УДК 519.6 *http://dx.doi.org/10.30970/vam.2026.36.00000*

ОПТИМІЗАЦІЯ МОДЕЛЕЙ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ ОБЧИСЛЮВАЛЬНО ОБМЕЖЕНИХ ПРИСТРОЇВ

А. Грабовецький, М. Баранов

Львівський національний університет імені Івана Франка,
вул. Університетська 1, Львів, 79000, Україна,
e-mail: andrii.hrabovetskyi@lnu.edu.ua, mykola.baranov@lnu.edu.ua

З огляду на стрімкий розвиток концепції периферійних обчислень, критично важливим завданням постає розгортання складних моделей машинного зору на пристроях з обмеженими апаратними можливостями. У роботі розглянуто методи оптимізації згорткових нейронних мереж для підвищення ефективності розпізнавання об'єктів при обмежених обчислювальних ресурсах. Проаналізовано існуючі підходи до стиснення моделей, виявлено їхні переваги та недоліки у контексті використання пам'яті та швидкодії в умовах обмежених обчислювальних ресурсів. Особливу увагу приділено вирішенню проблеми нестабільності часу обробки даних, яка часто виникає при розгортанні нейронних мереж на пристроях з малою потужністю. В основу дослідження покладено аналіз архітектури ShuffleNet та розробку вдосконаленого рішення на базі моделі RepVGG. Дослідження показало, що метод перемішування каналів моделі ShuffleNet зменшує обчислювальну складність, проте стикається з практичними проблемами перерозподілу даних. Використання методу дистиляції знань у поєднанні з процедурою квантування дало змогу побудувати модель, що має значно менше відхилення часу виконання та забезпечує прогнозовану швидкість роботи системи. Результати тестування підтверджують, що отримана модель перевершує за точністю квантизовану ShuffleNet, демонструючи при цьому вищу стабільність показників та раціональніше використання апаратних потужностей. Такий підхід дозволяє суттєво знизити обчислювальні витрати без критичних втрат якості розпізнавання, що є важливим для систем реального часу.

Ключові слова: згорткові нейронні мережі, оптимізація, дистиляція знань, обрізка, квантування.

1. Вступ

Останніми роками галузь штучних нейронних мереж для задач комп'ютерного зору зазнала значного розвитку. Кількість сфер застосування цих технологій постійно збільшується – це зумовило появу складних архітектур з мільярдами параметрів. Проте, недоліком цього прогресу стало значне зростання вимог щодо обчислювальних ресурсів та об'єму пам'яті. Втім, поруч з нарощуванням потужності цих моделей повстає проблема практичної імплементації. Висока точність цих моделей часто супроводжується більшим часом прямого проходу, через що прикладне застосування унеможливлюється або стає неефективним. Тому методи оптимізації нейронних мереж, спрямовані на зменшення складності моделей без суттєвої втрати точності є актуальним напрямом досліджень.

2. Згорткові нейронні мережі

Згорткові нейронні мережі (ЗНМ, англ. *convolutional neural network, CNN*) – мережі, архітектура глибокого навчання яких пристосована для задач комп'ютерного зору, таких як розпізнавання чи класифікація об'єктів. На відміну від

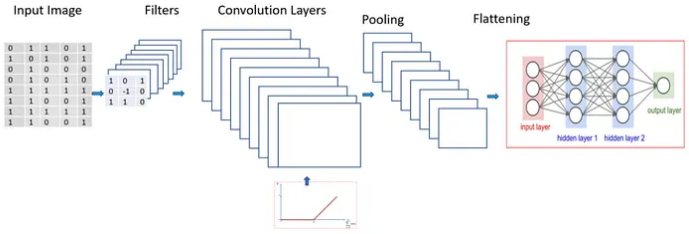

Рис. 1. Архітектура ЗНМ

стандартних багатошарових перцептронів, ЗНМ оперують даними у вигляді багатовимірних тензорів, завдяки чому є можливість зберігати та обробляти просторову структуру зображень у їхньому природньому вигляді, без необхідності попереднього перетворення у одновимірні вектори. Нейрони у згорткових шарах підключаються лише до локальних вікон – невеликих ділянок вхідного зображення. Завдяки цьому мережа здатна ефективно виділяти локальні ознаки, такі як лінії або кути. Один і той самий фільтр застосовується до всіх частин зображення. Цей фільтр називається ядром згортки. Завдяки ньому, кількість параметрів мережі в порівнянні з повнозв'язними шарами (де кожен нейрон має власний набір ваг для усіх пікселів) зменшується кардинально. Операції підвибірки з використанням ядра дозволяють зменшувати розмірність ознакових карт, завдяки чому модель стає інваріантною до невеликих зсувів або спотворень на зображенні.

На рис. 1 можна побачити узагальнену архітектуру ЗНМ. Далі у роботі буде наведено розбір двох моделей – ShuffleNet та RepVGG.

2.1. ShuffleNet

Shufflenet – ефективна архітектура ЗНМ, розроблена спеціально для обчислювально обмежених пристроїв. Основною метою створення цієї моделі була мінімізація кількості операцій з плаваючою комою при збережені високої точності. Головна ідея полягає у вирішені проблеми високої обчислювальної вартості стандартних 1x1 згорток, які у багатьох сучасних архітектурах споживають значну частку ресурсів. Для подолання цього бар'єру ShuffleNet використовує два ключові механізми – групові точкові згортки та операцію перемішування каналів.

У звичайних мережах класичні операції згортки є більш затратними в порівняння з груповими згортками, бо вони з'єднують кожен вхідний канал з кожним вихідним. Групова реалізація розбиває ці канали на декілька ізольовані підмножин. Кожна згортка працює лише у межах своєї групи. Таким чином, обчислювальна складність зменшується пропорційно кількості груп, наприклад, теоретична складність операції 1x1 зменшиться у 8 разів, якщо розділити канали на 8 груп. Однак така ізоляція створює проблему – вихідні дані певної групи отримуються лише з обмеженої підмножини вхідних каналів, і крос-канальна взаємодія ознак блокується, тому суттєво погіршується репрезентативність моделі. Саме для розв'язання цієї проблеми впроваджується механізм перемішування каналів.

Без цього перемішування мережа фактично розкладається на декілька вузьких мереж, які не знають про існування ознак одна одної. Перемішування діє як детермінована перестановка – на вхід подаються вихідні канали від кожної групи,

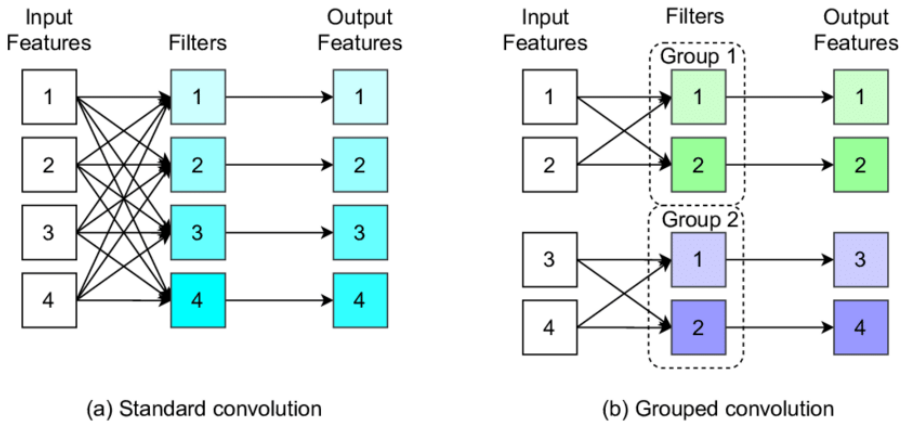

Рис. 2. Різниця між звичайною згорткою (a) та груповою (b)

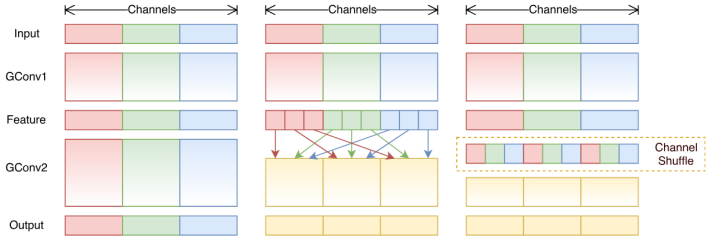

Рис. 3. Перемішування каналів. GConv – групова згортка

розділяються та перерозподіляються таким чином, що наступний шар групових згорток отримує дані від кожної групи попереднього шару. Завдяки цьому між групами відбувається повноцінний груповий обмін, і при цьому зберігається низька обчислювальна вартість, притаманна груповим операціям. Головним компонентом архітектури є ShuffleNet Unit. Це спеціалізований залишковий блок побудований за принципом пляшкового горла.

У базовому варіанті цей блок починається з групової згортки, після якої йде оператор перемішування каналів для відновлення зв'язності. Далі застосовується глибинно-роздільна згортка 3x3, яка обробляє просторові ознаки, а завершується блок ще однією групою згорток 1x1 для відновлення розмірності. Якщо блок виконує функцію зменшення роздільної здатності (Stride 2), до структури додається середня підвибірка на шляху швидкого з'єднання, а замість звичайного додавання ознак використовується їхня конкатинація, завдяки якій можна ефективніше збільшити ширину каналів.

Переваги ShuffleNet. Головною перевагою ShuffleNet є її безпрецидентна здатність до масштабування в умовах обмежениих обчислювальних можливостей. Завдяки використанню групових згорток 1x1 мережа може генерувати складніші репрезентації даних, не виходячи за межі можливостей умовного мобільного процесора. Механізм перемішування каналів елегантно вирішує проблему ізоляції інформації, тобто забезпечує глобальний зв'язок між ознаками, не додаючи при

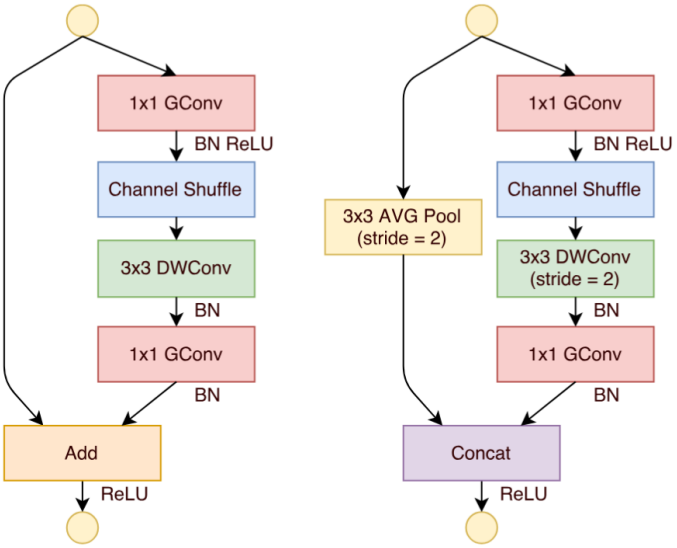

Рис. 4. ShuffleNet Unit

цьому нових параметрів, завдяки чому ShuffleNet не займатиме багато місця у пам'яті. Крім того, моделі на даній архітектурі демонструють стійкість до перенавчання на малих наборах даних через свою розріджену структуру, яка діє як механізм регуляризації.

Недоліки ShuffleNet. Попри високу теоретичну ефективність, ShuffleNet має певні практичні недоліки, пов'язані з особливостями сучасного апаратного забезпечення. Операція перемішування та групові згортки створюють значне навантаження на підсистему пам'яті через складний доступ до даних. Тому, реальна швидкість роботи на деяких GPU або NPU може бути повільнішою, ніж у простіших архітектур з більшою кількістю операцій, але лінійним доступом до пам'яті. Також варто зазначити, що при екстримальному зменшенні кількості груп, каналів, або при грубому квантуванні, точність моделі з архітектурою ShuffleNet може деградувати швидше, ніж у повнозв'язних аналогів, бо здатність мережі вловлювати тонкі кореляції між каналами обмежена структурою перемішування.

2.2. RepVGG

RepVGG – архітектура ЗНМ, яка пропонує унікальний компроміс для вирішення дилеми складності та швидкості. Вона була розроблена з метою повернути популярність простим архітектурам, подібним до оригінальної VGG, але з потужністю сучасних залишкових мереж. Основна ідея RepVGG полягає в концепції структурної перепараметризації, завдяки якій архітектура отримує переваги складних багатогілкових структур під час навчання, та високу швидкість моделі під час інференсу. Реалізація RepVGG основана на математичній лінійності згорткових операцій, завдяки якій можна трансформувати складну топологію навчання у максимально спрощену структуру інференсу.

Під час фази тренування кожен блок складається з трьох паралельних гілок – основної згортки 3x3, допоміжної згортки 1x1 та ідентичного зв'язку, кожна з цих

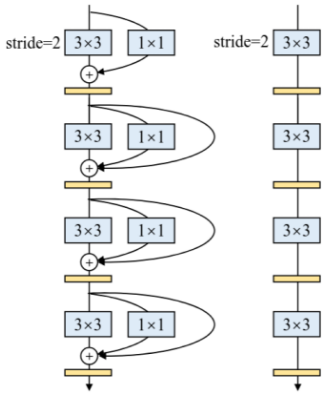

Рис. 5. Перепараметризація RepVGG

гілок супроводжується шаром пакетної нормалізації. Завдяки такій надмірності створюється складний ландшафт втрат, який полегшує збіжність моделі та дає змогу досягти високої точності. Для фінального використання модель проходить етап структурованої перепараметризації, під час якого модифікуються ваги ядра згортки та обчислюється новий вектор зміщення. Таким чином запобігаються обчислювальні витрати на нормалізацію під час роботи. Математично ці операції виглядають наступним чином: нехай x – результат згортки. Тоді операція пакетної нормалізації описується формулою:

$$\mathrm{BN}(x) = \gamma \cdot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{1}$$

де γ – параметр масштабування, β – параметр зсуву що вивчається, μ – середнє значення активацій, накопичене під час тренування, σ^2 – дисперсія активацій, накопичена під час тренування, ϵ – константа, що запобігає діленню на нуль. Для перетворення ваг згортки W та зміщення b у нові ваги $\hat{W}$ та $\hat{b}$, підставимо вираз згортки у формулу пакетної нормалізації, та отримуємо наступні формули:

$$\begin{aligned} \hat{W} &= W \cdot \frac{\gamma}{\sqrt{\sigma^2 + \epsilon}} \\ \hat{b} &= \frac{(b - \mu) \cdot \gamma}{\sqrt{\sigma^2 + \epsilon}} + \beta \end{aligned} \tag{2}$$

Після того, як кожна з гілок перетворена на окрему пару $\{\hat{W}, \hat{b}\}$, вони додаються одна до одної:

$$\begin{aligned} W_{final} &= \hat{W_{\text{3x3}}} + W_{\text{1x1\_to\_3x3}} + W_{\text{id\_to\_3x3}} \\ b_{final} &= \hat{b_{\text{3x3}}} + \hat{b_{\text{1x1}}} + \hat{b_{\text{id}}} \end{aligned} \tag{3}$$

У результаті три окремі обчислювальні потоки об'єднуються в один стандартний шар згортки 3x3. Фінальна модель стає абсолютно лінійною послідовністю шарів, позбавленою розгалужень, завдяки чому апаратне забезпечення може використовувати максимальну пропускну здатність пам'яті.

Переваги RepVGG. Головна перевага RepVGG полягає в її швидкості на етапі виконання. Через перепараметризацію модель перетворюється на просту

послідовність згорток 3x3, і таким чином вона стає більш дружньою до апаратного забезпечення. Сучасні архітектури GPU та NPU оптимізовані саме для щільних згорток 3x3, завдяки чому на практиці RepVGG часто працює швидше в порівнянні з інакшими архітектурами аналогічної точності. Окрім того, відсутність допоміжних гілок під час роботи суттєво знижує споживання оперативної пам'яті, оскільки системі не потрібно тримати результати декількох гілок для їх подальшого підсумовування. Ще однією перевагою є гнучкість – модель легко адаптується під задачі класифікації, детекції та сегментації.

Недоліки RepVGG. Головним недоліком RepVGG є значне збільшення часу та обчислювальних ресурсів, необхідних для навчання, адже тренувальна версія містить набагато більше операцій та параметрів через гілки у кожному блоці. Окрім того, процес перепараметризації додає ще один етап до робочого процесу – модель не можна просто зберегти та запустити, її потрібно сконвертувати у deploy версію, що вимагає написання додаткових скриптів для трансформації ваг.

3. Методи оптимізації

Оптимізація штучних мереж – комплексний процес, спрямований на покращення здатності моделей до навчання, підвищення її точності та зменшення обчислювальної складності. В межах цієї статі відокремимо два основних вектора: алгоритмічні оптимізатори, та методи стиснення та прискорення.

3.1. Алгоритмічні оптимізатори

Алгоритмічні оптимізатори вирішають, як саме оновляться ваги моделі для мінімізації функції втрат під час навчання моделі.

3.1.1. Стохастичний градієнтний спуск

Стохастичний градієнтний спуск – базовий алгоритм, в якому ваги оновлюються у напрямку протилежному до градієнтів. У сучасній практиці часто використовується спуск з моментом, завдяки якому можна уникнути застрягання в локальних мінімумах чи сідлових точках. Він описується наступною формулою:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta J(\theta_t) \\ \theta_{t+1} &= \theta_t - v_t \end{aligned} \tag{4}$$

Тут η – швидкість навчання, γ – коефіцієнт моменту, v_i – поточна швидкість.

3.1.2. Пропагація середнього квадратичного

Пропагація середнього квадратичного – Адаптивний метод, розроблений Джефрі Хінтоном. Він адаптує швидкість навчання для кожного градієнта окремо, Ділячи градієнт на рухоме середнє його квадратів. Математично він має наступний вигляд:

$$\begin{aligned} E[g^2]_t &= \beta E[g^2]_{t-1} + (1-\beta) g_t^2 \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \end{aligned} \tag{5}$$

Тут $E[g^2]_t$ – експоненціально згладжений квадрат градієнта, g_t – градієнт на поточному кроці, β – параметр згладжування.

3.1.3. Адаптивна оцінка моментів

Адаптивна оцінка моментів (надалі adam) – найпопулярніший оптимізатор на сьогоднішній день, бо він фактично поєднює в собі ідеї моменту та адаптивності. Він обчислює два моменти – експоненціальне рухоме середнє градієнтів m_i (аналог швидкості) та експоненціальне рухоме середнє квадратів градієнтів v_i (аналог нецентрованої дисперсії). Adam виглядає так:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \end{aligned} \tag{6}$$

Тут β_1, β_2 – коефіцієнти затухання першого та другого моментів, а $\hat{m}_i, \hat{v}_i$ – скориговані оцінки моментів, що компенсують їхнє початкове наближення до нуля.

3.2. Методи стиснення та прискорення

Методи стиснення та прискорення – методи, чия мета полягає в максимальному зменшенні кількості вхідних параметрів та обчислювальний операцій, зберігаючи при цьому точність, близьку до оригінальної моделі.

3.2.1. Квантування

Квантування нейронних мереж – процес, під час якого знижують розрядність чисел, що представляють параметри моделі. Наприклад, модель можна квантувати з 32-бітних чисел з плаваючою комою (надалі FP32) у восьмибітні цілі числа (надалі INT8).

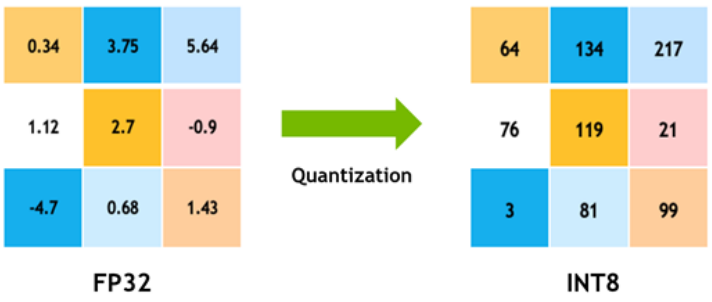

Рис. 6. Процес квантування

Основна мета цього процесу – суттєве зменшення обсягу пам'яті, необхідного для зберігання моделі, а також прискорення арифметичних операцій на апаратному рівні, оскільки цілочисельна арифметика виконується швидше від операцій з плаваючою точкою. Просте округлення дробових чисел до найближчого цілого є вкрай неефективним, бо призводить до катастрофічної втрати даних, особливо коли значення ваг зосереджені у дуже вузькому діапазоні поблизу нуля. Замість цього використовується лінійне квантування, яке проектує діапазон дійсних чисел на дискретну сітку за допомогою спеціального масштабуючого коефіцієнта та точки нуля. Формула деквантизації має наступний вигляд:

$$r = S(q - Z) \tag{7}$$

Тут r – дійсне число у FP32, S – масштабуючий коефіцієнт, q – квантоване число в INT8, а Z – точка нуля, яка вказує, яке саме число вважати нулем. Квантування має наступний вигляд:

$$q = round(\frac{r}{S} + Z) \tag{8}$$

Масштабування S – це різниця між максимумом та мінімумом дійсних чисел, поділена на кількість доступних рівнів у цілочисельному форматі (255 для INT8)

$$S = \frac{r_{max} - r_{min}}{q_{max} - q_{min}} \tag{9}$$

А точка нуля, яка зміщує діапазон обчислюється так:

$$Z = round(q_{min} - \frac{r_{min}}{S}) \tag{10}$$

Цей метод є надзвичайно швидким і чудово працює для статичних ваг мережі, проте для динамічних активацій він вразливий до поодиноких викидів. Залежно від етапу, на якому обчислюються ці параметри квантування, процес поділяється на дві великі категорії, першою з яких є квантування після навчання – через повністю натреновану мережу пропускають невеликий набір репрезентативних даних, щоб зібрати характеристику про діапазон активацій на кожному шарі та заморозити їх у цілочисельному форматі. Головна перевага такого підходу полягає в простоті та швидкості впровадження – немає необхідності перенавчати модель. Проте, застосування такого підходу для компактних архітектур призводить до значного падіння точності через накопичення помилок округлення на кожному шарі. Щоб вирішити цю проблему, застосовується складніший підхід – тренування з врахуванням квантизації. У цьому випадку процес квантування симулюється безпосередньо під час навчання моделі – під час прямого проходу ваги фальшиво квантуються до INT8, а потім одразу деквантуютсья назад, щоб мережа змогла підлаштувати свої параметри для компенсації втрат точності. Оскільки операція заокруглення є розривною і не має визначеної похідної, для зворотного поширення помилки використовується спеціальна апроксимація – градієнт функції втрат щодо квантованих ваг просто прирівнюється до градієнта щодо оригінальних дійсних ваг. Завдяки такому тренуванню мережа стає стійкою до квантування та здатна досягати вищої точності в порівнянні з квантуванням після тренування.

3.2.2. Обрізка

Обрізка нейронних мереж – метод архітектурної оптимізації, мета якого полягає у зменшенні розміру моделі та прискоренню часу прямого проходу за рахунок

видалення її надлишкових параметрів. Оскільки сучасні згорткові мережі містять значно більше ваг, ніж необхідно для визначення закономірностей у даних, обрізка дає змогу вилучити ту частину нейронів чи фільтрів, яка має найменший вплив на результат.

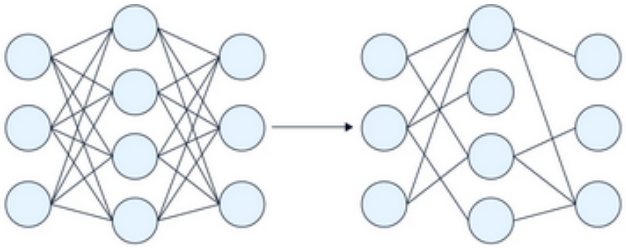

Рис. 7. Процес обрізки

Цей процес традиційно складається з трьох послідовних етапів – спочатку навчається базова перепараметризована модель до досягнення своєї максимальної збіжності, після застосовується алгоритм оцінки важливості для відсікання найменш значущих компонентів, а потім виконується етап донавчання для відновлення точності, яка неминуче просідає одразу після видалення ваг. Зазвичай застосовуються детерміновані критерії оцінки важливості, найпоширенішим з яких є критерій магнітуди, який базується на припущенні, згідно з яким ваги з найменшими абсолютними значеннями генерують найслабші сигнали при множенні на вхідні активації, а отже, роблять мінімальний внесок у загальну суму на виході нейрона, тобто вага ранжується виключно за своїм модулем $\text{score}(w) = |w|$, де всі параметри w, оцінка яких падає нижче за розрахований поріг, обнуляються. Залежно від способу застосування цього порогу обрізка буває структурованою або неструктурованою. В неструктурованій обрізці видаляються окремі зв'язки в матриці ваг, а структурована обрізка оперує цілими рядками, стовпцями чи каналами згортки. Варто відзначити, що неструктурована обрізка створює розріджені матриці, які можуть оброблятися повільніше.

3.2.3. Дистиляція знань

Дистиляція знань – метод компресії, який дає змогу передати приховану інформацію від масивної та високоточної моделі-вчителя до компактнішої моделі-учня. Дистиляція знань вирішує проблему жорстких міток – коли під час тренування модель правильний клас має ймовірність 1, а неправильний – 0. Натомість, під час тренування учня вчитель генерує м'які мітки, які показують ймовірність подібності об'єкта на зображенні на той чи інший клас. Завдяки інформації про міжкласову кореляцію учень має змогу формувати значно кращі внутрішні репрезентації візуальних ознак. Для того, щоб витягнути ці знання, в алгоритмі дистиляції використовується модифікована функція активації Softmax з додатковим параметром температури:

$$q_i = \frac{\exp(\frac{z_i}{T})}{\sum_j \exp(\frac{z_j}{T})} \tag{11}$$

Тут q_i – м'яка ймовірність, z_i – вихідні логіти мережі, T – температура. При $T = 1$ отримуємо стандартний Softmax, проте з ростом температури розподіл стає більш

пологим, підсилюючи значення навіть найменш імовірних класів, що робить їхні градієнти видимими для учня під час навчання. Загальна функція втрат для моделі-учня формується як зважена сума двох компонентів: традиційної перехресної ентропії між прогнозом учня та справжньою жорсткою міткою з датасету, а також дивергенції Кульбака-Лейблера між згладженими м'якими мітками вчителя та учня при однаковій підвищеній температурі.

$$L = \alpha L_{CE}(y, p) + (1 - \alpha) T^2 L_{KL}(q_{teacher}, q_{student}) \tag{12}$$

Тут α – коефіцієнт балансування між двома цілями, y – справжні мітки, p – звичайний прогноз учня($T = 1$), а T^2 необхідний для масштабування градієнтів від м'яких міток, щоб вони не губилися на фоні градієнтів від жорстких міток. Таким чином, компактний учень навчається імітувати не лише кінцеві рішення вчителя, але й саму логіку його мислення, досягаючи значно вищої точності розпізнавання, ніж якби він навчався на оригінальних даних самостійно з нуля.

4. Тренування та оптимізація

На початковому етапі як основу було обрано набір даних COCO (англ. Common Objects in Context – загальні об'єкти в контексті), який пройшов цільову попередню обробку – для зменшення розмірності вихідного простору та зниження загального обчислювального навантаження на мережу споріднені об'єкти, такі як легкові автомобілі, автобуси тощо, були об'єднані в єдиний клас транспортних засобів. На цьому датасеті спершу було натреновано надмірно параметризовану модель-вчителя на базі архітектури ShuffleNet із вхідною роздільною здатністю 416x416 пікселів. Велика кількість пікселів використовувалась для того, щоб вчитель зміг сформувати складні та деталізовані репрезентації візуальних ознак. Наступним кроком є процедура дистиляції знань – досвід вчителя дистилювався в учня на архітектурі RepVGG з меншим розширенням 320x320, задля підвищення швидкості роботи. Для максимального ущільнення репрезентацій до моделі-учня було додатково застосовано алгоритм структурної обрізки. З огляду на необхідність розгортання системи на пристроях із жорстко обмеженими обчислювальними ресурсами, підготовлені моделі були портовані у середовище NCNN. Фінальним етапом стало квантування моделей до цілочисельного формату INT8. Для забезпечення мінімальної деградації точності під час переходу від арифметики з плаваючою комою використовувалася репрезентативна підмножина затверджувальної вибірки, завдяки якій було згенеровано калібраційні таблиці. Завдяки цій таблиці NCNN мав змогу знаходити такі пороги відсікання, при яких втрата інформації мінімальна.

5. Аналіз результатів

Розглянемо роботу двох моделей – вчителя та учень. Для цього було написано використаємо алгоритм аналіху швидкості прямого проходу NCNN моделей.

Обидві моделі працюють. Учень навіть точніше розпізнав трамвай, який схожий на транспорт, хоча в наборі даних не було жодних трамваїв. Розглянемо тепер точність NCNN моделей на затверджувальній вибірці COCO. Результат наведений у табл. 1:

Можемо спостерігати, що точність учня вища за точність квантованого вчителя, та не сильно менша за точність початкової моделі-вчителя, натренованої в pytorch з точністю FP16. Варто відзначити, що учень може губити дрібні об'єкти

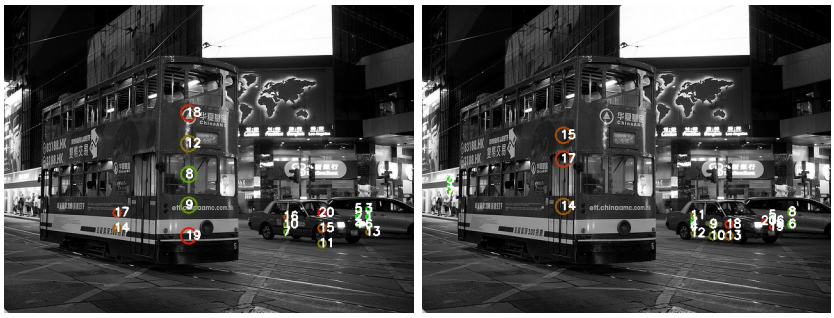

Рис. 8. Результати прогнозування двох моделей: учень (зліва) та вчитель (справа)

Таблиця 1

Точність NCNN моделей на вибірці COCO

Метрика	**IoU**	**Площа**	**Вчитель**	**Вчитель NCNN**	**Учень**
AP	0.50:0.95	all	0.365	0.296	**0.305**
AP	0.50	all	0.588	0.461	**0.463**
AP	0.75	all	0.373	0.313	**0.324**
AP	0.50:0.95	small	0.182	**0.154**	0.147
AR	0.50:0.95	all	0.491	**0.480**	0.465
AR	0.50:0.95	small	0.292	**0.279**	0.241
AR	0.50:0.95	medium	0.649	0.640	**0.648**
AR	0.50:0.95	large	0.793	0.781	**0.797**

(area=small на таблиці), бо його розширення значно менше. Водночас, він дуже добре справляється з великими об'єктами. Далі, Перевіримо час виконання всіх моделей в однакових умовах:

Таблиця 2

Порівняння праці учня та вчителя

Модель	**Середній час**	**Максимальний час**	**Мінімальний час**
Вчитель	148 мс	284мс	126 мс
Вчитель(NCNN)	**51мс**	259 мс	**45 мс**
Учень	64 мс	**90 мс**	62 мс

На таблиці видно, що учень працює не найшвидше, але стабільніше за інші моделі. Стабільна робота є вкрай важливим аспектом використання моделей в системах прийняття рішення в режимі реального часу.

6. Висновки

Отже, було проведено дослідження різних методів оптимізації ЗНМ та натреновано мережу, яка отримала незначні (6%) втрати точності, але працює значно

швидше (>2x) відносно початкового варіанту. Така модель має потенціал для розгорнення на різноманітних обчислювально-обмежених пристроях, як-от телефони, IP-камери, тощо.

Список використаної літератури

1. Renu K. Convolutional Neural Network: Feature Map and Filter Visualization / Renu. – K., 2018.
2. Xiaohan D. RepVGG: Making VGG-style ConvNets Great Again / D. Xiaohan, Z Xiangyu. – 2021.
3. Xiangou Z. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices / Z. Xiangyu, Z. Xinyu. – 2017.
4. Markus N. A White Paper on Neural Network Quantization / N. Markus, F. Marios. – 2021.

Стаття: надійшла до редколегії 14.02.2026
доопрацьована 04.03.2026
прийнята до друку 03.03.2026

OPTIMIZATION OF COMPUTER VISION MODELS FOR COMPUTATIONALLY LIMITED DEVICES

A. Hrabovetskyi, M. Baranov

Ivan Franko National University of Lviv,
1, Universytetska str., 79000, Lviv, Ukraine,
e-mail: andrii.hrabovetskyi@lnu.edu.ua, mykola.baranov@lnu.edu.ua

Given the rapid development of the edge computing concept, the deployment of complex computer vision models on hardware-constrained devices has become a critical task. This paper examines methods for optimizing convolutional neural networks to enhance object recognition efficiency under limited computational resources. Existing model compression approaches are analyzed, identifying their advantages and disadvantages regarding memory usage and performance in resource-constrained environments. Special attention is paid to addressing the issue of processing time instability, which frequently occurs when deploying neural networks on low-power devices. The study is based on an analysis of the ShuffleNet architecture and the development of an improved solution leveraging the RepVGG model. Research indicates that while ShuffleNet's channel shuffling method reduces computational complexity, it faces practical data reallocation challenges. The knowledge distillation application combined with quantization procedures enabled the construction of a model with significantly lower execution time variance and a predictable system speed. Testing results confirm that the resulting model outperforms quantized ShuffleNet in accuracy while demonstrating higher stability and more rational use of hardware capacities. This approach allows for a substantial reduction in computational costs without critical losses in recognition quality, which is essential for real-time systems.

Key words: convolutional neural network, optimization, knowledge distilation, pruning, quantization.