МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ЛЕКЦІЯ 1. ВСТУП ДО ШТУЧНОГО ІНТЕЛЕКТУ ТА МАШИННОГО НАВЧАННЯ В РОЗРОБЦІ ІГОР

Львів -- 2026

Лекція зі штучного інтелекту 2026-01

Вступ

Ця лекція відкриває курс «Штучний інтелект в ігрових застосунках». Мета курсу — надати студентам практичні навички застосування методів машинного навчання в контексті розробки ігор: від класифікації зображень та комп'ютерного зору до навчання з підкріпленням та генеративних моделей.

Ігрова індустрія є одним із найактивніших споживачів технологій ШІ. Від простих алгоритмів переслідування в Pac-Man (1980) до сучасних NPC із мовленнєвими моделями — кожне покоління ігор розширювало межі застосування ШІ. Сьогодні машинне навчання використовується для генерації контенту, адаптивної складності, тестування ігор, поведінки NPC та багатьох інших задач.

Теми, що розглядаються

1. Розуміння штучного інтелекту: три парадигми
2. Коротка історія ШІ: від Тюрінга до ChatGPT
3. Таксономія машинного навчання
4. Штучний інтелект в ігровій індустрії
5. Інструменти курсу та середовище розробки
6. Ключові поняття для лабораторної роботи №1
7. Огляд курсу та дорожня карта

Розуміння штучного інтелекту

Штучний інтелект (ШІ) — це галузь інформатики, що вивчає створення систем, здатних виконувати задачі, які традиційно вимагають людського інтелекту: розпізнавання образів, прийняття рішень, розуміння мови, планування тощо. Існують три основні парадигми побудови інтелектуальних систем.

Символьний (логічний) ШІ

Перший підхід до ШІ базувався на формальній логіці та маніпуляції символами. Системи цього типу працюють з явними правилами, що описують знання про предметну область.

- **Експертні системи** — набори правил типу «ЯКЩО... ТО...», створені фахівцями вручну
- **Бази знань і онтології** — структуровані графи понять та відношень між ними
- **Дерева рішень** — деревоподібні алгоритми прийняття рішень

Приклад у іграх: скінченні автомати станів (FSM) — класичний метод керування поведінкою NPC. Ворог може перебувати у стані «патрулювання», «переслідування» або «атака», переходячи між ними за чіткими правилами. Саме так працює більшість NPC у класичних іграх.

Обмеження: символьний підхід вимагає ручного кодування правил, погано масштабується та не здатний навчатися з даних.

Статистичний ШІ та машинне навчання

Другий підхід — побудова моделей, що автоматично знаходять закономірності в даних. Замість явного програмування правил, алгоритм отримує набір прикладів і самостійно виводить узагальнення.

- **Лінійна та логістична регресія** — базові статистичні моделі
- **Метод опорних векторів (SVM)** — класифікація з максимальним відступом
- **Випадковий ліс (Random Forest)** — ансамблеві методи на основі дерев рішень
- **Кластеризація (k-means, DBSCAN)** — групування об'єктів без міток

Приклад у іграх: процедурна генерація контенту — використання статистичних моделей для створення рівнів, ландшафтів, квестів із контрольованою різноманітністю.

Нейронні мережі та глибоке навчання

Третій і найсучасніший підхід використовує штучні нейронні мережі — математичні моделі, натхненні біологічними нейронами. «Глибоке навчання» (*deep learning*) означає використання мереж з багатьма прихованими шарами, що дозволяє автоматично вивчати ієрархічні ознаки з сирих даних.

- **Згорткові нейронні мережі (CNN)** — для обробки зображень
- **Рекурентні мережі (RNN, LSTM)** — для послідовних даних
- **Трансформери** — архітектура, що лежить в основі сучасних мовних моделей
- **Генеративні моделі (GAN, дифузійні моделі)** — для створення нового контенту

Приклад у іграх: сучасні NPC з мовленнєвими моделями (Nvidia ACE, Inworld AI), де діалоги генеруються великою мовною моделлю (LLM) в реальному часі замість заскриптованих відповідей.

Коротка історія ШІ: від Тюрінга до ChatGPT

Розвиток штучного інтелекту можна простежити через ключові віхи:

- **1950** — Алан Тюрінг формулює «тест Тюрінга» — критерій інтелектуальності машини
- **1956** — Дартмутська конференція — офіційне народження ШІ як наукової дисципліни
- **1960-70-ті** — Перші експертні системи, елементарні ігрові ШІ (шахові програми)
- **1980-ті** — «Зима ШІ» — розчарування в символьному підході; паралельно — поява нейронних мереж (backpropagation, 1986)
- **1997** — Deep Blue (IBM) перемагає чемпіона світу з шахів Гаррі Каспарова
- **2012** — AlexNet перемагає у змаганні ImageNet — переворот у глибокому навчанні для комп'ютерного зору
- **2015** — **ResNet** (He et al.) — залишкові з'єднання дозволяють тренувати мережі із 100+ шарами. Цю архітектуру ви будете використовувати у лабораторній роботі №1
- **2016** — AlphaGo (DeepMind) перемагає чемпіона світу з го Лі Седоля — прорив навчання з підкріпленням
- **2017** — Архітектура Transformer («Attention Is All You Need»); **Grad-CAM** — метод візуалізації рішень CNN, який ви реалізуєте у лабораторній роботі №1
- **2020** — GPT-3 — перша велика мовна модель загального призначення
- **2022** — ChatGPT, Stable Diffusion — генеративний ШІ стає доступним масовому користувачу
- **2023** — GPT-4, Claude — мультимодальні LLM (текст + зображення)
- **2024-2026** — AI-агенти, мультимодальні моделі реального часу, ШІ в ігровому продакшені

Зверніть увагу: **ResNet** (2015) та **Grad-CAM** (2017) — це ключові технології, з якими ви працюватимете вже у першій лабораторній роботі.

Таксономія машинного навчання

Машинне навчання (*machine learning*) — це підгалузь ШІ, де алгоритми навчаються з даних замість того, щоб бути явно запрограмованими. Виділяють чотири основні парадигми.

Навчання з учителем (Supervised Learning)

Це найпоширеніша парадигма, де модель тренується на наборі пар «вхідні дані — правильна відповідь» (мітки, *labels*). Модель навчається знаходити відображення з простору вхідних даних у простір відповідей.

Формально, маючи набір даних ${(x_i, y_i)}_{i=1}^{N}$, де x_i — вхідне зображення, а y_i — мітка, ми шукаємо параметри θ моделі f_θ, що мінімізують функцію втрат:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(x_i), y_i)$$

де ℓ — функція втрат (*loss function*), наприклад, **крос-ентропія** (*cross-entropy*) для задач класифікації.

Навчання з учителем розділяється на два типи задач:

- **Класифікація** — передбачення дискретного класу (мітки). Приклад: «це зображення — кіт» або «це зображення — собака»
- **Регресія** — передбачення неперервної величини. Приклад: передбачення рейтингу гравця на основі його статистики

Приклад у іграх: класифікація ігрових сцен за типами (денні/нічні, indoor/outdoor), що є основою лабораторної роботи №1.

Навчання без учителя (Unsupervised Learning)

Модель працює з даними без міток і шукає приховану структуру:

- **Кластеризація** — групування схожих об'єктів (k-means, DBSCAN). Приклад: сегментація гравців за стилем гри для персоналізації контенту
- **Зменшення розмірності** — PCA, t-SNE, UMAP — для візуалізації та стиснення даних
- **Виявлення аномалій** — знаходження нетипових поведінок. Приклад: автоматичне виявлення читерів

Навчання з підкріпленням (Reinforcement Learning)

Агент (*agent*) взаємодіє з середовищем (*environment*), виконуючи дії (*actions*) і отримуючи винагороди (*rewards*). Мета — навчитися стратегії (політиці), що максимізує сумарну винагороду.

- **Агент** — сутність, що приймає рішення (NPC, бот, ігровий персонаж)
- **Середовище** — ігровий світ, з яким взаємодіє агент
- **Нагорода** — числовий сигнал зворотного зв'язку (набрані очки, перемога/поразка)

Приклад у іграх: AlphaGo (го), AlphaStar (StarCraft II), OpenAI Five (Dota 2) — агенти, що досягли надлюдського рівня через самонавчання.

Генеративний ШІ (Generative AI)

Моделі, що створюють новий контент: тексти, зображення, звуки, 3D-об'єкти.

- **GAN (Generative Adversarial Networks)** — генератор проти дискримінатора
- **Дифузійні моделі** — Stable Diffusion, DALL-E — генерація зображень з текстового опису

- **Великі мовні моделі (LLM)** — GPT, Claude — генерація тексту, коду, діалогів

Приклад у іграх: процедурна генерація текстур та спрайтів, генерація квестових описів, NPC з динамічними діалогами.

Штучний інтелект в ігровій індустрії

Класичний game AI: детерміновані алгоритми

Найпершим і донині найпоширенішим підходом є програмування поведінки вручну:

- **Pac-Man (1980)** — чотири привиди мають різні «характери», реалізовані через прості алгоритми: Blinky переслідує безпосередньо, Pinky намагається перехопити попереду, Inky розраховує позицію відносно Blinky, Clyde переслідує, але тікає, коли наближається. Це не ML, але це game AI.
- **Скінченні автомати (FSM)** — стани + переходи: «патрулювання → виявлення ворога → переслідування → атака → відступ». Більшість NPC у іграх до 2010-х працювали саме так.
- **Дерева поведінки (Behavior Trees)** — гнучкіша альтернатива FSM, де дії організовані в деревоподібну структуру з пріоритетами. Використовуються в Halo, Unreal Engine.
- **Алгоритм A*** — пошук найкоротшого шляху на графі. Основа навігації NPC у більшості 3D-ігор (NavMesh в Unity та Unreal).

Стратегічний ШІ

У стратегічних іграх AI повинен планувати на довготривалу перспективу:

- **Civilization** — AI керує цілою цивілізацією: економіка, дипломатія, військо
- **StarCraft: Brood War** — AI-турніри (AIIDE, SSCAIT), де боти змагаються в стратегії реального часу
- **Шахові рушії** — від Deep Blue (1997, пошук по дереву) до Stockfish + NNUE (нейромережева оцінка позиції)
- **Monte Carlo Tree Search (MCTS)** — алгоритм, що поєднує випадковий пошук з деревом, ключовий для AlphaGo

Сучасний ML-driven game AI

Машинне навчання відкриває нові можливості:

- **Unity ML-Agents** — фреймворк для тренування ігрових агентів через навчання з підкріпленням безпосередньо в Unity
- **No Man's Sky** — процедурна генерація цілих планет з використанням алгоритмічних та ML-підходів
- **NPC з мовленнєвими моделями** — Nvidia ACE (Audio2Face, Nemotron), Inworld AI — NPC, що ведуть вільні діалоги замість заскриптованих
- **Автоматичне тестування ігор** — ML-агенти, що проходять рівні, знаходять баги та перевіряють баланс

Комп'ютерний зір для ігрових застосунків

Задачі комп'ютерного зору безпосередньо пов'язані з лабораторною роботою №1:

- **Класифікація зображень** — автоматичне категоризування ігрових скріншотів, ресурсів, текстур за типами
- **Grad-CAM** — розуміння, «куди дивиться» модель на ігровій сцені, виявлення упередженості моделі
- **Семантична сегментація** — класифікація кожного пікселя ігрового кадру для генерації мап прохідності, визначення зон інтересу
- **Комбінований конвеєр** — сегментація сцени → виявлення об'єктів → класифікація кожного об'єкта

Інструменти курсу та середовище розробки

Python та PyTorch

У цьому курсі ми використовуємо **Python** як основну мову програмування та **PyTorch** як фреймворк глибокого навчання. PyTorch забезпечує:

- Тензорні обчислення з підтримкою GPU (CUDA)
- Автоматичне диференціювання (autograd) для обчислення градієнтів
- Модулі для побудови нейронних мереж (`torch.nn`)
- Оптимізатори (`torch.optim`) — Adam, SGD та інші
- Завантажувачі даних (`torch.utils.data`) — `DataLoader` , `Dataset`

torchvision — бібліотека-компаньйон, що містить попередньо натреновані моделі (ResNet, DeepLabV3), утиліти для обробки зображень (`transforms`) та стандартні набори даних.

Google Colab та GPU

Для тренування нейронних мереж потрібний GPU (графічний процесор). У курсі ми використовуємо:

- **Google Colab** — безкоштовне хмарне середовище з GPU NVIDIA T4 (16 ГБ VRAM)
- **VS Code з розширенням Google Colab** — рекомендований спосіб роботи, що поєднує зручність локального IDE з хмарним GPU
- **Jupyter Notebooks** (`.ipynb`) — інтерактивний формат, де код, текст та візуалізації поєднані в одному документі
- **CUDA** — платформа NVIDIA для паралельних обчислень на GPU

Набори даних

У курсі ми працюватимемо з такими наборами даних:

- **ImageNet** — 1.2 млн зображень, 1000 класів. Використовується для попереднього навчання (pre-training) моделей, які ми потім адаптуємо під наші задачі
- **COCO (Common Objects in Context)** — набір із 21 класом об'єктів для семантичної сегментації (фон, кіт, собака, людина, автомобіль тощо)
- **Oxford-IIIT Pet** — 37 порід котів та собак, ~200 зображень на породу. Основний набір для лабораторної роботи №1
- **Caltech-101** — 101 категорія об'єктів. Альтернативний набір для лабораторної роботи №1

Огляд курсу: 18 лекцій

1. **Вступ до ШІ та машинного навчання в розробці ігор** (ця лекція)
2. Навчання з учителем: регресія, класифікація, оптимізація
3. Нейронні мережі: архітектура, тренування, регуляризація
4. Вступ до комп'ютерного зору та обробки зображень
5. Згорткові нейронні мережі та передавальне навчання
6. Виявлення об'єктів, сегментація та розуміння сцен
7. Обробка послідовностей: RNN, LSTM, увага
8. Архітектура Transformer
9. Великі мовні моделі: від пре-тренування до RLHF
10. Основи навчання з підкріпленням
11. Глибоке навчання з підкріпленням
12. Game AI: самогра, мультиагентні системи, практичні інструменти
13. Генеративні моделі: GAN, VAE, дифузійні моделі

14. Fine-tuning, RAG та адаптація до домену
15. AI-агенти: архітектура, інструменти, оркестрація
16. Машинне навчання в продакшені (MLOps)
17. Техніки ШІ в сучасній розробці відеоігор
18. Нові тенденції, етика та безперервне навчання в ШІ

Ключові поняття для лабораторної роботи №1

У першій лабораторній роботі ви будете працювати з наступними концепціями. Ознайомтеся з ними заздалегідь:

Архітектура та компоненти CNN:

- **Згортковий шар (Convolutional Layer)** — застосовує набір навчених фільтрів до вхідного зображення для виявлення ознак (краї, текстури, форми)
- **Шар підвибірки (Pooling Layer)** — зменшує просторові розміри карт ознак, зберігаючи ключову інформацію (Max Pooling)
- **Повнозв'язний шар (Fully Connected Layer)** — перетворює вивчені ознаки у фінальне передбачення
- **Функції активації** — ReLU (для прихованих шарів), Softmax (для виходу класифікатора — перетворює скори у ймовірності)
- **ResNet** — архітектура із залишковими з'єднаннями (*skip connections*): $H(x) = F(x) + x$, що дозволяють тренувати глибокі мережі
- **Backbone** — базова мережа (наприклад, ResNet-18), що витягує ознаки із зображення

Методи навчання:

- **Передавальне навчання (Transfer Learning)** — використання моделі, натренованої на великому наборі (ImageNet), як відправної точки для нової задачі
- **Заморожування шарів (Freezing)** — фіксація ваг ранніх шарів, які вже навчилися виявляти універсальні ознаки
- **Тонке налаштування (Fine-tuning)** — дотренування верхніх шарів моделі на власних даних
- **Аугментація даних (Data Augmentation)** — випадкові трансформації (обрізання, відзеркалення, зміна кольору) для збільшення ефективного обсягу набору даних
- **Функція втрат (Loss Function)** — міра різниці між передбаченням і правильною відповіддю. Для класифікації — **крос-ентропія (Cross-Entropy Loss)**
- **Оптимізатор (Optimizer)** — алгоритм оновлення ваг моделі (Adam, SGD)
- **Learning Rate Scheduler** — стратегія зменшення швидкості навчання протягом тренування

Візуалізація та сегментація:

- **Grad-CAM (Gradient-weighted Class Activation Mapping)** — техніка візуалізації, що показує теплову карту (*heatmap*) областей зображення, найважливіших для рішення класифікатора
- **Семантична сегментація** — класифікація кожного пікселя зображення. Результат — маска розміром $H \times W$, де кожен піксель має мітку класу
- **DeepLabV3** — архітектура для сегментації з atrous convolution (розширеними згортками) та модулем ASPP (Atrous Spatial Pyramid Pooling)
- **Сегментація екземплярів (Instance Segmentation)** — поєднання семантичної сегментації з виявленням окремих екземплярів об'єктів

Метрики оцінки якості:

- **Accuracy** — частка правильних передбачень серед усіх
- **Precision** — частка справді позитивних серед передбачених позитивних
- **Recall** — частка знайдених позитивних серед усіх справжніх позитивних
- **F1-score** — гармонійне середнє Precision та Recall
- **IoU (Intersection over Union)** — основна метрика сегментації: відношення площі перетину передбаченої та правильної маски до площі їх об'єднання

Висновок

У цій лекції ми оглянули три парадигми штучного інтелекту — символьну, статистичну та нейромережеву — і побачили, як кожна з них знаходить застосування в ігровій індустрії. Ми простежили розвиток ШІ від тесту Тюрінга до ChatGPT, розглянули чотири парадигми машинного навчання (з учителем, без учителя, з підкріпленням, генеративний ШІ) та їх конкретні приклади в іграх.

Ви ознайомилися з інструментами курсу (Python, PyTorch, Google Colab) та набули базове розуміння термінології, необхідної для виконання лабораторної роботи №1: класифікація зображень із передавальним навчанням, візуалізація Grad-CAM, семантична сегментація та їх поєднання у комбінований конвеєр.

На наступній лекції ми детально розглянемо навчання з учителем: лінійну регресію, градієнтний спуск, функції втрат та методи оптимізації.