*УДК* 519.6 *http://dx.doi.org/10.30970/vam.2026.36.14044*

# ОПТИМІЗАЦІЯ МОДЕЛЕЙ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ ОБЧИСЛЮВАЛЬНО ОБМЕЖЕНИХ ПРИСТРОЇВ

**А. Грабовецький, М. Баранов**

*Львівський національний університет імені Івана Франка,*
*вул. Університетська 1, Львів, 79007, Україна,*
*e-mail:* *andrii.hrabovetskyi@lnu.edu.ua*, *mykola.baranov@lnu.edu.ua*

З огляду на стрімкий розвиток концепції периферійних обчислень, критично важливим завданням стає розгортання складних моделей машинного зору на пристроях з обмеженими апаратними можливостями. 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 опрацювання даних, яка часто виникає під час розгортання нейронних мереж на пристроях з малою потужністю. Основою дослідження став аналіз архітектури ShuffleNet та розроблення вдосконаленого рішення на базі моделі RepVGG. Дослідження продемонструвало, 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 вищу стабільність показників і раціональніше використання апаратних потужностей. Такий підхід допомагає суттєво знизити обчислювальні витрати без критичних втрат якості розпізнавання.

*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 постає проблема практичної реалізації. 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 стандартних багатошарових перцептронів, ЗНМ оперують даними у вигляді багатовимірних тензорів, завдяки чому є можливість зберігати й обробляти просторову

структуру зображень у їхньому природньому вигляді, без необхідності попереднього перетворення в 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. Той самий фільтр застосовують до всіх частин зображення. Цей фільтр називається ядром згортки. Завдяки ньому кількість параметрів мережі порівняно з повнозв'язними шарами (де кожен нейрон має власний набір ваг для усіх пікселів), зменшується кардинально. Операції підвибірки з використанням ядра дають змогу зменшувати розмірність ознакових карт, завдяки чому модель стає інваріантною до невеликих зсувів або спотворень на зображенні.

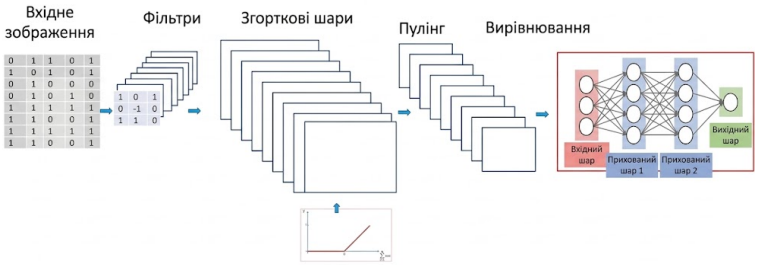


Рис. 1. Архітектура ЗНМ

На рис. 1 можна побачити узагальнену архітектуру ЗНМ. Далі буде наведено розбір двох моделей – ShuffleNet [3] та RepVGG [2].

## 2.1. ShuffleNet

ShuffleNet [3] – ефективна архітектура ЗНМ, розроблена спеціально для обчислювально обмежених пристроїв. Основна мета створення цієї моделі – мінімізація кількості операцій з плаваючою комою при збережені високої точності. Головна ідея полягає у вирішенні проблеми високої обчислювальної вартості стандартних 1x1 згорток, які у багатьох сучасних архітектурах споживають значну частку ресурсів. Для подолання цього бар'єра ShuffleNet використовує два важливі механізми – групові точкові згортки та операцію перемішування каналів.

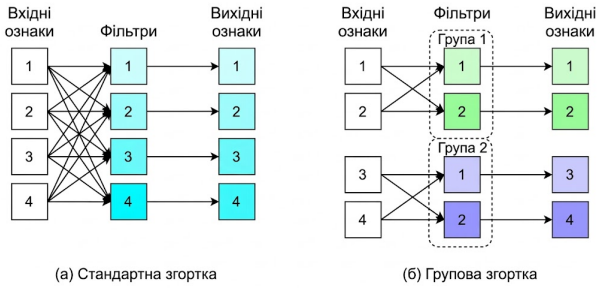


Рис. 2. Різниця між звичайною згорткою (а) та груповою (б)

У звичайних мережах класичні операції згортки більш затратні, порівняно груповими згортками, бо вони з'єднують кожен вхідний канал з кожним вихідним. Групова реалізація розбиває ці канали на кілька ізольованих підмножин. Кожна згортка працює лише у межах своєї групи. Отже, обчислювальна складність зменшується пропорційно до 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.

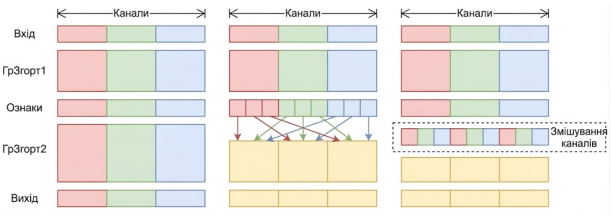


Рис. 3. Перемішування каналів. ГрЗгорт – групова згортка [3]

Без цього перемішування мережа фактично розкладається на кілька вузьких мереж, які не знають про існування ознак одна одної. Перемішування діє як детермінована перестановка: на вхід подаються вихідні канали від кожної групи, розділяються та перерозподіляються так, що наступний шар групових згорток отримує дані від кожної групи попереднього шару. Завдяки цьому між групами відбувається повноцінний груповий обмін, і зберігається низька обчислювальна вартість, притаманна груповим операціям. Головним компонентом архітектури є ShuffleNet Unit – спеціалізований залишковий блок, побудований за принципом пляшкового горла.

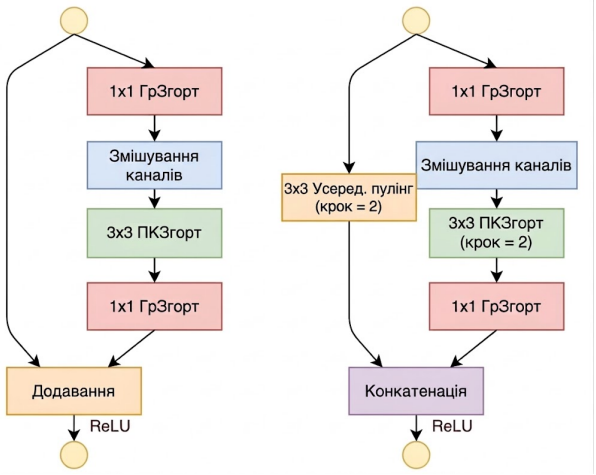


Рис. 4. ShuffleNet Unit. ГрЗгорт – групова згортка, ПКЗгорт – поканальна згортка [3]

У базовому варіанті цей блок розпочинається 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), то до структури додається середня підвибірка на шляху швидкого з'єднання, а замість звичайного додавання ознак використовується їхня конкатенація, завдяки якій можна ефективніше збільшити ширину каналів.

**Переваги ShuffleNet**. Головна перевага ShuffleNet – її безпрецедентна здатність до масштабування в умовах обмежених 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 нових параметрів, завдяки чому ShuffleNet не займатиме багато місця у пам'яті. Крім того, моделі на цій 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 з лінійним доступом до пам'яті. Також варто зазначити, що при екстремальному 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 [2] – архітектура ЗНМ, яка пропонує унікальний компроміс для вирішення дилеми складності та швидкості. Вона була розроблена, щоб повернути популярність простим архітектурам, подібним до оригінальної VGG, але з потужністю сучасних залишкових мереж. Основна ідея RepVGG полягає в концепції структурного перевизначення параметрів, завдяки якому архітектура отримує переваги складних багатогілкових структур під час навчання та високу швидкість моделі під час прямого проходу. Реалізація RepVGG ґрунтується на математичній лінійності згорткових операцій, завдяки якій можна трансформувати складну топологію навчання у максимально спрощену структуру прямого проходу.

Під час фази тренування кожен блок складається з трьох паралельних гілок – основної згортки 3x3, допоміжної згортки 1x1 та ідентичного зв'язку, кожна з цих гілок супроводжується шаром пакетної нормалізації. Завдяки такій надмірності створюється складний ландшафт втрат, який полегшує збіжність моделі та дозволяє досягти високої точності. Для фінального використання модель проходить етап структурованого перевизначення параметрів, під час якого модифікуються ваги ядра згортки й обчислюється новий вектор зміщення. Так мінімізуються обчислювальні витрати на нормалізацію під час роботи. Математично ці операції виглядають

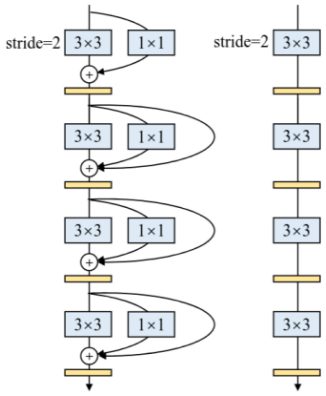


Рис. 5. Перевизначення параметрів RepVGG

так: нехай $x$ – результат згортки. Тоді операція пакетної нормалізації описується формулою

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

де $\gamma$ – параметр масштабування; $\beta$ – параметр зсуву що вивчається; $\mu$ – середнє значення активацій, накопичене під час тренування; $\sigma^2$ – дисперсія активацій, накопичена під час тренування; $\epsilon$ – 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 такі формули:

$$\hat{W} = W \cdot \frac{\gamma}{\sqrt{\sigma^2 + \epsilon}}$$

$$\hat{b} = \frac{(b - \mu) \cdot \gamma}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

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

$$W_{final} = \hat{W}_{\mathrm{3x3}} + \hat{W}_{\mathrm{1x1\_to\_3x3}} + \hat{W}_{\mathrm{id\_to\_3x3}}$$

$$b_{final} = \hat{b}_{\mathrm{3x3}} + \hat{b}_{\mathrm{1x1}} + \hat{b}_{\mathrm{id}}$$

У підсумку 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 перевизначення параметрів модель перетворюється на просту послідовність згорток 3x3, і отже, 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, порівняно з іншими архітектурами аналогічної точності. Крім того, відсутність допоміжних гілок під час роботи суттєво знижує споживання оперативної пам'яті, оскільки

системі не потрібно тримати результати кількох гілок для їх подальшого підсумовування. Ще однією перевагою є гнучкість – модель легко адаптується під задачі класифікації, детекції та сегментації.

**Недоліки RepVGG**. Головний недолік RepVGG – значне збільшення часу та обчислювальних ресурсів, необхідних для навчання, адже тренувальна версія містить набагато більше операцій і параметрів через гілки у кожному блоці. Окрім того, процес перевизначення параметрів додає ще один етап до робочого процесу – модель не можна просто зберегти та запустити, її потрібно перетворити у виробничу версію, що потребує написання додаткових скриптів для трансформації ваг.

## 3. Методи оптимізації

Оптимізація штучних мереж – комплексний процес, спрямований на поліпшення здатності моделей до навчання, підвищення точності та зменшення обчислювальної складності. У межах цієї статті відокремимо два основних вектори: алгоритмічні оптимізатори, методи стиснення та прискорення.

### 3.1. Алгоритмічні оптимізатори

Алгоритмічні оптимізатори визначають правило оновлення ваг моделі для мінімізації функції втрат під час навчання.

#### 3.1.1. Стохастичний градієнтний спуск

Стохастичний градієнтний спуск [5] – базовий алгоритм, у якому ваги оновлюються у напрямі, протилежному до градієнтів. На практиці часто застосовується версія з моментом, що дозволяє уникати локальних мінімумів і сідлових точок:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta_t),$$
$$\theta_{t+1} = \theta_t - v_t,$$

де $\eta$ – швидкість навчання; $\gamma$ – коефіцієнт моменту; $v_t$ – поточна швидкість.

#### 3.1.2. Поширення середнього квадратичного

Метод поширення середнього квадратичного [6], запропонований Джефрі Хінтоном, адаптує швидкість навчання для кожного параметра, ділячи градієнт на рухоме середнє його квадратів:

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta) g_t^2,$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}}\, g_t,$$

де $E[g^2]_t$ – експоненціально згладжений квадрат градієнта; $g_t$ – градієнт на поточному кроці; $\beta$ – параметр згладжування.

#### 3.1.3. Адаптивна оцінка моментів

Адаптивна оцінка моментів (Adam) [7] – найпопулярніший сучасний оптимізатор, що поєднує ідеї моменту та адаптивності. Він обчислює два рухомих моменти:

середнє градієнтів $m_t$ та середнє квадратів градієнтів $v_t$:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t,$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t},$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t,$$

де $\beta_1, \beta_2$ – коефіцієнти згасання першого та другого моментів; $\hat{m}_t, \hat{v}_t$ – скориговані оцінки моментів.

## 3.2. Методи стиснення та прискорення

Методи стиснення та прискорення спрямовані на максимальне скорочення кількості параметрів і обчислювальних операцій зі збереженням точності, близької до оригінальної моделі.

### 3.2.1. Квантування

Квантування нейронних мереж – процес зниження розрядності числового представлення параметрів моделі [8]. Наприклад, перехід від 32-бітних чисел з плаваючою комою (FP32) до восьмибітних цілих чисел (INT8) суттєво зменшує обсяг пам'яті та прискорює арифметичні операції.

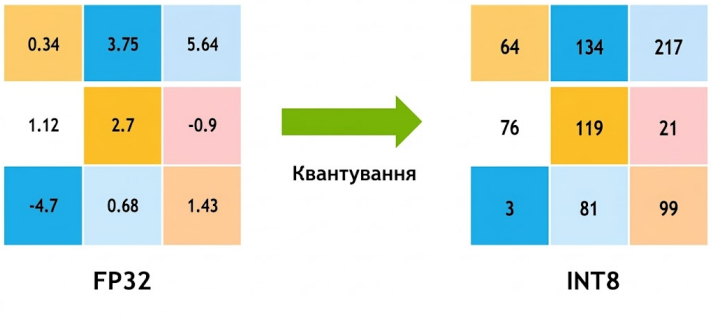


Рис. 6. Процес квантування

Просте заокруглення неефективне через катастрофічну втрату даних у вузьких діапазонах ваг. Тому використовується лінійне квантування: діапазон дійсних чисел проєктується на дискретну сітку за допомогою масштабуючого коефіцієнта $S$ та точки нуля $Z$. Формула деквантизації має такий вигляд:

$$r = S(q - Z),$$

де $r$ – дійсне число у FP32; $q$ – квантоване число в INT8. Квантування має такий вигляд:

$$q = \text{round}\left(\frac{r}{S} + Z\right).$$

Масштабуючий коефіцієнт обчислюється так:

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

Точка нуля має такий вигляд:

$$Z = \text{round}\left(q_{\min} - \frac{r_{\min}}{S}\right).$$

Залежно від етапу обчислення параметрів квантування розрізняють два підходи. *Квантування після навчання*: через натреновану мережу пропускають репрезентативні дані, збирають характеристики діапазонів активацій на кожному шарі та заморожують їх у цілочисельному форматі. Підхід простий у реалізації, проте для компактних архітектур може призводити до суттєвого падіння точності через накопичення помилок заокруглення. *Тренування з урахуванням квантизації*: процес квантування симулюється безпосередньо під час навчання – ваги псевдоквантуються до INT8 і негайно деквантуються, що дає змогу мережі компенсувати втрати точності. Оскільки операція заокруглення не має визначеної похідної, для зворотного поширення помилки градієнт функції втрат щодо квантованих ваг прирівнюється до градієнта щодо оригінальних ваг. Такий підхід забезпечує вищу точність порівняно з квантуванням після навчання.

### 3.2.2. Обрізка

Обрізка нейронних мереж [8] – метод архітектурної оптимізації, мета якого полягає у зменшенні розміру моделі та прискоренні часу прямого проходу завдяки видаленню її надлишкових параметрів. Оскільки сучасні згорткові мережі містять значно більше ваг ніж потрібно для визначення закономірностей у даних, то обрізка дозволяє вилучити ту частину нейронів чи фільтрів, яка має найменший вплив на результат.

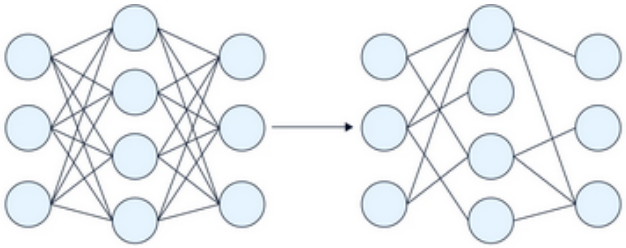

Рис. 7. Процес обрізки

Цей процес традиційно складається з трьох послідовних етапів – спочатку навчається базова перепараметризована модель до досягнення своєї максимальної збіжності, після чого 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 застосовують детерміновані критерії оцінки важливості, найпоширенішим з яких є критерій магнітуди: він 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. Вага ранжується тільки за своїм модулем $\text{score}(w) = |w|$, де всі параметри $w$, оцінка яких падає нижче за розрахований поріг, обнуляються. Залежно від способу застосування цього порога 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 зазначити, що неструктурована обрізка створює розріджені матриці, які можуть оброблятися повільніше.

### 3.2.3. Дистиляція знань

Дистиляція знань [9] – метод компресії, який дозволяє передати приховану інформацію від масивної та високоточної моделі-вчителя до компактнішої моделі-учня. Дистиляція знань вирішує проблему жорстких міток – коли під час тренування правильний клас має ймовірність 1, а неправильний – 0. Натомість, під час тренування учня вчитель генерує м'які мітки, які демонструють ймовірність подібності об'єкта на зображенні на той чи інший клас. Завдяки інформації про міжкласову кореляцію учень має змогу формувати значно ліпші внутрішні репрезентації візуальних ознак. Для того, щоб витягнути ці знання, в алгоритмі дистиляції використовується модифікована функція активації Softmax з додатковим параметром температури

$$q_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)},$$

де $q_i$ – м'яка ймовірність; $z_i$ – вихідні логіти мережі; $T$ – температура. При $T = 1$ отримуємо стандартний Softmax, проте зі збільшенням 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 набору даних, а також дивергенції Кульбака-Лейблера між згладженими м'якими мітками вчителя та учня за однинакової підвищеної температури

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

Тут $\alpha$ – 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. Отже, компактний учень навчається імітувати не лише кінцеві рішення вчителя, а 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) [10], 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 наборі даних спершу було натреновано надмірно параметризовану модель-вчителя на базі архітектури ShuffleNet із вхідною роздільною здатністю 416×416 пікселів. Велику кількість пікселів використовували для того, щоб вчитель зміг сформувати складні та деталізовані репрезентації візуальних ознак. Наступний крок – процедура дистиляції знань – досвід вчителя дистилювався в учня на архітектурі RepVGG з меншим розширенням 320×320, 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, за яких втрата інформації мінімальна.

## 5. Аналіз результатів

Розглянемо роботу двох моделей – вчителя та учня. Для цього використаємо алгоритм аналізу швидкості прямого проходу NCNN моделей.

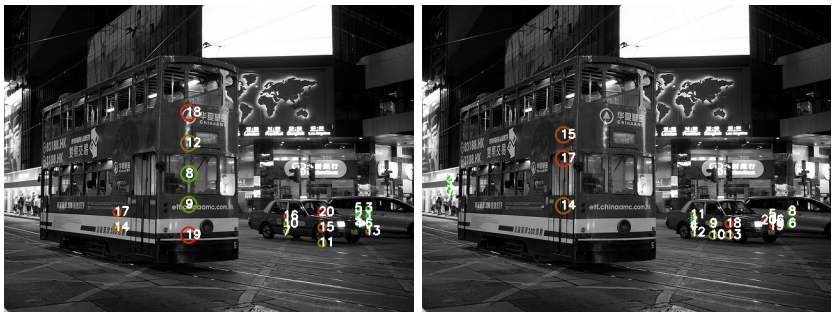

Рис. 8. Результати прогнозування двох моделей: учень (зліва) і вчитель (справа)

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 наведено у табл. 1.

У табл. 1 можемо спостерігати, що точність учня вища за точність квантованого вчителя та не надто менша за точність початкової моделі-вчителя, натренованої в PyTorch з точністю FP16. Варто зазначити, що учень може губити дрібні об'єкти (*area=small* в табл. 1), бо його розширення значно менше. Водночас він дуже добре справляється з великими об'єктами. Далі перевіримо час виконання всіх моделей в однакових умовах.

З табл. 2 видно, що учень працює не найшвидше, але стабільніше за інші моделі. Стабільна робота є вкрай важливим аспектом використання моделей у системах прийняття рішень у режимі реального часу.

*Таблиця* 1

Точність моделей при тестуванні затверджувальною вибіркою COCO.
AR – середня точність, AP – середня повнота

| **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** |

*Таблиця* 2

Час прямого проходу різних моделей

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

## 6. Висновки

Отже, було проведено дослідження різних методів оптимізації ЗНМ та натреновано мережу, яка отримала незначні (6 %) втрати точності, але працює значно швидше ($>2$x), порівняно з початковим варіантом. Така модель має потенціал для розгортання на різноманітних обчислювально-обмежених пристроях, як-от телефони, IP-камери тощо. Моделі вчителя та учня доступні за посиланням: https://github.com/domiNo39/models

## 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.
5. Che E. Stochastic gradient descent with adaptive data / E. Che, J. Dong, X.T. Tong // Operations Research. – 2026. – URL: https://arxiv.org/abs/2410.01195.
6. Li R. Stable gradient-adjusted root mean square propagation on least squares problem / R. Li, J. Xu, W. Xing. – 2024. – URL: https://arxiv.org/abs/2411.15877.
7. Kingma D.P. Adam: A method for stochastic optimization / D.P. Kingma, J. Ba. – 2014. – URL: https://arxiv.org/abs/1412.6980.
8. Han S. Learning both weights and connections for efficient neural network / S. Han, J. Pool, J. Tran, W. Dally // Advances in neural information processing systems. – 2015. –

Vol. 28. – URL: https://proceedings.neurips.cc/paper/2015/hash/ae39100063050e1075677-b6131d2557e-Abstract.html.

9. Hinton G. Distilling the knowledge in a neural network / G. Hinton, O. Vinyals, J. Dean // arXiv preprint. – 2015. – URL: https://arxiv.org/abs/1503.02531.
10. Lin T.-Y. Microsoft coco: Common objects in context / T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick // European conference on computer vision. – 2014. – P. 740–755. URL: https://arxiv.org/abs/1405.0312.

*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., 79007, 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.