

DOI: <https://doi.org/10.15276/ict.03.2026.58>

УДК 004.85:004.75:621.39

Оцінювання та оптимізація моделей машинного навчання для edge-обробки інформації мобільної мережі

Ревенко Петро Георгійович

ORCID: <https://orcid.org/0009-0003-7243-7826>; petroniy13@stud.op.edu.ua

Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна

АНОТАЦІЯ

У статті розглянуто задачу оцінювання та оптимізації моделей машинного і глибокого навчання для виконання на edge-вузлах під час обробки інформації мобільної мережі. Актуальність дослідження зумовлена тим, що моделі з близькою прогноною якістю можуть істотно відрізнятися за затримкою інференсу, розміром артефакту та вимогами до обчислювальних ресурсів, тому вибір лише за показником точності не гарантує придатності до edge-виконання. **Метою роботи** є формування узгодженого банку deployment-конфігурацій моделей машинного навчання на основі спільного оцінювання прогносної якості та характеристик виконання до і після оптимізації. Вхідним інформаційним об'єктом є інтегрований вектор стану мобільної мережі сталої структури, що містить статистичні, часові, подієві, ресурсні та якісні характеристики. Для однієї задачі класифікації порівнюються XGBoost, MLP, GRU, TCN та Transformer-подібна модель. Для нейронних моделей сформовано оптимізовані варіанти із застосуванням динамічного INT8-квантування, структурного pruning та knowledge distillation. Кожна конфігурація характеризується macro-F1, balanced accuracy, recall критичного класу, медіанною та p95-затримкою інференсу і розміром артефакту. Після quality-gate виконується Pareto-відбір конфігурацій, що не поступаються іншим одночасно за якістю, затримкою та розміром. **Експериментальне дослідження** проведено на синтетичній послідовності з 17 995 часових вікон і чотирма класами стану мережі. Найвищий macro-F1 серед початкових моделей отримано для GRU (0,918) і TCN (0,918). Структурне скорочення TCN зменшило розмір артефакту на 23,0 % і медіанну затримку приблизно на 10 % без погіршення macro-F1. Distillation Transformer-моделі скоротила розмір на 73,4 % і затримку майже вдвічі. Водночас INT8-квантування MLP і GRU суттєво зменшило розмір, але на контрольному CPU-профілі збільшило latency через накладні витрати runtime. **Результати показують**, що edge-оптимізацію доцільно оцінювати не за одним показником, а за сукупністю predictive quality і характеристик виконання. Сформований банк deployment-конфігурацій може використовуватися як вхідний набір кандидатів для подальшого адаптивного вибору моделі та місця її виконання в edge/cloud-інфраструктурі мобільної мережі.

Ключові слова: мобільні мережі, машинне навчання, глибоке навчання, edge computing, оптимізація моделей, стиснення моделей, квантування, pruning, knowledge distillation

Для цитування: Ревенко П. Г. “Оцінювання та оптимізація моделей машинного навчання для edge-обробки інформації мобільної мережі”. *Інформатика. Культура. Техніка.* 2026; Том 3 № 2 (4): 756–764. DOI: <https://doi.org/10.15276/ict.03.2026.58>

Актуальність. Сучасні мобільні мережі поєднують радіодоступ, ядро мережі, edge/MEC-компоненти та хмарні ресурси, які формують значні обсяги телеметрії, показників навантаження, подій та ресурсних характеристик. Машинне навчання застосовується для прогнозування трафіку, виявлення аномалій, оцінювання стану компонентів і підтримки ресурсного керування [1]. Перенесення частини обробки до edge-рівня дозволяє зменшити мережеву затримку і обсяг передавання даних, але одночасно робить обчислювальні обмеження одним із факторів, що визначають практичну придатність ML-моделі [2, 3].

Для однієї і тієї самої задачі можуть використовуватися моделі різних сімейств: градієнтний бустинг, багатоваріантні перцептрони, рекурентні та згорткові часові мережі, attention- і Transformer-подібні архітектури. У мобільних мережах такі моделі вже застосовуються для прогнозування ресурсних потреб, трафіку та аномальних станів [4, 5]. Однак максимальна прогнозна якість не є достатнім критерієм для edge-виконання. Більша модель може мати прийнятну точність, але перевищувати допустиму затримку, вимагати більше пам'яті або неефективно виконуватися на конкретному runtime.

Це зумовлює необхідність розглядати модель не лише як навчений алгоритм, а як deployment-конфігурацію з вимірюваними характеристиками. Для практичного використання важливими стають predictive quality, медіанна та хвостова latency, розмір артефакту, кількість параметрів і сумісність із доступним середовищем виконання. Окремою задачею є оптимізація базової моделі за допомогою quantization, pruning, knowledge distillation або архітектурного спрощення з повторним контролем якості після перетворення.

Отже, актуальним є формування відтворюваної процедури, яка дозволяє для єдиного інформаційного входу порівняти різні ML/DL-моделі, отримати їх оптимізовані технічні варіанти та сформувати компактний банк deployment-конфігурацій. Такий банк є проміжним рівнем між підготовкою даних і подальшим адаптивним вибором конкретної моделі та обчислювального вузла.

Огляд наукових досліджень та публікацій. Проблема ефективного виконання нейронних мереж в умовах обмежених ресурсів досліджується у зв'язку зі зростанням обчислювальної складності сучасних моделей. У роботі Sze та співавторів систематизовано фактори, що визначають енергоефективність, пропускну здатність і latency DNN, та показано необхідність спільного врахування архітектури мережі, пам'яті й апаратного середовища [6]. Для мобільних платформ окремо досліджено вплив компресії на розмір моделі, обчислення, затримку та енергоспоживання [7].

Основні напрями стиснення моделей охоплюють зменшення кількості параметрів, скорочення точності їх представлення, структурне спрощення і перенесення знань до компактнішої моделі. Узагальнений огляд принципів model compression та acceleration наведено в [8]. Детальніше pruning і quantization розглянуто в [9], де показано, що ці методи можуть використовуватися окремо або спільно, однак їх ефект залежить від структури мережі та реалізації. INT8-квантування дозволяє суттєво скорочувати обсяг пам'яті та за наявності оптимізованих integer kernels покращувати співвідношення accuracy-latency [10].

Knowledge distillation формує компактну student-модель за рахунок використання інформації teacher-моделі і є окремим класом підходів до компресії [11]. Іншим напрямом є початкове проєктування легких архітектур. Наприклад, MobileNetV2 демонструє можливість цілеспрямовано оптимізувати структуру моделі за кількістю операцій, параметрів і реальною затримкою [12]. Порівняльні дослідження методів compression показують, що одна й та сама техніка не забезпечує універсально найкращого результату: вибір залежить від архітектури, типу операцій, середовища виконання та допустимої втрати якості [13].

У задачах обробки мережових часових даних практичний інтерес становлять різні архітектурні сімейства. TCN застосовуються для аналізу часових рядів і виявлення відхилень у стилізованих мережах [14], attention-моделі використовуються для багаторизонтного прогнозування [15], а XGBoost залишається сильним базовим алгоритмом для структурованих ознак [16]. Це робить доцільним не пошук єдиної «найкращої» архітектури, а формування множини моделей з різними характеристиками.

У попередніх дослідженнях було проаналізовано застосування нейронних архітектур і методів оптимізації для потокової мультимодальної обробки в розподілених системах із edge computing [17], а також сформовано інтегроване ознакове представлення інформації мобільної мережі [18]. У цій роботі ці результати не є обов'язковою передумовою: достатньо, щоб стан мережі подавався фіксованим вектором або причинною послідовністю таких векторів. Невирішеним залишається питання єдиного протоколу порівняння різних сімейств моделей, їх оптимізованих варіантів і відбору технічних конфігурацій, придатних для подальшого edge-виконання.

Метою дослідження є оцінювання моделей машинного та глибокого навчання для обробки інтегрованого стану мобільної мережі та формування набору edge-придатних deployment-конфігурацій шляхом порівняння predictive quality, затримки інференсу й обсягу пам'яті до і після оптимізації моделей.

Формування банку моделей машинного навчання. Вхідним об'єктом дослідження є інтегрований вектор стану мобільної мережі $s(n)$ сталої розмірності. Він формується для n -го часового вікна та містить мережеві, ресурсні, статистичні, часові, події й quality-характеристики. Для цієї статті спосіб побудови $s(n)$ не обмежується конкретним алгоритмом: принциповим є однаковий вхід для всіх порівнюваних моделей і відсутність майбутньої інформації відносно моменту n .

Множину кандидатних моделей для однієї ML-задачі позначимо M . До неї можуть входити як класичні методи, так і нейронні архітектури різних сімейств:

$$M = \{M_1, M_2, \dots, M_{N_M}\}. \quad (1)$$

У проведеному дослідженні банк містив п'ять базових кандидатів: XGBoost, MLP, GRU, TCN і Transformer. XGBoost та MLP працювали безпосередньо з поточним $s(n)$, тоді як GRU, TCN і Transformer використовували причинну послідовність із шести останніх станів.

Результат застосування k -ї моделі задається як:

$$z_k(n) = M_k(s(n); \theta_k). \quad (2)$$

Для sequence-моделей запис $s(n)$ у формулі (2) слід трактувати як послідовність $[s(n-5), \dots, s(n)]$. Всі моделі навчаються на одному хронологічному train-інтервалі, налаштовуються за validation-частиною і лише після фіксації конфігурації оцінюються на test-частині. Це дозволяє відокремити властивості конкретної архітектури від впливу різних правил поділу даних. Архітектури та базові параметри моделей, використаних в експерименті, наведено в Таблиці 1.

Таблиця 1. Базовий банк моделей та їх конфігурації

Модель	Вхід	Основна конфігурація	Роль у порівнянні
XGBoost	$s(n)$	220 дерев; max depth = 5; learning rate = 0,05	Табличний baseline
MLP	$s(n)$	36-96-48-4; ReLU	Компактна нейромережева модель
GRU	$6 \times s(n)$	hidden size = 48	Рекурентна часова модель
TCN	$6 \times s(n)$	2 Conv1D; 48 каналів; dilation 1/2	Згорткова часова модель
Transformer	$6 \times s(n)$	d_model = 48; 2 шари; 4 heads	attention-модель

Джерело: розроблено автором

Оцінювання deployment-конфігурацій. Після навчання базова модель розглядається не лише як набір параметрів, а як технічна конфігурація, яку можна профілювати та оптимізувати. Для кожної конфігурації M_{kr} формується профіль, що містить quality і характеристики виконання:

$$p_{kr} = [q_{kr}^M, l_{kr}, mem_{kr}, size_{kr}]. \quad (3)$$

Тут q_{kr}^M – нормована прогнозна якість, у класифікаційній постановці представлена macro-F1; l_{kr} – медіанна затримка інференсу на контрольному профілі; mem_{kr} – вимога до оперативної пам'яті або її практичний проху; $size_{kr}$ – розмір збереженого артефакту моделі. Додатково фіксується p95 latency, оскільки середня або медіанна затримка не відображає хвостові затримки, критичні для SLA.

Базову, неоптимізовану версію моделі позначимо M_{k0} . Для оптимізованого варіанта оцінюється зміна predictive quality відносно цієї версії:

$$\Delta q_{kr} = q_{k0}^M - q_{kr}^M. \quad (4)$$

Конфігурація проходить quality-gate, якщо Δq_{kr} не перевищує наперед встановлену межу ε_q . У дослідженні використано $\varepsilon_q = 0,015$ за macro-F1. Це обмеження не означає, що оптимізована версія обов'язково повинна бути гіршою: після fine-tuning або distillation можливе і невелике покращення тестового показника:

$$M_{dep} = \{M_{kr} | \Delta q_{kr} \leq \varepsilon_q\}. \quad (5)$$

Після quality-gate серед допустимих варіантів виконується Pareto-відбір за трьома базовими координатами: macro-F1 максимізується, а latency та size мінімізуються. Конфігурація вилучається з deployment-банку, якщо існує інша, яка не гірша за всіма трьома показниками і краща хоча б за одним. Такий відбір є офлайн-процедурою і не визначає, яку

модель слід запускати в конкретний момент часу, він лише усуває очевидно доміновані технічні варіанти перед подальшим адаптивним вибором.

Оптимізація моделей для edge-виконання. Для різних архітектур застосовано різні способи оптимізації, оскільки одна техніка не однаково ефективна для всіх типів операцій. Для MLP і GRU використано динамічне INT8-квантування ваг лінійних та рекурентних шарів. Воно зменшує обсяг збереження та пам'яті для параметрів, але реальний виграш у latency залежить від наявності оптимізованих integer kernels у runtime [9], [10].

Для TCN застосовано структурний channel pruning. Після початкового навчання оцінювалася L1-значущість каналів, після чого кількість каналів у двох згорткових блоках зменшувалася з 48 до 40. Сформована компактна мережа донавчалася на train-частині, а її якість повторно оцінювалася без зміни test-інтервалу. На відміну від unstructured sparsity, структурне видалення каналів безпосередньо скорочує розмір наступних операторів і краще узгоджується з типовими dense kernels [9].

Для Transformer використано knowledge distillation. Двошарова teacher-модель з $d_model = 48$ передавала soft targets одношаровій student-моделі з $d_model = 32$ і меншим feed-forward блоком. Функція навчання student поєднувала крос-ентропію за цільовими класами та KL-divergence між розподілами teacher і student з температурним згладжуванням. Такий підхід дозволяє зменшити кількість шарів і параметрів, зберігаючи частину інформації, накопиченої більшою моделлю [11].

Для кількісної оцінки ефекту оптимізації відносно зміну довільної cost-характеристики у визначимо як:

$$r_y = 1 - \frac{y_{kr}}{y_{k0}}. \quad (6)$$

Для size і latency додатне r_y означає зменшення витрат. Водночас інтерпретація повинна враховувати характер конкретного runtime: наприклад, зменшення розміру INT8-моделі не гарантує прискорення для дуже малих мереж, якщо накладні витрати динамічного квантування більші за виграш від integer-обчислень.

Експериментальне дослідження. Для оцінювання моделей сформовано синтетичну послідовність станів мобільної мережі, що містила 18 000 часових кроків. Після формування причинних послідовностей довжини шість отримано 17 995 ML-прикладів. Вектор $s(n)$ мав 36 координат і поєднував мережеві KPI, показники навантаження та ресурсів, часові зміни і характеристики якості. Цільова змінна містила чотири класи: нормальний стан, перевантаження, деградація та критичний стан. Частки класів становили приблизно 71,3 %, 14,8 %, 10,0 % та 3,9 % відповідно.

Дані формувалися послідовно – стан у наступному вікні залежав від попереднього, а значення ознак мали інерційну складову, клас-залежні зсуви, шум і періодичні компоненти. Така схема не претендує на відтворення конкретної мережі оператора, але дозволяє контролювати часову залежність, дисбаланс класів і складність переходів. Поділ виконувався хронологічно у співвідношенні 65/15/20 % для train/validation/test без випадкового перемішування часових вікон.

Для нейронних моделей використовувалася зважена cross-entropy, що компенсувала дисбаланс класів, оптимізація виконувалася Adam, а найкращий стан моделі обирався за validation macro-F1. Для XGBoost використовувалися ваги класів. Профілювання latency проводилося у однопотоковому x86-64 CPU-режимі з batch size = 1 після 100 warm-up запусків; для кожної конфігурації виконувалося 500 повторів. У статті абсолютні значення latency трактуються як контрольний відтворюваний профіль, а не як універсальна характеристика всіх edge-пристроїв.

Результати базового порівняння наведено в Таблиці 2. Найвищий macro-F1 отримано для GRU та TCN. XGBoost показав сильний результат для поточного $s(n)$, однак поступився моделям, що використовували коротку причинну послідовність. Transformer у цій

конфігурації не перевищив GRU/TCN, що демонструє відсутність універсальної переваги більш складної архітектури на обмеженому часовому контексті.

Таблиця 2. Якість та характеристики базових моделей

Модель	Macro-F1	Balanced accuracy	Recall крит.	Розмір, КБ	Median, мс	p95, мс
XGBoost	0,897	0,913	0,894	2151,7	0,245	0,403
MLP-FP32	0,868	0,914	0,923	35,5	0,024	0,026
GRU-FP32	0,918	0,947	0,965	51,7	0,096	0,132
TCN-FP32	0,918	0,943	0,951	51,1	0,096	0,217
Transformer-FP32	0,899	0,932	0,937	164,9	0,199	0,409

Джерело: розроблено автором

Порівняння лише за макро-F1 не дає однозначної відповіді щодо edge-придатності. MLP має найнижчу затримку, але його макро-F1 приблизно на 0,05 нижчий, ніж у GRU/TCN. XGBoost забезпечує прийнятну якість, проте його серіалізований артефакт значно більший, а latency у контрольному CPU-профілі найвища серед розглянутих базових конфігурацій. GRU і TCN формують близький за якістю клас моделей, але TCN має дещо нижчу медіанну latency, тоді як GRU показав вищий recall критичного класу.

Результати оптимізації наведено в Таблиці 3. Усі чотири оптимізовані конфігурації пройшли quality-gate $\varepsilon_q = 0,015$. При цьому характер виграшу суттєво відрізнявся залежно від методу.

Таблиця 3. Результати оптимізації нейронних моделей

Конфігурація	Метод	Macro-F1	Розмір, КБ	Median, мс	Зміна size	Зміна latency
MLP-INT8	динамічне INT8	0,868	13,3	0,067	-62,5 %	+183,8 %
GRU-INT8	динамічне INT8	0,918	16,8	0,124	-67,5 %	+28,7 %
TCN-P17	structured pruning	0,919	39,3	0,086	-23,0 %	-10,1 %
Transformer-D	distillation	0,901	43,9	0,102	-73,4 %	-48,8 %

Джерело: розроблено автором

Квантування MLP і GRU зменшило розмір артефакту приблизно у 2,7 та 3,1 раза відповідно без суттєвої втрати predictive quality. Водночас на використаному CPU-профілі їх median latency збільшилася. Для MLP це особливо помітно: абсолютний час базової FP32-моделі настільки малий, що витрати на динамічні операції quantize/dequantize і виклик INT8 kernels переважають виграш від меншої точності. Цей результат узгоджується з необхідністю профілювати оптимізовану модель на цільовому runtime, а не вважати quantization автоматичним способом прискорення.

Структурний pruning TCN мав інший характер. Скорочення каналів зменшило розмір на 23,0 %, median latency – приблизно на 10 %, а макро-F1 не знизився і навіть незначно зріс до 0,919 після fine-tuning. Це означає, що початкова конфігурація мала певну надлишковість, яку вдалося усунути без втрати роздільної здатності моделі. Distillation Transformer дала найбільше одночасне скорочення size і latency: артефакт зменшився на 73,4 %, latency – майже на 49 %, тоді як макро-F1 зріс з 0,899 до 0,901. Однак навіть після compression ця конфігурація поступилася TCN-P17 за якістю та швидкістю.

Співвідношення якості та median latency для всіх базових і оптимізованих конфігурацій наведено на Рис. 1. Видно, що конфігурації формують кілька різних режимів: ультракомпактний MLP, високоякісні GRU/TCN і більш ресурсомісткі Transformer/XGBoost.

Це підтверджує доцільність збереження не однієї моделі, а декількох технічно різних кандидатів.

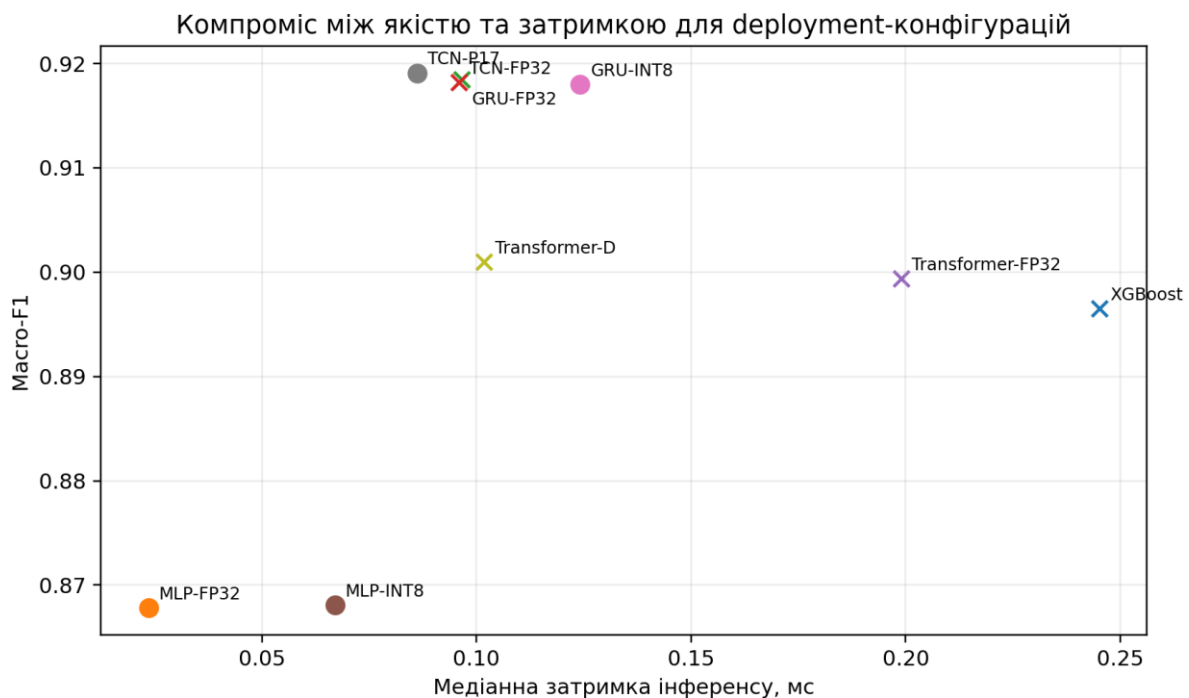


Рис. 1. Компроміс між predictive quality та затримкою deployment-конфігурацій

Джерело: розроблено автором за результатами експерименту

На Рис. 2 показано відносну зміну size та latency після оптимізації. Для size усі способи дали очікуваний вииграш. Для latency ефект виявився неоднорідним: pruning і distillation покращили показник, тоді як dynamic INT8 на контрольному runtime його погіршив. Отже, сам факт зменшення бітності ваг або розміру файлу не є достатнім доказом прискорення inference.

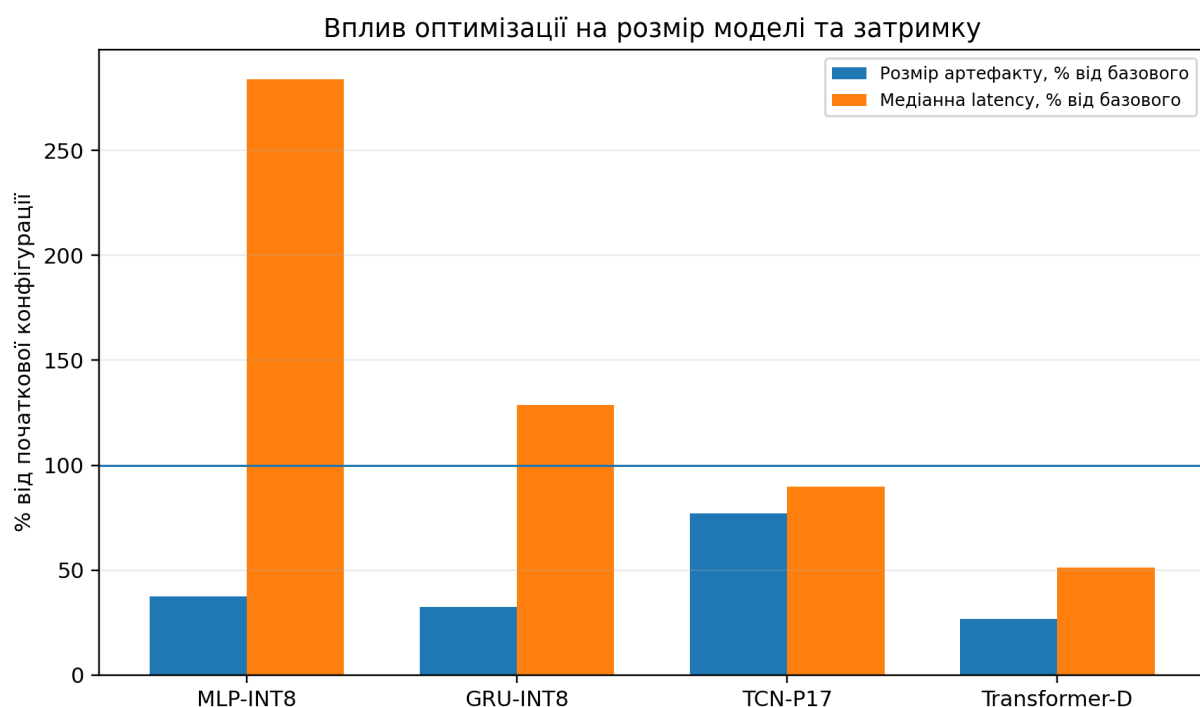


Рис. 2. Вплив оптимізації на розмір артефакту та затримку інференсу

Джерело: розроблено автором за результатами експерименту

Після quality-gate і Pareto-відбору сформовано deployment-банк, наведений у табл. 4. До нього увійшли конфігурації, що представляють різні компроміси. MLP-FP32 має мінімальну latency, MLP-INT8 – мінімальний розмір, GRU-INT8 забезпечує високу якість за дуже малому артефакті, а TCN-P17 показує найвищий macro-F1 серед недомінованих кандидатів із нижчою latency, ніж вихідні GRU/TCN. Інші конфігурації залишаються корисними як контрольні, але в межах трьох базових критеріїв вони домінуються принаймні одним кандидатом із Таблиці 4.

Таблиця 4. Сформований Pareto-банк deployment-конфігурацій

Конфігурація	Macro-F1	Розмір, КБ	Median, мс	Функціональна роль
MLP-FP32	0,868	35,5	0,024	Мінімальна latency
MLP-INT8	0,868	13,3	0,067	Мінімальний розмір
GRU-INT8	0,918	16,8	0,124	Висока якість / компактність
TCN-P17	0,919	39,3	0,086	Найкращий quality-latency баланс

Джерело: розроблено автором

Сформований банк не є результатом runtime-рішення і не визначає єдину «кращу» модель. Його призначення полягає у відсіюванні непродуктивних варіантів і збереженні альтернатив, кожна з яких може бути доцільною за різних ресурсних умов. На наступному рівні саме ці конфігурації можуть порівнюватися з урахуванням стану edge/cloud-вузла, допустимої затримки та поточного інформаційного контексту без повторного навчання моделей.

Висновки. У роботі сформовано процедуру оцінювання та оптимізації моделей машинного і глибокого навчання для edge-обробки інформації мобільної мережі. На відміну від вибору лише за predictive quality, запропонований підхід розглядає модель як deployment-конфігурацію та спільно враховує macro-F1, latency і розмір артефакту. Це дозволяє відокремити етап підготовки технічних кандидатів від подальшого адаптивного вибору моделі та місця її виконання.

Порівняння п'яти базових моделей показало, що складніша архітектура не гарантує кращого результату. Для дослідженої задачі GRU і TCN забезпечили macro-F1 близько 0,918, тоді як Transformer – 0,899. При цьому MLP мав істотно нижчу latency, що робить його корисним як швидкий кандидат для режимів із жорсткими часовими обмеженнями.

Експеримент підтвердив, що ефект оптимізації є архітектурно- і runtime-залежним. INT8-квантування MLP та GRU скоротило розмір на 62,5-67,5 %, але на контрольному CPU-профілі не прискорило inference. Структурний pruning TCN, навпаки, одночасно зменшив розмір і latency без втрати якості. Distillation Transformer скоротила розмір приблизно на 73 % і latency майже вдвічі, але не зробила цю модель Pareto-кращою за TCN.

У результаті Pareto-відбору сформовано компактний deployment-банк із чотирьох конфігурацій, що представляють різні компроміси між якістю, швидкістю та обсягом пам'яті. Такий банк створює обґрунтовану експериментальну основу для подальшого адаптивного вибору ML/DL-моделі та її виконання в розподіленій edge/cloud-інфраструктурі мобільної мережі.

Конфлікт інтересів: Автори декларують, що не мають конфлікту інтересів, зокрема фінансового, особистого, авторського чи будь-якого іншого характеру, який міг би вплинути на дослідження, а також на результати, опубліковані в цій статті

Фінансування: Дослідження проводилося без фінансової підтримки

Доступність даних: Усі дані доступні в цифровій або графічній формі в основному тексті рукопису

Використання засобів штучного інтелекту (ШІ): Автори підтверджують, що не застосовували інструментальні засоби ШІ для написання цієї роботи

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Zhang, C., Patras, P. & Haddadi, H. “Deep learning in mobile and wireless networking: A survey”. *IEEE Communications Surveys & Tutorials*. 2019; 21 (3): 2224–2287. DOI: <https://doi.org/10.1109/COMST.2019.2904897>.

2. Shi, W., Cao, J., Zhang, Q., Li, Y. & Xu, L. “Edge computing: Vision and challenges”. *IEEE Internet of Things Journal*. 2016; 3 (5): 637–646. DOI: <https://doi.org/10.1109/JIOT.2016.2579198>.
3. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K. & Zhang, J. “Edge intelligence: Paving the last mile of artificial intelligence with edge computing”. *Proceedings of the IEEE*. 2019; 107 (8): 1738–1762. DOI: <https://doi.org/10.1109/JPROC.2019.2918951>.
4. Bega, D., Gramaglia, M., Fiore, M., Banchs, A. & Costa-Pérez, X. “DeepCog: Cognitive network management in sliced 5G networks with deep learning”. *IEEE INFOCOM 2019 – IEEE Conference on Computer Communications*. 2019: 280–288. DOI: <https://doi.org/10.1109/INFOCOM.2019.8737488>.
5. Petrella, A., Miozzo, M. & Dini, P. “Mobile traffic prediction at the edge through distributed and deep transfer learning”. *IEEE Access*. 2024; 12: 191288–191303. DOI: <https://doi.org/10.1109/ACCESS.2024.3518483>.
6. Sze, V., Chen, Y.-H., Yang, T.-J. & Emer, J. S. “Efficient processing of deep neural networks: A tutorial and survey”. *Proceedings of the IEEE*. 2017; 105 (12): 2295–2329. DOI: <https://doi.org/10.1109/JPROC.2017.2761740>.
7. Nan, K., Liu, S., Du, J. & Liu, H. “Deep Model Compression for Mobile Platforms: A Survey”. *Tsinghua Science and Technology*. 2019; 24 (6): 677–693. DOI: <https://doi.org/10.26599/TST.2018.9010103>.
8. Cheng, Y., Wang, D., Zhou, P. & Zhang, T. “Model compression and acceleration for deep neural networks: The principles, progress, and challenges”. *IEEE Signal Processing Magazine*. 2018; 35 (1): 126–136. DOI: <https://doi.org/10.1109/MSP.2017.2765695>.
9. Liang, T., Glossner, J., Wang, L., Shi, S. & Zhang, X. “Pruning and quantization for deep neural network acceleration: A survey”. *Neurocomputing*. 2021; 461: 370–403. DOI: <https://doi.org/10.1016/j.neucom.2021.07.045>.
10. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. & Kalenichenko, D. “Quantization and training of neural networks for efficient integer-arithmetic-only inference”. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018: 2704–2713. DOI: <https://doi.org/10.1109/CVPR.2018.00286>.
11. Gou, J., Yu, B., Maybank, S. J. & Tao, D. “Knowledge distillation: A survey”. *International Journal of Computer Vision*. 2021; 129 (6): 1789–1819. DOI: <https://doi.org/10.1007/s11263-021-01453-z>.
12. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.-C. “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018. p. 4510–4520. DOI: <https://doi.org/10.1109/CVPR.2018.00474>.
13. Marinó, G. C., Petrini, A., Malchiodi, D. & Frasca, M. “Deep neural networks compression: A comparative survey and choice recommendations”. *Neurocomputing*. 2023; 520: 152–170. DOI: <https://doi.org/10.1016/j.neucom.2022.11.072>.
14. Mo, R., Pei, Y., Venkatarayalu, N. V., Nathaniel, P., Premkumar, A. B., Sun, S. & Foo, S. K. K. “Unsupervised TCN-AE-based outlier detection for time series with seasonality and trend for cellular networks”. *IEEE Transactions on Wireless Communications*. 2023; 22 (5): 3114–3127. DOI: <https://doi.org/10.1109/TWC.2022.3216004>.
15. Lim, B., Arik, S. Ö., Loeff, N. & Pfister, T. “Temporal fusion transformers for interpretable multi-horizon time series forecasting”. *International Journal of Forecasting*. 2021; 37 (4): 1748–1764. DOI: <https://doi.org/10.1016/j.ijforecast.2021.03.012>.
16. Chen, T. & Guestrin, C. “XGBoost: A scalable tree boosting system”. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016: 785–794. DOI: <https://doi.org/10.1145/2939672.2939785>.
17. Ревенко П. Г., Дерменжи Д. П. та Масєвський Ю. О. «Моделі та методи глибокого навчання для обробки поточкових мультимодальних даних у розподілених системах із граничними обчисленнями». *Наука і техніка сьогодні. Серія «Техніка»*. 2026; 57 (3): 2817–2830. DOI: [https://doi.org/10.52058/2786-6025-2026-3\(57\)-2817-2830](https://doi.org/10.52058/2786-6025-2026-3(57)-2817-2830).

18. Ревенко П. Г. та Стрельцов О. В. “Модель представлення та метод формування ознак потокової інформації мобільної мережі”. *Інформатика. Культура. Техніка*. 2026; 3 (1): 416–428. DOI: <https://doi.org/10.15276/ict.03.2026.33>.

Отримано 07.09.2026

Отримано після перегляду 03.10.2026

Прийнято 12.10.2026

DOI: <https://doi.org/10.15276/ict.03.2026.58>

UDC 004.85:004.75:621.39

Evaluation and optimization of machine learning models for edge processing of mobile network data

Petro H. Revenko¹⁾

ORCID: <https://orcid.org/0009-0003-7243-7826>; petroniy13@stud.op.edu.ua

¹⁾ Odesa Polytechnic National University, 1, Shevchenko Av. Odesa, 65044, Ukraine

ABSTRACT

The article addresses the evaluation and optimization of machine learning and deep learning models for execution on edge nodes when processing mobile network data. The relevance of the study is determined by the fact that models with similar predictive quality may differ substantially in inference latency, model artifact size, and computational requirements; therefore, selection based on accuracy alone does not guarantee suitability for edge deployment. The aim of the study is to build a consistent bank of deployment configurations by jointly evaluating predictive quality and execution characteristics before and after model optimization. The input is a fixed-dimensional integrated mobile network state vector containing statistical, temporal, event-based, resource, and data-quality features. XGBoost, MLP, GRU, TCN, and a Transformer-based model are compared for the same classification task. Optimized variants are produced using dynamic INT8 quantization, structured pruning, and knowledge distillation. Each configuration is characterized by macro-F1, balanced accuracy, recall of the critical class, median and p95 inference latency, and serialized model size. After a quality gate, Pareto screening removes configurations dominated simultaneously in predictive quality, latency, and size. The experimental study uses a synthetic sequence of 17,995 time windows with four mobile network state classes. The highest baseline macro-F1 is obtained by GRU and TCN, both close to 0.918. Structured TCN pruning reduces the model artifact by 23.0% and median latency by about 10% without degrading macro-F1. Distillation reduces the Transformer model size by 73.4% and almost halves its latency. In contrast, dynamic INT8 quantization greatly reduces the size of MLP and GRU models but increases latency on the reference CPU runtime because of quantization overhead. The results demonstrate that edge optimization should be assessed using a set of predictive and execution metrics rather than a single indicator. The resulting deployment bank can serve as a set of verified candidates for subsequent adaptive selection of a model and execution node in a distributed edge/cloud mobile network infrastructure.

Keywords: Mobile networks; machine learning; deep learning; edge computing; model optimization; model compression; quantization; pruning; knowledge distillation

ІНФОРМАЦІЯ ПРО АВТОРА



Ревенко Петро Георгійович - аспірант кафедри Комп'ютерних систем, Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна.

Галузь досліджень: машинне навчання в мобільних мережах, edge computing, оптимізація моделей машинного навчання, потокова обробка даних, розподілені інтелектуальні системи.

Petro H. Revenko - postgraduate, Department of Computer Systems, Odesa Polytechnic National University, 1, Shevchenko Ave. Odesa, 65044, Ukraine.

ORCID: <https://orcid.org/0009-0003-7243-7826>; petroniy13@stud.op.edu.ua

Research field: machine learning in mobile networks, edge computing, machine learning model optimization, streaming data processing, distributed intelligent systems