**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**ІКНІ**

Кафедра **ПЗ**



**ЗВІТ**

до лабораторної роботи №5

**на тему:** *“Вступ до PyTorch та генерація музики за допомогою рекурентних нейронних мереж (RNN).”*

**з дисципліни** *“Штучний інтелект в ігрових застосунках”*

**Лектор:**

асис. каф. ПЗ

Бауск О. Є.

**Виконав:**

ст. гр. ПЗ-33

Юшкевич. А.І.

**Прийняв:**

ст. викл. каф. ПЗ

Бауск О. Є.

«\_\_» \_\_\_\_\_\_\_\_\_ 2025р.

∑=\_\_\_\_\_

Львів – 2025

**Тема роботи:** Вступ до PyTorch та генерація музики за допомогою рекурентних нейронних мереж (RNN).

**Мета роботи:** Ознайомитись з основами бібліотеки глибокого навчання PyTorch, навчитись визначати та тренувати прості нейронні мережі, а також застосувати рекурентні нейронні мережі для задачі генерації музики у форматі ABC notation.

# Теоретичні відомості

**Вступ до PyTorch**

PyTorch — це популярна бібліотека глибокого навчання з відкритим кодом, відома своєю гнучкістю, простотою використання та динамічним графом обчислень. Вона широко використовується в дослідженнях та промисловості для створення та тренування нейронних мереж.

Основні концепції PyTorch:

1. **Тензори (Tensors)**: Багатовимірні масиви, схожі на NumPy ndarrays, але з додатковою можливістю виконувати обчислення на GPU для прискорення. Тензори є основними будівельними блоками для даних у PyTorch.
2. **Автоматичне диференціювання (Autograd)**: PyTorch автоматично обчислює градієнти для тензорів. Модуль torch.autograd відстежує операції над тензорами, дозволяючи легко реалізовувати зворотне поширення помилки для тренування мереж.
3. **Модулі (torch.nn.Module)**: Клас nn.Module є базовим для всіх нейронних мереж у PyTorch. Він дозволяє інкапсулювати параметри моделі та операції в зручні об'єкти. Мережі можна будувати, комбінуючи існуючі модулі (шари) або створюючи власні.
4. **Оптимізатори (torch.optim)**: Містить реалізації різноманітних алгоритмів оптимізації (наприклад, SGD, Adam), які використовуються для оновлення параметрів моделі під час тренування.
5. **Функції втрат (torch.nn)**: Надає стандартні функції втрат (наприклад, CrossEntropyLoss, MSELoss), що використовуються для оцінки різниці між прогнозами моделі та реальними даними.

PyTorch дозволяє визначати моделі двома основними способами:

* **torch.nn.Sequential**: Контейнер для послідовного з'єднання модулів. Зручний для простих мереж, де дані проходять через шари один за одним.
* **Підкласи torch.nn.Module**: Більш гнучкий підхід, де користувач визначає власні класи, успадковуючи nn.Module. Це дозволяє створювати складні архітектури з розгалуженнями, пропусками з'єднань тощо.

**Рекурентні Нейронні Мережі (RNN)**

Рекурентні нейронні мережі (RNN) — це клас нейронних мереж, спеціально розроблений для роботи з послідовними даними, такими як текст, часові ряди або музика. На відміну від стандартних мереж прямого поширення, RNN мають "пам'ять" завдяки наявності рекурентних (зворотних) зв'язків, які дозволяють інформації з попередніх кроків впливати на обчислення на поточних кроках.

Основна ідея RNN полягає в тому, що вихід мережі на кроці (t) залежить не тільки від входу на кроці (t), але й від прихованого стану мережі на кроці (t-1). Прихований стан (h\_t) оновлюється на кожному кроці і слугує як зведена інформація про попередню історію послідовності.

Існують більш складні варіанти RNN, такі як LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit), які краще справляються з проблемою зникаючих/вибухаючих градієнтів і можуть ефективніше навчатись на довгих послідовностях.

**Генерація музики за допомогою RNN**

RNN можна використовувати для генерації нової музики. Один з підходів — це "символьна RNN" (character RNN), де мережа навчається передбачати наступний символ у музичному записі, представленому у вигляді текстової послідовності.

**ABC Notation**: Це формат текстового запису музики, який використовує літери (A-G) для нот, цифри для тривалості, та інші символи для позначення ритму, тональності, тактів тощо.

**Процес генерації**:

1. **Підготовка даних**: Музичний корпус у форматі ABC notation перетворюється на послідовність символів. Створюється словник унікальних символів.
2. **Побудова моделі**: Створюється RNN (наприклад, LSTM), яка приймає на вхід послідовність символів і намагається передбачити наступний символ.
3. **Тренування**: Модель тренується на великому корпусі музики, мінімізуючи функцію втрат (наприклад, перехресну ентропію) між передбаченими та реальними наступними символами.
4. **Генерація (Sampling)**: Після тренування модель використовується для генерації нової музики. Починаючи з початкової послідовності ("seed"), модель ітеративно передбачає наступний символ, додає його до послідовності і використовує оновлену послідовність як вхід для наступного кроку. Ймовірнісний розподіл виходу мережі дозволяє вносити випадковість у процес генерації.

# Завдання

1. Розібратись у процесі підготовки даних, побудови моделі RNN, її тренування та генерації музики.
2. Виконати завдання по ходу лабораторної роботи.
3. Поекспериментувати з параметрами процесу тренування та генерації.
4. Запустити процес тренування RNN та дочекатися його завершення (або виконати достатню кількість ітерацій для отримання робочої моделі).
5. Використати навчену модель для генерації щонайменше одного прикладу музичного твору у форматі ABC.
6. Проаналізувати результати, отримані на кожному етапі, та задокументувати їх у звіті.

# Індивідуальне завдання

Індивідуальне завдання полягає в експериментуванні з параметрами процесу тренування та генерації:

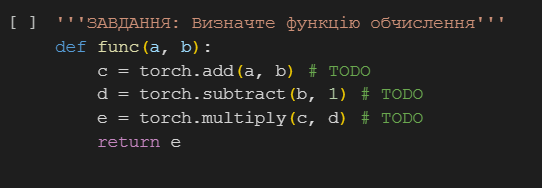
**Дослідження впливу гіперпараметрів**:

* Змініть розмір прихованого стану LSTM.
* Змініть швидкість навчання оптимізатора.
* Змініть довжину вхідної послідовності для тренування.

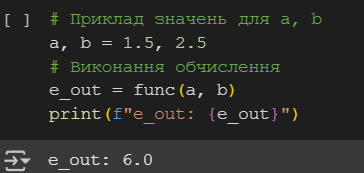
Проаналізуйте, як ці зміни впливають на швидкість навчання та суб'єктивну якість згенерованої музики.

# Хід роботи

1. **Вступ до PyTorch**

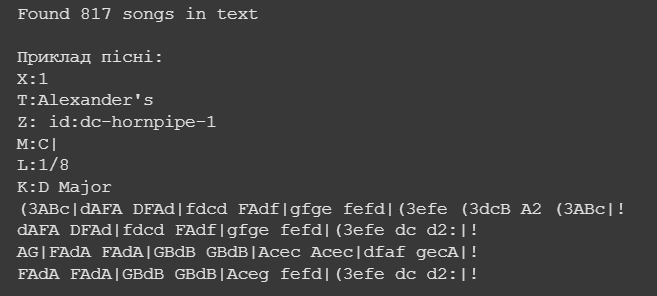


*Рис. 1. Функція обчислення*



*Рис. 2. Результат виконання функції*

1. **Генерація музики з RNN**

**

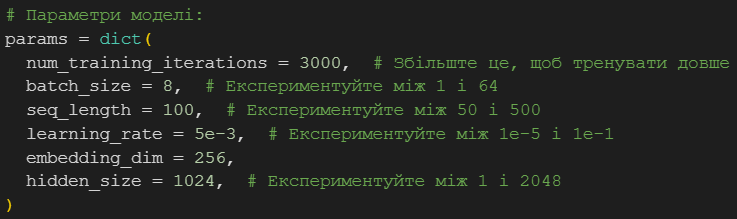
*Рис. 3. Приклад існуючої пісні у форматі ABC*



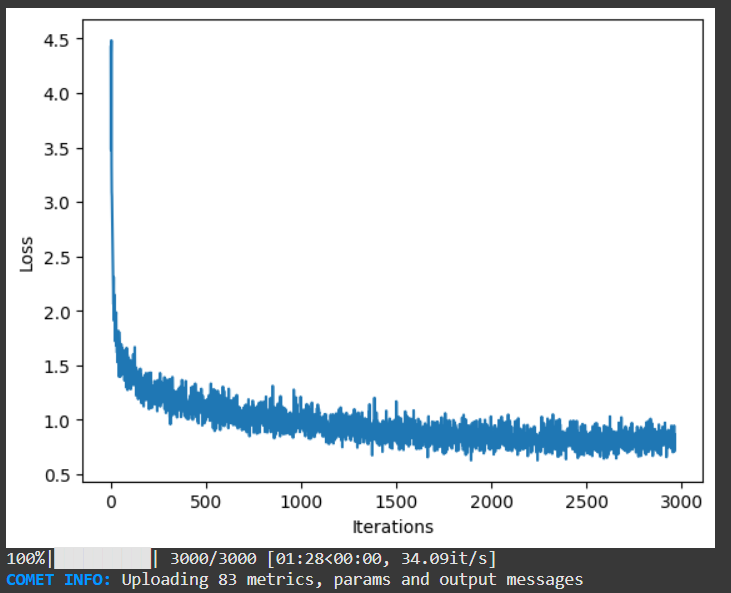
*Рис. 4. Результат ненатренованої моделі*



*Рис. 5. Втрати ненавченої моделі*



*Рис. 6. Використовуємо стандартні параметри*



*Рис. 7. Тренування моделі (успішне)*

**Текст, згенерований моделлю:**

X:107

T:Roillonneofer

Z: id:dc-reel-306

M:C

L:1/8

K:G Major

dGG2 dGBd|dGGG AGEG|gdBG AGEG|DGG2 EGD2:|!

X:109

T:Landonagey

Z: id:dc-reel-339

M:C

L:1/8

K:D Major

B|A3F ABdf|afed dcBA|FAA2 BAdB|A2FA ABde|f3d e2fa|edeg fdB|AFA d3|]!

X:hBrtha B,AFA|Bdd2 dfaf|efge fdBA|!

FAA2 BAFA|GBBA B2df|eBB2 eBB2|afef dBB2|A2FA dfed|!

cAA2 cAeA|fAA2 cdec|dcde fddB|Addc defe|!

d2fe dfed|Bdef g2ag|fgaf gfed|fga2 bage|!

X:43

T:Goss

Z: id the Town

Z: id:dc-slipjig-15

M:9/8

L:1/8

K:G Major

d3 e dB|A2 A2 G2|F2 A2 B2|c4 d2|B4 f2|ed cB Ag|!

f3 g g|af g2|d3 c B2|A4 G2|G4 F2|E4 D2|!

d4 F2|A2|G3 F G2|A2 B2 cB|A4 d2|c4 A2|B4 G2|A4 G2|G4 F4|!

FE F2 D2|G4 D2|G6|G4|]!

FE|D2 A2 F2 f2|G2 BA|D2 A2 F2|G3 B A/B/c/|!

Bc d2|cB B2 A2|B2 AB ce|f3 g|]!

X:155

T:Trippey's Pipers

Z: id:dc-reel-203

M:C

L:1/8

K:G Major

DGGF G2Bc|dGBG dGBd|eage d2Bd|efg2 d2BG|eBdB AGFG|]!

X:99

T:Maids of the Bridda

Z: id:dc-ocarolan-6

M:3/4

L:1/8

K:D Mixolydian

A|d^cd fdd|ede fde|fd^c d2A|dcA AGF|!

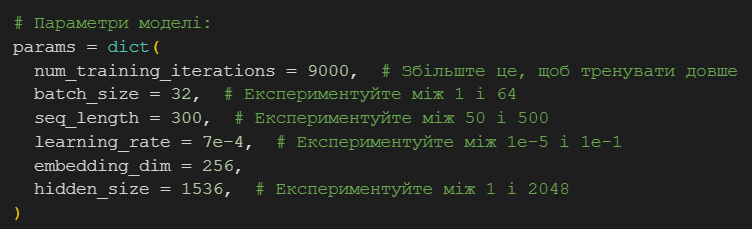
DGA BAG|Bdd Add|A3 efg|dBG FGA|!

Bee efg e2f|g2B

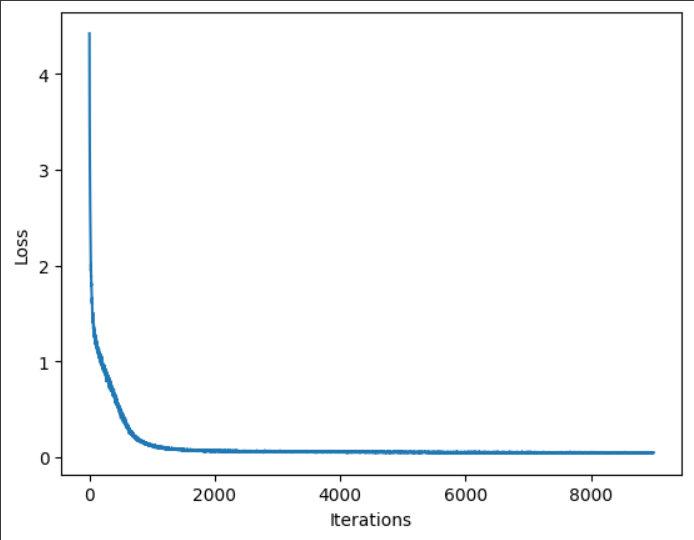
**Аудіо файл, згенерований натренованою моделлю:**

****

*Рис. 8.* *Аудіо файл, згенерований натренованою моделлю*

**

*Рис. 9. Використовуємо власні параметри*

******

*Рис. 10. Тренування моделі (успішне)*

**Тексти 5 мелодій, згенеровані моделлю:**

X:13

T:Kilfenora

Z: id:dc-slide-16

M:6/8

L:1/8

K:G Major

BcB BAB|G2B A2G|BcB BAG|A2B cBA|!

BcB BAB|G3 GBd|g2e dBG|A2B cBA:|!

g3 d3|ege d3|gfg aga|bab afd|!

g3 d3|ege d3|gfg aga|bgf g3:|!

X:16

T:Lock the Door

Z: id:dc-slide-17

M:6/8

L:1/8

K:E Dorian

B,|EDE BAG|FDF AFD|EDE BAB|AGF E2B,|!

EDE BAG|FDF AFA|d2B AGF|E3 E2:|!

A|B2c d2e|dcB AFA|B2c d2e|fdB fdA|!

B2c d2e|dcB AFA|d2B AGF|E3 E2:|!

X:17

T:Martin's Favourite

Z: id:dc-slide-37

M:6/8

L:1/8

K:G Major

d|g2d e2d|g2d e2d|BdB ABA|G2A BGE|!

D2E G2A|B2d dBG|BdB ABA|G3 G2:|!

E|D2E G2A|B2d dBG|BdB ABA|G2A BGE|!

D2E G2A|B2d dBG|BdB ABA|G3 G2:|!

X:18

T:Mary Willie's

Z: id:dc-slide-18

M:6/8

L:1/8

K:C Major

G|c2d e2d|cBc A3|f2G BAG|d2G BAG|!

c2d e2d|cBc A3|f2G BAG|c3 c2:|!

G|c2d e2g|age g2e|d2c d2g|fdg gfe|!

[1 Add gdB|Add edc|Bcd ecA|dfe d2:|!

X:139

T:Sixpenny Money

Z: id:dc-jig-117

M:6/8

L:1/8

K:D Major

fAA fAA|BAG FGE|DFA AFA|dfd ede|!

fAA fAA|BAG FGE|DFA AFA|dfd ede:|!

f3 g3|afd ecA|f3 g3|afd e3|!

f3 g3|afd ecA|DFA AFA|dfd ede:|!

**Аудіо файли, згенеровані натренованою моделлю:**

* ****
* ****
* ****
* ****
* ****

**Висновок:** Під час навчання RNN-моделі для генерації музики було проведено два основних тренування з різними гіперпараметрами. Перше тренування мало коротшу послідовність і менший обсяг даних за ітерацію, що дозволило швидко побачити базовий результат, але обмежило здатність моделі до складнішого структурування мелодії. Друге тренування використовувало глибшу архітектуру та більший обсяг даних, що потребувало більше часу, однак дало змогу моделі краще захопити повторювані патерни та варіативність у музиці.

Якість згенерованої музики суттєво покращилася після другого тренування. У першому випадку результатом стала одна мелодія, яка, хоча й звучала злагоджено, була досить монотонною і позбавлена розмаїття. Натомість друга спроба призвела до появи п’яти різних мелодій, кожна з яких мала структуровану форму — вступ, центральну частину з повторюваними елементами та завершення. Незважаючи на спільні мотиви, кожна композиція вирізнялася своїм характером, що, на мою думку, свідчить про успішніше навчання моделі.