Левицький А.А., магістрант,
Левківський В.Л., Ph.D, доцент
Державний університет «Житомирська політехніка»

МЕТОДИ ІНТЕГРАЦІЇ НЕЙРОННИХ МЕРЕЖ В ІГРОВІ ДОДАТКИ НА БАЗІ РУШІЯ UNITY, ЇХ ПЕРЕВАГИ ТА НЕДОЛІКИ

Нейронні мережі є потужним інструментом для аналізу та генерування даних. Для ігрових додатків вони можуть надати великі можливості у створенні тексту діалогів, поведінки об'єктів або навіть кардинально нових шляхів побудови ігроладу. Однак стоїть питання правильно підібраного методу інтеграції нейронної мережі в конкретній ситуації, особливо якщо інструмент, на якому розробляється ігровий додаток, відстає в роботі зі складними мережами. До таких також відноситься мова програмування C#, яка застосовується у популярному серед розробників ігровому рушії Unity.

Існує три основні методи інтеграції нейронної мережі в ігровий додаток на обраному рушії, а саме:

1. Створити нейронну мережу всередині ігрового додатка за допомогою мови програмування C#;
2. Використати пакунок Sentis ігрового рушія Unity та за допомогою бібліотеки ONNX (Open Neural Network Exchange) інтегрувати модель нейронної мережі, створеної іншим інструментом;
3. Запустити окрему програму на віддаленому сервері з нейронною мережею, до якої буде звертатись ігровий додаток.

Перший метод передбачає створення, навчання та використання нейронної мережі в ігровому додатку, використовуючи мову програмування C# та її бібліотек. Метод реалізує мережу всередині ігрового додатка та дає повний контроль над нею, а саме змінювати структуру, перенавчати мережу за допомогою вдосконалених наборів даних тощо. Однак він не є оптимальним при роботі з величезними наборами даних та тренування глибоких нейронних мереж. Також бібліотеки обраної мови обмежено використовують переваги графічних процесорів, що сповільняє мережу в порівнянні з конкурентами.

Другий метод полягає у застосуванні бібліотеку ONNX [1], яка дозволяє експортувати будь-яку нейронну мережу в модель та імпортувати в інше середовище зі всіма перевагами. Це дозволяє використовувати як існуючі мережі, так і створювати власні за допомогою кращих інструментів для створення та навчання мереж, як от мова Python та її бібліотеки. Однак метод не дозволяє напряму перенавчити імпортовану модель та обмежено може змінити її структуру, що не є гнучким у випадках, коли є намір динамічно доповнювати набір даних мережі. У випадку з ігровим рушієм Unity, пакунок Sentis [2] оптимізує використання такого методу.

Третій метод використовує сервер з нейронною мережею, до якого звертається додаток для отримання результату. Це дозволяє гравцям з непотужними системами грати в ігри зі складними нейронними мережами, а для розробників дає можливість їх вдосконалювати без потреби в змінах додатка. В такому разі мережу можна реалізувати за допомогою сторонніх інструментів та не виконувати імпорт для інтеграції в додаток. Однак метод потребує додаткових витрат та підтримки, що робить додаток актуальним лише певний проміжок часу, адже зазвичай сервери для подібних цілей підтримуються не довго.

В результаті було розглянуто три методи інтеграції нейронних мереж в ігрові додатки на базі рушія Unity та мови програмування C#. Після їх аналізу було зроблено наступні висновки щодо кожного з методів:

1. Створення нейронної мережі всередині додатка дає повний контроль над нею та не потребує додаткових інструментів, однак метод не підходить для глибокого навчання та обмежено використовує графічні процесори. Ним доречно інтегрувати прості мережі, а також проводити дослідження щодо їх роботи;
2. Використання пакунку Sentis та бібліотеки ONNX дають змогу створювати нейронні мережі за допомогою бажаних інструментів, а також використовувати існуючі моделі мереж, однак метод не дозволяє динамічно покращувати набори даних. Метод є універсальним та підходить для задач, де потрібні саме обчислювальні переваги мереж;
3. Сервер дає змогу вдосконалювати та керувати нейронною мережею без залежності до додатка, що дає змогу використовувати складні мережі з повним контролем та не навантажувати системи користувачів. Але метод потребує додаткових витрат та підтримки, а також він не довговічний. Він підходить для багатокористувацьких додатків, де результат обчислень повинен бути схожим у всіх, або в додатках зі складними нейронними мережами.

Список використаних джерел:

1. The Linux Foundation. Open Neural Network Exchange. 2019. URL: https://onnx.ai/ (дата звернення 19.02.2025).
2. Unity Manual. Sentis overview. 2025. URL: https://docs.unity3d.com/Packages/com.unity.sentis@2.1 (дата звернення: 19.02.2025).