^2N_y^2\left(N_x^3 + N_yN_x^2\right) + 0.02\vartheta_m N_x N_y \le t_{max} \rightarrow dnn_4 \in DNN_{avl}, \quad (2.83)$$

де t_{max} – максимально допустимий термін створення ГНМ.

В базовому випадку можливо вважати, що $t_{max} = t_{avl}$.

Зазначимо, що в багатьох випадках при створенні антивірусних засобів можливо використовувати БД прикладів, які характеризують комп'ютерні віруси. В цьому випадку можливо вважати, що термін створення маркованого

навчального прикладу, як термін створення немаркованого навчального прикладу дорівнює 0:

$$\vartheta_m = \vartheta_{nm} = 0. \quad (2.84)$$

Підстановка (2.84) в (2.76-2.79) дозволяє записати вирази для оцінки терміну побудови ГНМ, призначених для розпізнавання комп'ютерних вірусів у вигляді:

$$t_{dnn1,db} = 0.4\acute{\lambda}N_x^2N_y^2\left(N_x + N_y\right)^2, \quad (2.85)$$

$$t_{dnn2,db} = 2.5\acute{\lambda}N_x^2N_y^2\left(N_x + N_y\right)^2, \quad (2.86)$$

$$t_{dnn3,db} = 0.4\acute{\lambda}N_x^2N_y^2\left(N_x^3 + N_yN_x^2\right), \quad (2.87)$$

$$t_{dnn4,db} = 0.8\acute{\lambda}N_x^2N_y^2\left(N_x^3 + N_yN_x^2\right). \quad (2.88)$$

де $t_{dnn1,db}$, $t_{dnn2,db}$, $t_{dnn3,db}$, $t_{dnn4,db}$- термін побудови ГНМ типу dnn_1, dnn_2, dnn_3, dnn_4 у випадку доступності БД прикладів, що використовуються для розпізнавання комп'ютерних вірусів.

Використавши вирази (2.85-2.88), правило визначення множини допустимих видів ГНМ у випадку доступності БД діагностичних параметрів правило визначення множини допустимих видів ГНМ записане за допомогою виразів:

$$if\, 0.4\acute{\lambda}N_x^2N_y^2\left(N_x + N_y\right)^2 \le t_{max} \rightarrow dnn_1 \in DNN_{avl}, \quad (2.89)$$

$$if\, 2.5\acute{\lambda}N_x^2N_y^2\left(N_x + N_y\right)^2 \le t_{max} \rightarrow dnn_2 \in DNN_{avl}, \quad (2.90)$$

$$if 0.4\acute{\lambda} N_x^2 N_y^2 \left(N_x^3 + N_y N_x^2\right) \le t_{max} \rightarrow dnn_3 \in DNN_{avl}, \qquad (2.91)$$

$$if 0.8\acute{\lambda} N_x^2 N_y^2 \left(N_x^3 + N_y N_x^2\right) \le t_{max} \rightarrow dnn_4 \in DNN_{avl}, \qquad (2.92)$$

При формуванні виразів (2.89-2.92) використано розроблений принцип формування множини допустимих видів ГНМ (2.5). Формування множини допустимих типів ГНМ дозволило перейти визначення множини ефективних типів. Використано відповідний заданий виразами (2.6, 2.7) принцип такого визначення. На першому етапі розробки проведено формування множини критеріїв ефективності ГНМ. Базою формування стали результати робіт [89-92], адаптовані до задачі розпізнавання комп'ютерних вірусів та особливостей доступних видів ГНМ. Адаптація полягала у врахуванні наступних особливостей процесу розпізнавання комп'ютерних вірусів:

- Оскільки довжина програмного/машинного коду, що потрібно проаналізувати для розпізнавання комп'ютерного вірусу в загальному випадку може змінюватись, то для розпізнавання можливо застосувати ГНМ, пристосовані для аналоізу динамічних рядів даних.

- Створення репрезентативних БД сигнатур вірусів є достатньо складним завданням, тому ГНМ повинні мати можливість навчатись на навчальних вибірках, невеликих за обсягом.

- Оновлення БД сигнатур вірусів за рахунок внесення в них інформації про нові типи вірусів досить складно, тому ГНМ повинні мати змогу навчатись на навчальних вибірках в яких різні типи вірусів представлені непропорційною.

- Розмітка сигнатур ПЗ є ресурсоємним завданням, відповідно виникає вимога забезпечення можливості навчання ГНМ на немаркованих даних.

- В роботах [44, 46] вказана можливість розпізнавання комп'ютерних вірусів за рахунок аналізу двовимірного або багатовимірного представлення програмного коду. Таким чином виникає вимога до можливості врахування топології вхідних параметрів ГНМ.

- Традиційно важливі вимоги до засобів антивірусного захисту стосуються терміну та їх побудови, швидкості розпізнавання та ресурсоємності.

- Основні характеристики ефективності ГНМ стосуються їх обчислювальної ефективності, точності розпізнавання та стабільності навчання.

Перелік та короткий опис отриманих критеріїв наведені в табл.2.2.

Таблиця 2.2

Критерії ефективності виду глибокої нейронної мережі

Критерій	Опис критерію
H_1	Аналіз динамічних рядів даних
H_2	Мінімальний обсяг навчальної вибірки
H_3	Непропорційність прикладів навчальної вибірки
H_4	Навчання на не маркованих прикладах
H_5	Врахування топології вхідних параметрів
H_6	Швидкість навчання
H_7	Швидкість розпізнавання
H_8	Похибка навчання
H_9	Похибка розпізнавання
H_{10}	Ресурсоємність навчання
H_{11}	Ресурсоємність розпізнавання
H_{12}	Обчислювальна ефективність
H_{13}	Стабільність навчання
H_{14}	Апробованість в задачах розпізнавання комп'ютерних вірусів
H_{15}	Підтримка сучасними бібліотеками для моделювання ГНМ
H_{16}	Можливість використання сучасних мов програмування
H_{17}	Використання для різних типів вхідних параметрів
H_{18}	Розпізнавання обфускованого коду

Враховуючи дані табл. 2.2 вираз (2.7) для визначення ефективності виду ГНМ модифіковано так:

$$V(DNN_i) = \sum_{k=1}^{14} \alpha_k H_k(DNN_i), \qquad (2.93)$$

Відзначимо, що в виразі (2.93) за допомогою вагового коефіцієнта α

враховується значимість k-го критерію ефективності в умовах конкретної задачі розпізнавання комп'ютерних вірусів. Множину вказаних коефіцієнтів доцільно визначити шляхом експертного оцінювання [68].

В результаті можна стверджувати, що правило формування множини ефективних видів ГНМ можливо визначити за допомогою виразу (2.94), а правило знаходження найбільш ефективного виду ГНМ – за допомогою виразу (2.95).

$$if V\left(DNN_{avl,i}\right) \geq E \rightarrow DNN_{avl,i} \in DNN_{eff}, \quad (2.94)$$

$$if V\left(DNN_{eff,k}\right) = max\left(V\left(DNN_{eff,i}\right)\right)_I \rightarrow DNN_i = DNN_{eff,max}, \quad (2.95)$$

де $DNN_{avl,i}$- і-ий допустипий тип ГНМ, множина яких визначається за допомогою правил заданих виразами (2.80-2.83) та (2.98-2.92); E - мінімально допустима ефективність; $V\left(DNN_{eff,k}\right)$ - k-ий ефективний тип ГНМ; I – кількість ефективних типів ГНМ, $DNN_{eff,max}$- максимально ефективний тип ГНМ.

Розглянемо використання отриманих правил при вирішенні однієї із найбільш актуальних в області антивірусного захисту задачі розпізнавання веб-орієнтованих комп'ютерних вірусів.

Можливо стверджувати, що при розпізнаванні вхідні параметри ГНМ можливо асоціювати з потенційно небезпечними функціями скриптової мови програмування JavaScript, на якій написана переважна більшість сучасних вірусів такого типу. Кількість таких функцій не перевищує 200. Також базуючись на даних [16] можливо стверджувати, що кількість класів таких вірусів не перевищує 100. Тобто $N_x = 200$, $N_y = 100$.

Для випадку, коли БД діагностичних параметрів відсутні підставивши ці значення в (2.80-2.83) та провівши тривіальні спрощення отримано:

$$t_{dnn1} = 1{,}44 \times 10^{13}\acute{\lambda} + 4 \times 10^{2}\vartheta_{m}, \quad (2.96)$$

$$t_{dnn2} = 9 \times 10^{13}\acute{\lambda} + 10^{3}\vartheta_{nm}, \quad (2.97)$$

$$t_{dnn3} = 4{,}8 \times 10^{12}\acute{\lambda} + 4 \times 10^{2}\vartheta_{m}, \quad (2.98)$$

$$t_{dnn4} = 9{,}6 \times 10^{12}\acute{\lambda} + 4 \times 10^{2}\vartheta_{m}. \quad (2.99)$$

Після підстановки $N_x = 200$, $N_y = 100$ в (2.85-2.88) отримано подібні вирази для розрахунку терміну розробки ГНМ при доступності БД вказаних діагностичних параметрів:

$$t_{dnn1} = 1{,}44 \times 10^{13}\acute{\lambda}, \quad (2.100)$$

$$t_{dnn2} = 9 \times 10^{13}\acute{\lambda}, \quad (2.101)$$

$$t_{dnn3} = 4{,}8 \times 10^{12}\acute{\lambda}, \quad (2.102)$$

$$t_{dnn4} = 9{,}6 \times 10^{12}\acute{\lambda}. \quad (2.103)$$

У випадку приблизних розрахунків можна вважати, що максимально допустимий термін розробки ГНМ можливо розрахувати так:

$$t_{dnn} = k_{dnn} \times t_{all}, \quad (2.104)$$

де t_{dnn} - термін розробки ГНМ, k_{dnn} – коефіцієнт пропорційності, t_{all} – максимально допустимий термін розробки антивірусного засобу.

На основі даних [25] прийнято, що $k_{dnn} = 0{,}5$, а максимально допустимий термін розробки антивірусного засобу становить приблизно 1 рік.

Це дозволяє модифікувати вираз (2.104) так:

$$t_{dnn} = 1{,}5 \times 10^7 \text{с}. \quad (2.105)$$

Підставивши (2.96-2.99, 2.105) в (2.80-2.83) отримано вирази (2.106-2.109), що формують правило визначення допустимих видів ГНМ, призначених для розпізнавання веб-орієнтованих скриптових вірусів при відсутності доступу до БД діагностичних параметрів.

$$if 1{,}44 \times 10^{13} \acute{\lambda} + 4 \times 10^2 \vartheta_m \leq 1{,}5 \times 10^7 \rightarrow dnn_1 \in DNN_{avl}, \quad (2.106)$$

$$if 9 \times 10^{13} \acute{\lambda} + 10^3 \vartheta_{nm} \leq 1{,}5 \times 10^7 \rightarrow dnn_2 \in DNN_{avl}, \quad (2.107)$$

$$if 4{,}8 \times 10^{12} \acute{\lambda} + 4 \times 10^2 \vartheta_m \leq 1{,}5 \times 10^7 \rightarrow dnn_3 \in DNN_{avl}, \quad (2.108)$$

$$if 9{,}6 \times 10^{12} \acute{\lambda} + 4 \times 10^2 \vartheta_m \leq 1{,}5 \times 10^7 \rightarrow dnn_4 \in DNN_{avl}. \quad (2.109)$$

Схожа процедура підстановки (2.100-2.103, 2.105) в (2.89-2.92) дозволила записати вирази (2.110-2.113), що формують правило визначення допустимих видів ГНМ, призначених для розпізнавання веб-орієнтованих скриптових вірусів при наявності доступу до БД діагностичних параметрів.

$$if 1{,}44 \times 10^{13} \acute{\lambda} \leq 1{,}5 \times 10^7 \rightarrow dnn_1 \in DNN_{avl}, \quad (2.110)$$

$$if 9 \times 10^{13} \acute{\lambda} \leq 1{,}5 \times 10^7 \rightarrow dnn_2 \in DNN_{avl}, \quad (2.111)$$

$$if 4{,}8 \times 10^{12} \acute{\lambda} \leq 1{,}5 \times 10^7 \rightarrow dnn_3 \in DNN_{avl}, \quad (2.112)$$

$$if 9{,}6 \times 10^{12} \acute{\lambda} \leq 1{,}5 \times 10^7 \rightarrow dnn_4 \in DNN_{avl}. \quad (2.113)$$

Важливою перевагою розроблених правил формування множини допустимих та ефективних видів ГНМ, що задані виразами виду (2.80-2.83), (2.89-2.92), (2.94, 2.95), (2.106-2.113) є їх залежність тільки від таких умов задачі створення НМЗ розпізнавання як:

– Характеристики апаратного забезпечення.

– Ресурси на створення БД навчальних прикладів.

– Характеристики комп'ютерних вірусів, що підлягають розпізнаванню.

При цьому особливості доступних типів ГНМ враховані в математичному забезпеченні сформованих правил. Це дозволяє уникнути довготривалих експериментів пов'язаних з формуванням множини допустимих і ефективних видів ГНМ та забезпечує можливість автоматизації такого формування, що значно знижує вимоги до кваліфікації розробників НМЗ антивірусного захисту.

2.4. Висновки до другого розділу

В даному розділі вирішувалась наукова задача розвитку методологічної бази нейромережевого розпізнавання комп'ютерних вірусів. Основні результати розділу наступні:

Розроблена концептуальна модель, що за рахунок гармонізації термінів, формалізації процесу розпізнавання, конкретизації критеріїв оцінювання ефективності процесу нейромережевого розпізнавання комп'ютерних вірусів, дозволила деталізувати напрямки подальших досліджень.

Вперше розроблено принципи застосування нейронних мереж для розпізнавання комп'ютерних вірусів. На відміну від відомих у означених принципах відображено:

- оцінку допустимості використання виду глибокої нейронної мережі для розпізнавання комп’ютерних вірусів,

- визначення множини ефективних видів глибоких нейронних мереж для розпізнавання комп’ютерних вірусів,

- оцінювання ефективності виду глибокої нейронної мережі,

- представлення процесу функціонування програмного забезпечення у вигляді графа залежностей значень та станів,

- оцінювання безпечності програмного забезпечення за допомогою графів залежностей значень та станів.

Це забезпечило можливість підвищення ефективності нейромережевих моделей.

Вперше розроблені правила визначення ефективних видів глибоких нейронних мереж, які на відміну від існуючих, за рахунок застосування означених вище принципів та отриманих аналітичних виразів для оцінки допустимості і ефективності виду глибокої нейронної мережі, дозволяють уникнути довготривалих комп'ютерних експериментів пов'язаних з формуванням множини допустимих і ефективних видів ГНМ та забезпечують можливість автоматизації такого формування.

РОЗДІЛ 3. НЕЙРОМЕРЕЖЕВА МОДЕЛЬ ТА МЕТОДИ РОЗПІЗНАВАННЯ КОМП'ЮТЕРНИХ ВІРУСІВ

3.1. Модель формування параметрів навчальних прикладів глибокої нейронної мережі

Відповідно результатів досліджень, наведених в п. 2.1, та даних [11, 12] модель формування параметрів навчальних прикладів ГНМ повинна забезпечувати можливість використання в якості вхідних параметрів закодованих значень:

− викликів API-функцій (ψ_{api});

− байт-послідовності N-грамів (ψ_{bs});

− опкодів, вилучених з дизасебльованих файлів (ψ_{opc});

− результатів статистичного аналізу зразків шкідливих та безпечних програм (ψ_{st});

− значень регістрів 32-розрядних та 64-розрядних процесорів EAX, EBX, EDX, EDI, ESI, EBP, RAX, RBX, RDX, RDI, RSI, RBP (ψ_{reg});

− параметрів графів викликів API-функції (ψ_{gr});

− бінарного двохвимірного представлення програмного коду (ψ_{bin}); параметрів PE-заголовку файлу (ψ_{pe});

− параметрів графу залежностей значень та станів програмного забезпечення (ψ_{dvs}).

Таким чином перетворення вхідної інформації моделі у набір вхідних параметрів ГНМ описується виразами виду:

$$F_{\text{fpe}}(\Psi) = X, \tag{3.1}$$

$$\Psi = \left\{\psi_{\text{api}},\psi_{\text{bs}},\psi_{\text{opc}},\psi_{\text{st}},\psi_{\text{reg}},\psi_{\text{gr}},\psi_{\text{bin}},\psi_{\text{pe}},\psi_{\text{dvs}}\right\}, \tag{3.2}$$

де X–множина вхідних параметрів ГНМ, $\boldsymbol{\Psi}$ - множина типів діагностичих параметрів ПЗ.

З метою забезпечення релевантності впливу всіх вхідних параметрів на результати розпізнавання в моделі передбачено їх нормалізація. В базовому варіанті для цього пропонується використовувати вираз типу:

$$x_i = \frac{\psi_i - \psi_{min}}{\psi_{max} - \psi_{min}}, \tag{3.3}$$

де x_i – значення і-го вхідного параметру ГНМ, ψ_i – значення і-го діагностичного параметру, ψ_{max}, ψ_{min} – максимальне та мінімальне значення діагностичного параметру.

У випадку, коли зареєстровані діагностичні параметри мають не числовий, а категоріальний (символьний) характер передбачено їх попередня обробка за допомогою виразів типу:

$$\Psi = \{\psi\}_G, \tag{3.4}$$

$$\psi_g = g, g = 1 \ldots G\ , \tag{3.5}$$

де Ψ – множина діагностичних параметрів, G – кількість діагностичних параметрів.

Основною особливістю моделі формування параметрів навчальних прикладів ГНМ є можливість створення навчальних прикладів на основі параметрів графу залежностей значень та станів програмного забезпечення. Це забезпечило адаптацію ГНМ до вимогу щодо можливості розпізнавання обфускованого програмного коду, що в свою чергу зумовило використання в моделі процедури деобфускації. Розробка такої процедури базувалась на особливостях процесу обфускації.

Зазначимо, що і при розробці шкідливого ПЗ і у випадку застосування обфускації для захисту авторських прав логіка обфускації полягає у тому, щоб виключити з програмного коду більшість очевидних зв'язків, тобто трансформувати його таким чином, щоб вивчення і модифікації обфускованої програми було більш складною і витратною задачею ніж побудова нового алгоритму [26]. При цьому процедура обфускації має виконуватися в автоматичному режимі і характеризуватися мінімальнім кошторисом.

Для наведення точного визначення процесу обфускації введено наступні поняття:

- Початкового програмного коду.

- Процес трансформації.

- Множина алгоритмів, що виникають внаслідок трансформації – трансформований програмний код.

В результаті процес трансформації можливо записати у вигляді процедури, заданої виразом виду:

$$PR1 = TR(PR1) = \begin{cases} PR2_1 \\ PR2_2 \\ \dots \\ PR2_n \end{cases}, \quad (3.6)$$

де $PR1$ - початковий програмний код; $TR()$ - процес трансформації; $\{PR2_1 \dots PR2_n\}$- множина трансформованих алгоритмів.

При цьому, процес трансформації $TR()$ призводить до обфускації програмного коду при відповідності наступним вимогам:

1. Програмний код $\{PR2_1 \dots PR2_n\}$ функціонує аналогічно програмному коду $PR1$;

2. Програмний код $\{PR2_1 \dots PR2_n\}$ суттєво відрізняється від програмного коду $PR1$;

3. Застосування відомих алгоритмів реверсивної інженерії по відношенню до програмного коду $\{PR2_1 \dots PR2_n\}$ $PR2_1 \dots PR2_n$ не є ефективним;

4. Застосування відомих алгоритмів детрансформації програмного коду $\{PR2_1 \dots PR2_n\}$ у програмний код $PR1$ не є ефективним;

5. Кожне застосування процедури трансформації програмного коду $PR1$ створює новий програмний код $\{PR2_1 \dots PR2_n\}$, особливості побудови якого неможливо передбачити.

Розглянемо застосування розробленої процедури (3.1) для формалізації основних видів алгоритмів обфускації. Зазначимо, що відповідно [4, 5] класифікують дві основані групи таких алгоритмів:

1. Загальні (абстрактні) обфускаційні алгоритми.

2. Програмно орієнтовані алгоритми обфусикації.

Алгоритми обфускації першого типу не пов’язані з особливостями мови програмування і можуть бути використані навіть по відношенню до асемблерного коду. Більш ефективним вважається побудова обфускатора на основі абстрактного алгоритму процедури, що використовує всі переваги конкретного коду ПЗ [26].

Розглядаючи різновиди обфускаційних алгоритмів доцільно почати з даної загальної схеми, а далі проводити аналіз методів, що можуть бути використані при її застосуванні. Серед абстрактих алгоритмів обфускації найбільш універсальним є алгоритм Колберга (Collberg). Структура алгоритму Колберга показана на рис.3.1.

Виконання алгоритму Колберга можна умовно поділити на наступні чотири базові етапи:

1. Завантаження елементів програмного коду $PR1$;

2. Завантаження бібліотек;

3. Циклічний процес проведення процедури трансформації $TR()$ шляхом виділення фрагменту коду, що повторюється до досягнення необхідного рівня чи перевищення ресурсоємності системи;

4. Генерація програмного коду $PR2_n$.

Вхідними даними алгоритму Колберга являються:

1. Вихідні елементи програмного коду $PR1$ (C – Code)

2. Стандартні бібліотеки, що використовуються у програмному коді $PR1$;

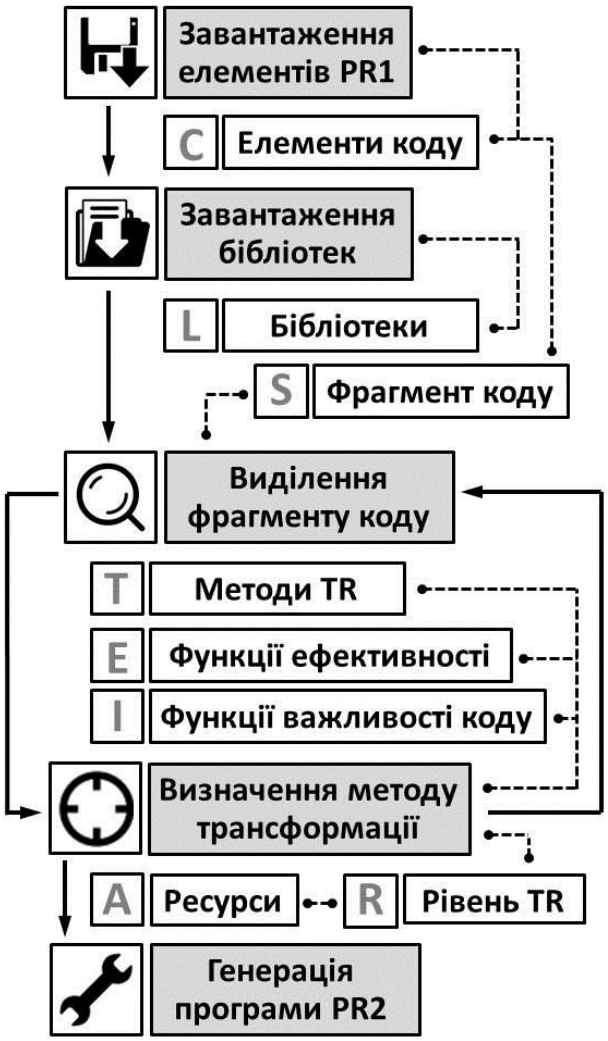

Рис. 3.1. Схема обфускації програмного коду за алгоритмом Колберга

3. Методі трансформації програмного коду (T – Transformation);

4. Фрагмент програмного коду $PR1$ (S – Segment), що підлягає трансформації;

5. Набір функцій, що визначають ефективність (E – Efficiency) методів трансформації;

6. Набір функцій, що визначають важливість фрагменту коду S;

7. Максимально допустимий об’єм системних ресурсів, що можуть бути використані для обфускації (A – Accept Cost);

8. Параметр, що вказує на необхідний рівень обфускації фрагменту програм-ного коду (R – Require Obfuscation).

Алгоритм Колберга представляє собою загальну схему процесу обфускації. При цьому спеціалізовані алгоритми, хоча і використовують вказаний алгоритм, однак мають певні відмінності. Ці відмінності визначаються методами обфускації, що можуть бути класифіковані як:

1. Лексична обфускація.

2. Обфускація даних.

3. Обфускація керування.

Лексична обфускація дозволяє швидко і без значних витрат апаратного ресурсу перевести програмний код у форму, яка не піддається аналізу програміста, але при цьому даний метод є надзвичайно нестійким по відношенню до алгоритмів деобфускації.

Обфускація даних, що полягає у трансформації структур даних відноситься до групи більш складних методів. Методи обфускації даних можна класифікувати розклавши на три підгрупи:

1. Обфускація збереження, що включає у себе зміну інтерпретацію типів даних до форм незручних для аналізу, перехід між глобальним та локальним збереженням даних, переведення статичних даних (що включають строки програмного коду) у процедурні, перехід від змінних до масивів змінних, а також кодування змінних.

2. Обфускація з’єднання, що полягає у об’єднанні змінних, реструктуруванні масивів і зміні ієрархії наслідування класів.

3. Обфускація перевпорядкування, що включає у себе зміну послідовності оголошення змінних та перевпорядкування процедур і функцій.

Дослідження підгруп обфускації даних показує, що дана група методів вимагає суттєве збільшення апаратного ресурсу, але також є більш стійкою до деобфускації.

Обфускація керування полягає у заплутуванні послідовності виконання програмного коду. Алгоритми даного методу захисту ґрунтуються на використанні непрозорих предикат. Зазначимо, що предикат $P()$, вважається непрозорим в тому випадку, коли результати його виконання є невідомими. При цьому, предикат, що завжди повертає значення “True” позначається як $P(t)$, а предикат, що завжди повертає значення “False” позначається як$P(f)$, а предикат, що може повертати одне з двох значень позначається як $P(t, f)$.

На рис. 3.2-3.4 показано три основні алгоритми обфускації програмного коду з використанням непрозорого предиката. Загальна схема полягає у тому, що блок програми PR розбивається на два блоки $PR1$і $PR2$, з’єднані через непрозорий предикат$P(t)$, $P(f)$ чи $P(t, f)$.

Структура загальної схеми наведена на рис. 3.2. Згідно зі схемою 3.2 блоки $PR1$ і $PR2$ об’єднуються через непрозорий предикат $P(t)$, при цьому для деобфускатора неочевидно, що блок $PR2$ виконується завжди.

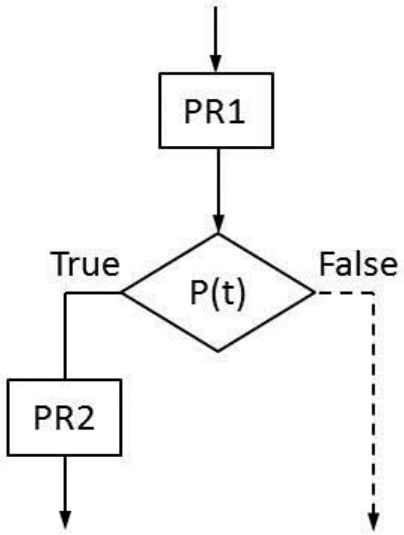

Рис. 3.2 Загальна схема алгоритму обфускації на базі непрозорого предикату

На рис. 3.3 показано вдосконалення даної схеми шляхом додавання у фальшиву гілку блоку $PR3$, що містить набір операцій, який є складним для аналізу деобфускатору і при цьому ніколи не виконується згідно функції непрозорого предикату $P(t)$.

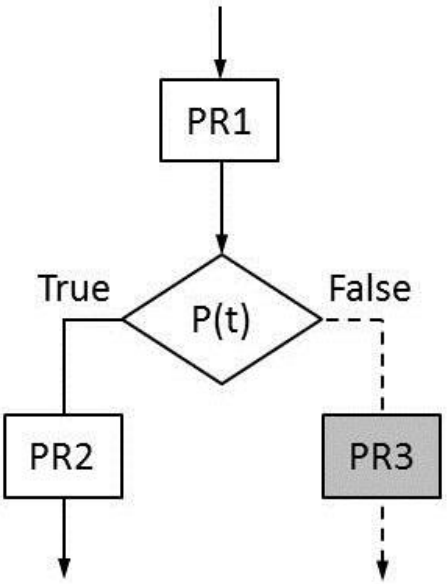

Рис. 3.3 Схема алгоритму обфускації на базі непрозорого предикату з використанням фальшивої гілки виконання програми

Схема показана на рис. 3.4 вказує на можливість застосування у алгоритмі обфускації керування непрозорого предикату типу $P(t, f)$.

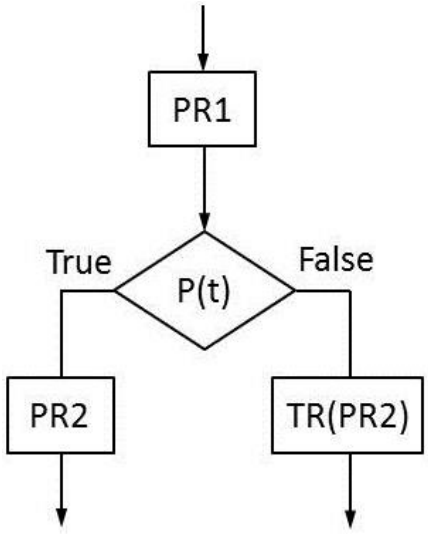

Рис. 3.4 Схема алгоритму обфускації на базі непрозорого предикату типу $P(t, f)$ з використанням фальшивої гілки виконання програми

Схема обфускації із застосуванням непрозорого предикату зумовлює необхідність виконання обох гілок, тому включає у себе два блоки:

- оригінальний блок *PR2*,

- трансформовану версію $TR(PR2)$, що виконує ту саму функцію, шо й $PR2$, але використовує інший код.

Таким чином у рамках застосування обфускації керування блоку програмного коду зумовлює вкладання алгоритмів обфускації у рамках окремих складових даного блоку.

Також слід зазначити, що непрозорі предикати можуть поділятися на:

- локальні;

- глобальні;

- міжпроцедурні.

Локальні предикати обчислюються при виконанні кожного окремого виразу. Глобальні предикати обчислюються при виконанні кожної окремої процедури, а міжпроцедурні - обчислюються при виконанні кількох процедур.

Ефективність алгоритмів обфускації керування у першу чергу залежить від непрозорих предикат, що мають бути достатньо стійкими і гнучкими у використанні. При цьому з точки зору використання апаратних ресурсів важливими параметрами є час виконання предикату та кількість операцій, що застосовуються при його використанні. Функції предиката для збільшення стійкості перед деобфускаційними алгоритмами, що базуються на технології статичного аналізу, мають мало відрізнятися від функцій ПЗ.

До обфускації керування також відносяться методи обчислювальної обфускації. Найбільш ефективний алгоритм обчислювальної обфускації відомий як алгоритм розширення умов циклів, як і в попередньому випадку базується на непрозорому предикаті, що імітує вплив на кількість виконань циклічного коду. Іншою ефективною схемою є алгоритм ліквідування бібліотечних викликів. Якщо програмне забезпечення використовує функції стандартних бібліотек, механізм роботи даних елементів програми буде відомим, що допоможе у процесі реверсивної інженерії. Тому імена функцій стандартних бібліотек також трансформують у рамках процесу обфускації.

Одним з варіантів даного підходу є використання у програмному забезпеченні власної версії бібліотек (що будується шляхом трансформації

стандартних бібліотек), це не впливає на час виконання програми, але суттєво збільшує розмір програми.

Розроблені структурні схеми алгоритмів обфускації програмного коду дозволили перейти до розробки процедуру деобфусікації. Вказана процедура базується на запропонованому принципі представлення процесу функціонування програмного забезпечення у вигляді графа залежностей значень та станів. Зазначимо, що формалізований опис даного принципу визначається виразами (2.9-2.11).

Проведений аналіз відомих методів деобфусикації програмного коду комп'ютерних вірусів дозволяє стверджувати, що процедура деобфускації багато в чому може бути подібна до процедури оптимізації програмного коду [9, 31]. Адже в процесі обфускаціі в програмний код часто додаються зайві операції та порушується його структура, що не впливає на функціональність програми, зашкоджує процесу вивчення алгоритмів функціонування. При цьому процес оптимізації, так само, як і процес деобфускації спрямований на ліквідацію зайвих операцій, тому на технічному рівні їх можна відносити до одного виду процесів.

Слід зазначити, що у такому разі, в ролі деобфускатора може виступати компілятор вихідного коду, який автоматично здійснює процес оптимізації і, тим самим, зменшує ефективність обфускації високого рівня. Для виконання програми на стороні серверу замість компілятора може використовуватися інтерпретатор, що не вносить змін до програмного коду, в інших випадках слід застосовувати обфускацію рівня асемблерного чи машинного коду. Також у якості базового деобфускаційного методів можна розглядати процедуру декомпіляції, яка дозволяє на основі машинного коду отримати текст програми на мові високого рівня, який буде зрозумілим для спеціаліста з реверсивної інженерії. Але відкритий код декомпіляційних алгоритмів дозволяє при розробці превентивних обфускаційних алгоритмів запобігти даному методу деобфускації.

Сучасні алгоритми є більш складними ніж стандартний процес декомпіляції і можуть бути класифіковані наступним чином:

1. Методи, що базуються на пошуку непрозорих предикатів.

2. Співставлення за зразками подібного програмного забезпечення чи подібних деобфускованих фрагментів.

3. Виявлення у програмному коді типових фрагментів, що не є функціональними.

4. Статистичний аналіз, що може бути використаний як у самому процесі деобфускації програмного коду так і для перевірки його коректності.

5. Динамічний аналіз потоку даних, що базується вивченні змін у функціональних елементах програми, що виникають у процесі роботи.

При цьому застосування графів можна розглядати як універсальний засіб, що може бути використано у багатьох алгоритмах деобфускації.

Сучасні алгоритми деобфускації, характеризуються наступними тенденціями побудов структури графів:

- Зменшення ролі призначень (assignment) графу, як засобу за допомогою якого інформація переноситься між операціями;

- Зменшення ролі потоку керування (control flow), що через ребра графу упорядковує фрагменти програмного коду;

- Зменшення ролі неявних залежностей (dependencies graph), для ідентифікації яких необхідно провести аналіз і знайти оператори, що визначають їх значення.

Граф залежностей значень і станів (Value State Dependence Graph) цілком відповідає зазначеним вимогам. У даному графі призначення не використовуються, поток керування застосовується лише для визначення відповідних значень значень операцій, а залежності казуються явно, включаючи умови, за яких воним мають місце. В термінології теорії графів [7, 8] граф залежностей значень і станів (ГЗЗС) може бути визначено як напрямлений позначений ієрархічний граф типу $G(T,E,l,S,S_0,S_\infty)$, що складається з функціональних елементів показаних на рис. 3.5.

До вказаних елементів відносяться:

1. Переходи (Transitions) T — вузли, що відповідають операціям;

2. Місця (Places) S — вузли, що відповідають результатам операцій;

3. Ребра (Edges) E — залежності від результатів операцій;

4. Функція розмітки (Labeling function) l відповідає кожній операції переходу;

5. Аргументи (Arguments) S_0 вказують на місця, які містять аргументи функцій на вході;

Результати (Results) S_∞ вказують на місця, які містять результати виконання функцій.

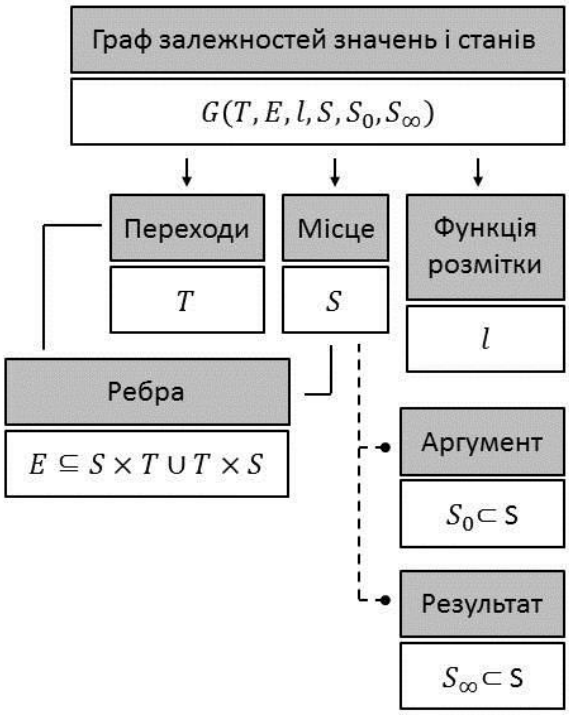

Рис. 3.5 Внутрішня структура графу залежностей значень і станів, що базується на ребрах, переходах та місцях

Кожне місце та кожне ребро графу типізується за значенням або станом. Тип ребра визначається кінцевими точками: ребром стану є ребро, кінцевою точкою якого є місце стану, ребром значення є ребро, кінцевою точкою якого є місце значення. Переходи представляють операції ГЗЗС, що здійснюються за допомогою функції розмітки, через пов'язаний з ними оператор.

Вхід I_T переходу T є місцем, пов'язаним з ним ребром. Можна вважати, що перехід є споживачем (consumer) місця.

Аналогічно, місце називається виходом O_T переходу T, якщо існує ребро від переходу до цього місця. У даному випадку, можна сказати, що перехід є постачальником (producer) місця. Набір входів переходу T називається операндами переходу або входами IS_T, а набір виходів T називається результатами переходу або виходами OS_T.

При формуванні ГЗЗС з метою використання при деобфускації потенційно шкідливого ПЗ та оптимізації коду слід дотримуватись наступних вимог:

1. Ациклічність: у ГЗЗС не використовуються графо-теоретичні коди.

2. Арність вузла: для кожного місця має існувати унікальний постачальник (тобто обумовлюється наявність окремого ребра $E \in T \times S$).

3. Лінійне застосування станів: стани виступають у ролі споживача не більше одного разу.

При використанні ГЗЗС у алгоритмах деобфускації програмного коду, компіляторах та декомпіляторaторах більш ефективним є визначення, що в якості функціональних елементів використовує вузли. Згідно з даним визначення ГЗЗС є напрямленим позначеним ієрархічним графом $G(N,E,l,N_0,N_\infty)$, що включає до своєї внутрішньої структури наступні складові:

- Вузли N (IS_T,T,OS_T) відповідають операціям з IS_T, $IS_N \in S$, що позначають входи вузла, а також операціям з OS_T, $OS_N \in S$, що позначають виходи вузла;

- Ребра $E = OS_{N1} \times IS_{N2}$ де $N_1, N_2 \subseteq N$ і $N_1 \neq N_2$ — залежності від результатів операцій, тип ребра залежить від його місця.

- Функція розмітки l відповідає кожній операції переходу.

- Вхідні вузли N_0 (IS_T,T,OS_T), де $N_0 \subset N$ відповідають за входи функцій.

- Вихідні вузли N_∞ (IS_T,T,OS_T), де $N_\infty \subset N$, відповідають за виходи функцій.

Внутрішня структура графу залежностей значень та станів, що базується на ребрах та вузлах наведена на рис. 3.6. Важливо вказати, що ребра ГЗЗС мають бути одного типу, а вузли характеризуються наступною системою рівнянь:

$$\begin{cases} IS_N = IS_T \\ OS_N = OS_T \end{cases}. \tag{3.7}$$

Для вхідних вузлів є додаткові умови:

$$\begin{cases} IS_T = \varnothing \\ OS_T = S_0 \end{cases}. \tag{3.8}$$

Аналогічно для вихідних вузлів справедливо:

$$\begin{cases} OS_T = \varnothing \\ IS_T = S_\infty \end{cases}. \tag{3.9}$$

Вузли, що використовуються у ГЗЗС, можна поділити на три типи:

- вузли обчислення,

- γ - вузли,

- комплексні вузли.

Вузли обчислення моделюють прості низькорівневі операції. Вони, свою чергу, можуть бути поділені на наступні види:

- вузли значень, що містять вхідні та вихідні значення без застосування додаткових дій,

- вузли констант, що подібні до вузлів значень, але не мають входів.

Вузли станів мають змішані входи та виходи і представляють операції додатковими діями, наприклад, такими як завантаження (load) або зберігання

(store). У свою чергу γ-вузли використовуються для вираження зумовленої поведінки для ГЗЗС. Вказані вузли на основі вхідного *p* здійснюють мультиплексування (multiplex) між двома наборами операндів *t* і *f*, виступаючі у ролі функції предикату.

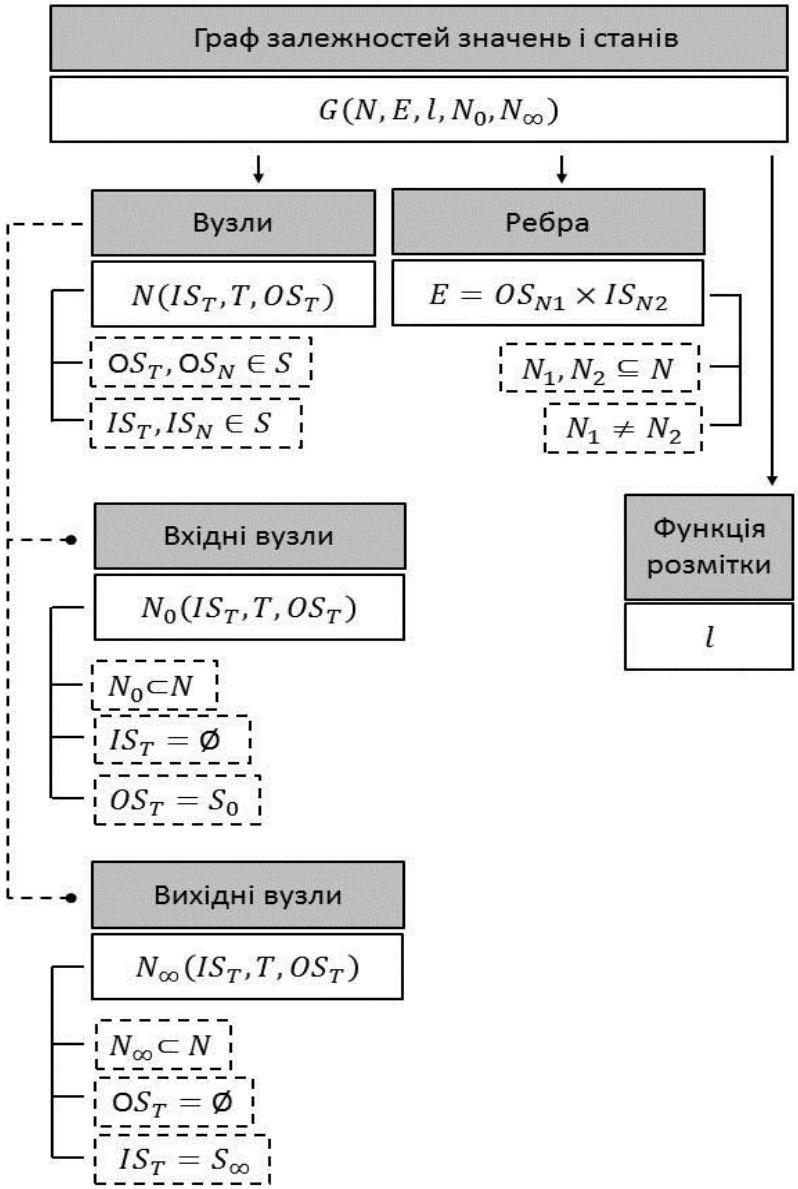

Рис. 3.6. Внутрішня структура графу залежностей значень та станів, що базується на ребрах та вузлах

Слід зазначити, що для здійснення даної операції операнди обох наборів мають характеризуватися одним типом *r*, так само як і результат виконання γ-вузла. Характерно, що γ-вузли у ГЗЗС єдиним типом вузлів, що демонструє неузгоджену поведінку. Комплексні вузли також називають областями

(regions). Область містить окремий граф (distinct graph) $G^{/}$, на який область, власне, можна і замінити. Характерно, що граф сам може містити області, а отже області, як окремий тип вузлів ГЗЗС, формують ієрархічні структури. Під час процесів оптимізації та деобфускації програмного коду області за певних умов можуть переходити від зовнішніх до внутрішніх і навпаки. Але при цьому слід пам'ятати властивість вкладання (nesting property), що накладає обмеження на ребра, що мають обєднувати вузли лише однієї області або підпорядковану область (child region). Окремим видом комплексних вузлів є θ вузли, загальна структура якого показана на рис. 3.7.

Рис. 3.7. Загальна структура θ -вузла

У ГЗЗС θ-вузол використовується виключно для моделювання циклів. Ілюстрацією такого використання є рис. 3.8, на якому наведена структурна схема циклу ГЗЗС з використанням θ-вузла.

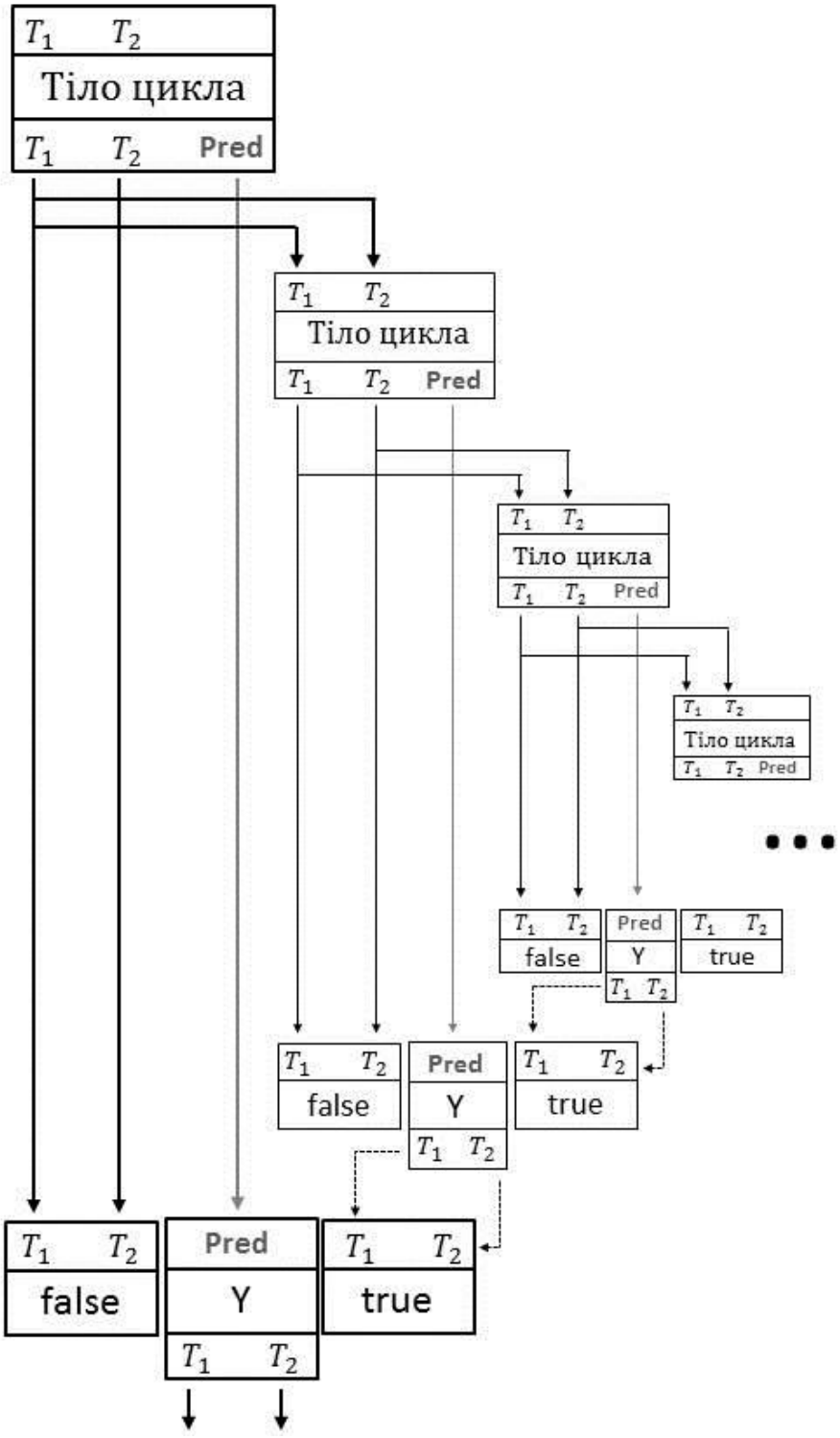

Рис. 3.8. Внутрішня структура тіла циклу з використанням θ-вузла

Подібно до γ-вузлів θ-вузол представляє собою окремий граф, але даний граф не заміняє собою вузол, тому що може являти собою нескінченний цикл, що і показано на рис. 3.8.

Слід зауважити, що на рис. 3.7, 3.8 семантика θ вузла показана як така, що базується на циклах, що керуються у кінці (tail controlled loop). Така структура θ вузла є базовою для ГЗЗС, але не обов'язковою [26].

Розглянемо застосування ГЗЗС на прикладі програми написаної на мові Сі де обчислюється простий математичний вираз типу:

$$z=z+x*y. \tag{3.10}$$

Зазначимо, що в виразі (3.7) змінні *x*, *y*, та z передаються в функцію по їх значенням. Лістинг відповідного програмного коду, написаного на мові програмування Сі, наведено на рис. 3.9.

```
int32_t fcalc ( int32_t x, int32_t y, int32_t z )
{
z = z + x * y;
return z;
}
```

Рис. 3.9. Лістинг програмного коду, для обчислення елементарного математичного виразу

При розрахунку виразу (3.7) функцією fcalc оператор «return» повертає результат обчислення. Також зазначимо, що дана функція має тип «int32» – тобто тип 32-бітної цілої змінної зі знаком, що є стандартним для бібліотек Сі. Відповідний до лістингу рис. 3.9 граф типу ГЗЗС показано на рис. 3.10.

Даний елементарний приклад наочно демонструє використання у ГЗЗС ребер станів та значень. При цьому ребра станів позначено пунктирною лінією.

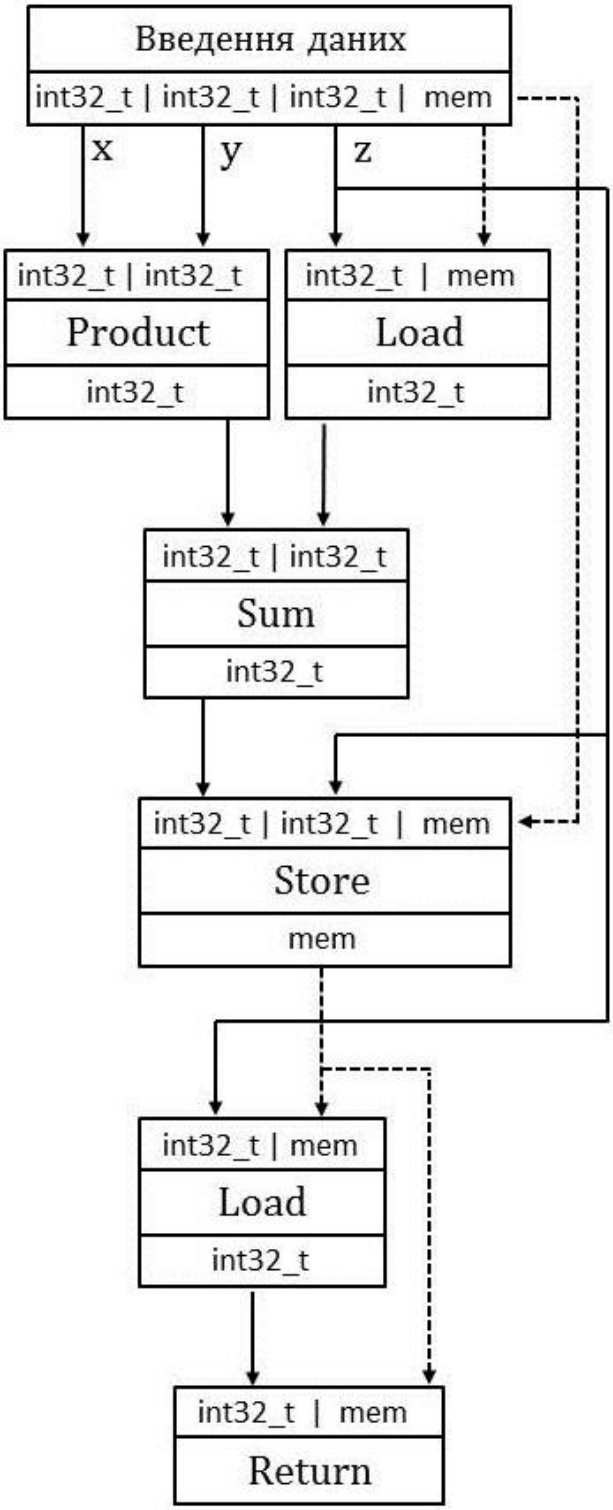

Рис. 3.10. Схема графу залежностей значень та станів для програмного коду, що реалізує обчислення елементарного математичного виразу

Як показано на рис. 3.10, у випадку коли порядок утримання вузлів значень («Product» і «Sum») підтримується автоматично, то вузли станів організуються за допомогою ребер станів. Таким чином в даній схемі виникає

потреба в тому, щоб вузел «Store» отримував на вході старий стан і давав на виході новий стан, що виконує роль входу для другого вузла “Load”. Слід також зазначити, що другий вузел “Load” додано лише для більшої наочності графу і у більш компактній схемі його можна було замінити вузлом «Return».

На рис. 3.11 показано більш складний приклад лістингу програми, що обчислює факторіал заданого числа n («uint32» – тип 32-бітної цілої змінної без знаку).

```
int32_t ffact ( uint32_t n )
{
uint32_t f = 1;
for ( ; n > 1; n--)
F = f * n;
return f;
}
```

Рис. 3.11. Лістінг програмного коду, що реалізує обчислення факторіалу

Наведений алгоритм з циклом дозволяє проілюструвати використання у ГЗЗС γ - і θ-вузлів (межі θ -вузла на рисунку позначено пунктирною лінією).

Схема графу залежностей значень та станів для програмного коду, що реалізує обчислення факторіалу наведена на рис. 3.12. Зазначимо, що цикл алгоритму керується у блоці head controlled loop, а тому γ-вузли мають бути розташовані навколо θ-вузлів, причому для випадку, коли жодна ітерація циклу не має бути виконана, застосовується хибна гілка (false branch) у той час як сама структура циклу розташовується у істинній гілці (true branch).

Таким чином, в результаті розробки отримана модель формування параметрів навчальних прикладів ГНМ, на відміну від відомих, забезпечує подання програмного забезпечення у вигляді ГЗЗС, базова структура якого показана на рис. 3.6, а особливості реалізації показані на рис. 3.7-3.12. В аналітичному вигляді вказаний граф можливо записати за допомогою виразів виду (3.11-3.15).

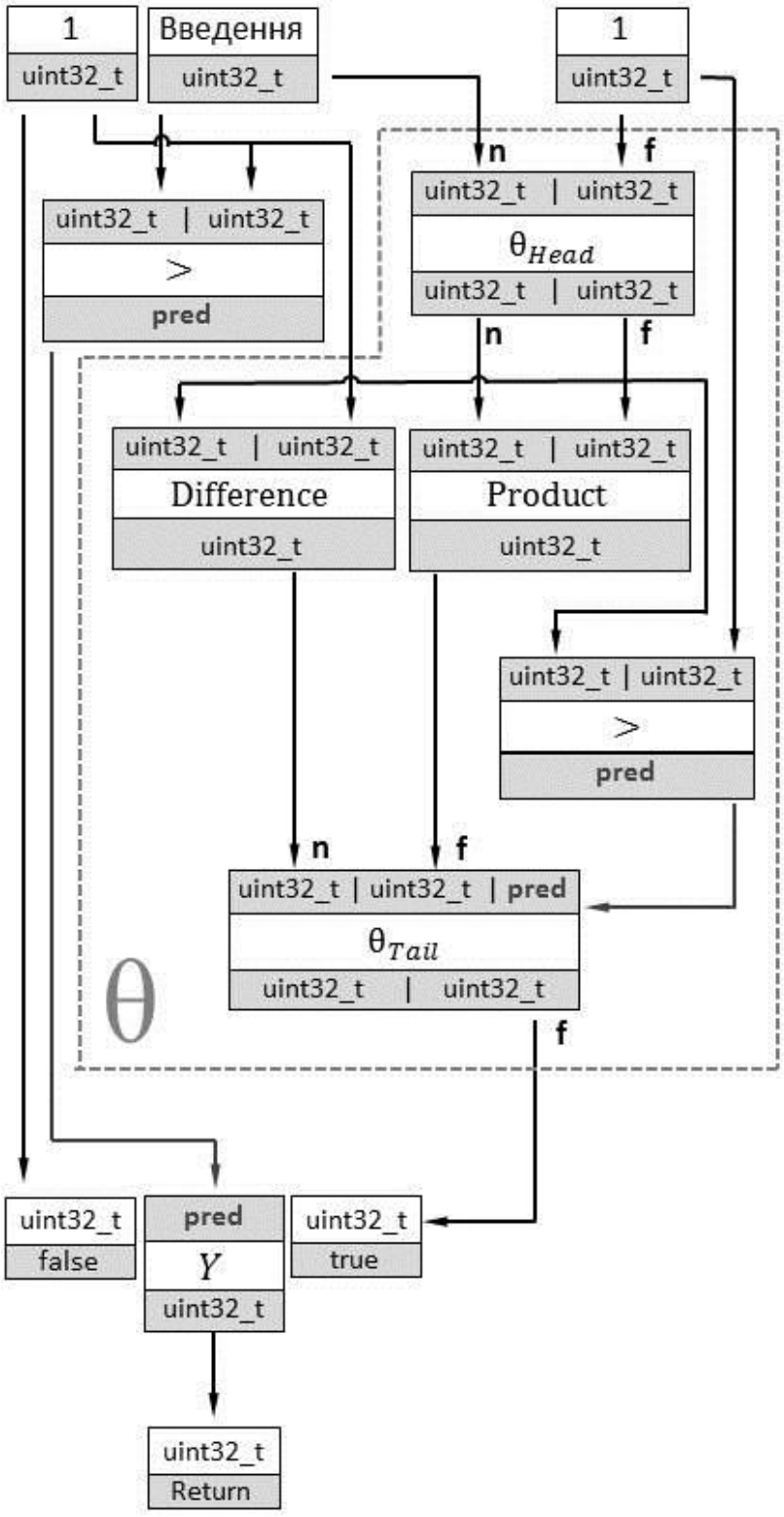

Рис. 3.12. Схема графу залежностей значень та станів для програмного коду, що реалізує обчислення факторіалу

$$G(N, E, L, N_0, N_1), \tag{3.11}$$

$$N(IS_T, T, OS_T), \tag{3.12}$$

$$E = OS_{N1} \times IS_{N2}, \tag{3.13}$$

$$N_0(IS_T, T, OS_T), \tag{3.14}$$

$$N_\infty(IS_T, T, OS_T), \tag{3.15}$$

де N - множина, що відповідають операціям з вхідною та вихідною інформацією вузлів; E - ребра, що відповідають результатам операцій; L – множина функцій розмітки, що відповідають кожній операції переходу; N_0 - вхідні вузли, що відповідають входам функцій; N_∞ - вихідні вузли , що відповідають виходам функцій.

В базовому варіанті множина діагностичних параметрів, яка відповідає параметрам ГЗЗС визначається виразом (3.16), що забезпечує можливість нейромережевого розпізнавання обфускованого програмного коду, характерного для сучасних поліморфних вірусів.

$$\psi_{dvs} = \{N, E, L, N_0, N_\infty\}, \tag{3.16}$$

де ψ_{dvs} – множина діагностичних параметрів для розпізнавання комп'ютерних вірусів визначена на основі ГЗЗС.

3.2. Метод визначення архітектурних параметрів глибокої нейронної мережі

Метод базується на розроблених принципах використання ГНМ для розпізнавання комп’ютерних вірусів, моделі правил визначення ефективних видів ГНМ та моделі формування параметрів навчальних прикладів ГНМ, що задані виразами (2.5-2.8), (2.12, 2.80-2.83, 2.89-2.95) та (3.1-3.9, 3.11) відповідно.

Перетворення інформації в цьому методі описується виразами виду:

$$\langle DNN_{ent}, A_{DNN}, R, H, \alpha, V, \Delta_d, M_v, M_p, \tau_{avl}, Q_{avl}, \varepsilon_{avl} \rangle \rightarrow \langle DNN_{ea}, A_{ea} \rangle, \quad (3.17)$$

$$DNN_{ent} = \{dnn_1, dnn_2, dnn_3, dnn_4\}, \quad (3.18)$$

$$A_{DNN} = \{A_{dnn_1}, A_{dnn_2}, A_{dnn_3}, A_{dnn_4}\}, \quad (3.19)$$

де H– множина критеріїв ефективності виду ГНМ, R– множина значень критеріїв ефективності, α – множина вагових коефіцієнтів критеріїв ефективності видів ГНМ, DNN_{ent} – множина доступних типів ГНМ, A_{DNN} – множина, що містить архітектурні параметри різних типів ГНМ, $\mathbf{A_{dnn1}}$ – множина архітектурних параметрів ГНМ без переднавчання, $\mathbf{A_{dnn2}}$ – множина архітектурних параметрів ГНМ з переднавчанням, $\mathbf{A_{dnn3}}$ – множина архітектурних параметрів ЗНМ з прямим поширенням сигналу, $\mathbf{A_{dnn4}}$ – множина архітектурних параметрів рекурентних ЗНМ, DNN_{ea} – множина найбільш ефективних та апробованих видів ГНМ, A_{ea}– архітектурні параметри найбільш ефективних та апробованих видів ГНМ, Δ_d– допустиме значення функції ефективності виду ГНМ, **V** – множина видів комп'ютерних вірусів, що мають бути розпізнані, M_v – множина доступних портретів комп'ютерних вірусів, M_p– множина портретів безпечних програм, τ_{avl}– допустимий термін побудови ГНМ, Q_{avl} – допустима ресурсоємність побудови ГНМ, ε_{avl}– допустима похибка розпізнавання. Метод передбачає виконання 6 етапів. Структурно-аналітична схема даного методу показана на рис. 3.13.

Етап 1 – визначення основних умов застосування ГНМ.

Вхідними даними етапу є множини комп'ютерних вірусів, що підлягають розпізнаванню (**V**) та множини доступних портретів комп'ютерних вірусів (M_v).

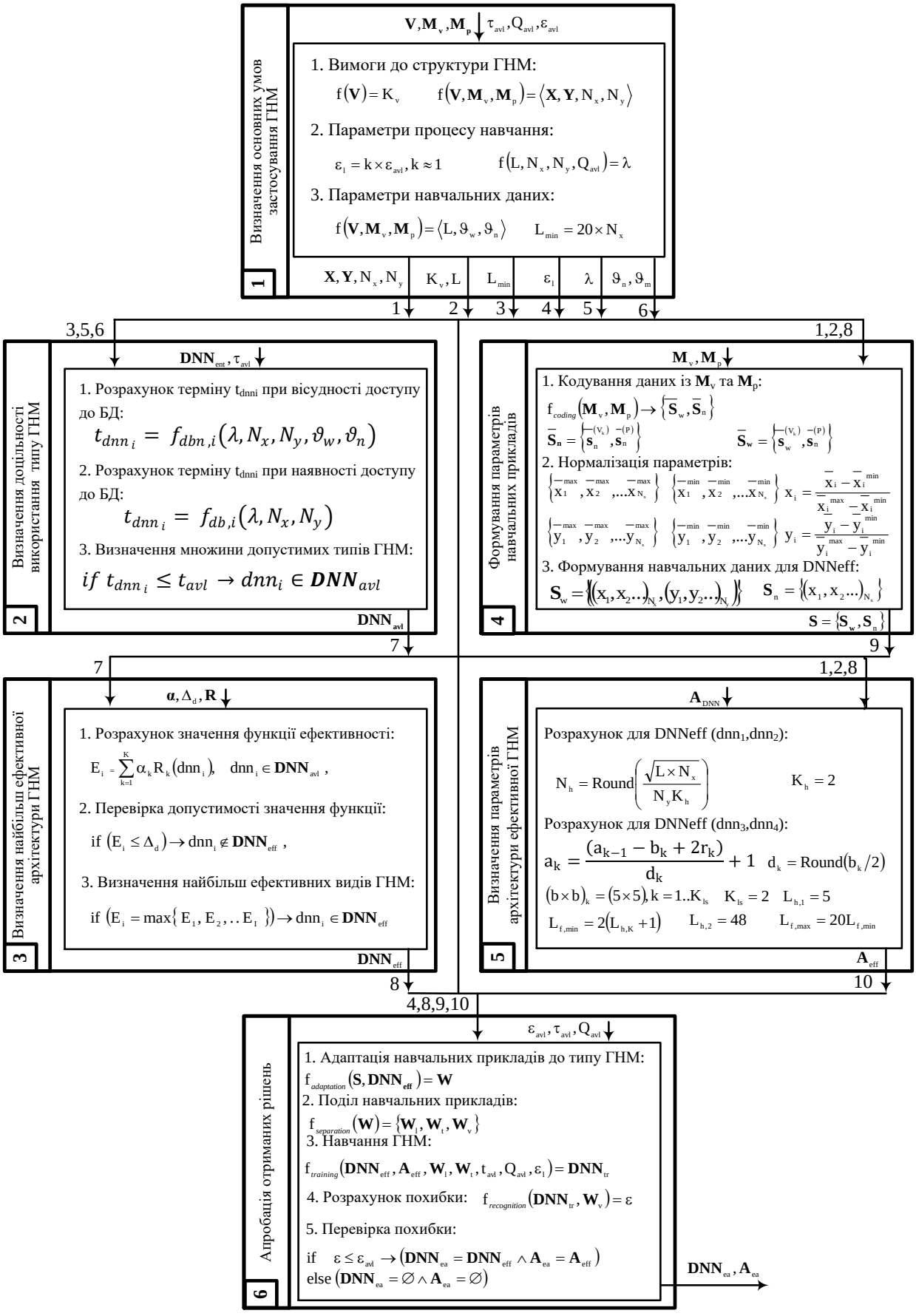

Рис. 3.13. Структурно-аналітична схема методу визначення архітектурних параметрів глибокої нейронної мережі

Крім того до вхідних даних етапу відносяться: множина портретів безпечних програм (M_p), допустимого терміну розробки ГНМ розпізнавання (τ_{avl}), допустимої ресурсоємності побудови ГНМ (Q_{avl}), та допустимої похибки розпізнавання комп'ютерних вірусів (ε_{avl}).

Етап розділено на три кроки.

Крок 1. Визначаються вимоги до структури ГНМ

- X - множина вхідних параметрів ГНМ, що формується за допомогою моделі, розробленої в п. 3.1. В першому наближенні елементи цієї множини еквівалентні нормалізованим значенням діагностичних параметрів.

- Y - множина вихідних параметрів ГНМ, що співвідноситься з видами комп'ютерних вірусів, які мають бути розпізнані.

- N_x - кількість вхідних параметрів ГНМ, що дорівнює кількості елементів множини **X**.

- N_y - кількість вихідних параметрів ГНМ.

При цьому:

$$\langle X, Y, N_x, N_y \rangle = f(V, M_v, M_p). \tag{3.20}$$

- K_v - кількість видів комп’ютерних вірусів, що мають бути розпізнані:

$$K_v = f(V). \tag{3.21}$$

Крок 2. На даному кроці визначаються параметри процесу навчання

- ε_l- максимально допустима похибка навчання. В першому наближенні:

$$\varepsilon_l = k \times \varepsilon_{avl}. \tag{3.22}$$

- λ – очікувана приведена тривалість однієї навчальної ітерації, що визначається виразом виду:

$$f\left(L, N_x, N_y, Q_{avl}\right) = \lambda. \quad (3.23)$$

Крок 3. Цей крок призначений для визначення параметрів навчальних даних

-ϑ_w – середній час, необхідний на створення одного навчального прикладу з очікуваним вихідним сигналом,

-ϑ_n – середній час, необхідний на створення одного навчального прикладу без очікуваного вихідного сигналу.

- L - загальна кількість доступних прикладів комп'ютерних вірусів та безпечних програм.

$$f\left(V, M_v, M_p\right) = \langle L, \vartheta_w, \vartheta_n \rangle. \quad (3.24)$$

- L_{min} - мінімально допустима кількість навчальних прикладів, що розраховується за допомогою виразів (2.20-2.22).

Етап 2 – визначення доцільності використання типу ГНМ

Вхідними даними етапу є множина доступних видів ГНМ (**DNN$_{ent}$**),τ_{avl}, ϑ_w, λ, ϑ_n, L_{min}, N_x, N_y. Етап розділено на три кроки.

Крок 1. На даному кроці визначається термін побудови кожного із доступних видів ГНМ для випадку відсутності доступу до БД параметрів, які характеризують комп'ютерні віруси.

$$t_{dnn_i} = f_{dbn,i}\left(\lambda, N_x, N_y, \vartheta_w, \vartheta_n\right), \quad (3.25)$$

де t_{dnn_i} – термін побудови і-го виду ГНМ, $f_{dbn,i}$ – вид функціональної залежності t_{dnn_i} від кількості вхідних та вихідних навчальних параметрів ГНМ, приведеної тривалості однієї навчально ітерації прикладів та тривалості створення одного навчального прикладу, за умови відсутності доступу до БД навчальних прикладів.

Для і-го виду ГНМ при розрахунку (3.25) використовуються вирази (2.76-2.79).

Крок 2. На даному кроці визначається термін побудови кожного із доступних видів ГНМ для випадку наявності доступу до БД прикладів, які характеризують комп'ютерні віруси:

$$t_{dnn_i} = f_{db,i}(\lambda, N_x, N_y). \quad (3.26)$$

$f_{dbn,i}$ – вид функціональної залежності t_{dnn_i} від кількості вхідних та вихідних навчальних параметрів ГНМ, у випадку доступності БД навчальних прикладів, що використовуються для розпізнавання комп'ютерних вірусів.

Для і-го виду ГНМ при розрахунку (3.26) використовуються вирази виду (2.85-2.88).

Крок 3. Визначається множина допустимих видів ГНМ.

$$if\, t_{dnn_i} \le t_{avl} \rightarrow dnn_i \in DNN_{avl}. \quad (3.27)$$

Зазначимо, що вираз (3.27) є компактним представленням виразів (2.80-2.83) для випадку відсутності доступу до БД навчальних прикладів та виразів (2.89-2.92) при наявності доступу до цих БД.

Виходом етапу є множина допустимих видів ГНМ DNN_{avl}.

Етап 3 – визначення найбільш ефективної архітектури ГНМ

Вхідними даними етапу являються множина DNN_{avl}, множина критеріїв ефективності виду ГНМ (***R***), множина вагових коефіцієнтів критеріїв ефективності (α) та мінімально допустима величина функції ефективності (Δ_d). При цьому:

$$R = \{R_1, R_2, \dots R_K\}, \quad (3.28)$$

$$\alpha = \{\alpha_1, \alpha_2, \dots \alpha_K\}, \quad (3.29)$$

де R_k – k-ий критерії ефективності, α_k – ваговий коефіцієнт k-го критерію ефективності, K – кількість критеріїв ефективності (в базовому випадку K=14).

Етап розділено на три кроки.

Крок 1. Для кожного типу допустимого виду ГНМ розраховується значення функції ефективності.

$$E_i = \sum_{k=1}^{K} \alpha_k R_k\,(dnn_i), dnn_i \in DNN_{avl}, \tag{3.30}$$

де E_i – функція ефективності для і-го виду ГНМ,

Крок 2. Перевірка допустимості величини функції ефективності для кожного допустимого виду ГНМ.

$$if(E_i \leq \Delta_d) \rightarrow dnn_i \notin DNN_{eff} \tag{3.31}$$

Крок 3. Визначення найбільш ефективних видів ГНМ.

$$if\big(E_i = max(E_1, E_2, \dots E_I)\big) \rightarrow dnn_i \in DNN_{eff}. \tag{3.32}$$

Виходом етапу являється множина ефективних видів ГНМ (DNN_{eff}).

Етап 4 – формування параметрів навчальних прикладів.

Етап орієнтовано на формування параметрів прикладів, що входять до складу навчальної та тестової вибірки ГНМ. Виконання етапу забезпечує можливість визначення параметрів архітектури найбільш ефективного виду ГНМ та можливість проведення експериментальних досліджень, спрямованих на апробацію розробленої ГНМ.

На вхід етапу подаються M_v, M_p, K_v, L,X, Y, N_x, N_y, $\mathrm{DNN_{eff}}$. Виконання даного етапу передбачає реалізацію. Трьох кроків.

Крок 1. Кодування даних, що входять до множини доступних портретів комп'ютерних вірусів (M_v) та до множини портретів безпечних програм (M_p). Перетворення інформації на даному кроці визначається виразами виду:

$$f_{coding}(M_v, M_p) \rightarrow \{\acute{S}_w, \acute{S}_n\}, \tag{3.33}$$

$$\acute{S}_w = \left\{\acute{s}_w^{(V_k)}, \acute{s}_w^{(P)}\right\}, \tag{3.34}$$

$$\acute{S}_n = \left\{\acute{s}_n^{(V_k)}, \acute{s}_n^{(P)}\right\}, \tag{3.35}$$

де f_{coding} – функція кодування, що визначається виразами (3.4, 3.5); $\acute{S}_w$ - множина, що містить приклади портретів ПЗ з очікуваним вихідним сигналом; $\acute{S}_n$ - множина, що містить приклади портретів ПЗ без очікуваного вихідного сигналу; $\acute{s}_w^{(V_k)}$ – множина портретів k-го виду комп'ютерного вірусу з очікуваним вихідним сигналом; $\acute{s}_w^{(P)}$ - множина портретів безпечних програм з очікуваним вихідним сигналом; $\acute{s}_n^{(V_k)}$ – множина портретів k-го виду комп'ютерного вірусу без очікуваного вихідного сигналу; $\acute{s}_n^{(P)}$ - множина портретів безпечних програм без очікуваного вихідного сигналу.

Крок 2. Нормалізація вхідних та вихідних параметрів. Для цього спочатку для кожного виду комп'ютерного вірусу та безпечних програм (тобто для кожного елементу множин $\acute{s}_w^{(V_k)}$, $\acute{s}_w^{(P)}$, $\acute{s}_n^{(V_k)}$, $\acute{s}_n^{(P)}$) для кожного вхідного параметру розраховується його максимальне та мінімальне значення. Тобто визначаються множини виду:

$$\left\{\acute{x}_1^{max}, \acute{x}_2^{max}, \dots \acute{x}_{N_x}^{max}\right\}, \tag{3.36}$$

$$\left\{\acute{x}_1^{min}, \acute{x}_2^{min}, \dots \acute{x}_{N_x}^{min}\right\}, \tag{3.37}$$

де $\acute{x}_i^{max}$, $\acute{x}_i^{min}$ - максимальне та мінімальне значення і-го вхідного параметру.

Після цього для кожної із множин $\acute{s}_w^{(V_k)}$, $\acute{s}_w^{(P)}$ розраховується максимальне та мінімальне значення кожного із вихідних сигналів. За рахунок цього визначаються множини виду:

$$\left\{\acute{y}_1^{max}, \acute{y}_2^{max}, \dots \acute{y}_{N_y}^{max}\right\}, \tag{3.38}$$

$$\left\{\acute{y}_1^{min}, \acute{y}_2^{min}, \dots \acute{y}_{N_y}^{min}\right\}, \tag{3.39}$$

де $\acute{y}_i^{max}$, $\acute{y}_i^{min}$ - максимальне та мінімальне значення і-го вихідного параметру.

Використовуючи (3.36-3.39) розраховуються нормалізовані значення кожного із вхідних та вихідних параметрів:

$$x_i = \frac{\acute{x}_i - \acute{x}_i^{min}}{\acute{x}_i^{max} - \acute{x}_i^{min}}, \tag{3.40}$$

$$y_i = \frac{\acute{y}_i - \acute{y}_i^{min}}{\acute{y}_i^{max} - y_i^{min}}, \tag{3.41}$$

де x_i, y_i – нормалізовані значення і-го вхідного та і-го вихідного параметрів.

Крок 3. Формування навчальних даних, що полягає у визначенні множин виду:

$$S_w = \left\{\left((x_1, x_2, \dots)_{N_x}, (y_1, y_2, \dots)_{N_y}\right)_1, \left((x_1, x_2, \dots)_{N_x}, (y_1, y_2, \dots)_{N_y}\right)_2, \dots\right\}, \tag{3.42}$$

$$S_n = \left\{\left((x_1, x_2, \dots)_{N_x}\right)_1, \left((x_1, x_2, \dots)_{N_x}\right)_2, \dots\right\}, \tag{3.43}$$

де S_w – множина навчальних прикладів, що містять очікуваний вихідний сигнал, S_n - множина навчальних прикладів, в яких очікуваний вихідний сигнал відсутній.

Виходом етапу є $S = \{S_w, S_n\}$– множина навчальних даних ГНМ.

Етап 5 – визначення параметрів архітектури найбільш ефективного виду ГНМ.

Вхідними даними етапу є DNN_{eff}, множина основних конструктивних параметрів ГНМ (A_{DNN}), N_x, N_y, L, а виходом – множина оптимізованих параметрів ефективних видів ГНМ (A_{eff}). При виконанні етапу використовуються, отримані в [84], вирази (3.44-3.48), що визначають значення архітектурних параметрів для тих видів ГНМ, що входять до складу DNN_{eff}.

$$N_h = Round\left(\frac{\sqrt{L \times N_x}}{N_y \times K_h}\right), \tag{3.44}$$

де N_h - кількість схованих нейронів в ГНМ типу dnn_1 та dnn_2; $Round$ - функція знаходження найближчого цілого числа; K_h - кількість схованих шарів нейронів, що в базовому випадку дорівнює 2.

$$a_k = \frac{(a_{k-1} - b_k + 2r_k)}{d_k} + 1, \tag{3.45}$$

$$(b \times b)_k = (5 \times 5), k = 1 \dots K_{ls}, \tag{3.46}$$

$$d_k = Round(b_k/2), \tag{3.47}$$

$$L_{f,min} = 2\left(L_{h,K} + 1\right), \tag{3.48}$$

$$L_{f,max} = 20L_{f,min}, \tag{3.49}$$

де K_{ls} – кількість шарів згортки в ГНМ типу dnn_3 та dnn_4; b_k – розмір k-го ядра згортки; a_k – розмір k-го шару згортки; r_k – кількість доповнюючих нулів для k-го шару згортки; d_k- масштабний коефіцієнт для k-го шару підвибірки; $L_{f,min}$, $L_{f,max}$ – мінімальна та максимальна кількість нейронів в повнозв'язному шарі; $L_{h,K}$ – кількість карт ознак в k-го шарі згортки.

Таким чином, результати розрахунків виразів (3.44-3.49) визначають складові множини $\boldsymbol{A_{eff}}$:

$$A_{eff} = \{N_h, a_k, b_k, d_k, L_{f,min}, L_{f,max}, K_{ls}\}, \tag{3.50}$$

<u>Етап 6 – апробація отриманих рішень</u>.

Етап полягає у реалізації експериментальних досліджень, спрямованих на підтвердження достатньої точності розпізнавання розроблених ГНМ при їх застосуванні в очікуваних умовах. Входом етапу є ε_{avl}, τ_{avl}, Q_{avl}, DNN_{eff},A_{eff}, S, а виходом DNN_{ea}, A_{ea}. Етап розділено на 5 кроків.

<u>Крок 1.</u> Множина навчальних даних S адаптується до типу ГНМ:

$$f_{\text{adaptation}}(S, \text{DNN}_{\text{eff}}) = W, \tag{3.51}$$

де $\boldsymbol{W}$ - множина навчальних даних адаптована до виду ГНМ.

Для ГНМ з прямим розповсюдженням сигналу в яких процедура переднавчання не передбачена, тобто для dnn_1, адаптація полягає у визначенні множини навчальних прикладів, кожен з яких може бути представлено виразом виду:

$$w_{k,m} = \left(\{x_1, x_2, \dots x_{N_x}\}_k \left\{y_1, y_2, \dots y_{N_y}\right\}_k\right), \tag{3.52}$$

де $w_{k,m}$ – k-ий маркований приклад, $\{x_1, x_2, \dots x_{N_x}\}_k$ – множини вхідних та вихідних даних для k-го маркованого прикладу.

Для ГНМ з прямим розповсюдженням сигналу в яких процедура переднавчання передбачена, тобто для dnn_2, крім маркованих прикладів, заданих виразом (3.52) створюються немарковані приклади. Останні задаються виразом типу:

$$w_{k,nm} = \left(\{x_1, x_2, \dots x_{N_x}\}_k\right). \quad (3.53)$$

де $w_{k,nm}$ – k-ий не маркований приклад.

По відношенню до dnn_1 особливістю формування навчальних даних для ГНМ типу dnn_3, що представляють собою ЗНМ з прямим поширенням сигналу є необхідність представлення вхідних параметрів у вигляді прямокутного рисунку з одним кольоровим каналом.

Таким чином, для адаптації $w_{k,m}$ до застосування dnn_3 кожен вхідний параметр типу x_iпідлягає перетворенню виду:

$$g(x_i) = \langle a_i, b_i, z_i \rangle, \quad (3.54)$$

де a_i, b_i – координати точки на площині x_i в декартовій системі координат, z_i – число, що відповідає кольору точки, g – функція.

Зазначимо, що розробка функції g детально описана в роботах [63-64] .

Таким чином, навчальні приклади для dnn_3, можуть бути представлені виразами виду:

$$w_{k,mp} = \left(\{\langle a_i, b_i, z_i \rangle_1, \langle a_i, b_i, z_i \rangle_2, \dots \langle a_i, b_i, z_i \rangle_{N_x}\}_k \left\{y_1, y_2, \dots y_{N_y}\right\}_k\right), \quad (3.55)$$

де $w_{k,mp}$ - k-ий маркований приклад для dnn_3.

Основною особливістю рекурентних ЗНМ до яких відносяться ГНМ типу dnn_4 є пристосованість до аналізу динамічних рядів даних, що визначає необхідність впорядкування навчальних прикладів відповідно номеру черги

подачі їх на вхід НММ. Тобто навчальна вибірка dnn_4 є складається з впорядкованих навчальних прикладів, заданих виразами виду (3.55).

Крок 2. На даному кроці W розділяється на навчальну, тестову та валідаційну вибірку:

$$f_{\text{separation}}(W) = \{W_l, W_t, W_v\}. \tag{3.56}$$

де $\boldsymbol{W}_\text{l}$ – множина тренувальних прикладів, $\boldsymbol{W}_\text{t}$ – множина тестових прикладів, $\boldsymbol{W}_\text{v}$ – множина валідаційних прикладів.

В першому наближенні для розподілу прикладів можна скористатись даними [66], що свідчать про доцільність розподілу прикладів в пропорції 8:1:1.

Крок 3. Даний крок співвідноситься з навчанням ГНМ, побудованої в результаті виконання попередніх етапів методу.

При цьому враховується максимально допустимий термін навчання та допустима ресурсоємність навчання. В аналітичному вигляді реалізація кроку визначається так:

$$f_{\text{training}}(\text{DNN}_{\text{eff}}, A_{\text{eff}}, W_l, W_t, \tau_{\text{avl}}, Q_{\text{avl}}) = \text{DNN}_{\text{tr}}, \tag{3.57}$$

Крок 4. На даному кроці перевіряється розраховується похибка розпізнавання валідаційних прикладів.

$$f_{\text{recognition}}(\text{DNN}_{\text{tr}}, W_v) = \varepsilon. \tag{3.58}$$

Крок 5. Цей крок спрямований на перевірку допустимості похибки розпізнавання валідаційних прикладів. Реалізація кроку визначається виразом виду:

$$if\,\varepsilon \le \varepsilon_{avl} \rightarrow \left(DNN_{ea} = DNN_{eff} \wedge A_{ea} = A_{eff}\right), \\ else\left(DNN_{ea} = \emptyset \wedge A_{ea} = \emptyset\right) \tag{3.59}$$

В результаті виконання п'ятого кроку визначається множина апробовано ефективних видів ГНМ (**DNN$_{ea}$**) та множина ефективних параметрів таких ГНМ (**A$_{ea}$**). Значення елементів цих множин і є виходом даного методу.

Зазначимо, що за рахунок застосування рішень, запропонованих в п. 2.2, 2.3, 3.1, описаний метод визначення архітектурних параметрів глибокої нейронної мережі забезпечує можливість зменшення обсягу експериментальних досліджень, пов'язаних з визначенням архітектурних параметрів ГНМ, призначеної для використання в НМЗ розпізнавання комп'ютерних вірусів.

3.3. Метод нейромережевого розпізнавання комп'ютерних вірусів

Розроблено метод нейромережевого розпізнавання комп'ютерних вірусів. Метод базується на розробленій моделі формування параметрів навчальних прикладів глибокої нейронної мережі та розробленому методі визначення архітектурних параметрів глибокої нейронної мережі.

Вхідними даними методу є **Q**$_{нмз}$ – вимоги до НМЗ розпізнавання комп'ютерних вірусів, **X**$_{апз}$ – характеристики апаратно-програмного забезпечення НМЗ, **R**$_{нмз}$ – ресурси на розробку НМЗ, **E**$_{нмз}$ – експертні дані для побудови НМЗ, **DNN**$_{ent}$ – доступні види НММ, A_{DNN}– параметри **DNN**$_{ent}$, **V** – множина видів комп'ютерних вірусів, що мають бути розпізнані, **БД** – бази даних, в яких міститься інформація, що може бути використана для формування портретів комп'ютерних вірусів та портретів безпечних програм, **G** – процедури, що визначають ефективність НМЗ, **U** – визначена множина критеріїв ефективності, D$_e$ – експертні дані для оцінки значущості критеріїв ефективності, що входять до складу **U**. Виходом методу єH_{eff}– параметри верифікованих НМЗ та S_{eff} – сигнал, що свідчить про достатню чи недостатню ефективність побудованого НМЗ.

Структурна схема методу нейромережевого розпізнавання комп'ютерних вірусів показана на рис. 3.14.

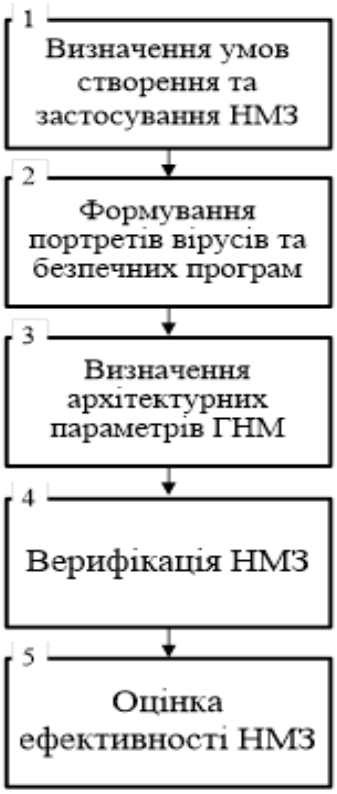

Рис. 3.14 Структурна схема методу нейромережевого розпізнавання вірусів

Перетворення інформації в даному методі описується виразами виду:

$$\langle Q_{\text{нмз}}, X_{\text{апз}}, R_{\text{нмз}}, E_{\text{нмз}}, DNN_{ent}, A_{\text{нмз}}, V, DB, G, U\rangle \rightarrow \langle H_{eff}, S_{eff}\rangle, \quad (3.60)$$

Метод передбачає виконання 5 етапів.

Етап 1 – визначення умов створення та застосування НМЗ.

На цьому етапі за допомогою процедури експертного оцінювання визначаються умови створення та застосування НМЗ розпізнавання комп'ютерних вірусів. Вхідними даними етапу є $\mathbf{Q}_{\text{нмз}}$, $\mathbf{X}_{\text{апз}}$, $\mathbf{R}_{\text{нмз}}$, $\mathbf{E}_{\text{нмз}}$. Виходом етапу є параметри, що відповідають умовам: $\mathbf{\Theta}_{\text{нв}}$ – формування навчальної вибірки, $\mathbf{\Theta}_{\text{гнм}}$ – розробки ГНМ, $\mathbf{\Theta}_{\text{апз}}$ – використання АПЗ, $\mathbf{\Theta}_{\text{чп}}$ – часу застосування НМЗ. Функціональна залежність між вхідною та вихідною

інформацією даного етапу описуються виразом виду:

$$f(Q_{\text{нмз}}, X_{\text{апз}}, R_{\text{нмз}}, E_{\text{нмз}}) \rightarrow \langle \Theta_{\text{нв}}, \Theta_{\text{гнм}}, \Theta_{\text{апз}}, \Theta_{\text{чс}}, \rangle, \quad (3.61)$$

Зазначимо, що при реалізації (3.61) доцільно застосовувати процедури експертного оцінювання розроблені в [76] для вирішення подібної задачі умови створення та застосування НМЗ для розпізнавання мережевих кібератак та аналізу голосових сигналів.

Етап 2 – формування портретів вірусів та безпечних програм.

Виконання етапу полягає в аналізі доступних баз даних для формування множин портретів комп'ютерних вірусів та портретів безпечного програмного забезпечення. Входом даного етапу є $E_{\text{нмз}}$ та множина доступних ***DB***.

$$f(DB, E_{\text{нмз}}) \rightarrow \langle M_v, M_p \rangle, \quad (3.61)$$

де $\langle M_v, M_p \rangle$ - кортеж портретів комп'ютерних вірусів та портретів безпечного програмного забезпечення.

Кортеж $\langle M_v, M_p \rangle$ являється виходом даного етапу.

Етап 3 – визначення архітектурних параметрів ГНМ.

Даний етап співвідноситься із розробленим методом визначення архітектурних параметрів ГНМ. Входом етапу є кортеж $\langle \text{DNN}_{\text{ent}}, A_{\text{DNN}}, R, H, \alpha, V, \Delta_d, M_v, M_p, \tau_{\text{avl}}, Q_{\text{avl}}, \varepsilon_{\text{alw}} \rangle$. В узагальненому вигляді перетворення інформації на даному етапі визначається виразами (3.17-3.19). Етап розділено на 5 кроків, що відповідають 1-5 етапам методу визначення архітектурних параметрів ГНМ.

Крок 1. Окреслення умов застосування ГНМ. Розрахунки даного кроку визначаються виразами (3.20-3.24). Входом кроку є множини ***V***, $\boldsymbol{M_p}$, τavl, $\boldsymbol{Q}_{\text{avl}}$, ε_{avl}. Виходом є кортеж $\langle L, \vartheta_w, \vartheta_n \rangle$.

Крок 2. Встановлення доцільності застосування типу ГНМ. Вхідними даними кроку є множина доступних видів ГНМ (**DNN$_{\text{ent}}$**),τ_{avl}, ϑ_w, λ, ϑ_n, $L_{\min}$, N_x, N_y. Після застосування до вхідної інформації виразів (3.25-3.27) виходом

кроку є DNN_{avl}.

Крок 3. Розрахунок виду ефективної архітектури. Розрахунки проводяться за допомогою виразів (3.28-3.32). На вхід кроку подаються DNN_{avl}, $\boldsymbol{R}$, α та Δ_d. Виходом кроку є DNN_{eff}.

Крок 4. Формування параметрів навчальних прикладів. На вхід етапу подаються M_v, M_p, K_v, L,X, Y, N_x, N_y, DNN_{eff}. Виходом являється $S = \{S_w, S_n\}$. При розрахунках використано вирази (3.33-3.43).

Крок 5. Розрахунок архітектурних параметрів ГНМ. Розрахунок базується на виразах (3.44-3.50). На вхід кроку подаються DNN_{eff}, A_{DNN}, N_x, N_y та L. Виходом кроку є $A_{eff} = \{N_h, a_k, b_k, d_k, L_{f,min}, L_{f,max}, K_{ls}\}$.

Етап 4 – верифікації НМЗ.

Верифікація НМЗ проводиться з позицій допустимості похибки розпізнавання. Вхідними даними етапу являються $\langle DNN_{eff}, A_{eff}\rangle$, $\square_{апз}$, $\square_{чп}$. Виходом етапу H_{eff} – множина параметрів верифікованих НМЗ. Передбачено проводити експериментальну верифікацію розроблених НМЗ, процес якої у формалізованому вигляді можливо записати так:

$$f\left(\langle DNN_{eff}, A_{eff}\rangle, \Theta_{апз}, \Theta_{чп}\right) = H_{eff}. \quad (3.62)$$

Етап 5 – оцінка ефективності НМЗ.

Оцінка реалізується з позицій забезпечення в побудованому НМЗ процедур, що забезпечують ефективне розпізнавання комп'ютерних вірусів в очікуваних умовах застосування. На вхід подається H_{eff}, **G**, **U**, $\mathbf{D_e}$. Виходом етапу є $\boldsymbol{S}_{eff}$ – сигнал, що свідчить про достатню чи недостатню ефективність побудованого НМЗ. Розрахунки етапу можливо представити за допомогою виразів виду (2.6, 2.7, 3.59).

В результаті проведених досліджень отримав подальший розвиток метод нейромережевого розпізнавання комп'ютерних вірусів, який за рахунок визначення параметрів ефективної нейромережевої моделі на базі ГНМ забезпечує адаптацію створених на його основі НМЗ розпізнавання комп'ютерних вірусів до очікуваних умов застосування.

3.4. Висновки до третього розділу

В даному розділі вирішувалась наукова задача розробки нейромережевої моделі та методів розпізнавання комп’ютерних вірусів.

Основні результати розділу наступні:

- Вперше розроблено модель формування параметрів навчальних прикладів глибокої нейронної мережі призначена для використання в засобах антивірусного захисту в які передбачено можливість використання в якості вхідних параметрів закодованих значень викликів API-функцій, байт-послідовності N-грамів, опкодів, вилучених з дизасебльованих файлів, результатів статистичного аналізу зразків шкідливих та безпечних програм, значень регістрів EAX, EBX, EDX, EDI, ESI, EBP, RAX, RBX, RDX, RDI, RSI, RBP, параметрів графів викликів API-функції, бінарного двохвимірного представлення програмного коду, параметрів PE-заголовку файлу та параметрів графу залежностей значень та станів програмного забезпечення. На відміну від відомих застосування в моделі графу залежностей значень та станів забезпечує можливість нейромережевого розпізнавання обфускованого програмного коду, характерного для сучасних поліморфних вірусів.

- Вперше розроблено метод визначення архітектурних параметрів глибокої нейронної мережі забезпечує можливість зменшення обсягу експериментальних досліджень, пов’язаних з визначенням архітектурних параметрів глибокої нейронної мережі, призначеної для використання в нейромережевих засобах розпізнавання комп’ютерних вірусів. Це забезпечило можливість підвищення ефективності нейромережевих методів розпізнавання комп'ютерних вірусів.

- Отримав подальший розвиток метод нейромережевого розпізнавання комп’ютерних вірусів, який на відміну від існуючих, за рахунок використання запропонованих елементів методологічної бази та запропонованого методу проектування архітектури глибокої нейронної мережі, забезпечує достатню похибку розпізнавання при різних умовах застосування з врахуванням обмежень щодо створення навчальної вибірки та обмежень щодо обчислювальних ресурсів системи антивірусного захисту.

РОЗДІЛ 4. НЕЙРОМЕРЕЖЕВА СИСТЕМА РОЗПІЗНАВАННЯ КОМП'ЮТЕРНИХ ВІРУСІВ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1. Архітектура нейромережевої системи

Розробка архітектури НМС проведена з позицій технічної реалізації запропонованого методу розпізнавання комп’ютерних вірусів за умов, наближених до застосування у вітчизняних засобах антивірусного захисту. Також при розробці структури НМС враховані рішення [53].

Відповідно рекомендацій [100] перший етап розробки було орієнтовано на побудову UML-діаграм системи розпізнавання. Використання мови UML пояснюється її апробованістю в задачах розробки програмного забезпечення для інформаційних систем різноманітного призначення та можливістю одночасного документування системної архітектури. Розроблено чотири основні UML-діаграми:

- прецендентів (Use case diagram),
- пакетів (Package diagram),
- компонентів (Component diagram),
- класів в (Class diagram).

Для розробки використано вільнодоступний об'єктно-орієнтований CASE-засіб проектування Rational Rose виробництва компанії IBM.

Розробку UML-діаграм розпочато з, показаної на рис. 4.1 діаграми прецендентів НМС розпізнавання, що використовується для моделювання функціональних вимог до системи. Зазначимо, що функціональні вимоги представлено у вигляді сценаріїв взаємодії користувачів з системою.

При побудові діаграми прецендентів враховано, що на першому рівні деталізації НМС можна вважати, що до її складу входить 4 актори: «Детектор», «Нейромережевий аналізатор», «Сигналізатор» та «Адміністратор». При цьому актори «Детектор», «Нейромережевий

аналізатор» та «Сигналізатор» представляють собою технічні засоби необхідні для отримання програмного коду, його нейромережевого аналізу та сигналізації про результати розпізнавання. Актор «Адміністратор» є діючою особою, що взаємодіє з системою.

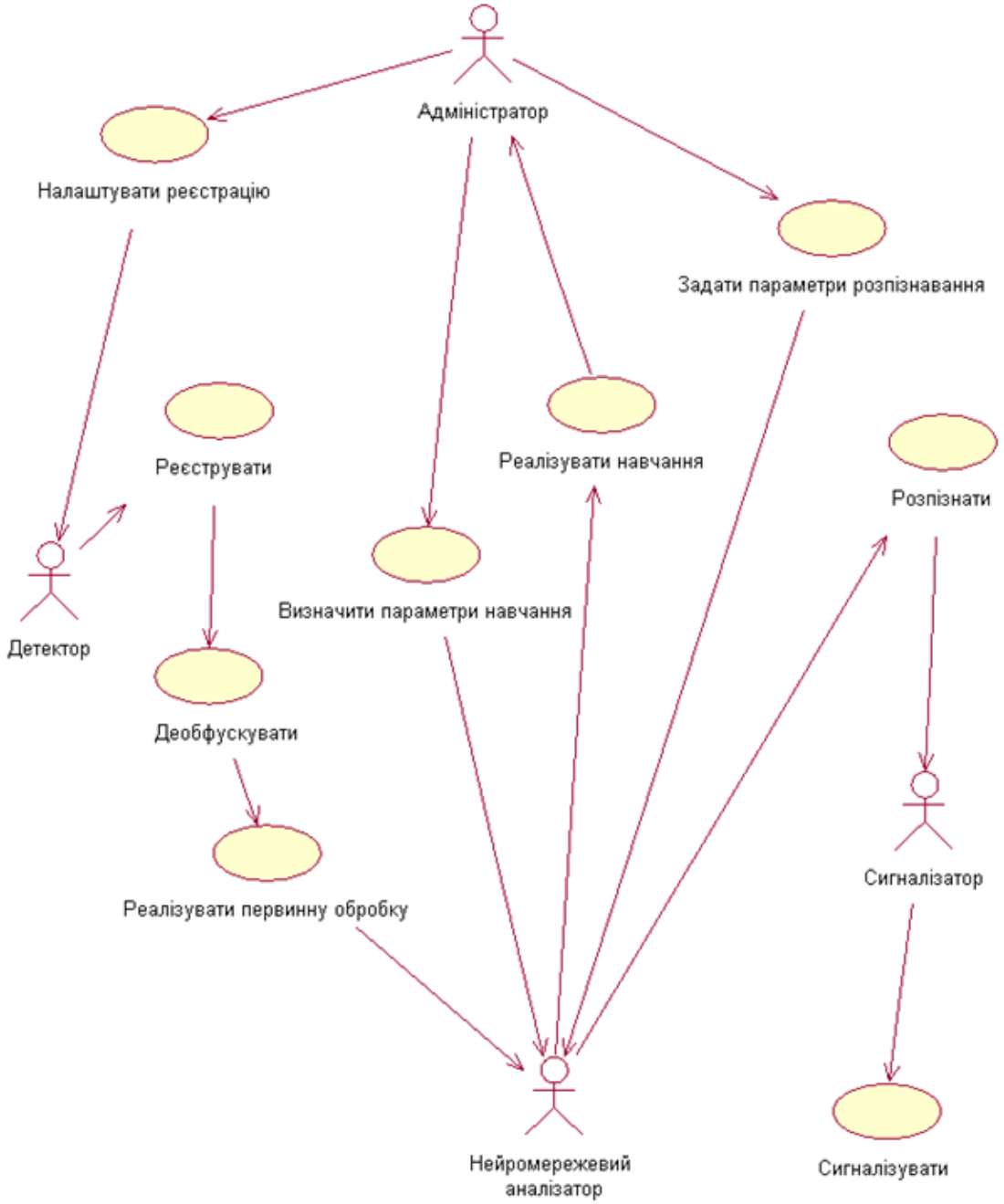

Рис. 4.1. Діаграма прецендентів

Характеристики, показаних на рис. 4.1, прецендентів використання наведено в табл. 4.1.

Таблиця 4.1

Характеристика прецендентів використання

Назва прецденту	**Мета прецденту**
Налаштувати реєстрацію	Визначити номенклатуру діагностичних параметрів, що підлягають реєстрації та часові параметри реєстрації
Реєструвати	Реалізувати реєстрацію діагностичних параметрів та зберегти зареєстровані дані
Деобфускувати	Провести деобфускацію програмного коду, тобто представити програмний код у вигляді параметрів, що характеризують відповідний граф залежностей та станів
Реалізувати первинну обробку	Представити діагностичні параметри у вигляді придатному для подачі у нейромережевий аналізатор
Визначити параметри навчання	Визначити учбову вибірку, параметри навчальних даних, необхідну точність навчання, максимально допустимий термін навчання, характеристики апаратно-програмного забезпечення
Реалізувати навчання	Розрахувати множину вагових коефіцієнтів глибокої нейронної мережі
Задати параметри розпізнавання	Задати множину образів, що підлягають нейромережевому аналізу, характеристики програмно-апаратного забезпечення, максимально допустимий термін розпізнавання
Розпізнати	Розрахувати вихідний сигнал нейромережевого аналізатора
Сигналізувати	Інтерпретувати вихідний сигнал нейромережевого аналізатора у вигляді розпізнаного класу

Діаграма пакетів, що побудована для спрощеного представлення структури НМС розпізнавання, показана на рис. 4.2.

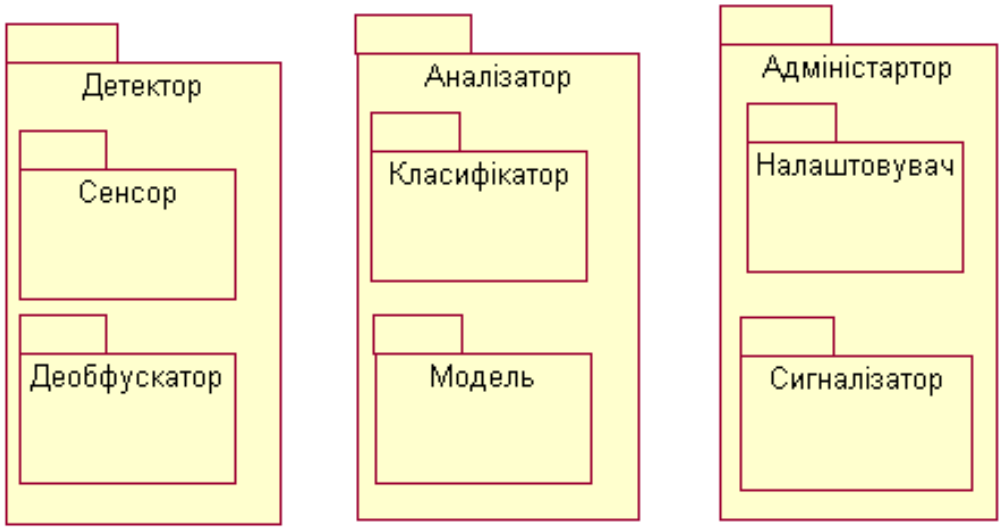

Рис. 4.2. Діаграма пакетів

На вказаній діаграмі відображено пакети та під пакети, з яких складається система розпізнавання:

– «Детектор» – пакет, що відповідає за реєстрацію та первинну обробку діагностичних параметрів. До складу цього пакету входять підпакети «Сенсор» та «Деобфускатор».

– «Аналізатор» – пакет, що відповідає за аналіз перехопленої інформації. До його складу входять підпакети «Модель» та «Класифікатор», що відповідають за побудову НММ та її застосування.

– «Адміністратор» – пакет, що відповідає за взаємодію адміністратора безпеки із системою розпізнавання. Цей пакет складається із двох підпакетів «Налаштовувач» та «Сигналізатор». Перший підпакет призначений для надання оператору системи можливості налаштовувати параметри розпізнавання. Другий пакет забезпечує адміністратора безпеки інформацію про функціонування НМС розпізнавання.

Діаграма компонентів системи розпізнавання комп'ютерних вірусів показана на рис. 4.3. До складу цієї діаграми входять три модулі «Detector», «NeuroNet» та «Admin». Вказані модулі визначають інтерфейс управління

НМС, отримання коду програмного забезпечення та його аналіз за допомогою ГНМ.

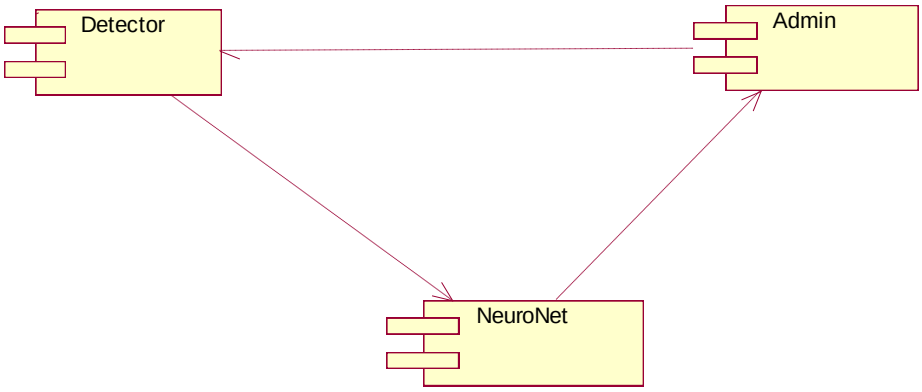

Рис. 4.3. Діаграма компонентів

Діаграма класів, що використовується для відображення структур класів, які складають архітектуру системи розпізнавання, показана на рис. 4.4.

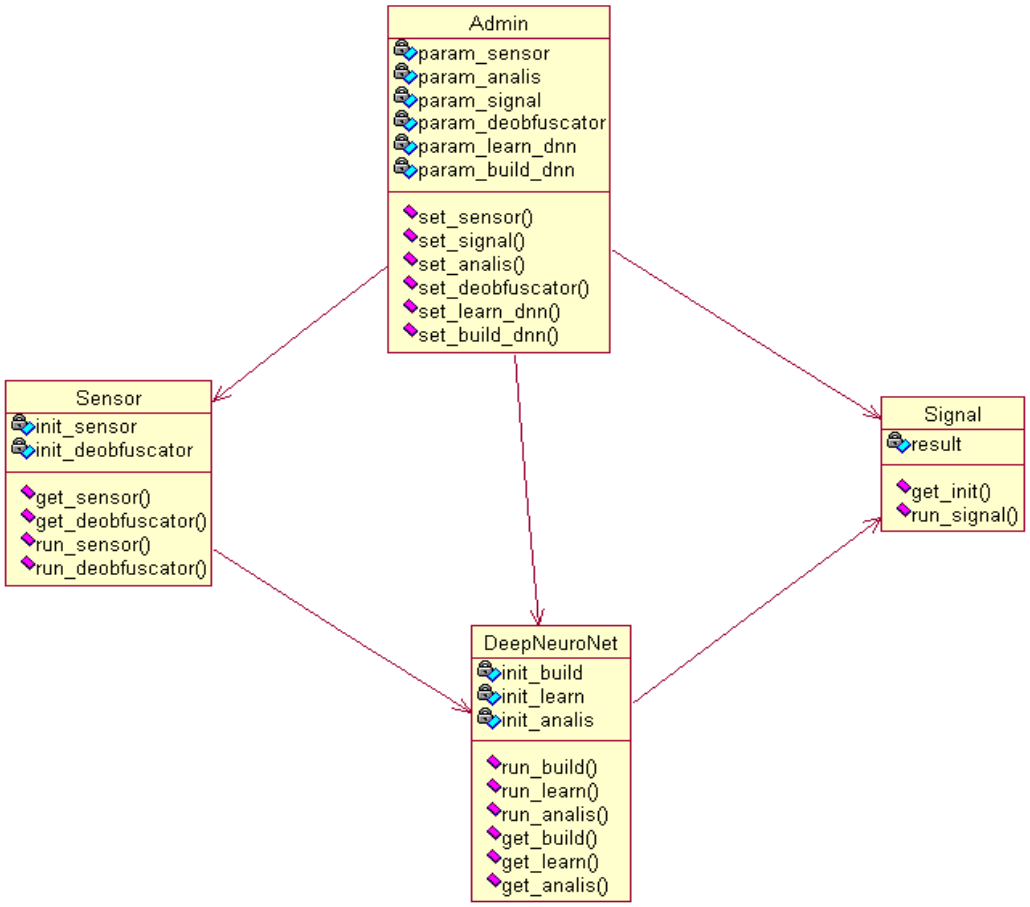

Рис. 4.4. Діаграма класів

Розробка UML-діаграм, показаних на рис. 4.1-4.4 забезпечила можливість реалізації другого етапу проектування архітектури НМС, що полягав в розробці її структурної схеми на другому етапі деталізації. Структурно НМС складається із окремих модулів, що об'єднані в 4 окремі блоки та модуль управління. Призначення та характеристика окремих блоків НМС розпізнавання комп'ютерних вірусів наведено в табл. 4.2.

Таблиця 4.2

Характеристика окремих блоків нейромережевої системи розпізнавання

Назва	**Характеристика блоку**
Блок визначення параметрів (БВП)	Визначення номенклатури та значень параметрів, що характеризують умови створення та застосування НМС.
Блок розробки навчальних прикладів (БРНП)	Попередня обробка та нормалізація вхідних та вихідних параметрів навчальних прикладів, визначення максимальної та мінімальної кількості навчальних прикладів для доступних типів ГНМ, формування навчальної вибірки.
Блок розробки архітектури ГНМ (БРА)	Визначення ефективних типів ГНМ, визначення найбільш ефективного типу ГНМ, розрахунок параметрів архітектури та навчання ефективних типів ГНМ.
Блок розпізнавання комп'ютерних вірусів (БРКВ)	Реалізація деобфускації програмного коду, розрахунок значень вхідних параметрів ГНМ, розпізнавання комп'ютерних вірусів та сигналізація про результати розпізнавання.

До складу блоку визначення параметрів входять:

- Модуль визначення діагностичних параметрів (ВДП) призначений для формування множини діагностичних параметрів, що будуть використовуватись для розпізнавання заданого типу комп'ютерних вірусів.

- Модуль визначення умов створення навчальної вибірки (УСНВ) призначений для визначення параметрів допустимої навчальної вибірки, визначення доступних баз даних, що можуть бути використані для формування навчальної вибірки.

- Модуль опису апаратних засобів (АПЗ) призначений для визначення параметрів апаратних засобів, що використовуються для розробки та застосування НМЗ.

- Модуль визначення часових параметрів (ВЧП) котрий призначений для розрахунку допустимих термінів щодо створення навчальної вибірки, навчання та побудови НММ.

До складу блоку розробки навчальних прикладів входять:

- Модуль оцінки обсягу навчальної вибірки (ОНВ) призначений для розрахунку обсягу навчальної вибірки необхідного для ефективного навчання доступних видів ГНМ.

- Модуль обробки навчальних прикладів (ОНП) застосовується для попередньої обробки параметрів навчальних прикладів для доступних видів ГНМ.

- Модуль формування навчальної вибірки (ФНВ) призначений для формування навчальної вибірки для кожного із допустимих типів ГНМ.

- Модуль перевірки навчальних даних (ПНД) призначений для перевірки якості обробки навчальних прикладів.

До складу блоку розробки архітектури ГНМ входять:

- Модуль формування множини допустимих типів ГНМ (МДТ) в котрому на основі аналізу множини доступних типів ГНМ реалізується визначення множини допустимих типів.

- Модуль формування множини ефективних типів ГНМ (МЕТ) в котрому на основі аналізу множини допустимих типів ГНМ проводиться визначення множини ефективних типів.

- Модуль визначення найбільш ефективного типу ГНМ (НЕТ) в котрому на основі аналізу множини ефективних типів ГНМ розраховується найбільш

ефективний тип.

- Модуль визначення параметрів архітектури (ПА) призначений для визначення параметрів архітектури найбільш ефективного типу ГНМ.

- Модуль реалізації навчання (РН) призначений для навчання ГНМ з визначеною архітектурою.

До складу блоку розпізнавання комп'ютерних вірусів входять:

- Модуль отримання діагностичної інформації параметрів (ОДІ) призначений для реєстрації та первинної обробки діагностичних параметрів.

- Модуль деобфускації програмного коду (ДПК) призначений для відображення процесу функціонування програмного забезпечення у вигляді графу залежностей і станів та перетворення параметрів цього графу до вигляду сприйнятого для подачі в ГНМ.

- Модуль визначення вхідних параметрів (ВП) призначений для розрахунку значень вхідних параметрів ГНМ.

- Модуль реалізації розпізнавання (РР) призначений для реалізації розпізнавання комп'ютерних вірусів за допомогою ГНМ.

- Модуль сигналізації (СР) призначений для сигналізації про результати розпізнавання ПЗ.

Крім того, до складу розробленої НМС входить модуль управління системою (МУС). Таким чином в цілому НМС розпізнавання комп'ютерних вірусів складається із 19 окремих модулів.

Передбачено, що розроблена НМС може функціонувати в таких режимах:

- Окреслення умов експлуатації (РОУЕ).

- Визначення налаштувань (РВН).

- Розпізнавання комп’ютерних вірусів (РКВ).

- Зупинки (РЗ).

Переключення режимів функціонування реалізується за допомогою модулю управління системою (МУС).

Структура запропонованої НМС розпізнавання комп'ютерних вірусів

показана на рис. 4.5.

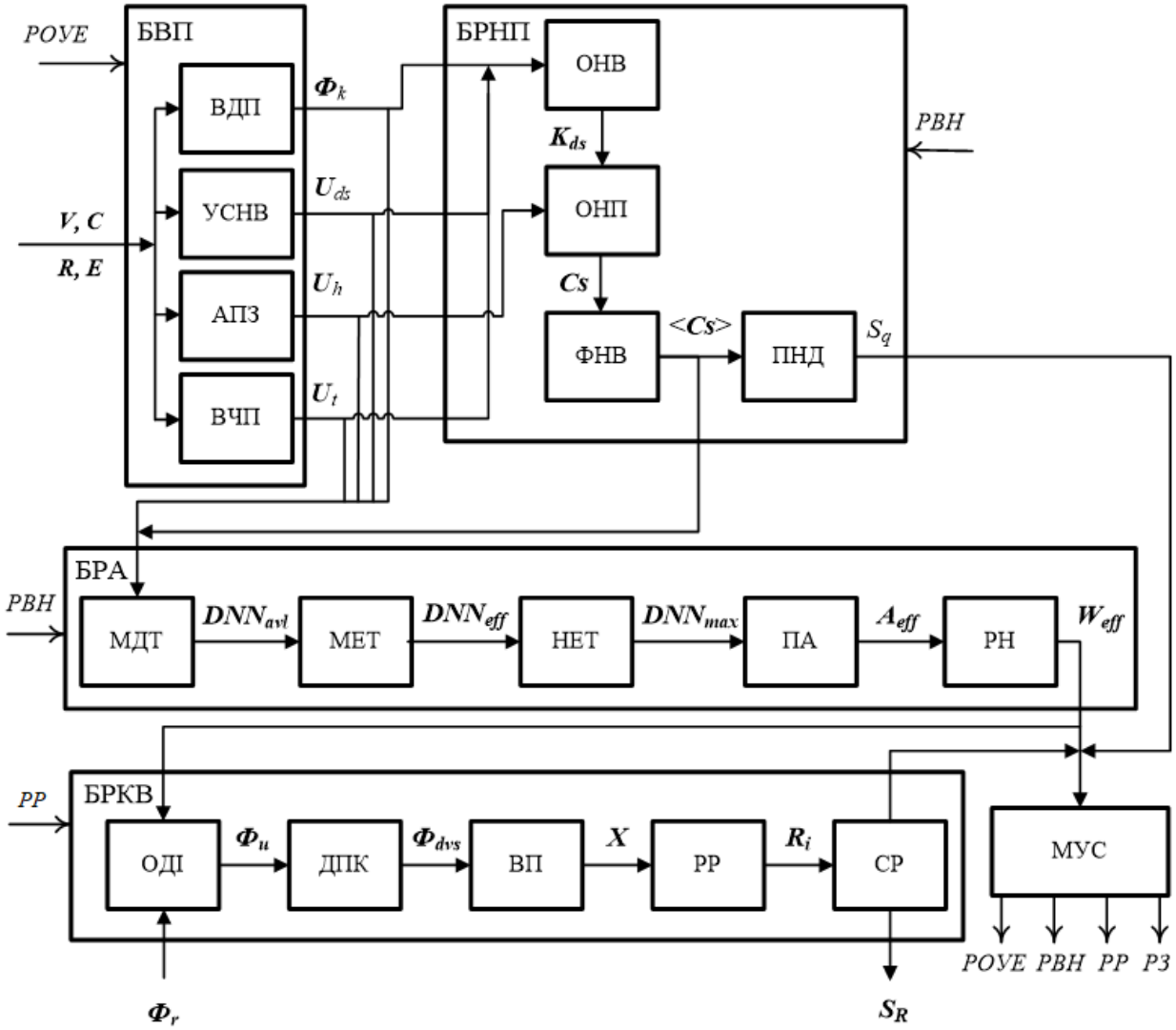

Рис. 4.5. Структура нейромережевої системи розпізнавання комп’ютерних вірусів

Передбачається, що першочерговим режимом функціонування, запропонованої НМС розпізнавання комп'ютерних вірусів, являється режим окреслення умов експлуатації.

Основним завданням режиму окреслення умов експлуатації є визначення параметрів, що характеризують очікувані умови застосування. В даному режимі спрацьовують модулі, що входять до складу БПВ. При цьому на вхід даного блоку входять: параметри вимог до НМС розпізнавання(***V***), характеристики доступного програмно-апаратного забезпечення системи (***C***),

ресурси (***R***), що виділяються на розробку системи та заданий термін її створення (***T***). Також на вхід БВП поступає множина експертних даних (***E***), що використовується для обробки ***V***, ***R*** та ***C***.

Результат спрацювання модулів даного блоку:

– ВДП – множина діагностичних параметрів ГНМ, що використовуються для розпізнавання заданих типів комп'ютерних вірусів ($\boldsymbol{\Phi}_k$). Для формування даної множини доцільно використовувати методи експертного оцінювання визначені в роботах [71, 91]. Попередня обробка діагностичних параметрів реалізується за допомогою виразів (3.3-3.5).

– УСНВ – кортеж параметрів, що характеризує умови створення навчальної вибірки ГНМ ($\boldsymbol{U}_{ds}$), компоненти якого визначаються при реалізації третього кроку першого етапу методу визначення архітектурних параметрів глибокої нейронної мережі.

– АПЗ – кортеж параметрів, що описують характеристики доступного апаратно-програмного забезпечення НСМ ($\boldsymbol{U}_h$). Вказані параметри визначаються за допомогою методів експертного оцінювання на основі даних табл. 2.1.

– ВЧП – кортеж параметрів, що визначають допустимі часові показники побудови та створення НМС розпізнавання комп'ютерних вірусів ($\boldsymbol{U}_t$). Компоненти кортежу визначаються виразами (2.26, 2.28, 2.29).

Якщо параметри, котрі характеризують умови розробки та використання НМС розпізнавання, не відповідають заданим обмеженням, то функціонування системи зупиняється. Якщо ж умови розробки та використання НМС розпізнавання відповідають заданим обмеженням, то система переводиться в режим визначення налаштувань. В цьому випадку спрацьовує БРНП та БРА.

Результат спрацювання модулів БРНП:

- ОНВ - множина значень мінімально допустимої кількості навчальних прикладів для кожного із допустимих видів ГНМ ($\boldsymbol{K}_{ds}$). Для розрахунку вказаних значень використовуються вирази (2.20-2.22, 2.44-2.46, 2.48-2.49).

- ОНП - множина навчальних прикладів з попередньо обробленими параметрами (***Cs***). Для попередньої обробки параметрів навчальних прикладів використовуються вирази (3.33-3.41).

- ФНВ - кортеж, що містить множини навчальних даних для кожного із допустимих типів ГНМ (<***Cs***>). Для формування вказаних множин використовуються вирази (3.41-3.43).

- ПНД – сигнал про якість обробки навчальних прикладів (S_q).

Для перевірки якості обробки, що реалізується в ПНД, відповідно до результатів [6], використовується константа Ліпшіца, що розраховується за допомогою виразу виду:

$$\begin{cases} K_{lp} = max\left(\frac{|y_i - y_j|}{\|x_i - x_j\|}\right), \\ i \neq j, \end{cases} \tag{4.1}$$

де K_{lp} - константа Ліпшіца, x_i, x_j – вектори вхідних параметрів для i-го та j-го навчального прикладу, y_i, y_j– вихідні параметри для i-го та j-го навчального прикладу.

Якість попередньої обробки вважається достатньою, якщо справджується вираз (4.2).

$$if\, K_{lp} \leq D_{Lp}. \tag{4.2}$$

В першому наближенні вважається, що $D_{Lp} = 50$.

Якщо вираз (4.2) не справджується, то слід, відповідно [41, 10], змінити методику нормалізації, структуру та обсяг навчальної вибірки.

Результат спрацювання модулів БРА:

- МДТ - множина допустимих типів ГНМ (***DNN**$_{avl}$*). Множина допустимих типів ГНМ формується за допомогою виразів (2.85-2.88, 2.89-2.92, 3.27).

- МЕТ - множина ефективних типів ГНМ ($\boldsymbol{DNN_{eff}}$). Множина ефективних типів ГНМ формується за допомогою виразів (3.30, 3.31).

- НЕТ – назва/назви найбільш ефективних типів ГНМ ($\boldsymbol{DNN_{max}}$). Для визначення використовуються вираз (3.32).

- ПА – множина архітектурних параметрів найбільш ефективного типу ГНМ ($\boldsymbol{A_{eff}}$). Елементи множини розраховуються за допомогою виразів (3.44-3.50). Якщо в результаті спрацювання модулю НЕТ визначено, що

$$K_{dnn_{max}} > 1, \tag{4.3}$$

де $K_{dnn_{max}}$ – кількість елементів множини $\boldsymbol{DNN_{max}}$.

Тобто визначено, що є декілька типів ГНМ з однаковим максимальним значенням функції ефективності, то

$$A_{eff} = \left\{A_1, A_2, \dots A_{K_{dnn_{max}}}\right\}. \tag{4.4}$$

- РН – множина вагових коефіцієнтів синаптичних зв'язків для найбільш ефективного типу ГНМ після реалізації навчання ($\boldsymbol{W_{eff}}$). Якщо умова (4.3) справджується, то

$$W_{eff} = \left\{W_1, W_2, \dots W_{K_{dnn_{max}}}\right\}. \tag{4.5}$$

Навчання реалізується за допомогою відповідного програмно-апаратного забезпечення.

Після спрацювання БРНП та БРА НМС за допомогою МУС переводиться в режим розпізнавання комп'ютерних вірусів. В цьому випадку спрацьовують модулі, що входять до складу БРКВ.

Результат спрацювання модулів БРКВ:

- ОДІ – множина первинно оброблених діагностичних параметрів

програмного забезпечення, що підлягає розпізнаванню ($\boldsymbol{\Phi_u}$). Обробка діагностичних параметрів реалізується за допомогою виразів (3.3-3.5). На вхід модулю подається множина зареєстрованих діагностичних параметрів ($\boldsymbol{\Phi_r}$).

- ДПК - множина параметрів, що відображають функціонування програмного забезпечення у вигляді графу залежностей та станів ($\boldsymbol{\Phi_{dvs}}$). Для формування множини використовуються вирази (3.1-3.16).

- ВП - множина вхідних параметрів ГНМ, котрі характеризують програмне забезпечення тип якого має бути розпізнано ($\boldsymbol{X}$). Для формування множини використовуються вирази (3.33-3.41).

- РР - розпізнаний тип програмного забезпечення ($\boldsymbol{R_i}$).

Якщо в результаті спрацювання модулю НЕТ, котрий входить до блоку БРА визначено хибність умови (4.3), то результат розпізнавання i-го типу ПЗ, що підлягає розпізнаванню, формується у вигляді виразу виду:

$$R_i = (i, Y), \tag{4.6}$$

де $\boldsymbol{Y}$ – вектор вихідних величин для найбільш ефективного типу ГНМ, i – номер типу ПЗ, що підлягає розпізнаванню.

Якщо в результаті спрацювання модулю НЕТ, котрий входить до блоку БРА визначено справедливість умови (4.3), то результат розпізнавання формується у вигляді виразу виду:

$$R_i = \left(i, Y_1, Y_2, \dots Y_{K_{dnn_{max}}}\right), \tag{4.7}$$

де Y_n – вектор вихідних величин для n-го типу ГНМ, що входить до множини $\boldsymbol{DNN_{eff}}$.

- СР - сигнал про результати розпізнавання наперед визначених типів ПЗ ($\boldsymbol{S_R}$).

Слід зазначити, що саме сигнал про результати розпізнавання комп'ютерних вірусів і є вихідним сигналом розробленої НМС.

Передбачається, що даний сигнал повинен подаватись на вхід системи антивірусного захисту для протоколювання подій та визначення та реалізації захисних заходів.

4.2. Експериментальна установка

Для забезпечення можливості проведення досліджень спрямованих на верифікацію запропонованих моделей і методів, що призначені для розпізнавання комп'ютерних вірусів розроблено експериментальну установку, яка є частковою реалізацією описаної в п. 4.1 НМС.

Основною частиною установки став створений за допомогою мови Python кросплатформуний програмний додаток «DNN analyzer», що дозволяє реалізувати ГНМ. В процесі розробки комплексу використана загальнодоступна бібліотека TensorFlow (розробка компанії Google) призначена для моделювання ГНМ. Головне вікно додатку показане на рис. 4.6.

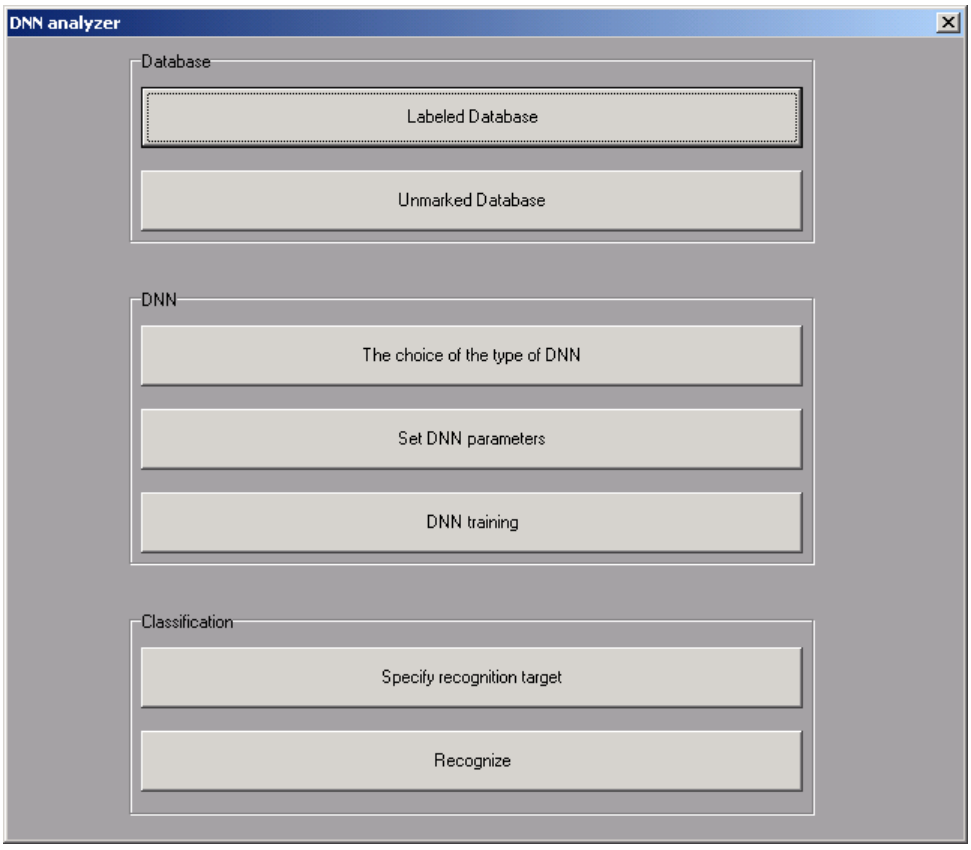

Рис. 4.6 Головне вікно додатку «DNN analyzer»

Управляючі елементи головного вікна розділені на три секції Database, DNN та Classification в котрих розміщені відповідні управляючі елементи (кнопки). Означені елементи інтерфейсу мають наступне призначення:

- Секція Database – вибір навчальної бази даних.

- Кнопка «Labeled Database» – ініціює режим вибору навчальної бази даних, що містить марковані приклади.

- Кнопка «Unmarked Database» – ініціює режим вибору навчальної бази даних, що містить не марковані приклади.

- Секція DNN – співвідноситься з реалізацією ГНМ.

- Кнопка «The choice of the type of DNN» - дозволяє обрати тип ГНМ.

- Кнопка «Set DNN parameters» - ініціює вибір архітектурних параметрів ГНМ.

- Кнопка «DNN training» - ініціює вибір параметрів навчання та запускає процес навчання ГНМ.

- Секція Classification - співвідноситься з застосуванням ГНМ для розпізнавання комп'ютерних вірусів.

- Кнопка «Specify recognition target» - ініціює режим вибору об’єктів, що підлягають розпізнаванню.

- Кнопка «Recognize» - запускає процес розпізнавання.

Використання програмного додатку «DNN analyzer» зводиться до послідовного застосування описаних елементів управління для вибору навчальної бази даних, вибору типу та задання архітектурних параметрів ГНМ, задання параметрів та реалізації навчання, вибору об'єкту, що має бути розпізнаний та власне реалізації процесу розпізнавання.

Особливості реалізації програмного додатку «DNN analyzer» полягали в застосуванні процедури онлайн навчання ГНМ та процедури формування вхідного шару ГНМ. Спрощена схема алгоритму процедури онлайн навчання показана на рис. 4.7. Спрощення полягає у відсутності на даній схемі елементів, що відповідають за часові обмеження процесу тренування та за особливості розділення навчальної вибірки на окремі блоки.

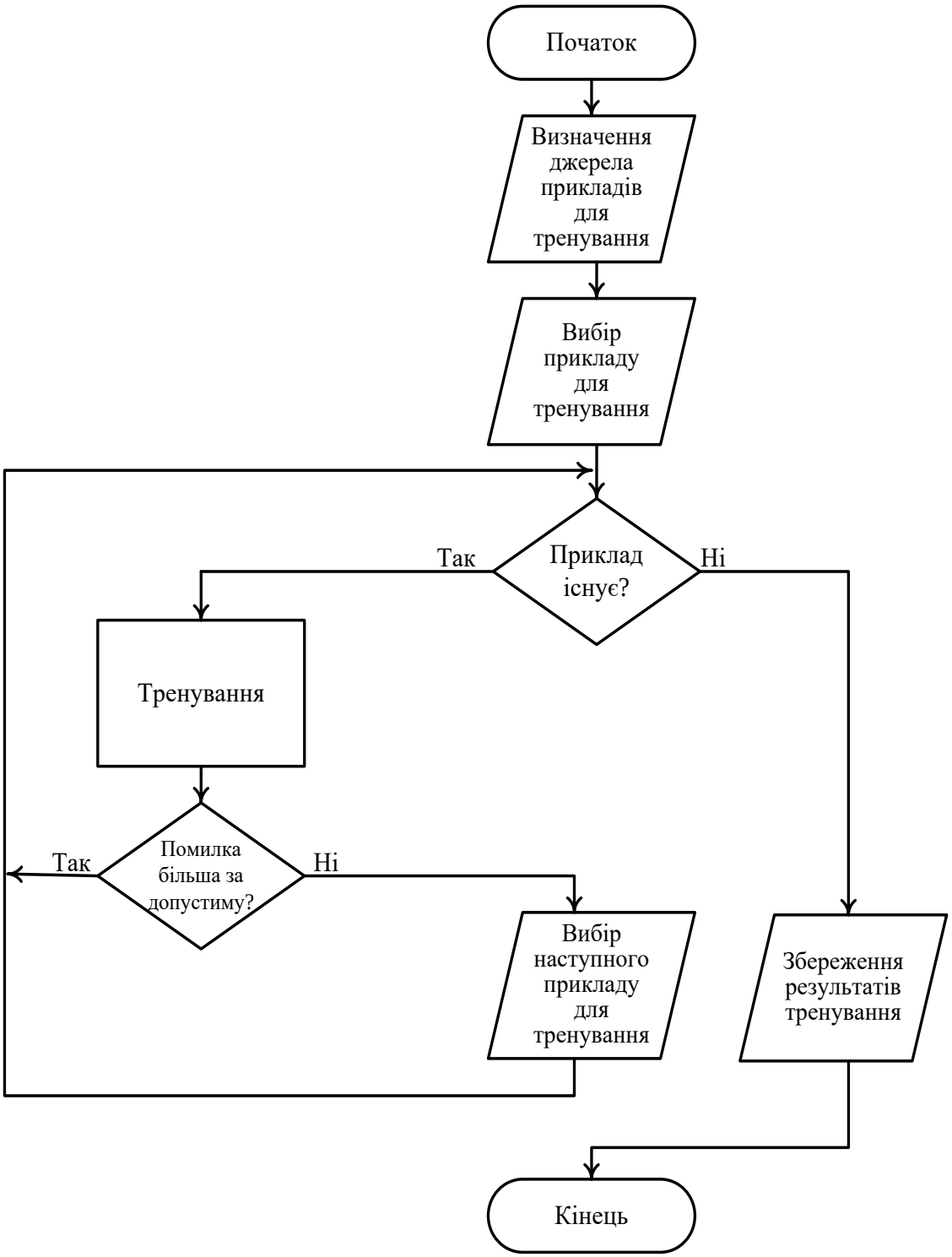

Рис. 4.7. Структура алгоритму навчання ГНМ

Схема алгоритму процедури формування вхідного шару ГНМ показана на рис. 4.8.

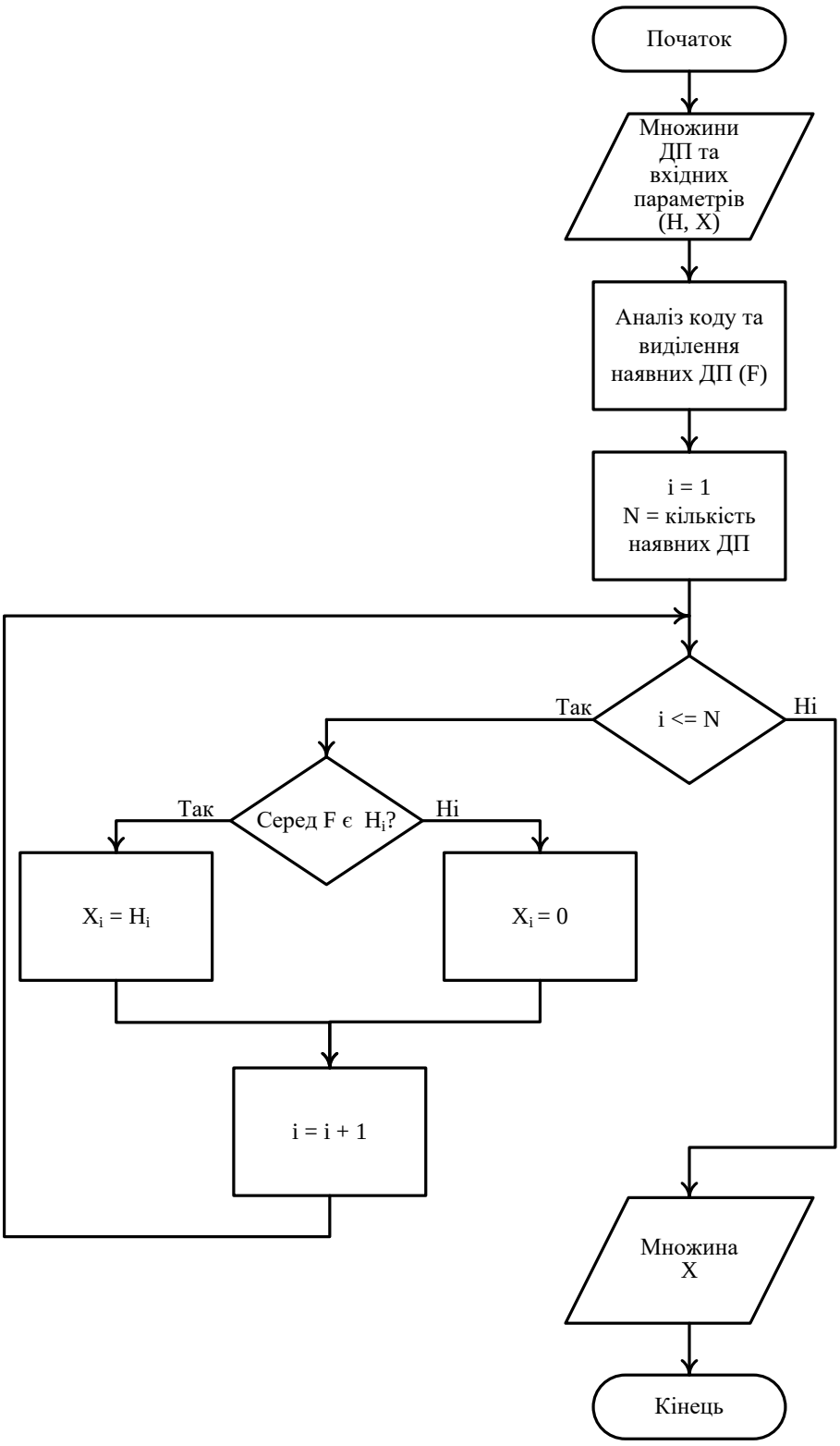

Рис. 4.8. Структура алгоритму формування вхідного шару ГНМ

Особливістю цієї процедури є використання БД наперед визначеної множини діагностичних параметрів та співвіднесення вхідних нейронів ГНМ з елементами вказаної БД. Це дозволяє уникнути можливих помилок при

формуванні вхідного шару ГНМ.

Структура алгоритму функціонування програмного додатку «DNN analyzer» показано на рис. 4.9.

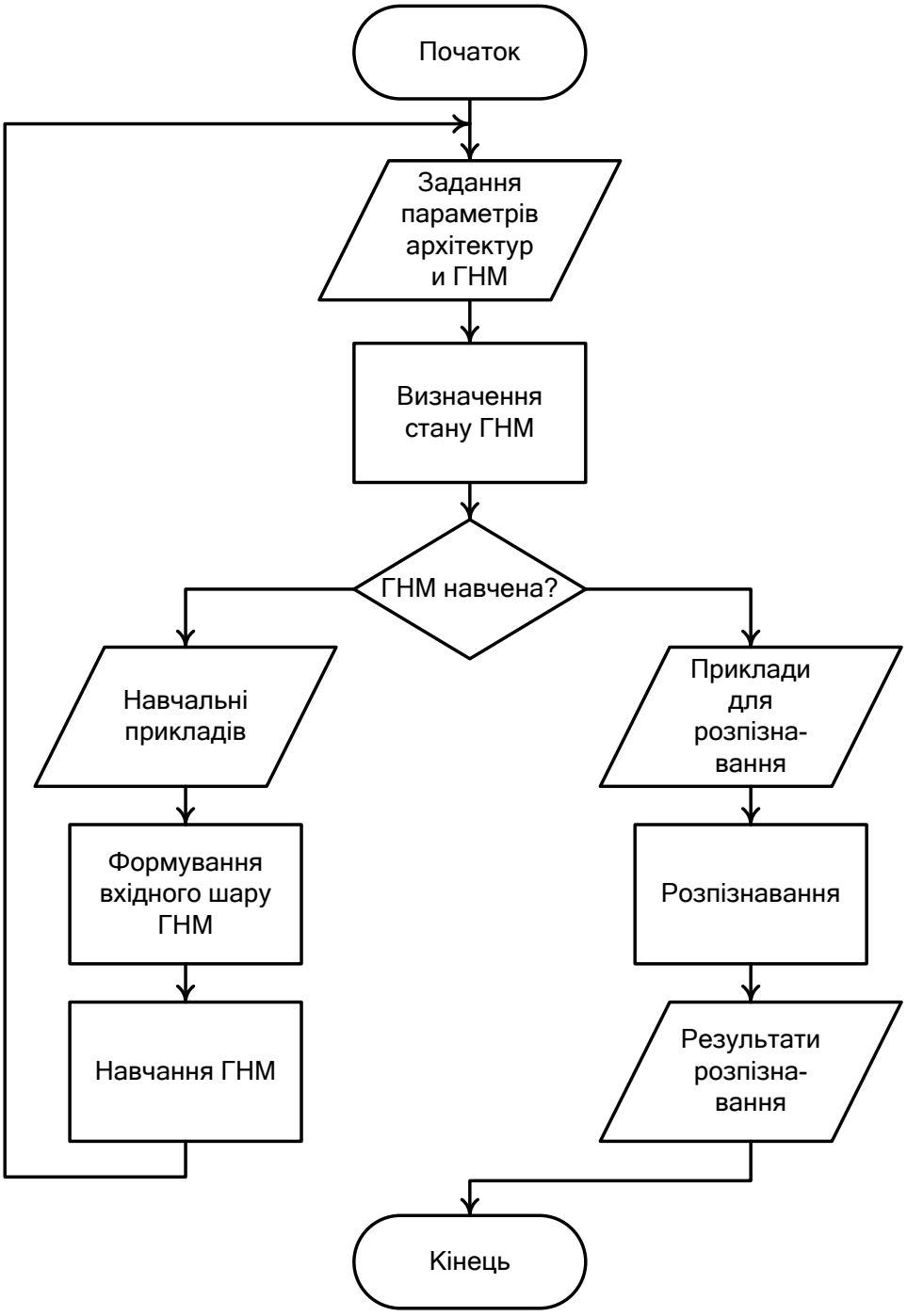

Рис. 4.9. Структура алгоритму функціонування програмного додатку «DNN analyzer»

В базовому випадку передбачено, що апаратне забезпечення

експериментальної установки повинно базуватись на персональному комп’ютері (AMD FX-9800P (2.7 - 3.6 ГГц) / RAM 32 ГБ / HDD 1 ТБ / AMD Radeon RX 540, 8 ГБ), котрий функціонує під управлінням операційної системи Windows 10. В подальшому можливо використовувати сервіси, що надають послуги в області хмарних обчислень.

4.3. Експериментальне дослідження

Розробка експериментальної установки дозволила перейти до досліджень, що пов’язані з доведенням ефективності розробленого методу визначення архітектурних параметрів ГНМ за умов:

- ГНМ використовується для розпізнавання Windows-орієнтованих комп’ютерних вірусів на основі аналізу використаних програмою потенційно небезпечних API-функцій операційної системи.

- На вхід ГНМ подається інформація, отримана в результаті сканування піддослідних файлів.

- Допустимий термін створення ГНМ складає 1 місяць. Тобто

$$\tau_{\text{avl}} = 2{,}6 \times 10^6 c. \tag{4.8}$$

- Для навчання та тестування ГНМ використовується опублікована компанією Microsoft БД комп’ютерних вірусів BIG-2015, опублікована за посиланням https://www.kaggle.com. Вказана БД дозволена для вільного використання в науково-практичних цілях для вирішення задач підвищення ефективності засобів антивірусного захисту комп'ютерних систем.

В БД BIG-2015 представлено приклади сигнатур 9 комп’ютерних вірусів. Для формування вказаних сигнатур використано програмний комплекс Interactive DisAssembler, що дозволяє вилучити із бінарного файлу метадані, які стосуються інструкцій мови Assembler, вмісту регістрів та даних і функцій, імпортованих із DLL. Характеристики БД BIG-2015 частково наведено в табл. 4.3.

Таблиця 4.3

Характеристика BIG-2015

Назва вірусу	**Кількість прикладів**
Ramnit	1541
Lollipop	2478
Kelihos_ver3	2942
Vundo	475
Simda	420
Tracur	751
Kelihos_ver1	398
Obfuscator.ACY	1228
Gatak	1013

При цьому застосування до дизасембльованого коду технології Flirt дозволяє визначити наявність в ньому потенційно небезпечних функцій управління розділами, управління файлами, роботи з реєстром, використання системної інформації, використання мережевих з'єднань, управління пам'яттю, використання сервісів, управління системою захисту об'єктів. В першому наближенні прийнято, що кількість таких функцій дорівнює 300. Таким чином, кількість вхідних параметрів ГНМ, а кількість вихідних параметрів. Тобто

$$N_x = 300, N_y = 10. \tag{4.9}$$

Оскільки для формування навчальної вибірки передбачається використання БД, то можливо вважати, що термін створення одного маркованого та немаркованого навчального прикладу дорівнює 0. Тобто:

$$\vartheta_w = 0, \vartheta_n = 0. \tag{4.10}$$

Також визначено, що приведена тривалість однієї навчальної ітерації ГНМ становить

$$\acute{\lambda} = 10^{-7}\text{с.}. \tag{4.11}$$

Підставивши дані (4.8-4.11) в вирази (2.85-2.88, 3.26), розраховано термін створення кожного із доступних типів ГНМ: $t_{dnn1,db} = 3721\text{с}$, $t_{dnn2,db} = 37200\text{с}$, $t_{dnn3,db} = 186\text{с}$, $t_{dnn4,db} = 3720\text{с}$. Оскільки для всіх типів ГНМ цей термін менший від допустимого, тобто справджуються умови, визначені виразами (2.89-2.92, 3.27), то доцільність використання ГНМ можна вважати доведеною.

Наступний етап розробки ГНМ було присвячено визначенню найбільш ефективного типу ГНМ. Для цього на за допомогою експертного методу парного порівняння було для кожного допустимого типу ГНМ визначено значення кожного із критеріїв ефективності, наведених в табл. 2.2. Також було проведене визначення значущості кожного із вказаних критеріїв. Отримані результати представлені в табл. 4.4 та в табл. 4.5. При розрахунках враховано, що критерії ефективності мають безрозмірний характер. Для *i*-го типу ГНМ значення *k*-го критерію дорівнює 1, якщо *k*-та вимога в основному забезпечується в цьому типі ГНМ, дорівнює 0,5, якщо забезпечується частково і дорівнює 0, якщо не забезпечується.

Таблиця 4.4

Вагові коефіцієнти критеріїв ефективності ГНМ

α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9
0,08	0,07	0,02	0,05	0,05	0,14	0,03	0,05	0,05
α_{10}	α_{11}	α_{12}	α_{13}	α_{14}	α_{15}	α_{16}	α_{17}	α_{18}
0,02	0,05	0,05	0,07	0,07	0,05	0,05	0,05	0,05

Підставивши дані табл. 4.4 та табл. 4.5 в вираз (3.30) розраховано, що значення функції ефективності для допустимих тидів ГНМ, дорівнюють H_{dnn1}=0.72, H_{dnn2}=0.77, H_{dnn3}=0.6, H_{dnn4}=0.58. З використанням виразу (3.32) визначено, що найбільш ефективною архітектурою є ГНМ з можливістю навчання в процесі експлуатації.

Таблиця 4.5

Значення критеріїв ефективності для допустимих типів ГНМ

Критерій	Тип ГНМ			
	$\boldsymbol{dnn}_1$	$\boldsymbol{dnn}_2$	$\boldsymbol{dnn}_3$	$\boldsymbol{dnn}_4$
H_1	0	0	0	1
H_2	0,5	1	0,5	0,5
H_3	1	1	1	0,5
H_4	0	1	0	0
H_5	0	0	1	1
H_6	1	0,5	0	0
H_7	1	1	0	0
H_8	1	1	1	0,5
H_9	1	1	1	0,5
H_{10}	1	0,5	0,5	0,5
H_{11}	1	1	0,5	0,5
H_{12}	0,5	0,5	1	1
H_{13}	1	0,5	1	0,5
H_{14}	1	1	0,5	0,5
H_{15}	1	1	0,5	0,5
H_{16}	1	1	1	0,5
H_{17}	1	0,5	1	1
H_{18}	1	0,5	0,5	0,5

Визначення типу ГНМ дозволило перейти до структурних параметрів. Кількість схованих нейронних шарів обрано з позицій максимального спрощення структури ГНМ і дорівнює $K_h = 2$. Для розрахунку кількості нейронів у кожному із схованих шарів використано вираз (3.44), компонентами якого є кількість вхідних нейронів, навчальних прикладів, вихідних нейронів та кількість схованих шарів нейронів. Відзначимо, що в БД кількість навчальних прикладів, що відповідають вірусам дорівнює 10868. В навчальній вибірці бажано, щоб кількість навчальних прикладів для комп'ютерних вірусів та безпечних програм була однаковою. При цьому формування навчальних прикладів, що стосуються безпечних програм не складає труднощів. Тому прийнято, що загальна кількість навчальних прикладів L=2×10868=21736. Підставивши в (3.44) N_x=300, N_y=10, K_h=2, L=221736 розраховано, що кількість нейронів у кожному із схованих шарів

дорівнює N_h=255. Відповідно результатів [25] прийнято, що в схованих шарах нейронів використовується функція активації типу ReLU, а у вихідному шарі функція активації типу Softmax.

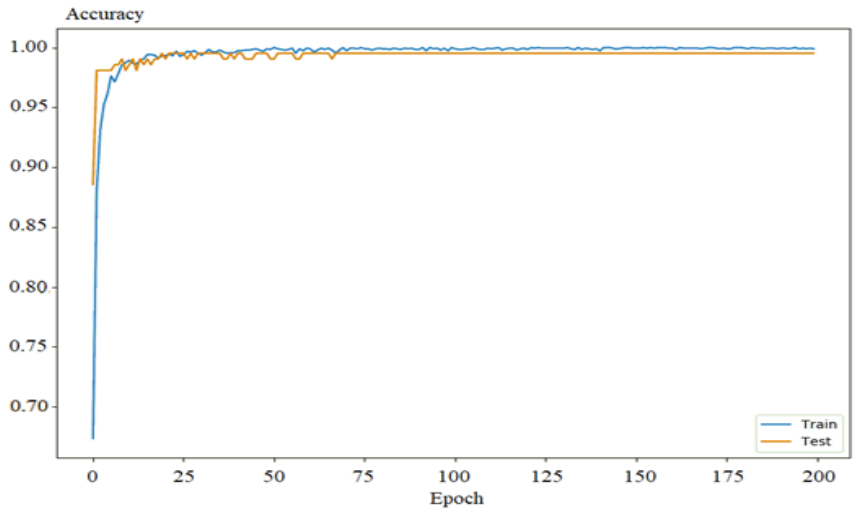

Рис. 4.10 Графіки залежності показника точності розпізнавання від кількості епох

Навчання розробленої ГНМ проводилось на протязі 100 епох. Графіки залежностей показників точності та втрат на тренувальних та тестових даних від кількості епох навчання показані на рис. 4.10 та рис. 4.11.

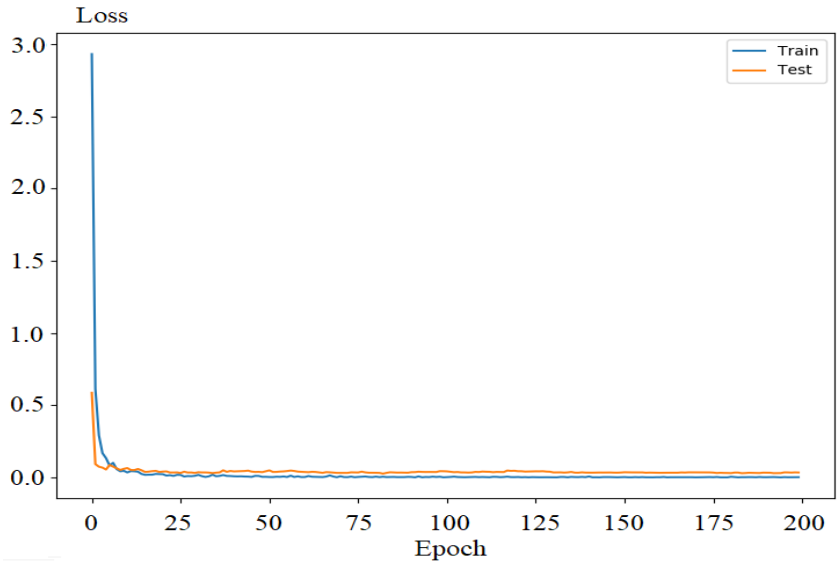

Рис. 4.11 Графіки залежності показника втрат при розпізнаванні від кількості епох

Зазначимо, що вказані показники точності та втрат розраховувались за допомогою виразів (1.2.7, 1.2.8). Після навчання на вхід ГНМ із БД BIG-2015 були подані приклади, що не використовувались при навчанні. Похибка розпізнавання для різних типів вірусів показана на рис. 4.12. Аналіз рис. 4.12 вказує на те, що найбільша похибка розпізнавання характерна для вірусів Simda, Tracur та Vundo. Це можна пояснити невеликою кількістю навчальних прикладів, що відповідають цим вірусам. При цьому середня похибка розпізнавання всіх видів вірусів дорівнює 0,036, що відповідає похибці сучасних антивірусних засобів. Також слід зазначити, що за рахунок використання запропонованого методу проектування архітектури ГНМ вдалось уникнути довготривалих чисельних експериментів, спрямованих на визначення доцільності її використання і на визначення її структурних параметрів, та приблизно в 1,5 рази зменшити обчислювальні витрати, пов’язані з визначенням вказаних архітектурних параметрів.

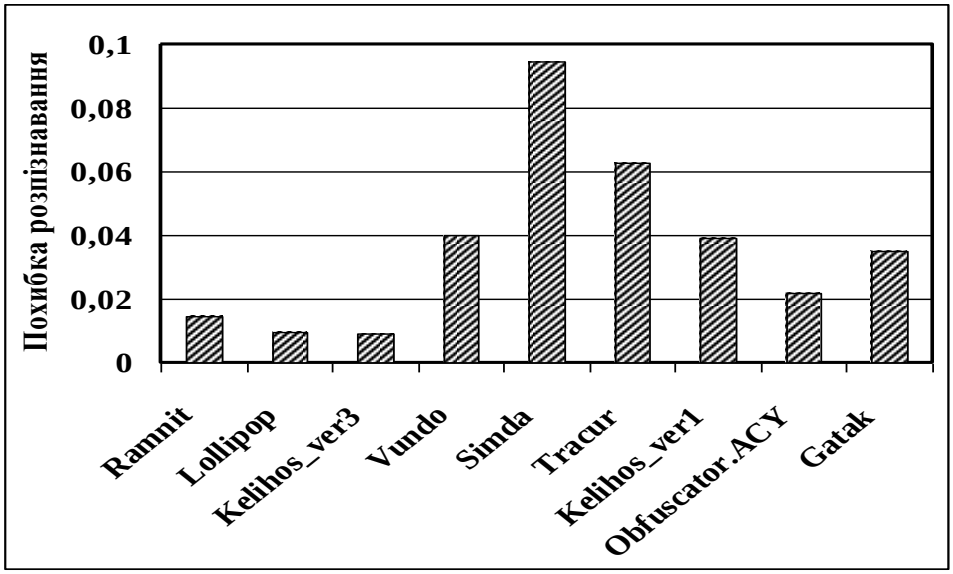

Рис. 4.12. Похибка розпізнавання

На заключному етапі досліджень проведено розрахунки, спрямовані на оцінювання ефективності запропонованого методу нейромережевого розпізнавання комп’ютерних вірусів. Для цього використаний загальновизнаний метод розрахунку ефективності НМЗ оцінювання

параметрів безпеки інформаційних систем та визначені критерії ефективності, що характеризують особливості сучасних НМЗ розпізнавання комп'ютерних вірусів. Математичне забезпечення цього методу становлять вирази (1.4.1, 1.4.2).

Основні результати застосування вказаного методу представлені в табл. 4.6. У процесі досліджень прийняті величини коефіцієнтів значущості параметрів ефективності α={0,1; 0,1; 0,1; 0,1; 0,1; 0,2; 0,1; 0,1; 0,1}. Відзначимо, що збільшення коефіцієнта значущості для R_6 пояснюється тим, що виконання відповідної процедури визначає принципову можливість навчання НМЗ за допомгою експертних знань.

Результати проведених розрахунків вказують на те, що ефективність розробленого НМЗ приблизно в 1,14 рази вища ніж у подібних відомих засобів.

Таблиця 4.6

Оцінка ефективності нейромережевих засобів розпізнавання комп'ютерних вірусів

№	Назва засобу	Параметр									
		R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_Σ
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
1	МПФС	1	1	0	0	0	1	0	0	1	0,4
2	ПКД	1	0	0	0	0	0	0	1	0	0,2
3	ПРМВ	1	1	0	0	0	0	0	0	0	0,2
4	КСВВ	1	1	0	0	0	0	0	0	1	0,3
5	ЕССАЗ	1	1	0	0	0	0	0	0	0	0,2
6	АМФНМ	1	1	0	0	0	0	0	0	1	0,3
7	МВФПК	1	1	0	0	0	0	0	0	0	0,2
8	ПВПВП	1	1	0	0	0	0	0	1	0	0,3
9	МНАНС	1	1	0	0	0	0	0	0	0	0,2
10	ПВДП	1	1	0	0	0	0	0	0	0	0,2
11	НСРА	1	1	0	1	1	0	0	0	1	0,5
12	МЗЗНМ	1	1	0	1	1	0	0	0	1	0,5
13	СНМА	1	1	0	1	1	0	0	0	1	0,5
14	НММПА	1	1	0	0	1	1	0	0	0	0,5
15	НМОПБ	1	1	0	1	1	1	0	0	1	0,7
16	СНРКВ	1	1	0	0	0	0	0	0	0	2
17	**НМРКВ**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**0,8**

Таким чином, результати досліджень підтверджують можливість підвищення ефективності розпізнавання комп’ютерних вірусів за рахунок застосування розроблених НММ та НМЗ. Також визначено, що основним напрямком удосконалення створеної НМС розпізнавання комп'ютерних вірусів є розробка методу для навчання НММ за допомогою експертних правил.

4.4. Висновки до четвертого розділу

Даний розділ присвячений вирішенню науково-практичної задач розробки нейромережевої системи розпізнавання комп'ютерних вірусів і проведення експериментальних досліджень, спрямованих на підтвердження достовірності основних результатів дисертаційної роботи. Основні результати розділу:

- Отримала подальший розвиток архітектура нейромережевої системи розпізнавання комп'ютерних вірусів, в якій на відміну від відомих передбачено використання модулів визначення найбільш ефективного типу глибокої нейронної мережі та деобфускації програмного коду, що забезпечує достатню точність розпізнавання і адаптацію до умов розробки і застосування.

- Розроблено експериментальну установку, яка забезпечує можливість проведення експериментів, спрямованих на перевірку достовірності основних результатів дисертаційної роботи.

- Проведені дослідження дозволяють стверджувати:

1. При очікуваних умовах застосування розроблена нейромережева система дозволить забезпечити помилку розпізнавання комп'ютерних вірусів в межах 0,036, що знаходиться на рівні кращих систем аналогічного призначення.

2. Використання розробленого методу нейромережевого розпізнавання комп'ютерних вірусів дозволяє приблизно в 1,14 разів підвищити ефективність нейромережевих засобів розпізнавання мережних кібератак.

ВИСНОВКИ

Результатом виконаної дисертаційної роботи є розв’язання актуальної науково-практичної задачі підвищення ефективності протидії комп’ютерним вірусам, за рахунок розробки і дослідження нових нейромережевих моделей, методів і засобів розпізнавання комп’ютерних вірусів, здатних оперативно пристосовуватись до умов використання і реагувати на виникнення нових видів вірусів.

У процесі виконання дисертаційної роботи отримані такі вагомі результати:

1. Проведено аналіз сучасних нейромережевих моделей та методів розпізнавання комп’ютерних вірусів, що показав наявність низки недоліків, пов’язаних з високою потребою в обчислювальних ресурсах, низькою адаптованістю до проведення аналізу обфускованого програмного коду та недостатньою ефективністю розпізнавання. В результаті виконаного аналізу обґрунтована перспективність застосування в контурі розпізнавання систем антивірусного захисту нейромережевих засобів. При цьому показано можливість застосування нейромережевих моделей як в поведінкових аналізаторів, так і при використанні сигнатурного аналізу.

2. Вперше розроблено концептуальну модель оцінювання глибоких нейронних мереж, яка дозволяє визначити множину сучасних нейромережевих моделей для побудови ефективних антивірусних засобів.

3. Вперше розроблено модель формування параметрів навчальних прикладів глибокої нейронної мережі, яка дозволяє будувати засоби нейромережевого аналізу обфускованого програмного коду.

4. Вперше розроблено метод визначення архітектурних параметрів глибокої нейронної мережі, призначеної для розпізнавання комп’ютерних вірусів, що дозволяє сформувати набір величин, які забезпечують пристосованість такої мережі до визначених умов застосування. Використання розробленого методу проектування дозволяє приблизно в 1,5 рази зменшити

обчислювальні витрати, пов’язані з визначенням значень архітектурних параметрів.

5. Отримав подальший розвиток метод нейромережевого розпізнавання комп’ютерних вірусів, який забезпечує достатню похибку розпізнавання при різних умовах застосування з урахуванням обмежень щодо створення навчальної вибірки та обмежень щодо обчислювальних ресурсів системи антивірусного захисту. Показано, що ефективність методу приблизно в 1,14 рази вища, ніж у подібних методів розпізнавання.

6. На базі запропонованого методу нейромережевого розпізнавання комп’ютерних вірусів розроблено алгоритмічне забезпечення та відповідна програмна модель системи, що дозволила з достатньою точністю розпізнавати комп’ютерні віруси та забезпечити оперативність створення алгоритмів функціонування апаратно-програмних засобів захисту інформації.

7. Експериментальне дослідження програмної моделі системи, а також впровадження та успішне практичне використання відповідних розробок підтвердило достовірність теоретичних положень та гіпотез, практичних розробок і висновків дисертаційної роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A. Damodaran, Di Troia F., C.A. Visaggio, T.H. Austin, M. Stamp, «A comparison of static, dynamic, and hybrid analysis for malware detection», *Journal of Computer Virology and Hacking Techniques*, 13(1), 1-12, 2017.

2. A. Sujyothi, S. Acharya, «Dynamic Malware Analysis and Detection in Virtual Environment», *International Journal of Modern Education and Computer Science*, 2017, Vol. 9, No. 3, p. 48. DOI: 10.5815/ijmecs.2017.03.06.

3. A.F. Agarap, F.J.H. Pepito, Towards Building an Intelligent Anti-Malware System: A Deep Learning Approach using Support Vector Machine (SVM) for Malware Classification, arXiv preprint arXiv:1801.00318. (2017).

4. A.H. Sung, J. Xu, P. Chavez, S. Mukkamala, «Static Analyzer of Vicious Executables (SAVE)», I*n Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC '04)*, IEEE Computer Society, Washington, DC, USA, 326–334. 2004.. https://doi.org/10.1109/CSAC.2004.37

5. B. Aitchanov, A. Korchenko, I. Tereykovskiy, I. Bapiyev, «Perspectives for using classical neural network models and methods of counteracting attacks on network resources of information systems», *News of the national academy of sciences of the republic of Kazakhstan series of geology and technical sciences*, Vol. 5, № 425 (2017), pp. 202 – 212.

6. B. Aitchanov, I. Bapiev, I. Terejkowski, L. Terejkowska, V. Pogorelov, «Calculation of expected output signal of neural network model for detecting of cyber-attack on network resources», *Information Technologies, Management and Society, The 15th International Scientific Conference Information Technologies and Management,* pp. 59–62, 2017.

7. B. Akhmetov, V. Lakhno, B. Akhmetov, Z. Alimseitova, «Development of sectoral intellectualized expert systems and decision making support systems in cybersecurity», Proceedings of the Computational Methods in Systems and Software, pp. 162-171.

8. B. Kolosnjaji, A. Zarras, G. Webster, C. Eckert, «Deep learning for classification of malware system call sequences», In Byeong Ho Kang and Quan Bai, editors, AI 2016: Advances in Artificial Intelligence, pp. 137–149, 2016.

9. B. Yadegari, B. Johannesmeyer, B. Whitely, and S. Debray. 2015. A

Generic Ap- proach to Automatic Deobfuscation of Executable Code. In 2015 IEEE Symposium on Security and Privacy, pp. 674–691. https://doi.org/10.1109/SP.2015.47

10. Chih-Ta Lin, Nai-Jian Wang, Han Xiao, and Claudia Eckert, «Feature selection and extraction for malware classification», *Journal of Information Science and Engineering*, 31:965–992, 2015.

11. D. Bilar, «Opcodes as predictor for malware», International Journal of Electronic Security and Digital Forensics, 1(2):156–168, 2007.

12. D. Bruschi, L. Martignoni, M. Monga. «Using Code Normalization for Fighting Self-Mutating Malware». Technical Report. – Milan: University, 2016. 14 p.

13. D. Ucci, L. Aniello, R. Baldoni, «Survey on the Usage of Machine Learning Techniques for Malware Analysis», *arXiv preprint arXiv*:1710.08189, 2018.

14. DLLMiner: structural mining for malware detection, Narouei M. [et al.], Security and communication networks, Vol. 8, no.18. pp. 3311-3322, 2015.

15. E. Filiol, «Metamorphism, Formal Grammars and Undecidable Code Mutation», *PWASET (World Academy of Science, Engineering and Technology)*, Vol. 20, pp. 1–7, 2007.

16. E. Rezende, G. Ruppert, T. Carvalho, A. Theophilo, F. Ramos, de Geus P., «Malicious Software Classification Using VGG16 Deep Neural Network's Bottleneck Features», *In Information Technology-New Generations*, pp. 51-59, Springer, Cham, 2018.

17. Eureka: A Framework for Enabling Static Malware Analysis, Sharif M. [et al.] ESORICS 2008: Computer Security - ESORICS 2008 pp 481-500.

18. F.A. Omotayo Dlamini and M. Jonathan, «Blackledge Asiru. Application of Artificial Intelligence for Detecting Derived Viruses», *16th European Conference on Cyber Warfare and Security (ECCWS 2017) (Dublin 2017 June 29-30), University College Dublin*, pp. 217-227.

19. F. Shaheen, B. Verma, and M. Asafuddoula, «Impact of Automatic Feature Extraction in Deep Learning Architecture», *In Proceedings of the International Conference on Digital Image Computing Techniques and Applications, (Queensland, Australia)*, 2016.

20. G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, «Self-normalizing neural networks», *Advances in neural information processing systems*, 2017.

21. G. Sigletos, G. Paliouras, C.D. Spyropoulos, «Combining Information Extraction Systems Using Voting and Stacked Generalization», *Journal of Machine Learning Research*, 6 (2005), pp. 1751-1782.

22. G.E. Dahl, J.W. Stokes, Li Deng, Dong Yu, «Large-scale malware classification using random projections and neural networks», *In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE*, pp. 3422–3426.

23. G.E. Hinton, R.R. Salakhutdinov, «Reducing the dimensionality of data with neural networks», *Science*, 313 (5786), pp. 504-507, 2006.

24. H.S. Anderson, A. Kharkar, B. Filar, P. Roth, «Evading machine learning malware detection. Black Hat», 2017.

25. Hu. Zhengbing, I. Tereykovskiy, L. Tereykovska, V. Pogorelov, «Determination of Structural Parameters of Multilayer Perceptron Designed to Estimate Parameters of Technical Systems», *International Journal of Intelligent Systems and Applications(IJISA)*, 2017, Vol.9, No.10, pp. 57-62. (SCOPUS) DOI: *10.5815/ijisa.2017.10.07, ISSN 2074-9058.*

26. I. Dychka, I. Tereikovskyi, L. Tereikovska, V. Pogorelov, S. Mussiraliyeva, «Deobfuscation of Computer Virus Malware Code with Value State Dependence Graph», *Advances in Computer Science for Engineering and Education. ICCSEEA 2018. Advances in Intelligent Systems and Computing*, Vol 754. Springer, Cham, pp.370-379. (SCOPUS) DOI: *https://doi.org/10.1007/978-3-319-91008-6_37*

27. I. Santos, F. Brezo, X. Ugarte-Pedrero, P.G. Bringas, «Opcode Sequences as Representation of Executables for Data-mining-based Unknown Malware Detection», *Published in Information Sciences: an International Journal*, Vol. 231, May, 2013, pp. 64-82.

28. I.M. Bapiyev, B.H. Aitchanov, I.A. Tereikovskyi, L.A. Tereikovska, A.A. Korchenko, «Deep neural networks in cyber attack detection systems», International Journal of Civil Engineering and Technology (IJCIET) Volume 8, Issue 11, November 2017, pp. 1086–1092.

29. J. Bai, J. Wang, G. Zou, «A Malware Detection Scheme Based on Mining Format Information», *The Scientific World Journal*, 2014.

30. J. Drew, T. Moore, M. Hahsler, «Polymorphic malware detection using sequence classification methods», *In Security and Privacy Workshops (SPW)*, 2016 IEEE. IEEE, pp. 81–87.

31. J. Rabek, R. Khazan, S. Lewandowski, R. Cunningham, «Detection of injected, dynamically generated, and obfuscated malicious code», *Proceedings of the 2003 ACM Workshop on Rapid Malcode*, pp. 76–82.

32. J. Saxe, K. Berlin, «Deep neural network based malware detection using two dimensional binary program features», *In Malicious and Unwanted Software (MALWARE)*, 2015 10th International Conference on. IEEE, pp. 11–20.

33. J. Saxe, K. Berlin, «Deep neural network based malware detection using two dimensional binary program features», *In Malicious and Unwanted Software (MALWARE), 2015, 10th International Conference on*, pp. 11-20.

34. J.Z. Kolter, M.A. Maloof, «Learning to detect and classify malicious executables in the wild», *The Journal of Machine Learning Research*, 7:2721–2744, 2006.

35. K. Hahn, «Robust Static Analysis of Portable Executable Malware», *HTWK* Leipzig, 2014.

36. L. Nataraj, D. Kirat, B.S. Manjunath, G. Vigna, «Sarvam: Search and retrieval of malware», *In Proceedings of the Annual Computer Security Conference (ACSAC) Worshop on Next Generation Malware Attacks and Defense (NGMAD)*, 2013.

37. L. Personnaz, I. Guyon, G. Dreyfus, «Collective computational properties of neural networks: New learning mechanisms», *Phys. Rev. A.*, Vol. 34, no. 5, pp. 4217-4228, 1986.

38. LeCun, Y., Bengio, Y., Hinton, G. «Deep learning», *Nature 521*, pp. 436–444, 2015.

39. M. Christodorescu, Somesh Jha, «Static Analysis of Executables to Detect Malicious Patterns», *In Proceedings of the 12th Conference on USENIX Security Symposium, Volume 12 (SSYM'03), USENIX Association, Berkeley, CA, USA*, pp. 12–12, 2003. http://dl.acm.org/citation.cfm?id=1251353.1251365

40. M. Farrokhmanesh, A. Hamzeh, «A novel method for malware detection

using audio signal processing techniques», *In Artificial Intelligence and Robotics (IRANOPEN)*, 2016 (pp. 85-91).

41. M. Rhode, P. Burnap, K. Jones, «Early-stage malware prediction using recurrent neural networks», *Computers & Security*, 77, pp. 578-594, 2018.

42. M. Yousefi-Azar, V. Varadharajan, L. Hamey, U. Tupakula, «Autoencoder-based feature learning for cyber security applications», *In 2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 3854-3861.

43. M. Zolotukhin, T. Hamalainen, «Detection of zero-day malware based on the analysis of opcode sequences», 2014, pp. 386–391.

44. M.G. Schultz, E. Eskin, E. Z., S. J. Stolfo, «Data mining methods for detection of new malicious executables», *In Proceedings of the IEEE Symp. on Security and Privacy*, pp. 38-49, 2001.

45. Mahmood Youse-Azar, Vijay Varadharajan, Len Hamey, Uday Tupakula, «Autoencoder-based feature learning for cyber security applications», *In Neural Networks (IJCNN), 2017 International Joint Conference on. IEEE*, pp. 3854–3861.

46. Manjunath. Malware images: Visualization and automatic classification, Nataraj L. [et al.], *In Proceedings of the 8th International Symposium on Visualization for Cyber Security, ACM*, 2018, pp. 41-47.

47. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, «Dropout: a simple way to prevent neural networks from overfitting», *The Journal of Machine Learning Research*, 15(1):1929– 1958, 2015.

48. Novel feature extraction, selection and fusion for effective malware family classification, M. Ahmadi, D. Ulyanov, S. Semenov [et al.], *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, New York: ACM*, 2016, pp. 183-194.

49. O. Tereykovskiy, V. Pogorelov «Cyberattack recognition with radial basis function neural network», *Projekt interdyscyplinarny projektem XXI wieku*, Tom 2, pp. 255-262, 2017. (Колективна монографія)

50. O.E. David, N.S. Netanyahu, «Deepsign: Deep learning for automatic malware signature generation and classification», *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2015.

51. P.V. Zbitskiy, «Code mutation techniques by means of formal grammars and automa- tons», *Journal in Computer Virology, Paris: Springer*, 2009, Vol. 5, Num. 2. pp. 88–99.

52. Parametric equation for capturing dynamics of cyber-attack malware transmission with mitigation on computer network, A Falaye Adeyinka, E. S. Oluyemi, N. V. Adama [et al.], *International Journal of Mathematical Sciences and Computing (IJMSC)*, 2017, Vol. 3, No.4, pp. 37-51.

53. R. Pascanu, J.W. Stokes, H. Sanossian, M. Marinescu, A. Thomas, «Malware classification with recurrent networks», *In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 1916-1920, 2015.

54. Razvan Pascanu, Jack W Stokes, Hermineh Sanossian, Mady Marinescu, Anil Thomas, «Malware classification with recurrent networks», *In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on. IEEE*, pp. 1916–1920.

55. S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, T. Yagi, «Malware detection with deep neural network using process behavior», *In Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, Vol. 2, pp. 577-582.

56. T. Chen, C.C. Guestrin, XGBoost: A Scalable Tree Boosting System, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785-794, 2016.

57. T. Kim, B. Kang, M. Rho, S. Sezer, E.G. Im, «A Multimodal Deep Learning Method for Android Malware Detection Using Various Features», *IEEE Transactions on Information Forensics and Security*, 14(3), pp. 773-788, 2019.

58. T. Shmelova, Y. Sikirda, N. Rizun, V. Lazorenko, V. Kharchenko, «Machine Learning and Text Analysis in an Artificial Intelligent System for the Training of Air Traffic Controllers», 2019, pp. 1–50.

59. V. Pogorelov, M. Karpinski, E. Ivanchenko, «Method of neural networks utilization for malware recognition», *The 10 th International Scientific Conference «ITSec» March*, pp. 58, 2020.

60. Virus detection using artificial neural networks, Shah S., H. Jani, S. Shetty [et al.], *International Journal of Computer Applications*, 2013, Vol. 84, No. 5, p. 17–23.

61. W. Huang, J.W. Stokes, «MtNet: a multi-task neural network for dynamic malware classification», *In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 399-418, Springer, Cham, 2016.

62. Wenyi Huang, Jack W Stokes, «MtNet: a multi-task neural network for dynamic malware classification», *In Detection of Intrusions and Malware, and Vulnerability Assessment. Springer*, pp. 399–418, 2016.

63. Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2), pp. 157-166, 1994.

64. Y. Otsuki, M. Ichino, S. Kimura, M. Hatada, and H. Yoshiura, «Evaluating payload features for malware infection detection", J. *of Inform. Process.*, Vol. 22, no. 2, pp. 376–387, 2014.

65. Y. Ye, L. Chen, D. Wang, T. Li, Q. Jiang, and M. Zhao, «Sbmds: an interpretable string based malware detection system using svm ensemble with bagging», *Journal in Computer Virology*, 5 (4):283–293, 2009.

66. Ye. Yanfang, Li Tao, Adjeroh Donald, S Sitharama Iyengar, «A survey on malware detection using data mining techniques», *ACM Computing Surveys (CSUR)* 50, 3 (2017), p. 41.

67. А. Артеменко, В. Головко, «Анализ нейросетевых методов распознавания компьютерных вирусов», *Молодежный инновационный форум ИНТРИ, 2010, Минск: ГУ «БелИСА»*, 239 с.

68. А. Корченко, И. Терейковский, Н. Карпинский, С. Тынымбаев, «Нейросетевые модели, методы и средства оценки параметров безопасности Интернет-ориентированных информационных систем», *Монографія, К. :ТОВ «Наш Формат»*, 2016, 275 с.

69. А.И. Иванов, «Нейросетевая защита конфиденциальных биометрических образов гражданина и его личных криптографических ключей», *Монография, Пенза: ПНИЭИ*, 2014, 57 с.

70. А.Ю. Киселевская, «Глубокие нейронные сети: автоматическое обучение распознаванию вредоносных программам. Генерация и классификация подписей», *Молодой ученый*, 2017, №. 47(181), С. 15-18.

71. Б. Айтчанов, И. Бапиев, А. Корченко, В. Погорелов, Л. Терейковская, «Концептуальная модель обеспечения эффективности нейросетевого распознавания кибератак», *Труды международной научно-практической конференции «Математические методы и информационные технологии макроэкономического анализа и экономической политики», посвященной празднованию 80-летнего юбилея академика НАН РК Абдыкаппара Ашимовича Ашимова*, *11-12*, С. 321–325, 2017.

72. Б. Бенджамин, Б. Ребекка, О. Тони, «Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений обработки естественного языка», *СПб.: Питер,* 2019, 368 с.

73. Б.С. Ахметов, А.И. Иванов, В.А. Фунтиков, А.В. Безяев, Е.А. Малыгина, «Технология использования больших нейронных сетей для преобразования нечетких биометрических данных в код ключа доступа», *Монография, Алматы: ТОО «Издательство LEM»*, 2014, 144 с.

74. В. Погорелов, «Проблематика використання нейромережевих систем розпізнавання кібератак», *Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво»*, №27, С. 67–74, 2017.

75. В. Погорєлов, «Використання графу залежностей значень і станів у задачі розпізнавання поліморфних комп’ютерних вірусів», *Стан та удосконалення безпеки інформаційно-телекомунікаційних систем (SITS’ 2020)*, Миколаїв, 2020, С. 34-35.

76. В. М. Міхайленко, Л. О. Терейковська, І. А. Терейковський., Б.Б. Ахметов, «Нейромережеві моделі та методи розпізнавання фонем в голосовому сигналі в системі дистанційного навчання, Монографія», *К.: ЦП «Компринтр»*, 2017, 252 с.

77. В. Погорелов, «Дослідження нейромережевих засобів розпізнавання кібератак на мережеві ресурси інформаційних систем», *Системні технлогії,* №5, С. 61–69, 2017.

78. В. Погорелов, «Нейромережевий метод розпізнавання комп’ютерних вірусів», *VI Міжнародна науково-практична конференція «Актуальні питання забезпечення кібербезпеки та захисту інформації»*, 2020, С. 88-93.

79. В.А. Хорошко, А.А. Чекатков, «Методы и средства защиты информации», *Киев: Издательство «Юниор»*, 2003, 504 с.

80. В.Д. Козюра, В.О. Хорошко, М. Є. Шелест Козюра В.Д., «Аналіз кібернетичної безпеки інформаційного суспільства», *Інформаційна безпека людини, суспільства, держави*, № 1, С. 163-170, 2017.

81. Державний стандарт України, Захист інформації. Технічний захист інформації. Терміни та визначення. ДСТУ 3396.2-97.

82. Е. Путин, А. Тимофеев, «Классификатор для статического обнаружения компьютерных вирусов, основанный на машинном обучении», *International Journal «Information Technologies & Knowledge»*, Vol. 8, № 2, 2014, pp. 103-112.

83. И.М. Бапиев, «Нейросетевые модели и методы противодействия атакам на сетевые ресурсы информационных систем», *Информационные системы Алматы, диссертация на соискание ученой степени доктора философии (PhD)*, 127 с, 2018.

84. I. Dychka, D. Chernyshev, I. Tereikovskyi, L. Tereikovska , V. Pogorelov, «Malware Detection Using Artificial Neural Networks», *Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing*, Vol. 938. Springer, Cham, pp.3-12, 2019. (SCOPUS) DOI: *https://doi.org/10.1007/978-3-030-16621-2_1*.

85. I. Tereikovskyi, V. Pogorelov, O. Tereikovskyi, «Determination of structural parameters of a multilayer cyber threat detection perceptron», *Aviation in the XXI-st Century,* 2018, pp. 3.3.1 – 3.3.4.

86. І. Терейковський «Нейромережевий поведінковий аналізатор антивірусної системи», *Захист інформації*, № 2, С. 67-70, 2012.

87. І. Терейковський, «Нейронні мережі в засобах захисту комп'ютерної інформації: монографія», *К.: ПоліграфКонсалтинг*, 2007, 209 с.

88. І. Терейковський «Вдосконалення алгоритму навчання багатошарового персептрону, призначеного для розпізнавання мережевих атак», *Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні*, Випуск 2(24), С. 65-70, 2012.

89. І. Терейковський, «Використання нейронних мереж при розпізнаванні макровірусів», *Правове, нормативне та метрологічне*

забезпечення системи захисту інформації в Україні, Випуск 2 (13), С. 176-183, 2006.

90. І. Терейковський, «Нейромережева методологія розпізнавання інтернет-орієнтованого шкідливого програмного забезпечення», *Безпека інформації*, Т. 19, № 1,С. 24-28, 2013.

91. І. Терейковський, «Нейромережеві моделі, методи і засоби оцінювання параметрів безпеки інтернет-орієнтованих інформаційних систем», *Дисертація д-ра техн. наук: 05.13.21, Нац. авіац. ун-т.*, Київ, 2015, 430 с.

92. І. Терейковський, О. Заріцький, Л. Терейковська, В. Погорелов, «Метод розробки архітектури глибокої нейронної мережі, призначеної для розпізнавання комп'ютерних вірусів», *Захист інформації*, Т. 20, № 3, С. 188-199, 2018.

93. Л. Терейковська, Є. Іванченко, В. Погорелов «Метод адаптації глибокої нейронної мережі до розпізнавання комп'ютерних вірусів», *Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво»*, Луцьк, Випуск № 35, С. 198-205, 2019.

94. Нормативний документ системи технічного захисту інформації. Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу НД ТЗІ 1.1-003-99.

95. О. Корченко, І. Терейковський, А. Білощицький, «Методологія розроблення нейромережевих засобів інформаційної безпеки Інтернет-орієнтованих інформаційних систем», *К.: ТОВ «Наш Формат»*, 2016, 249 с.

96. О. Г. Корченко, І. А. Терейковський, С. В. Казмірчук, «Верифікація нейромережевих методів розпізнавання кібератак», *Науково-технічний збірник «Управління розвитком складних систем» Київського національного університету будівництва і архітектури*, 2014, Випуск 17, С. 168-172.

97. О.Г. Руденко, Є.В. Бодянський, «Штучні нейронні мережі», *Навч. посіб., Харків: ТОВ «Компанія СМІТ»*, 2006, 404 с.

98. Оценка точности алгоритма распознавания вредоносных программ на основе поиска аномалий в работе процессов, М. В. Баклановский, А.Р., Ханов, К.М. Комаров [и др.], *Научно-технический вестник информационных технологий, механики и оптики*, 2016, Т. 16, №. 5, С. 823–830.

99. П.В. Збицкий, «Функциональная сигнатура компьютерных вирусов», *Доклады ТУСУРа*, № 1 (19), часть 2, июнь 2019, с. 75-76.

100. Р.Ю. Демина, «Особенности программной реализации алгоритмов методики формирования обучающего множества для бинарных классификаторов, используемых в антивирусном эвристическом статическом анализе», *Вестн. Астрахан. гос. техн. ун-та. Сер. управление, вычисл. техн. информ.*, 2017, номер 2, с. 62–68.

101. С. Николенко, А. Кадурин, Е. Архангельская, «Глубокое обучение», *СПб.: Питер*, 2018, 480 с.

102. С. Рассел, П. Норвиг, «Искусственный интеллект: современный подход», *4-е изд, пер.с англ. К.А. Птицына, М.: Вильямс*, 2017, 1408 с.

103. С. Семенов, С. Гавриленко, С. Глоба, О. Бабенко, «Розробка системи виявлення комп'ютерних вірусів на основі нейронної мережі АРТ-1», *Системи обробки інформаці*ї, 2015, Випуск 10 (135) с. 126-129.

104. У. Микелуччи, «Прикладное глубокое обучение. Подход к пониманию глубоких нейронных сетей на основе метода кейсов», *Пер. с англ., СПб.: БХВ-Петербург*, 2020, 368 с.

105. Шолле Франсуа, «Глубокое обучение на Python», *СПб.: Питер*, 2018, 400 с.

106. Я. Гудфеллоу, И. Бенджио, А. Курвилль, «Г93 Глубокое обучение», пер. с анг. А.А. Слинкина, 2-е изд., испр, М.: ДМК Пресс, 2018, 652 с.

Додаток А. Документи, що підтверджують впровадження результатів дисертації

АКТ ВПРОВАДЖЕННЯ
результатів дисертаційної роботи
«Нейромережеві моделі та методи розпізнавання комп'ютерних вірусів»
Погорелова Володимира Володимировича
на здобуття наукового ступеня кандидата технічних наук
за спеціальністю 05.13.21 – «Системи захисту інформації»

Комісія у складі: голова – керівник науково-дослідної роботи «Квантово-криптографічні методи захисту критичної інформаційної структури держави» (д.р. № 0117U006770), доктор технічних наук, доцент, заступник декана Факультету кібербезпеки, комп'ютерної та програмної інженерії Гнатюк С.О., члени комісії – кандидат технічних наук, відповідальний виконавець НДР Охріменко Т.О. та кандидат технічних наук, виконавець НДР Сидоренко В.М. склали акт, про те, що здобувачем було розроблено метод проектування архітектури глибокої нейронної мережі, призначеної для розпізнавання вірусів, який на відміну від існуючих, за рахунок використання запропонованих елементів методологічної бази дозволяє визначити архітектурні параметри, що забезпечують пристосованість такої мережі до очікуваних умов застосування. Отримав подальший розвиток метод нейромережевого розпізнавання комп'ютерних вірусів, який на відміну від існуючих, за рахунок використання запропонованих елементів методологічної бази та запропонованого методу проектування архітектури глибокої нейронної мережі, забезпечує достатню похибку розпізнавання при різних умовах застосування з врахуванням обмежень щодо створення навчальної вибірки та обмежень щодо обчислювальних ресурсів системи антивірусного захисту.

Вказані результати було використано у ході виконання науково-дослідної роботи «Квантово-криптографічні методи захисту критичної інформаційної структури держави» (д.р. № 0117U006770).

Даний акт не є підставою для проведення взаємних фінансових розрахунків.

Керівник НДР С. Гнатюк

Відповідальний виконавець НДР Т. Охріменко

Виконавець НДР В. Сидоренко

САЙФЕР

Системи захисту інформації

ТОВ «САЙФЕР ПРО»
Адреса: 04107, Київ, вул. Нагірна, 25/27
Тел./Факс: (044) 484-46-17, 484-46-12, 483-03-22
E-mail: info@cipher.com.ua
https://cipher.com.ua

вих. № 09/08-20

від «17» серпня 2020 р.

АКТ

впровадження результатів дисертаційної роботи Погорелова Володимира Володимировича за темою «Нейромережеві моделі та методи розпізнавання комп'ютерних вірусів» на здобуття наукового ступеня кандидата технічних наук у діяльність компанії ТОВ «САЙФЕР ПРО»

Комісія у складі: голова – директора ТОВ «САЙФЕР ПРО» Охріменка А.О., члени комісії – керівник проектів, кандидат технічних наук Боровікова О.М., провідний розробник Бойко С.Т., склали цей акт, про те, що здобувачем було розроблено метод проектування архітектури глибокої нейронної мережі, призначеної для розпізнавання вірусів, який на відміну від існуючих, за рахунок використання запропонованих елементів методологічної бази дозволяє визначити архітектурні параметри, що забезпечують пристосованість такої мережі до очікуваних умов застосування.

Програмна реалізація алгоритму, що розроблено на основі розробленого методу проектування архітектури глибокої нейронної мережі для розпізнавання вірусів, дозволяє в 1,5 рази зменшити обчислювальні витрати, пов'язані з визначенням значень архітектурних параметрів глибокої нейронної мережі, призначеної для розпізнавання вірусів у модулі захисту системи дистанційного управління банківськими рахунками через мережу Інтернет «ELPay».

Також, отримав подальший розвиток метод нейромережевого розпізнавання комп'ютерних вірусів, який на відміну від існуючих, за рахунок використання запропонованих елементів методологічної бази та запропонованого методу проектування архітектури глибокої нейронної мережі, забезпечує достатню похибку розпізнавання при різних умовах застосування з врахуванням обмежень щодо створення навчальної вибірки та обмежень щодо обчислювальних ресурсів системи антивірусного захисту. В результаті експериментального дослідження, доведено, що його ефективність в 1,14 рази вища, ніж у подібних методів розпізнавання.

Директор ТОВ «САЙФЕР ПРО» А.О. Охріменко

ПОГОДЖЕНО:

Проректор з навчальної роботи
Національного авіаційного
університету
_____ А.Г. Гудманян
«25» ЛЮТОГО 2020 р.

ЗАТВЕРДЖУЮ:

Проректор з наукової роботи
Національного авіаційного
університету
_____ В. Харченко
«25» ЛЮТОГО 2020 р.

АКТ

впровадження у навчальний процес результатів дисертаційної роботи Погорелова Володимира Володимировича «Нейромережеві моделі та методи розпізнавання комп'ютерних вірусів» на здобуття ступеня кандидата технічних наук.

Комісія у складі: голова – завідувач кафедри безпеки інформаційних технологій (БІТ) Корченко О.Г., завідувач кафедри комп'ютеризованих систем захисту інформації Казмірчук С.В., професор кафедри безпеки інформаційних технологій Терейковський І.А. склали даний акт про те, що результати дисертаційної роботи Погорелова Володимира Володимировича впровадженні у навчальний процес та використовуються на кафедрі БІТ у 2019-2020 навчальному році при викладанні дисципліни «Захист корпоративного програмного забезпечення».

№ з/п	Назва роботи, що впроваджується	Форма впровадження	Ефективність від впровадження
1.	Метод проектування архітектури глибокої нейронної мережі	Лекція	Ознайомлення студентів з сучасними методами розпізнавання комп'ютерних вірусів, типами сучасних комп'ютерних вірусів та ознаками зараження вірусами.
2.	Модель визначення ефективних видів глибоких нейронних мереж	Лекція	Ознайомлення студентів з сучасними принципами та моделями визначення ефективних видів глибоких нейронних мереж, які дозволяють зменшити обсяг експериментальних досліджень, спрямованих на окреслення перспективності використання сучасних нейромережевих моделей в антивірусних засобах.

Голова комісії,

завідувач кафедри безпеки
інформаційних технологій О. Корченко

Члени комісії:

завідувач кафедри комп'ютеризованих
систем захисту інформації С. Казмірчук

професор кафедри безпеки
інформаційних технологій І. Терейковський

ЗАТВЕРДЖУЮ
Заст. директора з наукової роботи
Інституту проблем моделювання
в енергетиці ім. Г.Є. Пухова
НАН України
д-р техн. наук, ст. наук. співр.
_____ О.А. Чемерис
«____» _____ 2020р.

АКТ
впровадження результатів дисертаційної роботи
Погорелова Володимира Володимировича
«Нейромережеві моделі та методи розпізнавання комп'ютерних вірусів»
на здобуття наукового ступеня кандидата технічних наук
за спеціальністю 05.13.21 – Системи захисту інформації

Результати дисертаційного дослідження Погорелова Володимира Володимировича «Нейромережеві моделі та методи розпізнавання комп'ютерних вірусів» на здобуття наукового ступеня кандидата технічних наук за спеціальністю за спеціальністю 05.13.21 – Системи захисту інформації використано в роботі Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України.

Здобувачем було розроблено концептуальну модель визначення ефективних видів глибоких нейронних мереж, яка за рахунок взаємопов'язаних принципів допустимості використання, визначення множини ефективних видів та оцінювання ефективності виду глибокої нейронної мережі дозволяє зменшити обсяг експериментальних досліджень, спрямованих на окреслення перспективності використання сучасних нейромережевих моделей для побудови антивірусних засобів. Розроблено модель формування параметрів навчальних прикладів глибокої нейронної мережі, яка за рахунок формального представлення закодованих значень викликів API-функцій, байт-послідовності N-грамів, опткодів, основних регістрів процесора, а також результатів статичного аналізу зразків шкідливих та безпечних програм, двомірної інтерпретації бінарного коду програми і параметрів графу залежностей значень та станів дозволяє будувати засоби нейромережевого аналізу обфускованого програмного коду.

Вказані результати було використано у ході виконання науково-дослідної роботи «Дослідження ризиків інформаційної безпеки об'єктів критичної інфраструктури ГТС України та розробка методології поводження з ними» (д.р. № 0118U002371).

Даний акт не є підставою для проведення взаємних фінансових розрахунків.

Керівник НДР
пров. наук. співр. ІПМЕ ім. Г.Є. Пухова
НАН України, канд. техн. наук, ст. наук. співр. А.М. Давиденко

Відповідальний виконавець НДР
ст. наук. співр. ІПМЕ ім. Г.Є. Пухова
НАН України, канд. техн. наук, ст. наук. співр. С.Я. Гільгурт

```
from sklearn.model_selection import KFold
import tensorflow as tf
import numpy as np
from .. import settings
import json
import random
import datetime
import os
import math
import file_parsers
import time

class TensorFlowModule(object):
    def train(self):
        if not os.path.isfile(settings.samples_vectors_filepath):
            print('Missing samples vectors file!')
            file_parsers.parse_sample_files()
        # load the data from a file
        with open(settings.samples_vectors_filepath, 'r') as infile:
            print('Loading feature vectors... please wait!')
            sample_set = json.load(infile)
        self.run_training(sample_set)
    def run_training(self,sample_set):
        input_size = len(sample_set['features'])
        samples = [sample for sample in sample_set['samples']]
        x,W,b,pred,y,cost_func =
self.create_model_vars_ops_placeholders(settings.n_classes,input_size)
        init = tf.global_variables_initializer()
        print('Initialized session variables!')
        saver = tf.train.Saver()
        config = tf.ConfigProto()
        config.gpu_options.allow_growth = True
        with tf.Session(config=config) as sess:
            print('Running session...')
            random.shuffle(samples)
            sample_batches = list(self.split_batches(samples,settings.batch_size))
            test_batch = sample_batches[0]
            train_batches = sample_batches[1:]
            sess.run(init)
            status = self.run_sess_with_epochs(cost_func,train_batches,sample_set,sess,x,y)
            if status == 1:
                print('Loss function failure... re-running session')
                tf.reset_default_graph()
                self.run_training(sample_set)
                return
            test_xs = [sample_set['samples'][sample]['vector'] for sample in test_batch]
            test_ys = [sample_set['samples'][sample]['is_malicious'] for sample in test_batch]
            print('Number of benign samples in test batch: ' + str(test_ys.count([0,1])))
            print('Number of malware samples in test batch: ' + str(test_ys.count([1,0])))
            tp, tn, fp, fn =sess.run(self.get_metrics_operations(pred,y),feed_dict={x: test_xs, y:
test_ys})
            tpr, tnr, fpr, fnr, accuracy, recall, precision, f1_score = self.calculate_metrics(tp,tn,fn,fp)
            print('TPR: ' + str(tpr) + ' | ' + 'TNR: ' + str(tnr) + ' | ' + 'FPR: ' + str(fpr) + ' | ' + 'FNR: ' +
str(fnr))
```

```
                print('Precision: ' + str(precision)  + ' | ' + 'Recall: ' + str(recall) + ' | ' + 'F1 Score:' +
str(f1_score) + ' | ' + 'Accuracy: ', str(accuracy))
                print('Total number of samples: ' + str(len(samples)))
                saver.save(sess,os.path.join(settings.work_dir,'model'))
                with open('latest_train_logs.json', 'w') as outfile:
                   json.dump({'timestamp':                                                    '{:%Y-%b-%d
%H:%M:%S}'.format(datetime.datetime.now()),
                              'n_files':  str(len(samples)), 'batch_size': str(settings.batch_size),
                              'n_train_batches':str(len(train_batches)),'benign_samples_test_batch':
str(test_ys.count([0, 1])),
                             'malicious_samples_test_batch':str(test_ys.count([1,    0])),    'tpr':    str(tpr),'tnr':
str(tnr),
                             'fpr':       str(fpr),'fnr':       str(fnr),'accuracy':       str(accuracy),       'precision':
str(precision),'recall': str(recall),'f1_score': str(f1_score),}, outfile)
             tf.reset_default_graph()
          def calculate_metrics(self,tp,tn,fn,fp):
             tpr = float(tp) / (float(tp) + float(fn))
             tnr = float(tn) / (float(tn) + float(fp))
             fpr = float(fp) / (float(fp) + float(tn))
             fnr = float(fn) / (float(fn) + float(tp))
             accuracy = (float(tp) + float(tn)) / (float(tp) + float(fp) + float(fn) + float(tn))
             recall = tpr
             precision = float(tp) / (float(tp) + float(fp))
             f1_score = (2 * (precision * recall)) / (precision + recall)
             return [tpr,tnr,fpr,fnr,accuracy,recall,precision,f1_score]
          def get_metrics_operations(self,pred,output):
             predicted_output = tf.argmax(pred, 1)
             actual_output = tf.argmax(output, 1)
             actual_ones = tf.ones_like(actual_output)
             actual_zeros = tf.zeros_like(actual_output)
             predicted_ones = tf.ones_like(predicted_output)
             predicted_zeros = tf.zeros_like(predicted_output)
             tp_op = tf.reduce_sum(tf.cast(tf.logical_and(tf.equal(actual_output, actual_ones),
                                          tf.equal(predicted_output, predicted_ones)), "float"))
             tn_op = tf.reduce_sum(tf.cast(tf.logical_and(tf.equal(actual_output, actual_zeros),
                                          tf.equal(predicted_output, predicted_zeros)), "float"))
             fp_op = tf.reduce_sum(tf.cast(tf.logical_and(tf.equal(actual_output, actual_zeros),
                                          tf.equal(predicted_output, predicted_ones)), "float"))
             fn_op = tf.reduce_sum(tf.cast(tf.logical_and(tf.equal(actual_output, actual_ones),
                                          tf.equal(predicted_output, predicted_zeros)), "float"))
             return [tp_op, tn_op, fp_op, fn_op]
          def create_model_vars_ops_placeholders(self,n_classes, input_size):
             x = tf.placeholder(tf.float32, [None, input_size],name='x')
             W = tf.Variable(tf.zeros([input_size, n_classes]),name='W')
             b = tf.Variable(tf.zeros(n_classes),name='b')
             pred = tf.nn.softmax(tf.matmul(x, W) + b,name='pred')
             y = tf.placeholder(tf.float32, [None, n_classes],name='y')
             cost_func = -tf.reduce_sum(y * tf.log(pred),name='cost_func')
             return [x,W,b,pred,y,cost_func]
          def evaluate_model(self):
             tf.reset_default_graph()
             if not(os.path.isfile(settings.samples_vectors_filepath)):
                print('Missing samples vectors file!')
                file_parsers.parse_sample_files()
             print('Evaluating model')
             print('Loading feature vectors...')
             with open(settings.samples_vectors_filepath, 'r') as infile:
```

```
                sample_set = json.load(infile)
            samples = [sample for sample in sample_set['samples']]
            k_fold = KFold(settings.n_folds)
            ctr = 0
            eval_results={'timestamp': '{:%Y-%b-%d %H:%M:%S}'.format(datetime.datetime.now()),'eval':{}}
            print('Total folds: ' + str(settings.n_folds))
            for train_indices, test_indices in k_fold.split(samples):
                ctr = self.run_evaluation_step(ctr,train_indices,test_indices,sample_set,eval_results)
            total_accuracy = 0
            total_recall = 0
            total_precision = 0
            total_f1_score = 0
            for key, value in (eval_results['eval']).items():
                total_accuracy += eval_results['eval'][key]['accuracy']
                total_precision += eval_results['eval'][key]['precision']
                total_f1_score += eval_results['eval'][key]['f1_score']
                total_recall += eval_results['eval'][key]['recall']
            macro_avg_accuracy = total_accuracy/settings.n_folds
            macro_avg_recall = total_recall/settings.n_folds
            macro_avg_precision = total_precision/settings.n_folds
            macro_avg_f1_score = total_f1_score/settings.n_folds
            eval_results['avg'] = {'accuracy':macro_avg_accuracy,'precision':macro_avg_precision,'recall':macro_avg_recall,'f1_score':macro_avg_f1_score}
            print('Macro-average Precision: ' + str(macro_avg_precision)  + ' | ' + 'Macro-average Recall: ' + str(macro_avg_recall)
                  + ' | ' + 'Macro-average F1 Score:' + str(macro_avg_f1_score) + ' | ' + 'Macro-average Accuracy: ', str(macro_avg_accuracy))
            with open('latest_eval_logs.json', 'w') as outfile:
                json.dump(eval_results, outfile)
        def run_evaluation_step(self,ctr,train_indices,test_indices,sample_set,eval_results):
            start_time = time.clock()
            samples = [sample for sample in sample_set['samples']]
            input_size = len(sample_set['features'])
            ctr += 1
            print('Split ' + str(ctr) + ' running')
            train_samples = []
            test_samples = []
            for index in train_indices:
                train_samples.append(samples[index])
            for index in test_indices:
                test_samples.append(samples[index])
            x, W, b, pred, y, cost_func = self.create_model_vars_ops_placeholders(settings.n_classes, input_size)
            init = tf.global_variables_initializer()
            status = 0
            with tf.Session() as sess:
                sess.run(init)
                train_batches = list(self.split_batches(train_samples, settings.batch_size))
                random.shuffle(train_batches)
                status = self.run_sess_with_epochs(cost_func, train_batches, sample_set, sess, x, y)
                if status != 1:
                    test_xs = [sample_set['samples'][sample]['vector'] for sample in test_samples]
                    test_ys = [sample_set['samples'][sample]['is_malicious'] for sample in test_samples]
                    tp, tn, fp, fn = sess.run(self.get_metrics_operations(pred, y), feed_dict={x: test_xs, y: test_ys})
```

```
                    tpr, tnr, fpr, fnr, accuracy, recall, precision, f1_score = self.calculate_metrics(tp,tn,fn,fp)
                    print('TPR: ' + str(tpr) + ' | ' + 'TNR: ' + str(tnr) + ' | ' + 'FPR: ' + str(fpr) + ' | ' + 'FNR: ' +
str(fnr))
                    print('Precision: ' + str(precision)  + ' | ' + 'Recall: ' + str(recall) + ' | ' + 'F1 Score:' +
str(f1_score) + ' | ' + 'Accuracy: ', str(accuracy))
                    eval_results['eval'][str(ctr)]                                                                    =
{'tpr':tpr,'tnr':tnr,'fpr':fpr,'fnr':fnr,'accuracy':accuracy,'recall':recall,'precision':precision,'f1_score':f1_score
}
                tf.reset_default_graph()
                if status == 1:
                    print('Loss function failure... re-running step')
                    self.run_evaluation_step(ctr-1,train_indices,test_indices,sample_set,eval_results)
                end_time = time.clock() - start_time
                print('Evaluation Time: ' + str("{:.4f}".format(end_time)))
                return ctr
            def test_model(self,type='test'):
                load_path = None
                if type is 'eval':
                    load_path = settings.eval_savepath
                    if not(os.path.isfile(load_path)):
                        print('Missing samples vectors file!')
                        file_parsers.parse_sample_files(type='eval')
                if type is 'test':
                    load_path = settings.test_vectors
                    if not(os.path.isfile(load_path)):
                        print('Missing samples vectors file!')
                        file_parsers.parse_test_file()
                print('Loading feature vectors...')
                with open(load_path, 'r') as infile:
                    test_samples = json.load(infile)
                input_size = len(test_samples['features'])
                print('Number of test files: ' + str(len(test_samples['samples'])))
                x, W, b, pred, y, cost_func = self.create_model_vars_ops_placeholders(settings.n_classes,
input_size)
                test_sample_names = [sha1 for sha1 in test_samples['samples']]
                test_sample_input      =      [test_samples['samples'][sha1]['vector']      for      sha1      in
test_samples['samples']]
                test_sample_output = None
                if type is 'eval':
                    test_sample_output    =    [test_samples['samples'][sha1]['is_malicious']    for    sha1    in
test_samples['samples']]
                saver = tf.train.Saver()
                sample_verdicts = {}
                with tf.Session() as sess:
                    print('Restoring trained model')
                    saver.restore(sess, os.path.join(settings.work_dir,'model'))
                    print('Testing...')
                    if type is 'eval':
                        print('Number of benign samples in test batch: ' + str(test_sample_output.count([0,1])))
                        print('Number of malware samples in test batch: ' + str(test_sample_output.count([1,0])))
                        tp,      tn,      fp,      fn      =sess.run(self.get_metrics_operations(pred,y),feed_dict={x:
test_sample_input, y: test_sample_output})
                        tpr, tnr, fpr, fnr, accuracy, recall, precision, f1_score = self.calculate_metrics(tp,tn,fn,fp)
                        print('TPR: ' + str(tpr) + ' | ' + 'TNR: ' + str(tnr) + ' | ' + 'FPR: ' + str(fpr) + ' | ' + 'FNR: ' +
str(fnr))
                        print('Precision: ' + str(precision)  + ' | ' + 'Recall: ' + str(recall) + ' | ' + 'F1 Score:' +
str(f1_score) + ' | ' + 'Accuracy: ', str(accuracy))
```

```
                print('Total number of samples: ' + str(len(test_sample_names)))
                with open('latest_test_logs.json', 'w') as outfile:
                    json.dump({'timestamp':                                                    '{:%Y-%b-%d
%H:%M:%S}'.format(datetime.datetime.now()),
                               'n_files':  str(len(test_sample_names)),
                               'benign_samples': str(test_sample_output.count([0, 1])),
                              'malicious_samples':str(test_sample_output.count([1, 0])),
                              'tpr': str(tpr),'tnr': str(tnr),'fpr': str(fpr),'fnr': str(fnr),
                              'accuracy': str(accuracy), 'precision': str(precision),'recall': str(recall),'f1_score':
str(f1_score),}, outfile)
            else:
                for sha1 in test_sample_names:
                    test_sample_input = [test_samples['samples'][sha1]['vector']]
                    output = sess.run(pred,feed_dict={x:test_sample_input})
                    verd = sess.run(tf.equal(tf.constant(0, dtype=tf.int64),tf.argmax(output,axis=1)))
                    if verd[0] is np.True_:
                        sample_verdicts[sha1] ="Malicious"
                    else:
                        sample_verdicts[sha1] ="Clean"
                    print(sha1 + ' is ' + sample_verdicts[sha1])
            tf.reset_default_graph()
            if type is 'test':
                return sample_verdicts
        def read_trained_stats(self):
            with open('latest_train_logs.json','r') as infile:
                stats = json.load(infile)
            return stats
        def read_test_stats(self):
            with open('latest_test_logs.json','r') as infile:
                stats = json.load(infile)
            return stats
        def run_sess_with_epochs(self,cost_func, train_batches, sample_set,tf_sess,x,y):
            print('Number of epochs: ' + str(settings.n_epochs))
            number_of_times = len(train_batches)
            print('Number of batches per epoch: ' + str(number_of_times))
            for z in range(settings.n_epochs):
                start_time = time.clock()
                avg_cost = 0
                for i in range(number_of_times - 1):
                    decayed_learning_rate    =    tf.train.exponential_decay(settings.learning_rate,    i,    1,
settings.decay_rate, staircase=False)
                    optimizer                                                                  =
tf.train.GradientDescentOptimizer(decayed_learning_rate).minimize(cost_func)
                    batch_xs = [sample_set['samples'][sample]['vector'] for sample in train_batches[i]]
                    batch_ys = [sample_set['samples'][sample]['is_malicious'] for sample in train_batches[i]]
                    _, c = tf_sess.run([optimizer, cost_func], feed_dict={x: batch_xs, y: batch_ys})
                    if math.isnan(c):
                        return 1
                    else:
                        avg_cost += c / number_of_times
                print('Epoch ' + str(z) + ' cost: ' + str(avg_cost))
                end_time = time.clock() - start_time
                print('Time taken: ' + str("{:.4f}".format(end_time)))
        def split_batches(self,l, n):
            """Yield successive n-sized chunks from l."""
            for i in range(0, len(l), n):
                yield l[i:i + n]
```

```
                if test_file == filename:
                    sha1 = get_sha1(filepath)
                    if not (is_pe(filepath)):
                        print(test_file + ' is not a valid Windows Portable Executable(PE)!')
                        print('Removing file...')
                        os.remove(filepath)
                        continue
                    dllNames = get_import_functions(filepath)
                    if test_file is not sha1:
                        os.rename(filepath, os.path.join(settings.test_dir, sha1))
                        filepath = os.path.join(settings.test_dir, sha1)
                    with open(os.path.join(settings.test_dir, 'import_' + sha1), 'w') as write_file:
                        for dll in dllNames:
                            write_file.write(dll + '\n')
                        print('Function calls for ' + sha1 + ' have been extracted and saved as import_' +
sha1)
                    print('Moving ' + test_file + ' to test archive...')
                    move(filepath, os.path.join(os.path.join(settings.test_dir, 'archive'), test_file))
                    break
            except Exception as e:
                print("Error %s" % e)
                with open(settings.error_path, 'a') as errorfile:
                    filepath = os.path.join(settings.test_dir, test_file)
                    error = 'Error ' + filepath + str(e) + '\n'
                    errorfile.write(error)
                os.rename(filepath, os.path.join(settings.test_dir, 'error_' + test_file))

        def pre_process_train_files():
            for dir in settings.list_of_dir:
                print('Pre-processing files in ' + dir)
                for file in os.listdir(dir):
                    try:
                        filepath = os.path.join(dir, file)
                        sha1 = get_sha1(filepath)
                        if not (is_pe(filepath) and is_file_size_valid(filepath)):
                            os.remove(filepath)
                            continue
                        dllNames = get_import_functions(filepath)
                        if len(dllNames) > 10:
                            if dir is settings.samples_malicious_dir:
                                av_detection = get_AV_percentage_detection(sha1)
                                if av_detection == -1 or not (av_detection > 75) or is_packed(filepath):
                                    os.remove(filepath)
                                    continue
                            else:
                                av_detection = get_AV_percentage_detection(sha1)
                                if av_detection == -1 or not av_detection < 5:
                                    os.remove(filepath)
                                    continue
                            if file is not sha1:
                                os.rename(filepath, os.path.join(dir, sha1))
                            with open(os.path.join(dir,'import_' + sha1), 'w') as write_file:
                                for dll in dllNames:
                                    write_file.write(dll + '\n')
                            print('Function calls for ' + sha1 + ' have been extracted and saved as import_' +
```

```
sha1)
                    else:
                        os.remove(filepath)
                        continue
                except Exception as e:
                    print("Error %s" % e)
                    with open(settings.error_path, 'a') as errorfile:
                        error = 'Error ' + (os.path.join(dir, file)) + str(e) + '\n'
                        errorfile.write(error)
            os.rename(filepath, os.path.join(dir, 'error_' + file))
```