

# Розширення доступності штучного інтелекту створює нові виклики для розробників вбудованих систем

Йенн Лефау (Yann LeFaou), Microchip Technology

Переклад та редагування: Андрій Середін, PhD, асистент кафедри ПРЕ, РТФ, КПІ ім. Ігоря Сікорського

**В статті розглянуто, як розширення доступності штучного інтелекту (ШІ) створює нові виклики для розробників вбудованих систем, які прагнуть створювати проєкти «ML at the Edge», що працюють ефективно, водночас відповідаючи мінімальним вимогам до процесора та обсягу пам'яті, а також забезпечуючи найнижче можливе енергоспоживання пристроїв Інтернету речей.**

Від систем відеоспостереження та контролю доступу до «розумних» фабрик і прогнозного технічного обслуговування — впровадження штучного інтелекту (ШІ), побудованого на моделях машинного навчання (МН, англ. machine learning, ML), стає все більш поширеним у промислових проєктах периферійної обробки даних в IoT. З огляду на таку поширеність, створення рішень на основі ШІ стало «демократизованим» — воно перетворилося зі спеціалізованої дисципліни для науковців, що займаються даними, на дисципліну, яку повинні розуміти розробники вбудованих систем. Проблема такої «демократизації» полягає в тому, що розробники не завжди мають достатньо знань, щоб коректно сформулювати задачу, яку потрібно вирішити, а також зібрати й організувати дані належним чином. До того ж на відміну від споживчих рішень, для промислових реалізацій ШІ існує дуже мало готових дата сетів (наборів даних), тому їх часто доводиться створювати з нуля — на основі власних даних користувача.

## СТАЄ ЗАГАЛЬНОПРИЙНЯТИМ

Штучний інтелект став мейнстримом, і за багатьма застосунками, які ми

сьогодні сприймаємо як належне — такими як обробка природної мови, комп'ютерний зір, прогнозне технічне обслуговування та добування даних — стоять глибоке навчання (*deep learning, DL*) і машинне навчання (*machine learning, ML*). Перші реалізації ШІ базувалися на хмарних або серверних рішеннях, що вимагали значних обчислювальних ресурсів, обсягів пам'яті та високої пропускнув здатності між застосунком ШІ/ML і периферією (кінцевим пристроєм). І хоча такі архітектури досі необхідні для генеративних застосунків ШІ, таких як ChatGPT, DALL-E чи Bard, останніми роками з'явилися рішення з обробкою ШІ на периферії, де дані обробляються в реальному часі безпосередньо в точці отримання первинної інформації. Такий підхід значно зменшує залежність від хмари, пришвидшує роботу системи або застосунку, знижує енергоспоживання та загальні витрати. Багато хто також вважає, що це підвищує рівень безпеки, хоча точніше буде сказати, що основний фокус безпеки зміщується із захисту переданих даних між хмарою та кінцевим пристроєм на забезпечення безпеки самого периферійного пристрою.

ШІ/ML на периферії можна реалізувати на традиційній вбудованій системі, де розробники мають доступ до потужних мікропроцесорів, графічних

процесорів і великої кількості пам'яті — ресурсів, подібних до тих, що використовуються в ПК. Однак зростає попит на комерційні та промислові IoT-пристрої з підтримкою ШІ/ML периферією, які зазвичай мають обмежені апаратні ресурси й в багатьох випадках працюють від батареї.

Потенціал використання ШІ/ML на периферії, що працює на апаратному забезпеченні з обмеженими ресурсами та енергоспоживанням дав поштовх до появи терміна TinyML. Приклади застосування TinyML уже існують у промисловості (наприклад, для прогнозного технічного обслуговування), автоматизації будівель (моніторинг довкілля), будівництві (контроль безпеки персоналу) та системах безпеки.

## ПОТІК ДАНИХ

Штучний інтелект (а також його підмножина — машинне навчання) потребує чітко визначеного робочого процесу — від збору/зберігання даних до розгортання моделі (рис. 1). У контексті TinyML оптимізація на кожному етапі цього процесу є критично важливою через обмежені ресурси вбудованої системи. Наприклад, вимоги TinyML до ресурсів зазвичай охоплюють тактові частоти від 1 до 400 МГц, обсяг оперативної пам'яті від 2 до 512 КБ і флеш-пам'ять від 32 КБ до 2 МБ. Крім того, за рівнем енергоспоживання від 150 мкВт до 23.5 мВт робота в межах такого обмеженого енергобюджету часто є складним завданням.

Ба більше, існує важливіше питання, або радше компроміс, коли йдеться про впровадження ШІ у вбудовану систему з



**Рис. 1. Спрощений алгоритм роботи ШІ. Не показано, але етап розгортання моделі також має повертати дані назад процес, потенційно впливаючи навіть на етап збору даних**

обмеженими ресурсами. Моделі відіграють ключову роль у поведінці системи, але розробникам часто доводиться балансувати між якістю/точністю моделі, що впливає на надійність і стабільність роботи, та продуктивністю — насамперед швидкістю обробки і рівнем енергоспоживання.

Ще одним ключовим фактором є вибір типу ШІ/ML, який буде застосовано. Загалом існує три типи алгоритмів, які можна використовувати: з навчанням під наглядом (*supervised*), без нагляду (*unsupervised*) та з підкріпленням (*reinforced*).

## РІШЕННЯ

Навіть розробники, які добре розуміють ШІ та ML, можуть постати перед труднощами при оптимізації кожного етапу робочого процесу ШІ/ML і пошуку ідеального балансу між точністю моделі та продуктивністю системи. То як же розробникам вбудованих систем без попереднього досвіду подолати ці виклики?

Передусім важливо не втрачати з поля зору той факт, що моделі, розгорнуті на IoT-пристроях з обмеженими ресурсами, будуть ефективними лише за умови, що сама модель є компактною, а завдання ШІ обмежується розв'язанням простої задачі.

На щастя, поява ML (а особливо TinyML) у сфері вбудованих систем призвела до створення нових (або вдосконалених) інтегрованих середовищ розробки (*integrated development environments, IDE*), програмних інструментів, архітектур і моделей, багато з яких є відкритими. Наприклад, TensorFlow™ Lite for Microcontrollers (TF Lite Micro) — це безплатна та відкрита програмна бібліотека для ML і ШІ. Вона розроблена для реалізації ML на пристроях з усього кількома кілобайтами пам'яті. Крім того, програми можна писати мовою Python, яка також є відкритою та безплатною.

Щодо середовищ розробки IDE, прикладом є MPLAB® X від компанії Microchip. Це IDE можна використовувати

вразом із MPLAB ML — плагіном до MPLAB X, спеціально розробленим для створення оптимізованого коду розпізнавання сенсорних даних з використанням ШІ для IoT. Завдяки AutoML, *MPLAB ML* повністю автоматизує всі етапи робочого процесу ШІ/ML, усуваючи потребу в повторюваному, рутинному та трудомісткому створенні моделей. Етапи виділення ознак, навчання, валідації та тестування забезпечують створення оптимізованих моделей, які відповідають обмеженням пам'яті мікроконтролерів і мікропроцесорів, дозволяючи розробникам швидко створювати та розгорнути ML-рішення на 32-бітних мікроконтролерах або мікропроцесорах на базі Arm® Cortex від Microchip.

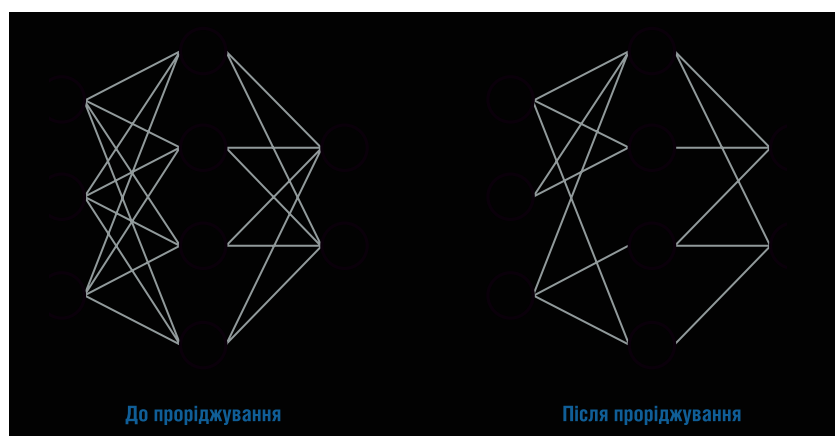
## ПРОЦЕС

Завдання з оптимізації робочого процесу можна спростити, почавши з готових наборів даних і моделей. Наприклад, якщо IoT-пристрій із підтримкою ML має вміння розпізнавати зображення, доцільно почати з наявного набору розмічених статичних зображень і відеокліпів для навчання моделі (а також її тестування та оцінювання); варто зазначити, що для алгоритмів машинного навчання з навчанням під наглядом потрібні саме розмічені дані.

Існує багато готових наборів зображень для застосунків комп'ютерного зору. Однак, оскільки вони призначені для використання на ПК, серверах або в хмарних середовищах, їх обсяг зазвичай є значним. Наприклад, ImageNet містить понад 14 мільйонів анотованих зображень.

Залежно від застосування ML, може знадобитися лише кілька підмножин даних, скажімо, багато зображень людей, але лише кілька зображень неживих об'єктів. Наприклад, якщо ML-камери використовуються на будівельному майданчику, вони можуть негайно подавати сигнал тривоги, якщо в полі зору з'являється людина без захисної каски. Для цього ML-модель має бути навчена, можливо, лише на кількох зображеннях людей з касками та без них. Водночас може знадобитися більший набір даних для типів касок і достатнє різноманіття в межах дата сету (наборів даних), щоб врахувати такі чинники, як різні умови освітлення.

Наявність правильних вхідних «живих» даних (в режимі реального часу) і відповідного набору даних, підготовка цих даних і навчання моделі відповідають етапам 1–3 на рисунку 1. Оптимізація моделі (етап 4 зазвичай полягає в її стисненні, що допомагає зменшити вимоги до пам'яті (RAM під час обробки та NVM для зберігання), а також скоротити затримку обробки.

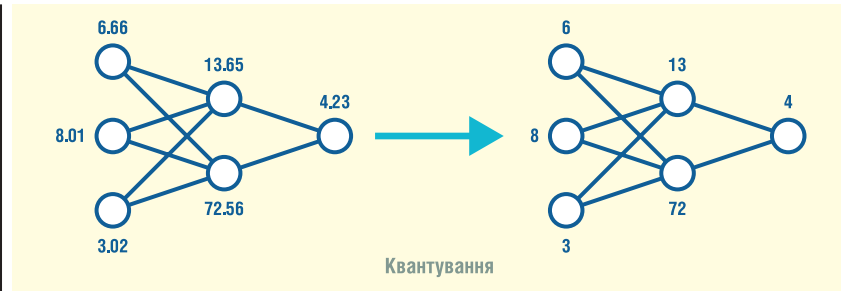


**Рис. 2. Проріджування зменшує щільність нейронної мережі. На схемі вага деяких зв'язків між нейронами встановлена на нуль. Не показано, але іноді проріджуються також самі нейрони**

Щодо обробки, багато алгоритмів ШІ, зокрема згорткові нейронні мережі (*convolutional neural networks, CNN*), мають труднощі з обробкою складних моделей. Одним із популярних методів стиснення є проріджування (*pruning*, рис. 2), яке має кілька типів: проріджування ваги (*weight pruning*), проріджування нейронів/вузлів (*unit/neuron pruning*) та ітеративне проріджування (*iterative pruning*).

Квантування — ще один популярний метод стиснення. Це процес перетворення даних із формату високої точності, наприклад, 32-бітного числа з плаваючою комою (FP32), у формат нижчої точності, скажімо, 8-бітне ціле число (INT8). Використання квантованих моделей (рис. 3) може бути враховане на етапі машинного навчання одним із двох способів:

- Постквантування (квантування після навчання) (*post-training quantization*) передбачає використання моделей, наприклад, у форматі FP32, а після завершення навчання — їх квантування для розгортання. Наприклад, стандартний TensorFlow можна використовувати для початкового навчання та оптимізації моделі на ПК. Потім модель можна квантувати і, за допомогою TensorFlow Lite, вбудувати в IoT-пристрій.
- Навчання з урахуванням квантування (*quantization-aware training, QAT*) імітує квантування, яке відбувається



**Рис. 3.** Квантовані моделі використовують нижчу точність, що дозволяє зменшити вимоги до пам'яті та обсягу сховища, а також підвищити енергоефективність, при цьому зберігаючи ту саму структуру моделі

під час виконання моделі (отримання логічного виведення) (*inference*), створюючи модель, яку інструменти нижчого рівня використовуватимуть для генерації квантованих версій (створення квантованих моделей).

Хоча квантування є корисним, його не слід застосовувати надмірно, адже це можна порівняти зі стисненням цифрового зображення шляхом зменшення кількості бітів для представлення кольорів та/або зменшення кількості пікселів, тобто настане момент, коли зображення стане складним для інтерпретації.

## ВИСНОВКИ

Як зазначалося на початку, ШІ вже міцно закріпився у сфері вбудованих систем. Однак така «демократизація» означає, що інженери-конструктори, які

раніше не мали потреби розуміти ШІ та ML, тепер стикаються з викликами впровадження рішень на основі ШІ у свої розробки.

Хоча створення ML-застосунків із максимально ефективним використанням обмежених апаратних ресурсів може здаватися складним завданням, для досвідчених розробників вбудованих систем це не нова проблема. Хороша новина полягає в тому, що інженерній спільноті вже доступна велика кількість інформації та навчальних матеріалів, а також середовища розробки, як-от MPLAB X, генератори моделей, як-от MPLAB ML, і відкриті набори даних та моделі. Ця екосистема допомагає інженерам з різним рівнем підготовки прискорити створення рішень на основі ШІ та ML, які тепер можна реалізувати навіть на 16-бітних або 8-бітних мікроконтролерах.

CN

## СЕРВЕР MODEL CONTEXT PROTOCOL (MCP) ДЛЯ ЗАБЕЗПЕЧЕННЯ ДОСТУПУ ДО ДАНИХ ПРО ПРОДУКТИ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Продовжуючи демонструвати свою прихильність до розвитку рішень на основі штучного інтелекту для розробників вбудованих систем, компанія **Microchip** оголосила про запуск сервера Model Context Protocol (MCP). Сервер MCP, який є інтерфейсом штучного інтелекту, безпосередньо підключається до сумісних інструментів штучного інтелекту та великих мовних моделей (Large Language Models, LLM), щоб надавати контекст, необхідний цим системам для відповіді на запитання. За допомогою простих діалогових запитів сервер MCP дозволяє користувачам отримувати перевірені, актуальні публічні дані Microchip, включаючи технічні характеристики продуктів, специфікації, запаси, ціни та терміни постачання.

Створений на основі стандартів MCP потокового HTTP, сервер надає контекстно-залежні та кодовані в JSON відповіді, оптимізовані для клієнтів штучного інтелекту, таких як копілоти, чат-боти зі штучним інтелектом, IDE на основі LLM та корпоративні агенти штучного інтелекту. Платформа підтримує широкий спектр додатків та інтегрує публічні дані Microchip безпосередньо в середовища розробки та інтелектуальні помічники.

«Запуск нашого сервера MCP — це ще один приклад того, як Microchip використовує штучний інтелект і надає інструмен-

ти на основі штучного інтелекту, які допомагають полегшити життя наших клієнтів, — сказав Річ Сімончіч (Rich Simoncic), головний операційний директор Microchip Technology. — Ми прагнемо використовувати можливості штучного інтелекту для підвищення продуктивності та стимулювання інновацій. Забезпечуючи миттєвий доступ до перевіреної інформації про продукти в рамках платформ штучного інтелекту, якими вже користуються розробники, ми усуваємо бар'єри та спрощуємо процес проектування за допомогою рішень Microchip».

Поєднуючи надійну технічну інформацію та інформацію про постачальників, сервер MCP може підвищити продуктивність проектування, оптимізувати автоматизацію та надати інженерам можливість швидше розробляти та приймати більш обґрунтовані рішення.

Цей запуск підкреслює постійну прихильність Microchip до просування цифрової трансформації та впровадження штучного інтелекту в екосистемах підприємств. Сервер Microchip MCP доступний для користувачів безплатно.

Відвідайте вебсайт компанії, щоб отримати доступ до HTML-коду кінцевої точки та розпочати роботу.

[www.microchip.com](http://www.microchip.com)