

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА

В. М. Бредіхін, В. О. Дідок, В. В. Черкасова

**РОЗВИТОК АРХІТЕКТУР НЕЙРОННИХ МЕРЕЖ:
ВІД КЛАСИЧНИХ ДО ІННОВАЦІЙНИХ**

МОНОГРАФІЯ

Харків
ХНУМГ ім. О. М. Бекетова
2025

УДК 004.7+004.8
Б87

Автори:

Бредіхін Володимир Михайлович, кандидат технічних наук, доцент;
Дідок Володимир Олегович, асистент;
Черкасова Валерія Володимирівна, асистент

Рецензенти:

Старкова Ольга Володимирівна, доктор технічних наук, професор, завідувач кафедри кібербезпеки та інформаційних технологій Харківського національного економічного університету імені Семена Кузнеця;

Міщеряков Юрій Валентинович, кандидат технічних наук, професор, професор кафедри системотехніки Харківського національного університету радіоелектроніки.

Рекомендовано до друку Вченою радою ХНУМГ ім. О. М. Бекетова, протокол № 2 від 29.08.2025.

Бредіхін В. М.

Б87 Розвиток архітектур нейронних мереж: від класичних до інноваційних : монографія / В. М. Бредіхін, В. О. Дідок, В. В. Черкасова ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2025. – 81 с.
ISBN 978-966-695-631-9

У монографії розглянуто еволюцію архітектур штучних нейронних мереж – від класичних моделей (перцептрон, багатошаровий перцептрон) до сучасних глибоких структур, зокрема згорткових (CNN), рекурентних (RNN, LSTM, GRU), трансформерних (BERT, GPT, ViT), мобільних та гібридних архітектур. Узагальнено їхні переваги, обмеження, а також прикладні сфери використання. Обґрунтовано критерії вибору архітектури залежно від типу даних, обчислювальних ресурсів і завдань класифікації. Наведено порівняльний аналіз моделей за точністю, швидкістю навчання та ефективністю.

Монографія призначена для викладачів, аспірантів і студентів ІТ-спеціальностей, а також фахівців з машинного навчання, комп'ютерного зору та інтелектуальних інформаційних систем.

УДК 004.7+004.8

ISBN 978-966-695-631-9

© В. М. Бредіхін, В. О. Дідок,
В. В. Черкасова
© ХНУМГ ім. О. М. Бекетова, 2025

ЗМІСТ

ПЕРЕДМОВА	5
ВСТУП	6
1 КЛАСИЧНІ АРХІТЕКТУРИ	9
1.1 Перцептрон (1958)	9
1.2 Багатошаровий перцептрон (MLP)	11
2 ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ (CONVOLUTIONAL NEURAL NETWORKS, CNN)	14
2.1 LeNet (1998)	15
2.2 AlexNet (2012)	16
2.3 VGG (2014)	18
2.4 ResNet (2015)	20
2.5 EfficientNet (2019)	22
2.6 Висновки щодо CNN	24
3 РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ (RNN)	26
3.1 Vanilla RNN	27
3.2 LSTM (1997)	28
3.3 GRU (2014)	30
3.4 Порівняння й інтеграція RNN-архітектур	32
3.5 Практичне застосування та місце RNN в сучасній екосистемі	34
3.6 Висновки щодо RNN	35
4 ТРАНСФОРМЕРИ (TRANSFORMERS)	37
4.1 Original Transformer (2017)	38
4.2 BERT (2018)	39
4.3 GPT (2018–2023)	41
4.4 Vision Transformers (ViT, 2020)	43
4.5 Вплив трансформерів і тенденції розвитку	44
4.6 Висновки щодо трансформерів	46
5 СПЕЦІАЛІЗОВАНІ ТА МОБІЛЬНІ АРХІТЕКТУРИ	49
5.1 MobileNet (2017)	49

5.2 NASNet (2018)	52
5.3 Взаємодоповнення в екосистемі глибинного навчання	54
5.4 Висновки щодо спеціалізованих та мобільних архітектур.....	55
6 ІНТЕГРОВАНІ ПІДХОДИ	57
6.1 Hybrid Models (поєднання CNN і RNN).....	57
6.2 Ensemble Architectures	59
6.3 Переваги та виклики інтегрованих рішень.....	61
6.4 Висновки щодо інтегрованих підходів	62
7 ПОРІВНЯННЯ АРХІТЕКТУР.....	64
ВИСНОВОК	69
Висновки щодо архітектури для різних типів задач	72
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ	75
ДОДАТОК А.....	78

ПЕРЕДМОВА

Упродовж останніх десятиліть штучні нейронні мережі пройшли шлях від простих моделей, що імітували функціонування біологічного нейрона, до складних архітектур, здатних вирішувати надскладні завдання класифікації, прогнозування, генерації та розуміння мовних, візуальних і часових патернів.

Метою цієї монографії є створення систематизованого огляду архітектур нейронних мереж, що використовуються у практиці глибинного навчання, з акцентом на їхній історичний розвиток, принципи побудови та сфери застосування.

Монографія створена з урахуванням потреб кількох цільових груп:

- студентів та аспірантів, які вивчають курси з машинного навчання, штучного інтелекту, Data Science;
- викладачів, які розробляють навчальні програми з сучасних ІТ-дисциплін;
- дослідників, що працюють у напрямках розробки нових архітектур нейронних мереж;
- практиків, які інтегрують ШІ-рішення в індустріальні або прикладні ІТ-системи.

У монографії розглянуто не лише теоретичні основи архітектур, а й приклади їхнього практичного використання, наведено порівняльні характеристики, що допомагають обґрунтовано обирати відповідну модель залежно від задачі, обсягу даних, вимог до ресурсів та очікуваного рівня точності.

Автори сподіваються, що поданий матеріал стане корисним довідковим і навчальним ресурсом, сприятиме поглибленому розумінню принципів функціонування нейронних мереж.

З повагою, Авторський колектив.

ВСТУП

Задача класифікації у машинному навчанні є одним із центральних напрямів розвитку сучасних обчислювальних систем, оскільки вона лежить в основі широкого спектра завдань: від розпізнавання образів у зображеннях та класифікації тексту до детектування хвороб у медичних дослідженнях. Її важливість полягає не лише в академічній цікавості, а й у глибокому практичному застосуванні: фінансові установи покладаються на алгоритми класифікації для виявлення шахрайства, медичні лабораторії – для діагностики на підставі знімків томографії, а соціальні мережі – для автоматичного моделювання користувацьких вподобань. У цих контекстах надзвичайно важливим стає питання обрання такої архітектури нейронної мережі, яка здатна забезпечити високу точність, ефективне використання обчислювальних ресурсів і високу швидкість навчання.

З огляду на різноманітність сучасних наборів даних та варіативність апаратних конфігурацій (від масивних GPU-кластерів до мікроконтролерів), дослідники розробили низку різнопланових архітектур, що оптимізують роботу під конкретні обмеження. Історично перші підходи до реалізації нейронних мереж були орієнтовані переважно на лінійно роздільні простори ознак. Проте, із розвитком апаратного забезпечення та появою великих обсягів даних, сфера застосування розширилася настільки, що сьогодні архітектури нейронних мереж проникають у найрізноманітніші галузі. Різні модифікації дозволяють навчатися на величезних наборах зображень, текстів чи навіть складних багатовимірних часових рядах, які кілька десятиріч тому було вкрай важко аналізувати ефективно.

Більше того, загальні підходи до моделювання, апробовані в нейронних мережах, знаходять своє застосування і в абсолютно різних

сферах, таких як управління сталим розвитком чи аналіз відтворення основних фондів у муніципальних системах.

Вибір архітектури залежить від низки чинників. По-перше, це продуктивність: у деяких сценаріях критичною є максимальна точність, а в інших – баланс між швидкістю прогнозування та обчислювальними витратами. По-друге, велике значення має обчислювальна ефективність. Класичні мережі на кшталт багат шарового перцептрона (MLP) вимагають значної кількості параметрів, що може перевищувати можливості мобільних пристроїв. Це призвело до активного розвитку легких архітектур, де кількість параметрів свідомо зменшується (MobileNet, GhostNet та ін.) [7; 9]. По-третє, різні методи неоднаково чутливі до великого обсягу даних чи, навпаки, до їхнього браку. Зокрема, глибокі згорткові мережі здатні обробляти мільйони зображень, тоді як менші за розміром мережі більше підходять для застосувань із малим набором даних.

У цьому посібнику буде викладено історію розвитку та принципові характеристики основних архітектур для задачі класифікації. Починаючи з найпростіших моделей (перцептрон, MLP), перейдемо до глибоких згорткових мереж (LeNet, AlexNet, VGG, ResNet, EfficientNet), рекурентних моделей (RNN, LSTM, GRU), трансформерів (Original Transformer, BERT, GPT, ViT), а також розглянемо спеціалізовані рішення (MobileNet, NASNet) та інтегровані підходи. При цьому приділимо особливу увагу практичним факторам на кшталт точності, швидкості, апаратних вимог і зручності розгортання.

Крім того, буде наведено приклади вдалих застосувань відповідних архітектур: від класифікації зображень із використанням ResNet до послідовної обробки тексту за допомогою LSTM чи трансформерів. Також подано літературні джерела [1–13], щоб проілюструвати історичний та сучасний контекст цих досліджень і продемонструвати, як протягом

останніх десятиліть змінювалися пріоритети в побудові моделей та пошуку нових архітектур.

Отже, первинна мета – створити систематизовану картину існуючих рішень, виокремити сильні та слабкі сторони кожного підходу, а також надати рекомендації щодо використання конкретної архітектури нейронної мережі у задачах класифікації з урахуванням специфіки даних та обчислювальної бази.

1 КЛАСИЧНІ АРХІТЕКТУРИ

Перші спроби розробити штучні нейронні мережі базувалися на ідеї емулювати біологічні нейрони та синапси. Незважаючи на значні обчислювальні та теоретичні обмеження того часу, ці спроби заклали фундаментальні принципи, які стали платформою для подальших еволюційних стрибків. Найвідомішими класичними архітектурами, що сформували науковий ландшафт, вважаються перцептрон (1958) і багат шаровий перцептрон (MLP). Їхній внесок полягає не лише в утвердженні загальної структури «нейронів» і «шарів», а й у формуванні базових уявлень про навчання на основі градієнтних методів, яке потім було розвинуто у більш складних мережах.

1.1 Перцептрон (1958)

Перцептрон, запропонований Френком Розенблаттом, був одним із перших формальних підходів до машинного навчання та штучного інтелекту загалом. Його структура включала вхідні сигнали, вагові коефіцієнти та лінійну функцію активації з порогом. Ідея полягала в тому, що у випадку лінійно роздільних класів можна підібрати таку гіперплощину, яка коректно відокремлюватиме приклади одного класу від прикладів іншого. Перцептронні правила навчання оновлюють ваги під час кожної ітерації, поступово зменшуючи кількість помилок на навчальних даних.

Важливим аспектом перцептрона є його геометрична інтерпретація: ваги нейрона формують вектор у просторі ознак, а вихідний поріг відповідає зміщенню гіперплощини. Така модель виявилася успішною у задачах, де класи справді можна розділити прямою (у двовимірному випадку) або гіперплощиною (у вищих розмірностях). Проте, коли задачі виходили за межі лінійної роздільності, перцептрон неспроможний був

адекватно розв'язати їх. Це спонукало науковців придивитися до багат шарових структур, які б посилювали здатність моделі до представлення більш складних закономірностей.

Для візуалізації базової структури перцептрона часто використовують просту схему (рис. 1). Вона складається з вхідного набору нейронів, кожен з яких подає сигнал до вихідного елемента через вагові коефіцієнти. Якщо сума сигналів перевищує поріг, вихід приймає одне значення, інакше – інше. Саме через такий чіткий пороговий механізм перцептрон у класичному вигляді має обмежену гнучкість.

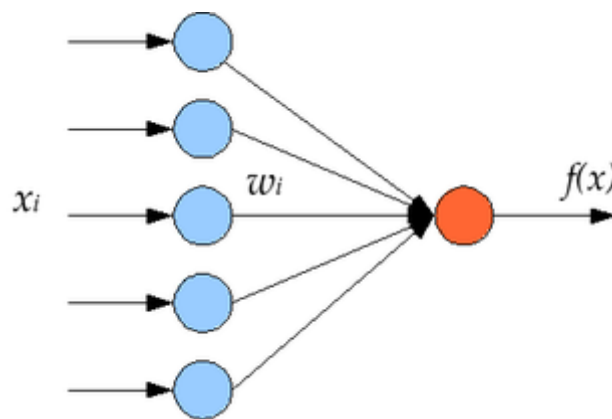


Рисунок 1 – Схема перцептрона, що демонструє найпростішу модель нейронної мережі для лінійно роздільних класів

Незважаючи на цей недолік, перцептронні ідеї залишили глибокий слід у подальших дослідженнях. По-перше, розроблений Розенблаттом алгоритм навчання заклав методологічну основу для градієнтних технік оптимізації, що стали надзвичайно важливими згодом, коли було винайдено багат шарові мережі. По-друге, критичні зауваження, зокрема викладені Мінським і Пейпертом, щодо неможливості розв'язати задачу XOR чисто лінійним класифікатором, дали поштовх до пошуку глибших, нелінійних архітектур.

1.2 Багатошаровий перцептрон (MLP)

Багатошаровий перцептрон (MLP) став ключовою відповіддю на обмеження класичного перцептрона, адже додавання прихованих шарів дозволило мережі набувати здатності до моделювання складних, нелінійних відношень у даних. Загальна структура MLP включає щонайменше три шари: вхідний, прихований (або кілька прихованих) та вихідний. Кожен нейрон у прихованому шарі може розглядатися як фільтр ознак різної складності, що вчиться автоматично на підставі зворотного поширення похибки (backpropagation).

Сучасне застосування MLP зазвичай базується на активаційних функціях, відмінних від класичного порогу. Найпоширенішими варіантами є:

- ReLU (Rectified Linear Unit), яка задає лінійне поведіння для додатних значень та обнуляє всі від'ємні. Її популярність пояснюється тим, що вона зменшує проблему затухання градієнтів у глибоких мережах, дозволяючи ефективніше навчати велику кількість шарів;
- Sigmoid, яка нормує вихід до проміжку (0; 1). Вона свого часу була особливо поширеною в задачах бінарної класифікації та у внутрішніх блоках рекурентних мереж, проте має проблему «насичення», коли градієнти стають дуже малими при великих або малих значеннях на вході;
- Tanh, що подібна за формою до Sigmoid, але виводить значення в діапазоні (-1; 1). Вона частково розв'язує проблему зміщення середнього значення виходу та показує кращу збіжність, утім також схильна до ефекту насичення.

MLP заклав основу для численних майбутніх досліджень з глибинного навчання, коли кількість прихованих шарів зростала, а також з'являлися додаткові елементи — такі як dropout для боротьби з перенавчанням, batch normalization для стабілізації розподілу вхідних сигналів упродовж навчання тощо. Хоча сучасні дослідники часто віддають перевагу складнішим архітектурам, багатошаровий перцептрон

залишається базовою «точкою відліку» в будь-якому огляді нейронних мереж.

Для покращення роботи у задачах нелінійної класифікації необхідно забезпечувати достатню глибину та ширину мережі, аби моделювати складні функції залежності між вхідними ознаками та вихідними категоріями. Багато залежить і від наявності даних: якщо доступний великий тренувальний набір, достатньо глибокі MLP-мережі можуть досягти серйозних результатів. Водночас вони не завжди оптимальні з погляду використання ресурсів, особливо коли порівнювати їх із пізнішими архітектурами на основі згорток чи механізму уваги.

На рисунку 2 наведено узагальнену схему з розподілом шарів на S-, A- та R-елементи.

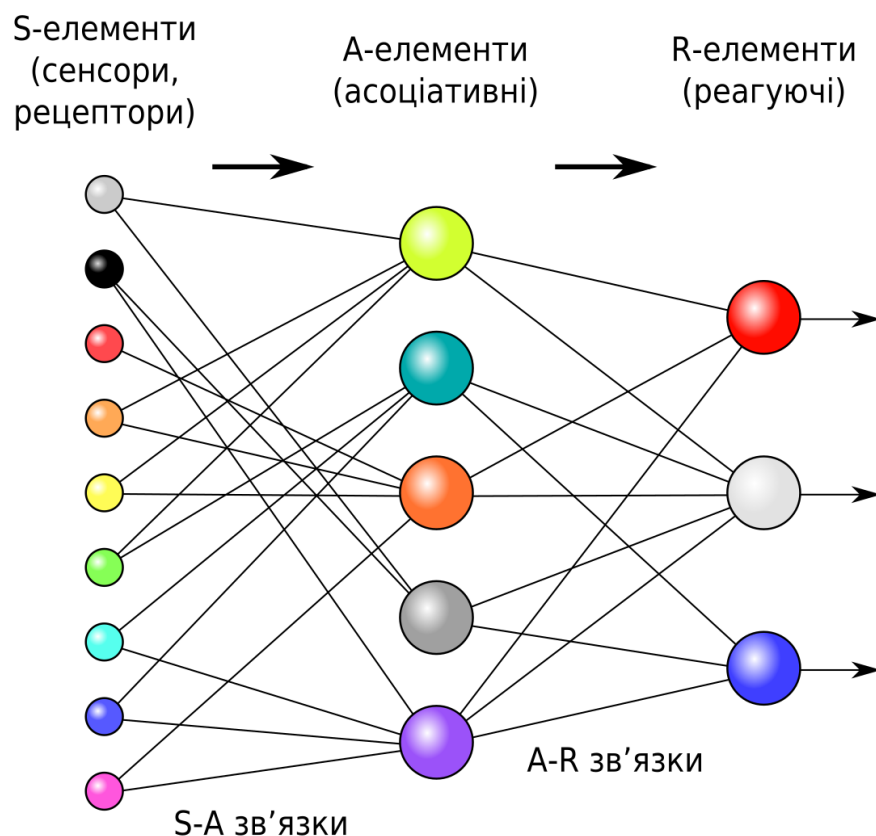


Рисунок 2 – Архітектура багатошарового перцептрона з розподілом на S-, A- та R-елементи для обробки даних

Така схема дозволяє показати послідовність (S, A) обчислень активації кожного шару та (R) перетворень, що мають формувальну функцію для відповідних внутрішніх представлень, по суті, створюючи все складніші ознаки з кожним наступним шаром.

Таким чином, класичні архітектури – перцептрон і багатошаровий перцептрон – відіграли вирішальну роль у становленні нейронних мереж як наукової дисципліни та практичного інструмента. Хоча їхні безпосередні можливості значно поступаються сучасним глибоким моделям (CNN, RNN, Transformers), вони стали тим «наріжним каменем», на якому збудовано всю потужну інфраструктуру глибинного навчання. У наступних розділах буде розглянуто, як ці базові ідеї суттєво розширилися із введенням операцій згортки, рекурентних механізмів та механізму уваги, що дало можливість вирішувати складні завдання класифікації не лише в площині просторових даних, а й у часових серіях та в контексті природномовних текстів.

2 ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ (CONVOLUTIONAL NEURAL NETWORKS, CNN)

Згорткові нейронні мережі, або Convolutional Neural Networks (CNN), сьогодні майже синонімічні з революцією в галузі комп'ютерного зору. Вони продемонстрували виняткову ефективність в аналізі зображень, розпізнаванні об'єктів, семантичній сегментації, а згодом стали незамінним інструментом у суміжних галузях на кшталт обробки сигналів та навіть у завданнях обробки тексту, де також можна інтерпретувати послідовності слів як певний «образ» з просторово-тимчасовою структурою [1; 3]. Концепція згортки, яка полягає в локальній обробці фрагментів вхідних даних з подальшим агрегаційним узагальненням, виявилася надзвичайно плідною саме завдяки тому, що зображення часто мають просторові кореляції: сусідні пікселі зазвичай пов'язані між собою. Мережі з операціями згортки дають змогу фільтрувати ознаки на різних масштабах і глибинах, що робить їх потужним інструментом для розпізнавання складних візуальних патернів.

Історично CNN почали активно розвиватися наприкінці 1980-х – на початку 1990-х років, проте справжній прорив відбувся пізніше. Ідея полягала в тому, щоб використовувати «вікно» фіксованого розміру, яке крокує (згортка) по вхідному зображенню й обчислює лінійну комбінацію піксельних інтенсивностей у певному регіоні. Ці операції повторюються з різними фільтрами, навчаючи мережу автоматично знаходити найоптимальнішу сукупність «шаблонів» для класифікації чи іншого завдання. Після одного або кількох згорткових шарів зазвичай застосовують pooling (найчастіше max pooling чи average pooling), аби зменшити просторову роздільність і зробити модель стійкішою до зсувів і деяких видів деформацій зображень [1]. Наприкінці традиційної CNN-моделі зазвичай використовують один або кілька повнозв'язних (fully

connected) шарів, які виконують функцію класифікатора, базуючись на вилучених мережею ознаках.

Загальну схему згорткової нейронної мережі часто ілюструють у спеціальній літературі через послідовність шарів згортки, pooling і фінального класифікаційного блоку (рис. 3).

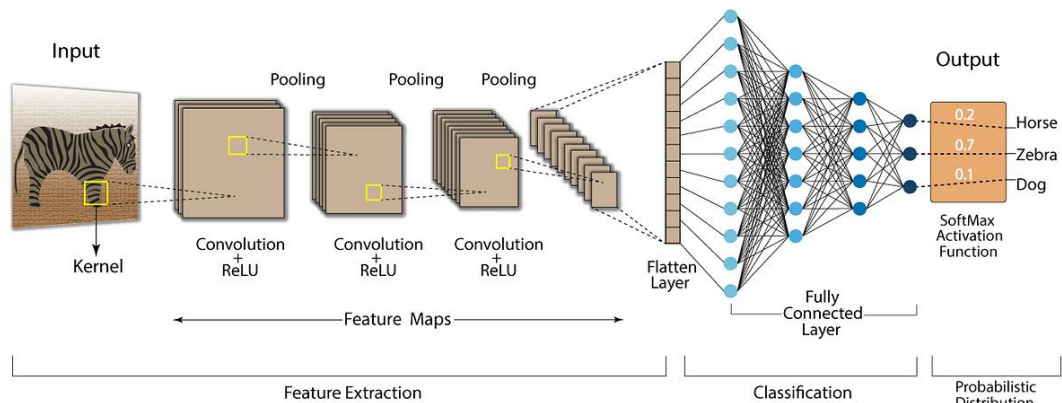


Рисунок 3 – Архітектура згорткової нейронної мережі: від екстракції ознак до класифікації за допомогою повнозв'язного шару та функції SoftMax

Раніше така архітектура застосовувалася переважно до розпізнавання рукописних цифр, але з часом охопила майже всю сферу класифікації зображень. У цьому розділі будуть розглянуті найбільш знакові етапи розвитку CNN: LeNet, AlexNet, VGG, ResNet та EfficientNet. Кожна з цих моделей відкривала нову віху в історії комп'ютерного зору, долаючи бар'єри точності, ефективності або складності навчання.

2.1 LeNet (1998)

LeNet, розроблена Яном ЛеКуном, стала однією з перших згорткових мереж, яка набрала популярності завдяки успішному використанню в розпізнаванні рукописних цифр на зображеннях розміру 28×28 пікселів (база даних MNIST). Основна структура LeNet включала кілька послідовних згорткових шарів, за якими йшли шари субдискретизації (pooling) та один чи два повнозв'язних шари перед виходом. Завдання

полягало в автоматичному вилученні базових ознак, таких як контури чи кути, а також у зменшенні розмірності проміжного простору за допомогою pooling-шарів.

Інноваційність LeNet полягала у використанні локальних рецептивних полів, де кожен нейрон згорткового шару зв'язувався лише з невеличкою ділянкою на вхідному зображенні. Це сприяло скороченню кількості параметрів порівняно з класичними багат шаровими перцептронами, де кожен нейрон у прихованому шарі пов'язаний з усіма пікселями вхідного шару. Така локальність добре узгоджується з природою зображень: локальні фрагменти містять детальну інформацію про текстуру, грані й кути, і їхня комбінація на наступних рівнях дозволяє зібрати глобальне уявлення про об'єкт.

LeNet продемонструвала, що CNN можуть автоматично вчитися витягувати релевантні ознаки без необхідності вдаватися до ручного налаштування фільтрів, як це робилося у попередніх методах комп'ютерного зору (наприклад, у методах, що базувалися на Gabor-фільтрах або на фільтрах Собеля). Завдяки відносно малій кількості шарів LeNet легко навчалася на тодішньому апаратному забезпеченні. Проте, коли постали складніші задачі зображення з більшою роздільністю і більшою кількістю класів, мережа мала недостатню потужність, щоб адекватно розпізнавати патерни більш високого рівня. Незважаючи на це, LeNet стала своєрідним «доказом концепції» та започаткувала хвилю досліджень у напрямку згорткових мереж [1].

2.2 AlexNet (2012)

Справжній прорив щодо застосування CNN у складніших завданнях стався, коли Алекс Крижижевський, Ілля Суцкевер і Джефрі Гінтон представили AlexNet на конкурсі з розпізнавання зображень ImageNet (ILSVRC) у 2012 році. Ця модель досягла різкого зменшення помилки

порівняно з попередніми методами, що покладалися на більш традиційні техніки вилучення ознак.

Ключовим фактором такого успіху було поєднання:

1. Достатньо глибокої архітектури, що містила п'ять згорткових шарів та три повнозв'язних шари.
2. Використання прискорення на графічних процесорах (GPU) для навчання на великому наборі даних ImageNet.
3. Застосування нових прийомів регуляризації, зокрема dropout, для боротьби з перенавчанням.
4. Використання функції активації ReLU, що суттєво пришвидшило збіжність, порівняно з більш традиційними сигмоїдними чи тангісними функціями.

Архітектура цієї мережі складалася зі згорткових шарів з невеликими полями (11×11 , 5×5 , 3×3) та pooling-шарів, які поступово зменшували розмір проміжних представлень, уможливлуючи нарощування кількості фільтрів у глибину. Після фінального pooling-шару дані «розгорталися» у вектор, який ставав вхідним сигналом для повнозв'язних шарів і завершальної класифікаційної частини з використанням SoftMax. Схематичне зображення AlexNet наведено на рисунку 4.

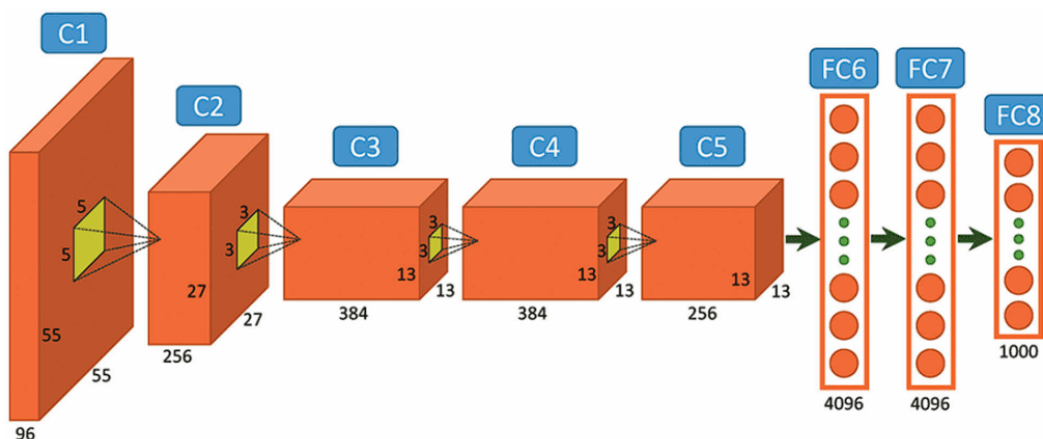


Рисунок 4 – Архітектура AlexNet: п'ять згорткових шарів для екстракції ознак та три повнозв'язних шари для класифікації

Перехід від відносно скромних завдань на кшталт MNIST (чорно-білі зображення 28×28) до набору ImageNet із сотнями тисяч різнокольорових зображень та тисячами класів продемонстрував масштабність CNN. AlexNet виявилася здатною засвоювати складні візуальні ознаки, починаючи від простих контурів і текстур у перших шарах, до «образних» фільтрів високого рівня у глибших шарах, які реагували на специфічні об'єкти на кшталт автомобілів або тварин. У підсумку величезний стрибок у точності збігся із початком «ери глибинного навчання» в комп'ютерному зорі [2].

AlexNet також популяризувала ідею масового використання GPU для розпаралелювання обчислень під час forward- та backward-прогонів, що дозволило тренувати глибші архітектури значно швидше, ніж це було можливо на CPU. Крім того, схеми ініціалізації ваг та прийоми на кшталт LRN (Local Response Normalization) чи Dropout стали обов'язковим елементом багатьох сучасних CNN-архітектур, хоч і згодом були замінені на більш ефективні методи нормалізації й регуляризації, такі як batch normalization [1].

2.3 VGG (2014)

VGG, запропонована Сімоньян і Зіссерманом [14], змінила підхід до дизайну згорткових мереж, пропонуючи просту, але водночас глибоку архітектуру. Основна ідея полягала в тому, щоб використовувати невеликі фільтри 3×3 , але збільшувати кількість шарів до 16 або навіть 19. На відміну від AlexNet, де у перших шарах застосовувалися великі фільтри (наприклад, 11×11), автори VGG показали, що навіть маленькі фільтри, накладені послідовно, здатні імітувати вплив великого фільтра, зберігаючи при цьому менше параметрів. Зросла і глибина мережі, адже додавання послідовних шарів « 3×3 » дає змогу обчислювати все більш абстрактні й складні ознаки.

Архітектура VGG була впорядкована блоками: кожен блок містив кілька послідовних згорткових шарів (зазвичай 2 або 3), після чого йшов шар pooling для зменшення просторових розмірів. Поступово кількість фільтрів у згорткових шарах зростала в геометричній прогресії (64, 128, 256, 512), щоб компенсувати втрату просторової інформації та накопичувати все більшу кількість каналів у глибину. Кінцева частина VGG містила кілька повнозв'язних шарів і SoftMax для класифікації. При цьому зберігалася концептуальна простота: лише згортки 3×3 , pooling 2×2 та стандартний набір повнозв'язних шарів.

Схема такої мережі наведена на рисунку 5. Попри доволі значну кількість параметрів (наприклад, понад 100 мільйонів у VGG-16), мережа досягла високої точності на ImageNet, підтвердивши гіпотезу, що глибина має велике значення для відтворення складних патернів. Однак навчання VGG було ресурсоємним: необхідні не лише обчислювальні потужності, а й чималий обсяг відеопам'яті на графічних картах для зберігання великої кількості параметрів.

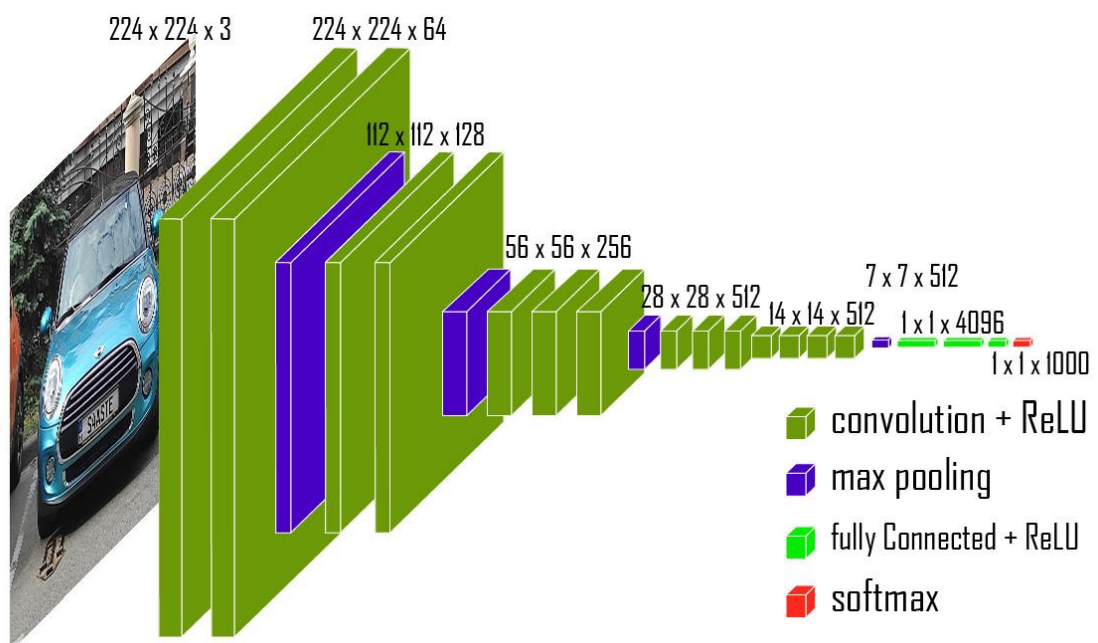


Рисунок 5 – Схема VGG з глибокими мережами та простою архітектурою

Концептуальна прозорість та елегантність роблять VGG важливим етапом у розвитку CNN. По-перше, ідея побудови мережі з однаковим розміром фільтрів у кожному згортковому шарі вплинула на низку подальших досліджень, які експериментували зі зменшенням розмірів фільтрів і збільшенням глибини. По-друге, незважаючи на порівняно великі вимоги до апаратних ресурсів, VGG залишилася популярною базовою архітектурою для задач перенесеного навчання, особливо коли необхідно узагальнити мережу, попередньо натреновану на великому наборі зображень, до нових завдань. Завдяки своїй «модульності» VGG відносно легко модифікувати й розширювати, додаючи, наприклад, сегментаційні «голови» для сегментації зображень чи змінюючи кількість вихідних класів.

2.4 ResNet (2015)

ResNet, або ж «мережа з резидуальними блоками», стала знаковою архітектурою, що вирішила проблему деградації градієнтів у глибоких мережах. Запропонована Хе Каймінем та його колегами, ResNet вирізнялася використанням спеціальних «shortcut connections» (скорочених зв'язків), які пропускали сигнал через один або кілька шарів, дозволяючи формувати функцію не просто $F(x)$ для входу x , а $F(x) + x$. Така конструкція дістала назву резидуального блоку (рис. 6), оскільки завдання глибоких шарів фактично зводилося до того, щоб навчитися моделювати різницю (резидуал) між вхідним сигналом і виходом. Це змінило уявлення про те, наскільки глибокими можуть бути CNN, оскільки ResNet-50, ResNet-101 і ResNet-152 навчилися без критичного затухання або вибуху градієнта.

На ідею авторів ResNet надихнув феномен «мережі пропускають навчання», коли глибші мережі інколи показували нижчу точність, ніж «достатньо глибокі», але не надмірно. Застосування резидуальних блоків

забезпечило механізм, через який градієнт від виходу міг передаватися майже без втрат на попередні шари, оминаючи проблемні ділянки в глибині мережі. У підсумку це дозволило стабільно тренувати мережі зі 100 і більше шарами.

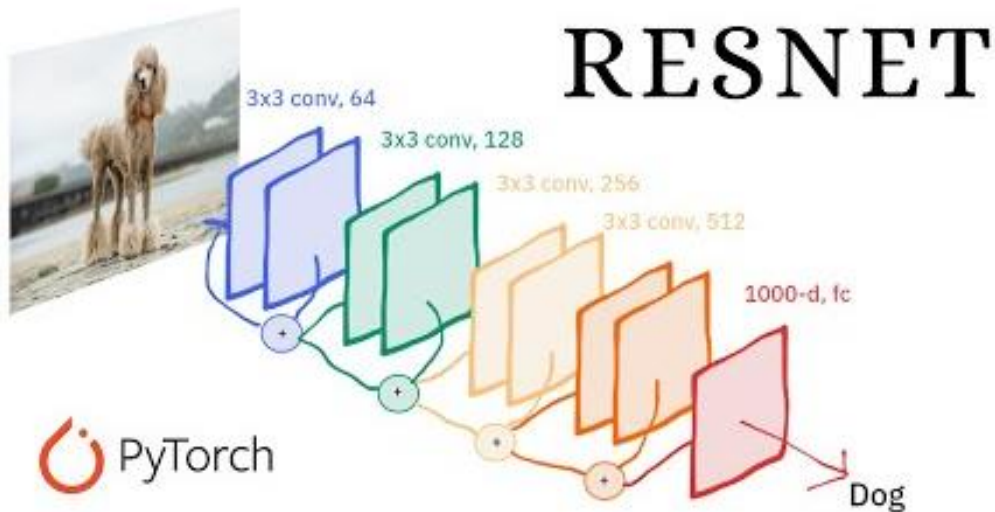


Рисунок 6 – Резидуальні блоки ResNet, що вирішують проблему затухання градієнтів

Архітектура ResNet зберегла ключові особливості успішних попередніх мереж, як-от використання малих фільтрів 3×3 , глобального average pooling перед фінальним класифікаційним шаром, а також перенесення ідеї збільшення кількості каналів із плином глибини. Однак ключова відмінність – впровадження коротких з'єднань – відкрила шлях до формування дуже глибоких мереж, що давало змогу дедалі краще моделювати складні закономірності у великих наборах даних [6]. У контексті ImageNet це означало ще більше зниження помилки класифікації порівняно з VGG, а також підвищення «продуктивності на параметр», коли глибші мережі не обов'язково вимагають надмірного збільшення обчислень, але натомість ефективніше «вчаться» представляти складні простори ознак.

ResNet фактично започаткувала ціле сімейство моделей: Wide ResNet, ResNeXt та інші. Усі вони базувалися на ідеї збереження з'єднання між шарами, щоб уникати проблем з градієнтом. Цей же принцип був згодом адаптований в інших галузях: у рекурентних мережах (Highway Networks), у трансформерах (додаючи skip connections між шарами енкодера і декодера) тощо. Тож можна сказати, що ResNet змінила наше розуміння меж глибокого навчання, продемонструвавши, як структурне вдосконалення архітектури дає змогу радикально покращити наочність та ефективність глибоких нейронних мереж.

2.5 EfficientNet (2019)

У 2019 році з'явилася серія моделей під назвою EfficientNet [10], яка сфокусувалася на збалансованому масштабуванні розмірів мережі та досягла вражаючих показників точності при обмеженій кількості параметрів. Творці EfficientNet запропонували систематичний підхід до пошуку оптимальної конфігурації глибини, ширини та роздільної здатності вхідних даних. Вони ввели поняття «compound scaling»: замість довільного збільшення лише глибини чи лише ширини мережі, пропонувалося пропорційно масштабувати всі виміри залежно від певного «коефіцієнта масштабування».

Базова модель EfficientNet B0 була знайдена автоматизованим пошуком архітектури (Neural Architecture Search), що дозволило оптимізувати початкову конфігурацію фільтрів і блоків. Далі, для отримання більших моделей (B1, B2, ..., B7) розміри мережі масштабували збалансовано, нарощуючи кількість каналів, глибину та роздільну здатність вхідного зображення. Порівняно з попередніми підходами, такими як ResNet чи VGG, EfficientNet продемонструвала суттєво вищу точність на ImageNet при меншій кількості параметрів і менших FLOPs (Floating Point Operations) [2]. Саме тому вона набула великої популярності

для реальних застосувань, де апаратні обмеження мають вирішальне значення.

Концепцію цієї мережі ілюструє рисунок 7. Кожен блок застосовує Mobile Inverted Bottleneck з використанням згорток із розширенням каналу та подальшим зворотним стисканням. Додатково часто застосовується методика squeeze-and-excitation, що допомагає мережі вибірково акцентувати увагу на релевантних каналах. Застосування таких інноваційних шарів дозволило підвищити якість розпізнавання складних образів, зберігаючи при цьому відносно низьке споживання обчислювальних ресурсів.

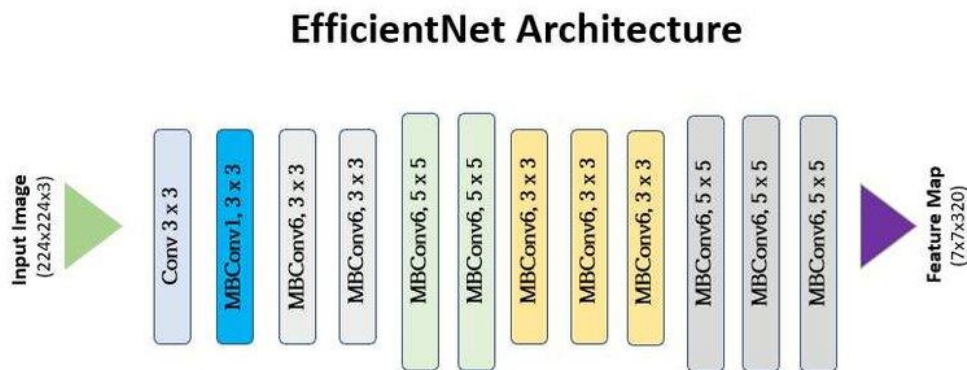


Рисунок 7 – Архітектура EfficientNet, яка балансує між продуктивністю та ефективністю параметрів

EfficientNet фактично продемонструвала, що навіть у випадку дуже глибоких моделей можливо уникнути «надмірної» кількості параметрів, якщо грамотно збалансувати ширину, глибину та роздільність. Це стало особливо корисним для застосувань, що потребують запуску мереж у режимі реального часу на пристроях з обмеженими ресурсами (наприклад, на мобільних телефонах, вбудованих системах чи роботехнічних платформах). Також виявилося, що підхід «compound scaling» може бути адаптований до інших типів архітектур, забезпечуючи систематичне

масштабування. Подальші варіації, як-от EfficientNet-L2 чи EfficientNetV2 [2], розширюють цей підхід, досягаючи ще кращих результатів на складних задачах ImageNet.

2.6 Висновки щодо CNN

У ході розвитку згорткових нейронних мереж можна простежити дві ключові тенденції. Перша стосується прогресивного збільшення глибини моделей: від LeNet, яка мала лише кілька згорткових шарів, до сучасних «гігантів» на кшталт ResNet-152 або EfficientNet-L2. Цей перехід наочно показав, що глибина дає змогу мережам успішніше вичленовувати складні патерни. Однак разом із тим зростає ризик затухання чи вибуху градієнта, що і стимулювало появу різних технік на кшталт резидуальних блоків. Друга тенденція пов'язана з оптимізацією використання обчислювальних ресурсів. AlexNet довела практичність навчання глибоких моделей на GPU, VGG встановила, що глибина може перевищувати 16–19 шарів, а ResNet показала, що можна будувати сотні шарів без деградації продуктивності за допомогою прямих зв'язків. EfficientNet зробила новий крок, запропонувавши збалансований підхід до масштабування з урахуванням апаратних обмежень.

Такі прориви неможливо відділити від розвитку апаратних ресурсів і методів оптимізації. Поява високопродуктивних графічних процесорів, спеціалізованих модулів TPU (Tensor Processing Units) та інших прискорювачів, а також програмних фреймворків (TensorFlow, PyTorch, MXNet тощо) зробила можливою реалізацію й тестування глибоких CNN у промислових масштабах. Особливо важливим стало масове застосування перенесеного навчання: якщо модель уже навчена на великому наборі даних, як-от ImageNet, її можна адаптувати під конкретні завдання (медичні, промислові, аграрні тощо) шляхом донавчання на менших датасетах.

Згорткові мережі зберігають провідні позиції в галузі візуального розпізнавання, хоча останнім часом їх активно доповнюють чи інколи витісняють трансформери для задач комп'ютерного зору (наприклад, Vision Transformer, Swin Transformer) [3; 4]. Але незалежно від цього, CNN залишаються фундаментальним інструментом із вражаючим спектром застосувань, що охоплює медичні зображення, супутникові та аерофотознімки, автономні транспортні системи, мультимедійний пошук тощо.

Цей розділ продемонстрував, як архітектура CNN послідовно еволюціонувала від LeNet, через AlexNet, VGG та ResNet, аж до EfficientNet, що зуміла поєднати високу точність із помірними вимогами до ресурсів. У подальшому огляді будуть розглянуті інші сімейства нейронних мереж, які фокусуються переважно на послідовних або текстових даних (рекурентні та трансформерні моделі), а також мобільні та гібридні архітектури, які знаходять своє місце в конкретних нішевих застосуваннях. Проте згорткові мережі, безумовно, залишаються наріжним каменем досліджень у сфері комп'ютерного зору та класифікації зображень.

3 РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ (RNN)

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) сформували одне з ключових технологічних рішень у випадках, де потрібно мати справу з послідовними даними: текстом, мовою, музикою, часовими рядами та будь-якими іншими формами інформації, в якій нинішній елемент залежить від попередніх. На відміну від згорткових мереж, що переважно працюють із просторовими патернами, RNN використовують механізм ітеративної обробки вхідних сигналів у часі, «запам'ятовуючи» контекст через внутрішній стан (hidden state) мережі. Цей внутрішній стан несе в собі інформацію про оброблену раніше частину послідовності, що дозволяє мережі враховувати часовий контекст під час формування виходу в кожен момент часу. Саме ця здатність «пам'ятати» вхідні фрагменти робить RNN незамінними в задачах мовного моделювання, автоматичного перекладу, розпізнавання мовлення, аналізу часових рядів у фінансах тощо [12; 13].

Проте історично прості (або «ванільні») RNN зіткнулися зі значними труднощами в процесі навчання, особливо коли йшлося про обробку довгих послідовностей. Існували дві основні проблеми: затухання градієнтів і їхній вибух (vanishing and exploding gradients). Перша робить майже неможливим ефективне навчання на далеких залежностях – коли слово чи сигнал на початку послідовності має критичне значення для ухвалення рішення в кінці. Друга може призводити до того, що значення градієнтів стають нестабільно великими, «ламаючи» процес навчання. Щоб подолати ці складнощі, науковці розробили вдосконалені механізми рекурентності, найвідомішими серед яких є архітектури LSTM (Long Short-Term Memory) [13] та GRU (Gated Recurrent Unit) [12]. Ці моделі доповнили базову RNN-будову так званими «вентильними» механізмами

(gates), що дозволяють мережі керувати потоком інформації всередині та між часовими кроками.

Розглянемо докладніше кожен із варіантів RNN, починаючи від найпростішої Vanilla RNN, далі переходячи до LSTM і GRU, які вже стали промисловим стандартом у багатьох мовних та часових застосуваннях.

3.1 Vanilla RNN

Vanilla RNN – це базовий тип рекурентної мережі, в якому на кожному кроці часу t з урахуванням вхідного вектора x_t та прихованого стану h_{t-1} обчислюють новий прихований стан h_t і вихід y_t .

Типова формула має вигляд:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h), \quad (1)$$

$$y_t = W_{hy}h_t + b_y \quad (2)$$

де W_{hh} , W_{xh} , W_{hy} – вагові матриці, а b_h , b_y – зміщення (bias).

Функція «tanh» або інша активаційна функція (sigmoid, ReLU) визначає, як саме нова інформація зміщується з попереднім станом мережі.

Ітеративний характер обчислень, у якому вихід у момент часу t залежить від стану в момент $t-1$, дозволяє мережі запам'ятовувати певний контекст. Утім, якщо залежність між поточним станом і інформацією, отриманою багато кроків тому, надто далека, під час зворотного поширення похибки (backpropagation through time) градієнти можуть ставати занадто малими. Тоді «потрібна» раніше інформація фактично втрачається, і модель не в змозі скористатися давнім контекстом.

Іншою важливою проблемою є нестабільне зростання градієнта, коли похибка починає експоненційно наростати при розгортанні мережі на великий часовий проміжок. Деякі задачі, наприклад обробка довгих речень, розпізнавання мовлення у великих аудіосегментах чи аналіз

фінансових часових рядів із тривалими періодами, не могли бути адекватно розв'язані базовою RNN через її неможливість утримувати довготривалу пам'ять.

Попри ці обмеження, Vanilla RNN зіграла фундаментальну роль у розумінні рекурентних структур. У період, коли ще не існувало більш просунутих механізмів, саме проста RNN дозволила зробити висновки про те, що рекурентність дійсно здатна впливати на обробку послідовної інформації краще, ніж класичні багатошарові перцептрони. Інженери та дослідники також розробили низку методів, аби частково компенсувати проблеми затухання і вибуху градієнтів: застосування інструменту gradient clipping, спеціальних схем ініціалізації або регуляризації. Утім, це не позбавило Vanilla RNN від «врожденного» недоліку, через який глибинне використання базової рекурентності в задачах з довгим контекстом ставало фактично марним.

При цьому базові RNN досі наявні в деяких задачах, де довжина оброблюваної послідовності не надто велика. Якщо модель має врахувати взаємозв'язок між сусідніми елементами (скажімо, кількома останніми кроками у часовому ряді), Vanilla RNN цілком може забезпечити прийнятну продуктивність без надмірного ускладнення обчислень. Вона залишається теоретичною «точкою відліку», на якій базується розуміння більш складних архітектур.

3.2 LSTM (1997)

Long Short-Term Memory (LSTM) – одна з найвпливовіших архітектур в історії глибинного навчання, розроблена Зеппом Гохрайтером і Юргеном Шмідхубером 1997 року [13]. Головна ідея LSTM полягає у введенні спеціальних «вентильних» механізмів (input gate, forget gate та output gate), а також окремої «комірки пам'яті» (cell state), яка слугує каналом для довгострокового збереження інформації. Завдяки такій

структурі мережа може вирішувати, яку інформацію залишити, а яку «забути», що суттєво знижує проблему затухання градієнтів.

Архітектура LSTM-блоку зазвичай описується формулами:

1. Forget gate:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f), \quad (3)$$

де σ – це сигмоїдна функція, яка виводить значення від 0 до 1, визначаючи, яку частку інформації з попереднього стану C_{t-1} потрібно забути.

2. Input gate:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \quad (4)$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c), \quad (5)$$

де i_t визначає, наскільки сильно нова кандидата пам'ять $\tilde{C}_t$ увійде до головного стану, а функція «tanh» задає масштаби цієї нової інформації.

3. Update cell state:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \quad (6)$$

тобто основний «потік пам'яті» оновлюється на основі того, що вирішив forget gate (видалити непотрібне) і input gate (додати нову інформацію).

4. Output gate:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \quad (7)$$

$$h_t = o_t * \tanh(C_t), \quad (8)$$

де o_t вирішує, яку частину новоствореного стану C_t необхідно пропустити на вихід мережі h_t .

Ключова перевага LSTM полягає в тому, що градієнт під час зворотного поширення має прямий шлях крізь лінійне оновлення стану C_tC_t . Це «розв'язує руки» мережі в розв'язанні задач, де необхідно зберігати інформацію протягом багатьох кроків часу, наприклад у машинному перекладі великих текстів або синтезі мовлення.

Із появою якісних реалізацій LSTM у фреймворках на кшталт TensorFlow, PyTorch чи Theano розпочався новий етап у розвитку мовних моделей та аналізі часових рядів. Про успішність LSTM свідчать численні роботи з машинного перекладу, зокрема Neural Machine Translation (NMT), у яких це була базова рекурентна архітектура, що дозволяла долати великі відстані залежностей у реченнях. Пізніше принцип LSTM став фундаментом навіть у деяких архітектурах для обробки зображень, де певним чином упорядковані згорткові блоки повторювалися для моделювання серійного процесу.

LSTM, тим не менш, не є «чарівним рішенням» від усіх проблем. Обчислювально вона складніша за Vanilla RNN, містить більше параметрів, а також повільніше навчається. Проте, з погляду архітектурної зручності, вона майже десять років була «золотим стандартом» рекурентних мереж у задачах, де потрібно враховувати довготривалі залежності, допоки на сцену не вийшли трансформери [3], які в деяких застосуваннях почали LSTM витісняти.

Водночас у низці практичних завдань, особливо на вбудованих пристроях чи системах з обмеженою пам'яттю, LSTM і надалі лишається привабливим варіантом.

3.3 GRU (2014)

Gated Recurrent Unit (GRU) була запропонована 2014 року групою дослідників під керівництвом Кьонга Чо та Йошуа Бенджіо [12]. Модель GRU можна розглядати як спрощений різновид LSTM із меншим числом

воріт та відсутністю окремого стану Ct_tCt . Замість цього GRU об'єднує механізм забування й механізм оновлення вхідних даних у дві основні «брами»: update gate та reset gate.

Типові рівняння для GRU мають вигляд:

1. Update gate:

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z), \quad (9)$$

що визначає, наскільки прихований стан мережі h_t буде оновлений новою інформацією, а яку частку попередньої інформації варто зберегти.

2. Reset gate:

$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r), \quad (10)$$

що вирішує, чи варто «забути» попередній стан перед розрахунком кандидата нового стану. Якщо r_t близький до нуля, інформація з h_{t-1} практично не враховується.

3. Нова кандидата стану:

$$\tilde{h}_t = \tanh(W_h[r_t * h_{t-1}, x_t] + b_h). \quad (11)$$

4. Оновлення поточного стану:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t. \quad (12)$$

Завдяки такій конструкції GRU зберігає більшість переваг LSTM у сенсі можливості моделювати довготривалі залежності та уникати катастрофічного затухання градієнтів, але має менше параметрів і простішу внутрішню структуру, що позитивно впливає на швидкість

навчання. На практиці GRU часто демонструє зіставну або навіть кращу продуктивність, ніж LSTM, особливо на середніх за розміром даних, поступаючись LSTM лише в окремих сценаріях, де дуже довготривалі залежності критичні.

Іще однією перевагою GRU є менша складність реалізації: порівняно з трьома брами LSTM плюс окремий стан пам'яті, у GRU реалізовано фактично дві брами та спільний прихований стан. Завдяки цьому архітектура GRU трохи «економніша» у використанні пам'яті та часу на крок навчання, що важливо в епоху зростання уваги до мобільних та вбудованих застосувань.

У низці наукових робіт та конкурсів на кшталт машинного перекладу, класифікації текстів і предикції часових рядів було показано, що GRU мережі можуть досягати або й перевершувати результати LSTM, водночас потребуючи менше обчислень [12]. Це особливо привабливо, коли часовий ряд чи текстові дані не такі довгі, а також коли критичним є час тренування.

3.4 Порівняння й інтеграція RNN-архітектур

І Vanilla RNN, і LSTM, і GRU загалом базуються на ідеї рекурентності, тобто ітеративного проходження крізь послідовність і накопичення інформації у прихованому стані. Найпоширенішою нині є LSTM, оскільки вона стала символом «глибокої пам'яті» для часових і текстових завдань. Проте GRU дедалі частіше обирають у застосунках, де необхідно збалансувати глибину контексту та економію ресурсів. Vanilla RNN актуальна переважно в тих сценаріях, де послідовність нетривала, і немає великої загрози втрати далекої залежності.

У деяких роботах було продемонстровано, що поєднання LSTM та GRU в багаторівневих мережах може дати додаткові переваги, зокрема у задачах мультимодальної обробки даних, де є потреба і у сильній

довготривалій пам'яті, і у більш швидкому локальному оновленні. Схематично таку інтеграцію можна ілюструвати за допомогою гібридної архітектури LSTM і GRU (рис. 8), де перші шари мережі можуть бути реалізовані на основі LSTM, а завершальні – на основі GRU, або навпаки. Цей підхід не є масовим стандартом, але використовується в окремих нішевих проєктах, які вимагають експериментів з різними комбінаціями.

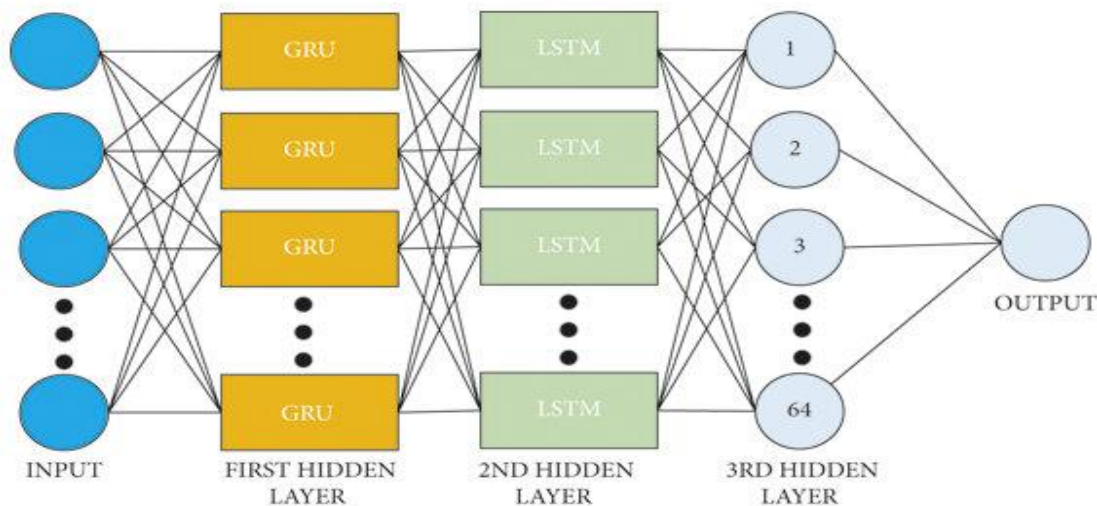


Рисунок 8 – Гібридна архітектура LSTM і GRU: інтеграція шарів для обробки послідовних даних

Істотним кроком у розвитку рекурентних архітектур стала ідея двонаправлених RNN (BiRNN). У цій конфігурації послідовність обробляється одночасно в «прямому» та «зворотному» напрямках, що дає змогу на кожному кроці враховувати не лише минулий, а й майбутній контекст (хоча у реальних застосуваннях не завжди є можливим заглядати вперед). BiLSTM і BiGRU використовуються тоді, коли вся послідовність доступна цілком, наприклад у завданнях синтаксичного аналізу тексту або розпізнавання облич, коли відео можна переглядати як уперед, так і назад.

Ще один підхід – додавання механізму уваги (attention) до RNN, що дало поштовх до створення перших версій seq2seq-моделей із увагою. Передусім увагу впровадили в NMT-системах, аби вирішити проблему,

коли вирок занадто довгий і мережі складно утримувати у внутрішньому стані детальну інформацію про віддалені слова. Механізм уваги дозволяє на кожному кроці генерації (декодування) безпосередньо звертатися до потрібних фрагментів вхідної послідовності. Саме з цього концепту виросли пізніші потужні трансформери [3].

3.5 Практичне застосування та місце RNN в сучасній екосистемі

Упродовж 2010-х років RNN стали базовим інструментом у величезному діапазоні застосувань обробки природної мови: від мовного моделювання та чат-ботів до розпізнавання мовлення (ASR) й машинного перекладу. Згодом, з появою архітектури Transformer [3], ситуація змінилася, і для багатьох мовних завдань трансформери виявилися ефективнішими та масштабованішими, оскільки вони позбавлені ітеративного процесу обробки послідовностей і можуть паралелізувати обчислення для всіх позицій відразу. Водночас RNN не зникли, а продовжують використовуватися у сценаріях, де довжина послідовностей відносно невелика або ж де паралелізація не настільки критична, а також там, де обмежені апаратні ресурси (де складно розгорнути велику трансформерну модель).

Однією з популярних сфер є аналіз часових рядів: фінансові прогнози, моделювання кліматичних даних, системи управління в реальному часі. У таких завданнях LSTM чи GRU добре справляються з притаманною цим даним послідовністю, а включення додаткових компонентів на кшталт attention або резидуальних зв'язків (аналогічно ResNet у CNN) дозволяє подовжити корисну пам'ять чи підвищити стійкість до шуму.

Ще одна галузь – робота із сенсорними даними, де потік сигналів від пристроїв IoT, медичних моніторів чи промислових сенсорів має часову природу, і RNN можуть «згортати» ці сигнали у векторні простори ознак,

застосовуючи фільтри в часовому вимірі. Хоча згорткові мережі (Temporal Convolutional Networks) теж успішно розв'язують такі завдання, LSTM і GRU лишаються релевантними через зручність експлуатації, особливо якщо потрібно враховувати дуже довготривалі залежності.

Окрім того, RNN-архітектури використовують у музичному моделюванні, зокрема для генерації послідовностей нот, відстеження гармонічних змін у часі та імпровізаційних структур. LSTM, GRU або їх комбінації дають можливість зберігати відомості про попередні музичні фрази й елементи мелодії, що створює більш плавний і логічний розвиток музичного контенту.

У сфері генеративних моделей (на кшталт auto-regressive генерації зображень або послідовностей на основі попередніх кроків) RNN теж відігравали важливу роль до появи потужніших моделей на основі трансформерів. Навіть зараз, коли трансформери домінують у текстових і мультимодальних задачах, RNN іноді застосовують як частину підсистеми для специфічних коротких послідовних обчислень чи побудови швидких енкодерів.

3.6 Висновки щодо RNN

Рекурентні нейронні мережі, попри наявність певних недоліків, дали можливість докорінно просунутися у вирішенні послідовних завдань, особливо в доменах природної мови та часових рядів. Vanilla RNN стали першим кроком, який продемонстрував ідею акумуляції контексту крізь часовий перебіг, але залишився вразливим до проблеми затухання і вибуху градієнтів. LSTM запропонували розв'язок у формі комірки пам'яті та трьох воріт, що дозволили «розтягувати» корисну інформацію на довгі інтервали часу, не втрачаючи її через мінливі градієнти. GRU, зі свого боку, оптимізували цю концепцію, спростивши внутрішню структуру та підвищивши ефективність навчання.

Хоча сучасна хвиля досліджень змістила фокус на трансформери через їхнє вміння паралелізувати обчислення і працювати з далекими залежностями без рекурентного «ланцюжка», RNN залишаються вагомим елементом технологічного стеку глибокого навчання. Вони часто є хорошим вибором у завданнях, де дані мають природну послідовність, проте не надто великі за довжиною, або ж коли система обмежена в обчислювальних ресурсах і розгортання складних трансформерів стає недоцільним. Із появою гібридних підходів, які поєднують рекурентну динаміку й механізми уваги, RNN також отримали нове життя, набуваючи здатності фокусуватися на релевантних частинах послідовності так само, як це роблять трансформери.

Таким чином, RNN – це важливий компонент «історичної лінії» розвитку глибокого навчання. Без розуміння базової рекурентної структури, а також ключових ідей, закладених в LSTM і GRU, неможливо повноцінно досягнути сучасні підходи до обробки послідовної інформації. У наступному розділі розглянемо, як еволюція RNN та механізму уваги призвела до появи трансформерів, що зайняли чільне місце у задачах машинного перекладу, сумаризації, генерації текстів і навіть комп'ютерного зору.

Проте, якою б досконалою не була нова архітектура, RNN, особливо LSTM і GRU, не втратили своєї актуальності для значної кількості практичних завдань, у яких критеріями вибору лишаються простота, ефективність та надійність навчання на порівняно коротких чи структурованих послідовностях.

4 ТРАНСФОРМЕРИ (TRANSFORMERS)

У контексті глибинного навчання трансформери стали одним із найбільш впливових проривів останніх років, особливо в галузі обробки природної мови та, згодом, у комп'ютерному зорі й суміжних доменах. Ця архітектура суттєво відрізняється від класичних рекурентних мереж тим, що вона не покладається на ітеративне обчислення станів у часовій послідовності, а натомість ґрунтується на механізмі «уваги» (attention). Завдяки такому підходу трансформери змогли ефективно враховувати контекст будь-якого елемента послідовності відносно всіх інших, що дало їм змогу гнучко масштабуватися до великих обсягів даних і обчислювальних ресурсів. Згорткові та рекурентні моделі раніше намагалися «розподіляти» інформацію вглиб, використовуючи багат шарову структуру, тоді як трансформери додатково організовують цю інформацію вздовж усього набору позицій відразу, уникаючи крок-за-кроком обробки. Така паралельна схема здатна значно швидше опрацьовувати великі послідовності, що особливо важливо для мовних моделей і машинного перекладу [3].

Витоки цієї архітектури пов'язані з появою Original Transformer 2017 року, який заклав концептуальний фундамент для BERT, GPT і наступних поколінь. У міру зростання обчислювальних можливостей наукова спільнота почала досліджувати, наскільки далеко може зайти ідея self-attention, розширюючи її межі від аналізу тексту до обробки зображень, відео та навіть складних мультимодальних завдань. Сьогодні трансформери є одним із найпопулярніших напрямів у глибинному навчанні та широко застосовуються у великих мовних моделях, інформаційних пошукових системах і навіть у складних наукових задачах, включно з біоінформатикою і фізикою високих енергій.

4.1 Original Transformer (2017)

Рік 2017 став поворотним у галузі обробки природної мови, коли команда дослідників із Google Brain та Університету Торонто опублікувала роботу «Attention Is All You Need», яка презентувала так званий Original Transformer. Традиційні підходи до машинного перекладу на той час зазнали кількох вдосконалень: від рекурентних seq2seq-моделей із LSTM чи GRU до механізмів уваги, які додавалися поверх рекурентних блоків. Проте автори Original Transformer радикально спростили цю ідею, запропонувавши повністю відмовитися від рекурентності та згорткових шарів, натомість покластися лише на багатоголовий механізм уваги (multi-head attention) у поєднанні з позиційними векторними уявленнями (positional encoding).

Ключове нововведення полягало в тому, що на кожному кроці декодування або енкодингу трансформер може «подивитися» на всі інші елементи вхідної або вихідної послідовності через матриці Q (queries), K (keys) та V (values). Це дає змогу визначати, які частини послідовності найрелевантніші для поточного стану обробки. На відміну від рекурентної логіки, де інформація послідовно протікає від одного кроку до наступного і створює залежність від попереднього стану, трансформер паралельно зважає внесок кожного позиційного елемента. Завдяки багатоголовому підходу мережа може одночасно аналізувати кілька різних типів залежностей або відносин між елементами.

Розділ на енкодер і декодер зберігся від попередніх seq2seq-рішень, проте кожен блок енкодера будувався з багатоголового self-attention і послідовності шарів нормалізації та повнозв'язних проєкцій. Декодер мав аналогічну структуру, але додатково включав «маскований» механізм уваги, що не дозволяв «дивитися наперед» на невідомі майбутні слова при генерації послідовності. Загалом такий підхід забезпечив високу гнучкість, оскільки трансформер міг паралельно опрацьовувати довгі речення, не

втрачаючи контексту на далеких позиціях, що було одним із головних обмежень RNN, навіть з LSTM чи GRU.

Окрім теоретичних переваг, Original Transformer зарекомендував себе як винятково продуктивне рішення у машинному перекладі й подібних задачах. На відомому наборі WMT (українська, німецька, французька, англійська та інші мови) він суттєво випередив попередні рекурентні seq2seq-моделі. Це спричинило лавиноподібне збільшення досліджень у напрямку attention-підходів, зокрема почали з'являтися масштабніші моделі з десятками шарів енкодера та декодера, які можливо було навчати на потужних графічних процесорах і TPU.

Іще одним важливим елементом Original Transformer стали позиційні кодування (positional encodings). Оскільки модель не має вбудованої рекурентності, їй потрібно було навчитися відрізняти позиції різних слів у реченні, і розуміти, що, наприклад, перше слово – це підмет, а четверте – це присудок. Позиційні вектори, які додаються до вхідних ембедингів, уможлиблюють це, надаючи мережі гармонічні ознаки, що залежать від порядкового номера позиції у вхідній чи вихідній послідовності.

Original Transformer заклав ідеологічні основи для численних наступників у сфері обробки природної мови. З'явилася велика кількість модифікацій і розширень, як-от Transformer-XL, які намагалися ще краще моделювати дуже довгі контексти, а також інші вдосконалення на кшталт перехресних механізмів уваги (cross-attention) між різними типами даних. Упровадження трансформерів дало початок новому стандарту в NLP, де глибокі рекурентні мережі почали дедалі частіше відходити на другий план.

4.2 BERT (2018)

Через рік після презентації Original Transformer компанія «Google» знову здивувала дослідників, опублікувавши модель BERT (Bidirectional

Encoder Representations from Transformers), яка на практиці заклала фундамент «епохи попередньо навченої мовної моделі» у широкому розумінні. Якщо Original Transformer становив енкодер-декодер для задачі перекладу, то BERT сконцентрувався на тому, як створювати глибоке двонаправлене уявлення речення, що можна буде використовувати у різноманітних завданнях NLP – від класифікації тексту й аналізу тональності до виявлення іменованих сутностей і пошуку відповідей на запитання.

Ідея BERT полягала в навчанні енкодера трансформера в масштабному самоконтрольованому режимі (self-supervised learning). Для цього пропонувалася задача Masked Language Modeling (MLM), де частина слів у реченні приховується (маскується), а мережі потрібно відновити їх, ґрунтуючись на решті контексту. Це навчання дозволяло моделі опановувати внутрішні відносини між словами на дуже великих корпусах текстів без явного ручного маркування даних. Паралельно застосовували мету Next Sentence Prediction (NSP), яка вимагала від моделі вміти визначати, чи є друге речення логічним продовженням першого.

BERT опинився в центрі уваги спільноти NLP після того, як він показав неперевершені результати на популярних стандартизованих бенчмарках (GLUE, SQuAD тощо). До того часу існували й інші підходи до передтренування мовних моделей, однак вони часто були односпрямованими (наприклад, GPT із авторегресивним навчанням). BERT же використовував двонаправлену увагу: кожен токен у реченні міг під час навчання дивитися водночас ліворуч і праворуч (зрозуміло, з відповідними модифікаціями маскування), що дозволяло отримувати більш цілісне уявлення про структуру та сенс фрази.

Після випуску BERT низка компаній і академічних груп почали вивчати можливості його розширення: зростала кількість шарів, кількість параметрів та обсяги тренувального корпусу. З'явилися моделі на кшталт

RoBERTa, DistilBERT, ALBERT, які пропонували покращення в швидкості або в точності. Крім того, BERT став основою для численних спеціалізованих моделей, як-от BioBERT (для біомедичних текстів), ClinicalBERT (для клінічних записів) тощо. Цей підхід до попереднього навчання виявився настільки успішним, що великі корпорації почали застосовувати BERT-подібні рішення в пошукових системах, чат-ботах, системах аналізу клієнтських звернень.

Однією з ключових особливостей BERT залишається його здатність бути «тонко налаштованим» (fine-tuned) під конкретне завдання з мінімальними змінами архітектури. Достатньо додати один-два додаткові шари або навіть просто лінійну «голову» поверх вихідних векторів енкодера й адаптувати ваги. Цей процес тонкого налаштування потребує значно менше даних і часу, ніж повноцінне навчання з нуля. У підсумку BERT продемонстрував, як потужні мовні моделі можуть стати універсальним «фреймворком» для всього ланцюга NLP-завдань, і надовго закріпив трансформерну архітектуру як домінантну в цій сфері.

4.3 GPT (2018–2023)

Архітектура GPT (Generative Pre-trained Transformer), розроблена компанією «OpenAI», стала тим самим «дзеркальним відображенням» ідей BERT, але в авторегресивному варіанті. Якщо BERT – це модель-енкодер, що навчається, маскуючи частину слів у середині речення, GPT обрала підхід генерування наступного слова, ґрунтуючись на попередніх. Така задача авторегресивного передбачення (language modeling) є більш традиційною, проте GPT поєднала її з масштабним попереднім навчанням на величезних корпусах текстів.

Перша версія GPT (2018) була порівняно невеликою за теперішніми мірками. Наступний реліз GPT-2 (2019) уже здійняв значний резонанс у спільноті: модель містила 1,5 мільярда параметрів і показала дивовижну

здатність до когерентного генерування тексту, особливо коли під час навчання використовували великий набір інтернет-текстів. OpenAI навіть тимчасово утримувалася від публікації повної версії GPT-2, посиляючись на міркування «безпеки» та можливі зловживання фейковим контентом.

У 2020–2021 роках світ побачили моделі GPT-3 із кількістю параметрів 175 мільярдів. Таке масштабування стало можливим завдяки стрімкому розвитку обчислювальної інфраструктури й оптимізаційних методів. GPT-3 продемонструвала ще більш переконливі результати в генерації тексту, його узагальненні, написанні програмного коду тощо. До того ж вона виявила здатність виконувати низку завдань «із одного прикладу» (few-shot learning), коли користувач дає лише кілька прикладів формату вхідних даних і бажаного виходу, а модель уже «інтуїтивно» розуміє, що потрібно робити.

Пізніші оновлення, зокрема GPT-3.5 і GPT-4, продовжили цю тенденцію, збагачуючи модель контекстними вікнами більшої довжини та покращуючи точність у різних тестах. З'явилися інтерактивні системи на основі GPT (наприклад, ChatGPT), здатні ефективно відповідати на складні питання, підтримувати бесіду, генерувати коди, розв'язувати рівняння і навіть створювати творчий контент на кшталт віршів чи оповідань. Попри те що ці моделі все ще можуть робити «галюцинації» або допускати логічні помилки, вони доводять, що авторегресивний трансформер зі значним масштабуванням може бути універсальним інструментом обробки та генерації тексту.

У випадку GPT з'явилася ідея in-context learning, де модель не лише запам'ятовує статистику, але й динамічно «налаштовується» під користувацькі сценарії, використовуючи підказки (prompts). Натомість BERT-підхід більше орієнтований на fine-tuning, коли модель донавчають на конкретному наборі маркованих даних. Обидва підходи зараз розвиваються паралельно, проте саме GPT-подібні моделі нині

привертають найбільшу увагу через їхню вражаючу здатність до універсального генеративного інтелекту.

4.4 Vision Transformers (ViT, 2020)

Успіх трансформерів у мовних завданнях стимулював науковців перевірити, чи можна подібні механізми уваги застосувати й у комп'ютерному зорі, де тривалий час домінували згорткові мережі (CNN). У 2020 році з'явилася модель Vision Transformer (ViT) [3], автори якої перевели зображення в послідовність «патчів» (patches), тобто невеликих блоків пікселів, і подали їх як вхід до класичного енкодера трансформера. Ідея полягала в тому, щоб розглядати кожен патч зображення як «токен», аналогічно до слова в NLP, і навчати мережу, щоб вона «уважно» звертала увагу на різні частини зображення, не вдаючись до згорткових операцій.

Ключовим було те, що ViT навчали на дуже великому датасеті (початково — JFT-300M від Google), що нараховував сотні мільйонів зображень. Така стратегія дозволила ViT досягнути широкий спектр візуальних патернів і досягти конкурентоспроможних результатів на ImageNet та інших стандартних наборах. Хоча базова реалізація ViT у початковому вигляді може трохи поступатися найоптимізованішим CNN за однакового розміру датасету, великі моделі ViT здатні показувати неабиякі переваги, коли йдеться про збільшення обсягу даних.

Архітектура Vision Transformer близька до мовних трансформерів, де кожен токен (патч) отримує позиційне кодування, а потім проходить через кілька послідовних блоків self-attention, мультигедної уваги та MLP-шарів. Позиційне кодування візуальних патчів забезпечує «просторове» розміщення фрагментів зображення, аналогічно до того, як позиційне кодування слів надає інформацію про індекс кожного токена в реченні. Головна відмінність від CNN полягає в тому, що ViT «не знає» про локальні згортки, а вчиться вилучати ознаки шляхом глобальної уваги.

Такий підхід робить модель здатною «зіставляти» довільні частини зображення без необхідності покрокового згорткового пошуку сусідніх пікселів.

У низці досліджень з'ясувалося, що ViT потребує менше регулярних прийомів під час навчання, ніж CNN, але водночас критично залежить від великого обсягу даних. Подальше розширення ідеї ViT призвело до появи гібридних моделей, які поєднують згорткові шари на початку (для більш ефективного вилучення локальних ознак) з архітектурою трансформера в подальших шарах (Swin Transformer, PVT, DeiT та інші [3; 4; 11]). Деякі з цих моделей застосовують механізм зсувних вікон (shifted windows), аби прискорити обчислення та скоротити пам'ять.

Vision Transformers продемонстрували, що «увага всюди» працює не лише для тексту, а й у випадку зображень, досягнувши рівня (або й перевершивши його) найкращих CNN у задачах класифікації. Такі результати заохотили дослідників до застосування трансформерів у багатьох суміжних напрямках зорових задач: семантичній сегментації, детекції об'єктів, генеруванні зображень і навіть відеоаналітиці. Хоча згорткові архітектури, на кшталт EfficientNet і ResNet, продовжують бути дуже потужним інструментом, ViT та інші трансформерні моделі все частіше інтегруються у виробничі конвеєри завдяки вмінню ефективно масштабуватися і пристосовуватися до завдань за умови наявності великого датасету або можливості попереднього навчання на «мегакорпусах».

4.5 Вплив трансформерів і тенденції розвитку

Поява Original Transformer, BERT, GPT і ViT засвідчила, наскільки гнучкою може бути архітектура, заснована на увазі. Трансформери надали можливість паралельної обробки великих послідовностей, що раніше не було доступним у рекурентних та згорткових моделях, і це відкрило шлях

до «масштабування» – збільшення кількості шарів, голов, параметрів і тренувальних прикладів. Такий масштабований підхід особливо добре працює в умовах великих обсягів даних.

З'явилися численні модифікації, які намагаються розширити концепцію трансформера на специфічні домени. У текстових завданнях розробляють варіанти з адаптованою пам'яттю або методами, що дозволяють обробляти надзвичайно довгі послідовності (наприклад, Longformer, BigBird тощо). У комп'ютерному зорі все більшої популярності набувають «гібридні» підходи, де перші шари можуть залишатися згортковими, а далі залучається механізм уваги для узагальненої обробки високорівневих ознак.

Паралельно з цим у науковій спільноті тривають дискусії про те, наскільки метод self-attention може замінити всі інші підходи. Дехто вважає, що трансформери справді стануть універсальною архітектурою, в якій згодом з'являться вбудовані згортки, рекурентність чи інші механізми як приватні випадки. Існують і критичні думки, де відзначають надмірні обчислювальні витрати трансформерів при обробці великих обсягів даних, оскільки обчислення матриць уваги масштабуються з $O(n^2)$ щодо довжини послідовності. Це вже стимулює дослідників розробляти efficient-transformer-архітектури з лінійною чи квазілінійною складністю відносно n , використовуючи такі ідеї, як sparse attention, kernel-based attention, low-rank approximations.

Щодо практичного впливу, то завдяки BERT, GPT і ViT все більше промислових організацій і дослідницьких груп роблять ставки саме на трансформери як на базову модель, яку можна «донавчати» для розв'язання завдань у мовній аналітиці, аналізі зображень чи мультимодальній інтеграції. У сфері злиття медійних форматів (текст, зображення, аудіо, відео) трансформери виявилися дуже гнучкими, оскільки їхня увага може поєднувати паралельно декілька різних типів

ознак. Існують уже системи multimodal GPT, що інтегрують зображення та текст в одному фреймворку, додаючи вхідні та вихідні «блоки» трансформера, спеціально пристосовані під відповідний формат даних.

Щодо напрямів подальшого прогресу, то виділяються три основні вектори. Перший: подальше збільшення масштабу (більше даних, більше параметрів, довші контексти, складніші задачі). Другий: розробка ефективніших алгоритмів і модифікацій «уваги», що зменшують часову та просторову складність (Swin Transformer, Linformer, Performer тощо). Третій: пошук «загальних» моделей, які можуть одночасно обробляти кілька типів даних або вирішувати низку різнорідних задач (universal models). При цьому скептики звертають увагу на те, що надмірне масштабування призводить до великих витрат електроенергії та грошей, а також може породити «галюцинації» чи інші непередбачувані ефекти в генерованому контенті.

У будь-якому разі трансформери відіграють чи не найголовнішу роль у формуванні сучасної картини глибинного навчання. Якщо ще декілька років тому їхня перевага здавалася доволі вузькоспеціалізованою (мовні завдання, машинний переклад), то нині вони проникли в різноманітні сфери від зорового сприйняття до рекомендативних систем і наукових симуляцій. Сам факт їхнього повсюдного успіху стимулює розробку нових апаратних прискорювачів (TPU, GPU останніх поколінь), які можуть виконувати матричні множення для великих розмірностей із підвищеною ефективністю.

4.6 Висновки щодо трансформерів

Архітектура трансформера виникла як несподівано проста й водночас надзвичайно потужна ідея: взаємодія між елементами послідовності може бути реалізована через матриці уваги без явної рекурентності чи згортки. Це поклало початок новому етапу в обробці

тексту і подальшому розвитку моделей типу BERT і GPT, які «вибухнули» в публічному просторі завдяки своїм унікальним можливостям. Вивільнена від покрокових обчислень залежність між будь-якими двома позиціями надала системі колосальну масштабованість, що особливо помітно у великих мовних моделях GPT-3, GPT-4 та інших гігантах із сотнями мільярдів параметрів.

Згодом з'ясувалося, що attention-парадигму можна доволі успішно інтерпретувати і в комп'ютерному зорі, у чому переконав весь науковий світ проєкт ViT. Тепер низка передових методів класифікації, детекції та сегментації зображень все частіше використовують різноманітні варіації трансформерних блоків, навіть якщо частина архітектури лишається згортковою. Із розвитком даних та обчислювальної інфраструктури трансформери все глибше проникають у наукові та індустріальні процеси, а також формують базис для досліджень мультимодальності та універсального штучного інтелекту.

Хоча на перший погляд трансформер можна вважати радикальним відходом від CNN і RNN, насправді тут простежується спадковість ідей. Механізм уваги виник як доповнення до рекурентних мереж у машинному перекладі, а робота над глибокими мережами для обробки зображень навчила дослідників оптимізувати велику кількість параметрів і шарів. Тому трансформери є продовженням загальної «епопеї глибинного навчання», що прагне систематично масштабувати та вдосконалювати методи вилучення ознак і обробки складних структур даних.

Таким чином, у царині класифікації (і не тільки) трансформери здатні охопити широке коло задач, від аналізу тексту до розпізнавання візуальних патернів. Їхня гнучкість і масштабованість, поряд із потенціалом комбінування з іншими підходами (енземблі, гібридні моделі, додаткові механізми уваги) роблять трансформери одним із головних напрямів сучасного глибинного навчання. Найважливіше те, що вони

зберігають провідні позиції й надалі, систематично розширюючись у нові сфери й витісняючи класичні архітектури там, де паралелізація і глобальна увага виявляються критичними для успіху.

5 СПЕЦІАЛІЗОВАНІ ТА МОБІЛЬНІ АРХІТЕКТУРИ

Упродовж ери глибинного навчання дослідники й інженери усвідомили, що далеко не в усіх середовищах існує змога застосовувати великі нейронні мережі з мільйонами чи навіть мільярдами параметрів. У реальних системах часто постають обмеження апаратних ресурсів, зокрема пам'яті, обчислювальної потужності, енергоспоживання. Це особливо актуально в контексті мобільних пристроїв, вбудованих систем, засобів «розумного» будинку та інших IoT-рішень. У відповідь на ці виклики стали розвиватися спеціалізовані та «полегшені» архітектури, які намагаються зменшити обчислювальну складність, кількість параметрів і пам'яті, водночас зберігаючи конкурентний рівень точності.

З іншого боку, швидкі темпи досліджень у глибинному навчанні створили середовище, де наукова спільнота почала шукати автоматизовані методи проектування ефективних структур мереж, аби звести до мінімуму «ручне» налаштування. Це призвело до розвитку напряму Neural Architecture Search (NAS), де використовують евристики чи методи навчання, щоб шукати оптимальні чи близькі до оптимальних архітектури під конкретні задачі й апаратні вимоги [1]. На перетині цих напрямів – оптимізація обчислень і автоматизація пошуку мереж – з'явилися такі моделі, як MobileNet (2017) та NASNet (2018). Обидві спрямовані на розв'язання проблеми ефективного навчання й інферування в умовах обмежених ресурсів.

5.1 MobileNet (2017)

MobileNet стала однією з перших «вірусно» популярних моделей, спеціально орієнтованих на роботу на пристроях із невисокою обчислювальною потужністю (смартфони, планшети, вбудовані одноплатні комп'ютери). Ініціаторами цього напряму були дослідники з

Google, які прагнули створити згорткову нейронну мережу із суттєво зменшеною кількістю параметрів і FLOPs (Floating Point Operations), порівняно з відомими на той час AlexNet, VGG чи навіть ResNet. Серед ключових технік, запроваджених у MobileNet, можна виділити глибинну (depthwise) і роздільну (pointwise) згортки [7].

Традиційні згорткові шари в CNN використовують 3D-фільтри, що одночасно враховують усі канали вхідного зображення чи проміжного тензора. Якщо глибина становить D каналів, то кожен фільтр має форму $k * k * D$. Хоча це й дає змогу «змішувати» інформацію між каналами, проте супроводжується значним зростанням кількості параметрів, особливо при збільшенні кількості вихідних каналів. У MobileNet цю операцію розбили на дві послідовні: depthwise-convolution, де кожен канал обробляється окремим фільтром $k * k * 1$, і pointwise-convolution $1 * 1 * D$, яка виконує лінійну комбінацію каналів для формування нового набору ознак. Така «факторизація» згортки зменшує обчислювальні витрати приблизно в 8–9 разів на один блок, залежно від параметрів, зберігаючи при цьому здатність вилучати релевантні просторово-канальні патерни.

Окрім цієї основної ідеї, MobileNet упровадив коефіцієнти масштабування (width multiplier і resolution multiplier), щоб можна було легко «розтягувати» або «звужувати» модель під конкретну конфігурацію пристрою. Якщо важлива продуктивність на старшому смартфоні, збільшують базову архітектуру, якщо ж працюють зі ще слабкішим пристроєм – зменшують кількість каналів (width) або роздільність вхідних зображень (resolution). Таке гнучке налаштування дозволило розробникам створювати «сімейство» MobileNet-моделей, кожна з яких орієнтована на певний баланс між точністю та швидкістю.

Поява MobileNet стимулювала чималий відсоток реальних застосувань у мобільній та робототехнічній індустрії, де задачі з класифікації чи детекції об'єктів мали виконуватися в режимі реального

часу без підключення до серверів. На відміну від більш ресурсомістких моделей, MobileNet здатна розпізнавати об'єкти при обмеженій кількості оперативної пам'яті й низькому енергоспоживанні. Це мотивувало Google та інших гравців ринку випускати наступні версії: MobileNetV2 і MobileNetV3 [7], в яких були додані механізми інвертованих бутстреп-блоків (inverted residual), а також елементи Squeeze-and-Excitation. У результаті якість класифікації на ImageNet ставала дедалі вищою, наближаючись до рівня більш «важких» мереж, натомість швидкість і компактність залишалися на передньому плані.

Сьогодні ідеї MobileNet простежуються у багатьох «легких» архітектурах, зокрема GhostNet [9], ShuffleNet, MnasNet, де також застосовують роздільні згортки та експерименти з топологією блоків. Це засвідчує, що факторизація згорток і різноманітні стратегії зменшення каналів стали стандартним арсеналом у проєктуванні нейронних мереж, які повинні працювати за відсутності потужного апаратного бекенду. Застосування MobileNet охоплює сфери від реальних систем відеоспостереження до медичних пристроїв, що аналізують зображення ультразвуку чи ЕКГ на місці, без пересилання даних до хмари.

Попри очевидні переваги, MobileNet, як і будь-яка «полегшена» архітектура, є компромісним рішенням. Кінцева точність, зазвичай, нижча, ніж у великих моделей (ResNet-152, EfficientNet-L2), коли немає обмежень апаратного рівня. Проте критична вимога реального часу або відсутність потужних процесорів є визначальним чинником, чому компактні мережі лишаються затребуваними. Крім того, навіть у промислових сценаріях часто поєднують MobileNet зі стратегічним донавчанням і квантованими вагами, щоб ще сильніше оптимізувати використання пам'яті і швидкодію під конкретну задачу.

5.2 NASNet (2018)

У 2018 році на міжнародній сцені глибинного навчання помітного розголосу набула концепція NASNet (Neural Architecture Search Network), презентована дослідниками з Google Brain. Цей проєкт був спрямований на пошук оптимальних архітектур саме під задачу класифікації зображень на ImageNet, причому в напівавтоматичному режимі. В основі ідеї лежав метод Neural Architecture Search (NAS), що формулює задачу побудови ефективної топології нейронної мережі як процес оптимізації в просторі можливих архітектур. На відміну від рукописних «інтуїтивних» дизайнів на кшталт ResNet чи MobileNet, інженери NAS прагнули зменшити людський фактор і довірити комп'ютеру пошук найкращої структури [1].

Ключовим внеском NASNet стало застосування підходу «контролера» (controller), що генерує потенційні «блоки» (cells) мережі, які потім перевіряють на ефективність за критеріями точності й обчислювальної складності. На ранніх етапах проєктів із NAS такого пошуку було дуже складно досягти, адже перебір можливих архітектур міг бути астрономічно великим. Утім, алгоритми еволюції та підсилювального навчання, разом із масовим використанням GPU-кластерів, дали змогу реалізувати цей підхід практично. Відпрацьовані схеми дозволяли оптимізувати структуру згорткових блоків, кількість каналів, порядок шарів, типи з'єднань (skip-connections або низхідні з'єднання), а також інші аспекти, які раніше дизайнери налаштовували вручну.

NASNet могла формувати «норми-блоки» (normal cells) і «скорочувальні блоки» (reduction cells), які послідовно розміщувалися вздовж глибини мережі. Наприклад, у разі завдання класифікації зображень, мережею глибоко пропускалися вхідні дані, а на виході використовували повнозв'язний шар чи глобальний average pooling для визначення класу. Під час експериментів було показано, що знайдені у такий спосіб архітектури здатні перевершувати моделі, створені

«руками» (ResNet, DenseNet, Inception) за точністю на ImageNet, інколи навіть за меншої кількості параметрів.

Попри великі досягнення, NASNet мала доволі істотну «ціну» – обчислювальну. На найраніших етапах доводилося витратити чималі GPU-ресурси для проходження сотень чи тисяч еволюційних поколінь, під час кожного з яких оцінювалося кілька нових варіантів архітектури. Це обмежувало застосування NASNet переважно великими корпораціями, що могли собі дозволити величезні кластери. Згодом було винайдено ефективніші стратегії, як-от «проксимальний» NAS, коли перед фінальною перевіркою на ImageNet нові архітектури тестувалися на менших датасетах (CIFAR-10, CIFAR-100), що зменшувало сукупну вартість пошуку [1]. У результаті з'явилися практичні моделі, які отримували найкращий чи один із найкращих результатів на ImageNet, поступаючись за точністю хіба що найдорожчим «гігантам» типу AmoebaNet.

Цікаво, що NASNet також започаткувала моду на комбінування ідей пошуку архітектури з підходами до оптимізації ресурсів. Деякі роботи намагалися «навчити» контролер враховувати не лише максимальну точність, а й розмір моделі або швидкість інференсу. Зокрема, у MnasNet, FBNet та інших подібних проєктах простір пошуку мережі визначався тим, наскільки швидко вона може працювати на цільовій апаратній платформі (GPU, CPU чи мобільний чип). Така задача ставала складнішою, проте й результат виявлявся більш практично придатним.

У контексті класифікації NASNet демонструвала виняткову ефективність і обіцяла відкрити нові горизонти, але епоха великих моделей на основі трансформерів частково відсунула на другий план CNN-орієнтований NAS. Тим не менш, методи NAS нікуди не зникли; навпаки, вони перетворилися на універсальніші інструменти, зокрема для пошуку гібридних архітектур, де поєднуються елементи згорткових шарів і механізм уваги. Багато ідей NASNet лягли в основу подальших досліджень

з теми автоконструювання моделей для мобільних пристроїв та загалом для обмежених систем.

5.3 Взаємодоповнення в екосистемі глибинного навчання

MobileNet і NASNet репрезентують дві різні, але взаємопов'язані парадигми. MobileNet рухалася від «ручного» компромісу між точністю й ефективністю, пропонуючи факторизацію згорток і масштабування мережі залежно від ресурсів. NASNet, навпаки, ішла шляхом дедалі більшої автоматизації, спираючись на алгоритми пошуку архітектур. Обидві концепції надзвичайно актуальні в добу, коли обчислювальні потреби глибинного навчання зростають, а мобільні пристрої стають усе розповсюдженішими.

У системах, де архітектори й розробники обмежені жорсткими рамками (до прикладу, 100 мілісекунд на обробку запиту, 200 МБ оперативної пам'яті, малий запас енергії), логічно звертатися до «легких» моделей. Задовольнившись зменшенням точності на кілька відсоткових пунктів, можна отримати значний вигравш у швидкості та стабільності, особливо коли модель має виконуватися безперервно на пристрої. MobileNet забезпечує гарний «стартовий» варіант, тоді як NASNet-підхід може згенерувати ще тонше оптимізовану структуру під конкретну задачу.

Підприємства та дослідницькі групи часто поєднують кроки автопошуку з подальшим тонким налаштуванням на цільовій апаратній платформі. Якщо стоїть мета надати додаток для смартфонів, що розпізнає об'єкти в режимі доповненої реальності, спочатку можна реалізувати NAS або взяти вже відому мобільну модель на зразок MobileNetV3, а потім адаптувати (prune, quantize, fine-tune) ваги під конкретний набір даних чи обчислювальний бюджет. Деякі середовища, наприклад TensorFlow Lite або PyTorch Mobile, пропонують зручні інструменти для глибшої

оптимізації, аж до перетворення ваг у 8-бітний формат, що суттєво знижує розмір і підвищує швидкість інференсу.

Окрім того, моделі, натхнені MobileNet чи NASNet, часто відіграють роль початкової «точки» в інших архітектурах. Деякі дослідники беруть їх за базу і «накладають» на верхівку додаткові модулі, наприклад механізм уваги для покращення класифікації складних класів, або додають спеціальні блоки для багатозадачного навчання (коли, крім класифікації, потрібно виконувати сегментацію чи виявляти ключові точки об'єктів). Завдяки компактності й ефективності первинного каркаса подібні гібридні рішення лишаються придатними для розгортання в мобільних середовищах, не вимагаючи ресурсоємних GPU-серверів.

У науковому сенсі розвиток «спеціалізованих» архітектур, до яких належить MobileNet, триває через зростання уваги до таких тем, як «green AI» – ідея мінімізації вуглецевого сліду та енергоспоживання моделей. Особливо важливо це стає при врахуванні того, що глибокі мережі можуть мати сотні мільйонів параметрів і споживати чималі потужності на етапі тренування та під час розгортання. Тому підходи, що пропонують «титультний» компроміс між розміром мережі й ефективністю, користуються зростаючим попитом.

5.4 Висновки щодо спеціалізованих та мобільних архітектур

MobileNet і NASNet презентують дві важливі складові сучасної екосистеми глибокого навчання, де все частіше постає вимога оптимізації під конкретне середовище. Якщо раніше фокус досліджень ішов на максимізацію точності (інколи за будь-яку ціну), то сьогодні інженерна думка спрямована також на те, аби ці моделі могли функціонувати у вбудованих системах, без доступу до масивних GPU чи кластерів. У цьому сенсі MobileNet – взірць методу «розумної» ручної оптимізації та факторизації згорток, тоді як NASNet є символом автоматизованого

пошуку й експериментування, що дозволяє мінімізувати роль суб'єктивних рішень архітектора.

Обидві лінії досліджень надихнули величезну кількість наслідувань та розширень, коли з'явилися не лише подальші версії MobileNet (V2, V3), а й цілі сімейства побудованих із використанням NAS. У підсумку можемо говорити про новий рівень гнучкості, котрий дозволяє підбирати обчислювальні схеми під конкретну задачу. Компактні моделі стали основою для багатьох мобільних застосувань, від реального часу розпізнавання жестів до діагностики сільськогосподарських культур за фотографіями з дронів, що надходять у польових умовах.

У наступних розділах будуть розглянуті інтегровані підходи, які поєднують різні архітектури (CNN, RNN, трансформери) або об'єднують кілька моделей в ансамблі. Проте нерідко в реальному середовищі такі системи містять саме мобільний чи спеціалізований компонент, коли постає необхідність розгортання на пристроях із сильними обмеженнями. З огляду на це важко переоцінити внесок MobileNet у формування «легкого» сегмента глибокого навчання, а також ідей NASNet, які підштовхнули дослідників до системного пошуку найкращого компромісу між ресурсами й точністю.

Спеціалізовані та мобільні архітектури продовжують розвиватися, отримуючи нові модифікації. Коли з'являються більш досконалі методи пошуку архітектур чи алгоритми оптимізації, інженери можуть швидко продукувати свіжі варіанти компактних мереж, що конкурують із більшими «гігантами» за ефективністю, залишаючись достатньо продуктивними. Це важливий напрям не тільки для споживчих застосувань, а й для галузей, де малі розміри й енергоефективність є критичними: медицина, робототехніка, аерокосмічні системи, автономні дрони. Синергія між ручним дизайном і автоматичним пошуком, ймовірно, продовжуватиме визначати еволюцію цього класу моделей, забезпечуючи дедалі ширші можливості для інтеграції глибокого навчання в реальне середовище.

6 ІНТЕГРОВАНІ ПІДХОДИ

Однією з характерних рис сучасних досліджень у галузі глибинного навчання є прагнення об'єднувати різні типи архітектур або декілька окремих моделей у межах спільної системи. Такий інтегрований підхід, коли синтезуються переваги різних методів, стає надзвичайно привабливим у складних завданнях класифікації, де дані можуть мати і просторову, і часову структуру одночасно або ж коли одночасно потрібно врахувати кілька поглядів на ті самі об'єкти. У цьому контексті можна виокремити щонайменше дві концепції, які набули великого розповсюдження: гібридні моделі (Hybrid Models), що поєднують, скажімо, CNN і RNN, а також ансамблеві архітектури (Ensemble Architectures), які передбачають одночасне використання кількох різновидів нейронних мереж чи інших алгоритмів, аби підвищити надійність і точність.

6.1 Hybrid Models (поєднання CNN і RNN)

У класичному глибинному навчанні існують дві усталені парадигми обробки даних: згорткова нейронна мережа (CNN), яка чудово працює з просторово структурованими вхідними сигналами (зокрема, зображеннями), та рекурентна нейронна мережа (RNN), ефективна в послідовних завданнях (часові ряди, тексти, аудіо). Проте в багатьох реальних задачах зустрічається комбінація просторової та часової (або послідовної) структури. Прикладом можуть бути відеодані, де кожен кадр сам по собі містить просторову інформацію, а послідовність кадрів утворює часовий контекст. Подібна ситуація спостерігається й у розпізнаванні рукописних рядків тексту, у біомедичних сигналів (ЕЕГ, ЕКГ, МРТ), а також у завданнях анотованого аудіо- та музичного аналізу.

Гібридний підхід передбачає, що CNN використовується для первинного вилучення ознак із кожного кадру або блока даних, а

рекурентна мережа (LSTM, GRU чи навіть Vanilla RNN) «прочитує» вихід CNN як послідовність векторів, враховуючи довготривалу залежність між цими векторами в часі. Схему такої інтеграції ілюструють різні варіації, зокрема й гібридна архітектура LSTM і GRU, де показано поєднання декількох рекурентних блоків для послідовної обробки вже вилучених CNN-фільтрів. Хоча в нашому випадку CNN і RNN можна комбінувати й інакше: наприклад, CNN-фронтенд для обробки сигналу, після чого GRU чи LSTM виступають у ролі бектенду.

Однією з перших популярних реалізацій такого підходу стала модель, де CNN-частина (часто ResNet або VGG) зберігає просторову інформацію, перетворює зображення на набір ознак, а потім RNN-частина зчитує цю послідовність ознак, щоби врахувати сусідні кадри або кроки. Із початком «ери відеоаналітики» такі гібриди стали практично стандартом: мережа CNN відповідає за вилучення локальних текстур, форм і кольорів, а рекурентні шарі «агрегують» ці дані в часі, відстежуючи появу нових об'єктів або зміну їхнього стану від кадру до кадру. Це є суттєвою перевагою для завдань динамічного розпізнавання діяльності людей, виявлення аномалій у відеопотоці, а також для формування складних описів («captioning») у відео.

Подібний принцип ефективно працює й за межами суто візуальних даних. Якщо в розпорядженні є набір сенсорних показників, кожен із яких можна подати у вигляді «просторового» вектора ознак, то CNN здатна у певний спосіб «компресувати» ці дані, а далі RNN уже моделює часовий аспект. Зокрема, в медицині для аналізу електрокардіограм (ЕКГ) або електроенцефалограм (ЕЕГ) іноді застосовують короткі «фрагменти» сигналу, які обробляє згорткова частина (шукаючи патерни у формі хвиль), а потім ці вектори, упорядковані за часом, аналізує LSTM чи GRU, щоб простежити глобальний перебіг фізіологічного процесу.

Гібридні архітектури можуть містити і більш «вибагливі» компоненти, як-от механізм уваги (attention), який додатково підсвічує найбільш релевантні часові позиції чи просторові фрагменти. Це значно підвищує адаптивність системи, даючи змогу зосереджуватися на ключових особливостях даних. Утім, у такому разі модель стає доволі великою й може вимагати потужного апаратного забезпечення під час навчання. Тому в практичних застосуваннях часто застосовують «урізані» CNN (MobileNet або іншу легку архітектуру) і GRU замість LSTM, щоб зберегти помірну кількість параметрів.

Таким чином, Hybrid Models поєднують локальні просторові патерни (детектовані CNN) і глобальні часові патерни (змодельовані RNN), даючи змогу отримати потужні рішення для мультимодальних і темпорально-орієнтованих задач. Така інтеграція є одним із найпоширеніших способів зробити класифікаційні системи більш чутливими до контексту, особливо коли контекст розгортається у часі й містить різнорівневі підказки про клас об'єкта чи стан середовища.

6.2 Ensemble Architectures

Паралельно з гібридизацією різних типів нейронних мереж розвивається ідея ансамблів (ensemble learning). У загальному розумінні, ансамбль – це комбінація кількох моделей, які можуть суттєво відрізнятися за архітектурою або ж бути однаковими, але з різними початковими ініціалізаціями чи різними наборами гіперпараметрів. Зміст полягає в тому, що колективне рішення, яке базується на підсумовуванні (або голосуванні) прогнозів різних членів ансамблю, часто буває точнішим і стабільнішим, ніж розв'язок однієї моделі.

В історії машинного навчання ансамблеві методи, зокрема Random Forest чи Gradient Boosting, доволі рано привернули увагу завдяки своїй здатності «усереднювати» помилки окремих слабких моделей. Зі

зростанням популярності глибоких нейронних мереж ідеї ансамблів теж перенеслися в цей контекст. У задачах класифікації зображень, тексту, аудіо, відео тощо нерідко помічають, що ансамбль із кількох CNN або з комбінації CNN і трансформера здатний покращити метрику точності, інколи навіть на кілька відсоткових пунктів. Це особливо відчутно, якщо моделі відрізняються за своєю природою (одна – ResNet, інша – EfficientNet), або якщо вони навчалися на різних фолдах даних і таким чином опанували трохи відмінні патерни.

Ансамблі часто застосовують і в медичних дослідженнях, коли точність моделі має критичне значення. Об'єднання декількох експертних систем дає змогу знизити ризик хибних позитивних чи негативних спрацьовувань, адже помилки однієї мережі можуть бути виправлені консенсусом інших. Така «підстраховка» завдяки різноманітності окремих мереж збільшує надійність загального результату. Проте час інференсу й обсяг ресурсів при використанні ансамблів можуть зростати лінійно з кількістю моделей. Деякі практичні системи, натомість, зберігають лише «полегшений» варіант ансамблю (наприклад, 2–3 моделі замість 5–10), або застосовують механізми згортання (knowledge distillation), аби об'єднати кілька мереж в одну «стиснену» модель.

Інший вектор розвитку ансамблів полягає в їхній спеціальній конфігурації, коли різні моделі отримують дещо відмінну інформацію на вході. Наприклад, у завданнях із сенсорними даними один з членів ансамблю може обробляти «сировинні» показники, а інший – їхню спектральну чи часову ентропію. Також у комп'ютерному зорі інколи застосовується стратегія, коли різні CNN обробляють різні ділянки зображення або застосовують різні масштаби (multiscale approach). Після цього прогнози об'єднуються, щоб отримати більш повну картину. Така схема виявляється особливо корисною, коли класифікацію здійснюють на

неоднорідних зображеннях чи супутникових знімках, де важливими можуть бути і глобальні структури, і дрібні деталі.

Нарешті, у межах ансамблей дедалі популярнішою стає ідея комбінування нейронних мереж із традиційними алгоритмами машинного навчання або зі статистичними методами. У задачах обробки сигналів чи наукової діагностики окрема глибока мережа може бути схильною до помилок через обмежений набір тренувальних даних, тоді як класичні лінійні чи деревоподібні моделі можуть надати вагомий «контроль» і стабільність, якої бракує. Утім, це залежить від галузі застосування: у більшості промислових кейсів простіше поєднувати саме декілька різновидів нейронних мереж, уникаючи складних несумісностей між фреймворками.

6.3 Переваги та виклики інтегрованих рішень

І гібридні моделі, і ансамблі дають суттєвий вииграш у точності та здатності до узагальнення, оскільки поєднують різні сили глибинного навчання. Однак їхній недолік полягає у зростанні складності. У випадку гібридних мереж зростає кількість блоків і потенційно параметрів (CNN-фронтенд, RNN-бектенд тощо), що означає більш тривале навчання й можливість різного роду «втрат градієнта» на межі між двома компонентами. У випадку ансамблів проблема ще масштабніша, адже потрібно навчати й зберігати кілька окремих моделей. Для деяких систем це неприпустимо з міркувань ресурсів або просто занадто дорого. Тому часто шукають компромісні підходи: або зберігають дуже малий склад ансамблю (2–3 різних моделі), або вдосконалюють гібридні компоненти, аби вони були якомога простішими.

Попри це у низці галузей інтегрований підхід є ледь не обов'язковим. Скажімо, в розпізнаванні тексту з природних зображень (OCR для вивісок, книжкових сторінок тощо) класична архітектура CRNN

(Convolutional Recurrent Neural Network) стала стандартом. Вона поєднує згортковий блок для вилучення ознак та бідирекційний LSTM чи GRU для посимвольної обробки, щоб врахувати горизонтальну послідовність символів. А в завданнях автоматичного субтитрування чи відеоаналізу нерідко використовують ансамбль із кількох CNN, кожна з яких вивчає «свій» ракурс даних (колірні канали, оптичний потік тощо), а потім об'єднують прогнози за допомогою RNN, яка відповідає за часовий аспект.

Ще одна сфера широкого застосування – медична діагностика, де універсальні рішення на одному типі мереж часто недосконалі, а складність постановки завдання (шумні дані, різноманітні умови зйомки, рідкісні патології) вимагають найвищого рівня надійності. Там гібриди CNN + RNN чи ансамблі з різних CNN-моделей є доволі звичним явищем, оскільки потреба «не пропустити» жодного випадку патології часто перевищує необхідність максимальної швидкості.

Таким чином, інтегровані системи можуть ефективніше схоплювати багатогранність вхідних даних та досягати більш високої точності класифікації, але водночас вимагають розвиненої інженерної інфраструктури для навчання й підтримки. Часто їхнє застосування виправдане лише тоді, коли ціна помилки дуже висока (медицина, безпека) або коли модель функціонує на серверних ресурсах із достатньо великими обчислювальними потужностями. Проте системи «легких» ансамблів чи компактних гібридних архітектур також знайдені в мобільній ніші, особливо коли необхідно гарантувати стійкість у мінливих умовах.

6.4 Висновки щодо інтегрованих підходів

Гібридні моделі та ансамблі уможливлюють комплексний підхід до задач класифікації, зокрема коли доводиться працювати з різнорідними або багатовимірними даними. Поєднання CNN із RNN дозволяє вирішувати проблеми, де водночас присутні просторові (зображення, кадри відео) й

часові (послідовність кадрів чи сенсорних зчитувань) аспекти. Ансамблі також надають більш високу надійність і узагальнювальну здатність завдяки колективному голосуванню чи усередненню прогнозів кількох моделей.

Незважаючи на те, що тренування та розгортання інтегрованих рішень зазвичай коштує дорожче, вони виправдовують ці витрати у сферах, де стратегічно важливо досягнути максимально можливого рівня точності. Деякі наукові колективи обирають шлях пошуку «комфортного» балансу, використовуючи лише обмежену кількість моделей в ансамблі або спрощені варіанти гібридів. Інші ж, особливо в науково-дослідницькому контексті чи критично важливих галузях (медицина, безпеність автономних систем), готові інвестувати в складні інтегровані системи.

У сукупності з попередніми розділами можна сказати, що сучасне глибинне навчання не обмежується використанням лише однієї архітектури – є тенденція до універсальності й міждоменних рішень. CNN, RNN, трансформери, мобільні та спеціалізовані моделі в різних конфігураціях можуть утворювати інтегровані «екосистеми», які здатні охоплювати широкий спектр прикладних задач. Такі підходи опираються на гнучкість інструментарію та багатий досвід взаємодії різних мереж, що дозволяє створювати доволі складні механізми, але з високим ступенем узагальнення та надійності.

7 ПОРІВНЯННЯ АРХІТЕКТУР

Питання порівняння різних нейронних архітектур для класифікації залишається предметом активних дискусій у науковій спільноті, оскільки кожний підхід має свої сильні та слабкі сторони. З розвитком глибинного навчання з'ясувалося, що немає «єдиного переможця», придатного для всіх типів задач і ресурсних обмежень. На результати впливають масштаб даних, тип вхідного сигналу (зображення, текст, часові ряди), апаратне середовище (потужні сервери чи мобільні пристрої), а також конкретні метрики, за якими оцінюють модель (точність, швидкість, пам'ять, стійкість до перешкод).

Якщо розглянути сучасні сімейства мереж, що найчастіше застосовуються у класифікаційних сценаріях, можна виокремити кілька ключових критеріїв для порівняння. Першим є точність (accuracy або інші метрики на кшталт F1-score, Recall, Precision). Багатошарові перцептрони (MLP) дають прийнятні результати для відносно простих і лінійно податливих задач, але у складніших випадках (наприклад, розпізнавання великої кількості об'єктів на зображеннях) глибокі згорткові мережі (CNN) або трансформери значно перевершують MLP. Серед CNN особливо високих значень точності у конкурсах ImageNet досягали ResNet і EfficientNet, демонструючи, що доробки на кшталт резидуальних з'єднань чи збалансованого масштабування глибини й ширини можуть помітно підвищити якість. Трансформери, зокрема Vision Transformers, на великих датасетах наближаються або перевершують згорткові архітектури, але за умови значних обчислювальних ресурсів.

Наступним критерієм є обчислювальна складність і швидкість роботи. Тут класичні MLP без оптимізацій можуть виявитися надмірно «роздутими» за кількістю параметрів, тоді як CNN пропонують відносно локальну згортку, що краще масштабується. Однак глибокі варіанти CNN

(VGG-19, ResNet-152, EfficientNet-B7) усе одно вимагають серйозних обчислювальних потужностей під час тренування. Трансформери зі своєю матрицею уваги, яка має квадратичну складність відносно довжини послідовності, можуть бути ще вимогливішими, особливо якщо йдеться про великі моделі на кшталт GPT-3 чи 4. У мобільному сегменті перевага схиляється до легких архітектур, представлених MobileNet, GhostNet та подібними, де глибинні та роздільні згортки зменшують кількість FLOPs без різкого падіння якості.

Важливо розглянути також потребу в даних і чутливість до розміру вибірки. Глибокі CNN, особливо ті, що мають сотні шарів, успішно опановують складні патерни, але за умови масштабних наборів даних. Натомість у випадку обмеженої кількості тренувальних прикладів такі мережі можуть перенавчатися або не досягти потрібного рівня узагальнення. Тоді стають у пригоді методи перенесеного навчання (transfer learning), коли модель, попередньо натренована на великому датасеті (наприклад, ImageNet), адаптується під конкретне вузьке завдання. Трансформери також потребують суттєвих обсягів тексту чи зображень, щоб показувати найкращі результати, й часто навчаються в режимі self-supervised. У деяких застосуваннях, особливо коли дані зосереджені в часовій площині, рекурентні мережі LSTM або GRU можуть краще розкривати змістові зв'язки без колосальних обсягів трейнінгу, проте в останні роки їх дедалі частіше заміщують трансформерними варіантами.

Ще один критерій – масштабованість. Якщо модель легко збільшує кількість параметрів, шарів або ширину й залишається стабільною при навчанні, це велика перевага. ResNet продемонструвала, що додавання шарів без ризику деградації стало можливим завдяки резидуальним блокам. EfficientNet увела «compound scaling», аби планомірно нарощувати розмір. Трансформери теж виявилися надзвичайно масштабовними, і GPT-3 із 175 мільярдами параметрів є одним із найпоказовіших прикладів. Однак така

масштабованість веде до колосальних обчислювальних витрат, що не завжди виправдано з погляду практичного застосування.

Значну увагу також приділяють енергоспоживанню і пам'яті, особливо в епоху «зелених» обчислень. Мобільні архітектури на кшталт MobileNet або NASNet, знайдені методами автоматизованого пошуку, стають дедалі актуальнішими для пристроїв із браком ресурсів. Якщо ж мережа розгортається на серверній інфраструктурі, де GPU-ферми здатні забезпечувати величезну пропускну здатність, можна звернутися до «важчих» моделей на кшталт ResNet-152 чи Vision Transformer large-scale. У задачах мультимедійної обробки, де необхідна швидкість у реальному часі, фахівці часто схиляються до компромісних мереж (ResNet-50, MobileNet, EfficientNet-B0), де прийнятний баланс між точністю й продуктивністю.

На додачу до узагальнених характеристик порівняння корисно звернутися до конкретних експериментальних показників. Далі наведена приблизна таблиця результатів для типових класифікаційних задач, таких як ImageNet (зображення високої роздільності), CIFAR-10 (малий набір 32×32) та IMDB або інші текстові датасети (для мовних архітектур). Потрібно зауважити, що точні числа можуть варіюватися залежно від гіперпараметрів і самого датасету, але загальна тенденція залишатиметься подібною.

Згідно з таблицею 1, VGG або AlexNet уже менше застосовують у нових проєктах через великий розмір і відносно нижчу продуктивність порівняно з ResNet чи EfficientNet. MobileNet виявляється компромісом для мобільних середовищ. У задачах на кшталт CIFAR-10 навіть невеликі моделі можуть досягати близької до 95–97 % точності. Для ImageNet більше значення мають розумні техніки масштабування або резидуальні з'єднання, оскільки це дозволяє суттєво поглибити мережу, не втрачаючи якість градієнтів. Трансформери (ViT) здатні перевершувати CNN, коли

йдеться про величезні набори зображень (наприклад, JFT-300M) і коли в розпорядженні є потужні обчислювальні кластери, але на середніх датасетах іноді їм бракує даних, і вони поступаються витонченим CNN.

Таблиця 1 – Приклад узагальнених результатів для типових задач класифікації

Архітектура	Точність на ImageNet / CIFAR-10 (у %)	Приблизна кількість параметрів	Примітки
MLP (кілька шарів)	60–80 / 85–90 (залежно від глибини та перенесеного навчання)	~10–20 млн	Проста структура, легко перенавчається; у складних задачах поступається CNN і трансформерам
LeNet / AlexNet / VGG (класичні CNN)	55–90 / 90–93	5–140 млн	LeNet дуже компактна, AlexNet і VGG доволі ресурсомісткі; застарілі порівняно з ResNet чи EfficientNet
ResNet (50–152 шарів)	75–78 (ImageNet top-1) / 94–96	26–60 млн	Висока точність, резидуальні блоки дозволяють масштабувати глибину без втрати стабільності
EfficientNet (B0–B7)	77–85 / 95–97	5–66 млн	Збалансоване масштабування глибини й ширини, вигідне співвідношення точності та FLOPs
MobileNet (V1–V3)	67–75 / 91–95	2–6 млн	Полегшена архітектура, придатна для мобільних пристроїв, компроміс між швидкістю та точністю
LSTM / GRU (RNN) для зображень	– / –	Від кількох до десятків млн	Не застосовуються безпосередньо до зображень, але у текстових / аудіо задачах можуть бути дуже ефективними
Transformers (ViT, BERT, GPT)	78–90+ / – (ViT large)	Від десятків млн до мільярдів	Висока точність у разі великомасштабних даних, вимагають значних апаратних ресурсів

Окрім точності, в реальних задачах важливу роль відіграє час обробки одного прикладу, пропускна здатність (кількість зразків за секунду) під час інференсу та можливість ефективно розподілити обчислення на GPU чи TPU. ResNet-50, EfficientNet-B0 або MobileNetV2

часто обирають як основи для перенесеного навчання через баланс точності й швидкості, тоді як більш громіздкі варіанти ResNet-152 чи EfficientNet-B7 використовують, коли критичні найкращі показники, а обчислювальні ресурси не є вузьким місцем. RNN чи LSTM-мережі здебільшого порівнюють окремо у мовних чи часових завданнях, де CNN не здатні безпосередньо обробляти послідовності, а трансформери можуть вигравати тільки за достатньої кількості даних і апаратури. Мобільні й NAS-архітектури набувають актуальності, коли стоїть завдання портативності й швидкості в обмежених середовищах.

Загальна тенденція полягає в тому, що вибір архітектури визначається насамперед природою даних, доступними обчислювальними ресурсами та вимогами до продуктивності. Якщо в проєкті критичною є точність на мультимільйонному датасеті, мають сенс найглибші архітектури (ResNet-152, EfficientNet-L2, ViT large). Якщо ж стоїть питання швидкого прототипування або запуску в реальному часі на смартфоні, переважають MobileNet або менші моделі ResNet. У сфері текстової класифікації, аналізу тональності чи машинного перекладу лідирують трансформери (BERT, GPT), тоді як для коротких послідовностей або коли потрібні невеликі ресурси, варто розглянути RNN (LSTM, GRU).

Таким чином, жодна модель не є універсально найкращою. Підсумкова ефективність залежить від кількох факторів. По-перше, від величини й різноманіття датасету. По-друге, від цілей: чи важлива мінімізація пропущених прикладів (Recall) чи, можливо, найвища точність за умови великої моделі. По-третє, від реальних обмежень, як-от пам'ять, час, розмір моделі. По-четверте, від можливості масштабного перенесеного навчання, коли попередньо натренована велика мережа може суттєво прискорити адаптацію до вузькопрофільних даних. Усе це впливає на вибір оптимального сімейства архітектур для конкретної задачі класифікації.

ВИСНОВОК

Феномен глибинного навчання, що став фундаментальним рушієм проривів у сфері штучного інтелекту, значною мірою залежить від вибору архітектури нейронної мережі. Кожна з розглянутих у цьому огляді моделей – від простого перцептрона й класичного MLP до складних трансформерів та інтегрованих рішень – має власне місце в історії й практиці машинного навчання. Тому належить уважно оцінювати їхні сильні та слабкі сторони, а також урахувувати специфіку конкретних задач і наявних ресурсів, перш ніж визначати оптимальне рішення для класифікації.

Загальний шлях розвитку нейронних архітектур починається з перцептрона, який хоч і давно вийшов за межі активного використання у глибоких сценаріях, проте свого часу заклав основи лінійного розділення класів і сформував початкові підходи до навчання вагових коефіцієнтів. Його еволюційним продовженням став багатошаровий перцептрон (MLP). Модель MLP може ефективно працювати з відносно невеликими наборами вхідних ознак і непогано справлятися із задачею базової нелінійної класифікації, утім не здатна належно охоплювати складні просторові чи часові залежності без суттєвого зростання кількості параметрів. У багатьох сучасних кейсах великі мережі MLP програють глибоким CNN або трансформерам, проте в освітніх цілях і невеликих комерційних застосуваннях досі можуть бути корисним вибором.

Згорткові нейронні мережі (CNN) стали квінтесенцією задач комп'ютерного зору. За допомогою механізму локальних рецептивних полів і згорткових фільтрів вони навчилися автоматично вилучати просторові патерни зображень, демонструючи високу точність класифікації, а також універсальність застосування (від розпізнавання рукописів і медичних знімків до аналізу відеопотоків). Від LeNet, що

обробляла рукописні цифри, до ResNet і EfficientNet, які навчилися долати затування градієнтів і збалансовувати глибину з шириною, згорткові мережі виробили набір потужних евристик для розв'язання візуальних задач. Сильні сторони CNN полягають у їхній здатності до ієрархічного вилучення ознак, природній обробці просторової структури та сталій точності на великих датасетах на кшталт ImageNet. Водночас слабким місцем є висока ресурсоемність у глибоких конфігураціях (наприклад, VGG-19, ResNet-152 або EfficientNet-B7), складність виконання реального часу на мобільних пристроях без спеціальних оптимізацій, а також необхідність великих обсягів даних для належного навчання, якщо немає можливості застосувати перенесене навчання.

Рекурентні нейронні мережі (RNN) стали відповіддю на потреби в аналізі послідовних даних: тексту, аудіо, часових рядів. Починаючи від Vanilla RNN, що страждала від проблеми затування градієнтів, дослідники перейшли до LSTM і GRU, де спеціальні вентиляльні механізми дозволили подовжити «пам'ять» моделі та долати втрату далекої інформації. Ці підходи чудово виявляють залежності в мовних задачах, музичних послідовностях, датасетах сенсорних сигналів. Сильна сторона RNN – ефективна робота з часовими контекстами та порівняно компактні розміри моделі, особливо в GRU. Слабка сторона – обмежена здатність до паралелізації, висока складність навчання на дуже довгих послідовностях, а також інколи поступається сучасним трансформерам у завданнях машинного перекладу та широкомасштабного аналізу тексту. Попри це LSTM і GRU не втратили актуальності в низці прикладних сфер і далі застосовуються там, де дані не настільки масштабні, а також коли критичним є простіший цикл інференсу.

Трансформери (Transformers) стали свого роду «революцією» в обробці тексту та, згодом, у зорових задачах. Архітектура, вільна від рекурентного ланцюжка і згорткових операцій, вибудована навколо

механізму self-attention, виявилася здатною ефективно паралелізувати обчислення і охоплювати довготривалі залежності. Моделі BERT, GPT, Vision Transformer (ViT) чи Swin Transformer показали, що увага може стати центральним компонентом для аналізу різних типів даних. Перевага трансформерів – висока точність і можливість масштабування, що відкриває простір для навчання мовних моделей із мільярдами параметрів. Слабке місце – критична залежність від величезних обсягів тренувального матеріалу, велике споживання ресурсів і значно ускладнене розгортання у пристроях з обмеженнями. На додачу, при недостатній кількості даних деякі трансформери можуть переобтяжуватися або навіть відставати від компактніших CNN чи RNN.

Серед спеціалізованих і мобільних рішень, зокрема MobileNet, GhostNet, MnasNet, NASNet та інших, особливу увагу приділено пошуку балансу між точністю та економністю обчислень. Це робить їх ключовими для завдань, де модель має функціонувати в реальному часі на смартфоні чи вбудованій системі з обмеженою пам'яттю та енергоспоживанням. У цих архітектурах використано такі евристики, як факторизація згорток (depthwise separable convolution) чи автоматичний пошук топології блоків (Neural Architecture Search). Основна перевага – можливість зберігати відносно високий рівень точності при суттєвому зниженні FLOPs і кількості параметрів, водночас найбільше їхнє обмеження – певна втрата максимально можливої продуктивності на користь компактності, тобто передбачуваний компроміс.

Отже, кожен клас моделей має свої переваги. Згорткові мережі відмінні для роботи з просторовою інформацією, рекурентні – для часового контексту, трансформери – для комплексних залежностей і масштабованих рішень, а мобільні архітектури – для середовищ із низькими ресурсами. Існують також інтегровані підходи, коли CNN і RNN чи CNN і трансформери об'єднуються задля досягнення кращої

узагальнювальної здатності; або коли застосовують ансамблі з кількох різних моделей, щоб підвищити надійність.

Висновки щодо архітектури для різних типів задач

Якщо головна мета – обробити велику кількість зображень із максимально можливою точністю, варто обрати глибоку згорткову мережу (ResNet, EfficientNet). Якщо дозволяють ресурси і є дуже великий датасет, Vision Transformer може перевершити CNN, але лише за умови правильного налаштування й застосування масштабних корпусів. Усі ці моделі легко адаптуються шляхом перенесеного навчання, що особливо вигідно в різних промислових сценаріях. Якщо потрібно вирішувати мовні завдання (класифікація тексту, аналіз тональності, питання-відповідь), BERT чи GPT-похідні моделі є першочерговим варіантом через їхню здатність оперувати значними обсягами тексту і глибоко розуміти контекст. Для менших наборів даних, коротких послідовностей або коли недоступні ресурси для тренування гігантських трансформерів, LSTM чи GRU можуть надати ефективне і швидке розв’язання, зберігаючи гнучкість у роботі з часовими характеристиками. Коли стоїть завдання запуску на мобільному пристрої з обмеженою пам’яттю, неможливо розгортати надто великі моделі. У цьому випадку доречно звернутися до MobileNet чи аналогічних «легких» рішень, що балансують між швидкістю обчислень і якістю класифікації. Якщо важливими є конкретні метрики, наприклад ліміти часу відгуку (latency) чи енергоефективність, можна розглядати варіанти NAS-архітектур, які знайшлися евристично і дають змогу динамічно налаштувати модель під конкретну платформу. Для задач, що містять довготривалі залежності в часі (наприклад, передбачення складних часових рядів, мовний переклад великих текстів), варто розглянути або трансформери з механізмом self-attention, або, якщо ресурси обмежені, рекурентні підходи на кшталт LSTM. Інтегровані моделі (CNN+RNN,

CNN+Transformer) зазвичай використовуються в мультимодальних сценаріях, як-от опис відео, де потрібно обробити і зображення (кадри), і часову послідовність. У сегменті високої точності, особливо там, де ціна помилки надзвичайно висока (діагностика в медицині, контроль якості у промисловості), корисно створювати ансамблі з декількох моделей. Такий підхід збільшує надійність системи, оскільки помилки окремої мережі можуть компенсуватися голосуванням чи агрегацією з іншими. Недолік – збільшення сукупних ресурсів навчання та інференсу, тому ансамблі частіше застосовують в офлайнних або критичних за точністю системах, а не в реальному часі.

Загалом можна сказати, що грамотно вибрана архітектура повинна відповідати характеру даних, масштабам і цілям проєкту. Сьогодні наукова й інженерна спільнота накопичила величезний багаж практичних методик, які дають змогу налаштовувати і комбінувати різні мережі, враховуючи багатоступеневі критерії від точності до часу інференсу, від пам'яті до складності реалізації. Із появою універсальних фреймворків (TensorFlow, PyTorch) і хмарних платформ тренування та розгортання масштабних моделей стало доступним навіть для відносно невеликих команд. Водночас розвиваються методи компресії (pruning, quantization, distillation), що спрощують перенос глибоких моделей у мобільну й вбудовану сферу.

У підсумку глибоке навчання наразі є гнучкою екосистемою, у межах якої перцептрони, CNN, RNN і трансформери не конкурують, а скоріше доповнюють одне одного, пристосовуючись під різні задачі й сценарії. Сучасна практика показує, що успішний результат часто полягає саме в компетентному поєднанні різних технологій. Але навіть якщо проєкт орієнтований лише на класифікацію зображень або лише на роботу з текстом, вибір правильної архітектури – невіддільна частина підвищення точності й продуктивності. Враховуючи, що прогрес у глибокому навчанні триває, наступні розробки можуть принести ще більш

«універсальні» підходи. Проте навіть сьогодні в арсеналі дослідника існує широкий спектр випробуваних рішень: перцептрони для простих сценаріїв, ResNet чи EfficientNet для візуальних даних, LSTM і GRU для невеликих послідовностей, потужні трансформери для масштабних текстових чи мультимодальних задач і компактні MobileNet/NASNet для мобільних. Кожен із цих варіантів здатний стати основою для високоякісної системи класифікації, якщо застосовувати його з урахуванням ключових вимог проекту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Elsken T. Neural Architecture Search: A Survey [Electronic resource] / T. Elsken, J. H. Metzen, F. Hutter // Journal of Machine Learning Research. — Electronic text data. — 2019. — Vol. 20, No. 55. — С. 1–21. — Regime of access: <https://arxiv.org/abs/1808.05377>, free (application date: 20.05.2025). — Header from the screen.
2. Tan M. EfficientNetV2: Smaller Models and Faster Training [Electronic resource] / M. Tan, Q.V. Le // Proceedings of the 38th International Conference on Machine Learning (ICML). — Electronic text data. — 2021. — P. 10096–10106. — Regime of access: <https://proceedings.mlr.press/v139/tan21a.html>, free (application date: 20.05.2025). — Header from the screen.
3. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale [Electronic resource] / A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn [et al.] // International Conference on Learning Representations (ICLR). — Electronic text data. — 2021. — Regime of access: <https://arxiv.org/abs/2010.11929>, free (application date: 20.05.2025). — Header from the screen.
4. Liu Z. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows [Electronic resource] / Z. Liu, Y. Lin, Y. Cao, H. Hu [et al.] // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). — Electronic text data. — 2021. — P. 10012–10022. — Regime of access: <https://arxiv.org/abs/2103.14030>, free (application date: 20.05.2025). — Header from the screen.
5. Simple Framework for Contrastive Learning of Visual Representations [Electronic resource] / T. Chen, S. Kornblith, M. Norouzi, G. Hinton // Proceedings of the 37th International Conference on Machine Learning (ICML). — Electronic text data. — 2020. — P. 1597–1607. —

Regime of access: <https://proceedings.mlr.press/v119/chen20j.html>, free (application date: 20.05.2025). – Header from the screen.

6. Momentum Contrast for Unsupervised Visual Representation Learning [Electronic resource] / K. He, H. Fan, Y. Wu, S. Xie, R. Girshick // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). — Electronic text data. — 2020. — P. 9729–9738. — Regime of access: <https://arxiv.org/abs/1911.05722>, free (application date: 20.05.2025). — Header from the screen.

7. Searching for MobileNetV3 [Electronic resource] / A. Howard, M. Sandler, B. Chen, W. Wang [et al.] // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). — Electronic text data. — 2019. — P. 1314–1324. — Regime of access: <https://arxiv.org/abs/1905.02244>, free (application date: 20.05.2025). — Header from the screen.

8. Selective Kernel Networks [Electronic resource] / X. Li, W. Wang, X. Hu, J. Yang // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). — Electronic text data. — 2019. — P. 510–519. — Regime of access: <https://arxiv.org/abs/1903.06586>, free (application date: 20.05.2025). — Header from the screen.

9. GhostNet: More Features from Cheap Operations [Electronic resource] / K. Han, Y. Wang, Q. Xu, C. Xu, Q. Tian // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). — Electronic text data. — 2020. — P. 1580–1589. — Regime of access: <https://arxiv.org/abs/1911.11907>, free (application date: 20.05.2025). — Header from the screen.

10. Tan M. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [Electronic resource] / M. Tan, Q. Le // Proceedings of the 36th International Conference on Machine Learning (ICML). — Electronic text data. — 2019. — P. 6105–6114. — Regime of access:

<https://arxiv.org/abs/1905.11946>, free (application date: 20.05.2025). – Header from the screen.

11. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions [Electronic resource] / W. Wang, E. Xie, X. Li, D. Fan [et al.] // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). — Electronic text data. — 2021. — P. 568–578. — Regime of access: <https://arxiv.org/abs/2102.12122>, free (application date: 20.05.2025). – Header from the screen.

12. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches [Electronic resource] / K. Cho, B. Merriënboer, D. Bahdanau, Y. Bengio // Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation. — Electronic text data. — 2014. — P. 103–111. — Regime of access: <https://arxiv.org/abs/1409.1259>, free (application date: 20.05.2025). – Header from the screen.

13. Hochreiter S. Long Short-Term Memory [Electronic resource] / S. Hochreiter, J. Schmidhuber // Neural Computation. — Electronic text data. — 1997. — Vol. 9, No. 8. — C. 1735–1780. — Regime of access: <https://www.bioinf.jku.at/publications/older/2604.pdf>, free (application date: 20.05.2025). – Header from the screen.

14. Simonyan K. Very Deep Convolutional Networks for Large-Scale Image Recognition [Electronic resource] / K. Simonyan, A. Zisserman // arXiv preprint arXiv:1409.1556. — Electronic text data. — 2014. — Regime of access: <https://arxiv.org/abs/1409.1556>, free (application date: 20.05.2025). – Header from the screen.

ДОДАТОК А

Додаткові ресурси, які можуть стати корисними у процесі практичного впровадження та подальшого вдосконалення моделей

1. Посилання на офіційні репозиторії та модельні «зоопарки»:

– TensorFlow Model Garden – набір прикладів і реалізацій CNN (ResNet, EfficientNet, MobileNet), трансформерів (BERT) та інших моделей під платформу TensorFlow [Електрон. ресурс]. – Режим доступу: <https://github.com/tensorflow/models>;

– PyTorch Vision Models – реалізації базових архітектур для зображень (ResNet, VGG, MobileNet, EfficientNet тощо) [Електрон. ресурс]. – Режим доступу: <https://docs.pytorch.org/vision/main/models.html>;

– Hugging Face Transformers – репозиторій із великою кількістю готових до використання трансформерів (BERT, GPT, ViT, T5) для обробки тексту та зображень у PyTorch і TensorFlow [Електрон. ресурс]. – Режим доступу: <https://huggingface.co/models>.

2. Приклади готового перенесеного навчання:

– Keras Applications – офіційні зразки попередньо натренованих моделей (VGG, ResNet, MobileNet, EfficientNet) у Keras, які можна швидко завантажити й донавчати на власних даних [Електрон. ресурс]. – Режим доступу: <https://keras.io/api/applications/>;

– PyTorch Hub – зручна платформа для підключення готових моделей до свого коду однією командою, містить приклади реалізацій і перенесеного навчання [Електрон. ресурс]. – Режим доступу: <https://pytorch.org/hub/>.

3. Інструментарії для оптимізації та деплоюменту:

– TensorFlow Model Optimization Toolkit – набір інструментів для поліпшення ефективності моделей (прискорення, зменшення розміру)

через pruning, quantization, clustering [Електрон. ресурс]. – Режим доступу: https://www.tensorflow.org/model_optimization;

- PyTorch Quantization – механізми 8-бітної та динамічної квантованої обробки, що дозволяють зменшити використання пам'яті та прискорити інференс на CPU [Електрон. ресурс]. – Режим доступу: <https://pytorch.org/docs/stable/quantization.html>;

- ONNX Runtime – фреймворк для ефективного розгортання нейронних мереж (CNN, RNN, Transformers) на різних платформах (CPU, GPU, мобільні) [Електрон. ресурс]. – Режим доступу: <https://onnxruntime.ai/>.

4. Додаткові приклади коду й туторіали:

- Kaggle Kernels – безліч готових зошитів із прикладами реалізацій різних архітектур (CNN, RNN, Transformers) на відкритих датасетах. Добре підходять для швидкого старту або адаптації до власних задач [Електрон. ресурс]. – Режим доступу: <https://www.kaggle.com/code>;

- Papers with Code – сервіс, де зібрано академічні статті й пов'язані з ними реалізації коду. Дозволяє побачити рейтинги результатів на популярних бенчмарках (ImageNet, COCO, GLUE тощо) й перевірити працездатність на практиці [Електрон. ресурс]. – Режим доступу: <https://paperswithcode.com/>.

5. Додаткові відомості про компресію та мобільні рішення:

- TensorFlow Lite – офіційний фреймворк для запуску нейронних мереж (MobileNet, EfficientNet та ін.) на мобільних і вбудованих пристроях із обмеженими ресурсами [Електрон. ресурс]. – Режим доступу: <https://www.tensorflow.org/lite>;

- Core ML (Apple) – середовище для розгортання моделей iOS-розробниками. Багато архітектур (ResNet, MobileNet) підтримуються офіційно з доволі доброю документацією [Електрон. ресурс]. – Режим доступу: <https://developer.apple.com/documentation/coreml>;

– Neural Architecture Search (NAS) Frameworks – приклади проєктів із NAS-підходами, що створюють оптимізовані під ресурси й цільові метрики архітектури (включно з NASNet-похідними) [Електрон. ресурс]. – Режим доступу: <https://autokeras.com/>.

6. Відкриті датасети для експериментів:

– ImageNet – головний бенчмарк для згорткових мереж, зображення високої роздільності [Електрон. ресурс]. – Режим доступу: <http://www.image-net.org/>;

– CIFAR-10, CIFAR-100 – популярні середні датасети (32×32) для швидких експериментів над архітектурами CNN [Електрон. ресурс]. – Режим доступу: <https://www.cs.toronto.edu/~kriz/cifar.html>;

– MNIST – рукописні цифри 28×28, підходить для найперших тестів і демонстрацій [Електрон. ресурс]. – Режим доступу: <http://yann.lecun.com/exdb/mnist/>;

– IMDB Reviews – набір текстів для аналізу настрою (класичний приклад застосування LSTM, GRU чи трансформерів) [Електрон. ресурс]. – Режим доступу: <https://ai.stanford.edu/~amaas/data/sentiment/>;

– WMT Datasets – набори для машинного перекладу, де можна випробувати рекурентні моделі чи трансформери [Електрон. ресурс]. – Режим доступу: <http://www.statmt.org/wmt/>.

Електронне наукове видання

БРЕДІХІН Володимир Михайлович,
ДІДОК Володимир Олегович,
ЧЕРКАСОВА Валерія Володимирівна

**РОЗВИТОК АРХІТЕКТУР НЕЙРОННИХ МЕРЕЖ:
ВІД КЛАСИЧНИХ ДО ІННОВАЦІЙНИХ**

МОНОГРАФІЯ

Відповідальний за випуск *В. В. Черкасова*
Редактор *О. В. Михаленко*
Комп'ютерне верстання *В. М. Бредіхін*

Підп. до друку 20.10.2025. Формат 60 × 84/16.
Ум. друк. арк. 4,6.

Видавець і виготовлювач:
Харківський національний університет
міського господарства імені О. М. Бекетова,
вул. Чорноглазівська, 17, Харків, 61002.
Електронна адреса: office@kname.edu.ua
Свідоцтво суб'єкта видавничої справи:
ДК № 8386 від 14.07.2025.