Чорноморський національний університет імені Петра Могили
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

Стрюк Олександр Сергійович

ДИСЕРТАЦІЯ

ОПТИМІЗАЦІЯ ГЕНЕРАТИВНИХ ЗМАГАЛЬНИХ НЕЙРОННИХ МЕРЕЖ В УМОВАХ АПАРАТНО-ПАРАМЕТРИЧНИХ ОБМЕЖЕНЬ

Спеціальність 123 — «Комп'ютерна інженерія»
Технічні науки
Подається на здобуття наукового ступеня доктора філософії

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


О. С. Стрюк

Науковий керівник Кондратенко Юрій Пантелійович, д.т.н., професор,
Заслужений винахідник України

Миколаїв — 2026

АНОТАЦІЯ

Стрюк О. С. Оптимізація генеративних змагальних нейронних мереж в умовах апаратно-параметричних обмежень. — Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 «Комп'ютерна інженерія». — Чорноморський національний університет імені Петра Могили, Миколаїв, 2026.

Сучасний розвиток систем штучного інтелекту (ШІ) демонструє експоненціальне зростання параметрів штучних нейронних мереж (НМ) і їх залежність від хмарних обчислень. Проте для задач реального часу чи з забезпеченням конфіденційності даних необхідність постійного зв'язку з хмарним сервером є суттєвим недоліком. Тому реалізація систем ШІ безпосередньо на мікрокомп'ютерах та кордонних пристроях є актуальною необхідністю. Разом з тим, розгортання ресурсномістких генеративних архітектур в умовах апаратно-параметричних обмежень (АПО) пов'язано з обчислювальною складністю та нестабільністю навчання НМ.

Дисертаційна робота присвячена вирішенню науково-практичної задачі розробки нових методів оптимізації генеративних змагальних нейронних мереж (ГЗМ), як систем ШІ, з урахуванням АПО та енергоефективності.

ГЗМ доповнюють існуючі датасети з недостатньою кількістю реальних даних додатковими штучними зразками для ефективного навчання НМ та підвищення якості зображень з низькою роздільною здатністю, зашумленнями та візуальними артефактами. Спектр застосування ГЗМ охоплює критично важливі сфери: енергетичну безпеку, національну оборону, охорону здоров'я, економіку та ін. В основі ГЗМ лежить антагоністична взаємодія двох НМ (генератора (G) і дискримінатора (D)), де G генерує нові зразки та мінімізує ймовірність їх D-виявлення, а D максимізує здатність розпізнавати згенеровані

дані і відрізняти їх від реальних, що дозволяє ГЗМ ефективно навчатися генерації даних, максимально наближених до реальних. Проте ефективне використання ГЗМ у областях від медичної діагностики та обробки супутникових даних до тестування стійкості біометричних систем проти кібератак обмежується нестабільністю навчання НМ та високими вимогами до системних характеристик обчислювальних комплексів.

Актуальною задачею залишається підвищення алгоритмічної ефективності ГЗМ для стійкої конвергенції змагальних архітектур та стабілізації процесів навчання. Особливої гостроти набуває необхідність адаптації ГЗМ до реалізації в умовах АПО, які зумовлені дефіцитом обчислювальної потужності, обмеженим обсягом оперативної та постійної пам’яті та низьким енергоспоживанням кордонних пристроїв. Це ускладнює застосування ГЗМ в системах реального часу без попередньої апаратно-алгоритмічної оптимізації. Дисертаційна робота спрямована на подолання зазначених недоліків ГЗМ, що підтверджує її актуальність та практичну цінність для розвитку сучасних систем ШІ.

Метою дослідження є розробка та удосконалення методів оптимізації ГЗМ для підвищення ефективності складних динамічних процесів їх навчання та генерації нових даних з розширенням спектру їх застосування в умовах АПО кордонних пристроїв, вбудованих систем контролю та мікропроцесорних архітектур.

Об’єктом дослідження є обчислювальні процеси навчання та функціювання ГЗМ у вбудованих комп’ютерних системах.

Предметом дослідження є методи, моделі та схемотехнічні рішення для оптимізації архітектур ГЗМ з реалізацією на кордонних пристроях в умовах їх апаратно-параметричних обмежень.

Новими науковими результатами дисертаційної роботи є:

Вперше розроблено математичну модель каскадної оптимізації ГЗМ, яка, на відміну від існуючих, базується на ієрархічній декомпозиції простору гіперпараметрів та врахуванні адаптивної динаміки навчання, що дозволяє забезпечити необхідну точність та швидкодію функціонування ГЗМ в умовах АПО кордонних пристроїв.

Вперше розроблено мультифазовий метод оптимізації навчання ГЗМ, який, на відміну від існуючих, базується на багаторівневому механізмі адаптивної конвергенції, що дозволяє запобігати колапсу моди та зникненню градієнтів функцій втрат без підвищення обчислювальної складності процедури навчання.

Удосконалено механізм адаптації ГЗМ, який ґрунтується на гібридизації каскадного та мультифазового методів в поєднанні з апаратом нечіткої логіки, що дозволяє комплексно підвищити ефективність навчання та якість генерації штучних даних в задачах виявлення аномалій, біометрії та комп'ютерного зору, зокрема знизити функцію втрат генератора у 3,5 рази, уникнути перенавчання дискримінатора (на рівні ~92%), прискорити збіжність ГЗМ в 7,6 разів, покращити метрику FID у 2,3 рази та досягти точності виявлення аномалій AUC = 0,92 із забезпеченням повноти (Recall) на рівні 1,0.

Набув подальшого розвитку програмно-апаратний метод для повного циклу функціонування ГЗМ на кордонних пристроях, який базується на інтеграції квантованого навчання та апаратно-орієнтованої каскадної оптимізації, що забезпечує реалізацію складних архітектур ГЗМ з врахуванням обмежень кордонних пристроїв та функціонування ГЗМ в режимі реального часу при зменшенні розміру імітаційної моделі ГЗМ в 3,9 рази та прискоренні процесу інференсу в 3,2 рази.

Для оцінки продуктивності ГЗМ у роботі застосовано методи кількісного аналізу із використанням комплексу сучасних метрик (FID, Loss Value,

Accuracy, Epochs to Convergence, Inference Time, Latency, Model Size, AUC, Recall), що дозволило об’єктивно порівняти ефективність розроблених методів із базовими підходами. Практична значущість дослідження підтверджена впровадженням результатів у наукові проєкти з Саарландським університетом ФРН та Інститутом проблем штучного інтелекту МОН і НАН України та в навчальний процес ЧНУ ім. П. Могили. Матеріали дисертаційної роботи доповідалися та обговорювалися на 8 міжнародних науково-технічних конференціях (IDAACS, DESSERT, ICTERI, ATIT, AICT, CMIS та ін.) та 3 Всеукраїнських наукових конференціях та семінарах.

У **вступній частині** дисертаційної роботи обґрунтовано актуальність теми, показано взаємозв’язок дослідження з іншими науковими напрямами та дисциплінами, визначено об’єкт і предмет дослідження, сформульовано мету та завдання роботи, окреслено використані методи дослідження, а також наведено відомості про наукову новизну та прикладне значення здобутих результатів. Наведено список публікацій за темою дослідження та інформацію про особистий внесок автора.

Перший розділ дисертації присвячений аналізу літературних джерел в області теоретичних основ проєктування та оптимізації ГЗМ, прикладного використання ГЗМ з реалізацією їх на мікропроцесорних і кордонних пристроях та мікрокомп’ютерах, перспектив їх застосування в різних областях людської діяльності. В результаті аналізу сучасного стану розвитку ГЗМ встановлено, що врахування особливостей проєктування ГЗМ на кордонних пристроях та їх АПО може здійснюватися шляхом (а) структурної реконфігурації архітектур ГЗМ, (б) розробки нових та модифікованих методів оптимізації та їх гібридної комбінації для процесів нейромережевого навчання та (в) шляхом апаратно-орієнтованої адаптації обчислювальних процесів і кордонних пристроїв.

У **другому розділі** проаналізовано математичні моделі, функції втрат та алгоритми навчання ГЗМ, а також методи досягнення конвергенції. Обґрунтовано стратегії стабілізації процесів навчання ГЗМ на основі модифікації функцій втрат та активації, пакетної нормалізації, транспонованої згортки, підрізки ваг та градієнтного штрафу з фокусом на оптимізацію гіперпараметрів та адаптацію архітектур для забезпечення необхідної продуктивності, швидкодії та точності в умовах АПО кордонних пристроїв. Окрему увагу приділено аналізу недоліків існуючих ГЗМ, в тому числі з реалізацією на кордонних пристроях, та методів їх проєктування. Детально висвітлені особливості процесів навчання ГЗМ з виникненням колапсу моди, конвергенції мереж генератора і дискримінатора, зникненням градієнтів функцій втрат та перенавчанням. Підкреслено потенціал структурно-параметричної оптимізації ГЗМ для адаптації їх математичних моделей, модифікації алгоритмів навчання та організації обчислювальних процесів на кордонних пристроях в умовах обмежень їх процесорної потужності, обсягів пам’яті і швидкодії.

Третій розділ дисертації присвячений стратегіям та методам оптимізації ГЗМ в умовах АПО з забезпеченням стабільності процесів навчання і підвищення якості генерації. Запропоновано каскадний метод оптимізації і механізм структурно-параметричного навчання ГЗМ, що забезпечує узгодження архітектури, гіперпараметрів і темпу навчання, мінімізуючи вплив експертних помилок на кінцеву ефективність моделі та її збіжність. Додатково розроблено метод мультифазової оптимізації, що розділяє процес навчання на декілька етапів, націлених на усунення окремих недоліків ГЗМ. Для підвищення ефективності та стабілізації навчання ГЗМ запропоновано механізм гібридизації каскадного та мультифазового методів.

Четвертий розділ дисертації присвячений імітаційному моделюванню та експериментальній валідації розроблених методів оптимізації для архітектур на основі стандартної ГЗМ та її модифікацій з різними функціями втрат (бінарної перехресної ентропії, втрат Вассерштейна зі штрафом за градієнт, перцептивних та змагальних втрат, втрат контенту та середньої абсолютної похибки, втрата МНК та ін.). Детально описано результати експериментів (реалізація ГЗМ на мові Python 3 та TensorFlow, Keras, PyTorch), проведених для перевірки ефективності розроблених методів (у різних контекстах застосування ГЗМ) на датасетах MNIST, SOCOFing та BIRDS 400, зокрема для задач біометрії, покращення роздільної здатності та виявлення аномалій з використанням комбінованих функцій втрат. В ході проведених експериментів вдалось забезпечити зниження функції втрат генератора (з 9,16 до 2,57), уникнути перенавчання дискримінатора та стабілізувати його точність (на рівні 92%), прискорити збіжність нейромережі (з 1300 до 170 епох), покращити метрику FID (з 45,9 до 19,4), а також досягти точності виявлення аномалій (AUC = 0,92) із забезпеченням повноти Recall (на рівні 1.0).

П’ятий розділ присвячено експериментальній верифікації адаптованих архітектур ГЗМ на кордонних пристроях (концепції On-Device Edge AI) для їх функціювання в умовах АПО. Серед спектру сучасних платформ для кордонних обчислень (таких як NVIDIA Jetson, Google Coral Dev Board, Intel Neural Compute Stick або FPGA-рішення) обґрунтовано в якості експериментальної платформи обрання одноплатного мікрокомп’ютера Raspberry Pi 5 (процесор Broadcom BCM2712, архітектура ARM Cortex-A76, 8 ГБ RAM). Raspberry Pi 5 не має спеціалізованих тензорних ядер, але його перевагами є низьке енергоспоживання, масо-габаритна компактність, універсальність до застосування в Інтернеті речей та низька вартість. Детально

описано ключовий етап реалізації динамічного квантування ваг моделі у цілочисельний формат INT8 (засобами бібліотеки torch.quantization) з формату з плаваючою комою FP32. Це забезпечило критичне зниження ресурсомісткості ГЗМ: розмір моделі зменшено з 2,14 МБ до 0,55 МБ, що дозволяє розміщувати модель у швидкій кеш-пам’яті процесора. Профілювання інференсу (PyTorch Profiler) підтвердило прискорення часу відгуку до 0,61 мс порівняно з базовою FP32-версією (1,95 мс), при збереженні прийнятної якості генерації зразків. Використання каскадного та мультифазового методів оптимізації також дозволило досягти структурної узгодженості та покращення метричних показників коректності відтворення при генерації рукописного тексту MNIST, як тестового типу даних. Отримані і детально представлені в розділі результати доводять, що оптимізовані ГЗМ здатні працювати автономно в режимі реального часу на кордонних пристроях, усуваючи залежність від хмарних обчислень.

У **висновках** узагальнено головні результати дисертаційної роботи та окреслено напрями подальших досліджень щодо апаратного прискорення, оптимізації та реалізації ГЗМ на різнотипних мікрокомп’ютерних платформах.

Ключові слова: штучний інтелект, ГЗМ, нейронні мережі, вбудовані системи, кордонні пристрої, мікрокомп’ютери, Інтернет речей, алгоритми, оптимізація, машинне навчання, глибоке навчання, виявлення аномалій, нечітка логіка, розпізнавання образів, енергоефективність

ABSTRACT

Striuk O. S. Optimization of Generative Adversarial Networks under Hardware-Parametric Constraints. — Qualification scientific work manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 123 “Computer Engineering”. — Petro Mohyla Black Sea State University, Mykolaiv, 2026.

The modern development of Artificial Intelligence (AI) systems demonstrates an exponential growth in the parameters of artificial Neural Networks (NNs) and their increasing reliance on cloud computing. However, for real-time tasks or those requiring high data privacy, the necessity of a constant connection to a cloud server remains a significant drawback. Therefore, the implementation of AI systems directly on microcomputers and edge devices has become an urgent necessity. At the same time, the deployment of resource-intensive generative architectures under hardware-parametric constraints (HPCs) is hindered by computational complexity and the instability of NN training.

This dissertation is dedicated to solving the scientific and practical problem of developing new methods for optimizing Generative Adversarial Networks (GANs) as AI systems, taking into account HPCs and energy efficiency.

GANs augment existing datasets that lack sufficient real-world data with additional synthetic samples for effective NN training and to improve the quality of images with low resolution, noise, and visual artifacts. The scope of GAN applications covers critical sectors: energy security, national defense, healthcare, economics, and others. At the core of a GAN lies the antagonistic interaction between two NNs: the Generator (G) and the Discriminator (D). Here, G generates new samples and minimizes the probability of their detection by D, while maximizes its ability to recognize generated data and distinguish it from real data. This allows the GAN to effectively learn to generate data that closely resembles real-world

samples. However, the efficient use of GANs in fields ranging from medical diagnostics and satellite data processing to testing the resilience of biometric systems against cyberattacks is limited by the instability of NN training and the high system requirements of computing complexes.

Improving the algorithmic efficiency of GANs for stable convergence of adversarial architectures and stabilization of training processes remains a pressing task. Of particular urgency is the need to adapt GANs for implementation under HPCs, which are characterized by limited computing power, restricted RAM and storage capacity, and low power consumption of edge devices. These factors complicate the deployment of GANs in real-time systems without prior hardware-algorithmic optimization. This dissertation is aimed at overcoming these GAN limitations, confirming its relevance and practical value for the advancement of modern AI systems.

The goal of the research is to develop and improve GAN optimization methods to enhance the efficiency of complex dynamic processes in their training and data generation, expanding their range of application within the HPCs of edge devices, embedded control systems, and microprocessor architectures.

The object of the research is the computational processes of GAN training and functioning within embedded computer systems.

The subject of the research is the methods, models, and circuit design solutions for optimizing GAN architectures for implementation on edge devices under hardware-parametric constraints.

Scientific novelty of the research:

For the first time, a mathematical model for cascade optimization of GANs has been developed. Unlike existing models, it is based on the hierarchical decomposition of the hyperparameter space and accounts for adaptive training

dynamics, ensuring the necessary accuracy and performance of GAN operations under the HPCs of edge devices.

For the first time, a multiphase method for GAN training optimization has been developed. Unlike existing methods, it is based on a multi-level adaptive convergence mechanism, which prevents mode collapse and the vanishing gradient problem of loss functions without increasing the computational complexity of the training procedure.

Has been improved — the GAN adaptation mechanism, based on the hybridization of cascade and multiphase methods combined with fuzzy logic. This allows for a comprehensive increase in training efficiency and synthetic data generation quality for anomaly detection, biometry, and computer vision tasks. Specifically, it reduces the generator loss function by 3.5 times, prevents discriminator overfitting (maintaining it at ~92%), accelerates GAN convergence by 7.6 times, improves the FID metric by 2.3 times, and achieves an anomaly detection accuracy of AUC = 0.92 with a Recall of 1.0.

Has been further developed — the software-hardware method for the full operational cycle of GANs on edge devices. It is based on the integration of quantized training and hardware-oriented cascade optimization, ensuring the implementation of complex GAN architectures while considering edge device constraints. This enables real-time GAN functionality, reducing the simulation model size by 3.9 times and accelerating the inference process by 3.2 times.

To evaluate GAN performance, quantitative analysis methods were applied using a suite of modern metrics (FID, Loss Value, Accuracy, Epochs to Convergence, Inference Time, Latency, Model Size, AUC, Recall), allowing for an objective comparison of the developed methods against baseline approaches. The practical significance of the research is confirmed by the implementation of results into scientific projects with Saarland University (Germany) and the Institute of Artificial

Intelligence Problems of the MES and NAS of Ukraine, as well as into the educational process at Petro Mohyla Black Sea State University. The dissertation materials were presented and discussed at 8 international scientific and technical conferences (including IDAACS, DESSERT, ICTERI, ATIT, AICT, CMIS, etc.) and 3 All-Ukrainian scientific conferences and seminars.

The **introductory part** of the dissertation substantiates the relevance of the topic, demonstrates the relationship between the study and other scientific fields and disciplines, defines the object and subject of research, formulates the goals and objectives, outlines the research methods used, and provides information on the scientific novelty and practical significance of the results obtained. A list of publications on the research topic and information regarding the author's personal contribution are also provided.

The **first chapter** of the dissertation is devoted to the analysis of literature sources in the field of theoretical foundations of GAN design and optimization, the applied use of GANs with their implementation on microprocessors, edge devices, and microcomputers, and the prospects of their application in various areas of human activity. As a result of the analysis of the current state of GAN development, it has been established that the specific features of designing GANs for edge devices under HPCs can be addressed through (a) structural reconfiguration of GAN architectures; (b) development of new and modified optimization methods and their hybrid combinations for NN training processes; (c) hardware-oriented adaptation of computational processes and edge devices.

In the **second chapter**, mathematical models, loss functions, and GAN training algorithms, as well as methods for achieving convergence, are analyzed. Strategies for stabilizing GAN training processes are substantiated based on the modification of loss and activation functions, batch normalization, transposed convolution, weight clipping, and gradient penalty, with a focus on hyperparameter

optimization and architectural adaptation to ensure the required performance, speed, and accuracy under the HPCs of edge devices. Special attention is paid to the analysis of the shortcomings of existing GANs, including those implemented on edge devices, and their design methods. The features of GAN training processes are covered in detail, including the occurrence of mode collapse, the convergence of generator and discriminator networks, vanishing gradients of loss functions, and overfitting. The potential of structural-parametric GAN optimization for adapting their mathematical models, modifying training algorithms, and organizing computational processes on edge devices under constraints of processing power, memory capacity, and throughput is highlighted.

The **third chapter** of the dissertation is dedicated to GAN optimization strategies and methods under HPCs, ensuring the stability of training processes and improving generation quality. A cascade optimization method and a mechanism for structural-parametric GAN training are proposed, ensuring the alignment of architecture, hyperparameters, and learning rate, thereby minimizing the impact of expert errors on the final efficiency of the model and its convergence. Additionally, a multiphase optimization method has been developed, which divides the training process into several stages aimed at eliminating specific GAN shortcomings. To increase efficiency and stabilize GAN training, a mechanism for the hybridization of cascade and multiphase methods is proposed.

The **fourth chapter** of the dissertation is dedicated to the simulation modeling and experimental validation of the developed optimization methods for architectures based on standard GANs and their modifications with various loss functions (Binary Cross-Entropy, Wasserstein loss with gradient penalty, perceptual and adversarial losses, content loss, Mean Absolute Error, Least Squares loss, etc.). The results of experiments (implemented in Python 3 using TensorFlow, Keras, and PyTorch) conducted to verify the effectiveness of the developed methods across various GAN

application contexts are described in detail. These tests utilized the MNIST, SOCOFing, and BIRDS 400 datasets, specifically targeting tasks in biometrics, super-resolution, and anomaly detection using combined loss functions. During the experiments, it was possible to reduce the generator loss function (from 9.16 to 2.57), prevent discriminator overfitting while stabilizing its accuracy (at 92%), accelerate neural network convergence (from 1300 to 170 epochs), improve the FID metric (from 45.9 to 19.4), and achieve an anomaly detection accuracy of AUC = 0.92 with a Recall of 1.0.

The **fifth chapter** focuses on the experimental verification of adapted GAN architectures on edge devices (On-Device Edge AI concepts) for operation under HPCs. Among the spectrum of modern edge computing platforms (such as NVIDIA Jetson, Google Coral Dev Board, Intel Neural Compute Stick, or FPGA solutions), the selection of the Raspberry Pi 5 single-board computer (Broadcom BCM2712 processor, ARM Cortex-A76 architecture, 8 GB RAM) as the experimental platform is substantiated. Although the Raspberry Pi 5 lacks specialized tensor cores, its advantages include low power consumption, compact form factor, versatility for Internet of Things (IoT) applications, and low cost. The key implementation stage of dynamic weight quantization from FP32 floating-point to INT8 integer format (using the torch.quantization library) is described in detail. This ensured a critical reduction in GAN resource intensity: the model size was decreased from 2.14 MB to 0.55 MB, allowing the model to be hosted within the processor's high-speed cache memory. Inference profiling (PyTorch Profiler) confirmed an improvement in response time to 0.61 ms compared to the baseline FP32 version (1.95 ms), while maintaining acceptable sample generation quality. The use of cascade and multiphase optimization methods also achieved structural consistency and improved metric indicators of reproduction fidelity during the generation of MNIST handwritten digits as test data. The results obtained and presented in detail in this

chapter prove that optimized GANs are capable of autonomous, real-time operation on edge devices, eliminating reliance on cloud computing.

In the **conclusions**, the main results of the dissertation are summarized, and directions for further research regarding hardware acceleration, optimization, and the implementation of GANs on various types of microcomputing platforms are outlined.

Keywords: Artificial Intelligence, GAN, Neural Networks, Embedded Systems, Edge Devices, Microcomputers, Internet of Things, Algorithms, Optimization, Machine Learning, Deep Learning, Anomaly Detection, Fuzzy Logic, Pattern Recognition, Energy Efficiency

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Праці, в яких опубліковані основні наукові результати дисертації:

1. *Стрюк О. С.,* Кондратенко Ю. П. Методи прикладного застосування генеративних змагальних мереж при обробці графічних даних. *Штучний інтелект*, 2023. № 3. С. 154–161. DOI: https://doi.org/10.15407/jai2023.03.154. **(ISSN 2710–1673) (Категорія Б)**

   (Особистий внесок: результати порівняльного аналізу методів прикладного застосування генеративних змагальних мереж при обробці графічних даних на основі проведених особисто експериментальних досліджень.)

   URL: https://jai.in.ua/index.php/архів?paper_num=1606.

2. *Striuk O.*, Kondratenko Y. Generative Adversarial Neural Networks and Deep Learning: Successful Cases and Advanced Approaches. *International Journal of Computing*, 2021. Vol. 20, No. 3. P. 339–349. DOI: https://doi.org/10.47839/ijc.20.3.2278. **(ISSN 1727–6209) (Scopus)**

   URL: https://computingonline.net/computing/article/view/2278.

3. *Striuk O. S.*, Kondratenko Y. P. Generative Adversarial Networks in Cybersecurity: Analysis and Response. In: C. Berger-Vachon, et al. (Eds) *Artificial Intelligence in Control and Decision-making Systems. Studies in Computational Intelligence*. Cham: Springer, 2023. Vol. 1087. P. 373–388. DOI: https://doi.org/10.1007/978-3-031-25759-9_18. **(Scopus)**

   URL: https://link.springer.com/chapter/10.1007/978-3-031-25759-9_18.

4. *Striuk O.*, Kondratenko Y. Implementation of Generative Adversarial Networks in Mobile Applications for Image Data Enhancement. *Journal of Mobile Multimedia*, 2023. Vol. 19, No. 03. P. 823–838. DOI: https://doi.org/10.13052/jmm1550-4646.1938. **(ISSN 1550–4646) (Scopus)**

   URL:https:journals.riverpublishers.com/index.php/JMM/article/view/15053.

5. *Striuk O.*, Kondratenko Y. Optimization Strategy for Generative Adversarial Networks Design. *International Journal of Computing*, 2023. Vol. 22, No. 3. P. 292–301. DOI: doi.org/10.47839/ijc.22.3.3223. **(ISSN 1727–6209) (Scopus)**

URL: https://computingonline.net/computing/article/view/3223.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

6. Generative Adversarial Neural Network for Creating Photorealistic Images / *Striuk O.*, Kondratenko Y., Sidenko I., Vorobyova A. *Information Control Systems & Technologies (ATIT 2020)*: Proceedings of the 2nd IEEE International Conference on Advanced Trends in Information Theory, Kyiv, 25–27 November, 2020. Kyiv, 2020. P. 368–371. DOI: 10.1109/ATIT50783.2020.9349326. **(Scopus)**

URL: https://ieeexplore.ieee.org/document/9349326/.

7. Adaptive Deep Convolutional GAN for Fingerprint Sample Synthesis / *Striuk O.*, Kondratenko Y. *Information Control Systems & Technologies (AICT 2021)*: Proceedings of the 4th IEEE International Conference on Advanced Information and Communication Technologies, Lviv, 21–24 September, 2021. Lviv, 2021. P. 193–196. DOI: 10.1109/AICT52120.2021.9628978. **(Scopus)**

URL: https://ieeexplore.ieee.org/document/9628978.

8. Cross-Domain Reconfigurable GAN with Fuzzy Components for Anomaly Detection / *Striuk O.*, Kondratenko Y. *Dependable Systems, Services and Technologies (DESSERT 2023)*: Proceedings of the 13th International Conference, Athens, 13–15 October, 2023. Athens, 2023. P. 1–5. DOI: 10.1109/DESSERT61349.2023.10416521. **(Scopus)**

URL: https://ieeexplore.ieee.org/document/10416521.

9. Gradient-Penalty GAN Framework for High-Fidelity Fingerprint Synthesis / *Striuk O.*, Kondratenko Y. *Computer Modeling and Intelligent Systems (CMIS 2025)*: Proceedings of the 8th International Workshop, Zaporizhzhia, 5 May,

2025. Zaporizhzhia, CEUR Workshop Proceedings, Vol. 3988, 2025. P. 175–188. **(ISSN 1613–0073) (Scopus)**

URL: https://ceur-ws.org/Vol-3988/paper15.pdf.

10. Математична модель генеративно-змагальної мережі / Воробйова А. І., *Стрюк О. С. Могилянські читання – 2020: Досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти*: матеріали XXIII Всеукраїнської науково-практичної конференції, м. Миколаїв, 16–20 листопада 2020 р., Миколаїв, 2020. С. 134–137.

URL: https://dspace.chmnu.edu.ua/jspui/handle/123456789/405.

11. Прикладна цінність ГЗМ як систем штучного інтелекту / *Стрюк О. С. Інтелектуальні інформаційні системи*: матеріали Всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів, м. Миколаїв, 28–31 січня 2020 р., Миколаїв, 2020. С. 35–38.

URL: http://bit.ly/4rjb0YQ.

12. Розгортання та інженерна оптимізація генеративних змагальних мереж на кордонних системах / *Стрюк О. С. Інтелектуальні інформаційні системи*: матеріали Всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів, м. Миколаїв, 4–5 грудня 2025 р., Миколаїв, 2025. С. 104–106.

URL: https://dspace.chmnu.edu.ua/jspui/handle/123456789/3054.

Наукові праці, які додатково відображають наукові результати дисертації:

13. Шевченко А. І., Барановський С. В., Білокобильський О. В., Кондратенко Ю. П., Козлов О. В., Сіденко Є. В., *Стрюк О. С.* та ін. *Стратегія розвитку штучного інтелекту в Україні: монографія* / за заг. ред. А. І. Шевченка. Київ: ІПШІ, 2023. С. 1-307. DOI: 10.15407/development_strategy_2023.

(Особистий внесок: підготовлені доповнення до розділів 1 «Парадигма», 3 «Мета і завдання», 8 «Наукове, кадрове та матеріальне забезпечення національної екосистеми ШІ», 10 «Прикінцеві положення»; підрозділів: 2.2 «Основні напрями досліджень ШІ» розділу 2, 7.1 «ШІ у сфері безпеки та оборони України», 7.4 «ШІ в промисловості та енергетиці», 7.7 «ШІ у сільському господарстві» розділу 7.)

URL: https://jai.in.ua/index.php/en/issues?paper_num=1545.

14. Tendencies and Challenges of Artificial Intelligence Development and Implementation / Kondratenko Y., Shevchenko A., Zhukov Y., Kondratenko G., *Striuk O. Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2023)*: proceedings of the 12th IEEE International Conference, Dortmund, 7–9 September, 2023. Dortmund, 2023. P. 221–226. DOI: 10.1109/IDAACS58523.2023.10348800. **(Scopus)**

    URL: https://ieeexplore.ieee.org/document/10348800.

15. Analysis of the Priorities and Perspectives in Artificial Intelligence Implementation / Kondratenko Y., Shevchenko A., Zhukov Y., Kondratenko G., *Striuk O. Systems, Services and Technologies (DESSERT 2023)*: Proceedings of the 13th International Conference, Athens, 13–15 October, 2023. Athens, 2023. P. 1–8. DOI: 10.1109/DESSERT61349.2023.10416432. **(Scopus)**

    URL: https://ieeexplore.ieee.org/document/10416432.

16. Kondratenko Y. P., Zhukov Y. D., Shevchenko A. I., Zhukova O. Y., *Striuk O. S.* AI and Digital Evolution in the Education System of Ukraine. In: V.I. Slyusar, et al. (Eds) *AI in Education Systems: Successful Cases and Perspectives*. River Publishers, Gistrup, Denmark, 2025. P. 47–74. **(Scopus)**

    URL: https://www.riverpublishers.com/book_details.php?book_id=1488.

17. Shevchenko A. I., Lande D. V., Bilokobylsky O. V., Kondratenko Y. P., Kozlov O. V., Sidenko I. V., *Striuk O. S.* et al. Regarding the Draft Strategy Development

of Artificial Intelligence in Ukraine. *Artificial Intelligence (Штучний інтелект)*, 2022. № 1. P. 8–157. DOI: https://doi.org/10.15407/jai2022.01.008. **(ISSN 2710–1673) (Категорія Б)**

(Особистий внесок: підготовлено матеріали щодо ключових напрямів досліджень ШІ у сфері безпеки та оборони, промисловості, енергетиці та сільському господарстві України.)

URL: https://jai.in.ua/index.php/en/issues?paper_num=1492.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ШІ	штучний інтелект
ГШІ	(англ. GenAI) генеративний штучний інтелект
ГМ	генеративні моделі
НД	набори даних (датасети)
МН	машинне навчання
ГН	глибоке навчання
НМ	нейронні мережі
ГЗМ	(англ. GAN), генеративні змагальні мережі (англ. Generative Adversarial Networks)
АПО	апаратно-параметричні обмеження
ВММ	великі мовні моделі
ДМ	дифузійні моделі
SRGAN	англ. Super Resolution GAN
ВГЗМ	(англ. WGAN), ГЗМ на базі втрати Васерштейна
СГС	стохастичний градієнтний спуск
FID	відстань Фреше (англ. Fréchet Inception Distance)
ІДГ	індекс достовірності генерації (англ. Inception Score)
MNIST	Модифікована база даних Національного інституту стандартів і технологій США (англ. Modified National Institute of Standards and Technology database)
МНК	метод найменших квадратів

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний розвиток систем штучного інтелекту (в т.ч. генеративних змагальних мереж, систем розпізнавання, класифікації, сегментації, обробки природної мови та ін.) демонструє експоненціальне зростання параметрів штучних нейронних мереж і залежність їх від хмарних обчислень при застосуванні у критично важливих сферах. Проте для розв’язання задач в реальному часі (автономна навігація тощо) або задач з забезпеченням конфіденційності даних (медицина, оборона) необхідність постійного зв’язку з хмарним сервером є суттєвим недоліком. Тому реалізація інтелектуальних систем безпосередньо на мікрокомп’ютерах та мікропроцесорних і кордонних пристроях (Edge AI) є актуальною необхідністю. Разом з тим, таке розгортання ресурсномістких генеративних архітектур в умовах апаратно-параметричних обмежень вбудованих та мобільних пристроїв наштовхується на проблеми обчислювальної складності та нестабільності нейромережевого навчання. Подолання цих проблем вимагає розробки та удосконалення ресурсоефективних методів оптимізації, що забезпечують адаптацію до динаміки навчання, архітектур нейронних мереж та параметрів CPU-орієнтованих платформ. Дисертаційна робота присвячена вирішенню практичної задачі розробки та впровадження нових моделей і методів проєктування та оптимізації генеративних змагальних нейронних мереж (ГЗМ), як систем штучного інтелекту, з урахуванням апаратно-програмних обмежень, енергоефективного використання обчислювальних ресурсів, підтримки продуктивності та надійності кордонних систем. ГЗМ забезпечують доповнення існуючих наборів даних (датасетів) додатковими штучними зразками для ефективного навчання нейронних мереж в умовах недостатньої кількості реальних даних та підвищення якості існуючих зображень з низькою роздільною здатністю, зашумленнями та візуальними

артефактами. Спектр застосування ГЗМ охоплює критично важливі сфери: енергетичну безпеку, національну оборону, охорону здоров'я, економіку та ін. В основі ГЗМ лежить антагоністична взаємодія двох нейронних мереж (генератора і дискримінатора), де генератор створює нові зразки та мінімізує ймовірність виявлення цих штучних даних дискримінатором, а дискримінатор максимізує здатність розпізнавати згенеровані штучні дані і відрізняти їх від реальних зразків, що дозволяє ГЗМ автоматично навчатися генерації даних, максимально наближених до реальних. При цьому, ГЗМ забезпечують високоточне формування статистичного розподілу даних, що є критично важливим для генерації реалістичних зразків у доменах з обмеженими вибірками (англ. low-data regimes), зокрема, при розв'язанні прикладних задач у областях від медичної діагностики та обробки супутникових даних до тестування стійкості біометричних систем проти кібератак, де отримання реальних даних є ускладненим або неможливим. Проте ефективне використання ГЗМ у цих галузях обмежується нестабільністю навчання та високими вимогами до системних характеристик обчислювальних комплексів.

Актуальною задачею також залишається вирішення проблем підвищення алгоритмічної ефективності ГЗМ для забезпечення стійкої конвергенції змагальних архітектур та стабілізації динамічних процесів навчання. Особливої гостроти набуває необхідність адаптації ГЗМ до реалізації в умовах апаратно-параметричних обмежень, які зумовлені дефіцитом обчислювальної потужності, обмеженим обсягом оперативної та постійної пам'яті, а також специфікою енергоспоживання кордонних пристроїв, мікрокомп'ютерних платформ та вбудованих інтелектуальних систем. Це ускладнює пряме застосування ГЗМ в системах управління та підтримки прийняття рішень у реальному часі без попередньої апаратно-параметричної оптимізації та удосконалення спеціалізованих методів

обчислень. Дисертаційна робота спрямована на подолання зазначених недоліків ГЗМ, що підтверджує її актуальність та практичну цінність для розвитку сучасних інтелектуальних систем.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційне дослідження виконано в рамках наукових програм, спрямованих на розвиток та підвищення ефективності систем штучного інтелекту (ШІ) та їх застосування в різних галузях.

Зокрема, робота відповідає програмам, що стосуються розробки інноваційних методів машинного навчання (МН) і глибокого навчання (ГН) для обробки великих даних та забезпечення інформаційної безпеки. Основні результати роботи апробовані на вітчизняних та міжнародних конференціях, що підтверджує її відповідність актуальним науковим напрямам.

Дисертаційні дослідження проводились: (а) відповідно до пріоритетних напрямків науково-дослідних робіт Чорноморського національного університету імені Петра Могили (наукові ініціативні кафедральні теми: «Системи і методи прийняття рішень в умовах нечіткості та невизначеності даних», 01.01.2022- 01.01.2025; «Моделі та методи розв'язання задач штучного інтелекту, машинного навчання, систем керування та прийняття рішень», 01.01.2025-01.01.2028); в рамках (б) державного науково-дослідницького проєкту Інституту проблем штучного інтелекту (ІПШІ) Національної академії наук України і Міністерства освіти і науки України (МОН) України «Створення стратегії розвитку штучного інтелекту в Україні» (2020-2022 рр.) на замовлення МОН України (№ ДР 0120U102299) та (в) міжнародного проєкту DAAD-Ostpartnerschaftsprogramm з Саарландським університетом, Саарбрюккен, Німеччина (2018-2020 рр.), спрямованого на розробку інтелектуальних мобільних роботів.

Об’єктом дослідження є обчислювальні процеси навчання та функціювання ГЗМ у вбудованих комп’ютерних системах.

Предметом дослідження є методи, моделі та схемотехнічні рішення для оптимізації архітектур ГЗМ з реалізацією на кордонних пристроях в умовах їх апаратно-параметричних обмежень.

Метою дослідження є розробка та удосконалення методів оптимізації ГЗМ для підвищення ефективності складних динамічних процесів їх навчання та генерації нових даних з розширенням спектру їх застосування в умовах апаратно-параметричних обмежень кордонних пристроїв, вбудованих систем контролю та мікропроцесорних архітектур.

Для досягнення цієї мети були поставлені наступні **завдання дослідження**:

1. Провести комплексний міждисциплінарний аналіз ГЗМ (як систем штучного інтелекту) з метою розширення області їх застосування при реалізації на кордонних пристроях і мікрокомп’ютерних платформах та визначення напрямків їх можливої оптимізації для розв’язання прикладних задач генерації штучних та аналізу і обробки реальних даних.
2. Висвітлити критичні недоліки алгоритмічно-організаційних процедур оптимізації ГЗМ та існуючих архітектурних варіацій ГЗМ (нестабільність градієнтів, колапс моди, висока чутливість до гіперпараметрів тощо), а також визначити шляхи подолання зазначених недоліків і подальшого вдосконалення реалізованих на кордонних пристроях ГЗМ.
3. Обґрунтувати необхідність уніфікованого підходу до проєктування ГЗМ для різнотипних прикладних застосувань та формалізувати задачу оптимізації і стабілізації ГЗМ в умовах апаратно-параметричних

обмежень для підвищення ефективності процесів їх навчання та генерації даних при розгортанні ГЗМ на кордонних пристроях.

4. Розробити каскадний і мультифазовий методи оптимізації та їх гібридне комбінування для підвищення ефективності навчання ГЗМ на кордонних пристроях.
5. Розробити реконфігуровану адаптивну архітектуру ГЗМ зі спеціалізованими реалізаціями для: а) синтезу біометричних даних, б) реалізації методів супер-роздільної здатності в мобільних платформах та системах комп'ютерного зору, в) виявлення аномалій з подоланням невизначеності інформації на основі застосування нечіткої логіки.
6. Розробити експериментальні зразки ГЗМ, провести експериментальні дослідження і створити фреймворк для навчання та інференсу ГЗМ на мікрокомп'ютерах (як на кордонних пристроях), зокрема на платформі Raspberry Pi 5.

Використано комплекс **методів дослідження**, зокрема: методи МН та ГН для розробки та тренування ГЗМ; методи обробки синтезованих даних для аналізу і покращення якості зображень до надвисокої роздільної здатності; методи теорії нечітких множин та нечіткої логіки для оцінки та інтерпретації результатів виявлення аномалій; різнотипні алгоритмічні та гіперпараметричні модифікації для розробки каскадного та мультифазового методів оптимізації навчання ГЗМ, а також їх гібридного комбінування; методи квантованого навчання, профілювання та тестування моделей на базі датасету MNIST для інтеграції та узгодження алгоритмів ГЗМ з апаратними обмеженнями кордонних пристроїв та мікрокомп'ютерних платформ; а також методи структурної організації та синтезу IoT-систем та вбудованих систем.

Наукова новизна отриманих результатів визначається особистим внеском автора у вирішення актуального наукового завдання галузі знань

«Інформаційні технології», яке полягає в структурній реконфігурації архітектур ГЗМ, розробці методів оптимізації та механізмів їх гібридного комбінування для процесів навчання ГЗМ та апаратно-орієнтованій адаптації кордонних пристроїв та мікрокомп'ютерних компонентів.

Найбільш значущими результатами дослідження, що становлять наукову новизну, розкривають суть роботи та виносяться на захист, є такі:

Вперше розроблено математичну модель каскадної оптимізації ГЗМ, яка, на відміну від існуючих, базується на ієрархічній декомпозиції простору гіперпараметрів та врахуванні адаптивної динаміки навчання, що дозволяє забезпечити необхідну точність та швидкодію функціонування ГЗМ в умовах апаратно-параметричних обмежень кордонних пристроїв.

Вперше розроблено мультифазовий метод оптимізації навчання ГЗМ, який, на відміну від існуючих, базується на багаторівневому механізмі адаптивної конвергенції, що дозволяє запобігати колапсу моди та зникненню градієнтів функцій втрат без підвищення обчислювальної складності процедури навчання.

Удосконалено механізм адаптації ГЗМ, який ґрунтується на гібридному комбінуванні каскадного та мультифазового методів в поєднанні з апаратом нечіткої логіки, що дозволяє комплексно підвищити ефективність навчання та якість генерації штучних даних в задачах виявлення аномалій, біометрії та комп'ютерного зору, зокрема знизити функцію втрат генератора у 3,5 рази, уникнути перенавчання дискримінатора (на рівні ~92%), прискорити збіжність ГЗМ в 7,6 разів, покращити метрику FID у 2,3 рази та досягти точності виявлення аномалій AUC = 0,92 із забезпеченням повноти (Recall) на рівні 1,0.

Набув подальшого розвитку програмно-апаратний метод реалізації повного циклу функціонування ГЗМ на кордонних пристроях, який базується на інтеграції квантованого навчання та апаратно-орієнтованої каскадної

оптимізації, що забезпечує реалізацію реконфігурованих архітектур ГЗМ з врахуванням апаратно-параметричних обмежень та їх функціювання в режимі реального часу зі зменшенням розміру імітаційної моделі в 3,9 рази та прискоренням процесу інференсу в 3,2 рази.

Для оцінки продуктивності ГЗМ у роботі застосовано методи кількісного аналізу із використанням комплексу сучасних метрик (FID, Loss Value, Accuracy, Epochs to Convergence, Inference Time, Latency, Model Size, AUC, Recall), що дозволило об’єктивно порівняти ефективність розроблених методів із базовими підходами. Практична значущість дослідження підтверджена впровадженням результатів у наукові проєкти з Саарландським університетом ФРН та Інститутом проблем штучного інтелекту МОН і НАН України, а також в навчальний процес ЧНУ ім. П. Могили. Матеріали дисертаційної роботи доповідалися та обговорювалися на 8 міжнародних науково-технічних конференціях (IDAACS’ 2023, DESSERT’ 2023, ICTERI, ATIT, AICT, CMIS та ін.) та 3 Всеукраїнських наукових конференціях та семінарах, опубліковані в 17 наукових працях.

Практичне значення одержаних результатів. Положення роботи є актуальними для використання у галузі знань 12 «Інформаційні технології», а також мають прикладний характер, що робить їх цінними для практичного впровадження в реальних IoT-системах та вбудованих компонентах систем контролю, управління та прийняття рішень, а також при застосуванні у навчально-освітньому процесі. Практична значущість дослідження полягає у розробці алгоритмічно-апаратного фреймворку для навчання та розгортання ГЗМ на кордонних пристроях з врахуванням їх обчислювальних та архітектурних обмежень. Завдяки застосуванню методів квантування ваг (INT8) та профілювання обчислювальних графів доведено можливість стійкого функціонування ГЗМ у режимі реального часу на базі мікрокомп’ютерної

платформи Raspberry Pi 5. Це дозволяє інтегрувати розроблені моделі ГЗМ як базовий елемент інтелектуального контуру автономних IoT-платформ, що мінімізує локальне апаратне навантаження та усуває критичну залежність інфраструктури від хмарних обчислень. Теоретичні положення та практичні рекомендації, які обґрунтовано в дисертаційній роботі, впроваджені в науково-технічні розробки ЧНУ ім. Петра Могили, Саарландського університету, ФРН та ІПШІ НАН і МОН України, а також в навчальний процес ЧНУ ім. Петра Могили у ході підготовки фахівців за освітньо-кваліфікаційними рівнями «бакалавр» та «магістр», зокрема використовуються при викладанні дисциплін «Проєктування інтелектуальних СППР», «Нейромережеві методи обчислювального інтелекту» та «Fuzzy models and methods for computational intelligence», а також у дипломному проєктуванні для спеціальностей 123 — «Комп'ютерна інженерія» та 122 — «Комп'ютерні науки». Результати даної роботи щодо генеративних змагальних мереж ввійшли до проєкту «Стратегії розвитку штучного інтелекту в Україні», що відображено в колективній монографії.

Особистий внесок здобувача полягає у розробці та впровадженні каскадного, мультифазового фазового та гібридного методів оптимізації ГЗМ, розробці вдосконаленої архітектури адаптивної глибокої згорткової ГЗМ, а також у розробці нових підходів до покращення якості зображень, виявлення аномалій на основі ГЗМ, впровадженні моделей та методів навчання та розгортання ГЗМ на кордонних пристроях. Роботи опубліковано здобувачем як самостійно, так і у співавторстві; здобувачем проведено аналіз сучасних систем штучного інтелекту, зокрема генеративних моделей, та запропоновані підходи до підвищення їх ефективності; розглянуто структуру та основні програмні компоненти генеративних змагальних мереж при виконанні практичних задач з генерування реалістичних зразків рукописних символів,

підвищення роздільної здатності, синтезування штучних відбитків пальців, виявлення аномалій у багатокритеріальних даних; наведено методи та узагальнену стратегію апаратно-параметричної оптимізації ГЗМ для мультимодального спектра застосування на кордонних пристроях в умовах їх апаратно-параметричних обмежень; розроблено програмні засоби керованого навчання та синтезу зразків за допомогою генеративних моделей безпосередньо на кордонних пристроях.

Результати дисертаційної роботи впроваджені при виконанні наукового проєкту ІПШІ МОН і НАН України (№ ДР 0120U102299), міжнародного DAAD-Ostpartnerschaftsprogramm проєкту в рамках академічної співпраці «ЧНУ, Україна - Саарландський університет, ФРН» та в навчальний процес ЧНУ ім. Петра Могили.

Основні ідеї та розробки, здійснені в рамках дисертаційного дослідження, у тому числі ті, що характеризують наукову новизну і практичне значення результатів, сформульовані і отримані здобувачем особисто.

Внесок автора конкретизовано у переліку опублікованих праць.

Апробація результатів дисертації. Праці з основними результатами (5 публікацій): сюди входять 2 статті у міжнародних журналах (International Journal of Computing, Journal of Mobile Multimedia), 1 розділ в монографії «Artificial Intelligence in Control and Decision-making Systems» (Springer, серія Studies in Computational Intelligence), що індексуються в Scopus, та 1 стаття у фаховому виданні України категорії «Б» (журнал «Штучний інтелект»). Праці апробаційного характеру (7 робіт): це матеріали доповідей на міжнародних та Всеукраїнських науково-технічних конференціях (ATIT, AICT, DESSERT, CMIS, «ІІС» та «Могилянські читання»), з яких 4 включені в Scopus. До додаткових публікацій (5 робіт) віднесено 1 колективну монографію «Стратегія розвитку штучного інтелекту в Україні», 1 розділ у монографії

видавництва River Publishers та матеріали доповідей, що відображають аспекти цифрової еволюції та впровадження ШІ в різних галузях людської діяльності.

Особистий внесок у представлених публікаціях полягає у повній математичній формалізації каскадного та мультифазового методів оптимізації навчання ГЗМ. Самостійно розроблено програмну архітектуру моделей ADCGAN та EAWGAN, проведено серію чисельних експериментів з різними наборами даних (MNIST, SOCOFing, Birds 400), а також виконано збір та аналіз статистичних метрик FID, AUC, Recall та інших. Крім того, проведено повний цикл апаратної імплементації та оптимізації розроблених рішень для кордонних пристроїв на базі мікрокомп'ютерної платформи Raspberry Pi 5 із застосуванням методів квантування ваг та профілювання. Участь співавторів була сфокусована на а) спільному обговоренні ідей, концептуальних підходів, плану та послідовності досліджень при підготовці робіт до опублікування; б) загальному науковому консультуванні та науково-методичній підтримці під час проведення досліджень в рамках опублікованих робіт; в) редагуванні матеріалів публікацій.

Структура та обсяг роботи. Дисертація включає вступ, 5 розділів, загальні висновки та 3 додатки. Загальний обсяг роботи складає 240 сторінок, з яких 189 сторінок займає основний текст. Робота містить 8 таблиць і 50 рисунків. Список літератури нараховує 120 використаних джерел.

РОЗДІЛ 1. АНАЛІЗ ГЕНЕРАТИВНИХ ЗМАГАЛЬНИХ МЕРЕЖ ЯК СКЛАДОВОЇ СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

1.1. Генеративні змагальні нейронні мережі для вирішення задач штучної генерації реалістичних зразків

Генеративно-змагальні мережі (або генеративні змагальні мережі, генеративно-змагальні нейронні мережі, ГЗМ, англ. generative adversarial networks, GAN, GANN) є імплементацією класу алгоритмів машинного і глибокого навчання, які використовуються при навчанні без учителя (англ. unsupervised learning), та представляють собою архітектуру з двох штучних нейронних мереж, які змагаються між собою в рамках гри з нульовою сумою. Концепція була запропонована та сформована американським вченим-дослідником Іеном Ґудфеллоу у 2014 році [1].

ГЗМ здатні генерувати високоякісні штучні зразки, які можуть бути використані як додаткові навчальні дані для інших систем штучного інтелекту, наприклад, спеціалізованих нейронних мереж, що вирішують завдання бінарної класифікації (на базі логістичної регресії), сегментації, прогнозування та ін. Додаткові навчальні дані здатні підвищити точність існуючих моделей, що вже знаходить своє практичне застосування в медицині, електроенергетиці, військовому секторі або у сфері кібербезпеки.

ГЗМ також відіграють важливу роль в дослідженнях проблем комп’ютерного зору (англ. computer vision). Моделі ГЗМ здатні суттєво покращувати фотографії, отримані як телескопами, так і розвідувальними дронами, і використовувати методи зменшення розмитості (англ. deblurring), інпейнтингу (англ. inpainting) та надвисокої роздільної здатності (англ. super resolution). Це значно підсилює існуючі системи ГН, покращуючи їх можливості та результати.

В основі ГЗМ лежить антагоністична взаємодія двох НМ (генератора (G) і дискримінатора (D)), де G генерує нові зразки та мінімізує ймовірність їх D-виявлення, а D максимізує здатність розпізнавати згенеровані дані і відрізняти їх від реальних, що дозволяє ГЗМ ефективно навчатися генерації даних, максимально наближених до реальних.

В якості зразків можуть виступати як графічні зображення, так і інші типи функціональних даних, наприклад, структури молекул, матеріалів, або мережеві сигнатури. Перша (генеративна) мережа намагається сформувати новий зразок комбінуючи первинні зразки (навчальні дані), використовуючи для цього змінні латентного простору. Друга (дискримінаційна) мережа вчиться розрізняти справжні та «підроблені» (тобто згенеровані ГЗМ) зразки. При цьому результати аналізу подаються на вхід генеративної мережі так, щоб вона змогла підібрати ще кращий набір латентних параметрів, а дискримінаційна мережа вже не змогла б відрізнити справжні зразки (реальні дані) від штучно згенерованих.

Характерною ознакою ГЗМ є те, що мережа-генератор постійно збільшує частоту помилок дискримінаційної мережі, синтезуючи нові покращені зразки, що імітують властивості реальних даних із максимальною точністю.

Для навчання дискримінатора використовуються набори реальних даних. Процес навчання дискримінаційної мережі забезпечує надходження зразків з набору даних, доки дискримінатор не досягне відповідного стандарту точності. Спочатку генератор приймає випадково відсортовані дані із латентного простору і фактично генерує штучні зразки з *шуму* (стохастичних випадкових значень). Після цього екземпляри, синтезовані генератором, аналізуються і оцінюються дискримінатором. В обох мережах застосовується метод зворотного поширення помилки, що забезпечує створення все більш

кращих зразків генератором, а дискримінатор в свою чергу ефективніше оцінює (тобто класифікує або відсортовує) синтезовані зображення. В оригінальній моделі ГЗМ генератор та дискримінатор проєктуються як повнозв'язна штучна нейронна мережа — багатошаровий перцептрон (англ. multilayer perceptron).

Даний підхід дозволив створювати графічні зображення наближені до фотографій (або навіть ідентичні їм за спектром характеристик), які демонструють максимально реалістичні властивості. Згодом діапазон можливостей розширився, а технологія знайшла своє застосування і при створенні інших повнофункціональних зразків (не тільки графічних).

Ранній опис схожої ідеї використання парадигми змагальності належить американському вченому Денні Гіллісу, який дослідив таку методологію в своїй праці 1990 року, де вивчалися особливості паразитів, що спільно розвиваються та покращують змодельовану еволюцію, як процедуру оптимізації [2]. В якості елементів конкурування були використані біологічні моделі.

Ідея формування шаблонів навчання в середовищі з конкурентними умовами була розглянута у 2013 році в дослідженні «A Coevolutionary Approach to Learn Animal Behavior through Controlled Interaction» під авторством Wei Li, Melvin Gauci, Roderich Groß [3], де було запропоновано коеволюційний підхід до вивчення поведінки тварин за допомогою контрольованої взаємодії. Ідея змагального навчання полягає в тому, що машинне навчання вдосконалюється шляхом введення агента-супротивника.

Опис змагальних методів навчання можна зустріти й у більш ранніх дослідженнях, наприклад, у роботі Юрґена Шмідхубера 1992 року [4], де були розглянуті методи вивчення факторіальних кодів шляхом мінімізації передбачуваності.

1.2. Важливість застосування генеративних змагальних нейронних мереж у комп’ютерній інженерії

Генеративний штучний інтелект (ГШІ, англ. GenAI) і генеративні змагальні мережі революціонізують комп’ютерну інженерію, автоматизуючи та оптимізуючи критично важливі процеси в проєктуванні апаратного забезпечення, вбудованих та кордонних систем і оптимізації схем. Ці технології дозволяють інженерам знаходити нові рішення, що виходять за межі традиційних методологій, підвищуючи ефективність та продуктивність у розробці обчислювального обладнання.

Одним із найзначніших застосувань ГШІ у комп’ютерній інженерії є проєктування інтегральних схем (ІС). Традиційне проєктування схем вимагає значних знань, багаторазових симуляцій і евристичних правил. Однак експерименти демонструють [5], що глибоке навчання, поєднане з еволюційними алгоритмами, може автономно генерувати складні дизайни мікросхем. Цей метод здатен оптимізувати макети антен, фільтрів, розподільників потужності та інших напівпровідникових компонентів, іноді створюючи нестандартні, але водночас ефективні рішення, які часто суперечать людській інтуїції. Виготовлені на основі такого підходу інтегральні схеми демонструють характеристики, що відповідають реальним властивостям, що забезпечує їх практичне застосування [6, 7, 8].

ГШІ також відіграє важливу роль у розробці та оптимізації мікросхем. Методи глибокого навчання з підкріпленням використовуються для покращення макетів мікросхем шляхом аналізу попередніх ітерацій дизайну та навчання на основі зворотного зв’язку про продуктивність. Такий підхід значно скорочує час, необхідний для оптимізації встановлення чипів, що є традиційно трудомістким і ресурсозатратним завданням. Такі компанії, як

Google, успішно впроваджують моделі ШІ для прискорення розробки спеціалізованих процесорів, таких як Tensor Processing Units (TPU), які є критично важливими для розподілення апаратних навантажень ШІ [9].

Окремим перспективним напрямом застосування генеративних моделей у комп'ютерній інженерії є автоматизація проєктування апаратного забезпечення. Зокрема, новітні дослідження демонструють ефективність використання генеративних рушіїв для автоматичного виявлення та синтезу топологій аналогових інтегральних схем. Наприклад, інструмент AnalogGenie дозволяє не лише автоматизувати рутинні процеси, а й генерувати нові, раніше невідомі топології схем, використовуючи підходи на основі послідовних графових представлень (sequence-based graph representations), що вирішує проблему відсутності масштабних наборів даних у цій ніші [10].

Крім того, методології, керовані ШІ, використовуються для розробки нейроморфних чипів, натхнених людським мозком. За допомогою структурованої обробки природної мови такі моделі, як Gemini, Perplexity, ChatGPT, Claude, Grok та інші, допомагають дослідникам у створенні детальних інструкцій для проєктування мікросхем, прискорюючи розробку нейронних мереж. Ці проєкти на основі ШІ відкривають нові можливості для створення енергоефективних обчислювальних архітектур, прокладаючи шлях до розвитку кордонних обчислень та застосувань штучного інтелекту [11].

Окрім проєктування схем, ГЗМ сприяють покращенню безпеки та надійності апаратного забезпечення. Вони використовуються для створення змагальних сценаріїв, що перевіряють та підсилюють стійкість апаратного забезпечення до потенційних кіберзагроз. Симулюючи різноманітні вектори атак, ГЗМ допомагають інженерам розробляти більш стійкі механізми шифрування та захисту для вбудованих та кордонних систем і мережевих пристроїв. Крім того, ГЗМ сприяють генерації реалістичних сенсорних даних

для робото-технічних і автономних систем, покращуючи їхнє навчання та валідацію без необхідності проведення масштабних експериментів у реальному світі.

Інтеграція ГШІ та ГЗМ у комп'ютерну інженерію змінює підходи до проєктування, оптимізації та безпеки апаратних систем. Ці методи значно спрощують складні інженерні завдання. Зі зростанням можливостей моделей ШІ їхня роль у комп'ютерній інженерії буде лише розширюватися, що сприятиме подальшим досягненням у галузі обчислювального обладнання та інтелектуальних систем.

1.3. Сфери застосування ГЗМ та прикладна цінність технології відповідно до галузі

1.3.1. Дослідження космосу та прикладна фізика

ГЗМ можуть використовуватися для покращення астрономічних зображень [12, 13, 14,15] та імітувати гравітаційне лінзування для дослідження темної матерії [16]. ГЗМ знайшли застосування в якості точного способу моделювання високоенергетичних струменів [17]. ГЗМ також були успішно натреновані для точної апроксимації вузини (англ. bottleneck) при симуляції експериментів з фізикою елементарних частинок [18].

Застосування технології ГЗМ в контексті існуючих та запропонованих експериментів CERN продемонструвало стійкий потенціал методики для прискорення моделювання й поліпшення точності моделювання [19].

1.3.2. Охорона здоров'я та проблеми тривалості життя

Глибинне навчання дозволяє швидко ідентифікувати потужні інгібітори кінази DDR1 [20]. Описаний процес розробки нових молекул із застосуванням генеративних моделей тривав всього 21 день, а молекули пройшли успішне випробування на мишах. Дана методика довела ефективність впровадження

глибинного навчання та генеративних моделей для прискореного відкриття ліків.

Розроблена глибинна генеративна модель GENTRL (генеративне тензорне навчання з підкріпленням) показала успішні результати з точки зору дизайну малих молекул *de novo*; її було застосовано як систему для виявлення потужних інгібіторів рецептора домену дискоїдину 1 (DDR1), молекул, які беруть участь у регуляції функцій клітин та пов'язані з фіброзом та іншими захворюваннями. Очікується, що цей метод може бути додатково вдосконалений як перспективний підхід до ідентифікації потенційних ліків [20].

ГЗМ продемонстрували здатність успішного застосування при розробці нових молекул для різних цільових протеїнів, що викликають запалення, фіброз та рак [21]. Під час експерименту дві кондиційні ГЗМ були об'єднані як один функціональний ланцюжок глибинного навчання (кондиційна ГЗМ та ГЗМ Васерштейна з градієнтним штрафом (англ. Wasserstein GAN with gradient penalty, WGAN-GP)) для досягнення експериментальних очікувань, — друга мережа покращила результати першої.

Методи глибинного навчання та ГЗМ активно використовуються в медичній візуалізації (рентгенографія, магнітно-резонансна томографія, позитронно-емісійна томографія тощо) як потужний інструмент, що дозволяє медичним експертам та рентгенологам виявляти важкі патології та захворювання на ранніх стадіях з високим відсотком точності [22, 23, 24, 25, 26, 27].

Оскільки на сьогоднішній день кількість медичних зображень багатьох захворювань є недостатньою, існує нагальна потреба в додаткових джерелах інформації для навчання моделей. ГЗМ здатні синтезувати високоякісні штучні зразки, які можуть бути успішно застосовані як вирішення цієї

проблеми і значно покращити результативність аналізу медичних зображень [28].

ГЗМ знаходить своє застосування як допоміжна технологія при навчанні нейронних мереж для завдань класифікації та сегментування. Враховуючи те, що реальні медичні відомості про пацієнтів є важкодоступними та часто захищені лікарською таємницею і нормами захисту персональних даних, доцільно розглядати можливість розширення тренувальних наборів даних за допомогою згенерованих (синтезованих) графічних зразків — знімків МРТ або рентгенівських зображень, штучно створених за допомогою генеративно-змагальної мережі. При роботі з наборами даних, які складаються з графічних зображень будь-якого типу (в даному випадку — медичні знімки), можна або синтезувати нові зображення (зразки) або випадковим чином модифікувати зразки вже існуючих зображень та додати їх до набору даних, що використовується для навчання мережі або групи мереж в якості додаткових/нових зразків, збільшуючи таким чином загальну ємність та потужність використовуваного навчального набору даних; також за допомогою ГЗМ та інших інструментальних методів можна застосовувати випадкове обертання, зсув зображення або додавання сфабрикованих шумів [25, 27, 29].

Відсутність належного обсягу навчальних даних може призвести до перенавчання нейронної мережі, тому кращий спосіб уникнути цього — надати нейронній мережі достатню кількість якісних навчальних даних. Генеративне моделювання з використанням ГЗМ-методик може заповнити цю прогалину за рахунок посилення наявних наборів даних новими синтезованими високоякісними зображеннями.

Глибока згорткова архітектура ГЗМ (Deep Convolutional GAN) здатна створювати фотореалістичні графічні зображення, які високою мірою

відповідають розподілу ознак реальних зображень з точки зору статистичної оцінки [14]. Комбінація двох генеративних моделей StackGAN та DCGAN продемонструвала здатність створення штучних графічних зразків із навіть більш високою роздільною здатністю зображення [15, 30].

Застосування архітектури Cycle-GAN було успішно впроваджено у плануванні променевої терапії для КТ та МРТ-знімків пацієнтів з пухлинами головного мозку [31]. Результати продемонстрували, що Cycle-GAN перевершує стандартну одиночну ГЗМ, яка працює зі спареними зображеннями. Також ГЗМ було використано при прогнозуванні можливості того, чи є у пацієнта рідкісне захворювання чи ні [32]; точність прогнозування була на 5% вищою у порівнянні зі стандартними методами.

Модель MedGAN використовує графові згорткові нейронні мережі для створення нових молекул із хімічною структурою, основою яких є хінолін. MedGAN ґрунтується на Wasserstein GAN і спеціалізується на генеруванні нових молекул, аналізуючи взаємозв’язки між атомами, забезпечуючи високу вірогідність генерації хінолінових молекул з унікальними властивостями [33].

1.3.3. Матеріалознавство

Так само, як індустріальна революція (що полягала у створенні машин, які могли б виконувати механічні завдання ефективніше, ніж люди), в галузі машинного навчання машини прогресивно навчаються визначати закономірності та знаходити зв’язки між властивостями та особливостями даних більш ефективно, ніж експерти. У матеріалознавстві машинне навчання здебільшого застосовується для вирішення завдань класифікації та регресивного аналізу.

Комбіновані методики машинного навчання і ГЗМ підтвердили свою прикладну результативність при відкритті нових матеріалів та прогнозуванні їх кристалічної структури, що знайшло відображення в роботі про останні

досягнення та застосування машинного навчання у твердотільному матеріалознавстві [34].

Одним із прогресивних напрямів досліджень у цій галузі є розробка принципово нового підходу для створення нових потрійних стійких кристалографічних структур із спостережуваних бінарних, тобто з таких, що містять лише два хімічні елементи.

ГЗМ було успішно застосовано для відкриття кристалографічних структур. Модель CrystalGAN продемонструвала ефективність у задачах виявлення міждоменних зв'язків у реальних даних та створення нових структур. Запропонована технологія може ефективно інтегрувати попередні знання, зібрані експертами [35]. CrystalGAN — перша генеративна змагальна нейронна мережа, розроблена для отримання наукових даних у галузі матеріалознавства. Також це перший підхід, який дозволив генерувати дані про структурну складність більш високого рівня, тобто дані про потрійні структури, де домени добре відокремлені від спостережуваних бінарних сполук. Метод CrystalGAN продемонстрував перспективні результати при вирішенні завдання з відкриття нових матеріалів для зберігання водню [35].

1.3.4. Типографія та рукописні зразки

ГЗМ також демонструє ефективність як інструмент для створення нових шрифтів та унікальних рукописних символів, таких як цифри та літери. Стандартний метод ГЗМ у поєднанні з оптимізатором Adam був протестований на наборі даних MNIST [36]. Експеримент продемонстрував потенціал цього підходу для створення високоякісних графічних зразків рукописних символів: переважна більшість отриманих зразків виглядали подібними до реальних цифр, їх можна було відрізнити за достатнім рівнем чіткості, структури, форми та не було значних графічних шумів [36].

Експериментальне коригування параметрів такої моделі має потенціал для покращення отриманих результатів.

Слід також зазначити, що подібна архітектура може бути використана у типографіці для створення нових моделей шрифтів, а також у прикладній криміналістиці для створення нових навчальних рукописних зразків, які можуть бути використані як частина наборів даних для навчальних систем розпізнавання рукописного тексту та ідентифікації.

1.3.5. Міждисциплінарні сфери застосування

ГЗМ можуть застосовуватися для реалізації різноманітних задач, а саме: для створення фотореалістичних зображень, які візуалізують новий дизайн елементів інтер'єру або промислових виробів (телескопів, роботів, корпусів ракет, сцен у комп'ютерних іграх і т.і. [37].

Удосконалену ГЗМ з автоматичною генерацією текстур було використано для поліпшення якості фотографій; акцент було зроблено на створенні реалістичних текстур, а не на піксельній деталізації. Результатом стала висока якість зображення при високій роздільній здатності [38].

Китайські дослідники представили модель *SSGAN*, яка здатна забезпечити ефективну стеганографію на основі генеративних змагальних мереж; було доведено що ГЗМ ефективні для створення більш релевантного та безпечного прикриття для **стеганографії** — спеціальної форми закодованого тайнопису, де приховане повідомлення не виглядає повідомленням як таким (на відміну від криптографії). Запропонована модель є ефективною для створення зображень, які мають більш високу візуальну якість [39].

1.3.6. Криміналістика

ГЗМ продемонстрували ефективність як допоміжна технологія для створення спеціальних наборів даних цифрових, синтезованих відбитків

пальців, використання яких не порушує засад конфіденційності. Якщо реальні біометричні зразки відбитків пальців відносяться до категорії персональних даних, а отже їх використання для навчання моделей може мати певні обмеження, то синтезовані зразки є повністю штучними, не мають відношення до реально існуючих людей і можуть використовуватися вільно («privacy-friendly samples») [40, 36, 41].

Згенеровані зразки штучних відбитків пальців представляють інтерес для різного роду прикладних досліджень: біологічних (вивчення папілярних ліній, зміна їх структури під впливом часу), криміналістичних (комп'ютерна ідентифікація слідів злочину, встановлення належності відбитків, реконструкція і відновлення пошкоджених відбитків), технологічних (різноманітні методи використання біометричних електронних замків).

Таким чином, ГЗМ дає змогу проводити дослідження відбитків пальців, без обмежень, пов'язаних з конфіденційною природою біометричних даних.

Контрольована генерація зразків за допомогою ГЗМ може також застосовуватись для створення фотороботів підозрюваних у скоєнні злочину, ідентифікації підробок (фото, відео, аудіо), покращення зображень, зафіксованих камерами спостереження.

1.3.7. Кібербезпека і проблематика Deep Fake

В сфері застосування шкідливого програмного забезпечення (англ. *malware*) ГЗМ також продемонстрували ефективність, як у системах кіберзахисту так і кібератак. Такі модифікації ГЗМ як MalGAN, tGAN, tDCGAN здатні створювати сигнатури шкідливого програмного забезпечення та обходити детектори на основі машинного навчання; також моделі на базі ГЗМ можуть класифікувати та виявляти атаки [42, 43, 44, 45, 46, 47,48].

PassGAN, CrackStation, RainbowCrack та HashCat — інструменти, які у комбінації створюють потужний фреймворк для підбору та зламу паролів, навіть максимально високої категорії складності [49].

Результати експериментального дослідження свідчать, що PassGAN вдалося вгадати 43,6% унікальних паролів зі списку слів Rockyou.txt (1 350 178 з 3 094 199) і 24,2% унікальних паролів з набору даних LinkedIn (10 478 322 з 43 354 871).

З метою оцінки дослідники виключили з тестового набору всі паролі, які були в навчальному наборі. Після цього PassGAN показав 34,6% і 34,2% відповідності для зразків Rockyou.txt і LinkedIn, відповідно. Крім того, навіть ті паролі, які не збігалися зі зразками з набору для тестування, містили всі властивості справжніх паролів, створених людським інтелектом; тому все ще можуть бути застосовані для зламу реальних облікових записів користувачів, незважаючи на відсутність у розглянутих наборах даних.

Варто також зазначити, що PassGAN у поєднанні з HashCat зламав на 51–73% більше унікальних паролів, ніж сам HashCat [49].

Останнім часом системи виявлення мережевих вторгнень на основі глибинного навчання досягли надзвичайно високого рівня ефективності виявлення [50]. Це саме та царина, де контрольована генерація може становити серйозну загрозу.

DoS-WGAN — модель ГЗМ, що дозволяє обходити системи детекції мережевих вторгнень [51]. Базою даного фреймворка є WGAN (Wasserstein Generative Adversarial Aetwork) з градієнтним штрафним підходом [51]. Щоб замаскувати зловмисні DoS-атаки під звичайний мережевий трафік, DoS-WGAN автоматично генерує сліди атак, які можуть обійти сучасні NIDS (Network Intrusion Detection System) і протоколи безпеки проти DoS-загроз. Модель створює трафік атаки, який дуже схожий на звичайний і тому не може

бути розпізнаний алгоритмами виявлення. Рівень виявлення NIDS на основі згорткових нейронних мереж знизився з 97,3% до 47,6% після того, як дослідники ввели згенерований трафік DoS-атаки в систему виявлення [51].

Отже цей аспект ГЗМ потребує особливо ретельного дослідження і аналізу.

Виявлення вторгнень і ботнетів є ще однією галуззю кібербезпеки, де фреймворки на основі ГЗМ знаходять все більш широке застосування. ГЗМ можуть синтезувати ворожий «зловмисний» трафік, спрямований на атаку детекторів без подальшої ідентифікації; ця методика дозволяє ефективно уникати детекторів побудованих на основі машинного навчання.

Виявлення вторгнень зазвичай здійснюється спеціальними системами, які відстежують мережу на наявність зловмисної активності або порушень правил (політик безпеки) і реалізовані у вигляді програмних або апаратних застосунків. Системи виявлення вторгнень (NIDS) відповідають за виявлення мережевих атак, що здійснюються зловмисним трафіком. Сьогодні алгоритми машинного навчання все частіше використовуються у системах виявлення вторгнень. Незважаючи на певний рівень ефективності таких систем, досі існують сумніви щодо їх надійності.

IDSGAN — інноваційний фреймворк, що базується на Wasserstein GAN і містить генератор, дискримінатор і чорну скриньку IDS — IDSGAN [52]. Основне завдання моделі полягає в синтезі ворожих сигнатур зловмисного трафіку, спрямованих на атаку систем детекції шляхом спуфінґу (маскування) і ухилення від виявлення. IDSGAN продемонструвала переконливі результати для неоднорідних типів атак, вплинувши на зниження показників виявлення різноманітних black-box IDS архітектур до 0.

Ще один експеримент [53] підтвердив вразливість систем ідентифікації на основі МН до змагального збурення, яке може призвести до порушення

функцій NIDS шляхом упорскування крихітного, незначного збурення до мережевого трафіку. Дослідники розробили змагальну ГЗМ-атаку, яка може ефективно уникати NIDS, що використовують алгоритми машинного навчання. Крім того, було виявлено, що ГЗМ можна застосовувати для «інокуляції» NIDS, щоб вони могли більш ефективно реагувати на змагальні збурення і потенційні загрози.

ГЗМ продемонстрували успішність при застосуванні для виявлення ботнетів. Ботнети є однією з найсерйозніших загроз у кібербезпеці й часто використовуються для масштабних зловмисних атак. Одним із головних завдань мережевої безпеки сьогодні є точне виявлення ботнетів.

Для розширення методів виявлення ботнетів запропонована модель на основі ГЗМ — Bot-GAN [54]. Отримані результати довели, що Bot-GAN є ефективним і може покращити класичні системи виявлення. Bot-GAN продемонстрував суттєві покращення з точки зору ефективності виявлення ботнетів, а також зниження рівня помилкових спрацьовувань.

Глибинні підробки або «deep fakes» — фальшиві графічні, відео- або аудіо-файли — становлять серйозну загрозу особистій безпеці людини, оскільки сфабриковані медіа-дані можуть бути використані для дискредитації особи шляхом нанесення шкоди її репутації (фальшива порнографія, фейкові новини, шахрайство, фінансові махінації тощо); вони можуть спровокувати політичну нестабільність, насильство або навіть військовий конфлікт.

Глибинні підробки зазвичай створюються за допомогою певних типів штучних нейронних мереж — *автокодерів, дифузійних моделей* і ГЗМ.

У випадку з автокодерами, модель за допомогою ланцюжка кодер-декодер дозволяє замінити обличчя однієї людини іншою (у відео чи фотографії; програми Reface, DeepNude та ін.).

Одним із ефективних інструментів, що покращують можливості глибинних підробок, є ГЗМ. Вона навчає декодер (генератор) та дискримінатор у змагальній взаємодії, що ускладнює ідентифікацію сфабрикованих даних, оскільки дві мережі послідовно розвиваються. Як тільки синтезована підробка буде виявлена, модель негайно виправить дефект, і подальше його виявлення вже буде вкрай ускладненим.

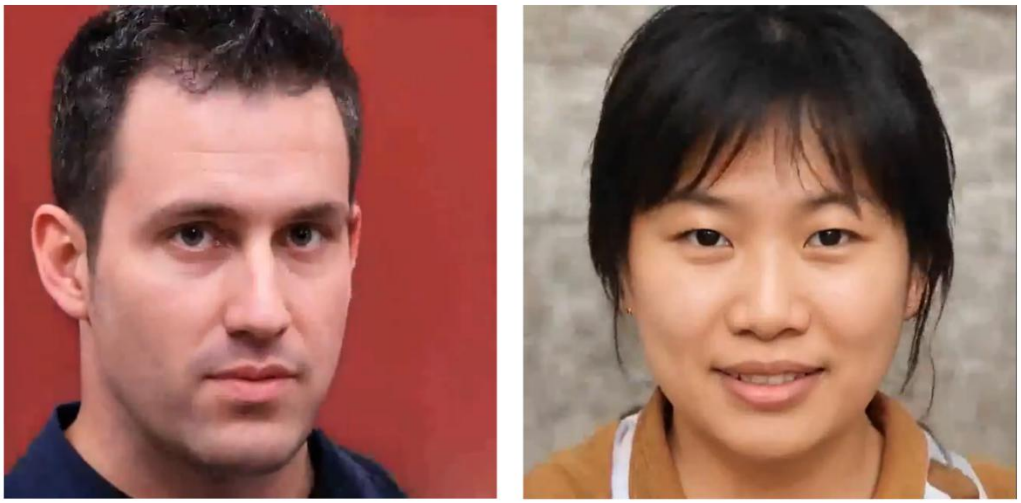

Рисунок 1.1. Обличчя людей, синтезовані автором дослідження за допомогою ГЗМ (StyleGAN).

Через підвищення якості deepfake-зразків, методи їх виявлення також потребують вдосконалення. Запропоновано навіть створити контрольний набір даних зі зразками глибинних підробок, що може допомогти у розробці ефективних методів їх виявлення. Це має спростити процес навчання алгоритмів виявлення, які вимагають великих обсягів навчальних даних.

Для кращого розуміння можливих методів протидії підробкам дуже важливо ретельно вивчати ГЗМ. Один із вже досліджених методів пропонує виявляти синтезовані підробки шляхом аналізу згорткових слідів (англ. convolutional traces). Цей підхід заснований на вивченні цифрових «відбитків» для ідентифікації згенерованих зображень та відмежування їх від реальних даних.

Таблиця 1 узагальнює предметну область, технології і моделі та їх ключові характеристики, які розглянуті вище.

Таблиця 1. Перелік галузей, моделей ГЗМ і їх характеристик.

Галузь	**Технології**	**Характеристика**
Дослідження космосу та прикладна фізика	GAN, SGAN, Morpheus, Fader	Моделі ГЗМ і глибинного навчання для відновлення характеристик у зображеннях галактик і фотометричної оцінки червоного зміщення. Активне впровадження у дослідженнях у галузі космології. Обмежена місткість навчальних наборів; орієнтація на предметні знання галузі; створені зразки зображень залежать від розміру навчальних наборів даних.
Охорона здоров'я та проблеми старіння	GAN, GENTRL, WGAN	Високоефективний підхід із перспективними результатами; подальше застосування методів на основі ГЗМ та їх обмеження в контексті дослідження механізмів старіння, принципів довголіття та винайдення нових ліків активно вивчаються.
Матеріалознавство	GAN, Crystal-GAN	Результативні експерименти з проєктуванням нових матеріалів.

Типографіка та рукописні зразки	GAN, DCGAN	Продемонстрована ефективність при створенні зразків рукописних символів, текстів, зразків шрифтів. Вимагає високоякісних наборів даних і складних алгоритмічних методів оптимізації.
Міждисциплінарні сфери застосування	GAN, SSGAN	Створення фотореалістичних зображень, які візуалізують новий промисловий дизайн; автоматична генерація для поліпшення якості фотографій; забезпечення ефективної стеганографії.
Криміналістика	GAN, ADCGAN	Ефективність при створенні спеціальних наборів даних цифрових, синтезованих відбитків пальців. Контрольована генерація може також застосовуватись для створення фотороботів підозрюваних у скоєнні злочину, ідентифікації підробок, покращення зображень обличь, зафіксованих камерами спостереження.
Кібербезпека і проблематика Deep Fake	MalGAN, tGAN, PassGAN, StyleGAN, CycleGAN, DeepFD, DoS-WGAN, IDSGAN, Bot-GAN	Одна з найбільш чутливих царин застосування ГЗМ, яка охоплює наступне: злом паролів, глибинні підробки фотографій, відео, аудіо, DDoS-

		атаки і захист від них, виявлення і протидія ботнетам.

Можна дослідити й проаналізувати більше прикладів, але перерахувати їх всі навряд чи можливо, оскільки технологія ГЗМ стрімко розвивається і нові дослідження з'являються фактично кожного дня. Розглянуті вище приклади є практичним підтвердженням високої ефективності використання генеративних змагальних нейронних мереж у широкому діапазоні прикладних областей в контексті обробки графічних і функціональних зразків.

В якості альтернативи ГЗМ також розглядаються інші генеративні моделі для подібного кола завдань:

- Варіаційні автокодери (VAE [90], β-VAE, VQ-VAE [91]) — мають оцінку щільності, стабільне навчання, кращу різноманітність, але **(а)** якість синтезованих зразків набагато нижча, ніж може забезпечити ГЗМ, і **(б)** повільна швидкість навчання.
- Авторегресійні моделі (PixelCNN [92], WaveNet, GPT) — більш різноманітні зразки, але навчання вимагає нагляду/підкріплення (supervision).
- Потокові моделі (NICE, RealNVP, Glow [93]) — згенеровані зразки мають нижчу якість.
- Гібридні моделі (VAE-GAN [94], VQGAN) — менш стабільні.
- Дифузійні моделі (DDPM [95], Stable Diffusion, DALL-E) — забезпечують точнішу генерацію у специфічних випадках, але вимагають більшої обчислювальної потужності.

ГЗМ є лише однією з багатьох технологій генерації даних. Кожна з цих технологій має свої переваги та недоліки. ГЗМ, в свою чергу, демонструють високі результати у реалістичній генерації зразків з меншими витратами часу

та обчислювальних ресурсів. Це робить ГЗМ особливо ефективними для застосування в умовах, де важлива швидкість, точність та реалістичність генерації. Зокрема, ГЗМ можуть ефективно розгортатись на кордонних пристроях, мікрокомп'ютерних платформах та в автономних вбудованих системах контролю, де критично важливою є генерація та обробка даних у режимі реального часу без залежності від хмарної інфраструктури.

1.4. Перспективне застосування і впровадження генеративних змагальних нейронних мереж

З огляду на експоненційне зростання обсягів даних, науковій спільноті та технологічній індустрії потрібні не лише ефективні інструменти для їх обробки, але й архітектури, здатні функціювати в умовах децентралізованих інфраструктур. Сьогодні дослідники мають безпрецедентний доступ до передових генеративних технологій ШІ для пошуку, відновлення графічних зображень, синтезу функціональних зразків та аналітики загроз. Сучасні моделі ГН дозволяють знаходити складні нелінійні патерни в масивах інформації, проте їхня практична цінність дедалі більше залежить від можливості розгортання цих систем поза межами великих хмарних кластерів.

Наведений вище аналіз успішних прикладів застосування ГЗМ підтверджує високу ефективність цього різновиду архітектур в астрономії, кібербезпеці, виявленні deepfake-контенту, матеріалознавстві та медицині. Утім, сучасний етап розвитку цих областей вимагає перенесення обчислювальних потужностей ближче до джерела даних — на рівень вбудованих та автономних систем.

Обробка інструментальних сигналів та зображень за допомогою ГЗМ здатна суттєво прискорити реконструкцію даних навіть для невивчених явищ. ГЗМ ефективно видаляють шум і відновлюють графічні чи спектральні дані з пошкодженими пікселями або інструментальними артефактами. Таким чином,

ГЗМ залишаються однією з найбільш перспективних технологій, проте їхнє впровадження в автономні системи (наприклад, бортові комп'ютери БпАК або мобільні IDS/IPS сенсори) жорстко лімітується наявними апаратними ресурсами.

Хоча ГЗМ, як відносно нова архітектура (з 2014 року), вже здійснила ссув парадигми у генерації контенту та розв'язанні питань комп'ютерної безпеки (моделювання TTPs, аналіз мережевих аномалій, синтез ботнет-трафіку), подальше масштабування технології стикається з фундаментальними бар'єрами.

Що стосується недоліків і обмежень технології, ключовою проблемою сьогодні є надмірна параметрична ємність більшості ГЗМ. Для навчання та навіть прямого інференсу таких мереж потрібні значні обсяги відеопамяті (VRAM), висока пропускна здатність шини даних та значні енергозатрати. Складні архітектури з мільйонами прихованих параметрів стають «вузиною» при спробі їх інтеграції в мікрокомп'ютерні платформи та IoT-пристрої. Використання таких інструментів без належної архітектурної оптимізації унеможливлює їх роботу в системах жорсткого реального часу.

Незважаючи на це, оптимізовані системи на основі ГН продовжують сприяти прогресу в прикладних дослідженнях. Отже, перспективи їх подальшого впровадження безпосередньо залежать від розвитку методів апаратно-програмного ко-дизайну. Вдале поєднання алгоритмів структурного стиснення із сучасними мікроархітектурами може призвести до значних результатів у розгортанні ШІ на кордонних пристроях.

Основним завданням даного розділу є поглиблений науковий аналіз сучасних практичних підходів щодо застосування ГЗМ, з акцентом на виявленні критичних недоліків у їхній архітектурі. Науковий аналіз меж

застосовності цих моделей в умовах дефіциту ресурсів є ключовим аспектом, що визначає напрямки подальших досліджень з їх вдосконалення.

Відповідно до огляду літератури, ГЗМ знайшли широке застосування в різних областях. Однак існуючі дослідження здебільшого концентруються на максимізації якості синтезу за рахунок необмежених обчислювальних ресурсів. У даній дисертації зроблено акцент на інженерно-оптимізаційному підході. Зокрема, увага зосереджена на тому, як методи зменшення розмірності, квантування ваг та коригування гіперпараметрів можуть суттєво знизити обчислювальне навантаження ГЗМ, зберігаючи високу якість синтезованих зразків та стабільність моделей в обмежених апаратно-параметричних умовах, що раніше не було достатньо досліджено.

Попри численні дослідження ГЗМ, залишається недостатньо вивченим вплив специфічних методів стиснення на стабільність змагальної взаємодії між генератором і дискримінатором. У цьому дослідженні пропонується новий підхід до підтримання належної стійкості ГЗМ шляхом інтеграції багатоетапних, ієрархічних оптимізаційних стратегій. Це дозволить не лише знизити параметричну складність мережі, але й мінімізувати ймовірність колапсу моди. Такий підхід відкриває нові можливості для автономного застосування ГЗМ на кордонних пристроях та мікрокомп’ютерних платформах.

1.4.1. Розгортання ГЗМ на вузлах кордонних обчислень та децентралізована генерація даних

Перенесення обчислювального навантаження з централізованих хмарних серверів на кордонні пристрої є стратегічним напрямом вирішення проблем затримки та безпеки передачі даних. У контексті стратегічної аналітики загроз безперервна трансляція сирих мережевих пакетів або телеметрії на центральний вузол створює додаткові вектори атак і

перевантажує канали зв’язку [96]. Інтеграція концепції федеративних ГЗМ (англ. Federated GANs) у кордонні системи дозволяє локальним вузлам самостійно синтезувати репрезентативні вибірки даних для навчання систем виявлення вторгнень [97]. Таким чином, кордонні пристрої забезпечують децентралізоване моделювання нормальної поведінки системи та виявлення аномалій нульового дня безпосередньо поблизу джерела даних, гарантуючи при цьому конфіденційність вихідної інформації [98].

1.4.2. Апаратно-програмна оптимізація генеративних архітектур для вбудованих систем контролю

Впровадження генеративних моделей у вбудовані системи контролю становить складну інженерну задачу через жорсткі апаратні обмеження таких пристроїв: лімітований обсяг оперативної пам’яті, низьку тактову частоту процесорів та суворі вимоги до енергоспоживання. Для забезпечення стабільного інференсу ГЗМ у системах реального часу застосовується комплекс методів структурного стиснення. Ключову роль відіграє квантування ваг [99] та алгоритми відсікання неінформативних нейронних зв’язків [100]. Крім того, надзвичайно ефективним підходом є дистиляція знань, у процесі якої ресурсномістка модель-вчитель передає узагальнені ознаки розподілу даних компактній моделі-учню [101]. Це дозволяє адаптувати архітектуру генератора для автономного виконання на мікрокомп’ютерах промислових або спеціалізованих оборонних комплексів із мінімальною втратою репрезентативної здатності.

1.4.3. Імплементація ГЗМ на сучасних мікрокомп’ютерних платформах

Практична реалізація нейромережевої генерації на рівні портативних та мобільних пристроїв базується на використанні мікрокомп’ютерних

архітектур класу System-on-Chip (SoC), оснащених апаратними прискорювачами ШІ — тензорними процесорами (NPU або Edge TPU) [102]. На відміну від універсальних мікропроцесорів, ці співпроцесори апаратно оптимізовані для паралельного виконання матричних операцій, які складають основу згорткових шарів ГЗМ [103]. Використання спеціалізованих фреймворків та оптимізуючих компіляторів (наприклад, TensorRT для мікрокомп'ютерів родини NVIDIA Jetson або оптимізаторів TensorFlow Lite для платформ із модулями Google Coral) суттєво прискорює прямий інференс нейромережі [104]. Це відкриває можливості для автономного синтезу високоточних даних та аналітичної обробки складних сигналів безпосередньо на місці їх збору, що є фундаментальною вимогою для систем тактичного моніторингу.

1.4.4. Автономна робототехніка та бортові обчислювачі безпілотних авіаційних комплексів (БпАК)

Інтеграція ГЗМ у системи керування автономною робототехнікою та бортові комп'ютери БпАК вирішує критичну проблему перенесення досвіду з віртуального середовища в реальне [105]. В умовах відсутності або пригнічення сигналів супутникової навігації (GPS/GNSS-denied environments) автономні апарати покладаються на технічний зір та інерціальні системи. Використання генеративних моделей на етапі навчання дозволяє синтезувати високореалістичні набори мультимодальних сенсорних даних (хмари точок, інфрачервоні та оптичні знімки ландшафту за різних погодних умов) для забезпечення стабільної доменної адаптації [106]. Розгортання оптимізованих генераторів безпосередньо на бортових архітектурах дозволяє робото-технічним комплексам у режимі реального часу імітувати відсутні або

спотворені фрагменти сенсорного потоку, що суттєво підвищує стійкість навігаційних алгоритмів до зовнішніх збурень [107].

1.4.5. Цифрова обробка сигналів та когнітивна радіоелектронна боротьба (РЕБ)

Еволюція засобів радіоелектронної боротьби зумовила перехід від статичних бібліотек радіоелектронних сигнатур до систем когнітивної РЕБ (англ. Cognitive Jamming), де ГЗМ відіграють роль ядра для динамічного моделювання спектру. У задачах цифрової обробки сигналів архітектура ГЗМ адаптується для роботи з одновимірними часовими рядами (1D-CNN) або частотно-часовими представленнями (спектрограмами та мікро-допплерівськими сигнатурами) [108]. Змагальний принцип дозволяє мережі-генератору синтезувати правдоподібні радіочастотні випромінювання та нові радарні хвильові форми, маскуючи власні канали зв’язку або генеруючи складні імітаційні завади для дезорієнтації ворожих систем радіолокації [109]. Водночас дискримінатор або умовна генеративна архітектура стає високоефективним інструментом спектральної аналітики та синтезу радіолокаційних зображень (наприклад, SAR-апертури), здатним виявляти приховані аномалії та ідентифікувати новітні типи радіоелектронних загроз на фоні інтенсивного теплового шуму [110].

1.5. Висновки за розділом 1

З врахуванням вищевикладеного у даному розділі:

1) Проведено критичний аналіз актуальних сучасних і класичних джерел, які присвячені висвітленню теоретичної основи і практичної імплементації ГЗМ.

2) Проведено комплексний міждисциплінарний аналіз ГЗМ (як систем штучного інтелекту) з метою розширення області їх застосування при

реалізації на кордонних пристроях і мікрокомп’ютерних платформах та визначення напрямків їх можливої оптимізації для розв’язання прикладних задач генерації штучних та аналізу і обробки реальних даних.

3) Встановлено необхідність: проведення комплексного аналізу сучасних математичних моделей та алгоритмів навчання ГЗМ, висвітлення критичних недоліків алгоритмічно-організаційних процедур оптимізації ГЗМ та їх існуючих архітектурних варіацій, а також визначення шляхів подолання цих недоліків і подальшого вдосконалення реалізованих на кордонних пристроях ГЗМ.

РОЗДІЛ 2. МАТЕМАТИЧНІ МОДЕЛІ І ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ ПРОЦЕСІВ НАВЧАННЯ ГЕНЕРАТИВНИХ ЗМАГАЛЬНИХ МЕРЕЖ

2.1. Математичні моделі та алгоритми навчання ГЗМ

Математична модель ГЗМ базується на ідеї пошуку рівноваги Неша в мінімаксній грі для двох гравців. У цій грі мережа-генератор намагається мінімізувати функцію втрат, яка є різницею між синтезованими даними, які вона створює, і реальними даними, тоді як мережа-дискримінатор намагається максимізувати функцію втрат шляхом правильної ідентифікації синтезованих даних. Дві мережі навчаються разом, доки не буде досягнуто рівноваги Неша, після чого генератор починає створювати такі дані, які неможливо відрізнити від реальних, а дискримінатор не в змозі відрізнити синтезовані дані від реальних.

Цільова функція ГЗМ (англ. objective function) — це функція, яку мережі генератора та дискримінатора намагаються оптимізувати під час навчання.

Алгоритм першої архітектури ГЗМ, представлений Іеном Гудфеллоу, математично можна описати таким чином [1]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))], \quad (2.1)$$

де G представляє мережу-генератор; D представляє мережу-дискримінатор; x це вектор — зразок реальних навчальних даних; z є шумом або вектором прихованого простору, отриманим із стандартного нормального розподілу; $p_z(z)$ означає апріорні змінні вхідного шуму; $p_{data}(x)$ означає апріорні змінні вхідних реальних даних; $\mathbb{E}$ це очікування; $D(x)$ це вихід мережі дискримінатора, який представляє ймовірність того, чи x дійсно походить від реальних даних, чи від генератора; $V(D,G)$ це функція значення D та G у мінімаксній грі для двох гравців [1].

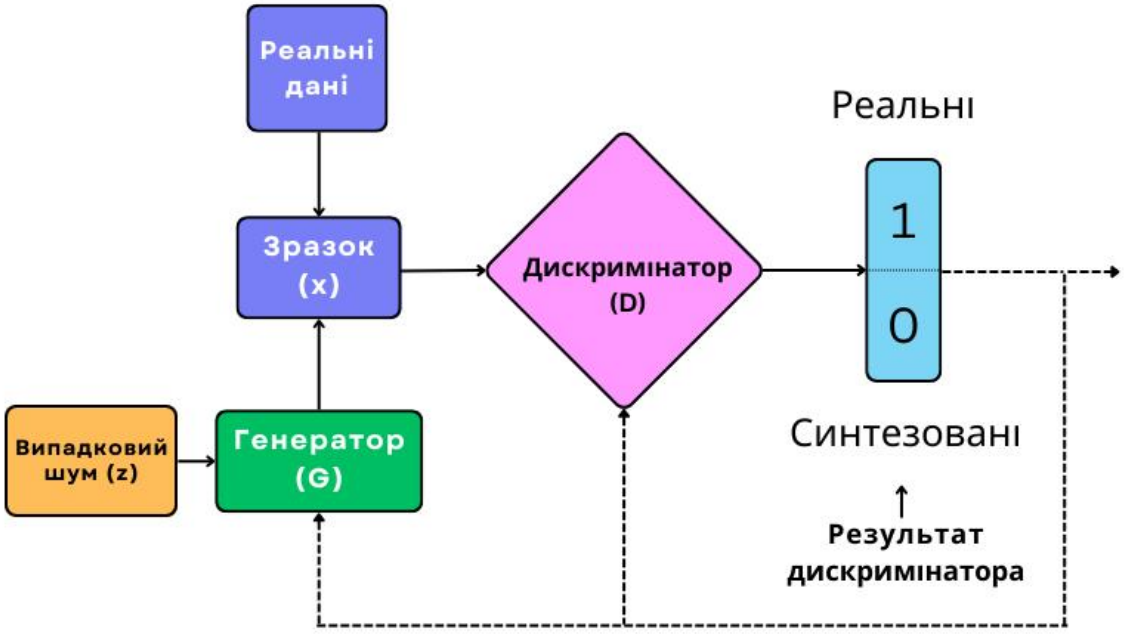

Рисунок 2.1. Блок-схема ГЗМ.

2.1.1 Цільові функції

На сьогоднішній день для навчання ГЗМ використовується багато різних цільових функцій (або функцій втрат), кожна з яких має свої переваги і недоліки. Нижче наведена стисла характеристика тих з них, які використовуються найчастіше і добре зарекомендували себе з точку зору рівня ефективності і точності.

Функції втрат відіграють вирішальну роль у навчанні мереж генератора та дискримінатора в ГЗМ, для яких доступні різні їх варіанти, і вибір відповідних функцій втрат для конкретного завдання, що поставлено перед ГЗМ, є важливим для досягнення оптимальних результатів. Вибір ефективної функції втрат для ГЗМ поєднує в собі експериментальне мислення, досвід і теоретичні знання.

Це має значний вплив на якість згенерованих вихідних даних. Розглянемо деякі з найпоширеніших функцій втрат, що використовуються в

ГЗМ. Аналізуючи характеристики цих функцій втрат, ми отримуємо краще розуміння їхніх сильних сторін і обмежень.

Мінімаксна гра (оригінальна ГЗМ Гудфеллоу). Цільова функція, яка використовується для навчання оригінальної ГЗМ, відрізняється від тих, що використовуються для інших типів нейронних мереж. ГЗМ Гудфеллоу використовує цільову функцію *мінімакс* (рівняння 2.1) [1], де генератор намагається мінімізувати різницю між розподілом синтетичних і реальних даних, тоді як дискримінатор намагається максимізувати різницю між ними. Втім, ця неопукла цільова функція може призвести до труднощів у конвергенції та стабільності під час навчання. Головними недоліками є схильність до колапсу моди і чутливість до гіперпараметрів (рівняння 2.1).

Бінарна перехресна ентропія (або БПЕ). БПЕ — це часто використовувана функція втрат у ГЗМ, яка вимірює різницю між виходом дискримінатора та цільовою міткою. Втрата БПЕ використовується для оновлення ваг дискримінатора під час процесу навчання та може бути описана таким чином:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\, h\left(x^{(i)},\theta\right) + \left(1-y^{(i)}\right)\log\left(1-h\left(x^{(i)},\theta\right)\right)\right], \quad (2.2)$$

де m це кількість зразків у всій партії, $-\frac{1}{m}\sum_{i=1}^{m}$ це середня втрата всієї партії, h представляє прогнози, зроблені моделлю, y представляє мітки для різних зразків, x представляє властивості, які передаються через передбачення, θ представляє параметри модуля, що обчислює передбачення.

Функція втрат Васерштейна (Wasserstein GAN або WGAN). Використовує відстань Васерштейна, також відома як Відстань рушія землі (англ. Earth Mover's Distance, EMD) як цільову функцію — це функція втрат на основі відстані, яка вимірює різницю між розподілом реальних і синтетичних даних. [59]. Це забезпечує більш стабільне навчання та кращі градієнти,

зменшуючи ймовірність колапсу моди. Використовує *критика* замість традиційного дискримінатора. Мінімізацію відстані Васерштейна математично можна зобразити наступним чином:

$$\min_G \max_C \mathbb{E}\left[\mathrm{C}(\boldsymbol{x}) - \mathrm{C}\big(\mathrm{G}(\boldsymbol{z})\big)\right], \tag{2.3}$$

де $\min_G$ являє собою мінімізацію втрат генератора, $\max_C$ представляє максимізацію втрат критика (або дискримінатора), $\mathbb{E}$ — очікування, представляє середнє значення за набором даних, $C(\boldsymbol{x})$ результат критика, якщо надані реальні дані, $C\big(G(\boldsymbol{z})\big)$ вихідні дані критика, коли надані штучні дані, згенеровані генератором.

Втрата Васерштейна використовується для оновлення ваг дискримінатора під час тренувального процесу та зарекомендувала себе як ефективна для стабілізації тренування ГЗМ.

До переваг цієї цільової функції відноситься більш стабільне навчання зі зменшеною ймовірністю колапсу моди, краща інформація про градієнт під час навчання. Щодо недоліків, функція втрат Васерштейна вимагає ретельного, зваженого «обрізання» ваг або градієнтного штрафу; повільніша конвергенція порівняно з деякими іншими ГЗМ.

ВГЗМ є модифікацією стандартної ГЗМ, де дискримінатор виводить число для кожного зразка, не класифікуючи його як справжній чи підроблений. Оскільки вихідне число не обов’язково має бути між 0 і 1, поріг 0,5 не можна використовувати для класифікації зразків. Натомість дискримінатор навчений давати вищі бали справжнім екземплярам, ніж фальшивим [59]. Саме тому класифікатор ВГЗМ називають «критиком», оскільки він не може відрізнити справжні зразки від підроблених.Метою дискримінатора є максимізація цієї функції, яка передбачає максимізацію різниці між його виходом на реальних зразках і його виходом на підроблених зразках [59].

Максимальна середня розбіжність (MCP). MCP — це функція втрат на основі відстані, яка вимірює різницю між розподілом реальних і синтетичних даних. MCP продемонструвала ефективність у стабілізації навчання ГЗМ і продукуванні високоякісних зразків [62]. Математично квадрат MCP-відстані між реальним розподілом $\mathbb{P}_r$ та згенерованим розподілом $\mathbb{P}_g$ для заданої ядрової функції k визначається так:

$$MMD^2(\mathbb{P}_r, \mathbb{P}_g) = \mathbb{E}_{x,x' \sim \mathbb{P}_r}[k(x, x')] + \mathbb{E}_{y,y' \sim \mathbb{P}_g}[k(y, y')] - 2\mathbb{E}_{x \sim \mathbb{P}_r, y \sim \mathbb{P}_g}[k(x, y)], \tag{2.4}$$

де x, x' — незалежні зразки з розподілу реальних даних $\mathbb{P}_r$; y, y' — незалежні зразки з розподілу згенерованих даних $\mathbb{P}_g$ (де $y = G(z)$, а z — вектор шуму); $k(\cdot,\cdot)$ — неперервна симетрична ядрова функція, наприклад, радіально-базисна функція, яка відображає дані у гільбертів простір із відтворюючим ядром; $\mathbb{E}$ — оператор математичного сподівання.

У ГЗМ генератор навчений мінімізувати MCP-відстань між згенерованим і реальним розподілом даних, що заохочує згенеровані дані відповідати реальному розподілу даних.

Втрата на основі методу найменших квадратів (Least Squares GAN, LSGAN). Мінімізує середню квадратичну помилку (MSE) між виходом дискримінатора та цільовими значеннями (0 для реальних зразків і 1 для синтезованих зразків). При правильній стратегії проєктування призводить до більш стабільного навчання та кращої якості зразків [60]:

$$\min_G, \min_D \mathbb{E}\left[(D(\boldsymbol{x}) - 1)^2\right] + E\left[D\left(G(\boldsymbol{z})\right)^2\right], \tag{2.5}$$

де $\min_G$ представляє мінімізацію втрат генератора, $\min_D$ представляє мінімізацію втрат дискримінатора, $\mathbb{E}$ очікування, середнє значення за набором

даних, $D(\boldsymbol{x})$ — вихідний результат дискримінатора, коли надано реальні дані, $D(G(\boldsymbol{z}))$ — вихідний результат дискримінатора, коли надано дані, згенеровані генератором, $(D(\boldsymbol{x})-1)^2$ — різниця в квадраті між виходом дискримінатора для реальних даних і 1 (цільове значення для реальних даних), $D\big(G(\boldsymbol{z})\big)^2$ — зведене в квадрат значення виходу дискримінатора для штучно синтезованих даних.

Це модифікована версія оригінальної ГЗМ, що використовує функцію втрат за методом найменших квадратів замість бінарної перехресної ентропії. Експериментально встановлено, що ГЗМ із такою функцією втрати створює зразки вищої якості та забезпечує більш стабільне навчання порівняно з оригінальною версією ГЗМ. До плюсів відноситься більш стабільне навчання з меншою кількістю зникаючих градієнтів, швидша конвергенція. До мінусів — чутливість до шуму в даних, може створювати зразки з низькою дисперсією, але нижчої якості.

Таку функцію втрати для ГЗМ обчислюємо наступним чином [60]:

$$\begin{aligned} \min_D V_{\mathrm{LSGAN}}(D) &= \tfrac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}a}(\boldsymbol{x})}[(D(\boldsymbol{x})-b)^2] + \tfrac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[(D(G(\boldsymbol{z}))-a)^2] \\ \min_G V_{\mathrm{LSGAN}}(G) &= \tfrac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[(D(G(\boldsymbol{z}))-c)^2] \end{aligned}, \tag{2.6}$$

де V представляє функцію значення, все інше є аналогічним до рівняння стандартної змагальної функції ГЗМ (рівняння 2.1) за винятком c, яке позначає значення, що G хоче щоб D «повірив» синтезованим даним.

Завісні втрати ГЗМ (Hinge Loss GAN). Така модифікація ГЗМ [111] використовує завісні втрати для ітераційного навчання генератора та дискримінатора. Завісні втрати «заохочують» дискримінатор мати запас принаймні «1» між балами, присвоєними реальним і згенерованим даним. Це,

як правило, призводить до більш стабільного навчання та швидшої конвергенції порівняно з деякими іншими варіантами ГЗМ. Для дискримінатора функція виглядає наступним чином:

$$L_D = -\mathbb{E}_{(x,y)\sim p_{data}}\big[\min\big(0,-1+D(x,y)\big)\big] - \mathbb{E}_{z\sim p_z, y\sim p_{data}}\big[\min\big(0,-1-D(G(z),y)\big)\big], \quad (2.7)$$

де L_D — втрата дискримінатора, $\mathbb{E}_{(x,y)\sim p_{data}}$ — оператор очікування, середнє значення за зразками даних (x,y), що отримані з розподілу даних p_{data}, $\min\big(0,-1+D(x,y)\big)$ обчислює мінімальне значення від 0 до $-1+D(x,y)$, $\mathbb{E}_{z\sim p_z, y\sim p_{data}}$ оператор очікування, що представляє середнє значення вибірок шуму z, взятих із розподілу шуму p_z, і вибірок даних y, отриманих із розподілу даних p_{data}, $\min\big(0,-1-D(G(z),y)\big)$ обчислює мінімальне значення між двома термами, це заохочує дискримінатор правильно класифікувати згенеровані дані як «реальні» з запасом принаймні «1».

Для генератора шарнірна функція втрати буде виглядати так [111]:

$$L_G = -\mathbb{E}_{z\sim p_z, y\sim p_{data}} D(G(z),y), \quad (2.8)$$

де L_G — втрата генератора, $\mathbb{E}_{z\sim p_z, y\sim p_{data}}$ — еквівалент такого ж терму в функції дискримінатора, $D(G(z),y)$ представляє вихідний результат дискримінатора, коли надаються дані синтезовані генератором штучних зразків.

У контексті ГЗМ завісні втрати використовуються для навчання дискримінатора, де y встановлюється рівним «1» для реальних зразків і «-1» для згенерованих зразків, а $f(x)$ є результатом дискримінатора щодо цього зразка.

Завісні втрати «штрафують» модель, якщо прогнозована оцінка для справжньої мітки менша за 1. Якщо оцінка більша або дорівнює 1, втрата дорівнює 0.

Кондиційна ГЗМ (Conditional GAN). Метою кондиційної цільової функції є мінімізація негативної логарифмічної ймовірності того, що дискримінатор присвоїть високу ймовірність реальним даним і низьку ймовірність згенерованим даним, коли це обумовлено міткою. По суті, це спонукає генератор створювати дані, які відповідають заданій умові. Математично, кондиційна функція ГЗМ може бути зображена наступним чином [101, 110]:

$$\min_G \max_D \mathbb{E}\left[\log\big(D(x|y)\big)\right] + \mathbb{E}\left[\log\Big(1 - D\big(G(z|y)\big)\Big)\right], \tag{2.9}$$

де решта термів залишається такою ж як і у рівняннях вище, але $D(x|y)$ представляє вихідні дані дискримінатора, коли надано реальні дані (x), обумовлені певною міткою (y), а $D\big(G(z|y)\big)$ — вихідні дані дискримінатора, коли надаються дані, згенеровані генератором $(G(z))$, обумовлені певною міткою (y).

Серед плюсів кондиційної ГЗМ слід відзначити можливість контролю створеного вмісту на основі заданих умов, що корисно для таких завдань, як трансляція із зображення в зображення та створення зразків із певних класів. Щодо недоліків, для такої архітектури потрібна додаткова умовна інформація, яка не завжди може бути доступною. Система також вкрай чутлива до якості кондиційних даних.

Існує багато інших варіацій цільових функцій, що застосовуються у генеративних змагальних мережах, але їх детальний аналіз виходить за межі даного дисертаційного дослідження.

ГЗМ супер роздільної здатності (SRGAN). Що стосується специфічної архітектури SRGAN, то у даному випадку використовується прикладна перцептивна втрата, яка є комбінацією втрат вмісту та конкурентної втрати. Рішення оцінюється на основі релевантних для сприйняття характеристик [57]:

$$l^{SR} = \underbrace{\underbrace{l_{\mathrm{X}}^{SR}}_{\text{втрата вмісту}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{змагальна втрата}}}_{\text{перцептивна втрата (для VGG)}}, \tag{2.10}$$

де l — це втрата, SR — супер роздільна здатність (англ. super resolution), l_{X}^{SR} — втрата вмісту, l_{Gen}^{SR} представляє змагальну втрату.

Що стосується змагальних втрат, в SRGAN до функції втрат додано генеративний компонент, який стимулює модель віддавати перевагу рішенням, які базуються на різноманітності природних образів, намагаючись ввести в оману дискримінатор. Генеративні втрати представлені наступним чином:

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -\log D_{\theta_D}\left(G_{\theta_G}(I^{LR})\right), \tag{2.11}$$

де l_{Gen}^{SR} представляє генеративну втрату, $D_{\theta_D}\left(G_{\theta_G}(I^{LR})\right)$ представляє ймовірність дискримінатора відносно того, що реконструйоване зображення є справжнім зображенням високої роздільної здатності, $G_{\theta_G}(I^{LR})$ — це реконструйоване зображення, I^{LR} — це низька роздільна здатність зображення.

Втрата VGG із шарами активації ReLU застосовується як функція втрат через її близькість до перцептивної подібності. Оскільки VGG19 [112] навчена на мільйонах зображень, виділення ознак такої попередньо навченої мережі забезпечує значні переваги кількісної оцінки з точки зору сприйняття, а також оцінює якість графічних зразків. Втрата вмісту розраховується наступним чином:

$$l_{VGG/i.j}^{SR} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left(\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}\left(G_{\theta_G}(I^{LR})\right)_{x,y}\right)^2, \tag{2.12}$$

де $G_{\theta_G}(I^{LR})$ означає репрезентацію реконструйованого зображення, I^{HR} є опорним зображенням, $W_{i,j}$ та $H_{i,j}$ представляють відповідні матриці карти функцій у мережі VGG.

Покращена ГЗМ супер роздільної здатності (ESRGAN). У даній архітектурі конвенційний дискримінатор замінено на релятивістський середній дискримінатор (англ. Relativistic average Discriminator (RaD)). Через що втрата дискримінатора (L_D^{Ra}) набуває наступного вигляду [58]:

$$L_D^{Ra} = -\mathbb{E}_{x_r}\left[\log\left(D_{Ra}\left(x_r, x_f\right)\right)\right] - \mathbb{E}_{x_f}\left[\log\left(1 - D_{Ra}\left(x_f, x_r\right)\right)\right], \quad (2.13)$$

де D_{Ra} — це релятивістський середній дискримінатор, $\mathbb{E}_{x_f}[\cdot]$ представляє процес отримання середнього значення для усіх згенерованих даних у міні-партії.

Змагальна втрата генератора представлена у дзеркальному вигляді:

$$L_G^{Ra} = -\mathbb{E}_{x_r}\left[\log\left(1 - D_{Ra}\left(x_r, x_f\right)\right)\right] - \mathbb{E}_{x_f}\left[\log\left(D_{Ra}\left(x_f, x_r\right)\right)\right], \quad (2.14)$$

де $x_f = G(x_i)$ та x_i позначають вхідне зображення з низькою роздільною здатністю.

Запропонована архітектура дозволяє генератору отримати переваги від градієнтів синтезованих даних і реальних даних у змагальному навчанні. У стандарті SRGAN використовується лише синтезована частина.

Що стосується перцептивної втрати, то повна втрата для генератора виглядає так:

$$L_G = L_{\text{percep}} + \lambda L_G^{Ra} + \eta L_1, \quad (2.15)$$

де $L_1 = \mathbb{E}_{x_i}\|G(x_i) - y\|_1$ позначає втрату вмісту; це оцінює відстань 1-норми від реконструйованого зображення $G(x_i)$ і фундаментальна істина (англ. ground truth) y; λ, η представляють коефіцієнти для балансування різноманітних умов збитків.

Важливо зауважити, що не існує універсального оптимального вибору функції втрат для певної архітектури ГЗМ. Вибір відповідної функції втрат для конкретного завдання повинен враховувати унікальні характеристики

проблеми, що вирішується. Отже функції втрат повинні обиратися в кожному випадку окремо.

Придатність функції втрат залежить від конкретних вимог завдання, набору даних і архітектури. Тому для досягнення оптимальної ефективності необхідний адаптивний підхід до вибору функції втрат. Це часто потребує використання експериментальних підходів для дослідження ефективності різних функцій втрат за різних умов.

Таблиця 2. Порівняльний аналіз ключових цільових функцій ГЗМ.

Цільова функція	**Переваги**	**Недоліки**	**Застосування**
BCE (Мінімакс)	Висока швидкість обчислень; простота реалізації.	Колапс моди; згасаючі градієнти.	Базові архітектури; прості датасети.
Васерштейна (WGAN)	Стабільність; плавні градієнти без згасання.	Ресурсоємність; складне налаштування ваг.	Складні багатовимірні розподіли.
Найменші квадрати (LSGAN)	Швидка збіжність; висока візуальна якість.	Висока чутливість до аномалій (викидів).	Завдання генерації реалістичних зображень.
Завісні втрати (Hinge)	Стійкість до викидів; обчислювальна ефективність.	Ризик осциляцій при дисбалансі мереж.	Масштабні моделі (наприклад, BigGAN).
MCP (MMD)	Незалежність від ідеального дискримінатора.	Квадратична складність обчислень $O(N^2)$.	Задачі з вимогою максимального різноманіття.

2.1.2 Збіжність

Конвергенція ГЗМ (збіжність) відноситься до процесу, коли мережі генератора та дискримінатора досягають стану рівноваги. У цьому стані генератор створює такі синтезовані дані, які мають настільки переконливі

властивості, що дискримінатор не може ефективно відрізнити їх від реальних даних.

Під час навчання генератор вчиться створювати дані, які поступово стають більш реалістичними, тоді як дискримінатор починає краще відрізняти реальне від підробленого. Це призводить до динамічної конкуренції, де обидві мережі вдосконалюються. Точка конвергенції досягається, коли жодна мережа не може значно перевершити іншу.

По суті, конвергенція ГЗМ схожа на роботу поліції, де генератор (фальшивомонетник) намагається створити синтезовані дані (підроблені гроші), які вводять в оману дискримінатор (поліцію), а дискримінатор вдосконалюється, щоб вловити помилки генератора (і спіймати фальшивомонетника). Коли вони збалансовуються, ГЗМ досягає конвергенції, і в результаті згенеровані дані виходять дуже схожі на реальні за спектром своїх властивостей.

За допомогою псевдокоду нижче можна проілюструвати базовий процес навчання конвенційної ГЗМ Ґудфеллоу [1]:

```
# Ініціалізація мережі генератора та дискримінатора.
θ_G, θ_D = initialize_G(), initialize_D()

# Цикл навчання.
for epoch in range(num_epochs):
    for x in dataset:
        # Оновлення дискримінатора.
        z = sample_noise()
        L_D = compute_L_D(θ_G, θ_D, x, z)
        θ_D = update_D(θ_D, L_D)

        # Оновлення генератора.
        z = sample_noise()
        L_G = compute_L_G(θ_G, θ_D, z)
        θ_G = update_G(θ_G, L_G)
```

Рисунок 2.2. Ілюстративний процес навчання ГЗМ, псевдо-код.

Ми можемо провести паралелі між конвергенцією ГЗМ і концепцією рівноваги Неша з теорії ігор. У рівновазі Неша це ситуація, коли жоден гравець не має стимулу змінювати свою стратегію, враховуючи стратегії інших гравців. Аналогічно, у конвергенції ГЗМ:

1. Генератор і дискримінатор схожі на двох гравців у грі.
2. Стратегія генератора полягає у створенні фейкових даних.
3. Стратегія дискримінатора полягає в тому, щоб відрізнити справжні дані від підроблених.

По мірі прогресії навчання:

- Генератор намагається вдосконалити свою стратегію, зробивши підроблені дані більш переконливими.
- Дискримінатор намагається вдосконалити свою стратегію, навчившись краще відрізняти справжні дані від підроблених.

Конвергенція ГЗМ відбувається, коли обидві мережі досягають точки, коли жодна не має сильного стимулу змінити свою стратегію. На цьому етапі:

- Генератор створює дані, які є настільки реалістичними, що дискримінатор не може достовірно відрізнити їх від реальних даних.
- Дискримінатор, навіть маючи найкращу стратегію, не може впевнено відрізнити справжні дані від підроблених.

Цей баланс подібний до рівноваги Неша, оскільки на цьому етапі ні генератор, ні дискримінатор не мають явної переваги над іншим. Вони обидва «сходяться» до точки, де їхні стратегії оптимізовані, і жодна не може в односторонньому порядку покращити свою продуктивність без відповідної адаптації іншої. Математично це означає, що часткові похідні відповідних функцій втрат щодо їхніх параметрів дорівнюють нулю:

1. Для генератора:

$$\frac{\partial L_G(\theta_G,\theta_D)}{\partial \theta_G} = 0, \quad (2.16)$$

2. Для дискримінатора:

$$\frac{\partial L_D(\theta_G,\theta_D)}{\partial \theta_D} = 0, \quad (2.17)$$

У точці рівноваги Неша генератор досягає оптимальної стратегії, за якої розподіл згенерованих даних p_g збігається з розподілом реальних даних p_{data}. Як наслідок, функція втрат дискримінатора стабілізується (але не мінімізується до нуля), оскільки він присвоює однакову ймовірність як реальним, так і фейковим зразкам ($D(x) \approx 0.5$), досягаючи максимуму ентропії. Фактично дискримінатор стає «випадковим вгадувачем» (англ. random guesser), тому що генератор працює ідеально.

Ця рівновага представляє стабільну точку в навчанні ГЗМ, де ні генератор, ні дискримінатор не мають переваги. Якщо один намагається змінити свою стратегію, інший адаптується, і вони залишаться в цьому збалансованому стані.

2.1.3. Методи досягнення конвергенції ГЗМ

Класичні методи

ГЗМ зазвичай навчаються за допомогою різних варіантів градієнтного спуску, таких як RMSprop [113], стандартний стохастичний градієнтний спуск (СГС) або Adam. Ці алгоритми оптимізації допомагають оновлювати параметри мережі (θ_G і θ_D) під час навчання. Використання добре налаштованого алгоритму оптимізації призводить до більш стабільної та швидшої конвергенції.

RMSprop адаптує швидкість навчання для кожного параметра на основі рухомого середнього квадратів градієнтів.

- α – швидкість навчання.
- ρ — швидкість спаду (зазвичай близька до 1, наприклад 0,9).

- $E[g^2]_t$ рухомим середнім квадратів градієнтів.

Правило оновлення для RMSprop має наступний вид [113]:

$$\begin{gathered} E[g^2]_t = \rho \cdot E[g^2]_{t-1} + (1-\rho) \cdot g_t^2 \\ \theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t \end{gathered}, \tag{2.18}$$

де $E[g^2]_t$ є експоненціально зваженим рухомим середнім квадратичного градієнта.

СГС — є найпростішим методом оптимізації, де швидкість навчання застосовується безпосередньо до градієнтів [114].

Правило оновлення для СГС виглядає так:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla J\left(\theta^{(t)}\right), \tag{2.19}$$

де $\theta^{(t)}$ представляє поточний набір параметрів на ітерації t (t — індекс ітерації), α – це швидкість навчання, $\alpha \nabla J\left(\theta^{(t)}\right)$ є градієнтом функції витрат $J(\theta)$ відносно параметрів θ у поточній точці $\theta^{(t)}$.

Adam поєднує в собі елементи техніки RMSprop і momentum [86]. Він адаптує швидкість навчання для кожного параметра на основі першого та другого моментів градієнтів. Правило оновлення для Adam має такий вид:

$$\begin{gathered} m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \\ v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \\ \widehat{m}_t = \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t = \frac{v_t}{1-\beta_2^t} \\ \theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t + \epsilon}} \cdot \widehat{m}_t \end{gathered}, \tag{2.20}$$

де m_t є першим моментом (середнім значенням) градієнтів, v_t є другим моментом (нецентрована дисперсія) градієнтів, β_1 та β_2 є експоненціальними темпами розпаду для моментів (як правило, близькі до 1, наприклад, 0,9 і 0,999), α – це швидкість навчання (крок), $\widehat{m}_t$ та $\hat{v}_t$ оцінки із поправкою на систематичну помилку, g_t — градієнт втрат стосовно параметрів на

тимчасовому кроці t, ϵ — невелика константа (наприклад, 10^{-8}) для запобігання ділення на нуль.

Інноваційні методи

Для досягнення конвергенції можуть бути застосовані також деякі неконвенційні методи, серед яких:

1) *Байєсівська оптимізація через динаміку Ланжевена*

Замість класичного градієнтного спуску, який прагне знайти точковий мінімум (point estimate), пропонується розглядати процес навчання як еволюцію розподілу ймовірностей параметрів у часі [115]. Для цього застосовується дискретизація стохастичного диференціального рівняння Ланжевена.

Цей метод додає контрольований шум до градієнтного кроку, що дозволяє моделі не «застрягати» в сідлових точках та локальних мінімумах, які є критичною проблемою для ГЗМ.

Математично оновлення параметрів θ на кроці t відбувається за правилом:

$$\theta_{t+1} = \theta_t - \frac{\eta_t}{2}\nabla_\theta L(\theta_t) + \sqrt{\eta_t T}\cdot \epsilon_t, \qquad (2.21)$$

де:

- η_t — крок навчання (learning rate), який може зменшуватися з часом;
- $\nabla_\theta L(\theta_t)$ — градієнт функції втрат (спрямовуюча сила);
- $\epsilon_t \sim \mathcal{N}(0, I)$ — вектори гаусівського шуму;
- T — параметр температури системи.

Терм $\sqrt{\eta_t T}\cdot \epsilon_t$ виконує роль теплового шуму в броунівському русі. При високих значеннях T (на ранніх етапах) шум домінує над градієнтом, забезпечуючи «exploration» — глобальне дослідження простору параметрів і вихід із неглибоких локальних мінімумів.

При $T \to 0$ (на пізніх етапах) система переходить у фазу «exploitation», точно збігаючись до глобального оптимуму, подібно до кристалізації речовини при охолодженні.

Для ГЗМ це критично важливо, оскільки дозволяє генератору уникати колапсу моди, «розмиваючи» розподіл параметрів.

2) *Стабілізація конвергенції за допомогою PID-регулювання*

Навчання ГЗМ можна розглядати як динамічну систему із зворотним зв'язком, де часто виникають небажані осциляції функції втрат. Класичні оптимізатори (SGD, Adam) працюють переважно як пропорційні контролери, реагуючи лише на поточну помилку. Для демпфування (гасіння) осциляцій та прискорення збіжності пропонується інтегрувати PID-контролер (Proportional-Integral-Derivative) безпосередньо в процес оновлення градієнтів.

Математично, цей метод буде виглядати наступним чином. Нехай V_t — це вектор оновлення ваг на кроці t. Модифікований закон оновлення [116] виглядає так:

$$V_t = K_P \cdot g_t + K_I \cdot \sum_{i=1}^{t} g_i + K_D \cdot (g_t - g_{t-1}),$$
$$\theta_{t+1} = \theta_t - \eta \cdot V_t, \qquad (2.22)$$

де $g_t = \nabla_\theta L(\theta_t)$ — поточний градієнт, а коефіцієнти відповідають за різні аспекти динаміки:

- K_P відповідає за стандартний градієнтний спуск. Реагує на поточну помилку (пропорційна складова).
- K_I накопичує історію градієнтів. Це дозволяє долати ділянки з малим градієнтом і усуває статичні помилки (інтегральна складова).
- K_D реагує на швидкість зміни градієнта (диференціальна складова). Це ключовий елемент для ГЗМ. Якщо градієнт різко змінюється (початок

осциляції), K_D створює протидію, «гальмуючи» різкі стрибки параметрів і стабілізуючи гру між генератором і дискримінатором.

Введення диференціальної компоненти (K_D) діє як «амортизатор» для навчального процесу, запобігаючи розбіжності траєкторій навчання, що є типовою проблемою при тренуванні змагальних мереж.

3) *Адаптивна нормалізація градієнтів*

Однією з ключових проблем навчання ГЗМ є нестабільність градієнтів, зокрема їх схильність до «вибуху» на початкових етапах або при дисбалансі між дискримінатором та генератором. Класична нормалізація до одиничної довжини (Unit Norm) вирішує проблему вибуху, але унеможливлює точну збіжність у фінальній фазі, оскільки фіксує крок навчання незалежно від близькості до локального мінімуму.

Для забезпечення балансу між стабільністю та точністю пропонується використовувати метод адаптивного відсікання градієнтів (Gradient Clipping) [117]. Суть методу полягає в обмеженні норми градієнта пороговим значенням C, зберігаючи при цьому його напрямок і природне затухання, коли норма стає меншою за поріг.

Модифікований градієнт $\hat{g}$ обчислюється за формулою:

$$\hat{g} = \frac{g}{\max\left(1, \frac{||g||}{C}\right)}, \tag{2.23}$$

де g — вихідний вектор градієнта $\nabla_\theta L(\theta_t)$, $||g||$ — евклідова норма вектору (L2-norm), а C — гіперпараметр порогу (clipping threshold). Далі оновлення параметрів відбувається стандартним чином — $\theta_{t+1} = \theta_t - \eta \cdot \hat{g}$. Ця функція діє як гібридний регулятор.

При $\| g \| > C$ (зона нестабільності) знаменник стає $\frac{\|g\|}{C}$, і ефективний градієнт масштабується до норми C. Це запобігає «вибуху» і розбіжності алгоритму.

При $\| g \| \leq C$ (зона збіжності) знаменник дорівнює 1, і $\hat{g} = g$. Це дозволяє градієнтам природним чином зменшуватися до нуля ($\hat{g} \to 0$), забезпечуючи точне потрапляння в локальний мінімум, чого не дозволяє жорстка нормалізація.

2.2. Дослідження, аналіз і варіанти розв'язання проблем, характерних для навчання ГЗМ

Проєктування та оптимізація ГЗМ може бути складним завданням через різноманітні технічні проблеми. У дисертаційному досліджені наведено комплексний аналіз методів вирішення цих проблем з точки зору оптимізації продуктивності ГЗМ. Дослідження охоплює низку компонентів дизайну, включаючи функції втрат, функції активації, нормалізацію партії (batch), обмеження ваги, градієнтний штраф, проблеми стабільності, оцінку продуктивності, міні-пакетну дискримінацію міні-серії та інші аспекти. У дослідженні також розглядаються різні методи, які використовуються для вирішення цих проблем.

2.2.2. Активаційні функції

У контексті проєктування та навчання ГЗМ, функції активації відіграють вирішальну роль і використовуються для введення нелінійності в мережі генератора та дискримінатора, що дозволяє моделі вивчати складні зв'язки між вхідними та вихідними даними. Вибір функції активації може мати значний вплив на продуктивність моделі, її точність і стабільність.

Сигмоїда — функція активації, яка відображає вхідні дані в діапазоні [0,1]. Визначається як:

$$\text{Sigmoid}\,(x) = \sigma(x) = \frac{1}{1+\exp\,(-x)}, \tag{2.24}$$

де x є вхідним значенням, σ є стандартним відхиленням, а exp є експоненціальною функцією; рисунок 2.3.

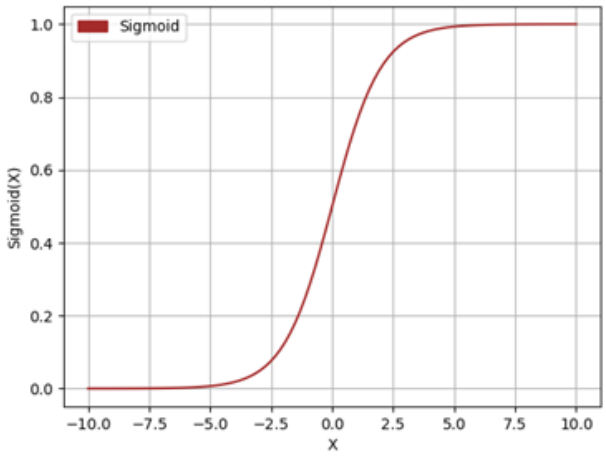

Рисунок 2.3. Функція логістичної активації.

Сігмоїда зазвичай використовується як функція активації вихідного шару генератора в ГЗМ, щоб гарантувати, що згенеровані вибірки знаходяться в бажаному діапазоні [63, 64].

ReLU (зрізаний лінійний вузол). ReLU є однією з найбільш широко використовуваних функцій активації в глибокому навчанні, включаючи ГЗМ [119].

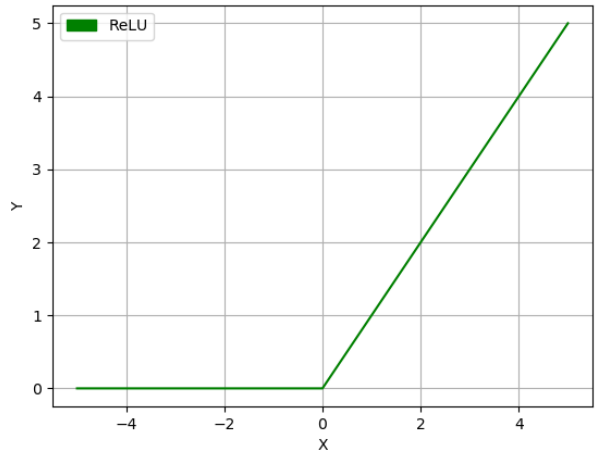

Рисунок 2.4. Зрізаний лінійний вузол або ReLU — функція нелінійної активації.

Це нелінійна функція активації, яку можна виразити за допомогою наступної нотації:

$$f(x) = (x)^+ = max(0, x), \tag{2.25}$$

де x це вхідне значення нейрона, max є виходом і позначає максимальну величину між нулем і вхідним значенням (рисунок 2.4).

ReLU є ефективною з точки зору обчислення і забезпечує швидшу конвергенцію під час навчання.

LeakyReLU. LeakyReLU — це варіант ReLU, який вирішує проблему «вмирання ReLU», коли градієнт ReLU стає нульовим для негативних вхідних даних, що призводить до припинення навчання відповідних нейронів [63, 64]. LeakyReLU (рисунок 2.5) визначається як:

$$\text{LReLU}\ (x) = max(0, x) + \ m * min(0, x), \tag{2.26}$$

де x — вхідне значення нейрона, max — максимальна величина між нулем і вхідним значенням, m — від'ємний нахил, min — мінімальне значення між нулем та x.

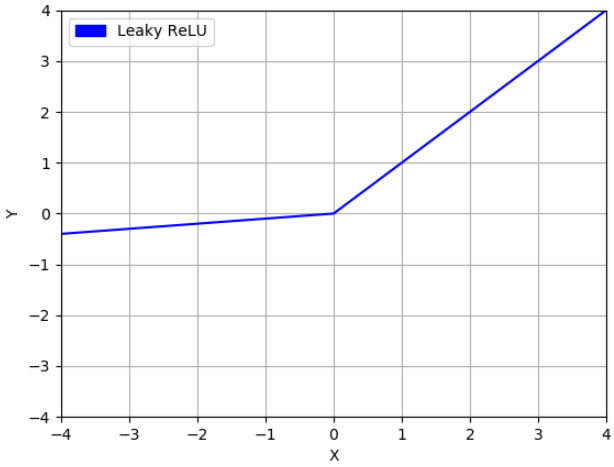

Рисунок 2.5. Негерметичний зрізаний лінійний вузол або LReLU — лінійний варіант ReLU.

LeakyReLU дозволяє деяким негативним значенням проходити через функцію активації, забезпечуючи більш стійкий градієнтний потік під час навчання [63].

Tanh (гіперболічний тангенс). Tanh — це функція активації, яка відображає вхідні дані в діапазон [-1,1]. Визначається як:

$$\text{Tanh}\,(x) = \tanh\,(x) = \frac{\exp\,(x) - \exp\,(-x)}{\exp\,(x) + \exp\,(-x)}, \tag{2.27}$$

де tanh — функція гіперболічного тангенса, x — результат генератора, а exp — експоненціальна функція.

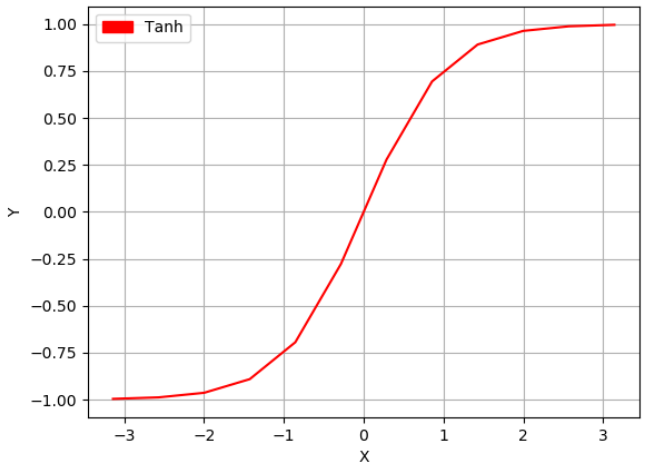

Рисунок 2.6. Гіперболічний тангенс або tanh — гіперболічна функція активації.

Функція гіперболічного тангенса (tanh) корисна в ГЗМ, оскільки вона генерує значення від -1 до 1, сумісні з тим самим діапазоном, що й нормалізовані реальні дані. Завдяки плавному градієнту вона допомагає стабілізувати процес навчання та пом'якшує проблему «вибухових» градієнтів, хоча все ще може спостерігатись зникаючий градієнт.

Крім того, tanh може допомогти мережі генератору виробляти ширший діапазон вихідних значень, роблячи його більш різноманітним і реалістичним [63, 64].

Softmax. Softmax — це функція активації, що використовується для задач багатокласової класифікації в ГЗМ [63]. Вона відображає результат моделі на розподіл ймовірностей для кількох класів. Softmax визначається як:

$$\text{Softmax}\,(x_i) = \frac{\exp(x_i)}{\sum_j \exp\,(x_j)}, \tag{2.28}$$

де x_i це вихід i-го нейрона у вихідному шарі і сума береться для всіх нейронів у вихідному шарі.

Функції активації, розглянуті вище, є критично важливим компонентом ГЗМ, який допомагає ввести нелінійність у моделі. Не існує єдиної «найкращої» функції активації, оскільки кожна функція має власний набір сильних і слабких сторін. Сигмоїда і tanh зазвичай використовуються в ГЗМ через їх здатність нормалізувати вихідні значення між -1 і 1, але вони страждають від проблеми зникнення градієнта. ReLU та LeakyReLU ефективні завдяки своїй простоті та здатності вирішувати проблему зникнення градієнта, але підходять не для всіх прикладних завдань. Softmax використовується в задачах мультикласової класифікації для створення розподілу ймовірностей, але вона не підходить для завдань регресії. Вибір функції активації залежить від конкретних вимог завдання та розуміння характеристик і компромісів кожної функції.

2.2.3. Пакетна нормалізація

Процес нормалізації вхідних даних для кожного шару нейронів називається пакетною нормалізацією і широко використовується в глибоких нейронних мережах. Цей процес підвищує стійкість мережі до змінних розподілів вхідних даних [65]. У ГЗМ пакетна нормалізація виконує роль інструмента для забезпечення стабільності навчання як генератора, так і дискримінатора [66].

У ГЗМ мережа генератор використовує вектор випадкового шуму для створення синтетичних зображень або зразків, тоді як мережа дискримінатор приймає зображення як вхідні дані та намагається відрізнити справжні зображення від підроблених. Навчання цих мереж відбувається періодично, коли генератор намагається згенерувати зображення, які можуть ввести в

оману дискримінатор, а дискримінатор прагне точно класифікувати справжні та підроблені зображення.

Пакетна нормалізація передбачає нормалізацію вхідних даних для кожного шару мережі на основі статистики міні-пакетів даних, що обробляються. Зокрема, пакетна нормалізація передбачає віднімання середнього значення та ділення на стандартне відхилення вхідних значень для кожного шару. Це має ефект центрування та масштабування даних, полегшуючи навчання мережі [63, 64, 67].

```
# Псевдо-код для пакетної нормалізації
def batch_normalization(x, gamma, beta, epsilon=1e-5):
    # Обчислення середнього та дисперсії
    mu = np.mean(x, axis=0)
    var = np.var(x, axis=0)

    # Нормалізація
    x_norm = (x - mu) / np.sqrt(var + epsilon)

    # Зміна масштабу та зсув
    out = gamma * x_norm + beta

    return out
```

Рисунок 2.7. Пакетна (batch) нормалізація, псевдо-код..

У ГЗМ пакетна нормалізація зазвичай застосовується до входів мережі дискримінатора, оскільки це може допомогти зменшити проблему колапсу моди. Завдяки нормалізації вхідних даних для дискримінатора, мережа з меншою ймовірністю буде схильна до перенавчання відносно конкретних характеристик зображення та зможе краще відрізняти реальні зображення від підроблених [68].

Пакетна нормалізація також може бути застосована до мережі генератора, але рідше, оскільки це може зробити мережу більш чутливою до невеликих змін у векторі вхідного шуму, що може призвести до нестабільності в процесі навчання. Однак деякі варіації ГЗМ, такі як кондиційні ГЗМ, можуть

покращити результати генерації у разі застосування пакетної нормалізації до мережі генератора.

2.2.4. Підрізка ваги та градієнтний штраф

Підрізка ваги та градієнтний штраф є двома додатковими методами, які використовуються при навчанні ГЗМ для покращення стабільності та стимулювання генерації зображень вищої якості [41].

Підрізка ваги відноситься до стратегії, яка використовується для запобігання надмірному зростанню ваг мережі дискримінатора під час процесу навчання. Ця техніка передбачає встановлення попередньо визначеного порогового значення та, після кожної ітерації навчання, обмеження вагових коефіцієнтів дискримінатора, щоб переконатися, що вони залишаються нижче цього порогового значення. Через обмеження вагових коефіцієнтів заздалегідь визначеним діапазоном підрізання вагових коефіцієнтів може пом’якшити проблему колапсу моди та завадити дискримінатору стати надмірно «впевненим» у своїх прогнозах.

Однак підрізка ваги має деякі недоліки. По-перше, це може призвести до втрати важливої дискримінаційної інформації, особливо якщо ваги підрізаються занадто агресивно. Крім того, обмеження ваги може призвести до нестабільності в процесі навчання, особливо в поєднанні з іншими методами, такими як пакетна нормалізація.

Хоча підрізка ваги може бути ефективною технікою для запобігання надмірного збільшення ваги дискримінатора під час навчання, вона має кілька потенційних недоліків [59, 69].

По-перше, обмеження ваги може призвести до зникнення або вибуху градієнта, особливо коли поріг встановлено занадто низьким або занадто високим відповідно. Це може перешкодити процесу навчання та негативно вплинути на продуктивність моделі.

По-друге, обмеження ваги може призвести до неплавної оптимізації, що може ускладнити пошук загального оптимуму під час навчання.

По-третє, обмеження ваги може бути важко встановити оптимально, і оптимальне значення може відрізнятися залежно від конкретного набору даних і архітектури моделі.

Підрізка ваги може зменшити виразну силу моделі та обмежити її здатність фіксувати складні патерни в даних.

Штрафуючи дискримінатор за створення градієнтів, які відхиляються від бажаних значень, штраф градієнта може допомогти зменшити проблему колапсу моди та покращити загальну якість згенерованих зразків. Крім того, градієнтне покарання є більш ефективним, ніж підрізка ваги, у покращенні стабільності та конвергенції процесу навчання ГЗМ [69].

Однак градієнтний штраф також має деякі обмеження. По-перше, обчислення може бути складним з точки зору обчислень, особливо для великих моделей ГЗМ з багатьма шарами. Крім того, ефективність градієнтного штрафу залежить від конкретного вибору функції «покарання» та заданих гіперпараметрів і може вимагати занадто ретельного налаштування для досягнення оптимальних результатів. Такий метод може призвести до зникнення або збільшення градієнтів, особливо якщо терм штрафу встановлено занадто високим. Це може перешкоджати процесу навчання та негативно вплинути на продуктивність моделі [69].

1-ліпшицеве відображення (1-L) — це властивість функцій, яка вимірює, наскільки їхній вихід може змінитися, коли вхід змінюється на незначну величину. У контексті ГЗМ неперервність 1-L часто використовується для забезпечення стабільності та конвергенції процесу навчання запобігаючи, щоб функція дискримінатора стала надто чутливою до невеликих змін у вхідних

даних. Дотримуючись цього обмеження, ГЗМ можуть створювати більш стабільні та якісніші згенеровані зразки [69].

Якщо дискримінатор ГЗМ надто чутливий до невеликих змін у вхідних даних, він може призначити дуже різні оцінки двом схожим зображенням (зразкам), навіть якщо вони обидва справжні. Це може призвести до нестабільності та низької продуктивності моделі.

Щоб вирішити цю проблему, деякі ГЗМ використовують градієнтний штраф, який забезпечує 1-L неперервність шляхом додавання терму регуляризації до функції втрат дискримінатора. Цей терм штрафує дискримінатор, якщо його градієнти щодо вхідних даних перевищують певний поріг, що допомагає гарантувати, що дискримінатор не буде надто чутливим до невеликих змін у вхідних даних [69].

Загалом забезпечення 1-L неперервності є важливим фактором під час проєктування та навчання ГЗМ, оскільки це може допомогти запобігти нестабільності та покращити якість згенерованих даних.

Підводячи підсумок, підрізка ваги та градієнтний штраф є двома додатковими методами, які зазвичай використовуються під час навчання ГЗМ для покращення стабільності та сприяння створенню зображень (або інших зразків) вищої якості. У той час як підрізка ваги може допомогти запобігти, щоб дискримінатор не став занадто «впевненим» у своїх прогнозах, градієнтний штраф може спонукати дискримінатор створювати більш реалістичні градієнти та покращити загальну якість створених зразків. Однак обидві техніки мають обмеження і повинні бути ретельно налаштовані для досягнення оптимальних результатів [69].

2.2.5. Проблема стабільності

Стабільність є основною проблемою при навчанні ГЗМ і пов’язана зі складністю підтримки балансу між мережами генератора та дискримінатора

під час навчання. Нестабільність може проявлятися кількома способами, такими як осцилюючі або вибухові функції втрат, зникнення градієнтів або колапс моди [66, 59].

Існує кілька методів, які можна використовувати для вирішення проблем стабільності в ГЗМ, включно з уже згаданими, такими як пакетна нормалізація, обмеження ваги та градієнтний штраф [70]. На додаток до цих, деякі інші широко використовувані методи підвищення стабільності в ГЗМ:

Одностороннє згладжування мітки. Цей процес передбачає згладжування міток для справжніх зображень, щоб вони були меншими за 1, а мітки для підроблених зображень — більшими за 0. Це заохочує дискримінатор бути менш певним у своїх прогнозах, що може допомогти запобігти надмірному домінуванню дискримінатора під час навчання [71].

Співставлення ознак. Співставлення ознак передбачає модифікацію функції втрат генератора, щоб спонукати згенеровані зображення відповідати проміжним ознакам, витягнутим дискримінатором, а не лише кінцевому виходу. Таким чином, відбувається заохочення генератора створювати більш різноманітні зображення, а дискримінатор має меншу ймовірність надмірного використання певних особливостей зображення.

Спектральна нормалізація. Спектральна нормалізація передбачає нормалізацію ваг дискримінатора на основі спектральної норми кожної вагової матриці. Це може допомогти зменшити константу Ліпшица дискримінатора та підвищити стабільність процесу навчання [72].

Регуляризація різноманітності. Регуляризація різноманітності передбачає модифікацію функції втрат генератора для заохочення генерації різноманітних зображень шляхом накладання на генератор штрафу за створення схожих зображень. Це може допомогти зменшити проблему колапсу моди та покращити загальну якість створених зображень [73].

Виключення або дропаут (англ. dropout). Виключення — це процес регуляризації, який використовується в мережі генераторі ГЗМ для запобігання перенавчанню та для покращення продуктивності узагальнення. Цей метод працює шляхом випадкової дезактивації частини нейронів під час кожної ітерації навчання, змушуючи генератор покладатися на різні підмножини нейронів для створення зразків. Застосування дропауту вимагає ретельного налаштування швидкості виключення та вибору шарів для виключення [74].

Ці підходи не є взаємовиключними і можуть бути поєднані залежно від конкретних потреб моделі ГЗМ.

2.2.6. Транспонована згортка

Незважаючи на розвиток нових цільових функцій, вагомим кроком у подоланні нестабільності навчання ГЗМ стала оптимізація самої топології цього класу нейронних мереж, зокрема поява архітектури глибокої згорткової ГЗМ (англ. DCGAN, Deep Convolutional GAN). Замість модифікації функції втрат, цей підхід стабілізував класичну мінімаксну гру за рахунок відмови від повнозв'язних шарів на користь згорткових, використання пакетної нормалізації та специфічних функцій активації (ReLU та LeakyReLU). Ключовим архітектурним елементом генератора глибокої згорткової ГЗМ, що дозволив синтезувати зображення високої роздільної здатності з латентного простору, стала операція транспонованої згортки.

Транспонована згортка (також відома як деконволюція або дробово-поступова згортка) є ключовою операцією у багатьох ГЗМ. Вона використовується в генераторній мережі для підвищення роздільної здатності відображень ознак низької роздільної здатності, які створюються початковими шарами мережі [75].

Операція транспонованої згортки по суті є зворотною до звичайної операції згортки. У звичайній операції згортки ядро застосовується до вхідного

зображення для створення набору вихідних відображень ознак. В операції транспонованої згортки ядро застосовується до вихідних відображень ознак, щоб створити набір відображень ознак із підвищеною дискретизацією.

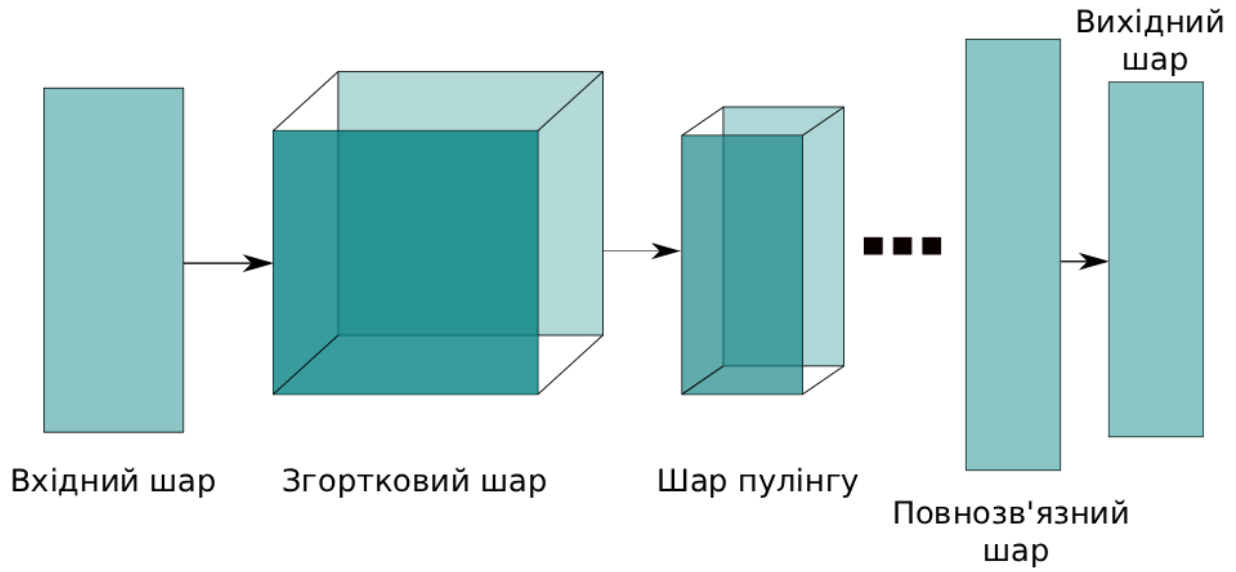

Рисунок 2.8. Архітектура згорткової нейронної мережі (CNN).

Математично операція транспонованої згортки в генераторі глибокої згорткової ГЗМ здійснюється через матричні перетворення. Якщо звичайну згортку можна подати як множення розрідженої матриці ваг C на вектор вхідних ознак X для отримання вихідного вектора Y меншої розмірності ($Y = CX$), то зворотний процес (підвищення дискретизації) реалізується множенням на транспоновану матрицю C^T:

$$Y' = C^T X', \tag{2.29}$$

де X' — вхідний вектор низької роздільної здатності (наприклад, з латентного простору), а Y' — згенерований вихідний вектор вищої роздільної здатності.

Для практичного проєктування архітектури генератора глибокої згорткової ГЗМ критично важливо контролювати розмірність тензорів на кожному шарі. Просторова розмірність вихідної карти ознак O (висота або ширина) при транспонованій згортці обчислюється за формулою:

$$O = (I - 1) \times S - 2P + K, \quad (2.30)$$

де I — розмірність вхідної карти ознак, S — крок згортки, P — розмір заповнення, а K — розмір ядра згортки.

Транспонована згортка зазвичай використовується в поєднанні з параметром кроку для підвищення дискретизації. Параметр кроку визначає відстань між вихідними пікселями, і якщо встановити для нього значення більше 1, операція ефективно виконує підвищення дискретизації (крок у 2 одиниці подвоїть роздільну здатність відображень ознак).

Одним з важливих моментів під час використання транспонованої згортки є вибір заповнення (англ. padding). У звичайній згортці вихідний розмір менший за вхідний, оскільки операція згортки «видаляє» деякі пікселі по краях. У транспонованій згортці вихідний розмір може бути більшим за вхідний, якщо застосовано відповідне заповнення. Вибір заповнення може вплинути на якість створених зображень і є важливим гіперпараметром, який слід враховувати при розробці моделі на основі ГЗМ.

Транспонована згортка дозволяє генератору створювати зображення високої роздільної здатності з вхідних даних із низькою роздільною здатністю. Використовуючи серію транспонованих шарів згортки для поступового збільшення роздільної здатності відображень ознак, генератор може створювати зображення, які є набагато більшими та складнішими, ніж початковий вхідний вектор шуму.

Хоча транспонована згортка є ефективною технікою, вона не позбавлена недоліків. Одна проблема полягає в тому, що вона може провокувати появу артефактів та шахових візерунків на отриманих зображеннях. Інша потенційна проблема полягає в тому, що навчання може бути складним, особливо при використанні великих кроків або глибоких нейронних мереж. Це пов’язано зі здатністю операції посилювати невеликі помилки у вхідних даних, що може

призвести до нестабільного навчання. Однак ці проблеми можна вирішити шляхом ретельного налаштування транспонованих параметрів згортки, таких як налаштування розмірів ядра та кроків, впровадження пропускних з'єднань або використання альтернативних методів підвищення дискретизації.

Підсумовуючи, транспонована згортка є цінним активом у наборі інструментів ГЗМ, оскільки вона дозволяє створювати високоякісні зображення з високою роздільною здатністю з низьковимірних вхідних векторів.

2.2.7. Прокляття розмірності

Термін «прокляття розмірності» застосовується відносно труднощів, які виникають під час обробки даних великої розмірності. У контексті ГЗМ це може становити значні проблеми як для генератора, так і для мережі дискримінатора [76].

Однією з головних проблем, пов'язаних із прокляттям розмірності, є розрідженість даних великої розмірності. Зі збільшенням розмірності вхідних даних експоненціально зростає кількість даних, необхідних для повного покриття вхідного простору. Це може ускладнити навчання ГЗМ на високорозмірних даних, оскільки обсяг навчальних даних, необхідних для досягнення високої продуктивності, може швидко стати непомірно великим.

Ще одна проблема, пов'язана з прокляттям розмірності, — це підвищена складність мереж генератора та дискримінатора. Зі збільшенням розмірності вхідних даних збільшується також кількість параметрів у мережах генератора та дискримінатора. Це може ускладнити навчання обох мереж, оскільки проблема оптимізації стає складнішою, а ризик перенавчання зростає.

Для подолання прокляття розмірності в ГЗМ використовують наступні методи: зменшення розмірності, регуляризація та трансферне навчання. Процес обмеження розмірності можна використовувати для зменшення

розмірності вхідних даних, що сприяє кращому навчанню ГЗМ. Методи регуляризації можуть бути використані для запобігання перенавчанню та покращення можливостей генералізації мереж. Трансферне навчання може бути застосовано, щоб отримати переваги від попередньо навчених моделей для схожих задач, що може зменшити кількість навчальних даних, необхідних для оптимальної продуктивності.

2.3. Особливості реалізації ГЗМ на кордонних пристроях

Сучасний вектор розвитку систем ШІ вимагає розгортання складних генеративних моделей безпосередньо на кордонних пристроях (IoT-сенсори, мобільні платформи, автономні системи комп'ютерного зору, вбудовані мікрокомп'ютерні системи контролю). Для прикладних задач, таких як локальний синтез зображень високої роздільної здатності, поліпшення якості вхідних візуальних даних або обробка зображень у режимі реального часу, автономне використання ГЗМ на edge-платформах набуває критичного значення.

Однак, перенесення класичних архітектур ГЗМ на такі пристрої стикається з жорсткими апаратними обмеженнями: гострим дефіцитом оперативної пам'яті, низькою обчислювальною потужністю процесорів та обмеженим енергоспоживанням. Як показав аналіз цільових функцій та методів стабілізації у попередніх підрозділах, розв'язання проблеми математичної збіжності зазвичай вимагає збільшення складності моделі (наприклад, обчислення додаткових градієнтних штрафів або використання ресурсномістких регуляризаторів). Це створює пряме протиріччя з ресурсними лімітами мікрокомп'ютерів. На сьогодні практика впровадження ГЗМ на таких пристроях зводиться до розрізнених, ситуативних рішень.

Зазначене протиріччя обґрунтовує необхідність створення уніфікованого підходу до проєктування ГЗМ для реалізації на кордонних пристроях та

мікрокомп’ютерних платформах. Такий підхід має стандартизувати процес розробки і поєднати обґрунтований вибір математично стабільної та обчислювально легкої цільової функції з обов’язковою інтеграцією алгоритмів апаратної адаптації. Практична реалізація вимагає застосування методів квантування ваг, структурного прунінгу та дистиляції знань. Застосування такого комплексного фреймворку дозволить суттєво зменшити розмір моделі та прискорити інференс генератора без критичної втрати якості синтезованих даних, забезпечуючи високопродуктивний інференс у режимі жорсткого реального часу.

2.4. Висновки за розділом 2

З врахуванням викладеного вище у даному розділі:

1) Проведено комплексний аналіз сучасних математичних моделей та алгоритмів навчання ГЗМ. Розглянуто цільові функції, принципи конвергенції і методи її досягнення за допомогою алгоритмів оптимізації на базі стохастичного градієнтного спуску.

2) Висвітлено критичні недоліки алгоритмічно-організаційних процедур оптимізації ГЗМ та існуючих архітектурних варіацій ГЗМ (нестабільність градієнтів, колапс моди, висока чутливість до гіперпараметрів тощо), а також визначено шляхи подолання зазначених недоліків і подальшого вдосконалення реалізованих на кордонних пристроях ГЗМ.

3) Запропоновано та обґрунтовано використання інноваційних методів досягнення конвергенції, таких як: байєсівська оптимізація через динаміку Ланжевена, стабілізація конвергенції за допомогою PID-регулювання та адаптивна нормалізація градієнтів.

4) Обґрунтовано необхідність уніфікованого підходу до проєктування ГЗМ для різнотипних прикладних застосувань при реалізації на кордонних пристроях та мікрокомп’ютерних платформах.

РОЗДІЛ 3. СТРАТЕГІЯ ТА МЕТОДИ ОПТИМІЗАЦІЇ ГЕНЕРАТИВНИХ ЗМАГАЛЬНИХ НЕЙРОННИХ МЕРЕЖ

3.1. Нестабільність навчання ГЗМ

Нестабільність ГЗМ є однією з основних проблем, що ускладнює їх ефективне навчання. Генератор прагне створювати дані, які максимально схожі на реальні, тоді як дискримінатор намагається відрізнити реальні дані від згенерованих. Однак така архітектура породжує кілька проблем, які можуть призводити до нестабільності, колапсу моди або надто швидкої конвергенції.

1. *Колапс моди* (англ. mode collapse) є поширеним явищем, коли генератор навчається відтворювати лише невелику підмножину можливих варіантів даних. Це означає, що замість генерування різноманітних зразків, генератор виробляє обмежену кількість майже ідентичних зображень, які дискримінатор вважає «правильними» (рисунок 3.1).

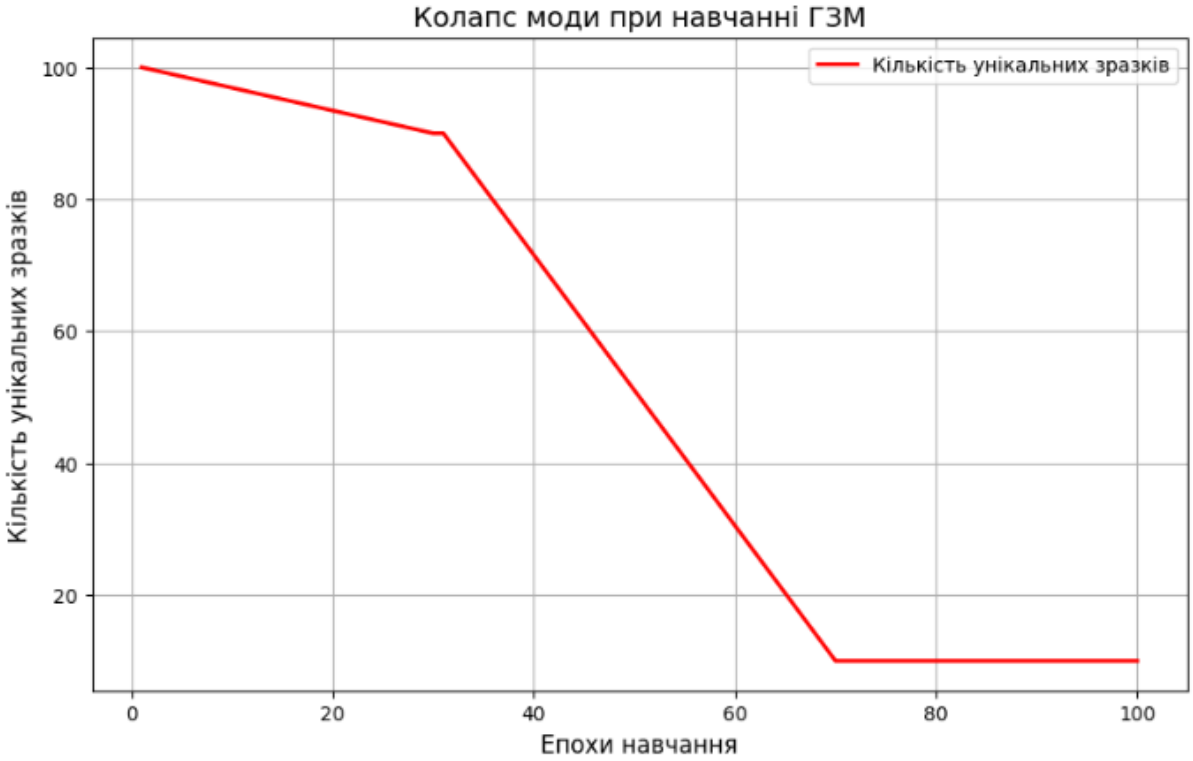

Рисунок 3.1. Колапс моди.

Причинами колапсу моди можуть бути:

- *Нестабільна взаємодія між генератором і дискримінатором*. Оскільки генератор і дискримінатор мають різні цілі, процес їхнього спільного навчання може призвести до ситуацій, коли генератор знає, як «переконати» дискримінатор, але при цьому виробляє обмежену кількість варіантів.
- *Невдалий вибір функції втрат*. Використання класичної функції втрат на основі логарифмічної правдоподібності часто призводить до того, що генератор мінімізує функцію втрат, не враховуючи різноманітність згенерованих даних.

2. *Надто швидка конвергенція* виникає, коли генератор або дискримінатор настільки добре навчаються, що процес навчання зупиняється передчасно.

На наведеному графіку (рисунок 3.2) продемонстрована надто швидка конвергенція моделей генератора та дискримінатора, що є однією з критичних проблем стабільності навчання ГЗМ. Синя крива відображає стрімке зниження втрат генератора майже до нуля, тоді як червона крива показує зростання втрат дискримінатора, що свідчить про виникнення суттєвого дисбалансу в їхній змагальній взаємодії.

Така динаміка вказує на ситуацію, коли одна з мереж (у даному випадку генератор) надто швидко «перемагає», через що процес навчання зупиняється передчасно, не дозволяючи моделі досягти якісного синтезу різноманітних зразків або належної узагальнюючої здатності.

Це призводить до ситуації, коли генератор не встигає навчитися створювати якісні зразки, оскільки дискримінатор вже може легко їх відрізнити.

Рисунок 3.2. Швидка конвергенція між генератором та дискримінатором.

Декілька чинників впливають на надто швидку конвергенцію:

- *Дисбаланс навчання*. Якщо дискримінатор навчається швидше, ніж генератор, то генератор може зупинитися на неефективному стані, оскільки дискримінатор надто добре виконує свою функцію. Це призводить до проблем з конвергенцією.
- *Нестійкі градієнти*. Через використання складних функцій активації та негладких функцій втрат градієнти можуть стати дуже великими або дуже малими, що ускладнює оптимізацію і може призвести до швидкої, але нестійкої конвергенції.

3.2. Оптимізація ГЗМ

Оптимізація — це процес налаштування параметрів моделей штучних нейронних мереж для досягнення кращої продуктивності, і який грає ключову

роль у забезпеченні стабільності навчання. Для різних типів ГЗМ завдання стабілізації цього процесу та пошуку оптимальних параметрів має свої особливості.

Класичні ГЗМ. У класичних ГЗМ використовується стохастичний градієнтний спуск (SGD) або його варіації. У таких системах основні проблеми виникають через занадто агресивну або надто консервативну оптимізацію, що призводить до колапсу моди або нестабільної конвергенції.

Стильові ГЗМ (англ. StyleGAN). У таких ГЗМ використовується специфічна архітектура, що включає адаптивну нормалізацію та стильові вектори для контролю за різними рівнями деталізації зображень. Така архітектура може бути менш вразливою до колапсу моди, але залишається чутливою до параметрів оптимізації, таких як вибір алгоритму оптимізації (наприклад, Adam) та налаштування швидкості навчання. Важливо забезпечити правильний баланс між стабільністю та різноманітністю згенерованих зразків.

Великі ГЗМ (англ. BigGAN). Цей підхід використовує більш масштабні мережі та збільшену потужність обчислень, що дозволяє досягти високої якості генерації зображень. Однак, масштабованість таких моделей підвищує вимоги до точності оптимізації, оскільки велика кількість параметрів може призвести до значної нестабільності в процесі навчання.

Сьогодні існує безліч модифікацій різних архітектур ГЗМ, але всі вони у тій чи іншій мірі залежать від обраних методів оптимізації, алгоритмів, методів, стратегій та окремих фреймворків, що впливають як на сам процес навчання ГЗМ, на їх продуктивність, так і на кінцевий результат згенерованих даних.

3.2.1. Важливість стратегії оптимізації в ГЗМ

Для досягнення високої якості та стабільності згенерованих ГЗМ результатів, необхідна ретельна оптимізація моделей та їх архітектур. Оптимізація грає ключову роль у навчанні цього типу штучних нейронних мереж, впливаючи на швидкість конвергенції, якість вихідних даних та стабільність навчального процесу. Без належної оптимізації моделі ГЗМ можуть страждати від нестабільності, низької якості результатів та інших проблем, що робить оптимізацію критичною для ефективності ГЗМ.

Основною метою цього розділу є дослідження та впровадження оптимізаційних методів для покращення продуктивності генеративних змагальних мереж.

Оптимізація є критично важливим аспектом під час навчання та *тонкого налаштування* ГЗМ. Успіх моделі значною мірою залежить від тонкого балансу між двома нейронними мережами: генератором і дискримінатором. Належна оптимізація гарантує, що ці мережі навчаються з правильною швидкістю, запобігаючи поширеним проблемам. Ефективні методи оптимізації, такі як адаптивний темп навчання та відсікання градієнта, можуть допомогти підтримувати цей баланс, дозволяючи ГЗМ створювати високоякісні та різноманітні зразки.

Тонке налаштування моделей ГЗМ також вимагає ретельної оптимізації для адаптації попередньо навчених мереж до нових наборів даних або конкретних завдань. Під час тонкого налаштування вкрай важливо відкоригувати темп навчання та застосувати методи регуляризації, щоб запобігти перенавчанню, особливо якщо новий набір даних невеликий або значно відрізняється від вихідних навчальних даних. Ефективні стратегії оптимізації, такі як передавальне навчання (transfer learning), можуть

використовувати попередньо підготовлені вагові коефіцієнти для прискорення процесу конвергенції та покращення продуктивності моделі для нового завдання. Ретельно керуючи процесом оптимізації, можна підвищити надійність і універсальність ГЗМ, зробивши модель більш ефективною в різних задачах, від синтезу зображень до покращення даних.

3.2.2. Оцінка продуктивності моделей ГЗМ в контексті їх оптимізації

Через відсутність універсальних критеріїв аналіз ефективності ГЗМ є комплексною задачею. В рамках дисертаційного дослідження застосовано комплексну методологію оцінки продуктивності моделей, що поєднує такі метрики, як відстань Фреше (FID) та індекс достовірності генерації (Inception Score, IS), які аналізують якість і різноманітність створених зображень. Додатково використано показники точність та повнота (Precision/Recall), а також перцептивна довжина шляху (Perceptual Path Length, PPL). Такий підхід дозволив об’єктивно оцінити перцептивну структуру латентного простору та продемонструвати переваги запропонованих моделей над базовими підходами. Розглянемо ці методи більш детально.

1. *Відстань Фреше на базі Inception (англ. Fréchet Inception Distance, FID)*. Відстань Фреше (ВФ) є загальновживаним показником для оцінки якості зображень, створених ГЗМ, шляхом порівняння розподілу створених зображень із розподілом реальних зображень, які використовуються для навчання моделі. ВФ обчислює квадрат відстані Васерштейна між двома багатофакторними Гауссами: один представляє розподіл характеристик, отриманих із згенерованих зображень за допомогою мережі Inception v3, а інший представляє розподіл характеристик із реальних зображень. Нижча Відстань Фреше вказує на те, що згенеровані зображення більш схожі на

справжні зображення. Відстань Фреше є покращенням порівняно з індексом достовірності генерації, оскільки Відстань Фреше безпосередньо порівнює згенеровані зображення з реальними зображеннями, а не просто оцінює різноманітність згенерованих зразків [78, 79,80].

Графік на рисунку 3.3 демонструє гауссівські розподіли для реальних (синій) і згенерованих (жовтий) зображень. Середні значення кожного розподілу позначені крапками, а еліпси представляють коваріаційні матриці (розмах та орієнтацію). Відстань Фреше враховує як різницю в середніх значеннях, так і форму (коваріацію) розподілів. Цей графік ілюструє, як ВФ оцінює подібність між двома розподілами.

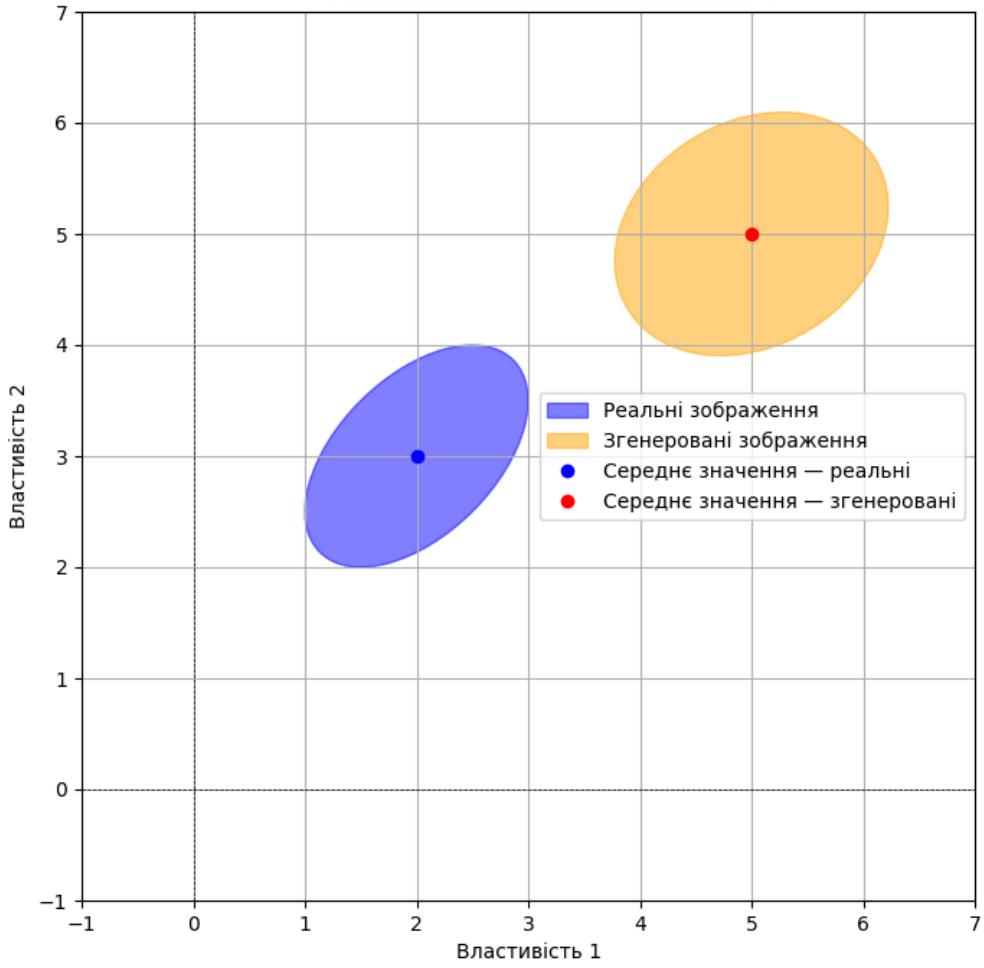

Рисунок 3.3. Візуалізація відстані Фреше між розподілами реальних та згенерованих зображень.

Математично процес можна зобразити через набори ознак, витягнуті з реальних і згенерованих зображень відповідно за допомогою мережі Inception v3 — X_r та X_g. Обидва набори припускаються такими, що підкоряються багатовимірному нормальному розподілу. На прикладі відношення $X_r \sim N(\mu_r, \Sigma_r)$, де $\sim N$ — це нормальний розподіл, μ_r — вектор середніх значень реальних зображень, Σ_r — коваріаційна матриця реальних зображень. У формулі $X_g \sim N(\mu_g, \Sigma_g)$ іншим є лише нижній індекс g, який позначає згенеровані зображення. Отже, Відстань Фреше між двома багатовимірними нормальними розподілами виражається наступним чином [79, 80]:

$$FID = \left|\left|\mu_r - \mu_g\right|\right|^2 + Tr\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r\Sigma_g\right)^{\frac{1}{2}}\right), \quad (3.1)$$

де Tr представляє слід матриці (сума її діагональних елементів), $\left|\left|\mu_r - \mu_g\right|\right|^2$ — це відстань між середніми векторами реальних та згенерованих зображень, а $Tr\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r\Sigma_g\right)^{\frac{1}{2}}\right)$ — це міра різниці в коваріаціях обох розподілів.

2. *Індекс достовірності генерації (ІДГ, англ. Inception Score, IS)*. ІДГ — ще один часто використовуваний показник для оцінки якості зображень, створених ГЗМ. Він вимірює різноманітність і якість створених зображень, обчислюючи ентропію прогнозованих міток класу для кожного зображення, а потім обчислює експоненту середньої ентропії [81]. Математично ІДГ можна зобразити наступним чином:

$$IS = \exp\left(\mathbb{E}_x\left[\, D_{KL}\left(p(y|x)||p(y)\right)\right]\right), \quad (3.2)$$

де $p(y|x)$ — це розподіл ймовірностей класів для згенерованого зображення x, отриманий від моделі Inception, $p(y)$ — це середній розподіл класів для всіх згенерованих зображень, тобто показник різноманітності, $D_{KL}\left(p(y|x)||p(y)\right)$ — це відстань Кульбака-Лейблера (КЛ-дивергенція), яка вимірює відмінність

між розподілом класів для конкретного зображення і середнім розподілом класів. Чим більша ця розбіжність між розподілами, тим вищою вважається чіткість та різноманітність згенерованих зображень.

ІДГ не враховує подібність між згенерованими та реальними зображеннями, на відміну від відстані Фреше. Він лише оцінює різноманітність і чіткість згенерованих зображень. Іноді високий ІДГ може бути досягнутий, навіть якщо зображення далекі від реальних, але різноманітні.

4. *Точність та повнота (англ. precision and recall)*. Точність та повнота — це показники, які зазвичай використовуються в завданнях класифікації, але їх також можна використовувати для оцінки продуктивності ГЗМ. Точність вимірює частку згенерованих зображень, які класифікуються як реальні, тоді як повнота вимірює частку реальних зображень, які класифікуються як такі [82].

Точність (precision) можна визначити як частку згенерованих зображень, які правильно класифікуються як реальні:

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{3.3}$$

де *TP* — це кількість згенерованих зображень, які класифікуються як реальні, *FP* — кількість згенерованих зображень, які класифікуються як реальні, але насправді не є такими.

Повнота (recall) вимірює, скільки реальних зображень вдалося охопити:

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{3.4}$$

де *FN* — кількість реальних зображень, які були класифіковані як нереальні.

4. *Перцептивна довжина шляху (ПДШ)*. Перцептивна довжина шляху — це метрика для оцінки різноманітності зображень, створених ГЗМ. Вона вимірює середню відстань у просторі ознак між парами зображень, які

знаходяться поруч одне з одним у латентному просторі. Низькі показники ПДШ вказують на більшу різноманітність і вищу якість створених зображень [83, 84]:

$$L_{perceptual} = \mathbb{E}_{z_1,z_2}\left[\frac{\|f(G(z_1))-f(G(z_2))\|_2}{\|z_1-z_2\|_2}\right], \quad (3.5)$$

де $f\big(G(z)\big)$ — це представлення згенерованого зображення в просторі ознак, $\| f(G(z_1)) - f(G(z_2)) \|_2$ — це евклідова відстань між зображеннями в просторі ознак, $\| z_1 - z_2 \|_2$ — це відстань між двома точками в латентному просторі (z_1 та z_2 — дві точки в латентному просторі, які знаходяться на відстані δ_z одна від одної), $\mathbb{E}$ — математичне сподівання, тобто середнє значення для всіх пар точок, $G(z)$ — функція генератора, яка перетворює точку в латентному просторі z на згенероване зображення.

Оцінка продуктивності є важливим аспектом будь-якої моделі машинного навчання, і ГЗМ не є винятком. У контексті ГЗМ оцінка є особливо складною через відсутність чіткої об’єктивної функції та суб’єктивний характер якості створених зразків. Розглянуті вище метрики оцінювання призначені для забезпечення кількісного вимірювання продуктивності ГЗМ і є певним дороговказом для подальшої розробки та вдосконалення архітектур ГЗМ. Оскільки цей тип нейронних мереж продовжує розвиватися та знаходить нові сфери застосування, розробка більш ефективних і надійних методів оцінки продуктивності залишатиметься активною областю досліджень.

3.3. Деякі оптимізаційно-орієнтовані методи покращення продуктивності ГЗМ

Окремим класом методів, які продемонстрували здатність покращувати продуктивність і точність експериментальних моделей, спроєктованих для цілей і задач цього дисертаційного дослідження, є оптимізаційно-орієнтовані методи. В рамках дослідження розроблено нові методи цього класу та

запропоновано їх гібридне комбінування, що дало змогу отримати якісні і стабільні результати. З метою обґрунтування запропонованих рішень доцільно спершу розглянути базові оптимізаційно-орієнтовані методи, що склали теоретичне та методологічне підґрунтя дослідження. Розглянемо деякі з них.

3.3.1. Одностороннє згладжування міток

Одностороннє згладжування міток — це техніка, яка використовується для покращення здатності ГЗМ до узагальнення. Метод передбачає модифікацію міток, які використовуються для мережі дискримінатора під час навчання, шляхом заміни двійкових міток 0 і 1 згладженими мітками 0 і α, де α є значенням між 0 і 1.

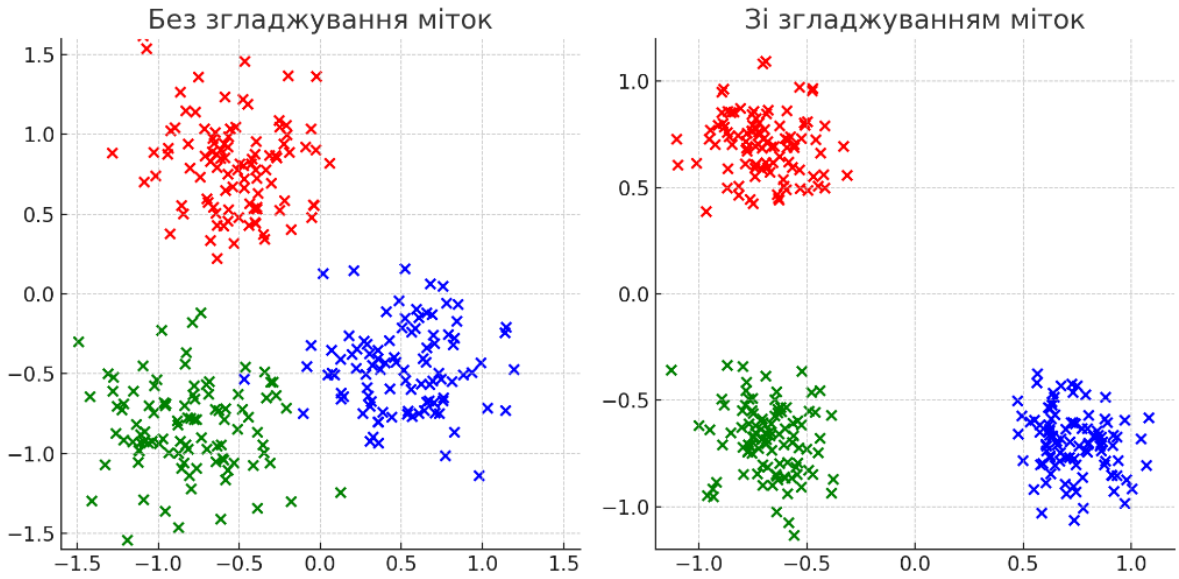

Рисунок 3.4. Порівняння розподілу даних у просторі ознак: ліворуч — без використання згладжування міток, праворуч — із застосуванням згладжування (спостерігається чіткіша кластеризація).

Метою згладжування міток є запобігання тому, щоб дискримінатор став надто «впевненим» у своїх передбаченнях, що могло б призвести до перенавчання та поганих показників узагальнення. Згладжуючи мітки, дискримінатор заохочується бути менш певним у своїх прогнозах, що може допомогти покращити узагальнюючу продуктивність моделі ГЗМ.

Цей процес корисний не тільки для дискримінатора ГЗМ, але й для інших завдань бінарної класифікації із залученням нейронних мереж. Часто спостерігається, що класифікатори дають правильні результати, але з надто впевненими ймовірностями. Цю надмірну впевненість можна пом'якшити шляхом застосування одностороннього згладжування міток. Ідея полягає в тому, щоб замінити цільове значення для реальних прикладів на дещо менше за 1, наприклад, 0,9, що допомагає згладити розподіл виходу дискримінатора.

Якщо позначити позитивні цільові мітки як α, а негативні як β, то оптимальний дискримінатор D стає:

$$D(x) = \frac{\alpha p_{\text{data}}(x) + \beta p_{\text{model}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}, \tag{3.6}$$

де $p_{\text{data}}(x)$ представляє початковий розподіл даних (або, принаймні, наше наближення до нього), $p_{\text{model}}(x)$ позначає розподіл, якого ми прагнемо досягти.

Коли $\beta \neq 0$, згенеровані зразки з p_{model} не мають стимулу наближатися до істинного розподілу даних; натомість це підсилює поточну поведінку генератора. Тому ми згладжуємо тільки позитивні мітки до α, залишаючи негативні мітки на рівні 0. Такий підхід запобігає тому, щоб дискримінатор подавав генератору надто великі сигнали градієнта.

Одностороннє згладжування міток називається «одностороннім», оскільки згладжується лише позитивна мітка (тобто мітка для реальних даних), тоді як негативна мітка (тобто мітка для згенерованих даних) залишається незмінною. Це пов'язано з тим, що згладжування негативної мітки може призвести до нестабільності навчання та погіршення продуктивності [66, 72].

3.3.2. Міні-пакетна дискримінація

Міні-пакетна дискримінація — це метод ГЗМ, який покращує різноманітність і якість створених зображень. Він інтегрує додатковий рівень

до мережі дискримінатора, який обчислює міру відстані між характеристиками згенерованих зображень та характеристиками інших зображень у межах однієї міні-партії (пакета) [66].

Мета міні-пакетної дискримінації полягає в тому, щоб заохотити мережу генератор створювати більш різноманітні зображення, штрафуючи її за створення зображень, які надто схожі одне на одне. Це може допомогти запобігти колапсу моди. Показник відстані, що використовується в розрізненні міні-пакетів, може мати різні форми, але зазвичай підхід полягає в обчисленні відстані L1 або L2 між характеристиками згенерованих зображень і характеристиками інших зображень у тій самій міні-партії. Отримані значення відстані потім конкатенуються з характеристиками згенерованих зображень і передаються через мережу дискримінатор. Відстані L1 і L2 — це два типи показників відстані, які використовуються в лінійній алгебрі, теорії матриць та машинному навчанні для вимірювання різниці чи подібності між двома векторами. Відстань L1, також відома як Манхеттенська відстань, є сумою абсолютних різниць між відповідними елементами двох векторів. Відстань L2, також відома як евклідова відстань, є квадратним коренем із суми квадратів різниць між відповідними елементами двох векторів.

Загалом, міні-пакетна дискримінація є простою та ефективною методикою для покращення різноманітності та якості зображень, створених ГЗМ, заохочуючи мережу генератор створювати більш різноманітні зображення та запобігаючи колапсу моди. Це особливо корисно для створення зображень з високою роздільною здатністю або для завдань, де цільовий розподіл має багато мод [66].

3.3.3. Відбір зразків та метод усічення

Метод усічення (або техніка усічення) використовується для балансування між різноманітністю та якістю зображень, згенерованих ГЗМ.

Метод передбачає модифікацію процесу вибірки з латентного простору генератора шляхом відсікання значень, що виходять за межі певного діапазону (або перцентиля). Зазвичай це дозволяє значно підвищити якість та реалістичність згенерованих зразків шляхом уникнення зон низької щільності ймовірності, хоча і може призводити до деякого зниження варіативності (різноманітності) вихідних даних [85].

3.4. Алгоритми для реалізації стратегії оптимізації

Деякі з базових алгоритмів оптимізації згадувались у попередньому розділі, але доцільно розглянути найбільш популярні з них у контексті оптимізації ГЗМ та їх недоліків. Щоб підвищити продуктивність моделей ГЗМ, були досліджені алгоритми оптимізації на основі градієнтного спуску, такі як стохастичний градієнтний спуск, RMSProp, AdaGrad, Momentum, Adadelta, Adagrad та Adam [86].

Стохастичний градієнтний спуск (СГС) — це простий і широко використовуваний алгоритм оптимізації для нейронних мереж, який оновлює параметри моделі шляхом обчислення градієнта функції втрат відносно параметрів і коригування їх у напрямку, протилежному градієнту. Це ефективний алгоритм, але при використанні в ГЗМ він може страждати від нестабільності через високий ступінь коливань градієнтів. Це може призводити до того, що моделі затримуються в локальних мінімумах або навіть до повної неуспішності навчання.

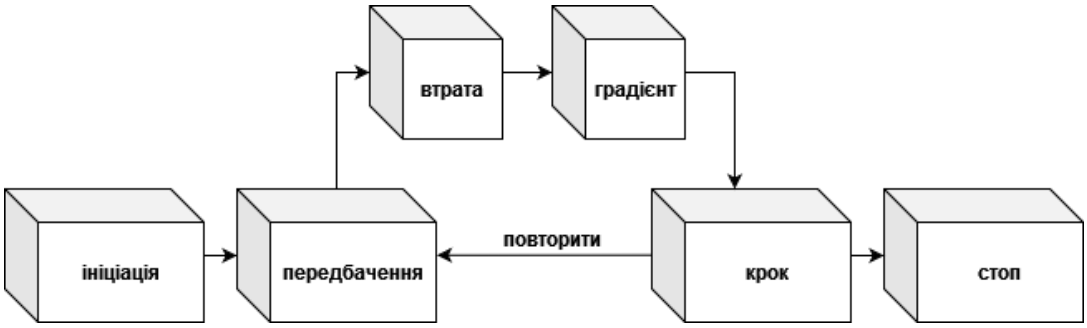

Рисунок 3.5. Процес градієнтного спуску.

У ГЗМ класичний градієнтний спуск зазвичай не рекомендується використовувати без модифікацій, але може бути корисним у випадках, коли дані дуже великі і потрібно просте та швидке обчислення градієнтів.

RMSProp — це алгоритм оптимізації на основі градієнта, який адаптує темп навчання для кожного параметра на основі експоненційного ковзного середнього квадратичних градієнтів, що зменшує вплив шумних градієнтів. Добре підходить для моделей зі швидкими змінами градієнтів, але може призводити до того, що модель застрягає на плато і перестає поліпшуватися. Не гарантує стабільності ГЗМ, що може бути проблемою. Доцільно використовувати, коли модель потребує стабільнішого навчання в порівнянні з *Adam*, особливо в умовах коливальних градієнтів.

AdaGrad адаптує темп навчання для кожного параметра індивідуально, забезпечуючи більші кроки оновлення для рідкісних ознак та менші — для тих, що зустрічаються часто.

AdaGrad має тенденцію до зниження швидкості навчання з часом, що може бути невигідно для ГЗМ, де важливо швидко реагувати на зміни в навчанні дискримінатора і генератора. Доцільно використовувати у випадках, коли дані дуже розріджені, і важливо забезпечити невеликі, але стабільні оновлення ваг.

Імпульс (англ. momentum) використовує ковзне середнє попередніх градієнтів для прискорення СГС, що допомагає уникнути локальних мінімумів і швидше збігатися.

Adadelta є розширенням алгоритму AdaGrad, яке адаптує темп навчання на основі ковзного середнього градієнтів і оновлень, що додатково зменшує вплив шумних градієнтів.Може бути занадто консервативним, що призводить до повільного навчання. У ГЗМ це може бути проблемою, оскільки швидка реакція на зміни у функції втрат є критичною. Доцільно використовувати, коли

потрібно уникнути дуже великої амплітуди градієнтів, і важливо зберегти контроль над процесом навчання.

Adam — популярний варіант для ГЗМ, але він схильний до колапсу моди, коли генератор починає створювати дуже схожі зображення, які дискримінатор легко розпізнає. Висока швидкість навчання може призводити до швидкого, але неякісного зближення. Adam адаптує темп навчання для кожного параметра на основі першого та другого моментів градієнтів, що призводить до більш надійної конвергенції та покращеної продуктивності порівняно з іншими алгоритмами оптимізації. Серед цих алгоритмів Adam продемонстрував, що може досягати кращої продуктивності порівняно з RMSProp, незалежно від налаштувань гіперпараметрів, що було апробовано експериментально на логістичній регресії та багатошарових і згорткових нейронних мережах [86]. Adam доцільно використовувати, коли потрібна швидка конвергенція, і коли модель має достатню кількість параметрів, щоб дозволити врахування градієнтів із зменшеними коливаннями.

Надам (англ. Nadam or Nesterov-accelerated Adam) — є модифікацією алгоритму Adam, що поєднує адаптивний темп навчання з прискореним градієнтом Нестерова. Попри вищу швидкість збіжності, надмірне коригування кроку в цьому методі може спричинити вихід за межі оптимальних значень градієнтів, що іноді призводить до зниження точності та стабільності процесу навчання. Його доцільно використовувати коли потрібно прискорити процес тренування та уникнути коливань, але важливо слідкувати за тим, щоб не перевищити доцільну швидкість.

У таблиці нижче наведене порівняння ефективності різних алгоритмів оптимізації в різних умовах. Тут враховані такі параметри, як стабільність навчання, ризик колапсу моди, швидкість конвергенції, та необхідність налаштування гіперпараметрів.

Таблиця 3. Порівняння алгоритмів оптимізації [77].

Алгоритм	Стабільність тренування	Колапс моди	Швидкість конвергенції	Налаштування гіперпараметрів
SGD	Низька	Середній	Низька/Середня	Прості
Adam	Середня	Високий	Висока	Високі
RMSprop	Середня	Середній	Середня	Середні
Nadam	Середня	Високий	Висока	Високі
Adagrad	Висока	Низький	Низька	Прості
Momentum	Середня	Середній	Висока	Середні
Adadelta	Висока	Низький	Низька	Прості

Залежно від конкретного завдання, обраний алгоритм оптимізації може суттєво впливати на результат. Для більшості ГЗМ моделей, що працюють зі складними зображеннями, Adam залишається раціональним вибором через свою швидкість, але може потребувати додаткових заходів для уникнення колапсу моди. RMSprop і Adagrad можуть бути ефективними варіантами для стабільності, тоді як SGD рідко використовується через високу нестабільність.

Оптимізація ГЗМ все ще залишається активною сферою досліджень, з постійними зусиллями, спрямованими на розробку математичних основ і поліпшення програмного та апаратного забезпечення.

3.5. Каскадний метод оптимізації ГЗМ

У дослідженні запропоновано новий метод каскадної оптимізації для навчання ГЗМ, який підвищує ефективність моделі шляхом декомпозиції процесу навчання на окремі етапи. Кожен етап спрямований на вирішення специфічних завдань генерації та класифікації, що забезпечує стабільну збіжність моделі (рисунок 3.6).

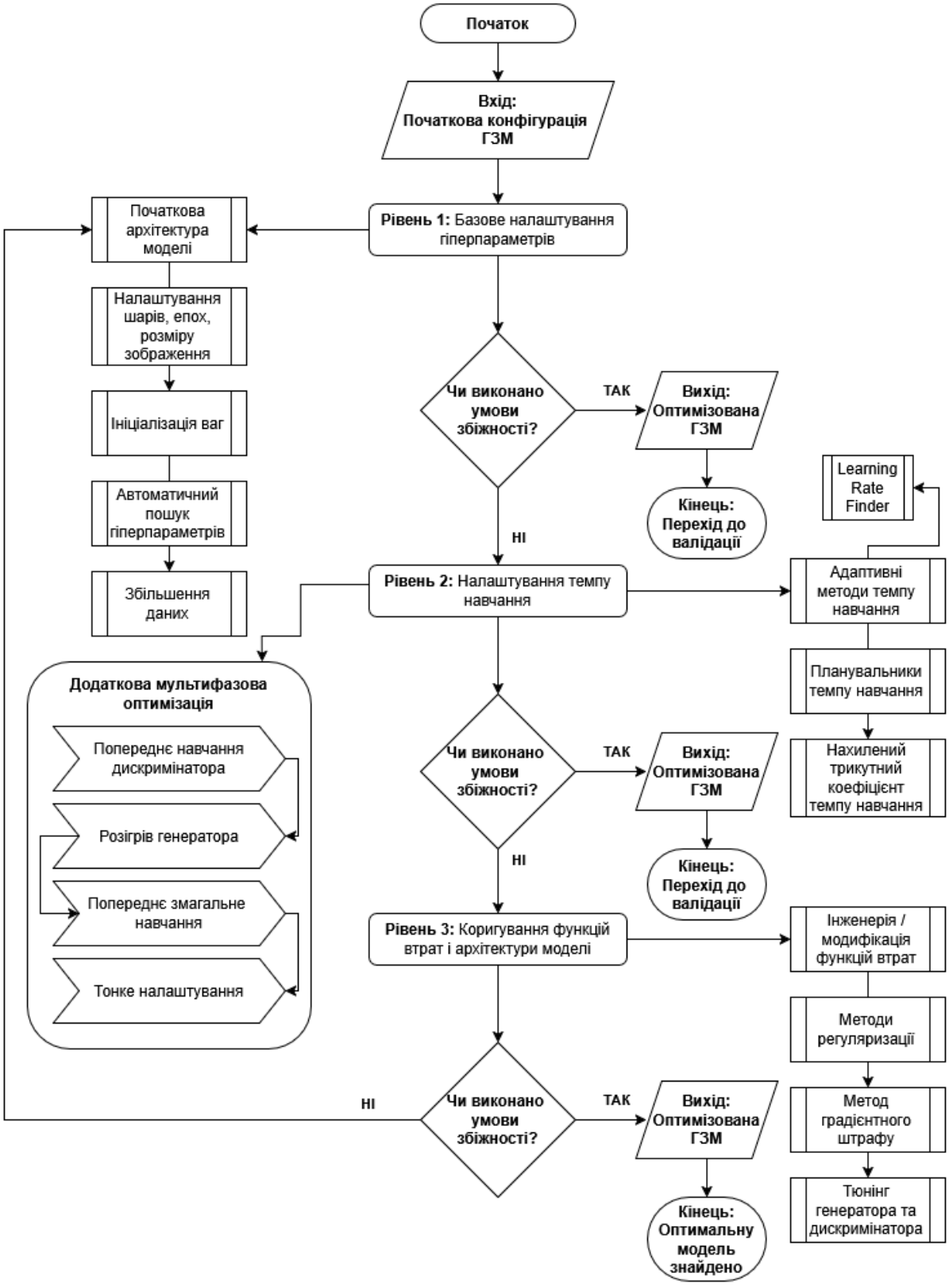

Рисунок 3.6. Блок-схема каскадного методу оптимізації.

Каскадний метод оптимізації є багаторівневим фреймворком, розробленим для підвищення стабільності навчання ГЗМ. Він складається з трьох основних рівнів оптимізації, які застосовуються послідовно (англ. sequential refinement strategy).

- Грубе налаштування (англ. coarse tuning) — швидкий пошук субоптимальних параметрів.
- Точне (тонке) налаштування (англ. fine-tuning) — мінімізація ризику потрапляння в локальні мінімуми та покращення генералізації.
- Реконфігурація (англ. architecture/loss re-engineering) — структурна зміна моделі у випадку недостатньої ефективності попередніх етапів.

3.5.1 Математична модель каскадного методу

Для забезпечення строгості запропонованого каскадного методу та можливості його подальшої алгоритмічної реалізації, розглянемо його математичну формалізацію. Задачу ефективного навчання ГЗМ в умовах обмежених апаратно-параметричних ресурсів доцільно сформулювати як процес багатоетапної, умовної оптимізації, що базується на ієрархічній декомпозиції простору пошуку гіперпараметрів.

Процес структурно-параметричної оптимізації генеративних змагальних мереж вимагає чіткої специфікації вхідних даних та граничних умов. Введемо початковий контекст задачі $\mathcal{C}_{\text{task}}$, що подається на вхід алгоритму:

$$\mathcal{C}_{\text{task}} = \langle \mathcal{D}_{\text{raw}}, \mathrm{R}_{\max}, \Omega_{\mathcal{A}}, \epsilon \rangle, \tag{3.7}$$

де $\mathcal{D}_{\text{raw}}$ — цільовий набір навчальних даних, $\mathrm{R}_{\max}$ — вектор апаратно-параметричних обмежень цільового обчислювального вузла (зокрема, ліміти доступної пам’яті та обчислювальної складності); $\Omega_{\mathcal{A}}$ — допустимий підпростір пошуку легковагових архітектур; ϵ — цільовий поріг якості

генерації (порогове значення обраної метрики), що визначає критерій ранньої зупинки (англ. early stopping).

З урахуванням заданого контексту $\mathcal{C}_{\text{task}}$, задачу каскадної оптимізації формалізуємо як процес багатоетапного пошуку. Базову модель ГЗМ визначаємо як кортеж:

$$M = \langle G(\mathbf{z}; \theta_G), D(\mathbf{x}; \theta_D), \mathcal{L}, \mathcal{H}, \mathcal{A} \rangle, \tag{3.8}$$

де G та D — генератор та дискримінатор, $\mathbf{z}$ представляє вектор латентного простору (шум), θ_G — вектор параметрів генератора, $\mathbf{x}$ — вектор (тензор) вхідних даних, θ_D — вектор параметрів дискримінатора, $\mathcal{L}$ — функціонал втрат, $\mathcal{H}$ — простір гіперпараметрів, $\mathcal{A}$ — архітектурна топологія ($\mathcal{A} \in \Omega_{\mathcal{A}}$).

Задачу каскадної оптимізації формуємо як пошук фінальної конфігурації оптимізованої моделі — M^*, що мінімізує цільову метрику розбіжності J при виконанні обмежень на ресурси R_{const} (ліміти пам'яті, обчислювальна складність):

$$M^* = \arg\min_{M \in \Omega} \{ J(M, \mathcal{D}_{\text{raw}}) \mid \text{Cost}(M) \leq \text{R}_{\max} \}, \tag{3.9}$$

де у $\text{Cost}(M) \leq \text{R}_{\max}$ нерівність виконується поелементно.

Запропонований метод реалізує глобальний оператор оптимізації Φ як композицію трьох вкладених відображень, що діють у просторах зі змінною розмірністю на макрорівні m:

$$\Phi^{(m)} = \Phi_3 \circ \Phi_2 \circ \Phi^{(m)}{}_1, \tag{3.10}$$

Перший рівень (базова ініціалізація Φ_1) формалізуємо як конвеєр послідовних операторів, що формують початковий простір станів:

$$\Phi_1 = \varphi_{\text{aug}} \circ \varphi_{\text{hpo}} \circ \varphi_{\text{init}} \circ \varphi_{\text{config}} \circ \varphi_{\text{arch}}, \tag{3.11}$$

де φ_{arch} — вибір початкової архітектурної топології; φ_{config} — налаштування структури (шари, епохи); φ_{init} — ініціалізація вагових коефіцієнтів; φ_{hpo} —

оператор оптимізації гіперпараметрів; φ_{aug} — оператор збільшення (аугментації) даних.

Другий рівень (адаптивна динаміка Φ_2) інтегрує модуль $\eta_{\text{opt}} = \mathcal{F}_{LRF}(\mathcal{D}_{raw}, M)$ (англ. Learning Rate Finder) для налаштування темпу навчання $\eta(t)$, де $t, t+1$ — дискретні індекси ітерацій внутрішнього циклу. Зокрема, застосовується нахилений трикутний коефіцієнт (англ. slanted triangular learning rate), що математично виражається як функція з фазою швидкого лінійного зростання та тривалого експоненційного (або лінійного) спаду.

На цьому ж рівні внутрішній цикл оновлення ваг модифікуємо через мультифазову стратегію Ψ_{multi}. Алгоритм не є лінійним, а містить ітеративний альтернуючий цикл. Після попереднього навчання дискримінатора $\mathcal{T}_{\text{pre(D)}}$, ініціюємо цикл «розігріву» та змагального навчання:

$$\theta^{(k)} = \mathcal{T}_{\text{adv}}(\mathcal{T}_{\text{warm}(G)}(\theta^{(k-1)}), \text{для } k = 1 \dots K, \tag{3.12}$$

де $\theta^{(k)}$ — оновлений стан ваг після чергового застосування зв'язки «розігрів + змагання», $\mathcal{T}_{\text{adv}}$ — оператор змагального навчання, $\mathcal{T}_{\text{warm}(G)}$ — оператор «розігріву» генератора. Використовуємо модель у стані $k-1$, спочатку «розігріваємо» її, а потім проводимо змагальне навчання і отримуємо новий проміжний стан $\theta^{(k)}$. Лише після стабілізації цього циклу застосовується оператор тонкого налаштування $\theta_{\text{final}} = \mathcal{T}_{\text{fine}}\left(\theta^{(K)}\right)$.

Третій рівень (структурна корекція Φ_3) активується для модифікації ландшафту функції втрат у разі відсутності глобальної збіжності. Оскільки ГЗМ є мінімаксною грою двох акторів, результуючий функціонал втрат розділяємо на систему незалежних рівнянь для дискримінатора та генератора:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}_{D\text{-base}} + \lambda_{\text{GP}}(\| \nabla_{\mathbf{x}} D(\mathbf{x}) \|_2 - 1)^2 + \lambda_{\text{reg}} \| \theta_D \|_2^2 \\ \mathcal{L}_G = \mathcal{L}_{G\text{-base}} + \lambda_{\text{reg}} \| \theta_G \|_2^2 \end{cases}, \tag{3.13}$$

де $\mathcal{L}_D$ та $\mathcal{L}_G$ — результуючі (модифіковані) функції втрат дискримінатора та генератора відповідно, $\mathcal{L}_{D\text{-base}}$ та $\mathcal{L}_{G\text{-base}}$ — їхні базові функції втрат, $\lambda_{\text{GP}}(\| \nabla_{\mathbf{x}} D(\mathbf{x}) \|_2 - 1)^2$ — градієнтний штраф для забезпечення 1-L, λ_{GP} — коефіцієнт інтенсивності штрафу (скаляр), $\lambda_{\text{reg}} \| \theta_D \|_2^2$ та $\lambda_{\text{reg}} \| \theta_G \|_2^2$ — L2-регуляризація відповідних мереж (англ. weight decay).

Окрім безпосередньої модифікації цільових функцій, на цьому рівні додатково може вводитися параметр алгоритмічного балансування — коефіцієнт n_{critic}, який визначає співвідношення кількості кроків оновлення ваг дискримінатора до генератора.

Розглянемо як досягаються глобальні умови збіжності. Перехід між етапами регламентується критерієм ранньої зупинки. Якщо на етапах $k \in \{1,2\}$ досягається поріг збіжності $J\big(\Phi_k(M)\big) \leq \epsilon$, каскад переривається і система переходить до стадії фінальної валідації. Якщо після виконання рівня 3 система збігається, фіксується успішне завершення навчання та оптимізації. У випадку недосягнення збіжності на Рівні 3, ініціюється глобальний цикл зворотного зв'язку $\Phi_3 \to \Phi^{(m+1)}{}_1$ (до оператора φ_{arch}) для модифікації початкової архітектури.

3.5.2 Перший рівень — коригування базових гіперпараметрів

На першому рівні оптимізації проєктується попередня архітектура моделі, а саме розробка мереж генератора (G) і дискримінатора (D) на базі обраної архітектури (стандартної або, наприклад, згорткової).

Далі здійснюється коригування базових гіперпараметрів моделі, таких як: визначення оптимальної кількості шарів для генератора і дискримінатора, визначення оптимальної кількості епох для навчання моделі, вибір оптимального розміру вхідних та вихідних зображень. Також відбувається ініціалізація ваг за допомогою таких методів, як ініціалізація Xavier або He.

Ці параметри мають значний вплив на загальну продуктивність і стабільність моделі. Вибір правильних значень для цих параметрів може значно покращити якість генерованих зображень.

Далі використовуються методи автоматизованого пошуку гіперпараметрів, таких як Grid Search, Random Search або Bayesian Optimization, для визначення оптимальних значень базових гіперпараметрів.

Також, за наявності достатніх обчислювальних ресурсів, може застосовуватися підхід нейронного архітектурного пошуку (NAS) для автоматизованого проєктування архітектури нейронних мереж. Враховуючи високу обчислювальну складність класичних методів NAS, у рамках даного дослідження доцільно використовувати оптимізовані алгоритми (наприклад, диференційований пошук архітектури або пошук у фіксованому просторі блоків), що дозволяє знайти баланс між максимізацією ефективності моделі та апаратними обмеженнями.

На цьому ж оптимізаційному рівні застосовуються методи збільшення навчальних даних, таких як перевертання, обертання, масштабування та модифікація колірних каналів, щоб покращити здатність дискримінатора до узагальнення.

Деталізуючи вплив коригування базових гіперпараметрів, слід зазначити, що налаштування таких параметрів, як кількість шарів у генераторі та дискримінаторі, значно впливає на здатність моделі вивчати складні патерни в даних. Наприклад, занадто мала кількість шарів може призвести до недосягнення необхідної комплексності моделі, що ускладнює генерацію реалістичних зразків, тоді як надмірна кількість шарів може спричинити перенавчання або надмірну складність, що уповільнює процес навчання.

Вибір методів автоматизованого пошуку, таких як Grid Search та Random Search, обґрунтований необхідністю дослідження великого простору

гіперпараметрів з мінімальними затратами часу. Баєсова оптимізація додатково дозволяє враховувати попередню інформацію про простір гіперпараметрів, що пришвидшує процес оптимізації. Використання NAS (Neural Architecture Search) є обґрунтованим, коли йдеться про автоматичне проєктування складних архітектур, особливо для задач, де продуктивність моделі є критичним фактором.

Математично перший рівень можна проілюструвати наступним чином — на першому етапі оптимізуємо кількість шарів для генератора G і дискримінатора D, ініціалізуємо ваги:

$$W_G, W_D \sim \mathcal{N}(0, \sigma^2), \tag{3.14}$$

де W_G — це ваги генератора, W_D — ваги дискримінатора, σ^2 — дисперсія при ініціалізації ваг.

Використовуємо пошук оптимальних гіперпараметрів:

$$\theta = \arg\min_{\theta} \mathcal{L}(\theta), \tag{3.15}$$

де θ — набір гіперпараметрів моделі, які оптимізуємо (наприклад, кількість шарів у генераторі та дискримінаторі, розміри міні-пакетів, темп навчання тощо).

Якщо продуктивність незадовільна за двома критеріями:

$$\text{FID} \leq \text{FID}_{\text{threshold}}, \tag{3.16}$$

$$| \mathcal{L}_{\text{train}}(t) - \mathcal{L}_{\text{val}}(t) | \leq \epsilon, \tag{3.17}$$

де, FID — відстань Фреше (FID/ВФ), $\text{FID}_{\text{threshold}}$ — встановлений поріг для ПВФ (якщо значення ВФ менше або дорівнює цьому порогу, вважається, що модель генерує реалістичні зображення), $L_{\text{train}}(t)$ — функція втрат на навчальних даних на кроці t, $L_{\text{val}}(t)$ — функція втрат на валідаційних даних на кроці t, ϵ — допустиме відхилення між втратами на навчальних і валідаційних наборах (якщо різниця між цими втратами менша або дорівнює

ϵ, то модель вважається стабільною, і можна припинити подальшу оптимізацію), — тоді переходимо до наступного рівня оптимізації.

Другий критерій визначається нерівністю $| \mathcal{L}_{\text{train}}(t) - \mathcal{L}_{\text{val}}(t) | \leq \epsilon$. У контексті навчання змагальних мереж, де динаміка функцій втрат має осцилюючий характер, цей вираз слугує індикатором відсутності перенавчання дискримінатора (англ. generalization gap). Ця умова розглядається у комплексі з першим критерієм (FID): якщо значення FID задовольняє порогову умову (генерація якісна), а різниця втрат не перевищує ϵ (навчання стабільне і модель не “запам’ятовує” вибірку), процес оптимізації на поточному рівні вважається успішно завершеним. У якості порогового значення FID тут і надалі встановлено загальноприйняте значення FID = 20.

3.5.3 Другий рівень — налаштування темпу навчання

Якщо перший рівень каскаду не забезпечує належний рівень продуктивності моделі, застосовується наступний рівень.

Другий рівень оптимізації зосереджується на налаштуванні темпу навчання (англ. learning rate). Це критичний параметр, який впливає на швидкість та стабільність процесу навчання. На цьому етапі застосовуються наступні методи.

Адаптивні методи налаштування коефіцієнту швидкості (темпу) навчання, такі як Adam, RMSprop або Nadam, які автоматично коригують швидкість навчання під час тренування. Також застосовуються варіації типів коефіцієнтів швидкості навчання з елементами *нечіткої логіки* з використанням адаптивних методів, що враховують поточний стан навчання і відповідним чином коригують швидкість навчання.

Планувальники зміни темпу навчання (наприклад, Cosine Annealing, Step Decay або Exponential Decay), які дозволяють динамічно змінювати швидкість навчання на різних етапах тренування.

Нахилений трикутний коефіцієнт темпу навчання — метод, що використовує динамічну зміну швидкості навчання протягом навчання, що дозволяє досягти кращої конвергенції.

Метод нахиленого трикутного коефіцієнта швидкості навчання (Slanted Triangular Learning Rates) є ефективним підходом для динамічної адаптації темпу навчання під час тренування нейронних мереж, включаючи ГЗМ. Цей метод передбачає варіювання темпу навчання протягом тренувального процесу у формі нахиленого трикутника, де початковий темп навчання поступово збільшується до піку, після чого зменшується до кінцевого значення.

Головна ідея полягає у тому, щоб спочатку дозволити моделі швидко навчатися, ефективно переміщуючись у просторі параметрів, а потім поступово зменшувати темп, що дозволяє більш точно наблизитись до оптимальних значень параметрів. Це допомагає уникнути як перенавчання, так і застрягання у локальних мінімумах.

Підхід нахиленого трикутного коефіцієнта є особливо корисним для ГЗМ, оскільки він дозволяє збалансувати навчання генератора і дискримінатора. Це зменшує ризик колапсу моди та інших проблем, пов’язаних із надто швидким чи повільним навчанням. Важливо зазначити, що цей метод можна комбінувати з іншими підходами, такими як адаптивні методи налаштування темпу навчання, наприклад Adam або RMSprop, що забезпечує додаткову стабільність і покращує якість згенерованих зображень.

Застосування цього методу дозволяє підвищити стабільність і продуктивність моделі, забезпечуючи кращі результати на етапах як попереднього, так і основного тренування ГЗМ.

Нижче наведено математичне зображення нахилених трикутних коефіцієнтів навчання, які спочатку лінійно збільшують коефіцієнт навчання,

а потім лінійно зменшують його відповідно до наступного графіку оновлення [87]:

$$\tau_c = \lfloor T \cdot \phi \rfloor, \tag{3.18}$$

$$p = \begin{cases} \frac{t}{\tau_c}, & \text{при } t < \tau_c \\ 1 - \frac{t-\tau_c}{T-\tau_c}, & \text{інакше} \end{cases}, \tag{3.19}$$

$$\eta_t = \eta_{\max} \cdot \frac{1+p\cdot(\rho-1)}{\rho}, \tag{3.20}$$

де T — це число ітерацій навчання; ϕ — це гіперпараметр фази (частка ітерацій), на яку лінійно збільшується темп навчання; τ_c — це ітерація, коли відбувається перехід від збільшення до зменшення темпу навчання; p — це функція прогресу, яка визначає нормовану частку ітерацій для відповідного збільшення або зменшення темпу навчання; ρ — це коефіцієнт співвідношення, який визначає, у скільки разів найменший (базовий) темп навчання менший від максимального η_{max} (максимальне значення швидкості навчання); η_t — швидкість навчання на ітерації t.

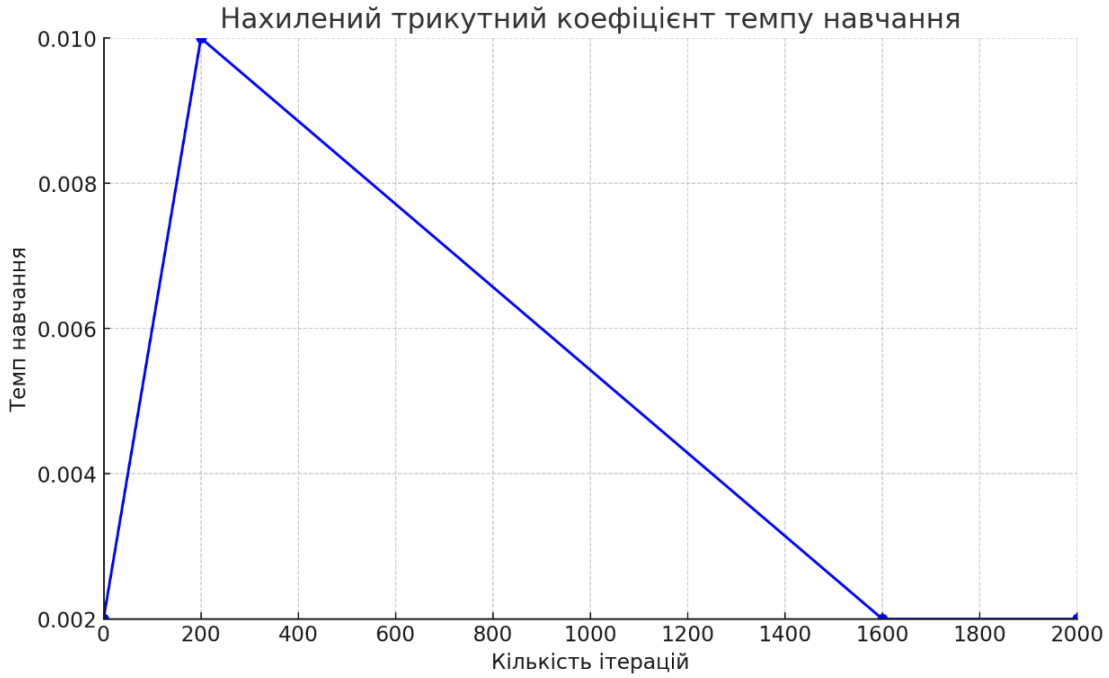

Рисунок 3.7. Нахилений трикутний коефіцієнт темпу навчання.

Для більш точного налаштування темпу навчання важливим інструментом є метод *Learning Rate Finder*, запропонований Леслі Смітом [88]. Цей метод передбачає короткий навчальний цикл, під час якого темп навчання поступово збільшується експоненційно від дуже малого значення до великого. Протягом циклу навчання здійснюється моніторинг функції втрат на кожній ітерації. Це дозволяє побудувати графік залежності динаміки втрат від темпу навчання. На такому графіку виокремлюються три ключові зони:

- Зона низького темпу навчання характеризується майже статичними значеннями функції втрат. Це свідчить про низьку швидкість збіжності або повну відсутність навчання через недостатню величину кроку оновлення ваг.
- Зона оптимального темпу навчання охоплює діапазон значень, де спостерігається інтенсивне зниження функції втрат. Даний інтервал є найбільш ефективним для швидкої та стабільної оптимізації моделі. Важливо зазначити, що межі цієї зони не завжди є чітко визначеними.
- Зона надмірного темпу навчання визначається різким зростанням або нестабільністю функції втрат. У цій області виникають значні коливання (осциляції), що призводить до порушення процесу конвергенції та деградації результатів.

Метод Learning Rate Finder дозволяє точно визначити оптимальний діапазон темпу навчання для моделі, що значно покращує ефективність навчання та допомагає уникнути поширених проблем, таких як надто повільна або надто швидка конвергенція. Цей метод став особливо популярним у практичних застосуваннях глибокого навчання завдяки своїй простоті та ефективності.

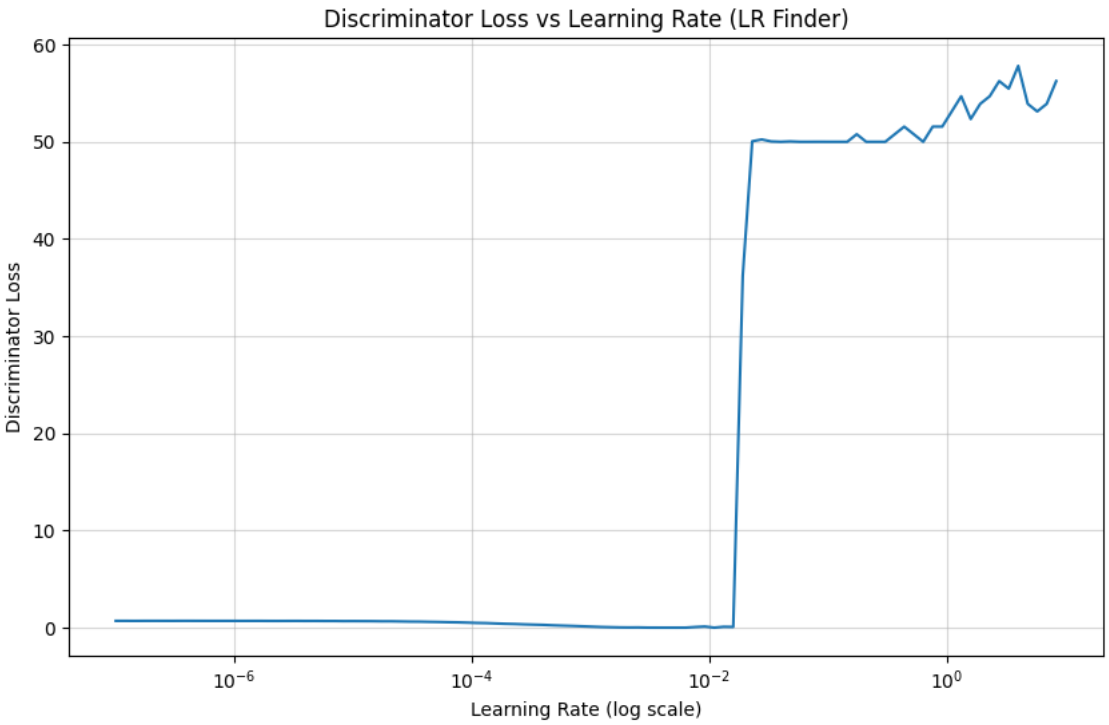

Рисунок 3.8. Результати роботи алгоритму Learning Rate Finder для дискримінатора ГЗМ. Графік демонструє залежність функції втрат від експоненційного зростання темпу навчання.

На відміну від класичних мереж, де функція втрат зазвичай має U-подібну форму, для ГЗМ спостерігається зона стабільності (loss ≈ 0) з наступним різким зростанням втрат («вибух градієнтів») при досягненні критичного значення $LR \approx 10^{-2}$. Це підтверджує необхідність використання менших значень кроку навчання ($10^{-4}…10^{-3}$) для забезпечення стабільності змагального процесу.

Learning Rate (Scale)	Discriminator Loss	Характеристика процесу
10^{-6} ... 10^{-4}	0.69 -> 0.40	Повільна збіжність
10^{-3}	≈ 0.05	Оптимальна зона (Minimum Loss)
> 10^{-2}	> 50.0	Розбіжність (Gradient Explosion)

Рисунок 3.9. Експериментальний графік пошуку темпу навчання (LR Finder) для дискримінатора.

Коригування темпу навчання є вирішальним для забезпечення стабільності процесу навчання. Неправильно підібраний темп може призвести до або надмірної динаміки моделі, яка не може стабілізуватися (надто високий темп), або до застрягання в локальних мінімумах (надто низький темп). Наприклад, в експериментах зі складними архітектурами ГЗМ, адаптивні методи налаштування темпу, такі як Adam, показують значно кращі результати завдяки здатності динамічно підлаштовувати швидкість навчання на основі перших і других моментів градієнтів.

Вибір планувальників зміни темпу навчання, таких як Cosine Annealing або Exponential Decay, дозволяє підлаштовувати швидкість навчання залежно від етапу тренування. Це особливо корисно на завершальних етапах навчання, коли необхідно досягти точного зближення (конвергенції) моделей.

Математично, адаптивне коригування темпу навчання α_t на основі другого моменту градієнтів можна представити наступним чином: якщо перший рівень каскаду не забезпечує необхідної продуктивності, застосовуємо адаптивні методи для темпу навчання α:

$$\alpha_t = \frac{\alpha_0}{\sqrt{\hat{v}_t} + \epsilon}, \tag{3.21}$$

де α_t — темп навчання на кроці t, який змінюється в процесі навчання за допомогою адаптивних методів або планувальників темпу навчання, α_0 — початкове значення темпу навчання, яке може коригуватися адаптивними методами, $\hat{v}_t$ — середнє другого моменту градієнтів, яке використовується оптимізатором Adam для адаптивного коригування темпу навчання, ϵ — допустиме відхилення між втратами на навчальних і валідаційних наборах. Виконання умови $|\mathcal{L}_{\text{train}} - \mathcal{L}_{\text{val}}| \leq \epsilon$ свідчить про мінімізацію розриву узагальнення (англ. generalization gap) та стабілізацію моделі, що є достатнім алгоритмічним критерієм для ініціалізації ранньої.

Альтернативним підходом є використання планувальників зміни темпу навчання:

$$\alpha_t = \alpha_{\min} + \frac{1}{2}(\alpha_{\max} - \alpha_{\min})\left[1 + \cos\left(\frac{\pi t}{T}\right)\right], \quad (3.22)$$

де T — кількість ітерацій або епох навчання, використовується для планування змін темпу навчання.

Перевіряємо критерії:

$$FID \leq FID_{threshold} \text{ та } | \mathcal{L}_{train}(t) - \mathcal{L}_{val}(t) | \leq \epsilon, \quad (3.23)$$

Якщо критерії не задоволено — продовжуємо оптимізацію.

3.5.4 Третій рівень — коригування функцій втрат і архітектури моделі

На третьому рівні здійснюється коригування функцій втрат і загальної архітектури моделі. Застосовуються наступні методи.

Інженерія нових функцій втрат або модифікація існуючих, таких як перехресна ентропія з обмеженнями на ваги або комбіновані функції втрат (наприклад, поєднання ГЗМ Васерштейна та ГЗМ на базі методу найменших квадратів).

Додавання нових методів регуляризації, таких як Dropout, Batch Normalization або Spectral Normalization, для покращення стабільності та запобігання перенавчанню.

Метод градієнтного штрафу (англ. gradient penalty) для покращення стабільності навчання, як це робиться в ГЗМ Васерштейна з градієнтним штрафом (WGAN-GP).

Внесення змін до архітектури генератора і дискримінатора для покращення їх взаємодії і результативності. Це може включати додавання або видалення шарів, зміну типів шарів (наприклад, використання шарів зворотного розподілу), застосування різних методів регуляризації тощо.

На третьому рівні додаткове коригування функцій втрат дозволяє досягти тонкого балансу між генератором та дискримінатором, що є критичним для уникнення проблем, таких як домінування дискримінатора або генератора. Наприклад, використання модифікованих функцій втрат дозволяє значно підвищити стабільність навчання, зменшуючи ризик зникнення градієнтів.

Внесення змін до архітектури моделі, таких як додавання або видалення шарів, використання різних методів нормалізації, дозволяє більш точно налаштувати взаємодію між генератором і дискримінатором. Це забезпечує кращу адаптацію моделі до складних розподілів даних і підвищує якість згенерованих зразків.

Математично, евристика третього методу має наступну логічну структуру. Якщо критерії продуктивності не досягнуті у двох попередніх каскадах, відбувається процес корекції функцій втрат. Наприклад, для ГЗМ на базі втрати Васерштейна:

$$\mathcal{L}_G = -\mathbb{E}_{\tilde{x}\sim p_g}[D(\tilde{x})],\ \mathcal{L}_D = -\mathbb{E}_{x\sim p_r}[D(x)] + \mathbb{E}_{\tilde{x}\sim p_g}[D(\tilde{x})], \tag{3.24}$$

де $\mathcal{L}_G$ — функція втрат для генератора, $\mathcal{L}_D$ — функція втрат для дискримінатора, $\tilde{x}$ — згенеровані зразки а p_g — їх розподіл, $x \sim p_r$ — справжні зразки.

Додаємо регуляризацію (це може бути, метод дропаут або пакетна нормалізація):

$$\hat{z}_i = \frac{z_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}, \tag{3.25}$$

$$\begin{cases} \hat{z}_i = \dfrac{z_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \\ y_i = \gamma \hat{z}_i + \beta \end{cases}, \tag{3.26}$$

де $\hat{z}_i$ — нормалізоване значення активації нейрону z_i в методі пакетної нормалізації, μ_i та σ_i середнє значення і дисперсія активацій по міні-пакету відповідно, γ та β — параметри масштабування та зміщення, що налаштовуються під час навчання.

Перевіряємо, чи виконуються критерії задовільності:

$$FID \leq FID_{threshold} \text{ та } | \mathcal{L}_{train}(t) - \mathcal{L}_{val}(t) | \leq \epsilon, \quad (3.27)$$

Цей підхід дозволяє використовувати два критерії (FID та зниження втрат), забезпечуючи комплексну оцінку якості та стабільності навчання моделі.

Об’єднана взаємодія компонентів каскадного методу оптимізації з критеріями FID і зниженням втрат на всіх трьох рівнях може бути представлена у наступному вигляді.

Каскадний метод оптимізації ГЗМ, псевдокод

```
Алгоритм 1. Каскадний метод оптимізації ГЗМ

Вхідні дані:
    Dataset X          // Навчальна вибірка
    P_z                // Розподіл шуму (для Алг. 2)
    H_space            // Простір гіперпараметрів
    target_FID         // Поріг генерації
    target_gap         // Поріг узагальнення

Вихідні дані:
    G*, D* // Оптимізовані моделі

Функція Main_Cascade_Loop(X, P_z, H_space):
    # --- РІВЕНЬ 1: Базове налаштування ---
    G, D = Init_Architecture(H_space)   // Вкл. пошук гіперпараметрів
    Init_Weights(G, D, method="Xavier/He")

    Train_Base(G, D, X)                 // Фіксований LR, аугментація
```

```
    Якщо Check_Convergence(G, D, X):
        Повернути Final_Validation(G, D, X)

    # --- РІВЕНЬ 2: Налаштування темпу навчання ---
    lr_optimal = Run_LR_Finder(G, D, X) // Пошук межі LR
    Scheduler = Configure_Scheduler(type="STLR", max_lr=lr_optimal)

    Train_Adaptive(G, D, X, Scheduler)  // Навчання за розкладом

    Якщо Check_Convergence(G, D, X):
        Повернути Final_Validation(G, D, X)

    # --- РІВЕНЬ 3: Глибока модифікація ---
    Loss_Func = Set_Loss("WGAN-GP")
    Apply_Spectral_Norm(D)              // Регуляризація
    Apply_Gradient_Penalty(D)           // Стабілізація

    G, D = Multiphase_Optimization(X, P_z) // Виклик Алгоритму 2
    Train_FineTuning(G, D, X, Loss_Func)   // Тонке налаштування

    Якщо Check_Convergence(G, D, X):
        Повернути Final_Validation(G, D, X)
    Інакше:
        Log_Error("Відсутність збіжності") // Глобальна помилка
        Повернути Restart_Cascade_With_New_Architecture()

Допоміжна функція Check_Convergence(G, D, X):
    current_FID = Evaluate_FID(G, X)
    Loss_Gap = |Loss_train - Loss_val|

    Якщо (current_FID < target_FID) ТА (Loss_Gap < target_gap):
        Повернути True
    Повернути False
```

3.6. Метод мультифазової оптимізації ГЗМ

Запропонована методика передбачає розділення процесу навчання ГЗМ на чотири послідовні оптимізаційні фази. Такий підхід, відомий як навчання за розкладом (англ. curriculum learning), демонструє високу ефективність при роботі зі складними розподілами даних, зокрема масивами зображень високої роздільної здатності. Фазова оптимізація дозволяє поступово підвищувати якість генерованих зразків, мінімізувати ризик колапсу моди та забезпечувати стабільність збіжності моделі.

Фаза 1 — попереднє навчання дискримінатора. Перша фаза зосереджена на формуванні надійної бази для дискримінатора. Процес розпочинається з навчання на реальних даних цільового розподілу. Як негативні приклади (англ. fake samples) на цьому етапі використовуються прості шумові дані або низькоякісні згенеровані зразки. Для підвищення узагальнюючої здатності та запобігання перенавчанню застосовуються методи регуляризації, такі як Dropout та аугментація даних. Ключовим завданням фази є досягнення стабільної класифікації реальних та синтетичних зразків, що підтверджується моніторингом метрик точності дискримінатора.

Фаза 2 — попередня ініціалізація («розігрів») генератора. На цьому етапі відбувається ініціалізація генератора (випадковими вагами або з використанням попередньо навченого автоенкодера). Навчання генератора здійснюється за допомогою спрощених функцій втрат (наприклад, на базі середньоквадратичної похибки, MSE, або покомпонентного порівняння ознак), що дозволяє наблизити вихідний розподіл до реального без жорсткого змагального тиску. Мета фази — навчити генератор створювати базові структури об’єктів. Оцінка результатів проводиться через візуальний контроль

та кількісні метрики. Періодично може активуватися дискримінатор для короткострокових ітерацій, щоб підтримувати мінімальний змагальний баланс.

Фаза 3 — попереднє змагальне навчання. Фаза передбачає введення повноцінної змагальної взаємодії між мережами, однак із використанням обмежених або спрощених функцій втрат для уникнення ранньої нестабільності (наприклад, вибуху градієнтів). Складність взаємодії збільшується поступово. Це дозволяє синхронізувати швидкості навчання обох мереж перед переходом до основного циклу. Контроль ефективності здійснюється за допомогою проміжних метрик стабільності.

Фаза 4 — точне доналаштування змагального навчання. Це фінальна стадія, що відповідає стандартному циклу навчання ГЗМ, де генератор і дискримінатор оптимізуються одночасно. Використовуються ваги, отримані на попередніх фазах (попередньо навчений дискримінатор з Фази 1 та «розігрітий» генератор з Фази 2). Для уникнення домінування однієї з мереж застосовується стратегія періодичного заморожування ваг одного з компонентів. Для підвищення робастності моделі вводиться ін’єкція шуму та пертурбації у вхідні дані дискримінатора. Опціонально можуть застосовуватися методи ансамблювання (усереднення виходів кількох ітерацій генератора) для покращення фінальної якості генерації. Для забезпечення стійкості алгоритму запроваджено механізм відкату (англ. rollback): у випадку детекції вибуху градієнтів або різкої деградації метрик, система автоматично повертається до фази попереднього змагального навчання.

Мультифазовий метод оптимізації забезпечує керовану еволюцію складності завдання для нейронних мереж. Декомпозиція процесу навчання на чотири етапи дозволяє нівелювати типові проблеми ГЗМ, такі як нестабільність градієнтів на старті та колапс моди, що робить цей підхід критично важливим для роботи з великими та складними наборами даних.

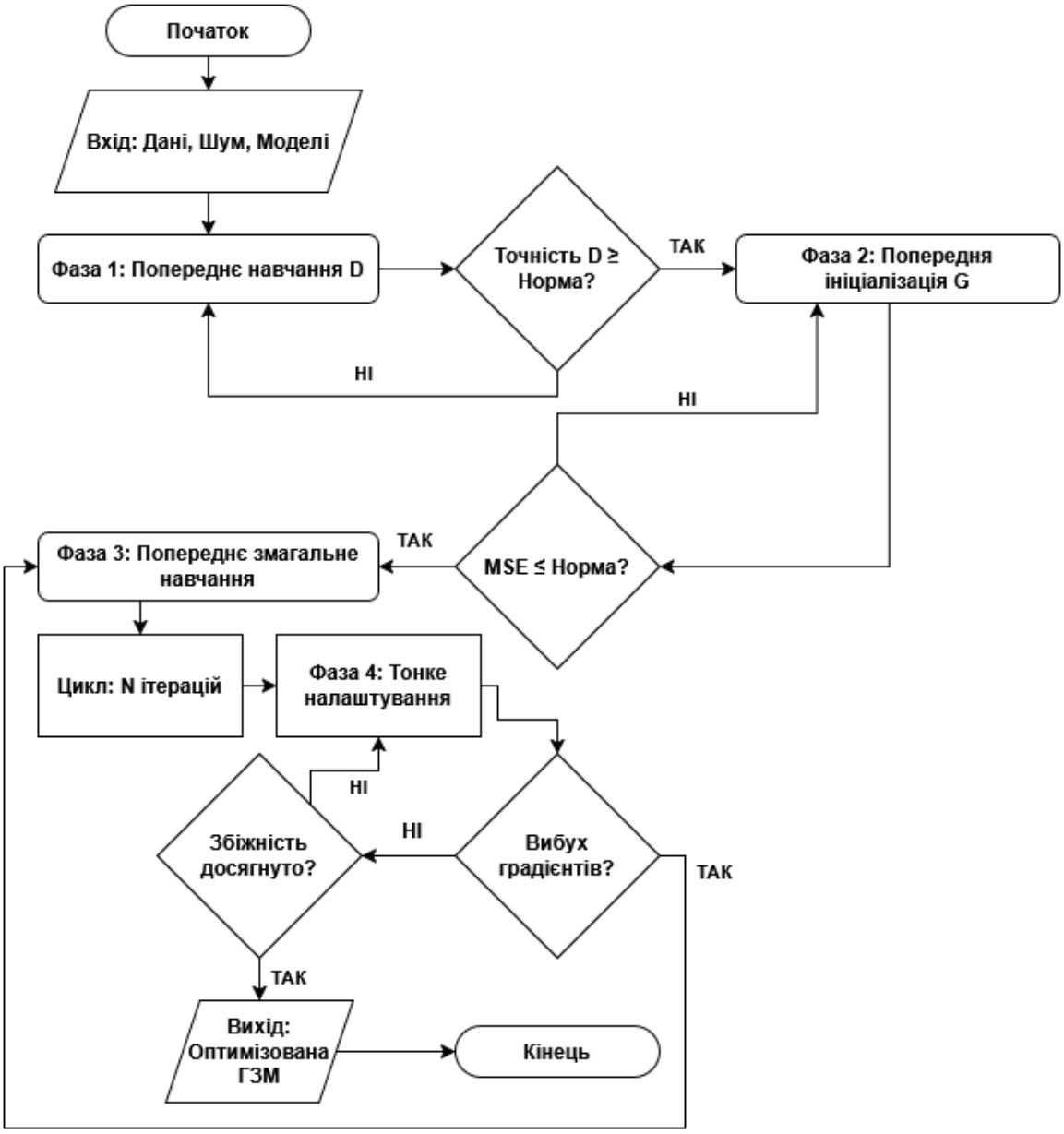

Рисунок 3.10. Блок-схема мультифазового методу оптимізації.

Мультифазова оптимізація ГЗМ, псевдокод

```
Вхідні дані:
    Dataset X (реальні зображення)
    P_z (розподіл латентного шуму)
    N_pretraining (кількість ітерацій попереднього змагання)
Ініціалізація:
    G (Генератор), D (Дискримінатор) з вагами ~ N(0, 0.02)
```

```
# --- ФАЗА 1: Попереднє навчання Дискримінатора ---
Повторювати поки Accuracy(D) < Threshold_1:
    x_real = sample(X)              // Вибірка реальних даних
    x_fake = noise(P_z)             // Генерація базового шуму
    x_real, x_fake = Apply_Augmentation(x_real, x_fake) //
Аугментація
    L_D = Loss(D(x_real), 1) + Loss(D(x_fake), 0)
    Update(D, minimize L_D)         // Оновлення ваг D (з Dropout)

# --- ФАЗА 2: Попередня ініціалізація (розігрів) генератора ---
Freeze(D)                           // Блокування ваг дискримінатора
Повторювати поки MSE(G(z), x_target) > Threshold_2:
    z = sample(P_z)
    x_target = sample(X)            // Структурні цільові патерни
    L_G_warmup = MSE(G(z), x_target)
    Update(G, minimize L_G_warmup)  // Оновлюємо тільки G

# --- ФАЗА 3: Попереднє змагальне навчання ---
Мітка: Phase3_Start
Unfreeze(D)                         // Розблокування ваг D
Для iter від 1 до N_pretraining:
    z = sample(P_z); x = sample(X)
    L_D = Simplified_GAN_Loss_D(x, G(z))
    L_G = Simplified_GAN_Loss_G(G(z))
    Update(D, minimize L_D, lr=small) // Оновлення з малим LR
    Update(G, minimize L_G, lr=small)

# --- ФАЗА 4: Тонке налаштування (Fine-tuning) ---
Повторювати до збіжності (Convergence):
    Для k кроків:                   // Цикл навчання критика
        x = sample(X); z = sample(P_z)
        L_D_wgan = WGAN_GP_Loss_D(x, G(z)) + Noise_Injection
        Update(D, minimize L_D_wgan)
```

```
    z = sample(P_z)
    L_G_wgan = WGAN_GP_Loss_G(G(z))
    Update(G, minimize L_G_wgan)

    Якщо iter % Checkpoint == 0:
        current_FID = Evaluate_FID(G, X)
        Якщо Loss_D АБО Loss_G -> Infinity (Вибух градієнтів):
            Restore_Weights(G, D)   // Відкат до стабільного стану
            Goto Phase3_Start       // Повернення на попередню фазу
        Інакше якщо current_FID < target_FID:
            Save_Model(G, D)

Вихід: Оптимізовані моделі G*, D*
```

3.7. Критерії збіжності та контроль стабільності ГЗМ

Оцінка ефективності розробленої архітектури здійснюється комплексно, поєднуючи макрорівень (управління каскадом) та мікрорівень (контроль фазових переходів), що забезпечує системний моніторинг динаміки навчання.

На макрорівні каскадного алгоритму ключовими індикаторами якості виступають глобальні метрики: відстань Фреше, яка оцінює розбіжність між розподілами ознак реальних та синтетичних даних, та індекс достовірності генерації, що вимірює чіткість і різноманітність згенерованих об'єктів. Зниження ВФ свідчить про зростання реалістичності генерації, і разом зі стабілізацією розриву узагальнення ці метрики слугують математичним тригером для успішного переходу між рівнями каскаду. Варто зазначити, що хоча ВФ та ІДГ є індустріальним стандартом, розроблений метод є гнучким і дозволяє інтеграцію специфічних інших цільових метрик (наприклад, PSNR або SSIM), якщо цього вимагає специфіка конкретної задачі.

На мікрорівні мультифазової оптимізації (Рівень 3) застосовуються локальні критерії переходу. Оцінка Фази 1 (попереднє навчання) базується на досягненні цільової точності бінарної класифікації дискримінатором. Ефективність Фази 2 (ініціалізація генератора) вимірюється мінімізацією похибки реконструкції відносно цільових структурних патернів. На фінальних змагальних етапах критичним критерієм стає безперервний моніторинг норми градієнтів: моніторинг їх аномального зростання автоматично ініціює алгоритмічний відкат системи до попереднього стабільного стану.

Таким чином, розроблений каскадний мультифазовий фреймворк забезпечує детермінований підхід до налаштування ГЗМ. На відміну від традиційного евристичного («хаотичного») підбору параметрів, ця ієрархічна стратегія послідовно долає типові проблеми ГЗМ — від колапсу моди до згасання градієнтів, гарантуючи баланс між обчислювальними витратами та робастністю фінальної моделі.

3.8. Модульність та гібридне розгортання фреймворку

Запропонована методика розроблена як гнучкий науково-практичний фреймворк, архітектура якого вирізняється високим ступенем модульності. Це означає, що каскадний та мультифазовий методи не є жорстко залежними один від одного і можуть ефективно розгортатися як самостійні інструменти. Зокрема, ізольоване застосування каскадного підходу є доцільним для швидкого налаштування класичних моделей ГЗМ, коли пріоритетом є автоматизований пошук архітектурних гіперпараметрів та адаптація темпу навчання. З іншого боку, мультифазова оптимізація може діяти як незалежний модуль для «м’якого» навчання (англ. curriculum learning) вже затверджених складних архітектур, де критично важливо запобігти ранньому колапсу моди за рахунок фазового розділення тренування.

Найвищий рівень робастності та продуктивності фреймворку розкривається саме у гібридній комбінації методів. У такому сценарії мультифазова оптимізація інкапсулюється як спеціалізований механізм на етапах базового каскаду (Рівень 2). Таким чином каскадний алгоритм стабілізує ландшафт втрат та знаходить оптимальні базові ваги, формуючи кращі початкові умови. Своєю чергою, мультифазовий алгоритм стабілізує змагальну взаємодію, нівелюючи вибухи градієнтів під час фінального тонкого налаштування.

Завдяки такій архітектурній варіативності, розроблений фреймворк дозволяє масштабувати обчислювальний процес відповідно до умов конкретної задачі. Залежно від наявних апаратних ресурсів, часових квот та специфіки цільового розподілу даних, можливе ізольоване застосування одного з методів, або ж задіяння їх гібридної комбінації для досягнення високої стабільності ГЗМ.

3.9. Перспективні напрями досліджень параметричної та архітектурної оптимізації ГЗМ

Одним із ключових аспектів, що потребує поглибленого вивчення, є стохастична та адаптивна оптимізація темпу навчання і архітектурна структуризація ГЗМ, особливо в контексті їх автономного розгортання на кордонних обчислювальних пристроях. Хоча в рамках даного дослідження було розглянуто ефективність базових адаптивних методів (Adam, RMSprop, Nadam) та принципів проєктування ГЗМ, специфіка сідлових точок у топології функцій втрат ГЗМ в умовах жорстких апаратних обмежень вимагає розробки більш спеціалізованих підходів.

Основними векторами подальшої роботи вбачаються:

- Дослідження гібридних планувальників для мінімізації обчислювальних витрат. Аналіз ефективності комбінованих стратегій, таких як Cosine

Annealing with Warm Restarts (SGDR) або циклічних змін темпу навчання. Важливо дослідити, як періодичні «стрибки» темпу допомагають моделі виходити з локальних мінімумів (колапс моди) при суттєво меншій кількості ітерацій, що є критичним для економії енергоресурсів мікрокомп'ютерних систем.

- Адаптація на основі нечіткої логіки для вбудованих систем. Розробка обчислювально ефективних вбудованих контролерів на базі нечіткої логіки, які динамічно регулюють темп навчання залежно від нелінійних співвідношень між втратами генератора і дискримінатора та метриками якості. Це може дозволити автоматизувати процес балансування мереж у реальному часі безпосередньо на кінцевому пристрої без необхідності ресурсномістких градієнтних обчислень.
- Апаратна імплементація та стиснення моделей. Подальша адаптація каскадного мультифазового фреймворку для розгортання на мікрокомп'ютерах та спеціалізованих NPU, що передбачає дослідження методів глибокого квантування ваг та алгоритмічного відсікання синаптичних ваг. Це може дозволити зберегти стабільність спроєктованої моделі при мінімізації використання пам'яті та енергоспоживання.
- Теоретико-інформаційна оптимізація латентного простору. Застосування принципів теорії інформації для мінімізації надмірності між шарами генератора та дискримінатора. Аналіз динаміки взаємної інформації та концепції інформаційної вузини дозволить математично обґрунтувати межі стиснення моделей ще на етапі фази тонкого налаштування, роблячи їх оптимальними для автономних систем.

Окреслені напрямки досліджень мають потенціал трансформувати підхід до імплементації ГЗМ на кордонних пристроях та мікрокомп'ютерних

платформах, перетворивши вибір темпу навчання з евристичної процедури на керований, математично обґрунтований процес, а проєктування моделей — ієрархічно впорядкованою інженерною процедурою.

Це відкриває нові можливості для підвищення стабільності моделей у задачах широкого спектру застосування.

3.10. Потенційні виклики та стратегії їх подолання

Основні виклики, що виникають під час реалізації каскадного методу, та стратегії їх вирішення узагальнено в таблиці 4.

Таблиця 4. Виклики каскадного методу.

Тип виклику	Опис проблеми	Стратегія подолання
Рання нестабільність	Висока амплітуда коливань Loss-функції на старті через відсутність сформованих границь рішень у дискримінатора.	• Попереднє навчання D (Pre-training) • Використання WGAN-GP (обмеження градієнтів) • Зменшений Learning Rate на старті (Warm-up)
Дисбаланс навчання	Домінування одного з компонентів (зазвичай дискримінатора), що призводить до проблеми зникаючих градієнтів.	• Динамічне коригування частоти оновлення ваг (k кроків D на 1 крок G) • Тимчасове заморожування сильнішого компонента • Одностороннє згладжування мітки
Колапс моди	Генератор створює одноманітні зразки з низькою ентропією, ігноруючи різноманіття реальних даних.	• Спектральна нормалізація • Dropout у дискримінаторі • Додавання шуму до вхідних даних • Міні-пакетна дискримінація

3.11. Висновки за розділом 3

У даному розділі вирішено важливе науково-практичне завдання, що полягає у розробці методів архітектурної структуризації та алгоритмічної стабілізації ГЗМ для їх подальшого розгортання в умовах обмежених обчислювальних ресурсів кордонних пристроїв та мікрокомп’ютерних платформ. Встановлено, що критичними факторами, які обмежують

застосування ГЗМ, є нестабільність градієнтів, ризик колапсу моди, чутливість до ініціалізації ваг та налаштування гіперпараметрів, а також значна вимогливість до обчислювальних ресурсів.

Наукова новизна отриманих результатів:

1. Вперше розроблено математичну модель каскадної оптимізації ГЗМ, яка, на відміну від існуючих, базується на ієрархічній декомпозиції простору гіперпараметрів та врахуванні адаптивної динаміки навчання, що дозволяє забезпечити необхідну точність та швидкодію функціонування ГЗМ в умовах апаратно-параметричних обмежень кордонних пристроїв.
2. Вперше розроблено мультифазовий метод оптимізації навчання ГЗМ, який, на відміну від існуючих, базується на багаторівневому механізмі адаптивної конвергенції, що дозволяє запобігати колапсу моди та зникненню градієнтів функцій втрат без підвищення обчислювальної складності процедури навчання.
3. Удосконалено механізм адаптації ГЗМ, який ґрунтується на гібридному комбінуванні каскадного та мультифазового методів. Обґрунтовано модульний характер розробленої каскадної мультифазової архітектури, що передбачає можливість її експлуатації як у вигляді інтегрованого гібридного комплексу, так і шляхом структурної декомпозиції на незалежні підсистеми. Доведено, що подібна архітектурна адаптивність забезпечує динамічне масштабування обчислювального навантаження залежно від цільових вимог, що є необхідною умовою для імплементації ресурсномістких ГЗМ в умовах автономних кордонних обчислень.

РОЗДІЛ 4. ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ ТА АНАЛІЗ ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ МЕТОДІВ ОПТИМІЗАЦІЇ ГЗМ

4.1. Генерація графічних зразків рукописного тексту

З метою реалізації стратегії та методів оптимізації генеративних змагальних нейронних мереж, запропонованих у попередньому розділі, які б забезпечили більш стабільне і ефективне навчання з метою покращення точності моделей, розроблено різнотипні ГЗМ і проведено ряд експериментів.

Конвенційна ГЗМ (англ. Vanilla GAN) [36]. Перша спроєктована модель була створена з метою синтезу реалістичних зображень, схожих на зразки з набору даних MNIST, який складається з рукописних цифр. Щоб досягти цього, було імплементовано кілька методів і підходів до оптимізації. Зокрема а) комбінацію функції активації LeakyReLU, б) бінарної крос-ентропійної функції втрати та в) одностороннє згладжування міток для стабілізації навчання.

Використання згладжування міток передбачає призначення цільового значення 0,9 реальним зображенням замість радикального 1, що може зробити навчання дискримінатора більш надійним. Також було застосовано оптимізатор Adam з низькою швидкістю навчання та використано шум як вхідні дані для генератора для поступового синтезування зображень. Крім того, реалізовано функцію візуалізації для моніторингу прогресу навчання та якості згенерованих зразків. Таким чином було проведено як параметричну, так і структурну оптимізацію.

Мета — навчити модель створювати реалістичні зображення, схожі на рукописні зразки MNIST. Підхід передбачав вибір архітектури, нормалізацію, функції активації, функції втрат, згладжування міток і методи оптимізації для підвищення стабільності навчання та якості зображення, що в кінцевому

підсумку спрацювало на досягнення головної мети — створення високоякісних синтетичних зображень [36].

У межах цього дослідження для практичної реалізації та тестування ГЗМ було використано стаціонарний ПК, мову програмування Python версії 3.7.6, а також бібліотеки TensorFlow і Keras.

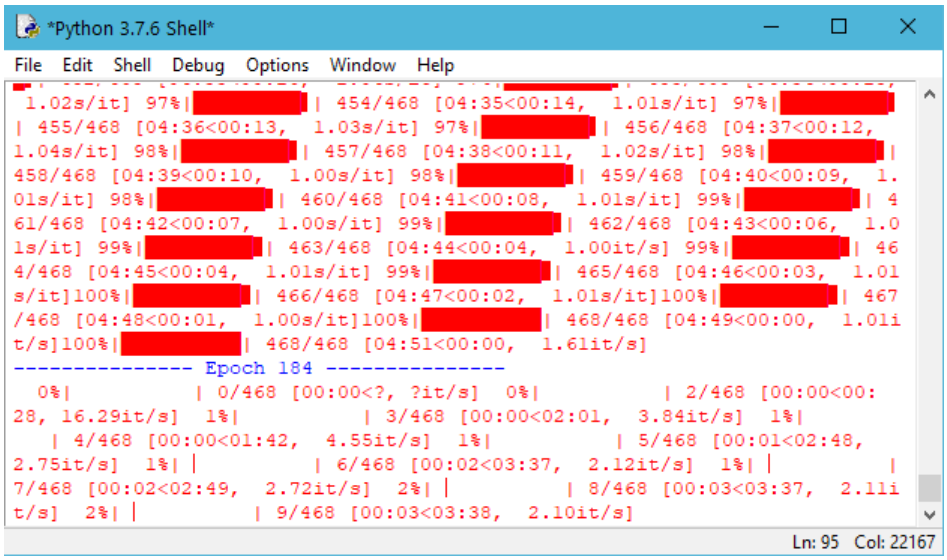

Рисунок 4.1. Ілюстрація процесу генерації зображень.

Для стохастичної оптимізації застосовувався алгоритм Adam. Цей метод дозволяє використати процес стохастичного градієнтного спуску. Алгоритм базується на адаптивних оцінках моментів нижчого порядку та розроблений для оптимізації стохастичних цільових функцій, що базуються на градієнтах першого порядку. Відповідно до роботи Кінгми та його колег [86], алгоритм Adam є «обчислювально ефективним, вимагає мало пам'яті, інваріантний до діагонального масштабування градієнтів та добре підходить для задач, які є великими як за обсягом даних, так і за кількістю параметрів». Цей алгоритм був вбудований у мережеву архітектуру для виконання стохастичної оптимізації в обох моделях — як у генераторі, так і в дискримінаторі. Основне експериментальне завдання полягало в оптимізації швидкості навчання та стабілізації процесу генерації в розробленій нейронній мережі.

4.1.1. Експериментальні дослідження

Для навчання нейронної мережі використовувалася база даних MNIST. База даних MNIST складається з 60 000 навчальних графічних зразків і 10 000 тестових зразків (рисунок 4.2). База даних MNIST є стандартом де-факто для оцінки продуктивності моделей у задачах розпізнавання рукописних цифр, і вона широко використовується в дослідженнях машинного навчання та комп'ютерного зору. База даних складається з зображень розміром 28x28 пікселів, що надає змогу моделі ефективно аналізувати та обробляти вхідні дані з низькою роздільною здатністю. Кожне зображення унікальне і представляє одну цифру від 0 до 9, що робить набір даних ідеальним для тренування та тестування алгоритмів, зокрема генеративних змагальних нейронних мереж. Завдяки своїй простоті та стандартизації, MNIST дозволяє порівнювати результати різних моделей на основі однорідного набору даних, що значно полегшує аналіз ефективності різних архітектур нейронних мереж. Використання MNIST у контексті даного дослідження забезпечує можливість не лише створювати високоякісні синтетичні зображення, але й виконувати детальне оцінювання ефективності запропонованих методів оптимізації та архітектур генеративних змагальних мереж.

Рисунок 4.2. Реальні зразки рукописних цифр з набору даних MNIST.

Процес навчання моделі охоплював 400 ітераційних циклів (епох). На початковому етапі результати генерації характеризувалися високим рівнем дисперсії та низькою візуальною якістю: морфологічні ознаки символів були слабко вираженими, а контури об’єктів залишалися нечіткими через значну концентрацію стохастичного шуму та артефактів. Проте вже після першої епохи навчання модель продемонструвала здатність до апроксимації структури вхідних даних, сформувавши графічні образи, що відповідають топологічним характеристикам цифр.

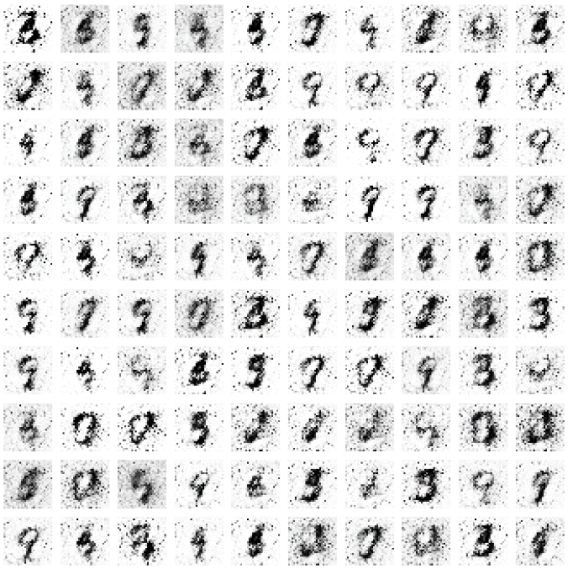

Рисунок 4.3. Графічні зображення цифр, згенеровані розробленою в рамках дослідження ГЗМ, отримані після першого циклу навчання нейронної мережі.

Аналіз динаміки навчання станом на 10-ту епоху (рисунок 4.4) ілюструє успішне завершення фази початкової ініціалізації та перехід системи до активного мінімаксного протистояння. На ранніх етапах (до 5-ї епохи) спостерігається стрімке зростання точності дискримінатора до значень, близьких до абсолютного максимуму, що свідчить про його високу здатність до швидкої екстракції базових ознак тренувальної вибірки. Проте ключовим індикатором стабільності обраної архітектури є переломний момент, зафіксований після 7-ї епохи: функція втрат генератора, досягнувши свого

пікового значення, переходить у фазу впевненого спаду, що супроводжується первинною тенденцією до зниження точності дискримінатора. Така поведінка метрик підтверджує, що генератор успішно долає ризик затухання градієнтів, ефективно адаптується до жорстких критеріїв дискримінатора та починає формувати релевантні структурні патерни, закладаючи надійний фундамент для подальшої конвергенції моделі.

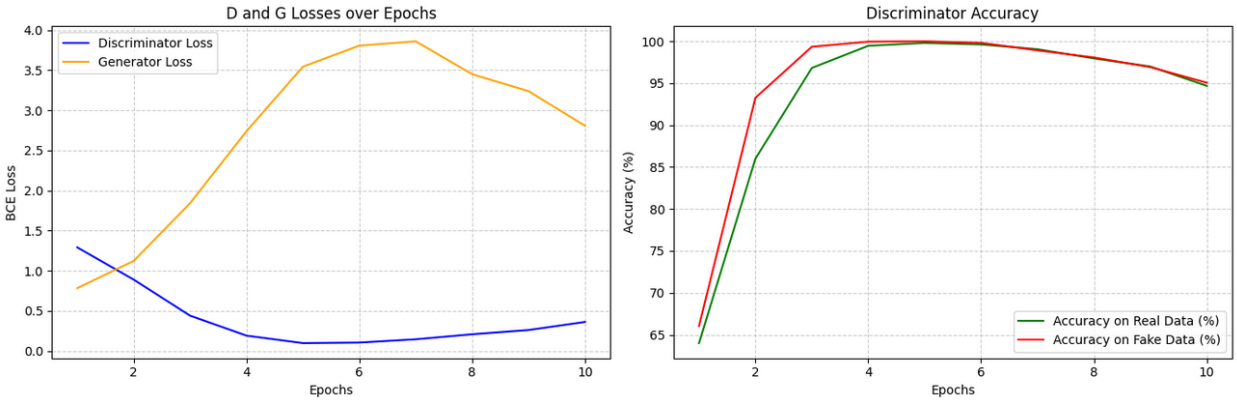

Рисунок 4.4. Графік втрат після десятого циклу навчання.

Порівнюючи ці результати з зразками, отриманими після завершення двадцятої епохи тренування (рисунок 4.5), можна помітити, що згенеровані зображення набули відчутної структури та форми.

Рисунок 4.5. Зразки були створені після двадцятої епохи.

Отримані зразки значно більше нагадували цифри, ніж результати першої епохи, що свідчить про ефективність вибраного алгоритму тренування та оптимізації. Проте лише чверть зразків можна було вважати успішними; інші символи не мали чіткої форми, містили сліди шуму або зовсім не були схожі на цифри.

Аналіз стану системи на 200-й епосі (рисунок 4.6) підтверджує досягнення довготривалої стабільності процесу генерації. Після встановлення базової рівноваги на попередніх етапах, модель успішно закріплюється у фазі тонкого налаштування простору параметрів. Метрики демонструють стійку поведінку без критичних осциляцій чи ознак затухання градієнтів, що є поширеним ризиком для тривалих циклів оптимізації. Головною особливістю цього етапу є стабільне утримання генератором своєї репрезентативної здатності та повна відсутність колапсу моди на значному часовому проміжку. Це доводить, що обрана архітектура ефективно впоралася із завданням тривалої апроксимації складного розподілу даних, забезпечивши надійне та послідовне збереження структурних ознак цільових об’єктів протягом усього тренувального періоду.

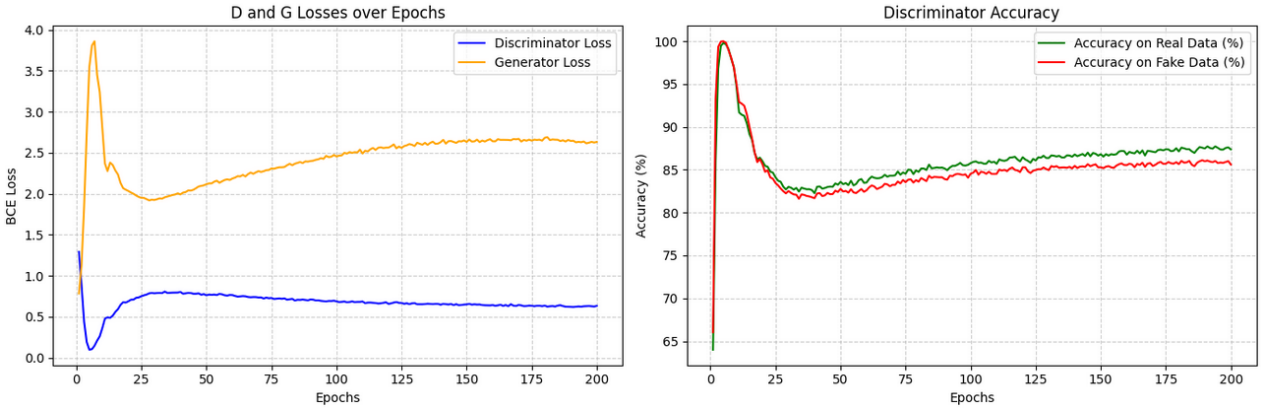

Рисунок 4.6. Графік втрат після 200-го циклу навчання.

Результати 400-ї епохи тренування свідчать про успішність експерименту, оскільки переважна більшість отриманих символів була схожою

на реальні цифри, вони відзначалися достатнім рівнем чіткості, структурованості, а також відсутністю значного графічного шуму. Для отримання ще більш якісних зображень експеримент можна повторити з тими ж параметрами, але збільшивши кількість епох до 500 або 1000. Проте без подальшого вдосконалення та оптимізації це збільшить тривалість генерації кінцевих результатів.

Результати експерименту з оптимізованою ГЗМ представлені на рисунку 4.7.

Рисунок 4.7. Зразки, згенеровані ГЗМ після 400 епох.

4.1.2. Порівняння ефективності моделей

Порівняння результатів моделей із застосуванням запропонованої каскадної стратегії оптимізації та моделі без оптимізації. *Модель А* була оптимізована із застосуванням каскадного методу, *модель Б* представляє собою класичну реалізацію ГЗМ без модифікацій.

Таблиця 5. Порівняння моделей.

Метрика	Модель А	Модель Б
Втрата генератора	2.57	9.16
Втрата дискримінатора	0.47	0.00
Точність дискримінатора	91.84%	100%
Час навчання (секунди)	2832	2600
FID	~19.4	~45.9

Модель А, яка використовує каскадний метод оптимізації, демонструє значно нижчу середню втрату генератора (2.57 проти 9.16 у Моделі Б). Це свідчить про те, що вдосконалення, такі як Batch Normalization, LeakyReLU, Tanh та вдосконалені оптимізатори (Adam із OneCycleLR), сприяють більш ефективному навчанню генератора.

У Моделі Б дискримінатор має майже нульову втрату (0.000782) і 100% точність, що свідчить про те, що дискримінатор домінує над генератором. Це може бути ознакою «перенавчання» дискримінатора, через що генератор не може навчитися створювати якісніші зразки.

У Моделі А точність дискримінатора становить 91.84%, що вказує на більш збалансоване навчання між генератором і дискримінатором. Це краще відображає реальну взаємодію між цими двома компонентами ГЗМ.

Модель А потребує трохи більше часу для навчання (2832 секунд проти 2600 секунд у Моделі Б), що очікувано через використання більш складних алгоритмів оптимізації. Проте додатковий час виправданий покращеною продуктивністю моделі ГЗМ.

Модель А більш ефективно навчає генератор створювати зразки, які важче відрізнити від справжніх (FID < 20). У той час як Модель Б, через брак удосконалень, не досягає такого рівня якості.

Цей експеримент демонструє потенціал генеративних змагальних нейронних мереж у створенні високоякісних графічних зразків за умови правильного вибору параметрів і налаштувань нейронної мережі. У результаті генеративна змагальна нейронна мережа створила нові зразки цифр, які виглядають як рукописні, і отримані результати можна вважати фотореалістичними графічними зразками.

Отримані результати підтверджують потенціал ГЗМ у створенні якісних графічних зображень, що дозволяє розглядати цей метод як універсальну систему генерації та механізм, який може бути застосований у різних галузях і для вирішення завдань різної складності.

4.2. Генерація фотореалістичних зображень відбитків пальців за допомогою адаптивної глибокої згорткової ГЗМ

Відбитки пальців є одними з найважливіших біометричних характеристик, що використовуються для ідентифікації осіб у різних сферах діяльності, включаючи правоохоронні органи та судову експертизу. Через їх унікальність і незмінність протягом життя вони є ефективним інструментом для розпізнавання. Проте використання реальних відбитків пальців для навчання моделей глибокого навчання має обмеження через ризики порушення конфіденційності. Щоб обійти ці обмеження, пропонується використовувати штучно згенеровані зображення відбитків, які не пов’язані з реальними особами. Вони можуть бути використані для навчання моделей без ризику порушення приватності.

4.2.1. Архітектура моделі та методи оптимізації

Для вирішення задачі генерації фотореалістичних відбитків пальців була розроблена адаптивна глибока згорткова генеративна змагальна мережа (ADCGAN). Модель базується на архітектурі Deep Convolutional GAN (DCGAN), яка була модифікована для покращення результатів генерації.

Структура генератора. Генератор використовує набір шарів, включаючи згорткові транспоновані шари (англ. convolutional-transpose), нормалізацію партій (англ. batch norm) і функцію активації ReLU. Ці шари дозволяють поступово перетворювати вхідний шум у реалістичні зображення. Вхідні дані для генератора — це латентний вектор, що представляє випадковий шум, з якого модель генерує зображення.

Генератор моделі спроєктований шляхом поступової трансформації латентного вектору в об’єм даних, еквівалентний зображенню. Вхідний латентний вектор z вибирається із стандартного нормального розподілу. Структура генератора включає наступні шари:

- **ConvTranspose2d:** виконує операцію транспонованої згортки для просторового збільшення проміжних репрезентацій, що дозволяє трансформувати низькорозмірний вектор шуму у фінальне зображення.
- **BatchNorm2d:** реалізує просторову пакетну нормалізацію відображень. Це забезпечує стабілізацію динаміки навчання та прискорює конвергенцію шляхом зменшення внутрішнього коваріативного зсуву.
- **ReLU:** функція активації, яка шляхом нелінійного перетворення просторових відображень дозволяє генератору моделювати складні ієрархічні ознаки зображень, зберігаючи при цьому стабільний градієнтний потік під час навчання.

Ці шари спільно дозволяють генератору послідовно покращувати якість створюваних зображень відбитків пальців у процесі навчання.

Структура дискримінатора. Дискримінатор, у свою чергу, містить шари звичайних згорток (англ. strided convolution), пакетну нормалізацію та функцію активації LeakyReLU. Він отримує зображення (реальне або згенероване) і визначає, чи належить воно до справжніх даних.

Дискримінатор включає такі ключові елементи:

- **Conv2d:** виконує операцію двовимірної згортки для екстракції локальних просторових ознак. Використання згортки з певним кроком забезпечує зниження просторової розмірності вхідного тензора, формуючи ієрархічні відображення без втрати критичної семантичної інформації.
- **BatchNorm2d:** пакетна нормалізація, яка стабілізує процес оптимізації шляхом центрування та масштабування просторових відображень..
- **LeakyReLU:** нелінійна функція активації, що гарантує збереження ненульових градієнтів для будь-яких вхідних даних; запобігає блокуванню процесу оптимізації та дозволяє мережі стабільно формувати та розпізнавати складні ієрархічні ознаки зображень.

Ці шари ГЗМ забезпечують ефективну роботу дискримінатора, дозволяючи йому більш точно розрізняти реальні та згенеровані зображення.

4.2.2. Методи оптимізації

У процесі навчання було використано кілька методів оптимізації відповідно до запропонованих у попередніх розділах фреймворків.

Функція втрат. Для навчання генератора і дискримінатора використовувалася *функція бінарної крос-ентропії* (BCELoss). Ця функція визначає оцінку якості розпізнавання дискримінатором реальних і згенерованих зображень.

Оптимізатор Adam. Модель використовувала оптимізатор Adam з різними значеннями швидкості навчання для генератора і дискримінатора. Для

генератора було обрано значення швидкості навчання 0.0001, а для дискримінатора — 0.0004. Ці параметри дозволили досягти балансу між швидкістю збіжності та стабільністю навчання моделі.

Підбір гіперпараметрів. Проведено кілька експериментів для налаштування гіперпараметрів, таких як розмір партії, розмір зображень, кількість каналів та розмір латентного вектора. Оптимальними були обрані наступні параметри: розмір партії — 128, розмір зображень — 64x64 пікселі, розмір латентного вектора — 100, кількість каналів — 1 для зображень у градаціях сірого.

Ініціалізація ваг. Для забезпечення стабільного навчання моделей генератора та дискримінатора використана спеціальна ініціалізація ваг. Для згорткових шарів ваги ініціалізувалися нормальним розподілом із середнім значенням 0 і стандартним відхиленням 0.02. Це дозволяє уникнути проблем, пов’язаних із занадто великими або занадто малими вагами на початку навчання, що може призводити до нестабільності моделі.

Оптимізація цих параметрів допомогла досягти стабільного навчання і високої якості генерації зображень.

4.2.3. Навчання та результати

Для навчання моделі було використано базу даних Sokoto Coventry Fingerprint Dataset (SOCOFing) [120], що містить 6000 зображень відбитків пальців.

Після декількох циклів навчання ADCGAN почала генерувати синтетичні зображення, які вже нагадували реальні дактилограми. На перших етапах навчання модель створювала зображення з великим рівнем шуму і нечіткими контурами. Перші результати свідчили про необхідність подальшої оптимізації та корекції гіперпараметрів.

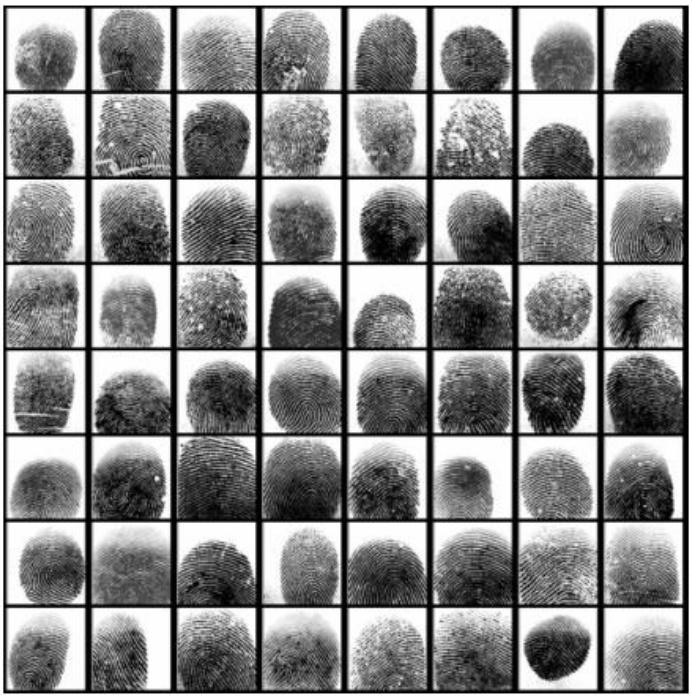

Рисунок 4.8. Реальні дані SOCOFing — скани відбитків пальців.

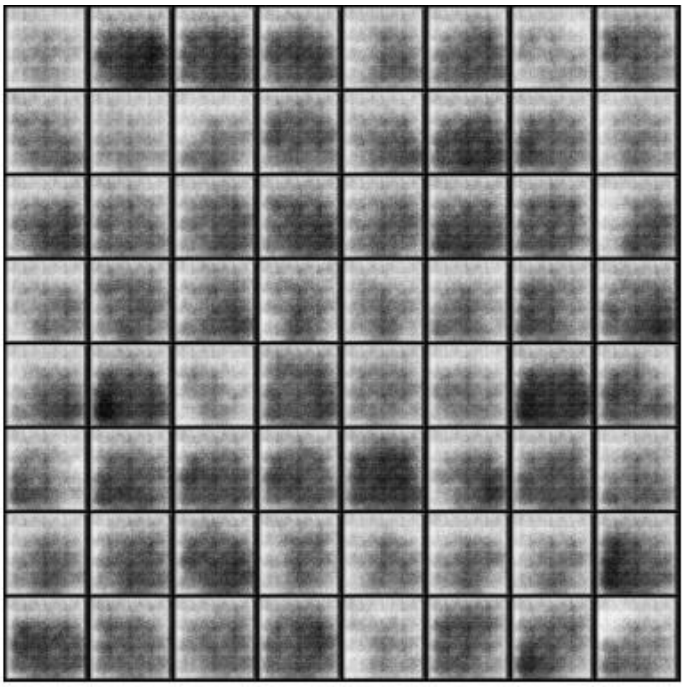

Рисунок 4.9. Отримані результати після двох навчальних епох.

Кращі результати були досягнуті після 1000-1300 епох навчання, коли згенеровані зображення набули чіткіших форм і стали максимально наближеними до реальних відбитків. Проте після 1400 епох спостерігалася

проблема дублювання зразків, що вказує на можливий колапс моди, що потребувало подальшого вдосконалення моделі.

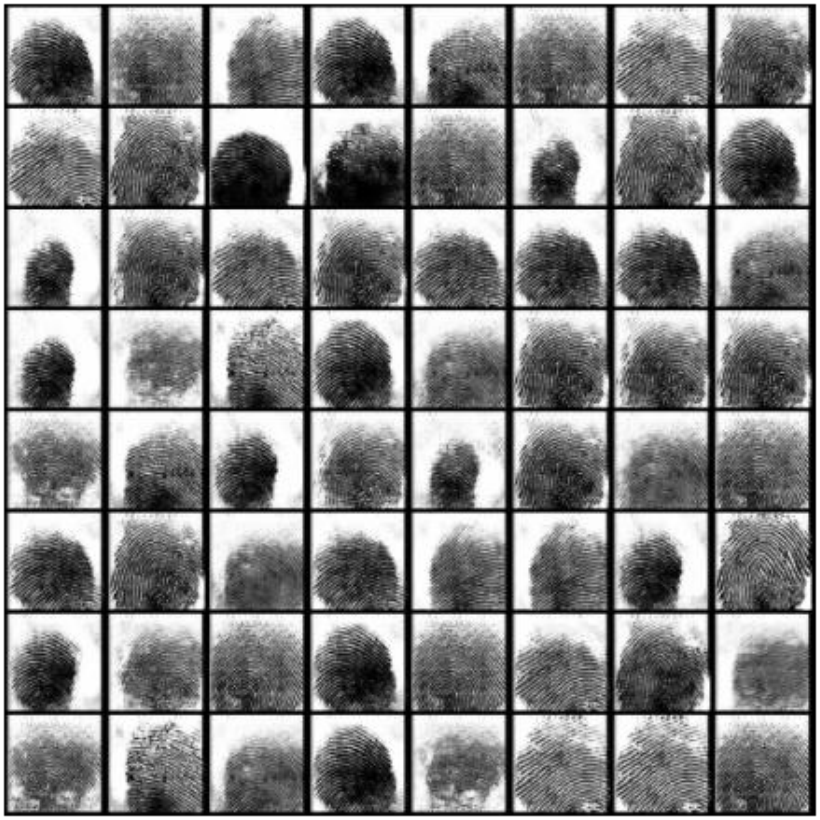

Рисунок 4.10. Отримано результати після 1400 епох тренувань.

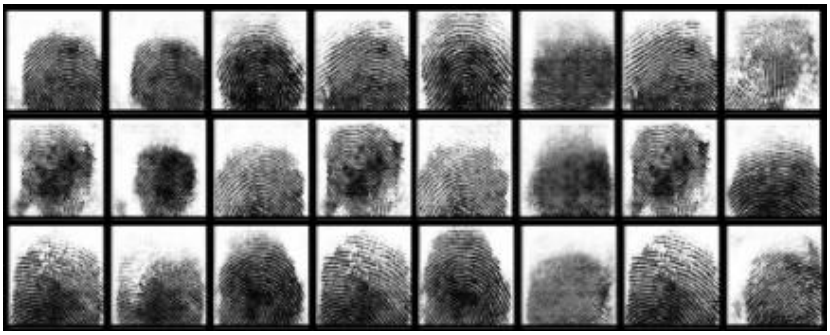

Рисунок 4.11. Згенеровані результати, отримані від 1000 до 1300 епох навчання.

4.2.4. Поліпшення якості зображень

На кожному етапі навчання модель ADCGAN генерувала нові зображення відбитків пальців, які порівнювалися з реальними зображеннями. Зображення, отримані після 1000-1300 епох, були найбільш якісними і

близькими до реальних дактилограм. Ці результати допомогли виявити оптимальну кількість епох для тренування моделі і підкреслили необхідність подальшої оптимізації.

Після чого модель було значною мірою вдосконалено, залишивши усі переваги швидкодії глибокої згорткової ГЗМ та її незначної вибагливості до споживання ресурсів системи та електроенергії.

4.2.5. Вдосконалена модель

Вдосконалена модель (EAWGAN), що була використана для задачі синтезу відбитків пальців, реалізує ВГЗМ (ГЗМ на базі втрати Васерштейна) з градієнтним штрафом (WGAN-GP), що забезпечує перевагу у порівнянні з першою моделлю. Замінюючи традиційну функцію втрат на основі бінарної крос-ентропії на відстань Васерштейна, ВГЗМ усуває поширені проблеми, такі як колапс моди та нестабільність навчання, завдяки застосуванню обмеження Ліпшиця через терм градієнтного штрафу. На відміну від попередньої моделі, яка використовує сигмоїдні активації та бінарну класифікацію, ВГЗМ відкидає ймовірнісні виходи на користь оцінок критика, що забезпечує гладкіші та більш інформативні градієнтні оновлення. Також до моделі було застосовано каскадний та фазовий методи оптимізації. Додатково, WGAN-GP покращує динаміку навчання завдяки рідшим оновленням генератора (кожні 5 кроків дискримінатора, а не кожен крок) та використанню нормалізації за інстанціями у дискримінаторі, що сприяє кращій передачі градієнтів та більшій різноманітності зразків. Ця модель також збільшує роздільну здатність вихідних зображень до 128×128, що дозволяє генерувати зображення вищої якості, що особливо корисно для завдань, які потребують точного відтворення деталей, таких як синтез відбитків пальців. Ці вдосконалення роблять WGAN-GP більш надійною та стабільною генеративною моделлю, особливо для застосувань, що включають структуровані та високорозмірні зображення.

Рисунок 4.12. Згенеровані ГЗМ (EAWGAN) результати, отримані після 170 епох навчання.

Рисунок 4.13. Згенеровані ГЗМ (EAWGAN) результати, отримані після 200 епох навчання.

Модель продемонструвала здатність до генерації набагато більш реалістичніших зразків відбитків пальців за суттєво менший проміжок часу. Реалістичних результатів вдалося досягти вже за 170 епох навчання. Обидві моделі продемонстрували очікувані показники FID після каскадної оптимізації (моделі окремо протестовані на датасеті MNIST з метою оптимізації обчислювальних ресурсів та часу). Каскадна ADCGAN FID на рівні ~16, а каскадна EAWGAN FID на рівні ~20. Результати проведених експериментів підтвердили ефективність моделей на базі ГЗМ для створення фотореалістичних відбитків пальців. Ці синтетичні зображення можуть бути використані для створення нових «безпечних для конфіденційності» наборів даних, для генерації реалістичних сенсорних даних для біометричних систем

або для збільшення розміру та різноманітності існуючих публічних наборів даних.

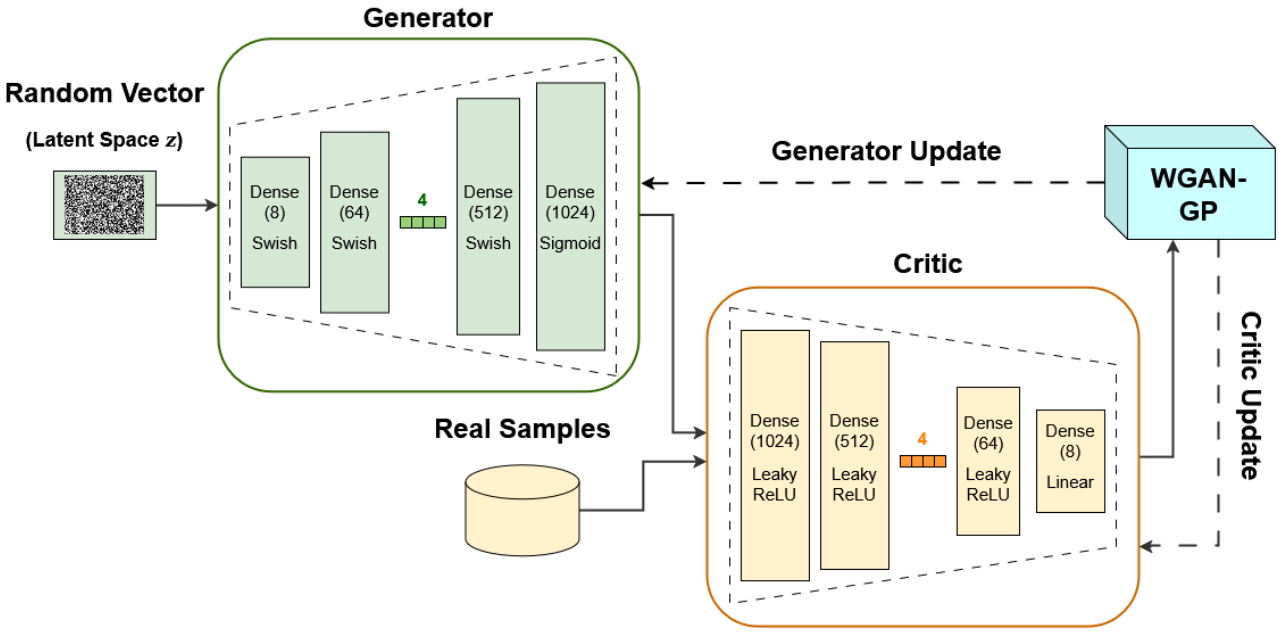

Рисунок 4.14. Архітектура моделі на базі WGAN-GP.

Результати проведених експериментів підтвердили ефективність моделей на базі ГЗМ для створення фотореалістичних відбитків пальців. Ці синтетичні зображення можуть бути використані для створення нових «безпечних для конфіденційності» наборів даних, для генерації реалістичних сенсорних даних для біометричних систем або для збільшення розміру та різноманітності існуючих публічних наборів даних.

4.3. Використання генеративних змагальних мереж у мобільних додатках для покращення якості зображень

ГЗМ є одним із найсучасніших інструментів для розв’язання завдань, пов’язаних із маніпуляціями зображеннями. Вони здатні синтезувати, комбінувати та відновлювати графічні зразки високої якості, що робить їх незамінними у сферах, де потрібна висока точність зображень, таких як медицина, безпека, та цифрова фотографія. У цьому підрозділі розглядається

застосування ГЗМ для покращення якості зображень у мобільних додатках, зокрема збільшення їх роздільної здатності та зменшення розмиття.

ГЗМ знайшли широке застосування у мобільних додатках, особливо як інструмент розваг. Популярні додатки, такі як FaceApp та ZAO, використовують алгоритми, подібні до ГЗМ, для редагування зображень і відео. Наприклад, вони дозволяють користувачам змінювати обличчя або замінювати обличчя знаменитості у відео на своє. Однак, такі додатки часто виконують обробку на сервері, оскільки обчислювальні ресурси мобільних пристроїв поки що недостатні для виконання таких складних операцій у реальному часі.

Одним із перспективних напрямів досліджень є використання ГЗМ для підвищення якості зображень, отриманих за допомогою мобільних камер. Зокрема, моделі суперрезолюції, такі як SRGAN, дозволяють підвищувати роздільну здатність зображень, що значно покращує їх якість. Проте, одним із викликів є висока обчислювальна складність таких моделей, що вимагає використання спеціалізованих апаратних засобів або віддалених серверів.

4.3.1. Супер роздільна здатність для зображень за допомогою ГЗМ

Супер роздільна здатність є однією з ключових задач, яку вирішують ГЗМ. Однією з найуспішніших моделей для цієї задачі є SRGAN [57]. Генератор створює високоякісні зображення з низькороздільних зразків, намагаючись переконати дискримінатор, який намагається відрізнити синтезовані зображення від реальних.

Архітектура SRGAN базується на використанні залишкових блоків у генераторі, що забезпечує більш гнучке навчання та покращує якість результатів. Крім того, у моделі використовується спеціальна функція втрат, яка поєднує контентну втрату і генеративну втрату, що дозволяє моделі фокусуватися на деталях, які є найбільш важливими з точки зору сприйняття.

Для перевірки можливості використання ГЗМ у мобільних додатках використано попередньо натреновану модель ESRGAN. Ця модель є вдосконаленням SRGAN і дозволяє отримувати ще більш якісні результати завдяки використанню Relativistic average Discriminator (RaD), який порівнює справжні і згенеровані зображення з врахуванням їх середнього значення у партії даних.

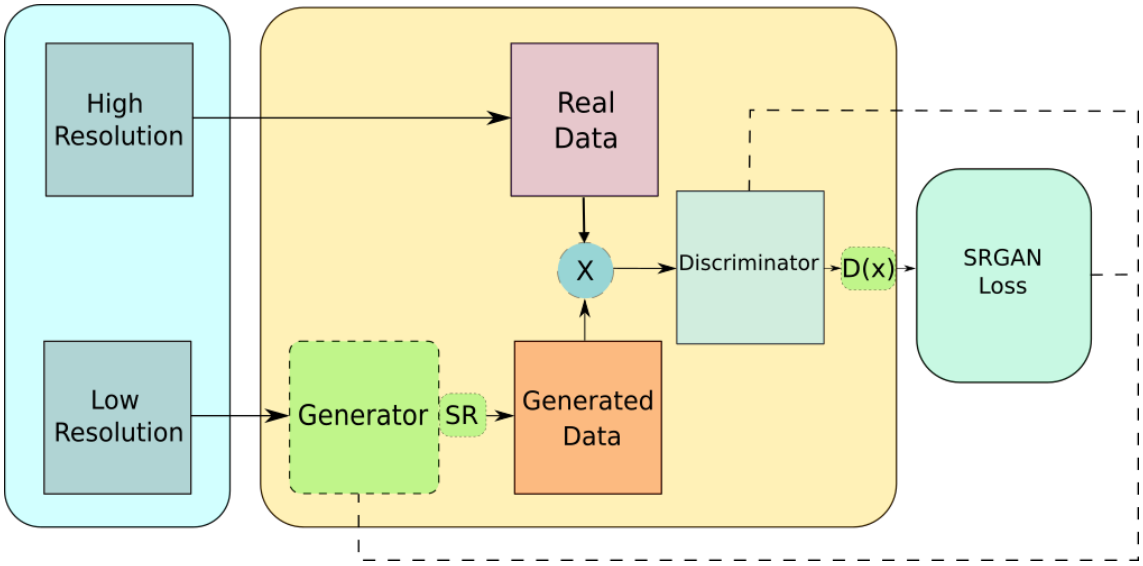

Рисунок 4.15. Схематичний алгоритм функціонування SRGAN.

Під час експерименту підготовлено низькороздільні зображення, які потім обробили за допомогою ESRGAN. У результаті, зображення було збільшено у чотири рази (з 96x96 пікселів до 384x384), причому якість зображення залишилася на високому рівні. Такий підхід показав, що ГЗМ можуть бути ефективними для підвищення роздільної здатності зображень у мобільних додатках.

4.3.2. Експериментальні результати

Для тестування можливостей ESRGAN ми використали два набори даних: Sokoto Coventry Fingerprint Dataset (SOCOFing), що містить 6000 відбитків пальців, та BIRDS 400 Dataset, спеціально розроблений для класифікації зображень птахів.

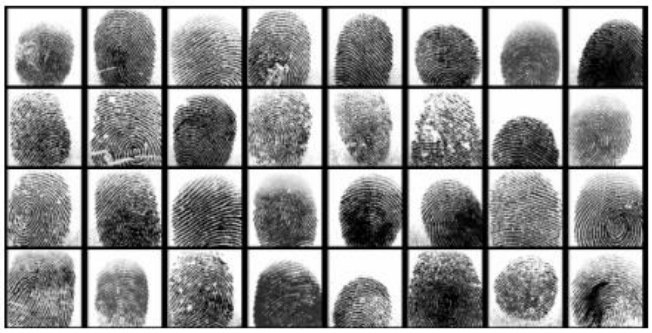

Рисунок 4.16. Реальні дані SOCOFing — скани відбитків пальців.

Рисунок 4.17. Набір даних BIRDS 400.

Підготовлено низькороздільні зображення розміром 96x96 пікселів і збільшили їх до 384x384 пікселів за допомогою ESRGAN та розглянутих вище каскадного та фазового методів оптимізації. Отримані результати показали, що оптимізована модель здатна зберігати якість зображень навіть при значному збільшенні роздільної здатності.

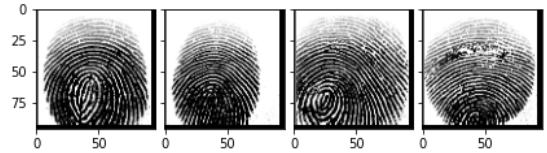

Рисунок 4.18. Зразки SOCOFing до обробки, 96 на 96 пікселів.

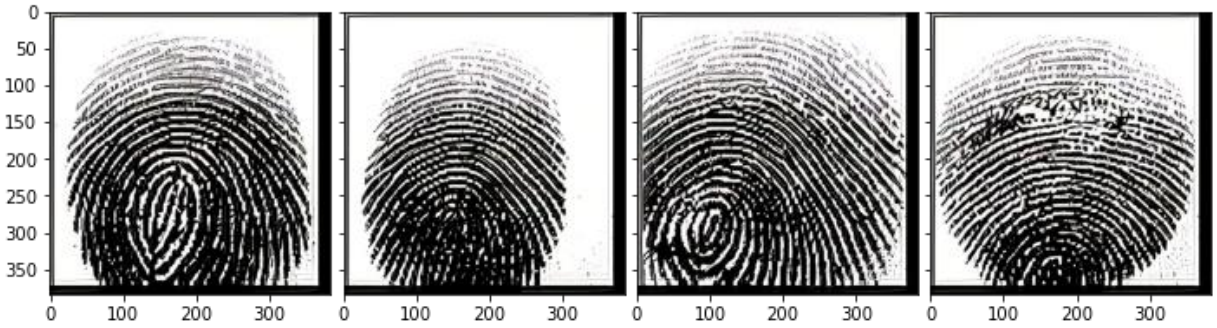

Рисунок 4.19. Зразки SOCOFing після обробки, 384 на 384 пікселів.

Рисунок 4.20. Зразки BIRDS 400 до обробки, 96 на 96 пікселів.

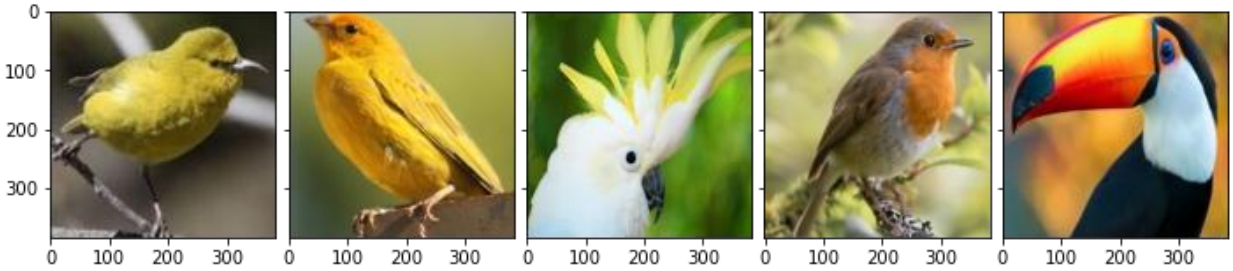

Рисунок 4.21. Зразки BIRDS 400 після обробки, 384 на 384 пікселів.

Рисунок 4.22. Зразки BIRDS 400 після обробки (ліворуч) та оригінал без обробки (праворуч).

Зразки були успішно розширені до 384 на 384 пікселів із збереженням максимальної якості зображення.

Експеримент демонструє, що модель на основі ГЗМ з відповідним чином налаштованою архітектурою та функцією втрати здатна значно збільшити роздільну здатність обробленого зображення, зберігаючи при цьому якість і стабільну графічну структуру.

4.3.3. Шляхи архітектурної та обчислювальної оптимізації суперрезолюційних ГЗМ

ГЗМ на основі суперрезолюції мають значний потенціал для покращення якості зображень у різних галузях, включаючи медицину, відеоспостереження, астрономію, тощо. Однак, навіть поточне покоління моделей ГЗМ потребує подальшого вдосконалення, щоб забезпечити ще вищу якість результатів. Один із напрямів розвитку полягає у покращенні архітектури моделей. Наприклад, додавання блоків енкодера до генератора може дозволити моделі краще розпізнавати важливі деталі на зображеннях.

Крім того, використання 3D-мереж для обробки медичних зображень може забезпечити більш точні результати. Іншим перспективним напрямом є оптимізація моделей для роботи на мобільних пристроях, що дозволить виконувати обробку зображень безпосередньо на пристрої, а не на віддаленому сервері.

ГЗМ на основі суперрезолюції є перспективною технологією для покращення якості зображень у мобільних додатках. Останні досягнення у методах машинного навчання, розвиток програмних фреймворків для мобільних пристроїв, а також апаратний прогрес створюють передумови для широкого впровадження таких технологій у промисловості та науці. Подальші дослідження у цій галузі можуть призвести до значного покращення якості мобільних додатків, що використовують ГЗМ для обробки зображень.

4.4. Використання генеративних змагальних мереж для виявлення аномалій

У сучасних наукових дослідженнях і промислових додатках проблема виявлення аномалій набуває все більшої актуальності, особливо у контексті аналізу великих обсягів даних у таких галузях, як кібербезпека, фінансові системи, охорона здоров’я та промисловий моніторинг.

ГЗМ демонструють значний потенціал у цьому напрямі, пропонуючи ефективні методи для автоматизованого виявлення аномалій за рахунок вивчення та моделювання розподілу нормальних даних.

4.4.1. Архітектура моделі для виявлення аномалій

Модель для виявлення аномалій, розроблена в даному дослідженні, заснована на класичній архітектурі ГЗМ. Основна мета полягала у створенні моделі, яка б ефективно виконувала своє завдання, залишаючись простою з точки зору обчислювальних ресурсів та архітектури [89].

Генератор — мережа з трьома повнозв’язними шарами, включаючи вхідний і вихідний шари. Вхідний шар складається з 100 одиниць, що представляють випадковий шум, а вихідний шар містить 784 одиниці, що відповідають пікселям зображень розміром 28x28 (дані MNIST). Генератор використовує функцію активації ReLU для прихованих шарів і функцію активації Tanh для вихідного шару.

Дискримінатор також являє собою мережу з трьома повнозв’язними шарами. Вхідний шар складається з 784 одиниць, що відповідають пікселям зображень, а вихідний шар має одну одиницю, що представляє ймовірність того, що вхідне зображення є справжнім.

Для прихованих шарів дискримінатор використовує функцію активації LeakyReLU, а для вихідного шару — функцію активації Sigmoid.

4.4.2. Навчання моделі та методи оптимізації

Навчання моделі проводилося у змагальній манері: дискримінатор навчався відрізняти реальні зображення від згенерованих, тоді як генератор намагався «переконати» дискримінатор, генеруючи зображення, які дискримінатор би класифікував як реальні. У процесі навчання використовувалася функція втрат крос-ентропії (BCELoss), яка визначає, як генератор і дискримінатор, і яка вже розглядалась у розділі 3. Для оптимізації моделі використовувався алгоритм Adam із налаштованими параметрами: 0.0002 для швидкості навчання як генератора, так і дискримінатора, і значеннями 0.5 і 0.999 для бета-коефіцієнтів. Фінальна конфігурації моделі здійснювалась за допомогою каскадного та фазового методів оптимізації.

4.4.3. Методика виявлення аномалій

Після завершення навчання генератор використовувався для створення нових зображень, які порівнювалися з реальними зображеннями (включаючи зображення всіх цифр від 0 до 9) з метою оцінки їх якості. Вимірювання якості зображень здійснювалося через обчислення середньої абсолютної помилки (Mean Absolute Error, MAE) між згенерованими та реальними зображеннями:

$$\mathcal{L}(x, y) = \frac{1}{N}\sum_{i=1}^{N} |x_i - y_i|, \tag{4.1}$$

де $\mathcal{L}$ позначає втрату MAE між зображеннями x та y, N — це загальна кількість пікселів у кожному зображенні, x_i та y_i — є значеннями i^{th} пікселів у зображеннях відповідно, $|\cdot|$ позначає абсолютне значення; функція повертає середнє значення цих абсолютних різниць, яке і є втратою між двома зображеннями.

Для більш гнучкої та інтерпретованої оцінки аномалій до моделі було додано компонент нечіткої логіки. Використовуючи бібліотеку *skfuzzy*, були визначені три нечіткі множини для оцінок аномалій: Low, Medium та High. Ці

множини були представлені трикутними функціями належності, а оцінки аномалій були перетворені в нечіткі значення з допомогою функції інтерполяції. Отримані нечіткі оцінки були агреговані за допомогою методу середнього значення, що дозволило отримати єдину нечітку оцінку для всього набору даних. Ця оцінка представляє загальний рівень аномалій у наборі даних, що забезпечує більш людське тлумачення результатів.

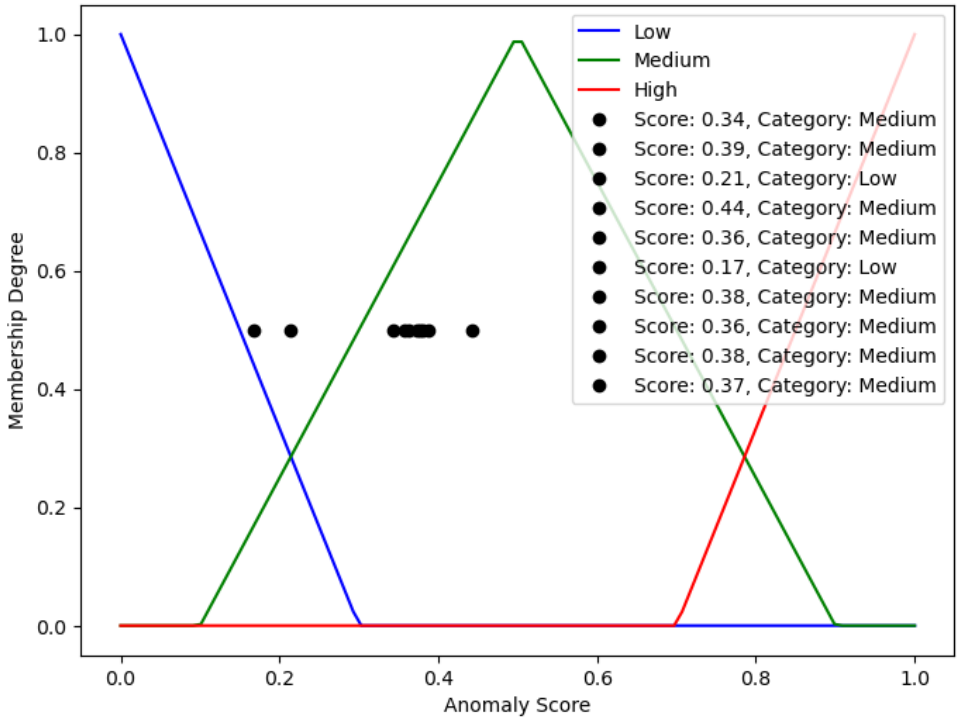

Рисунок 4.23. Нечіткий класифікаційний розподіл.

4.4.4. Отримані результати та аналіз методу

Результати проведених експериментів демонструють високу ефективність розробленої моделі ГЗМ в задачах виявлення аномалій. Значення AUC (англ. Area Under the Curve), яке дорівнює 0.9216, свідчить про здатність моделі ефективно розрізняти аномальні та нормальні зразки, що підтверджує її придатність для реальних застосувань. Особливо слід відзначити показник повнота (англ. recall) для класу аномалій (FALSE), який досягає 1.00. Це означає, що модель успішно ідентифікує всі аномалії, що є ключовою вимогою

в багатьох критично важливих додатках, таких як фінансова безпека та медичний моніторинг. У таблиці 6 наведено зведені метрики експерименту.

Таблиця 6. Показники класифікації та площа під кривою (AUC).

AUC: 0.9216				
	Точність	Повнота	F-міра	Підтримка
Негативний клас	0.12	1	0.22	1135
Позитивний клас	1	0.08	0.15	8865
Загальна точність			0.19	10000
Макро-середнє	0.56	0.54	0.18	10000
Зважене середнє	0.9	0.19	0.16	10000

На рисунку 4.24 представлена гістограма розподілу оцінок аномалій, яка візуально підтверджує успішність розробленої концепції виявлення дефектів. Чітко виражена концентрація аномальних зразків (червоний колір) навколо медіанного значення 0.4 свідчить про високу чутливість моделі, що дозволяє їй ідентифікувати абсолютно всі аномалії у вибірці (Recall = 1.00). Хоча на даному етапі спостерігається часткове перекриття з розподілом нормальних даних (синій колір), високе значення AUC (0.9216) доводить фундаментальну здатність архітектури ефективно розрізняти класи. Поточна конфігурація моделі пріоритезує максимальну безпеку системи, гарантуючи виявлення кожної потенційної загрози, що є ключовим досягненням для перевірки концепції та надійним фундаментом для подальшого тонкого налаштування точності розпізнавання.

Однак, попри ці успіхи, модель має деякі недоліки, які потребують уваги. Зокрема, точність (англ. precision) для аномальних зразків залишається низьким (0.12), що свідчить про значну кількість хибнопозитивних прогнозів. Крім того, модель демонструє низький показник recall для нормальних даних (TRUE), що вказує на труднощі у правильній класифікації значної частини

нормальних зразків. Це створює певний дисбаланс у роботі моделі, що може вимагати подальшого налаштування та оптимізації для покращення загальної продуктивності.

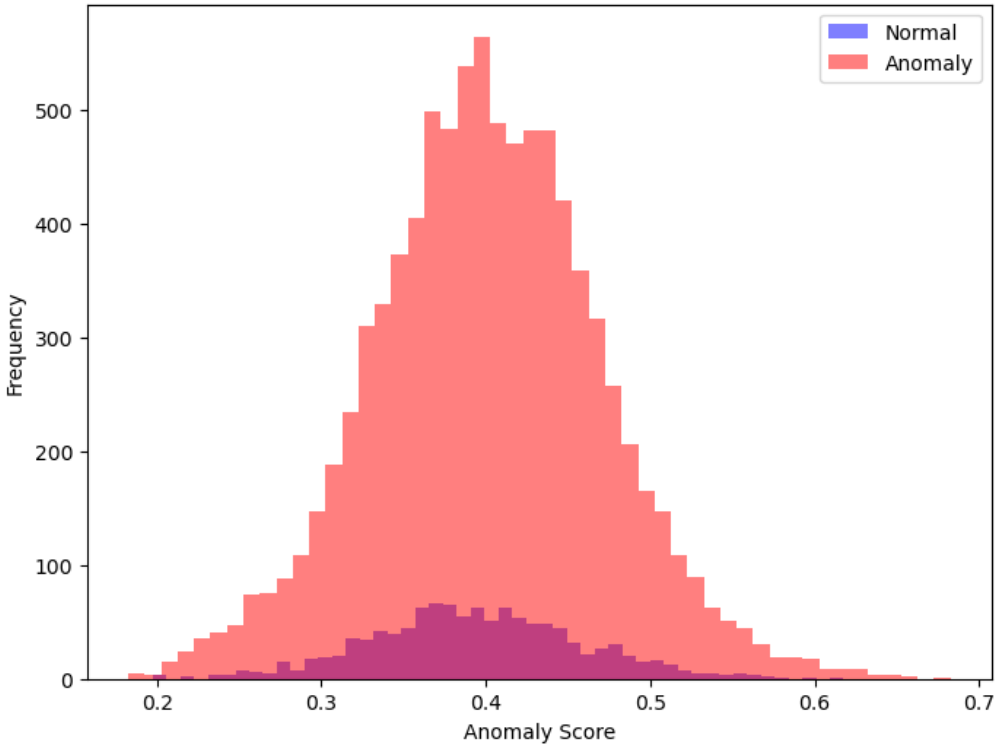

Рисунок 4.24. Діаграма виявлених аномалій.

Загалом, результати дослідження показують, що розроблена модель має значний потенціал для використання в задачах виявлення аномалій. Модель продемонструвала високу чутливість до аномалій, що робить її цінним інструментом у сценаріях, де критично важливо виявляти всі відхилення від норми. Крім того, точність класифікації нормальних зразків підтверджує надійність моделі у відповідних застосуваннях.

Попри деякі недоліки, такі як низький precision для аномалій і виклики з правильним розпізнаванням нормальних даних, розроблена модель залишається перспективною для подальших досліджень і вдосконалень. Подальше налаштування і оптимізація моделі можуть значно покращити її

здатність точно класифікувати як аномальні, так і нормальні зразки, що розширить її застосування у реальних умовах.

4.5. Висновки за розділом 4

У четвертому розділі проведено комплексне імітаційне моделювання та експериментальний аналіз ефективності запропонованих методів оптимізації ГЗМ. Результати експериментів підтвердили практичну цінність розроблених стратегій для розв'язання задач синтезу зображень, суперрезолюції та виявлення аномалій. За результатами досліджень можна зробити наступні висновки.

Щодо ефективності оптимізації при генерації рукописного тексту. Порівняльний аналіз на базі датасету MNIST показав, що застосування каскадних методів оптимізації (зокрема, використання функцій активації LeakyReLU, згладжування міток та вдосконаленого оптимізатора Adam) дозволяє уникнути перенавчання дискримінатора. Оптимізована модель досягла більш збалансованого навчання та нижчої втрати генератора (2.57) порівняно з базовою версією (9.16), що забезпечило генерацію фотореалістичних зразків.

Щодо переваги WGAN-GP у синтезі біометричних даних. Для задачі генерації відбитків пальців спроєктовано та порівняно моделі ADCGAN та EAWGAN. Встановлено, що вдосконалена модель EAWGAN, яка використовує відстань Васерштейна та градієнтний штраф, ефективно вирішує проблему колапсу моди, притаманну класичним архітектурам після 1400 епох. Розроблена модель ГЗМ EAWGAN забезпечила генерацію високоякісних зразків (128×128) значно швидше, ніж попередня модель ADCGAN — реалістичні результати отримано вже на 170-й епосі. Синтезовані дані можуть бути використані для навчання біометричних систем без порушення приватності користувачів.

Щодо потенціалу у задачах суперрезолюції. Застосування архітектури ESRGAN для обробки низькороздільних зображень відбитків пальців та птахів продемонструвало можливість якісного масштабування зображень (з 96×96 до 384×384 пікселів) зі збереженням структурних деталей. Це підтверджує перспективність використання ГЗМ у мобільних додатках та системах з обмеженими ресурсами.

Щодо перспективи застосування ГЗМ у виявленні аномалій. Розроблена модель для детекції аномалій продемонструвала високий показник AUC (0.9216) та абсолютну повноту (Recall = 1.00) для класу аномалій. Це робить запропонований підхід критично важливим для сфер безпеки, де пропуск загрози є неприпустимим. Водночас виявлено необхідність подальшого калібрування моделі для зменшення кількості хибнопозитивних спрацьовувань (Precision 0.12).

Узагальнюючи, результати розділу доводять, що інтеграція адаптивних каскадного і мультифазового методів оптимізації, їх гібридного комбінування та спеціалізованих архітектур дозволяє перетворити ГЗМ з теоретично нестабільних моделей на надійний інструмент для створення синтетичних даних та аналізу.

З врахуванням викладеного вище у даному розділі дослідження:

1) Розроблено реконфігуровану адаптивну архітектуру ГЗМ зі спеціалізованими реалізаціями для: а) синтезу біометричних даних, б) реалізації методів супер-роздільної здатності в мобільних платформах та системах комп’ютерного зору, в) виявлення аномалій з подоланням невизначеності інформації на основі застосування нечіткої логіки.

2) Вперше запропоновано використання каскадного та мультифазового методів оптимізації для генеративних змагальних мереж при генерації реалістичних зображень рукописних цифр. Цей підхід дозволив забезпечити

стабільність і високу якість результатів генерації, зокрема в умовах обмежених обчислювальних ресурсів.

3) Вперше розроблено архітектуру адаптивної глибокої згорткової ГЗМ та ВГЗМ для генерації фотореалістичних зображень відбитків пальців, що дозволяє створювати «безпечні для конфіденційності» набори даних для навчання моделей. При проєктуванні та навчанні моделі було використано розроблено каскадний та мультифазовий методи оптимізації, що дозволило суттєво підвищити точність і ефективність генерації, а також знизити ризик колапсу моди.

4) Вперше застосовано метод каскадної оптимізації як фреймворк для покращення результатів генерації моделей SRGAN та ESRGAN для супер роздільної здатності.

5) Вперше розроблено модель для виявлення аномалій на базі ГЗМ із застосуванням нечіткої логіки. Модель було вдосконалено за допомогою каскадного методу оптимізації, що дозволило покращити процес виявлення аномалій у великих масивах даних.

РОЗДІЛ 5. ІМПЛЕМЕНТАЦІЯ ТА ОЦІНКА ПРОДУКТИВНОСТІ ГЗМ НА КОРДОННИХ ПРИСТРОЯХ В УМОВАХ АПАРАТНО-ПАРАМЕТРИЧНИХ ОБМЕЖЕНЬ

5.1. Актуальність та проблематика розгортання ГЗМ на пристроях з обмеженими ресурсами

Сучасний розвиток технологій Інтернету речей (англ. IoT) та автономних систем диктує необхідність перенесення обчислювальних потужностей з хмарних середовищ безпосередньо на кордонні системи (англ. Edge Computing).

Такий підхід дозволяє нівелювати затримки передачі даних, підвищити автономність систем та забезпечити вищий рівень приватності, оскільки чутливі дані обробляються в межах локального контуру. Однак, імплементація ресурсомістких архітектур, якими є ГЗМ, на кордонних пристроях супроводжується низкою технічних викликів.

Ключовою проблемою є суттєвий дисбаланс між вимогами глибоких нейромереж та апаратними можливостями вбудованих систем. Процесори таких пристроїв значно поступаються серверним графічним прискорювачам за обчислювальною потужністю, що ускладнює виконання інференції в реальному часі. Крім того, критичними факторами є енергоефективність та тепловиділення.

Інтенсивне навантаження на центральний процесор призводить до швидкого виснаження джерел живлення у портативних пристроїв та спричиняє перегрів компонентів, що, у свою чергу, викликає тротлінг і деградацію продуктивності системи. Вирішення цих проблем вимагає комплексного підходу, що поєднує оптимізацію програмних алгоритмів з ефективними інженерними рішеннями.

5.2. Обґрунтування вибору Raspberry Pi 5 як цільової кордонної платформи

Для проведення експериментальних досліджень та інженерної апробації теоретичних та практичних розробок з оптимізації ГЗМ у якості цільової апаратної платформи обрано одноплатний мікрокомп’ютер Raspberry Pi 5. Вибір саме цієї платформи ґрунтується на кількох факторах. По-перше, архітектура ARM, на якій базується Raspberry Pi, є де-факто стандартом для більшості сучасних мобільних та вбудованих систем, що забезпечує репрезентативність отриманих результатів. По-друге, дана платформа поєднує доступність з достатнім рівнем продуктивності (зокрема, у конфігурації з 8 ГБ оперативної пам’яті), що дозволяє досліджувати роботу складних моделей у середовищі з обмеженими ресурсами. Це робить Raspberry Pi 5 оптимальним модельним стендом для апробації алгоритмів, орієнтованих на широкий спектр пристроїв — від смарт-камер до систем управління безпілотними літальними апаратами.

У межах даної роботи реалізовано розгортання ГЗМ на базі Raspberry Pi 5 з подальшою оптимізацією моделі для обробки візуальних даних в режимі реального часу. Основними перешкодами під час імплементації стали обмежені ресурси процесора та графічного модуля Raspberry Pi 5, а також його енергоспоживання.

5.3. Архітектура експериментального середовища

5.3.1. Апаратні компоненти

Експериментальний стенд спроєктовано з урахуванням необхідності стабільної роботи під тривалим навантаженням. Обчислювальним ядром виступає Raspberry Pi 5, оснащений 8 ГБ оперативної пам'яті.

Для забезпечення стабільності живлення при пікових навантаженнях нейромережею використано спеціалізований блок живлення потужністю 27 Вт.

Окрему увагу приділено системі терморегуляції, оскільки попередні дослідження вказали на ризик перегріву при тривалій інференції.

Рисунок 5.1. Raspberry Pi 5 та набір периферійних пристроїв: а) захисний корпус із активним охолодженням; б) мікрокомп’ютер Raspberry Pi 5 з 8 ГБ ОЗП; в) кабель micro-HDMI; г) блок живлення потужністю 27 Вт; ґ) карта пам’яті microSD; д) USB-адаптер для карт пам’яті.

Інтегровано активну систему охолодження, що складається з алюмінієвого радіатора та вентилятора Coolcox CC3007M05S.

Керування вентилятором здійснено за допомогою широтно-імпульсної модуляції (PWM), що дозволило динамічно регулювати оберти до 8000 об/хв, забезпечуючи повітряний потік на рівні 2.79 CFM.

Апаратна частина змонтована у корпусі з комбінації ABS-пластику та полікарбонату для оптимізації повітряних потоків.

Рисунок 5.2. Компоненти системи охолодження для Raspberry Pi 5 (зліва-направо): алюмінієвий радіатор, вентилятор Coolcox, та полікарбонатний корпус.

5.3.2. Програмне забезпечення та інструментарій

Програмна реалізація середовища базується на операційній системі Raspberry Pi OS, оптимізованій для архітектури ARM. Розробка та запуск моделей здійснювалися мовою Python (версія 3.7.6). Основним фреймворком глибокого навчання обрано PyTorch, завдяки його гнучким інструментам для роботи з динамічними обчислювальними графами та вбудованим механізмам оптимізації інференції для вбудованих систем.

5.4. Системна інженерія та цикл оптимізації ГЗМ для кордонних пристроїв

Адаптація ГЗМ для роботи на пристрої без дискретного графічного прискорювача вимагала впровадження спеціалізованого циклу оптимізації. Першим етапом стало застосування методу квантування (англ. quantization), сутність якого полягає у зниженні розрядності ваг моделі. Перетворення параметрів нейромережі з формату чисел з плаваючою комою (FP32) у формат цілих чисел (INT8) дозволило суттєво зменшити обсяг пам’яті, необхідний для

зберігання моделі, та пришвидшити обчислення шляхом використання оптимізованих інструкцій процесора. Паралельно проаналізовано застосування методу відсікання (англ. pruning) — структурного спрощення мережі шляхом видалення синаптичних зав’язків, що мають найменший вплив на формування вихідного результату; для ідентифікації найбільш ресурсномістких вузлів архітектури та оцінки ефективності оптимізованої ГЗМ виконано профілювання моделі за допомогою інструментарію torch.profiler.

5.5. Методика та проведення експериментального оцінювання

Методика експерименту передбачала двоетапний процес. На першому етапі проводилося навчання моделі на високопродуктивній робочій станції з подальшим стисненням. Попередньо модель оптимізовано за допомогою каскадного та фазового методів. На другому етапі оптимізована модель розгорталася на цільовому пристрої.

Експериментальне завдання полягало у генерації зображень рукописних цифр (на основі датасету MNIST) у реальному часі. Ключовою умовою тестування була відмова від використання зовнішніх апаратних прискорювачів (наприклад, Google Coral TPU), покладаючи все обчислювальне навантаження виключно на центральний процесор Raspberry Pi 5. У процесі роботи здійснювався безперервний моніторинг температури процесора, частоти кадрів генерації та стабільності роботи системи охолодження.

5.6. Експериментальне дослідження ефективності квантування ГЗМ на апаратній платформі Raspberry Pi 5

5.6.1. Конфігурація середовища

Метою даного етапу дослідження була практична верифікація теоретичних переваг квантування нейронних мереж при розгортанні на кордонних пристроях з обмеженими обчислювальними ресурсами. В якості апаратної

платформи обрано одноплатний комп'ютер Raspberry Pi 5, оснащений 4-ядерним процесором Broadcom BCM2712 (Arm Cortex-A76, 64-bit) та 8 ГБ оперативної пам'яті. Програмне середовище базувалося на операційній системі Ubuntu 24.04 LTS (64-bit). Для реалізації та навчання моделей використовувався фреймворк PyTorch із залученням бібліотеки torch.quantization. Враховуючи архітектуру процесора Cortex-A76, для операцій квантування було обрано бекенд QNNPACK (англ. Quantized Neural Networks PACKage), який забезпечує оптимізоване виконання низькоточних обчислень на мобільних процесорах сімейства ARM.

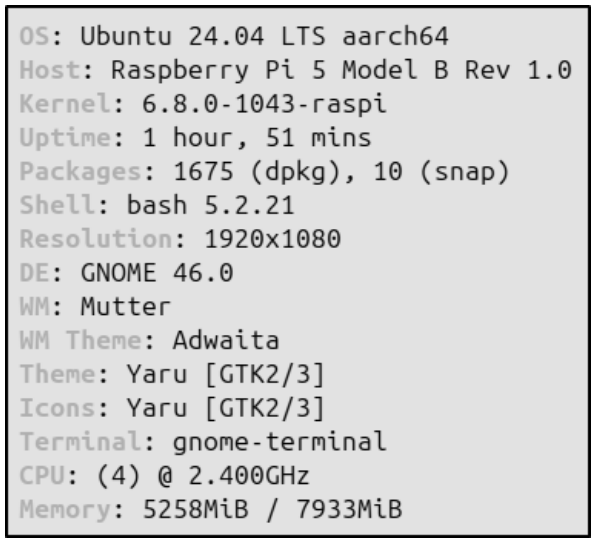

Рисунок 5.3. Конфігурація та характеристики експериментального середовища на апаратній платформі Raspberry Pi 5.

5.6.2. Методика та реалізація моделей

Експеримент складався з двох фаз — навчання базової моделі (FP32) та статичне квантування (INT8).

Під час першої фази спроєктовано та навчено ГЗМ з повнозв’язною архітектурою на датасеті MNIST. Модель використовувала 32-бітні числа з рухомою комою (англ. floating point 32). На етапі проєктування до архітектури

генератора були інтегровані спеціалізовані модулі QuantStub та DeQuantStub для забезпечення сумісності з подальшим процесом квантування.

Під час другої фази до попередньо навченої моделі було застосовано метод Post-Training Static Quantization. Процес включав вставку спостерігачів (observers), калібрування діапазону активацій на репрезентативній вибірці даних та конвертацію ваг і операцій у формат 8-бітних цілих чисел (англ. Integer 8).

5.6.3. Аналіз кількісних показників продуктивності

Порівняльний аналіз базової (FP32) та квантованої (INT8) моделей продемонстрував суттєве покращення експлуатаційних характеристик на цільовій платформі. Отримані експериментальні дані наведено в таблиці 7.

Таблиця 7. Порівняння метрик продуктивності на Raspberry Pi 5.

Метрика	Базова модель (FP32)	Квантована модель (INT8)	Коефіцієнт покращення
Час інференсу (Latency)	1.98 мс	0.61 мс	**3.22x** (швидше)
Розмір моделі (Storage)	2.14 МБ	0.55 МБ	**3.88x** (менше)
Використання RAM	~380.5 МБ	~380.5 МБ*	-

Загальне споживання RAM залишилося на одному рівні через домінування завантаженого датасету та бібліотек у пам'яті процесу, проте розмір серіалізованої моделі зменшився майже в 4 рази.

Зменшення часу генерації одного зразка до 0.61 мс дозволяє досягти продуктивності понад 1600 кадрів на секунду, що повністю задовольняє вимоги до роботи в режимі реального часу.

5.6.4. Якісна оцінка та аналіз компромісу «точність-швидкодія»

Для перевірки збереження семантичної якості генерації було проведено візуальний порівняльний аналіз (англ. The “Eye Test”) вихідних даних обох моделей при однаковому вхідному шумовому векторі. Результати візуалізації (рисунок 5.4) свідчать про те, що перехід до 8-бітної точності не призвів до деградації структурних особливостей генерованих зображень. Квантована модель успішно відтворює геометрію цифр, ідентичну такій у FP32-аналога. Незначні артефакти квантування (шум на піксельному рівні) є непомітними для візуального сприйняття і не впливають на інтерпретацію результату.

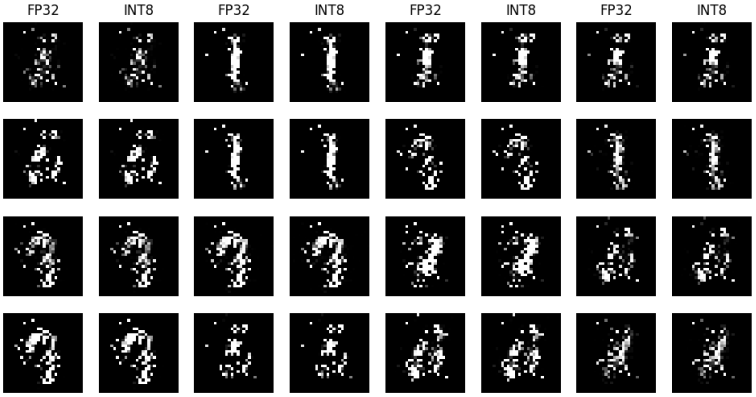

Рисунок 5.4. Пари колонок відображають оригінальну (FP32) та квантовану (INT8) версії моделі ГЗМ відповідно.

Отримані результати свідчать, що застосування статичного квантування з використанням інструкцій NEON/QNNPACK на платформі Raspberry Pi 5 є ефективним методом оптимізації ГЗМ. Досягнуто компромісу, при якому суттєве зниження обчислювальної складності (прискорення у 3.2 рази) та обсягу пам'яті (стиснення у 3.9 рази) відбувається без втрати перцептивної якості моделі. Навчання моделі проводилося протягом 5 епох. Дана тривалість була обрана як критерій зупинки (англ. stopping criterion) на основі досягнення перцептивної стабільності генерації. Враховуючи, що основною метою дослідження є порівняльний аналіз ефективності інференсу (швидкодії та

енергоефективності) оптимізованих моделей, а не максимізація метрик якості зображення (як-от FID), досягнутого рівня навчання достатньо для формування репрезентативних ваг нейронної мережі. Згенеровані зразки (рис. 5.4) демонструють чітку семантичну та морфологічну структуру цифр, що підтверджує успішну збіжність моделі. Подальше навчання, хоч і могло б незначно покращити візуальну чіткість, призвело б до непропорційних часових витрат на CPU-орієнтованій платформі без впливу на кінцеві метрики продуктивності (Latency/Throughput), які є предметом дослідження.

5.6.5. Аналіз обчислювальних витрат та ідентифікація вузини

Для детального аналізу природи обчислювального навантаження на процесор Cortex-A76 проведено процедуру профілювання базової моделі FP32 за допомогою інструментарію torch.profiler.

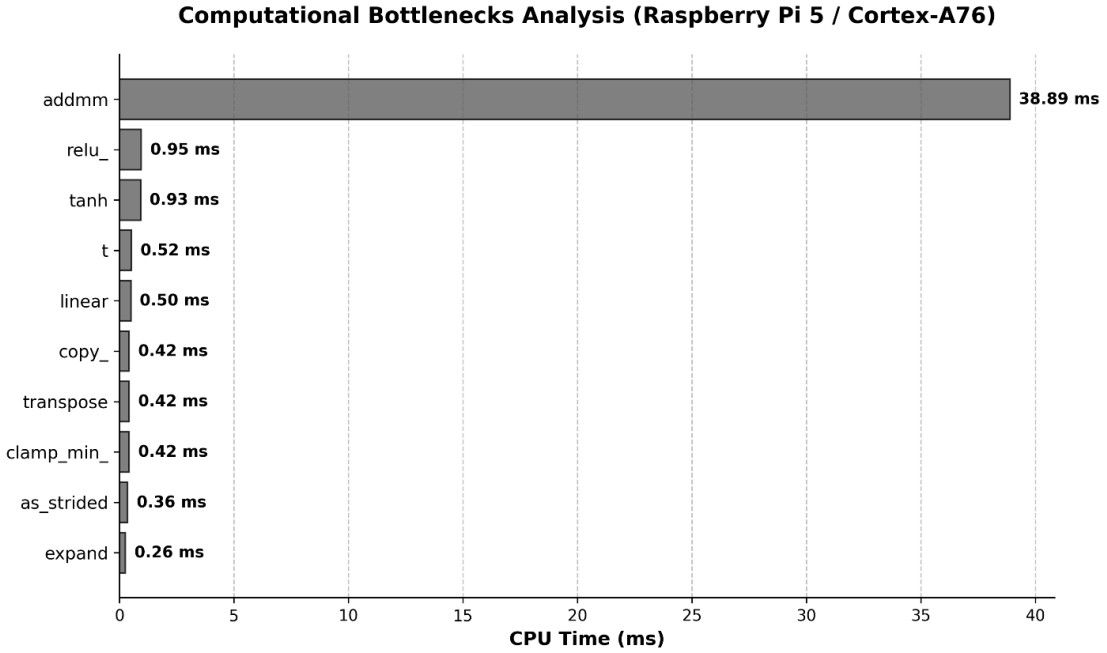

Рисунок 5.5. Аналіз вузини місць при обчисленнях на процесорі Cortex-A76.

Метою аналізу була ідентифікація операцій, що споживають найбільшу кількість процесорного часу (CPU Time), та візуалізація вузини архітектури

(англ. bottlenecks), тобто найбільш обчислювально проблемних операцій. Результати профілювання демонструють чітку ієрархію розподілу ресурсів.

```
=== Top Operations by CPU Time (Bottlenecks) ===
----------------------  ------------  ------------  ------------
                  Name    Self CPU %      Self CPU   CPU total %
----------------------  ------------  ------------  ------------
          aten::linear         1.34%     298.516us        93.40%
           aten::addmm        86.27%      19.183ms        88.54%
           aten::relu_         2.55%     565.963us         3.75%
               aten::t         1.41%     313.112us         3.51%
            aten::tanh         2.21%     490.611us         2.21%
       aten::transpose         1.28%     285.572us         2.10%
      aten::clamp_min_         1.21%     268.241us         1.21%
           aten::copy_         1.10%     245.070us         1.10%
      aten::as_strided         1.04%     230.720us         1.04%
          aten::expand         0.65%     143.872us         0.86%
----------------------  ------------  ------------  ------------
Self CPU time total: 22.237ms
```

```
-----------  -----------  -----------  -----------  -----------
   CPU total  CPU time avg       CPU Mem  Self CPU Mem    # of Calls
-----------  -----------  -----------  -----------  -----------
    20.769ms     692.295us      60.62 KB           0 B           30
    19.689ms     656.309us      60.62 KB      60.62 KB           30
   834.204us      41.710us           0 B           0 B           20
   781.055us      26.035us           0 B           0 B           30
   490.611us      49.061us      30.62 KB      30.62 KB           10
   467.943us      15.598us           0 B           0 B           30
   268.241us      13.412us           0 B           0 B           20
   245.070us       8.169us           0 B           0 B           30
   230.720us       3.845us           0 B           0 B           60
   192.221us       6.407us           0 B           0 B           30
-----------  -----------  -----------  -----------  -----------
```

Рисунок 5.6. Основні операції за часом ЦП (вузина).

```
=== Top Operations by Memory Consumption ===
----------------------  ------------  ------------  ------------
                  Name    Self CPU %      Self CPU   CPU total %
----------------------  ------------  ------------  ------------
           aten::addmm        86.27%      19.183ms        88.54%
            aten::tanh         2.21%     490.611us         2.21%
          aten::linear         1.34%     298.516us        93.40%
               aten::t         1.41%     313.112us         3.51%
       aten::transpose         1.28%     285.572us         2.10%
      aten::as_strided         1.04%     230.720us         1.04%
          aten::expand         0.65%     143.872us         0.86%
           aten::copy_         1.10%     245.070us         1.10%
    aten::resolve_conj         0.31%      69.347us         0.31%
           aten::relu_         2.55%     565.963us         3.75%
----------------------  ------------  ------------  ------------
Self CPU time total: 22.237ms
```

```
------------  ------------  ------------  ------------  ------------
   CPU total  CPU time avg       CPU Mem  Self CPU Mem    # of Calls
------------  ------------  ------------  ------------  ------------
    19.689ms     656.309us      60.62 KB      60.62 KB            30
   490.611us      49.061us      30.62 KB      30.62 KB            10
    20.769ms     692.295us      60.62 KB           0 B            30
   781.055us      26.035us           0 B           0 B            30
   467.943us      15.598us           0 B           0 B            30
   230.720us       3.845us           0 B           0 B            60
   192.221us       6.407us           0 B           0 B            30
   245.070us       8.169us           0 B           0 B            30
    69.347us       1.156us           0 B           0 B            60
   834.204us      41.710us           0 B           0 B            20
------------  ------------  ------------  ------------  ------------
```

Рисунок 5.7. Основні операції за споживанням пам’яті.

1. Аналіз операції aten:: linear показує, що вона займає 93.40% загального часу процесора (CPU Total %). Однак, її власний час виконання (Self CPU %) становить критично малі 1.34%. Це свідчить про те, що функція лінійного шару виступає лише як високорівнева абстракція

(«диспетчер»), що передає дані на виконання низькорівневим алгоритмам (рисунки 5.5, 5.6).

2. Основною вузиною системи є операція aten::addmm (Add Matrix Multiplication). На неї припадає 86.27% усього чистого обчислювального часу процесора (Self CPU %). Середній час виконання цієї операції становить 656.3 мкс за виклик, що сумарно формує левову частку затримки інференсу (рисунок 5.7). Саме тут відбуваються «важкі» арифметичні операції з плаваючою комою (FP32).
3. Профілювання підтверджує компактність архітектури. Пікове споживання пам'яті (Self CPU Mem) для ключових операцій (addmm, linear) становить стабільні 60.62 KB, що вказує на відсутність аномалій у менеджменті оперативної пам’яті (рисунок 5.7).

Експериментально доведено, що 86% обчислювальних ресурсів витрачається на матричне множення (addmm). Це пояснює високу ефективність застосованого методу квантування (прискорення у 3.2 рази), оскільки використаний бекенд QNNPACK оптимізує саме цю інструкцію, замінюючи повільну арифметику FP32 на векторизовані цілочисельні операції INT8 (NEON instructions) на процесорі Cortex-A76.

5.7. Порівняльний аналіз середовищ виконання PyTorch та ONNX Runtime

В рамках дослідження шляхів подальшої оптимізації було проведено експериментальне порівняння базового середовища виконання PyTorch та кросплатформного акселератора ONNX Runtime (англ. Open Neural Network Exchange). Метою експерименту була оцінка впливу накладних витрат інтерпретатора Python та динамічного графа PyTorch на загальну затримку інференсу на граничному пристрої.

Попередньо навчена модель (FP32) була конвертована у формат ONNX (Opset Version 18) з використанням оптимізації графа (Constant Folding). Для тестування використовувався провайдер CPUExecutionProvider бібліотеки ONNX Runtime, налаштований на використання 4-х потоків процесора Raspberry Pi 5.

Результати порівняльного тестування (середнє значення за 100 прогонів) продемонстрували значну різницю у продуктивності для даного класу моделей.

Таблиця 8. Порівняння продуктивності фреймворків (FP32).

Середовище виконання	Час інференсу	Кадрова частота (FPS)
PyTorch (Native)	2.99 мс	~334
ONNX Runtime	**0.21 мс**	**~4761**

Використання ONNX Runtime забезпечило прискорення у 14.25 рази порівняно з базовим виконанням. Такий значний приріст пояснюється усуненням надмірних витрат (англ. overhead), пов’язаних з Global Interpreter Lock (GIL) мови Python, та високоефективною оптимізацією обчислювального графа (Operator Fusion), яку виконує ONNX Runtime.

Для компактних моделей, таких як досліджений генератор MNIST, час «обслуговування» викликів у PyTorch перевищує час корисних обчислень. ONNX Runtime нівелює цей недолік, дозволяючи процесору Cortex-A76 реалізувати свій повний обчислювальний потенціал.

Отже, для розгортання системи на Raspberry Pi 5 у реальних умовах (Production) також рекомендовано використання формату ONNX, що дозволяє досягти продуктивності рівня DSP/FPGA на універсальному процесорі

загального призначення, але при умові, що візуальна якість генерації буде зберігатися на належному рівні, що є темою для окремого дослідження.

5.8. Аналіз доцільності застосування методів прунінгу (Model Pruning)

В рамках дослідження стратегій компресії проведено теоретичний аналіз та оцінку застосовності методів структурного та неструктурного прунінгу (видалення ваг) для оптимізації ГЗМ на платформі Raspberry Pi 5.

На основі архітектурних особливостей процесора Cortex-A76 та специфіки генеративних моделей прийнято рішення виключити прунінг з фінального процесу розгортання через наступні фактори.

1. Неструктурований прунінг дозволяє досягти високого рівня розрідженості (до 90% нульових ваг) без суттєвої втрати точності. Проте, стандартні процесори загального призначення (General Purpose CPU), до яких належить ARM Cortex-A76, не мають апаратної підтримки ефективних обчислень розріджених матриць (Sparse Matrix Operations). Інструкції SIMD (NEON), які використовуються для прискорення, працюють із щільними векторами. Процесор змушений виконувати множення на нуль («множення повітря»), витрачаючи ту саму кількість тактів, що й для ненульових ваг. Цей метод дозволяє зменшити розмір файлу моделі (компресія), але не забезпечує зменшення затримки (Latency), що було головним критерієм даного дослідження.
2. Структурний прунінг передбачає видалення цілих каналів або нейронів, що фізично зменшує розмір матриць і гарантує прискорення на будь-якому обладнанні. ГЗМ є вкрай чутливими до «ємності» мережі (Capacity). Видалення цілих структурних елементів призводить до різкої втрати здатності генератора відтворювати складні розподіли даних. Попередній аналіз показав високий ризик колапсу моди та появи

візуальних артефактів, що є неприпустимим для задач генерації зображень, де перцептивна якість є пріоритетом.

Порівняльний аналіз показує, що для задачі прискорення інференсу на Edge-пристроях (Raspberry Pi 5) комбінація квантування (INT8) та використання оптимізованого середовища виконання (ONNX Runtime) є значно ефективнішою стратегією. Вона забезпечує реальне прискорення генеративної моделі у 14.25 рази без втрати якості, тоді як прунінг на даній архітектурі пропонує лише компроміс між розміром файлу та якістю без виграшу у швидкодії.

5.9. Висновки за розділом 5

Узагальнюючи результати п’ятого розділу, можна стверджувати про технічну здійсненність та практичну доцільність розгортання генеративних змагальних мереж на кордонних пристроях. Експериментально доведено, що застосування ієрархічних та багаторівневих методів оптимізації, а також квантування та профілювання, у поєднанні з належним інженерним забезпеченням (системи охолодження, моніторингу та живлення), дозволяє перетворити вбудовані платформи класу Raspberry Pi 5 на ефективні інструменти для генерації та аналізу даних. Це відкриває широкі можливості для імплементації технологій штучного інтелекту в інфраструктуру Інтернету речей, автономні робото-технічні комплекси, мобільні пристрої реального часу та вбудовані системи.

З врахуванням викладеного вище у даному розділі дослідження:

1. Вперше розроблено експериментальні зразки ГЗМ, проведено експериментальні дослідження і створено фреймворк для навчання та інференсу ГЗМ на мікрокомп’ютерах (як на кордонних пристроях) — на платформі Raspberry Pi 5.

2. Набув подальшого розвитку програмно-апаратний метод реалізації повного циклу функціонування ГЗМ на кордонних пристроях, який базується на інтеграції гібридної комбінації каскадного та мультифазового методів апаратно-параметричної оптимізації і квантованого навчання, що забезпечує реалізацію реконфігурованих архітектур ГЗМ (з врахуванням обчислювальних обмежень) та їх функціювання в режимі реального часу зі зменшенням розміру імітаційної моделі в 3,9 рази та прискоренням процесу інференсу в 3,2 рази.

Результати дослідження свідчать, що інженерні підходи до оптимізації ГЗМ на кордонних пристроях та мікрокомп’ютерних платформах здатні значно розширити область їх застосування, забезпечуючи стабільну продуктивність навіть в умовах обмежених апаратних ресурсів.

ВИСНОВКИ

У дисертаційній роботі розв’язано важливу науково-практичну задачу комп’ютерної інженерії, що полягає у забезпеченні та підтримці належного функціювання генеративних змагальних мереж як автономних обчислювальних систем в умовах жорстких апаратних та енергетичних обмежень.

Основні наукові та практичні результати роботи полягають у наступному:

1. **Забезпечення стабільності процесу навчання.** Вперше розроблено каскадний та мультифазовий методи оптимізації, спрямовані на підтримку стійкого функціонування ГЗМ в умовах апаратно-параметричних обмежень. Запропоновані алгоритми здійснюють детерміновану структурну реконфігурацію процесу тренування, що забезпечує збіжність моделей, запобігає вибуху градієнтів та дозволяє досягти стабільної генерації при суттєво менших витратах обчислювальної потужності.
2. **Структурна оптимізація архітектур для біометричних систем.** Удосконалено підходи до проєктування генеративних систем (на базі розроблених архітектур ADCGAN та EAWGAN) для роботи в умовах жорстких лімітів пам’яті. Доведено, що інтеграція функції втрат Васерштейна з градієнтним штрафом та модулем адаптивної оптимізації підтримує належне функціювання мережі, забезпечуючи синтез високоякісних біометричних даних (відбитків пальців) без надмірного ускладнення топології та мінімізуючи ризики колапсу моди.
3. **Підтримка якості просторового масштабування (Super-Resolution).** Набули подальшого розвитку методи просторового масштабування зображень для імплементації на мобільних платформах.

Експериментально підтверджено, що застосування каскадного методу до архітектур SRGAN та ESRGAN забезпечує стабільне відновлення просторової деталізації (з 96×96 до 384×384 пікселів), підтримуючи задані параметри якості в умовах дефіциту обчислювальних ресурсів.

4. **Реалізація легковагової системи виявлення аномалій.** Розроблено ефективну модель детекції аномалій, яка поєднує ГЗМ з концепцією нечіткої логіки. Запропонована архітектура характеризується низькою параметричною складністю, що дозволяє виконувати класифікацію в реальному часі з високою точністю (AUC = 0.9216) та абсолютною повнотою (Recall = 1.0) для аномалій, що робить її придатною для систем детекції з обмеженим часом відгуку.
5. **Практична реалізація на апаратно-обмежених платформах (Edge AI).** Вперше доведено можливість ефективного навчання та розгортання розроблених ГЗМ на кордонних системах на прикладі Raspberry Pi 5. Шляхом застосування методів квантування (INT8), адаптованих під архітектуру процесорів ARM, та профілювання досягнуто стабільної роботи моделей у режимі реального часу. Це підтверджує, що запропоновані методи оптимізації дозволяють долати апаратні бар'єри, перетворюючи ресурсномісткі ГЗМ на ефективні інструменти для кордонних обчислень.

Результати роботи формують науково-практичну основу для створення автономних систем штучного інтелекту, здатних ефективно функціювати в умовах дефіциту енергії та обчислювальних потужностей. Розроблені рішення можуть бути безпосередньо імплементовані у широкий спектр апаратно-обмежених інженерних середовищ: від портативних біометричних сенсорів до автономних дронів та вбудованих систем контролю. Розроблені методи

оптимізації також можуть бути адаптовані для інших типів нейронних мереж, що відкриває нові перспективи для їхнього застосування.

Майбутні дослідження доцільно спрямувати на теоретико-інформаційний аналіз генеративних архітектур, зокрема інтеграцію принципів теорії інформації для глибокої оптимізації латентних просторів ГЗМ. Перспективним напрямом є застосування концепції інформаційної вузини (англ. information bottleneck) та аналіз динаміки взаємної інформації (англ. mutual information) між нейромережевими шарами. Це дозволить математично обґрунтувати фундаментальні межі стиснення моделей, мінімізувати структурну надмірність мереж та розробити нові інформаційні критерії збіжності, що забезпечать максимальну репрезентативність і робастність автономних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Goodfellow I. et al. Generative Adversarial Networks. Advances in Neural Information Processing Systems. 2014. Vol. 27. P. 2672–2680.
2. Hillis W. D. Co-evolving parasites improve simulated evolution as an optimization procedure. Physica D: Nonlinear Phenomena. 1990. Vol. 42, no. 1–3. P. 228–234.
3. Li W., Gauci M., Groß R. A Coevolutionary Approach to Learn Animal Behavior Through Controlled Interaction. GECCO '13: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation. Amsterdam, 2013. P. 223–230.
4. Schmidhuber J. Learning Factorial Codes by Predictability Minimization. Neural Computation. 1992. Vol. 4, no. 6. P. 863–879.
5. Researchers Used Deep Learning and an Evolutionary Algorithm to Design Chips in Minutes. The Batch. 2025. Онлайн за посиланням — https://bit.ly/4jpplAP.
6. Karahan E. A. et al. Deep-learning enabled generalized inverse design of multi-port radio-frequency and sub-terahertz passives and integrated circuits. Nature Communications. 2024. Vol. 15, no. 1. Art. 10734.
7. AI unveils strange chip functionalities. Tech Xplore. 2025. Онлайн за посиланням — https://bit.ly/4b9G1Ks.
8. Regenwetter L., Nobari A. H., Ahmed F. Deep Generative Models in Engineering Design: A Review. Journal of Mechanical Design. 2022. Vol. 144, no. 7. Art. 071704.
9. Computers Making Computers. The Batch. 2021. Онлайн за посиланням — https://bit.ly/4smHvHw.

10. Gao J., Cao W., Yang J., Zhang X. AnalogGenie: A Generative Engine for Automatic Discovery of Analog Circuit Topologies. arXiv preprint arXiv:2503.00205. 2025. URL: https://arxiv.org/abs/2503.00205
11. How AlphaChip transformed computer chip design. Google DeepMind. 2024. Онлайн за посиланням — https://bit.ly/49jxB0I.
12. Schawinski K. et al. Generative adversarial networks recover features in astrophysical images of galaxies beyond the deconvolution limit. Monthly Notices of the Royal Astronomical Society: Letters. 2017. Vol. 467, no. 1. P. L110–L114.
13. Jetchev N., Bergmann U., Vollgraf R. Texture Synthesis with Spatial Generative Adversarial Networks. arXiv preprint arXiv:1611.08207. 2016. URL: https://arxiv.org/abs/1611.08207
14. Chan M. C., Stott J. P. Deep-CEE I: fishing for galaxy clusters with deep neural nets. Monthly Notices of the Royal Astronomical Society. 2019. Vol. 490, no. 4. P. 5770–5787.
15. Fussell L., Moews B. Forging new worlds: high-resolution synthetic galaxies with chained generative adversarial networks. Monthly Notices of the Royal Astronomical Society. 2019. Vol. 485, no. 3. P. 3203–3214.
16. Mustafa M. et al. CosmoGAN: creating high-fidelity weak lensing convergence maps using Generative Adversarial Networks. Computational Astrophysics and Cosmology. 2019. Vol. 6, no. 1. Art. 1.
17. CosmoGAN: Training a neural network to study dark matter. Phys.org. 2019. Онлайн за посиланням — https://bit.ly/4pYBSgX.
18. Paganini M., de Oliveira L., Nachman B. Learning particle physics by example: Location-aware generative adversarial networks for physics synthesis. Computing and Software for Big Science. 2017. Vol. 1, no. 1. Art. 4.

19. Salamani D. et al. Deep Generative Models for Fast Shower Simulation in ATLAS. 2018 IEEE 14th International Conference on e-Science (e-Science). Amsterdam, 2018. DOI: 10.1109/eScience.2018.00091.
20. Zhavoronkov A. et al. Deep learning enables rapid identification of potent DDR1 kinase inhibitors. Nature Biotechnology. 2019. Vol. 37, no. 9. P. 1038–1040.
21. Méndez-Lucio O. et al. De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. Nature Communications. 2020. Vol. 11, no. 1. Art. 10.
22. Lan L. et al. Generative Adversarial Networks and Its Applications in Biomedical Informatics. Frontiers in Public Health. 2020. Vol. 8. Art. 164. DOI: 10.3389/fpubh.2020.00164.
23. Xun S. et al. Generative adversarial networks in medical image segmentation: A review. Computers in Biology and Medicine. 2022. Vol. 140. Art. 105063. DOI: 10.1016/j.compbiomed.2021.105063.
24. Gong M. et al. Generative Adversarial Networks in Medical Image Processing. Current Pharmaceutical Design. 2021. Vol. 27, no. 15. P. 1856–1868. DOI: 10.2174/1381612826666201125110710.
25. Skandarani Y., Jodoin P.-M., Lalande A. GANs for Medical Image Synthesis: An Empirical Study. Journal of Imaging. 2023. Vol. 9, no. 3. Art. 69. DOI: 10.3390/jimaging9030069.
26. Li X. et al. When medical images meet generative adversarial network: recent development and research opportunities. Discover Artificial Intelligence. 2021. Vol. 1. Art. 5. DOI: 10.1007/s44163-021-00006-0.
27. DuMont Schütte A. et al. Overcoming barriers to data sharing with medical image generation: a comprehensive evaluation. npj Digital Medicine. 2021. Vol. 4. Art. 141. DOI: 10.1038/s41746-021-00507-3.

28. Guarnera L., Giudice O., Battiato S. DeepFake Detection by Analyzing Convolutional Traces. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). 2020. P. 2841–2850.
29. Marouf M. et al. Realistic in silico generation and augmentation of single-cell RNA-seq data using generative adversarial networks. Nature Communications. 2020. Vol. 11, no. 1. Art. 166. DOI: 10.1038/s41467-019-14018-z.
30. Karras T. et al. Training generative adversarial networks with limited data. Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20). Red Hook, NY : Curran Associates Inc., 2020. Art. 1015. P. 12104–12114.
31. Wolterink J. M. et al. Deep MR to CT Synthesis using Unpaired Data. Simulation and Synthesis in Medical Imaging (SASHIMI 2017). Cham : Springer, 2017. Vol. 10557. P. 14–23. DOI: 10.1007/978-3-319-68127-6_2.
32. Li W. et al. Semi-supervised Rare Disease Detection Using Generative Adversarial Network. Machine Learning for Health (ML4H) Workshop at NeurIPS 2018. Montreal, 2018. arXiv:1812.00547.
33. Macedo B., Ribeiro Vaz I., Taveira Gomes T. MedGAN: optimized generative adversarial network with graph convolutional networks for novel molecule design. Scientific Reports. 2024. Vol. 14. Art. 1212. DOI: 10.1038/s41598-023-50834-6.
34. Schmidt J. et al. Recent advances and applications of machine learning in solid-state materials science. npj Computational Materials. 2019. Vol. 5. Art. 83. DOI: 10.1038/s41524-019-0221-0.
35. Nouira A., Crivello J.-C., Sokolovska N. CrystalGAN: Learning to Discover Crystallographic Structures with Generative Adversarial Networks. AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge

Engineering (AAAI-MAKE 2019). Palo Alto, CA, 2019. Vol. 2350. (CEUR Workshop Proceedings).

36. Striuk O., Kondratenko Y., Sidenko I., Vorobyova A. Generative Adversarial Neural Network for Creating Photorealistic Images. 2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT). Kyiv, 2020. P. 368–371. DOI: 10.1109/ATIT50783.2020.9349326.
37. Greenemeier L. When Will Computers Have Common Sense? Ask Facebook. Scientific American. 2016. June 20. Онлайн за посиланням — https://bit.ly/49jlwZa.
38. Sajjadi M. S. M., Schölkopf B., Hirsch M. EnhanceNet: Single Image Super-Resolution Through Automated Texture Synthesis. Proceedings of the IEEE International Conference on Computer Vision (ICCV). Piscataway, NJ : IEEE, 2017. P. 4501–4510. DOI: 10.1109/ICCV.2017.481.
39. Shi H., Dong J., Wang W., Qian Y., Zhang X. SSGAN: Secure Steganography Based on Generative Adversarial Networks. Advances in Multimedia Information Processing – PCM 2017. Cham : Springer, 2018. Vol. 10735. P. 534–544. DOI: 10.1007/978-3-319-77380-3_51.
40. Seidlitz S. et al. Generation of Privacy-friendly Datasets of Latent Fingerprint Images using Generative Adversarial Networks. Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2021). 2021. P. 345–352. DOI: 10.5220/0010251603450352.
41. Striuk O., Kondratenko Y. Gradient-Penalty GAN Framework for High-Fidelity Fingerprint Synthesis. Proceedings of The Eighth International Workshop on Computer Modeling and Intelligent Systems (CMIS 2025). Zaporizhzhia, 2025. Vol. 3988. P. 175–188. (CEUR Workshop Proceedings).
42. Hu W., Tan Y. Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN. arXiv preprint arXiv:1702.05983. 2017.

43. Guo S. et al. A Black-Box Attack Method against Machine-Learning-Based Anomaly Network Flow Detection Models. Security and Communication Networks. 2021. Vol. 2021. Art. 5578335. DOI: 10.1155/2021/5578335.
44. Shahpasand M., Hamey L., Vatsalan D., Xue M. Adversarial Attacks on Mobile Malware Detection. 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile). Hangzhou, 2019. P. 17–20. DOI: 10.1109/AI4Mobile.2019.8672711.
45. Kargaard J. et al. Defending IT systems against intelligent malware. 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT). Kyiv, 2018. P. 411–417. DOI: 10.1109/DESSERT.2018.8409169.
46. Taheri R., Shojafar M., Alazab M., Tafazolli R. Fed-IIoT: A Robust Federated Malware Detection Architecture in Industrial IoT. IEEE Transactions on Industrial Informatics. 2021. Vol. 17, no. 12. P. 8442–8452. DOI: 10.1109/TII.2020.3043458.
47. Kim J. Y., Bu S. J., Cho S. B. Malware Detection Using Deep Transferred Generative Adversarial Networks. Neural Information Processing (ICONIP 2017). Cham : Springer, 2017. Vol. 10634. P. 556–564. DOI: 10.1007/978-3-319-70087-8_58.
48. Kim J. Y., Bu S. J., Cho S. B. Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders. Information Sciences. 2018. Vol. 460–461. P. 83–102. DOI: 10.1016/j.ins.2018.04.092.
49. Hitaj B., Gasti P., Ateniese G., Perez-Cruz F. PassGAN: A Deep Learning Approach for Password Guessing. Applied Cryptography and Network Security (ACNS 2019). Cham : Springer, 2019. Vol. 11464. P. 217–237. DOI: 10.1007/978-3-030-21568-2_11.

50. Vinayakumar R., Soman K. P., Poornachandran P. Applying convolutional neural network for network intrusion detection. 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). Udupi, 2017. P. 1222–1228. DOI: 10.1109/ICACCI.2017.8126009.
51. Yan Q. et al. Automatically synthesizing DoS attack traces using generative adversarial networks. International Journal of Machine Learning and Cybernetics. 2019. Vol. 10. P. 3387–3396. DOI: 10.1007/s13042-019-00925-6.
52. Lin Z., Shi Y., Xue Z. IDSGAN: Generative Adversarial Networks for Attack Generation against Intrusion Detection. Advances in Knowledge Discovery and Data Mining. 26th Pacific-Asia Conference (PAKDD 2022). Chengdu, 2022. Vol. 13282. P. 79–91. DOI: 10.1007/978-3-031-05981-0_7.
53. Usama M. et al. Generative Adversarial Networks For Launching and Thwarting Adversarial Attacks on Network Intrusion Detection Systems. 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC). Tangier, 2019. P. 78–83. DOI: 10.1109/IWCMC.2019.8766353.
54. Yin C. et al. An enhancing framework for botnet detection using generative adversarial networks. 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD). Chengdu, 2018. P. 228–234. DOI: 10.1109/ICAIBD.2018.8396200.
55. Striuk O., Kondratenko Y. Generative Adversarial Neural Networks and Deep Learning: Successful Cases and Advanced Approaches. International Journal of Computing. 2021. Vol. 20, no. 3. P. 339–349. DOI: 10.47839/ijc.20.3.2278.
56. Striuk O., Kondratenko Y. Adaptive Deep Convolutional GAN for Fingerprint Sample Synthesis. 2021 IEEE 4th International Conference on

Advanced Information and Communication Technologies (AICT). Lviv, 2021. P. 193–196. DOI: 10.1109/AICT52120.2021.9628978.

57. Ledig C. et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. P. 105–114. DOI: 10.1109/CVPR.2017.19.
58. Wang X. et al. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. Computer Vision – ECCV 2018 Workshops. Cham : Springer, 2019. Vol. 11133. P. 63–79. DOI: 10.1007/978-3-030-11021-5_5.
59. Arjovsky M., Chintala S., Bottou L. Wasserstein Generative Adversarial Networks. Proceedings of the 34th International Conference on Machine Learning (ICML 2017). Sydney, 2017. Vol. 70. P. 214–223.
60. Mao X. et al. Least Squares Generative Adversarial Networks. Proceedings of the IEEE International Conference on Computer Vision (ICCV). Venice, 2017. P. 2813–2821. DOI: 10.1109/ICCV.2017.304.
61. Lim J. H., Ye J. C. Geometric GAN. arXiv preprint arXiv:1705.02894. 2017.
62. Li C.-L. et al. MMD GAN: towards deeper understanding of moment matching network. Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017). Long Beach, 2017. P. 2200–2210.
63. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016.
64. Deisenroth M. P., Faisal A. A., Ong C. S. Mathematics for Machine Learning. Cambridge : Cambridge University Press, 2020. 417 p.
65. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd

International Conference on Machine Learning (ICML 2015). Lille, 2015. Vol. 37. P. 448–456.

66. Salimans T. et al. Improved Techniques for Training GANs. Advances in Neural Information Processing Systems 29 (NIPS 2016). Barcelona, 2016. P. 2226–2234.
67. Radford A., Metz L., Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. International Conference on Learning Representations (ICLR). San Juan, 2016.
68. Xiang S., Li H. On the Effects of Batch and Weight Normalization in Generative Adversarial Networks. arXiv preprint arXiv:1704.03971. 2017.
69. Gulrajani I. et al. Improved training of Wasserstein GANs. Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017). Long Beach, 2017. P. 5769–5779.
70. Roth K. et al. Stabilizing training of Generative Adversarial Networks through regularization. Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017). Long Beach, 2017. P. 2015–2025.
71. Hui J. GAN — Ways to improve GAN performance. Towards Data Science. 2018. Онлайн за посиланням — https://bit.ly/490oZgs.
72. Miyato T., Kataoka T., Koyama M., Yoshida Y. Spectral Normalization for Generative Adversarial Networks. International Conference on Learning Representations (ICLR). Vancouver, 2018.
73. Ayinde B. O., Nishihama K., Zurada J. M. Diversity Regularized Adversarial Deep Learning. Artificial Intelligence Applications and Innovations. 15th IFIP WG 12.5 International Conference (AIAI 2019). Cham : Springer, 2019. P. 292–306. DOI: 10.1007/978-3-030-19823-7_24.

74. Srivastava N. et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014. Vol. 15. P. 1929–1958.
75. Dumoulin V., Visin F. A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285. 2016.
76. Brownlee J. A Gentle Introduction to Transfer Learning for Deep Learning. Machine Learning Mastery. 2019. Онлайн за посиланням — https://bit.ly/4sgYPNX.
77. Striuk O., Kondratenko Y. Optimization Strategy for Generative Adversarial Networks Design. International Journal of Computing. 2023. Vol. 22, Iss. 3. P. 292–301. DOI: 10.47839/ijc.22.3.3223.
78. Heusel M. et al. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017). Long Beach : Curran Associates Inc., 2017. P. 6629–6640.
79. Thakur A. How to Evaluate GANs using Frechet Inception Distance (FID). Weights & Biases. 2021. Онлайн за посиланням — https://bit.ly/49nCfLb.
80. Brownlee J. How to Implement the Frechet Inception Distance (FID) for Evaluating GANs. Machine Learning Mastery. 2019. Онлайн за посиланням — https://bit.ly/4qcVFtd.
81. Barratt S., Sharma R. A Note on the Inception Score. arXiv preprint arXiv:1801.01973. 2018.
82. Sajjadi M. S. M. et al. Assessing generative models via precision and recall. Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18). 2018. P. 5234–5243.
83. Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks. Proceedings of the 2019 IEEE/CVF

Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, 2019. P. 4396–4405.

84. Karras T. et al. Analyzing and Improving the Image Quality of StyleGAN. Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, 2020. P. 8107–8116.
85. Brownlee J. A Gentle Introduction to BigGAN the Big Generative Adversarial Network. Machine Learning Mastery. 2019. Онлайн за посиланням — https://bit.ly/4aC574s.
86. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations (ICLR). San Diego, 2015.
87. Howard J., Ruder S. Universal Language Model Fine-tuning for Text Classification. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018). Melbourne, 2018. Vol. 1. P. 328–339. DOI: 10.18653/v1/P18-1031.
88. Smith L. N. Cyclical Learning Rates for Training Neural Networks. 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). Santa Rosa, 2017. P. 464–472. DOI: 10.1109/WACV.2017.58.
89. Striuk O., Kondratenko Y. Cross-Domain Reconfigurable GAN with Fuzzy Components for Anomaly Detection. 2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT). Athens, 2023. P. 1–5. DOI: 10.1109/DESSERT61349.2023.10416521.
90. Kingma D. P., Welling M. Auto-Encoding Variational Bayes. International Conference on Learning Representations (ICLR). Banff, 2014.
91. van den Oord A., Vinyals O., Kavukcuoglu K. Neural Discrete Representation Learning. Advances in Neural Information Processing Systems (NIPS 2017). Long Beach, 2017. P. 6306–6315.

92. van den Oord A. et al. Conditional Image Generation with PixelCNN Decoders. Advances in Neural Information Processing Systems (NIPS 2016). Barcelona, 2016. P. 4790–4798.
93. Kingma D. P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions. Advances in Neural Information Processing Systems (NeurIPS 2018). Montréal, 2018. P. 10215–10224.
94. Larsen A. B. L., Sønderby S. K., Larochelle H., Winther O. Autoencoding beyond pixels using a learned similarity metric. Proceedings of the 33rd International Conference on Machine Learning (ICML 2016). New York, 2016. Vol. 48. P. 1558–1566.
95. Ho J., Jain A., Abbeel P. Denoising Diffusion Probabilistic Models. Advances in Neural Information Processing Systems (NeurIPS 2020). 2020. Vol. 33. P. 6840–6851.
96. Ferdowsi A., Saad W. Generative Adversarial Networks for Distributed Intrusion Detection in the Internet of Things. 2019 IEEE Global Communications Conference (GLOBECOM). Waikoloa, HI, 2019. P. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9014102.
97. Rasouli M., Sun T., Rajagopal R. FedGAN: Federated Generative Adversarial Networks for Distributed Data Generation. arXiv preprint arXiv:2006.07228. 2020.
98. Hardy C., Le Merrer E., Sericola B. MD-GAN: Multi-Discriminator Generative Adversarial Networks for Distributed Datasets. 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS). Rio de Janeiro, 2019. P. 866–877. DOI: 10.1109/IPDPS.2019.00095.
99. Wang P. et al. QGAN: Quantized Generative Adversarial Networks. arXiv preprint arXiv:1901.08263. 2019.

100. Wang H., Gui S., Yang H., Liu J., Wang Z. GAN Slimming: All-in-One GAN Compression by A Unified Optimization Framework. arXiv preprint arXiv:2008.11062. 2020.

101. Li M., Lin J., Ding Y., Liu Z., Zhu J.-Y., Han S. GAN Compression: Efficient Architectures for Interactive Conditional GANs. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, WA, 2020. Art. 533. DOI: 10.1109/CVPR42600.2020.00533.

102. Merenda M., Porcaro C., Iero D. Edge Machine Learning for AI-Enabled IoT Devices: A Review. Sensors. 2020. Vol. 20, no. 9. Art. 2533. DOI: 10.3390/s20092533.

103. Reuther A., Michaleas P., Jones M., Gadepally V., Samsi S., Kepner J. Survey of Machine Learning Accelerators. 2020. DOI: 10.1109/HPEC43674.2020.9286149.

104. Süzen A. A., Duman B., Sen B. Benchmark Analysis of Jetson TX2, Jetson Nano and Raspberry PI using Deep-CNN. 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). 2020. P. 1–5. DOI: 10.1109/HORA49412.2020.9152915.

105. Bousmalis K. et al. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. 2018 IEEE International Conference on Robotics and Automation (ICRA). Brisbane, 2018. P. 4243–4250. DOI: 10.1109/ICRA.2018.8460875.

106. Hoffman J., Tzeng E., Park T., Zhu J.-Y., Isola P., Saenko K., Efros A., Darrell T. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. Proceedings of the 35th International Conference on Machine Learning (ICML 2018). Stockholm, 2018. Vol. 80. P. 1989–1998.

107. Tzeng E., Hoffman J., Saenko K., Darrell T. Adversarial Discriminative Domain Adaptation. 2017 IEEE Conference on Computer Vision and Pattern

Recognition (CVPR). Honolulu, 2017. P. 2962–2971. DOI: 10.1109/CVPR.2017.316.

108. Alnujaim I., Kim Y. Augmentation of Doppler Radar Data Using Generative Adversarial Network for Human Motion Analysis. Healthcare Informatics Research. 2019. Vol. 25, no. 4. P. 344–349. DOI: 10.4258/hir.2019.25.4.344.
109. Saarinen V., Koivunen V. Radar Waveform Synthesis Using Generative Adversarial Networks. 2020 IEEE Radar Conference (RadarConf20). Florence, 2020. P. 1–6. DOI: 10.1109/RadarConf2043947.2020.9266709.
110. Wang J., Li J., Sun B., Zuo Z. SAR image synthesis based on conditional generative adversarial networks. The Journal of Engineering. 2019. Vol. 2019, no. 19. P. 5930–5933. DOI: 10.1049/joe.2019.0696.
111. Goni O. et al. HingeRLC-GAN: Combatting Mode Collapse with Hinge Loss and RLC Regularization. Pattern Recognition: 27th International Conference (ICPR 2024). Cham : Springer, 2024. Vol. 15325. P. 370–385. DOI: 10.1007/978-3-031-78389-0_25.
112. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. 3rd International Conference on Learning Representations (ICLR 2015). San Diego, 2015. P. 1–14.
113. Hinton G. Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude. University of Toronto. 2012. Онлайн за посиланням — https://bit.ly/4uxe1rs
114. Robbins H., Monro S. A Stochastic Approximation Method. The Annals of Mathematical Statistics. 1951. Vol. 22, no. 3. P. 400–407. DOI: 10.1214/aoms/1177729586.
115. Welling M., Teh Y. W. Bayesian Learning via Stochastic Gradient Langevin Dynamics. Proceedings of the 28th International Conference on Machine Learning (ICML 2011). Bellevue, WA, 2011. P. 681–688.

116. An W. et al. A PID Controller Approach for Stochastic Optimization of Deep Networks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Salt Lake City, UT, 2018. P. 8522–8531. DOI: 10.1109/CVPR.2018.00889.
117. Pascanu R., Mikolov T., Bengio Y. On the difficulty of training recurrent neural networks. Proceedings of the 30th International Conference on Machine Learning (ICML 2013), PMLR 28(3), 2013. P. 1310-1318.
118. Cortes C., Vapnik V. Support-vector networks. Machine Learning. 1995. Vol. 20, no. 3. P. 273–297.
119. Nair V., Hinton G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. Proceedings of the 27th International Conference on Machine Learning (ICML 2010). Haifa, 2010. P. 807–814.
120. Shehu Y. I. et al. Sokoto Coventry Fingerprint Dataset. arXiv preprint arXiv:1807.10609. 2018. DOI: 10.48550/arXiv.1807.10609.

Додаток А

Список публікацій здобувача за темою дисертації

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Праці, в яких опубліковані основні наукові результати дисертації:

1. *Стрюк О. С.,* Кондратенко Ю. П. Методи прикладного застосування генеративних змагальних мереж при обробці графічних даних. *Штучний інтелект*, 2023. № 3. С. 154–161. DOI: https://doi.org/10.15407/jai2023.03.154. **(ISSN 2710–1673) (Категорія Б)**

   (Особистий внесок: результати порівняльного аналізу методів прикладного застосування генеративних змагальних мереж при обробці графічних даних на основі проведених особисто експериментальних досліджень.)

   URL: https://jai.in.ua/index.php/архів?paper_num=1606.

2. *Striuk O.*, Kondratenko Y. Generative Adversarial Neural Networks and Deep Learning: Successful Cases and Advanced Approaches. *International Journal of Computing*, 2021. Vol. 20, No. 3. P. 339–349. DOI: https://doi.org/10.47839/ijc.20.3.2278. **(ISSN 1727–6209) (Scopus)**

   URL: https://computingonline.net/computing/article/view/2278.

3. *Striuk O. S.*, Kondratenko Y. P. Generative Adversarial Networks in Cybersecurity: Analysis and Response. In: C. Berger-Vachon, et al. (Eds) *Artificial Intelligence in Control and Decision-making Systems. Studies in Computational Intelligence*. Cham: Springer, 2023. Vol. 1087. P. 373–388. DOI: https://doi.org/10.1007/978-3-031-25759-9_18. **(Scopus)**

   URL: https://link.springer.com/chapter/10.1007/978-3-031-25759-9_18.

4. *Striuk O.*, Kondratenko Y. Implementation of Generative Adversarial Networks in Mobile Applications for Image Data Enhancement. *Journal of Mobile Multimedia*, 2023. Vol. 19, No. 03. P. 823–838. DOI: https://doi.org/10.13052/jmm1550-4646.1938. **(ISSN 1550–4646) (Scopus)**

   URL:https:journals.riverpublishers.com/index.php/JMM/article/view/15053.

5. *Striuk O.*, Kondratenko Y. Optimization Strategy for Generative Adversarial Networks Design. *International Journal of Computing*, 2023. Vol. 22, No. 3. P. 292–301. DOI: doi.org/10.47839/ijc.22.3.3223. **(ISSN 1727–6209) (Scopus)**

   URL: https://computingonline.net/computing/article/view/3223.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

6. Generative Adversarial Neural Network for Creating Photorealistic Images / *Striuk O.*, Kondratenko Y., Sidenko I., Vorobyova A. *Information Control Systems & Technologies (ATIT 2020)*: Proceedings of the 2nd IEEE International Conference on Advanced Trends in Information Theory, Kyiv, 25–27 November, 2020. Kyiv, 2020. P. 368–371. DOI: 10.1109/ATIT50783.2020.9349326. **(Scopus)**

   URL: https://ieeexplore.ieee.org/document/9349326/.

7. Adaptive Deep Convolutional GAN for Fingerprint Sample Synthesis / *Striuk O.*, Kondratenko Y. *Information Control Systems & Technologies (AICT 2021)*: Proceedings of the 4th IEEE International Conference on Advanced Information and Communication Technologies, Lviv, 21–24 September, 2021. Lviv, 2021. P. 193–196. DOI: 10.1109/AICT52120.2021.9628978. **(Scopus)**

   URL: https://ieeexplore.ieee.org/document/9628978.

8. Cross-Domain Reconfigurable GAN with Fuzzy Components for Anomaly Detection / *Striuk O.*, Kondratenko Y. *Dependable Systems, Services and Technologies (DESSERT 2023)*: Proceedings of the 13th International Conference, Athens, 13–15 October, 2023. Athens, 2023. P. 1–5. DOI: 10.1109/DESSERT61349.2023.10416521. **(Scopus)**

   URL: https://ieeexplore.ieee.org/document/10416521.

9. Gradient-Penalty GAN Framework for High-Fidelity Fingerprint Synthesis / *Striuk O.*, Kondratenko Y. *Computer Modeling and Intelligent Systems (CMIS 2025)*: Proceedings of the 8th International Workshop, Zaporizhzhia, 5 May,

2025. Zaporizhzhia, CEUR Workshop Proceedings, Vol. 3988, 2025. P. 175–188. **(ISSN 1613–0073) (Scopus)**

URL: https://ceur-ws.org/Vol-3988/paper15.pdf.

10. Математична модель генеративно-змагальної мережі / Воробйова А. І., *Стрюк О. С. Могилянські читання – 2020: Досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти*: матеріали XXIII Всеукраїнської науково-практичної конференції, м. Миколаїв, 16–20 листопада 2020 р., Миколаїв, 2020. С. 134–137.

URL: https://dspace.chmnu.edu.ua/jspui/handle/123456789/405.

11. Прикладна цінність ГЗМ як систем штучного інтелекту / *Стрюк О. С. Інтелектуальні інформаційні системи*: матеріали Всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів, м. Миколаїв, 28–31 січня 2020 р., Миколаїв, 2020. С. 35–38.

URL: http://bit.ly/4rjb0YQ.

12. Розгортання та інженерна оптимізація генеративних змагальних мереж на кордонних системах / *Стрюк О. С. Інтелектуальні інформаційні системи*: матеріали Всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів, м. Миколаїв, 4–5 грудня 2025 р., Миколаїв, 2025. С. 104–106.

URL: https://dspace.chmnu.edu.ua/jspui/handle/123456789/3054.

Наукові праці, які додатково відображають наукові результати дисертації:

13. Шевченко А. І., Барановський С. В., Білокобильський О. В., Кондратенко Ю. П., Козлов О. В., Сіденко Є. В., *Стрюк О. С.* та ін. *Стратегія розвитку штучного інтелекту в Україні: монографія* / за заг. ред. А. І. Шевченка. Київ: ІПШІ, 2023. С. 1-307. DOI: 10.15407/development_strategy_2023.

(Особистий внесок: підготовлені доповнення до розділів 1 «Парадигма», 3 «Мета і завдання», 8 «Наукове, кадрове та матеріальне забезпечення національної екосистеми ШІ», 10 «Прикінцеві положення»; підрозділів: 2.2 «Основні напрями досліджень ШІ» розділу 2, 7.1 «ШІ у сфері безпеки та оборони України», 7.4 «ШІ в промисловості та енергетиці», 7.7 «ШІ у сільському господарстві» розділу 7.)

URL: https://jai.in.ua/index.php/en/issues?paper_num=1545.

14. Tendencies and Challenges of Artificial Intelligence Development and Implementation / Kondratenko Y., Shevchenko A., Zhukov Y., Kondratenko G., *Striuk O. Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2023)*: proceedings of the 12th IEEE International Conference, Dortmund, 7–9 September, 2023. Dortmund, 2023. P. 221–226. DOI: 10.1109/IDAACS58523.2023.10348800. **(Scopus)**

    URL: https://ieeexplore.ieee.org/document/10348800.

15. Analysis of the Priorities and Perspectives in Artificial Intelligence Implementation / Kondratenko Y., Shevchenko A., Zhukov Y., Kondratenko G., *Striuk O. Systems, Services and Technologies (DESSERT 2023)*: Proceedings of the 13th International Conference, Athens, 13–15 October, 2023. Athens, 2023. P. 1–8. DOI: 10.1109/DESSERT61349.2023.10416432. **(Scopus)**

    URL: https://ieeexplore.ieee.org/document/10416432.

16. Kondratenko Y. P., Zhukov Y. D., Shevchenko A. I., Zhukova O. Y., *Striuk O. S.* AI and Digital Evolution in the Education System of Ukraine. In: V.I. Slyusar, et al. (Eds) *AI in Education Systems: Successful Cases and Perspectives*. River Publishers, Gistrup, Denmark, 2025. P. 47–74. **(Scopus)**

    URL: https://www.riverpublishers.com/book_details.php?book_id=1488.

17. Shevchenko A. I., Lande D. V., Bilokobylsky O. V., Kondratenko Y. P., Kozlov O. V., Sidenko I. V., *Striuk O. S.* et al. Regarding the Draft Strategy Development

of Artificial Intelligence in Ukraine. *Artificial Intelligence (Штучний інтелект)*, 2022. № 1. P. 8–157. DOI: https://doi.org/10.15407/jai2022.01.008. **(ISSN 2710–1673) (Категорія Б)**

(Особистий внесок: підготовлено матеріали щодо ключових напрямів досліджень ШІ у сфері безпеки та оборони, промисловості, енергетиці та сільському господарстві України.)

URL: https://jai.in.ua/index.php/en/issues?paper_num=1492.

Додаток Б

Лістинґ програм (вихідний код)

ЛІСТИНҐ ПРОГРАМ (ВИХІДНИЙ КОД)

ADCGAN

```
import os, random, torch, torch.nn as nn, torch.optim as optim
import torchvision.datasets as dset, torchvision.transforms as
transforms
import torchvision.utils as vutils
import numpy as np, matplotlib.pyplot as plt

# Встановлюємо випадкове зерно для відтворюваності експерименту
seed = 999
random.seed(seed); torch.manual_seed(seed)

# Визначаємо конфігурацію гіперпараметрів та шляхи до даних
dataroot = "../input/socofing-real-only/socofing_real_only"
workers, batch_size, image_size = 2, 128, 64
nc, nz, ngf, ndf = 1, 100, 64, 64
num_epochs, lr, beta1 = 400, 0.0002, 0.5
device = torch.device("cuda:0" if torch.cuda.is_available() else
"cpu")

# Завантажуємо датасет із попередньою обробкою та нормалізацією
dataset = dset.ImageFolder(root=dataroot,
transform=transforms.Compose([
    transforms.Grayscale(nc), transforms.Resize(image_size),
    transforms.CenterCrop(image_size), transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
]))
dataloader = torch.utils.data.DataLoader(dataset,
batch_size=batch_size, shuffle=True, num_workers=workers)

# Функція ініціалізації ваг для стабілізації навчання (Normal distr.
0.0, 0.02)
```

```
def weights_init(m):
    classname = m.__class__.__name__
    if 'Conv' in classname: nn.init.normal_(m.weight.data, 0.0, 0.02)
    elif 'BatchNorm' in classname: nn.init.normal_(m.weight.data,
1.0, 0.02); nn.init.constant_(m.bias.data, 0)

# Архітектура Генератора: (Z-вектор -> Зображення)
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.main = nn.Sequential(
            nn.ConvTranspose2d(nz, ngf * 8, 4, 1, 0, bias=False),
nn.BatchNorm2d(ngf * 8), nn.ReLU(True),
            nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1,
bias=False), nn.BatchNorm2d(ngf * 4), nn.ReLU(True),
            nn.ConvTranspose2d(ngf * 4, ngf * 2, 4, 2, 1,
bias=False), nn.BatchNorm2d(ngf * 2), nn.ReLU(True),
            nn.ConvTranspose2d(ngf * 2, ngf, 4, 2, 1, bias=False),
nn.BatchNorm2d(ngf), nn.ReLU(True),
            nn.ConvTranspose2d(ngf, nc, 4, 2, 1, bias=False),
nn.Tanh()
        )
    def forward(self, input): return self.main(input)

# Архітектура Дискримінатора: (Зображення -> Ймовірність)
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.main = nn.Sequential(
            nn.Conv2d(nc, ndf, 4, 2, 1, bias=False),
nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(ndf, ndf * 2, 4, 2, 1, bias=False),
nn.BatchNorm2d(ndf * 2), nn.LeakyReLU(0.2, inplace=True),
```

```
            nn.Conv2d(ndf * 2, ndf * 4, 4, 2, 1, bias=False),
nn.BatchNorm2d(ndf * 4), nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(ndf * 4, ndf * 8, 4, 2, 1, bias=False),
nn.BatchNorm2d(ndf * 8), nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(ndf * 8, 1, 4, 1, 0, bias=False), nn.Sigmoid()
        )
    def forward(self, input): return self.main(input)

# Ініціалізуємо моделі, функцію втрат та оптимізатори
netG, netD = Generator().to(device), Discriminator().to(device)
netG.apply(weights_init); netD.apply(weights_init)
criterion, fixed_noise = nn.BCELoss(), torch.randn(64, nz, 1, 1,
device=device)
optD = optim.Adam(netD.parameters(), lr=lr, betas=(beta1, 0.999))
optG = optim.Adam(netG.parameters(), lr=lr, betas=(beta1, 0.999))

# Розпочинаємо цикл навчання
print("Розпочинаємо навчання...")
G_losses, D_losses = [], []

for epoch in range(num_epochs):
    for i, data in enumerate(dataloader, 0):
        # 1. Оновлення Дискримінатора: maximize log(D(x)) + log(1 -
D(G(z)))
        netD.zero_grad()
        real_cpu = data[0].to(device)
        b_size = real_cpu.size(0)
        label = torch.full((b_size,), 1., dtype=torch.float,
device=device)
        output = netD(real_cpu).view(-1)
        errD_real = criterion(output, label)
        errD_real.backward()

        noise = torch.randn(b_size, nz, 1, 1, device=device)
```

```
        fake = netG(noise)
        label.fill_(0.)
        output = netD(fake.detach()).view(-1)
        errD_fake = criterion(output, label)
        errD_fake.backward()
        optD.step(); errD = errD_real + errD_fake

        # 2. Оновлення Генератора: maximize log(D(G(z)))
        netG.zero_grad()
        label.fill_(1.)
        output = netD(fake).view(-1)
        errG = criterion(output, label)
        errG.backward()
        optG.step()

        G_losses.append(errG.item()); D_losses.append(errD.item())

        if i % 100 == 0:
            print(f'[{epoch}/{num_epochs}][{i}/{len(dataloader)}]
Loss_D: {errD.item():.4f} Loss_G: {errG.item():.4f}')

print("Навчання завершено. Формуємо графіки та результати.")

# Візуалізуємо результати: динаміка втрат та згенеровані зразки
plt.figure(figsize=(10,5))
plt.plot(G_losses, label="G"), plt.plot(D_losses, label="D")
plt.xlabel("Ітерації"), plt.ylabel("Loss"), plt.legend(), plt.show()

with torch.no_grad(): fake = netG(fixed_noise).detach().cpu()
plt.figure(figsize=(8,8)); plt.axis("off"); plt.title("Згенеровані
зображення")
plt.imshow(np.transpose(vutils.make_grid(fake, padding=2,
normalize=True), (1,2,0)), cmap='gray')
plt.show()
```

Anomaly GAN

```
# !pip install scikit-fuzzy

import torch, torch.nn as nn, torch.optim as optim
import torchvision, torchvision.transforms as transforms
import numpy as np, matplotlib.pyplot as plt
import skfuzzy as fuzz
from torch.utils.data import DataLoader
from sklearn.metrics import roc_auc_score, roc_curve, auc

# Фіксуємо параметри для відтворюваності
torch.manual_seed(42)
np.random.seed(42)

class Config:
    lr = 0.0002
    epochs = 250        # Кількість епох навчання
    batch_size = 128
    z_dim = 100         # Розмірність латентного вектора
    normal_class = 1    # Клас MNIST, який вважаємо "нормою"
    device = torch.device("cuda:0" if torch.cuda.is_available() else
"cpu")

# 1. Завантаження та підготовка даних
# Використовуємо лише один клас для навчання (One-Class
Classification)
transform = transforms.Compose([transforms.ToTensor(),
transforms.Normalize((0.5,), (0.5,))])

# Завантажуємо повні датасети
mnist_train = torchvision.datasets.MNIST(root='./data', train=True,
transform=transform, download=True)
```

```
mnist_test = torchvision.datasets.MNIST(root='./data', train=False,
transform=transform, download=True)

# Фільтруємо навчальні дані: залишаємо лише "нормальний" клас
train_subset = [img for img, lbl in mnist_train if lbl ==
Config.normal_class]
train_loader = DataLoader(train_subset, batch_size=Config.batch_size,
shuffle=True)
test_loader = DataLoader(mnist_test, batch_size=Config.batch_size,
shuffle=False)

# 2. Архітектура моделей (MLP GAN)
class Generator(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(Config.z_dim, 256), nn.ReLU(),
            nn.Linear(256, 512), nn.ReLU(),
            nn.Linear(512, 784), nn.Tanh()
        )
    def forward(self, x): return self.net(x).view(-1, 1, 28, 28)

class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(784, 512), nn.LeakyReLU(0.2),
            nn.Linear(512, 256), nn.LeakyReLU(0.2),
            nn.Linear(256, 1), nn.Sigmoid()
        )
    def forward(self, x): return self.net(x.view(-1, 784))

# Ініціалізація моделей та оптимізаторів
```

```
G, D = Generator().to(Config.device),
Discriminator().to(Config.device)
opt_g = optim.Adam(G.parameters(), lr=Config.lr, betas=(0.5, 0.999))
opt_d = optim.Adam(D.parameters(), lr=Config.lr, betas=(0.5, 0.999))
criterion = nn.BCELoss()

# 3. Цикл навчання
print("Розпочинаємо процес навчання GAN...")

for epoch in range(Config.epochs):
    for imgs, _ in train_loader:
        imgs = imgs.to(Config.device)
        curr_batch = imgs.size(0)

        # --- Оновлення Дискримінатора ---
        opt_d.zero_grad()
        real_labels = torch.ones(curr_batch, 1).to(Config.device)
        fake_labels = torch.zeros(curr_batch, 1).to(Config.device)

        # Loss на справжніх даних
        loss_real = criterion(D(imgs), real_labels)

        # Loss на штучних даних
        z = torch.randn(curr_batch, Config.z_dim).to(Config.device)
        fake_imgs = G(z)
        loss_fake = criterion(D(fake_imgs.detach()), fake_labels)

        loss_d = loss_real + loss_fake
        loss_d.backward()
        opt_d.step()

        # --- Оновлення Генератора ---
        opt_g.zero_grad()
        loss_g = criterion(D(fake_imgs), real_labels)
```

```
        loss_g.backward()
        opt_g.step()

    if (epoch + 1) % 50 == 0:
        print(f"Epoch [{epoch+1}/{Config.epochs}] Loss D:
{loss_d.item():.4f}, Loss G: {loss_g.item():.4f}")

# 4. Виявлення аномалій
print("Розпочинаємо тестування та розрахунок метрик...")
anomaly_scores, labels = [], []

with torch.no_grad():
    for imgs, lbls in test_loader:
        imgs = imgs.to(Config.device)
        # Генеруємо зразки з випадкового шуму для порівняння
        z = torch.randn(imgs.size(0), Config.z_dim).to(Config.device)
        recon_imgs = G(z)

        # Векторизований розрахунок L1 loss (Contextual Loss)
        # Середнє значення абсолютної різниці по кожному зображенню
        batch_scores = torch.mean(torch.abs(recon_imgs - imgs),
dim=[1, 2, 3])

        anomaly_scores.extend(batch_scores.cpu().numpy())
        labels.extend(lbls.numpy())

# Підготовка міток: 0 - норма (наш клас), 1 - аномалія
y_true = np.array([0 if l == Config.normal_class else 1 for l in
labels])
y_scores = np.array(anomaly_scores)

# Розрахунок метрик якості
roc_val = roc_auc_score(y_true, y_scores)
print(f'Final AUC Score: {roc_val:.4f}')
```

```
# 5. Категоризація ризиків (Fuzzy Logic)
print("Застосовуємо нечітку логіку для інтерпретації результатів...")

# Нормалізація оцінок аномальності до [0, 1] для Fuzzy системи
norm_scores = (y_scores - y_scores.min()) / (y_scores.max() -
y_scores.min())
x_fuzzy = np.linspace(0, 1, 100)

# Визначення функцій належності (Membership Functions)
mf_low = fuzz.trimf(x_fuzzy, [0, 0, 0.3])      # Низький ризик
mf_med = fuzz.trimf(x_fuzzy, [0.1, 0.5, 0.9])  # Середній ризик
mf_high = fuzz.trimf(x_fuzzy, [0.7, 1, 1])     # Високий ризик

# Візуалізація результатів
plt.figure(figsize=(12, 5))

# Графік ROC-кривої
fpr, tpr, _ = roc_curve(y_true, y_scores)
plt.subplot(1, 2, 1)
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC =
{roc_val:.2f}')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlabel('False Positive Rate'); plt.ylabel('True Positive Rate')
plt.title('ROC Curve'); plt.legend()

# Графік нечітких множин
plt.subplot(1, 2, 2)
plt.plot(x_fuzzy, mf_low, 'b', label='Low')
plt.plot(x_fuzzy, mf_med, 'g', label='Medium')
plt.plot(x_fuzzy, mf_high, 'r', label='High')
# Відображаємо зріз точок для прикладу
plt.scatter(norm_scores[:50], [0.5]*50, c='k', marker='.',
label='Test Samples')
```

```
plt.title('Fuzzy Anomaly Categorization')
plt.legend(); plt.grid(True, alpha=0.3)
plt.show()
```

EAWGAN GP

```
import os, torch, torch.nn as nn, torch.optim as optim
import torchvision.transforms as transforms, torchvision.datasets as
dset, torchvision.utils as vutils
from torch.autograd import grad

# 1. Конфігурація та ініціалізація
dataroot = "../input/socofing-real-only/socofing_real_only"
batch_size, img_size, z_dim = 64, 128, 100
epochs, lr, beta1, beta2 = 200, 0.0002, 0.5, 0.999
lambda_gp = 10.0
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
os.makedirs('output', exist_ok=True)

# 2. Завантаження даних
dataloader = torch.utils.data.DataLoader(
    dset.ImageFolder(root=dataroot, transform=transforms.Compose([
        transforms.Resize((img_size, img_size)),
transforms.Grayscale(),
        transforms.ToTensor(), transforms.Normalize([0.5], [0.5])
    ])), batch_size=batch_size, shuffle=True, num_workers=2,
pin_memory=True
)

# 3. Архітектура моделей
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.fc = nn.Linear(z_dim, 512 * 8 * 8) # Project and reshape
```

```
        self.net = nn.Sequential(
            nn.BatchNorm2d(512), nn.ReLU(True),
            nn.ConvTranspose2d(512, 256, 4, 2, 1),
nn.BatchNorm2d(256), nn.ReLU(True),
            nn.ConvTranspose2d(256, 128, 4, 2, 1),
nn.BatchNorm2d(128), nn.ReLU(True),
            nn.ConvTranspose2d(128, 64, 4, 2, 1),
nn.BatchNorm2d(64),  nn.ReLU(True),
            nn.ConvTranspose2d(64, 1, 4, 2, 1),   nn.Tanh()
        )
    def forward(self, z): return self.net(self.fc(z).view(-1, 512, 8,
8))

class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        def block(in_f, out_f, norm=True):
            layers = [nn.Conv2d(in_f, out_f, 4, 2, 1)]
            if norm: layers.append(nn.InstanceNorm2d(out_f,
affine=True))
            layers.append(nn.LeakyReLU(0.2, inplace=True))
            return layers
        self.net = nn.Sequential(
            *block(1, 64, norm=False), *block(64, 128),
            *block(128, 256), *block(256, 512),
            nn.Conv2d(512, 1, 4, 1, 0)
        )
    def forward(self, x): return self.net(x).view(-1, 1)

def weights_init(m):
    if 'Conv' in m.__class__.__name__: nn.init.normal_(m.weight.data,
0.0, 0.02)
```

```
    elif 'BatchNorm' in m.__class__.__name__:
nn.init.normal_(m.weight.data, 1.0, 0.02);
nn.init.constant_(m.bias.data, 0)

# 4. Допоміжна функція Gradient Penalty
def compute_gp(D, real, fake):
    alpha = torch.rand(real.size(0), 1, 1, 1, device=device)
    interp = (alpha * real + (1 - alpha) * fake).requires_grad_(True)
    d_interp = D(interp)
    gradients = grad(outputs=d_interp, inputs=interp,
grad_outputs=torch.ones_like(d_interp),
                     create_graph=True, retain_graph=True,
only_inputs=True)[0]
    return ((gradients.view(gradients.size(0), -1).norm(2, dim=1) -
1) ** 2).mean() * lambda_gp

# 5. Ініціалізація та цикл навчання
G, D = Generator().to(device), Discriminator().to(device)
G.apply(weights_init); D.apply(weights_init)
opt_G = optim.Adam(G.parameters(), lr=lr, betas=(beta1, beta2))
opt_D = optim.Adam(D.parameters(), lr=lr, betas=(beta1, beta2))
fixed_noise = torch.randn(64, z_dim, device=device)

print("Розпочинаємо навчання WGAN-GP...")

for epoch in range(epochs):
    for i, (real_imgs, _) in enumerate(dataloader):
        real_imgs = real_imgs.to(device)

        # --- Тренування Дискримінатора ---
        opt_D.zero_grad()
        z = torch.randn(real_imgs.size(0), z_dim, device=device)
        fake_imgs = G(z).detach()
```

```
        # WGAN Loss: D(x) - D(G(z)) + GP
        loss_D = -torch.mean(D(real_imgs)) + torch.mean(D(fake_imgs))
+ compute_gp(D, real_imgs, fake_imgs)
        loss_D.backward()
        opt_D.step()

        # --- Тренування Генератора (кожні 5 ітерацій) ---
        if i % 5 == 0:
            opt_G.zero_grad()
            gen_imgs = G(z) # Regenerate to keep graph
            loss_G = -torch.mean(D(gen_imgs))
            loss_G.backward()
            opt_G.step()

        if i % 100 == 0:
            print(f"[Epoch {epoch}/{epochs}] [Batch {i}] [D loss:
{loss_D.item():.4f}] [G loss: {loss_G.item() if 'loss_G' in locals()
else 0:.4f}]")

    if (epoch + 1) % 10 == 0:
        vutils.save_image(G(fixed_noise).data[:64],
f"output/epoch_{epoch+1}.png", nrow=8, normalize=True)

# Зберігаємо фінальні моделі
torch.save(G.state_dict(), "generator.pth");
torch.save(D.state_dict(), "discriminator.pth")
print("Навчання завершено, моделі збережено.")
```

SRGAN

```
import torch, torch.nn as nn, torch.optim as optim
import torchvision, torchvision.transforms as transforms
import numpy as np, matplotlib.pyplot as plt
from torch.utils.data import DataLoader, Dataset
```

```
from torchvision.utils import make_grid

# Фіксуємо випадковість для відтворюваності експериментів
torch.manual_seed(42)

class Config:
    upscale_factor = 4      # Множник збільшення (7x7 -> 28x28)
    batch_size = 64
    lr = 0.0002
    epochs = 20             # Достатньо для MNIST, щоб побачити
результат
    device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
    hr_size = 28            # Зображення високої якості
    lr_size = 7             # Зображення низької якості

# 1. Підготовка даних (Dataset)
# Ми створюємо кастомний клас Dataset, який повертає пару:
# (зображення низької якості, зображення високої якості)
class SRMNIST(Dataset):
    def __init__(self):
        self.mnist = torchvision.datasets.MNIST(root='./data',
train=True, download=True,

transform=transforms.Compose([

transforms.ToTensor(),

transforms.Normalize((0.5,), (0.5,))
                                                ]))
        self.resize_lr = transforms.Resize((Config.lr_size,
Config.lr_size), interpolation=transforms.InterpolationMode.BICUBIC)

    def __len__(self): return len(self.mnist)
```

```
    def __getitem__(self, idx):
        hr_img, _ = self.mnist[idx]
        lr_img = self.resize_lr(hr_img) # Штучне погіршення якості
        return lr_img, hr_img

dataloader = DataLoader(SRMNIST(), batch_size=Config.batch_size,
shuffle=True, num_workers=2)

# 2. Архітектура моделей
# Базовий залишковий блок (Residual Block) для Генератора
class ResidualBlock(nn.Module):
    def __init__(self, channels):
        super(ResidualBlock, self).__init__()
        self.block = nn.Sequential(
            nn.Conv2d(channels, channels, 3, 1, 1, bias=False),
            nn.BatchNorm2d(channels),
            nn.PReLU(),
            nn.Conv2d(channels, channels, 3, 1, 1, bias=False),
            nn.BatchNorm2d(channels)
        )
    def forward(self, x): return x + self.block(x)

# Генератор: Low Res -> Super Res
class Generator(nn.Module):
    def __init__(self, res_blocks=4):
        super(Generator, self).__init__()
        # Вхідний шар
        self.conv1 = nn.Sequential(nn.Conv2d(1, 64, 9, 1, 4),
nn.PReLU())

        # Послідовність залишкових блоків (ResNet backbone)
        self.res_blocks = nn.Sequential(*[ResidualBlock(64) for _ in
range(res_blocks)])
```

```
        self.conv2 = nn.Sequential(nn.Conv2d(64, 64, 3, 1, 1,
bias=False), nn.BatchNorm2d(64))

        # Upsampling шари (PixelShuffle): 7 -> 14 -> 28
        self.upsample = nn.Sequential(
            nn.Conv2d(64, 256, 3, 1, 1), nn.PixelShuffle(2),
nn.PReLU(),
            nn.Conv2d(64, 256, 3, 1, 1), nn.PixelShuffle(2),
nn.PReLU()
        )

        # Вихідний шар
        self.conv3 = nn.Conv2d(64, 1, 9, 1, 4)
        self.tanh = nn.Tanh()

    def forward(self, x):
        out1 = self.conv1(x)
        out = self.res_blocks(out1)
        out = self.conv2(out) + out1 # Skip connection через весь
блок
        out = self.upsample(out)
        return self.tanh(self.conv3(out))

# Дискримінатор: визначає, чи зображення є оригінальним HR, чи
згенерованим SR
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        def block(in_c, out_c, stride):
            return nn.Sequential(
                nn.Conv2d(in_c, out_c, 3, stride, 1),
                nn.BatchNorm2d(out_c),
                nn.LeakyReLU(0.2, inplace=True)
```

```
            )

        self.net = nn.Sequential(
            nn.Conv2d(1, 64, 3, 1, 1), nn.LeakyReLU(0.2,
inplace=True),
            block(64, 64, 2),   # 14x14
            block(64, 128, 1),
            block(128, 128, 2), # 7x7
            block(128, 256, 1),
            block(256, 256, 2), # 4x4 (padding/kernels можуть дати 3
або 4, адаптуємо пулінгом)
            nn.AdaptiveAvgPool2d(1),
            nn.Flatten(),
            nn.Linear(256, 1),
            nn.Sigmoid()
        )

    def forward(self, x): return self.net(x)

# 3. Ініціалізація
netG, netD = Generator().to(Config.device),
Discriminator().to(Config.device)
optG = optim.Adam(netG.parameters(), lr=Config.lr)
optD = optim.Adam(netD.parameters(), lr=Config.lr)
criterion_GAN = nn.BCELoss() # Adversarial Loss
criterion_Content = nn.MSELoss() # Pixel-wise Loss (замість VGG для
MNIST)

# 4. Цикл навчання
print("Розпочинаємо навчання SRGAN (x4 Upscaling)...")

g_losses, d_losses = [], []

for epoch in range(Config.epochs):
```

```
    for i, (lr_imgs, hr_imgs) in enumerate(dataloader):
        lr_imgs, hr_imgs = lr_imgs.to(Config.device),
hr_imgs.to(Config.device)
        batch_size = lr_imgs.size(0)

        real_label = torch.ones(batch_size, 1, device=Config.device)
        fake_label = torch.zeros(batch_size, 1, device=Config.device)

        # --- Тренування Генератора ---
        optG.zero_grad()
        sr_imgs = netG(lr_imgs) # Super-Resolved зображення

        # Loss складається з двох частин:
        # 1. Content Loss (щоб цифра виглядала як цифра)
        # 2. Adversarial Loss (щоб текстура виглядала чіткою/реальною
для D)
        loss_content = criterion_Content(sr_imgs, hr_imgs)
        loss_adv = criterion_GAN(netD(sr_imgs), real_label)

        # Вага 1e-3 для adversarial частини є стандартною практикою
для стабільності
        loss_G = loss_content + 1e-3 * loss_adv
        loss_G.backward()
        optG.step()

        # --- Тренування Дискримінатора ---
        optD.zero_grad()
        loss_real = criterion_GAN(netD(hr_imgs), real_label)
        loss_fake = criterion_GAN(netD(sr_imgs.detach()), fake_label)
        loss_D = (loss_real + loss_fake) / 2
        loss_D.backward()
        optD.step()

        g_losses.append(loss_G.item())
```

```
        d_losses.append(loss_D.item())

    if (epoch + 1) % 5 == 0:
        print(f"[Epoch {epoch+1}/{Config.epochs}] Loss D:
{loss_D.item():.4f}, Loss G: {loss_G.item():.4f}")

print("Навчання завершено. Формуємо результати.")

# 5. Візуалізація результатів
netG.eval()
with torch.no_grad():
    # Беремо тестовий батч
    test_lr, test_hr = next(iter(dataloader))
    test_lr, test_hr = test_lr.to(Config.device)[:8],
test_hr.to(Config.device)[:8]
    generated_sr = netG(test_lr)

    # Інтерполюємо LR для візуального порівняння (Bicubic Upscale)
    bicubic_sr = nn.functional.interpolate(test_lr, scale_factor=4,
mode='bicubic', align_corners=False)

    # Об'єднуємо: Low Res (Upscaled) | SRGAN Output | Original High
Res
    img_grid = torch.cat((bicubic_sr, generated_sr, test_hr), dim=2)

    plt.figure(figsize=(15, 6))
    plt.axis("off")
    plt.title("LR (Bicubic) vs SRGAN vs Original HR")
    plt.imshow(np.transpose(make_grid(img_grid.cpu(), nrow=8,
padding=2, normalize=True), (1, 2, 0)), cmap='gray')
    plt.show()
```

Edge GAN Pi

```
import os, time, psutil, torch, torch.nn as nn, torch.optim as optim
from torchvision import datasets, transforms
from torch.utils.data import DataLoader

# 1. Конфігурація середовища
# Ми використовуємо CPU, оскільки це база для подальшого квантування
(Quantization) на архітектурі ARM
class Config:
    batch_size = 64        # Оптимізовано для обмеженої RAM Pi 5
    lr = 0.0002
    epochs = 5             # Достатньо для Proof-of-Concept
    latent_dim = 100
    device = torch.device("cpu")
    model_dir = "models"

os.makedirs(Config.model_dir, exist_ok=True)

# 2. Завантаження даних
# Використовуємо стандартну нормалізацію для MNIST [-1, 1]
dataloader = DataLoader(
    datasets.MNIST('./data', train=True, download=True,
transform=transforms.Compose([
        transforms.ToTensor(), transforms.Normalize((0.5,), (0.5,))
    ])),
    batch_size=Config.batch_size, shuffle=True
)

# 3. Архітектура моделей (Ready for Quantization)
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
```

```
        # Додаємо QuantStub/DeQuantStub для майбутньої підтримки
Quantization-Aware Training (QAT)
        self.quant = torch.quantization.QuantStub()
        self.dequant = torch.quantization.DeQuantStub()

        self.net = nn.Sequential(
            nn.Linear(Config.latent_dim, 256), nn.ReLU(True),
            nn.Linear(256, 512), nn.ReLU(True),
            nn.Linear(512, 784), nn.Tanh()
        )

    def forward(self, x):
        # Потік даних: Квантування входу -> Обробка -> Деквантування
виходу
        x = self.quant(x)
        x = self.net(x)
        x = self.dequant(x)
        return x.view(-1, 1, 28, 28)

class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.net = nn.Sequential(
            nn.Linear(784, 512), nn.LeakyReLU(0.2, inplace=True),
            nn.Linear(512, 256), nn.LeakyReLU(0.2, inplace=True),
            nn.Linear(256, 1),   nn.Sigmoid()
        )

    def forward(self, x):
        return self.net(x.view(-1, 784))

# 4. Ініціалізація
G, D = Generator().to(Config.device),
Discriminator().to(Config.device)
```

```
opt_G = optim.Adam(G.parameters(), lr=Config.lr)
opt_D = optim.Adam(D.parameters(), lr=Config.lr)
criterion = nn.BCELoss()

# 5. Функції моніторингу (Profiling)
def get_resource_usage():
    # Вимірюємо споживання пам'яті процесом у МБ
    process = psutil.Process(os.getpid())
    return process.memory_info().rss / 1024 / 1024

def measure_inference(model, input_tensor):
    # Вимірюємо затримку (latency) на CPU
    start = time.time()
    with torch.no_grad():
        _ = model(input_tensor)
    return (time.time() - start) * 1000  # ms

# 6. Цикл навчання
print(f"Розпочинаємо навчання на {Config.device} (RPi 5)...")

for epoch in range(Config.epochs):
    start_epoch = time.time()
    for real_imgs, _ in dataloader:
        bs = real_imgs.size(0)
        real_imgs = real_imgs.to(Config.device)
        real_labels = torch.ones(bs, 1).to(Config.device)
        fake_labels = torch.zeros(bs, 1).to(Config.device)

        # --- Тренування Дискримінатора ---
        opt_D.zero_grad()
        loss_real = criterion(D(real_imgs), real_labels)

        z = torch.randn(bs, Config.latent_dim).to(Config.device)
        fake_imgs = G(z)
```

```
        loss_fake = criterion(D(fake_imgs.detach()), fake_labels)

        loss_D = loss_real + loss_fake
        loss_D.backward()
        opt_D.step()

        # --- Тренування Генератора ---
        opt_G.zero_grad()
        loss_G = criterion(D(fake_imgs), real_labels)
        loss_G.backward()
        opt_G.step()

    print(f"Epoch [{epoch+1}/{Config.epochs}] | Loss D:
{loss_D.item():.4f} | Loss G: {loss_G.item():.4f} | Time:
{time.time()-start_epoch:.1f}s")

# 7. Збереження та профілювання
save_path = os.path.join(Config.model_dir, "gan_mnist_fp32.pth")
torch.save(G.state_dict(), save_path)
print(f"\nМодель збережено: {save_path}")

print("Розпочинаємо профілювання інференсу...")
G.eval()
test_input = torch.randn(1, Config.latent_dim).to(Config.device)

# "Прогрів" кешу CPU
G(test_input)

latency = measure_inference(G, test_input)
ram_usage = get_resource_usage()

print(f"Inference Latency (CPU): {latency:.2f} ms")
print(f"RAM Usage: {ram_usage:.2f} MB")
```

Додаток В

Акти впровадження та використання результатів дисертаційної роботи

UNIVERSITÄT DES SAARLANDES

DAAD
Ostpartnerschaftsprogramm

BSNU

ACT-CERTIFICATE

about the main results' implementing of the project "Modeling, comparative analysis and design of intelligent mobile robots and their components for movement on various ferromagnetic surfaces"

Forschungsprojekt im Rahmen der Kooperation zwischen der Universität des Saarlandes und der Petro-Mohyla-Schwarzmeeruniversität, Mykolajiv für den Zeitraum 2018-2020

1. Principal Investigators

Petro-Mohyla-Schwarzmeeruniversität, Mykolajiv
Name: Yuriy P. KONDRATENKO
Prof. Dr.Sc.
Address:
Petro Mohyla Black Sea State University
10, 68th Desantnykiv Str.
Mykolaiv, 54003, Ukraine

Email: y_kondrat2002@yahoo.com,
Tel.: +380 512 464074
Fax: +380 512 500333, +380 512 500069

Universität des Saarlandes
Name: Joachim RUDOLPH
Prof. Dr.-Ing.
Address: Universität des Saarlandes
Lehrstuhl für Systemtheorie und
Regelungstechnik, Campus A5 1
66123, Saarbrücken,
Germany
Email: j.rudolph@lsr.uni-saarland.de
Tel.: +49 (0) 681 302-64721
Fax: +49 (0) 681 302-64722

2. Ukrainian team members

Zaporozhets Yuriy, Ph.D., Professor;
Kondratenko Galyna, Ph.D., Associate professor;
Kozlov Oleksiy, Ph.D., Associate professor;
Gerasin Oleksandr, Ph.D.;
Topalov Andriy, Ph.D.;
Taranov Mykyta, Ph.D. student;
Skakodub Oleksandr, Ph.D. student;
Striuk Oleksandr, Ph.D. student.

German team members

Christian Wolf, PhD candidate

3. Main research results.

1) The hybrid multi-agent method and information technology for parametric optimization of fuzzy speed control system for the multi-purpose caterpillar mobile robot (MR) able to move along inclined and vertical ferromagnetic surfaces is developed, that combines the advantages of particle swarm algorithms and local search algorithms based on the elite strategy. The obtained method and information technology allow to effectively optimize various parameters of fuzzy control systems for providing high quality indicators of the MR speed control.

UNIVERSITÄT DES SAARLANDES

DAAD
Ostpartnerschaftsprogramm

BSNU

2) Step-by-step neuroevolution based method and information technology for designing the automatic control system with neural network controller of the spatial motion of caterpillar mobile robot able to move on inclined and vertical ferromagnetic surfaces is obtained, that use genetic algorithm and complex fitness function considering quality of control for both output coordinates (speed and angle of the MR). Proposed method and information technology has allowed to design efficient neural network controller for achieving high quality indicators of spatial motion control taking into account the mutual influence of control channels of the MR's speed and angle that confirms its high efficiency.

3) Combination-generation method for computing geometrical parameters of clamping forces patches localization is proposed taking into account the problem of clamping force calculation and earlier investigated main features of electromagnetic interaction between the mobile robot's electromagnetic driver and ferromagnetic surface. Obtained results show rather high processing accuracy and ability to be used in the experimental setup for fast Hall sensors' data stream computing at clamping force calculation by computerized tools.

4) A series of new dependable capacitive slip displacement sensors with a conical configuration of the registration element, which can operate under corrosive and harsh industrial conditions, are developed. The proposed technical solutions and mathematical models generally improve reliability of robot's operation in complicated environments, help at the designing and choosing of the most appropriate sensor's parameters in industrial conditions for the best grasping manipulated objects.

5) The experimental model of the remote automatic control system based on Internet of things approach is designed for caterpillar MRs, that allows to control the spatial movement of the robots in "point-to-point" network as well as from any place of the world in the presence of access to the Internet. The usage of the given system makes it possible to remotely access to the group robot-control and monitoring processes of outside experts while insufficient qualifications of an attendant.

6) 31.10.2019: defense of the O.S. Gerasin's thesis "Models and means of the control systems of multipurpose mobile robots on ferromagnetic surfaces" for the obtaining the scientific degree of the Candidate of Technical Sciences by the specialty 05.13.07 – automation of control processes in Odessa National Polytechnic University, Ministry of Education and Science of Ukraine.

7) The main results of the project are presented and published by joint Ukrainian- German teams (including Scopus-publications):

- Yuriy Kondratenko, Yuriy Zaporozhets, Joachim Rudolph, Oleksandr Gerasin, Andriy Topalov, Oleksiy Kozlov: Modeling of clamping magnets interaction with ferromagnetic surface for wheel mobile robots. *International Journal of Computing*, Vol.17, Issue 1, Open Access, 2018, pp. 33-46. *(Scopus)*
- O.S. Gerasin, O.V. Kozlov, G.V. Kondratenko, J. Rudolph, Y.P. Kondratenko: Neural controller for mobile multipurpose caterpillar robot. *Proceedings of the 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Vol. 1. Metz, France. 2019, pp. 222-227. *(Scopus)*
- Y. P. Kondratenko, A. V. Kozlov: Parametric optimization of fuzzy control systems based on hybrid particle swarm algorithms with elite strategy, *Journal of Automation and Information Sciences, Vol. 51, Issue 12*, New York: Begel House Inc., 25-45, 2019. *(Scopus)*
- Y.Kondratenko, O.Kozlov, O.Gerasin: Neuroevolutionary approach to control of complex multicoordinate interrelated plants. *International Journal of Computing*, Vol. 18, Issue 4, Open Access, 2019, pp. 502-514. *(Scopus)*
- O. Gerasin, Y. Zaporozhets, Y. Kondratenko: Models of magnetic driver interaction with ferromagnetic surface and geometric data computing for clamping force localization patches. *Proceedings of the IEEE Second International Conference on Data Stream Mining & Processing*, August 21-25, 2018, Lviv, Ukraine, pp. 44-49. *(Scopus)*
- O. Gerasin, Y. Kondratenko, A. Topalov: Dependable robot's slip displacement sensors based on capacitive registration elements. *Proceedings of the 9th IEEE International Conference on Dependable Systems, Services and Technologies (DESSERT'2018)*, 24-27 May, 2018, Kyiv, Ukraine, pp. 378-383. *(Scopus)*

UNIVERSITÄT DES SAARLANDES

DAAD
Ostpartnerschaftsprogramm

BSNU

- O.S. Gerasin, A.M. Topalov, M.O. Taranov, O.V. Kozlov, Y.P. Kondratenko: Remote IoT-based control system of the mobile caterpillar robot. *CEUR Workshop Proceedings*, Vol. 2740, 2020, pp. 129-136. *(Scopus)*

- O.V. Kozlov, O.S. Gerasin, Y.P. Kondratenko, V.O. Kushnir: Automation of the monitoring and control processes of a mobile robot for processing of large inclined surfaces. *SHIPBUILDING & MARINE INFRASTRUCTURE*, Vol. 1, Issue 9, 2018, pp. 59-66.

- Ye.A. Balitskyi, O.S. Gerasin: Remote intelligent control system of mobile robot. *Proceedings of the All-Ukrainian scientific-practical conference of higher education seekers and young scientists "Automation and Electrical Engineering"*, November 15-16, 2018, Mykolaiv, Ukraine: NUS, 2018, p. 16. (in Ukrainian)

- O.S. Gerasin, O.V. Kozlov, Y.P. Kondratenko, O.S. Skakodub: Mathematical modeling of multipurpose caterpillar mobile robot for vertical movement. Scientific notes of V.I. Vernadsky Taurida National University. Series: technical sciences. Vol. 30, Issue 69, Part 1, No. 3, 2019, pp. 70 - 79. (in Ukrainian)

- M.V. Tarabcev, A.S. Gerasin: Features of control of the mobile robot's clamping force. *Proceedings of the All-Ukrainian Scientific and Technical Conference "Modern problems of automation and electrical engineering – 2019"*, April 5-6, 2018, Mykolaiv, NUS, pp. 8-10. (in Russian)

- O.V. Kozlov, Y.P. Kondratenko, O.I. Trilevich: Synthesis and optimization of fuzzy control systems of multipurpose mobile robots. *Mohyla's readings – 2020: Proceedings of the All-Ukrainian scientific-practical conference*, Mykolaiv, ChNU, 2020, pp. 11-13. (in Ukrainian)

- O.S. Gerasin, O.S. Povorozniuk, A.M. Topalov, B.O. Kilimanov: Computer model of the caterpillar mobile robot for automated execution of technological operations on the ship hull. *Proceedings of the XI International Scientific and Technical Conference "Innovations in shipbuilding and ocean engineering"*, Part 2, Mykolaiv, NUS, 2020, pp. 152-155. (in Ukrainian)

- Patent of Ukraine No. 125523. Propulsion wheel of mobile robot. Y.P. Kondratenko, Y.M. Zaporozhets, O.S. Gerasin, G.V. Kondratenko; published 10.05.2018, Bul. No. 9. (in Ukrainian)

- Patent of Ukraine No. 126444. Method of magnetically operated displacement of mobile robot / Y.P. Kondratenko, Y.M. Zaporozhets, O.S. Gerasin, M.O. Taranov; published 25.06.2018, Bul. No. 12. (in Ukrainian)

Place, date, signatures

From the Petro-Mohyla Black Sea National University,

Mykolaiv, February 15, 2021 — Yuriy P. KONDRATENKO

From Saarland University,

Saarbrücken, February 15, 2021 — Joachim RUDOLPH

Lehrstuhl für
Systemtheorie und Regelungstechnik
Universität des Saarlandes, Geb. A5 1
Postfach 151150 · 66041 Saarbrücken
Prof. Dr.-Ing. habil. J. Rudolph

«ЗАТВЕРДЖУЮ»
В. о. ректора
Чорноморського національного
університету ім. Петра Могили
проф. Леонід КЛИМЕНКО
«15» 04 2026 р.

АКТ
впровадження у навчальний процес результатів дисертаційної роботи на здобуття наукового ступеня доктора філософії Стрюка Олександра Сергійовича

Комісія у складі: канд. техн. наук, доцентки Бойко А. П. – деканки факультету комп'ютерних наук ЧНУ ім. П. Могили, канд. техн. наук, доцента Сіденка Є. В. – завідувача кафедри інтелектуальних інформаційних систем ЧНУ ім. П. Могили, д-ра техн. наук, професора Козлова О. В. – професора кафедри інтелектуальних інформаційних систем ЧНУ ім. П. Могили, підписала даний акт про те, що основні теоретичні положення наукових досліджень аспіранта кафедри інтелектуальних інформаційних систем Стрюка О. С., проведених в рамках дисертаційної роботи, впроваджені та використовуються в навчальному процесі ЧНУ ім. П. Могили, а саме:

– принципи проєктування спеціалізованих моделей штучних нейронних мереж – при викладанні дисципліни «Проєктування інтелектуальних СППР»;

– програмно-алгоритмічні засоби систем побудови, оптимізації та налаштування генеративних змагальних моделей, зокрема згорткових мереж і систем генеративного ШІ – при викладанні дисциплін «Нейромережеві методи обчислювального інтелекту»;

– проєктування генеративних змагальних мереж у контексті моделювання нечітких сценаріїв і покращення точності та гнучкості нечітких моделей – при викладанні дисципліни «Fuzzy models and methods for computational intelligence».

Результати дисертаційної роботи використовуються, зокрема, в лекційному матеріалі та лабораторних роботах, а також при виконанні кваліфікаційних робіт здобувачами за спеціальностями 123 «Комп'ютерна інженерія» та 122 «Комп'ютерні науки».

Деканка факультету комп'ютерних наук ЧНУ ім. П. Могили, канд. техн. наук, доцентка — Анжела БОЙКО

Завідувач кафедри ІІС ЧНУ ім. П. Могили, канд. техн. наук, доцент — Євген СІДЕНКО

Професор кафедри ІІС ЧНУ ім. П. Могили, д-р техн. наук, професор — Олексій КОЗЛОВ

ІНСТИТУТ ПРОБЛЕМ ШТУЧНОГО ІНТЕЛЕКТУ
МІНІСТЕРСТВА ОСВІТИ І НАУКИ УКРАЇНИ
І НАЦІОНАЛЬНОЇ АКАДЕМІЇ НАУК УКРАЇНИ
вул. М. Житомирська, 11, офіс 5а, м. Київ, 01001
(тел./факс): +38 044 278-37-59, (тел./факс): +38 044 248 06 23
E-mail: ipai.kiev@gmail.com Код ЄДРПОУ 02095826

Від 15.12.2023 № 147/01-12
На № _____ від _____

Ректору Чорноморського національного університету ім. Петра Могили
д.т.н. професору Клименку Л.П.
б. 10, вул. 68 Десантників,
Миколаїв, 54003, Україна

Вельмишановний Леоніде Павловичу !

Повідомляю Вас про участь аспіранта Чорноморського національного університету імені Петра Могили Стрюка Олександра Сергійовича у виконанні держбюджетної науково-дослідної теми «Створення стратегії розвитку штучного інтелекту в Україні» (2020-2022 рр., ДР 0120U102299) на замовлення Міністерства освіти і науки України у складі робочої групи д.т.н., професора Кондратенка Ю.П.

Аспірант Стрюк О.С. поєднав результати своєї дисертаційної роботи, зокрема, пов'язані з генеративними змагальними мережами - ГЗМ, з науково-практичною роботою під час розробки Стратегії розвитку штучного інтелекту в Україні. Результати його дисертаційних досліджень в цьому напрямку допомогли при формулюванні стратегічних рішень та практичних застосувань у контексті регуляції та контролю штучного інтелекту.

Пропозиції і доповнення Стрюка О. С., що стосувались машинного/глибокого навчання, генеративних моделей і загальних методів штучного інтелекту були включені у відповідні розділи Стратегії (Загальна парадигма. Основні напрями досліджень штучного інтелекту. Мета і завдання Стратегії розвитку штучного інтелекту в Україні (2023 – 2030). Штучний інтелект у сфері безпеки і оборони, у промисловості та енергетиці, у сільському господарстві. Наукове, кадрове та матеріальне забезпечення національної екосистеми штучного інтелекту. Прикінцеві положення Стратегії розвитку штучного інтелекту в Україні).

Результати роботи знайшли відображення у опублікованій колективній монографії «Стратегія розвитку штучного інтелекту в Україні», ISBN: 978-617-7894-89-5, https://doi.org/10.15407/development_strategy_2023 та у відповідній статті у журналі «Regarding the Draft Strategy Development of Artificial Intelligence in Ukraine (2022 – 2030)». Штучний інтелект, №1, 2022. С. 8-157, співавтором яких є аспірант Стрюк О.С.

Висловлюємо вдячність за внесок групи професора Кондратенка Ю.П. у наукову складову та підтримку ініціатив, пов'язаних із дослідженням проблем штучного інтелекту і його нормативної регуляції, як в рамках розробки Стратегії, так і поза межами проєкту.

З повагою,
член-кореспондент НАН України
доктор технічних наук, професор,
директор Інституту проблем штучного
інтелекту МОН і НАН України А.І. Шевченко