Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Стінський Віталій Владиславович

студент групи АС-151

**КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА**

Програмне забезпечення для розпізнавання графічного зображення музичних нот методами машинного навчання

Спеціальність:

121 – Інженерія програмного забезпечення

Спеціалізація:

Інженерія програмного забезпечення

Керівник:

Комлева Наталія Олегівна,

к.т.н., доцент

Одеса – 2020

**ЗМІСТ**

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ………….…………………….4  
ЗАВДАННЯ НА РОЗРОБКУ РОЗДІЛУ «ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ»………………………………………………6  
[АНОТАЦІЯ 4](#_Toc58679028)

[ВСТУП 8](#_Toc58679029)

[1ОПИС ПРОБЛЕМ ІСНУЮЧИХ СИСТЕМ РОЗПІЗНАВАННЯ МУЗИЧНИХ НОТ………………………………………………………………………………...9](#_Toc58679030)

[1.1Опис аналогів програми для розпізнавання музичних нот 9](#_Toc58679031)

[1.2Порівняння аналогів із системою розпізнавання музичних нот та опис їх проблем. 9](#_Toc58679032)

[2РОЗРОБКА СИСТЕМИ РОЗПІЗНАВАННЯ МУЗИЧНИХ НОТ 11](#_Toc58679033)

[2.1Алгоритми нейронних мереж 11](#_Toc58679034)

[2.2Структури нейронних мереж 23](#_Toc58679035)

[2.3Алгоритм видання струни та номеру ладу визначеній ноті 29](#_Toc58679036)

[3СПЕЦИФІКАЦІЇ ВИМОГ ДО СИСТЕМИ РОЗПІЗНАВАННЯ МУЗИЧНИХ НОТ………………………………………………………………………………..32](#_Toc58679037)

[3.1Варіанти використання системи розпізнавання нот 32](#_Toc58679038)

[3.2Нефункціональні вимоги до системи розпізнавання нот 40](#_Toc58679039)

[4ПРОЕКТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ МУЗИЧНИХ НОТ 42](#_Toc58679040)

[4.1Проектування архітектури системи 42](#_Toc58679041)

[4.2Структура та організація класів 43](#_Toc58679042)

[5ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПІЗНАВАННЯ МУЗИЧНИХ НОТ……………………………………………………………………………….55](#_Toc58679043)

[5.1Особливості створення структури нейронної мережі відносно мови програмування Java. 55](#_Toc58679044)

[5.2Особливості алгоритму розпізнавання музичних рядів відносно мови програмування Java 57](#_Toc58679045)

[5.3Створення алгоритму розпізнавання музичних стовпціввідносно мови програмування Java 58](#_Toc58679046)

[5.4Створення алгоритму для розпізнавання наявності нот у музичних стовпцяхвідносно мови програмування Java 59](#_Toc58679047)

[5.5Створення алгоритму для розпізнавання пониження чи підвищення нотивідносно мови програмування Java…………………………………………….61](#_Toc58679048)

[5.6Створення алгоритму для розпізнавання нот та малювання табулатуривідносно мови програмування Java 62](#_Toc58679049)

[5.7Приклад використання програми 63](#_Toc58679050)

[6ВИЗНАЧЕННЯ ВЛАСТИВОСТЕЙ СИСТЕМИ РОЗПІЗНАВАННЯ МУЗИЧНИХ НОТ 65](#_Toc58679051)

[6.1Оцінювання точності результатів роботи системи розпізнавання музичних нот………………………………………………………………………………...65](#_Toc58679052)

[6.2Експериментальне визначення часу переходу від теорії до практичної гри музикантів – новачків 66](#_Toc58679053)

[7ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ 68](#_Toc58679054)

[ВИСНОВКИ 71](#_Toc58679055)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 72](#_Toc58679056)

[ДОДАТОК А. МАТЕРІАЛИ ТЕЗИСІВ 74](#_Toc58679057)

[ДОДАТОК Б. ЛІСТИНГ ПРОГРАМИ 77](#_Toc58679058)

[ДОДАТОК В. ПИТАННЯ З ОХОРОНИ ПРАЦІ ТА БЕЗПЕКИ У НАДЗВИЧАЙНИХ СИТУАЦІЯХ 91](#_Toc58679059)

[В.1 Аналіз умов праці згідно індивідуального завдання 91](#_Toc58679060)

[В.2 Охорона праці. Індивідуальне завдання. Розрахунок занулення на робочому місці 92](#_Toc58679061)

[В.3 Безпека у надзвичайних ситуаціях техногенного характеру. Індивідуальне завдання. Розрахунок по безпеці при надзвичайній ситуації – пожежа 94](#_Toc58679062)

[В.4 Заходи з охорони праці та безпеки у надзвичайних ситуаціях. 96](#_Toc58679063)

[В.5 Безпека у суспільстві в умовах загрози COVID – 19 96](#_Toc58679064)

|  |  |
| --- | --- |
|  | Міністерство освіти і науки України |
| Одеський національний політехнічний університет | |
| Навчально-науковий інститут комп’ютерних систем | |
| Кафедра системного програмного забезпечення | |
| Рівень вищої освіти: | другий (магістерський) |
| Спеціальність: | 121 – Інженерія програмного забезпечення |
| Спеціалізація: | Інженерія програмного забезпечення |

|  |
| --- |
| **ЗАТВЕРДЖУЮ** |
| **Завідувач кафедри** |
|  |
| « » 20 р. |

|  |
| --- |
| **ЗАВДАННЯ** |
| **НА КВАЛІФІКАЦІЙНУ РОБОТУ** |

Стінський Віталій Владиславович, група АС-151

|  |
| --- |
| 1. Тема роботи: Програмне забезпечення для розпізнавання графічного зображення музичних нот методами машинного навчання.   Керівник роботи: Комлева Наталія Олегівна, к.т.н., доцент |

затверджені наказом ректора від «29» жовтня 2020 р. № 412-в

1. Зміст роботи

Опис проблем існуючих систем розпізнавання музичних нот; розробка системи розпізнавання графічних нот; специфікації вимог до системи розпізнавання музичних нот; проектування системи розпізнавання музичних нот; програмна реалізація системи розпізнавання музичних нот; визначення властивостей системи розпізнавання музичних нот; охорона праці та безпека у надзвичайних ситуаціях.

1. Перелік ілюстративного матеріалу

Згідно слайдів презентації

1. Консультантирозділівроботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
| завдання видав | завдання  прийняв |
| ОП та БНС | Москалюк А.Ю., доц. |  | 10.11.2020 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачізавдання: « 1»вересня 20 20 р.

**КАЛЕНДАРНИЙПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів кваліфікаційної роботи | Строк  виконання | Примітка |
| 1 | Опис проблем існуючих систем розпізнавання музичних нот | 01.09 – 05.09 | Виконано |
| 2 | Розробка системи розпізнавання графічних нот | 06.09 – 01.10 | Виконано |
| 3 | Специфікації вимог до системи розпізнавання музичних нот | 02.10 – 09.10 | Виконано |
| 4 | Проектування системи розпізнавання музичних нот | 10.10 – 17.10 | Виконано |
| 5 | Програмна реалізація системи розпізнавання музичних нот | 18.10 – 20.11 | Виконано |
| 6 | Визначення властивостей системи розпізнавання музичних нот | 20.11 – 01.12 | Виконано |
| 7 | Охорона праці та безпека у надзвичайних ситуаціях | 02.12 – 04.12 | Виконано |
| 8 | Оформлення пояснювальної записки | 05.12 – 12.12 | Виконано |

|  |  |
| --- | --- |
| **Здобувач вищоїосвіти** | В.В. Стінський |
| **Керівникроботи** | Н. О. Комлева |

**ЗАВДАННЯ НА РОЗРОБКУ РОЗДІЛУ «ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ»**

|  |
| --- |
| Стінського Віталія Владиславовича, група АС-151 |
| Навчально-науковий інститут комп’ютерних систем |
| Кафедра системного програмного забезпечення |

Тема роботи: Програмне забезпечення для розпізнавання графічного зображення музичних нот методами машинного навчання.

Змістрозділу:

* 1. Організація та управління охороною праці на робочому місті інженера – програміста.
  2. Визначення основних параметрів умов праці робочого місця.
  3. Розрахунок занулення.
  4. Безпека під час надзвичайної ситуації техногенного характеру.
  5. Заходи з охорони праці та безпеки у надзвичайних ситуаціях.
  6. Безпека у суспільстві в умовах загрози COVID– 19.

|  |  |
| --- | --- |
| Керівник роботи | Консультант з охорони праці та БНС |
| Н. О. Комлева | А. Ю. Москалюк |
| « » 20 р. | « » 20 р. |

## АНОТАЦІЯ

Метою даної роботи є зниження часу на перехід починаючих музикантів від теорії до практики, шляхом розпізнавання нот та генеруванням для них табулатури за допомогою методів машинного навчання. Основою нейронної мережі став метод зворотного розповсюдження та метод на основі bruteforce. Методи розробки базуються на технологіях Java, Maven, ImageIO, JavaFX. Як результат роботи було виконане програмне забезпечення для розпізнання музичних нот із графічних зображень для гітари.

Ключові слова: програмування, розпізнавання музичних нот, нейронні мережі, генерація табулатури, Java, Maven, ImageIO, JavaFX.

**ABSTRACT**

The aim of this work is to accelerate the transition of novice musicians from theory to practice, by recognizing notes and generating tabs for them using machine learning methods. The basis of the neural network was the method of back propagation and the method based on brute force. Development methods are Java, Maven, ImageIO, JavaFX technologies. As a result, software for recognizing musical notes from graphic images was implemented.

Key words: programming, music note recognition, neural networks, tabulation generation, Java, Maven, ImageIO, JavaFX.

# ВСТУП

Значною частиною нашого життя являється музика, багато людей задумується про те, щоб не тільки слухати її але й грати. Купуючи музичний інструмент, люди одразу бажають перейти на етап гри, але далеко не у всіх є час на відвідування музичних занять щоб вивчити сольфеджіо та зрозуміти нотну грамоту.

Саме краще нотна грамота засвоюється на практиці, коли музикант бачить ноту, та її місце на ладі свого інструменту і може одразу ж її зіграти. Такий підхід працював при наявності вчителя музики, але в сьогоденні потрібен тільки комп’ютер. Саме такий підхід дає можливість швидко приступити від теорії та практики, та в процесі вивчити нотну грамоту.

Важливим елементом даного підходу являються нейронні мережі,які вміють розпізнавати музичні ряди та ноти з картинки й генерувати для них табулатуру, розпізнаючи ноти і вказуючи на якому ладу чи клавіші гратиметься нота.

Метою даної роботи являється зниження часу наперехід від теорії до гри на інструменті для музикантів – новачків за допомогою програми по розпізнаванню нот, яка створює для них табулатуру.

Для досягнення цієї мети розглянуто то вирішено наступні завдання:

* Розглянуто принципи роботи нейронних мереж[1].
* Розглянуто алгоритми функціонування нейронних мереж, а саме:
* Алгоритм взаємодії між нейронами та зв’язками.
* Функції нормалізації даних.
* Функції підрахунку помилки.
* Розглянуто алгоритми подачі вхідних даних на шар вхідних нейронів.
* Розглянуто принципи побудови графічного інтерфейсу[2].
* Виконано проектування та програмну реалізацію програми.
* Виконано тестування програми.

# ОПИС ПРОБЛЕМ ІСНУЮЧИХ СИСТЕМ РОЗПІЗНАВАННЯ МУЗИЧНИХ НОТ

# Опис аналогів програми для розпізнавання музичних нот

Опис предметної області

Предметною областю являються всі програми для графічного розпізнавання музичних нот.

Наприклад:

* AttenOrc
* MusicNotes

В даній предметній області виділяються такі критерії:

* Розпізнавання динамічного зображення – часто зображення, які потрібно розпізнати не мають чіткого формату розширення, тому дуже важливо щоб програма змогла адаптувати та розпізнати зображення.
* Розпізнавання повноцінного листка (А4) – важливо щоб програма могла розпізнати весь листок з нотами, а не по рядку. Це підвищує зручність використання.
* Вірогідність правильної відповіді - один із найважливіших параметрів, так як від нього залежить правильність гри музиканта.
* Генерування табулатури – для музикантів початківців дуже важливий цей пункт, тому що більшість з них не знає на якому ладі має гратись та чи інша нота, і завдяки генерації табулатури музиканти початківці одразу можуть приступити до гри та починають орієнтуватись на ладах набагато швидше.

# Порівняння аналогів із системою розпізнавання музичних нот та опис їх проблем

Існуючі рішення даної задачі описані в табл. 1.1.

Таблиця 1.1 - Порівняння якостей розробленої системи та її аналогів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Рішення | Розпізнавання динамічного зображення | Розпізнавання повноцінного листка (А4) | Вірогідність правильної відповіді | Генерування табулатури | Середня оцінка |
| NNTabs | 1 | 1 | 0.98 | 1 | 3.98 |
| AttenOrc | 0 | 0.5 | 0.95 | 1 | 2.45 |
| MusicNotes | 0 | 1 | 0.80 | 0 | 1.8 |
| Опис аналогів:   1. AttenOrc – программа для графічного розпізнавання нот, яка розроблювалась в рамках код-марафону у 2018 році. Мінусами даної програми являється:  * Неможливість розпізнавання різних розширень зображень. * Програма не може опрацьовувати не велику кількість музичних рядів (до 3х шт.). * Програма не вийшла в фінальну версію. | | | | |

1. MusicNotes – програма, яка була розроблена у 2015 році для розпізнавання музичних нот. Мінуси програми:

* Неможливість розпізнавання різних розширень зображень
* Відсутня генерація табулятури

# РОЗРОБКА СИСТЕМИ РОЗПІЗНАВАННЯ МУЗИЧНИХ НОТ

## Алгоритми нейронних мереж

2.1.1 Спрощений опис алгоритмів нейронних мереж:

Спрощений опис алгоритму першої нейронної мережі:

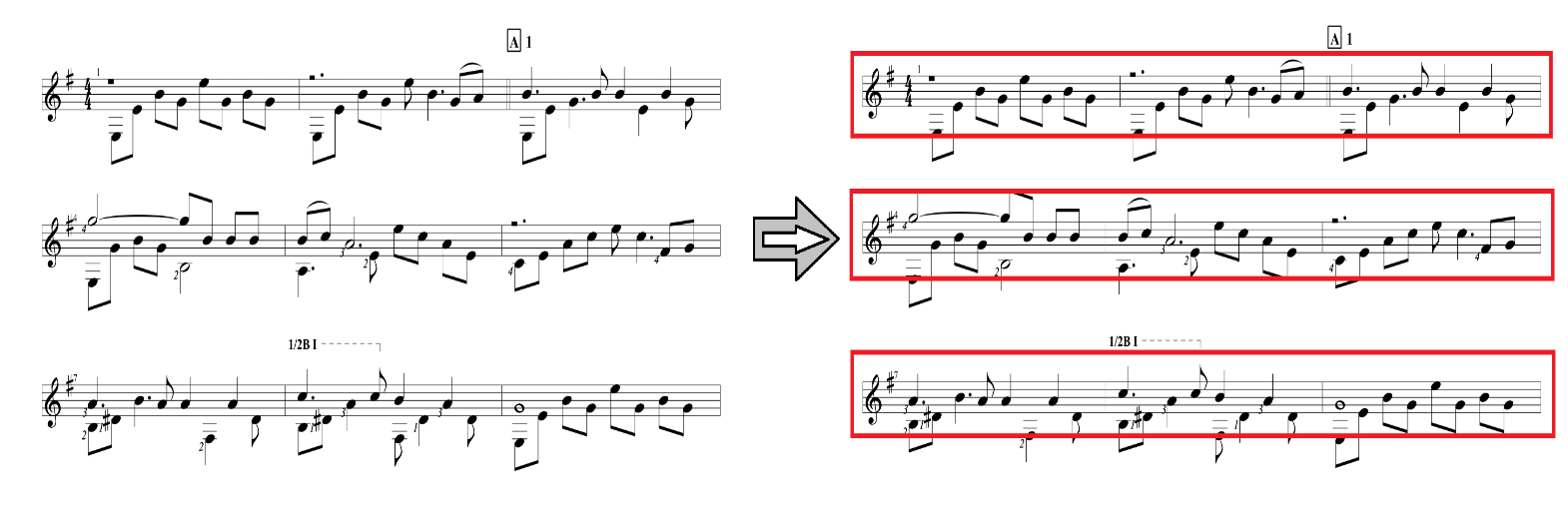
1. Для початку проводиться нормалізація картинки до розмірів 660x925.
2. Далі нейронна мережа по розпізнаванню музичних стовбців сканує отриману картинку, знаходить та записує координати музичних рядів, як це зображено на рис 2.1.

Рисунок. 2.1 – Візуалізація процесу знаходження музикальних рядів

1. Перша нейронна мережа передає записані координати другій нейронній мережі та закінчує свій алгоритм.

Спрощений опис алгоритму другої нейронної мережі:

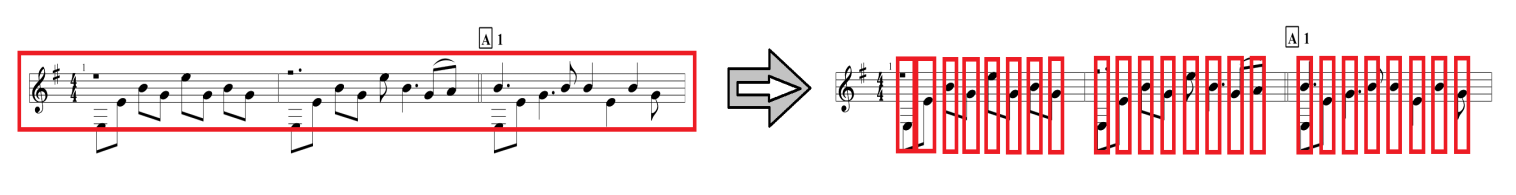
1.  Друга нейронна мережа аналізує дані отримані від першої нейронної мережі, позначає координати знайдених стовпців з нотами, як це зображено на рис 2.2.

Рисунок 2.2 – Процес знаходження нотних стовпців на музичних рядах

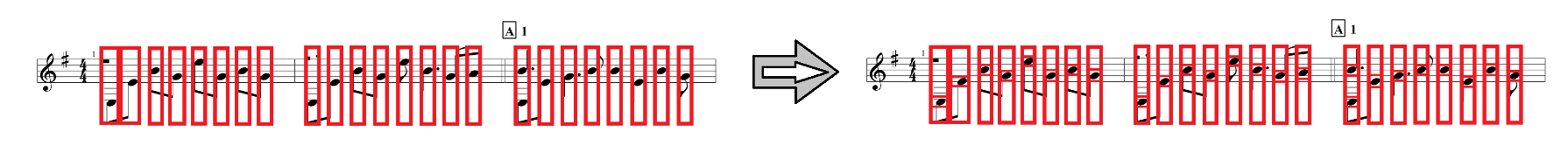
1. Третя нейронна мережа проходиться по кожному знайденому стовпцю та знаходить ноти на них, як зображено на рис. 2.3.

Рисунок 2.3 – Процес знаходження нот в нотних стовпцях

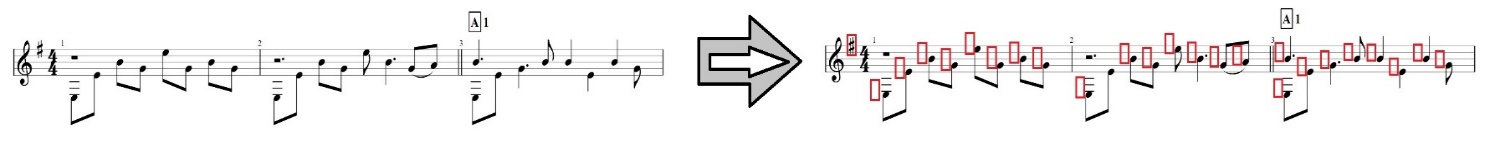
1. Четверта нейронна мережа проходиться по верхньому полі коло скрипічного ключа та по координатам кожної знайденої ноти з зсувом в 12 пікселів у ліву сторону та шукає, фіксуючи їх, знаки підвищення чи пониження ноти, як це зображено на рис 2.4.

Рисунок 2.4 – Процес знаходження підвищення чи пониження нот

1. Бот отримує данні про знайдені ноти, та генерує код відносно їх координат на музичному стані.
2. Після генерації коду бот генерує табулатуру, керуючись наступними правилами.

* Пріоритет розставляння в одному стовпці нот йде зверху вниз.
* Відсутність перекриття іншою нотою.
* Найнижчий лад.

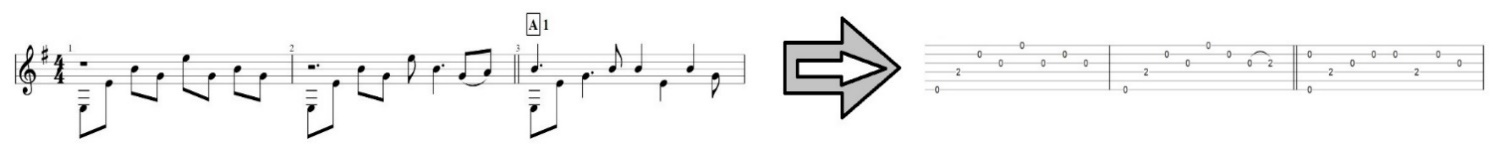
Фінальний варіант генерація табулатури зображений на рис. 2.5.

Рисунок 2.5 – Процес генерації табулатури

**2.1.2 Блок схеми та опис алгоритмів роботи нейронних мереж**

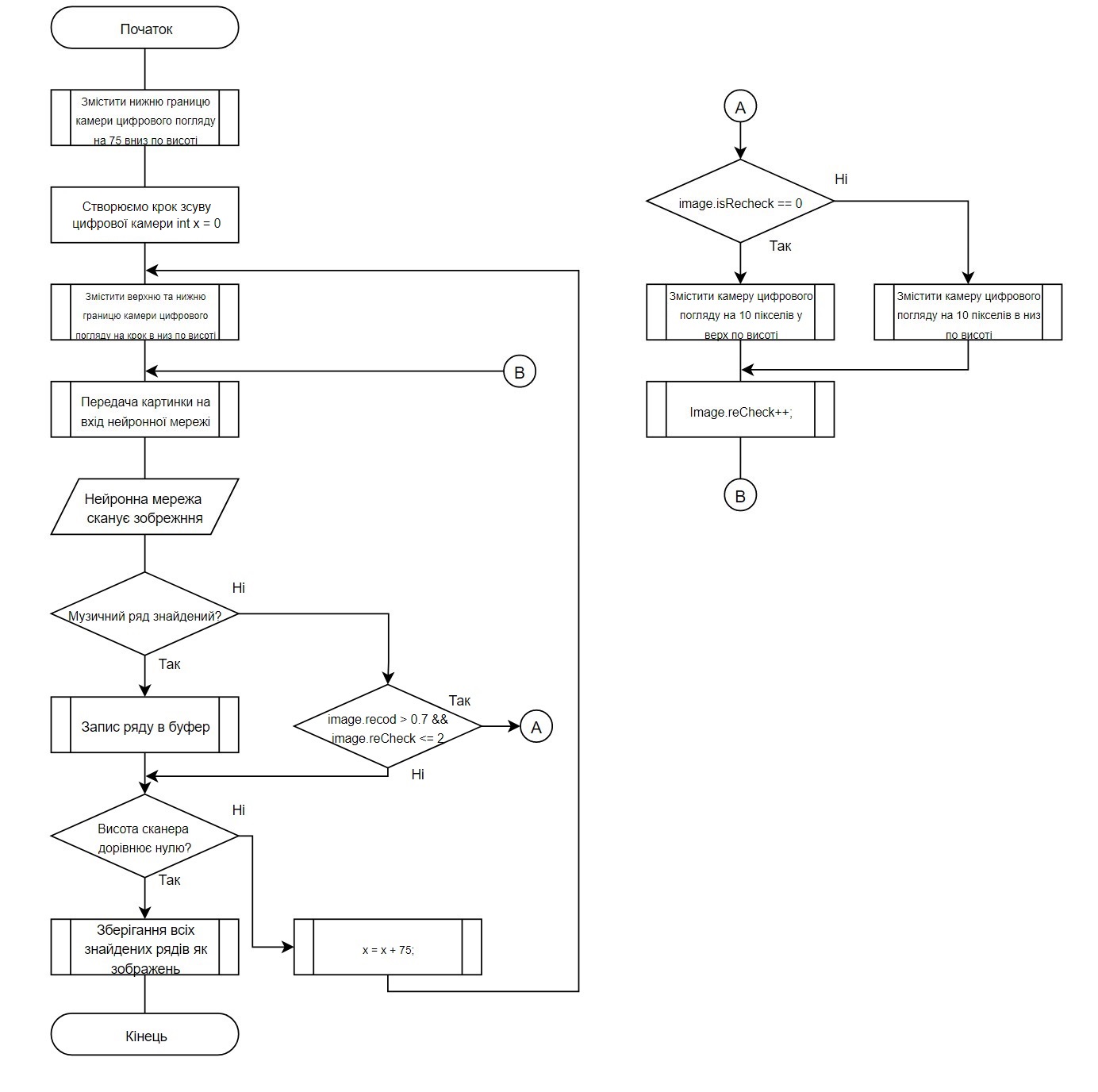
На рис. 2.6 зображена блок – схема алгоритму нейронної мережі, яка розпізнає музичні ряди:

Рисунок 2.6 – Блок схема алгоритму нейронної мережі, яка розпізнає музичні ряди

Опис блок схеми алгоритму нейронної мережі, яка розпізнає музичні ряди:

1. Камера цифрового погляду зменшується на 75 пікселів із нижньої границі (нейронна мережа розпізнає картинку кусками, тому потрібне таке зміщення).
2. Створюється крок зсуву цифрової камери х = 0.
3. Змістити верхню та нижню границю камери цифрового погляду на х низ по висоті.
4. Передається обрізанакартинка на вхід до нейронної мережі.
5. Нейронна мереже сканує зображення.
6. Музичний ряд знайдений? Якщо так, переходимо до пункту 13.
7. Рівень розпізнання зображення більший ніж 0.7 та картинка перевірялась менше 3х раз? Якщо ні, переходимо до пункту 14.
8. Картинка розпізнається перший раз? Якщо ні, пропускаємо пункт 9.
9. Зміщуємо камеру цифрового погляду на 10 пікселів у верх по висоті. Пропускаємо пункт 10.
10. Зміщуємо камеру цифрового погляду на 10 пікселів в низ по висоті.
11. Збільшуємо кількість разів розпізнавання картинки на 1.
12. Переходимо до пункту 4.
13. Запис координат музичного ряду
14. Висота сканеру дорівнює нулю? Якщо ні, переходимо до пункту 15.
15. х = х + 75 (крок зміщення камери). Перехід до пункту 3.
16. Зберігання всіх знайдених рядів як зображень.

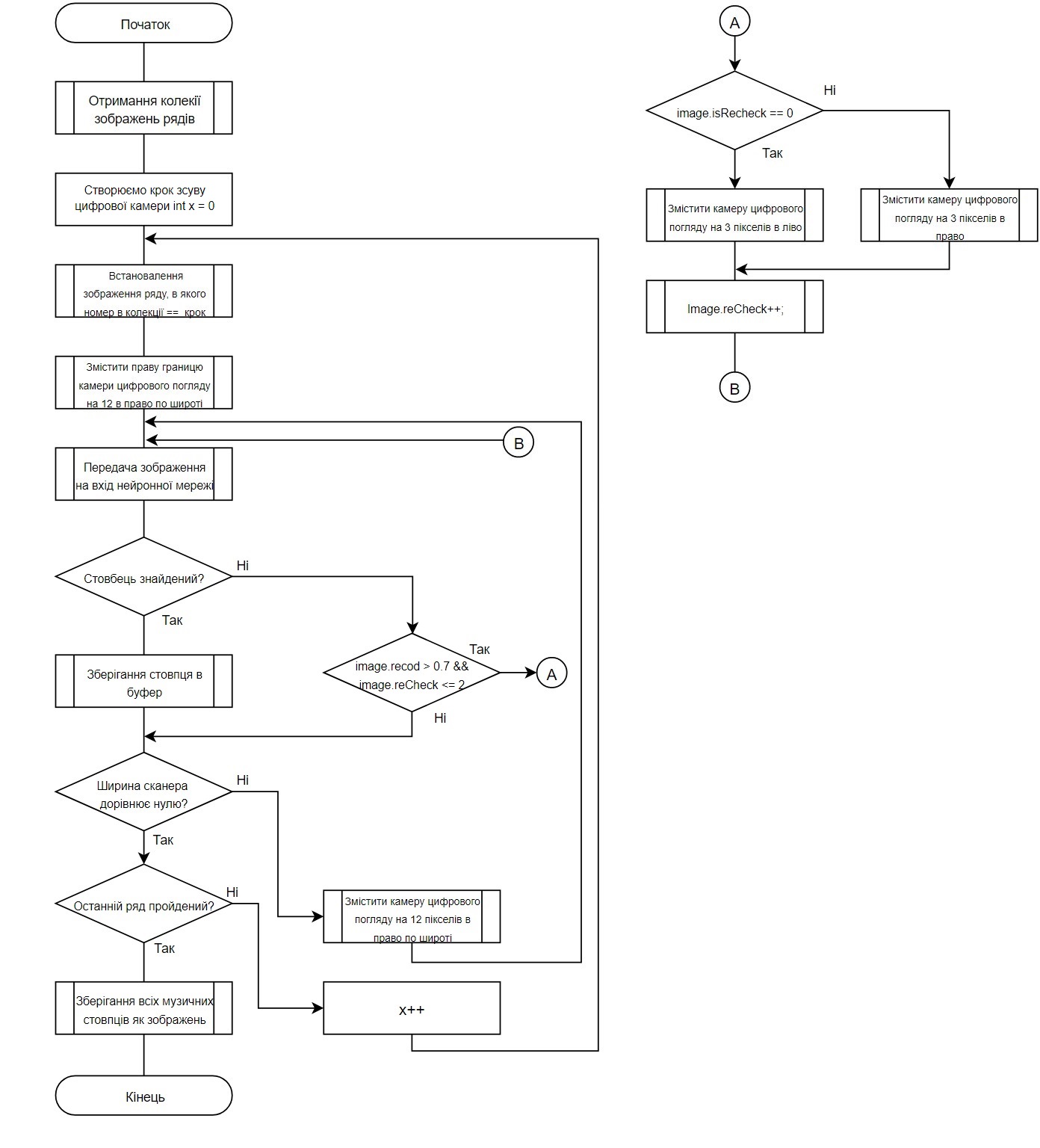
На рис. 2.7 зображена блок – схема алгоритму нейронної мережі, яка розпізнає музичні стовпці:

Рисунок 2.7 – Блок схема алгоритму нейронної мережі, яка розпізнає музичні стовпці

Опис блок схеми алгоритму нейронної мережі, яка розпізнає музичні стовпці.

1. Отримання колекції зображень рядів.
2. Створюється крок зсуву цифрової камерих = 0.
3. Встановлення зображення ряду, в якого номер в колекції дорівнює номеру кроку.
4. Змістити праву границю камери цифрового погляду на 12 пікселів в право по широті
5. Передача картинки на вхід нейронної мережі.
6. Музична стовпець знайдений? Якщо такпереходимо на пункт 13.
7. Рівень розпізнання зображення більший ніж 0.7 та картинка перевірялась менше 3х раз? Якщо ні, переходимо до пункту 14.
8. Картинка розпізнається перший раз? Якщо ні, пропускаємо пункт 9.
9. Зміщуємо камеру цифрового погляду на 3 пікселя ліво . Пропускаємо пункт 10.
10. Зміщуємо камеру цифрового погляду на 3 пікселя в право.
11. Збільшуємо кількість разів розпізнавання картинки на 1.
12. Переходимо до пункту 5.
13. Зберігання стовпця в буфер.
14. Ширина сканеру дорівнює нулю? Якщо ні переходимо на пункт 15.
15. Змістити камеру цифрового погляду на 12 пікселів в право по широті.Переходимо до пункту 5.
16. Останній ряд пройдений? Якщо ні переходимо до пункту 11.
17. Збільшуємо крок на 1 . Переходимо до пункту 3. (Здійснюється перехід до наступного ряду).
18. Зберігання всіх музичних стовпців як зображень.

На рис. 2.8 зображена блок – схема алгоритму нейронної мережі, яка розпізнає наявність музичних нот.

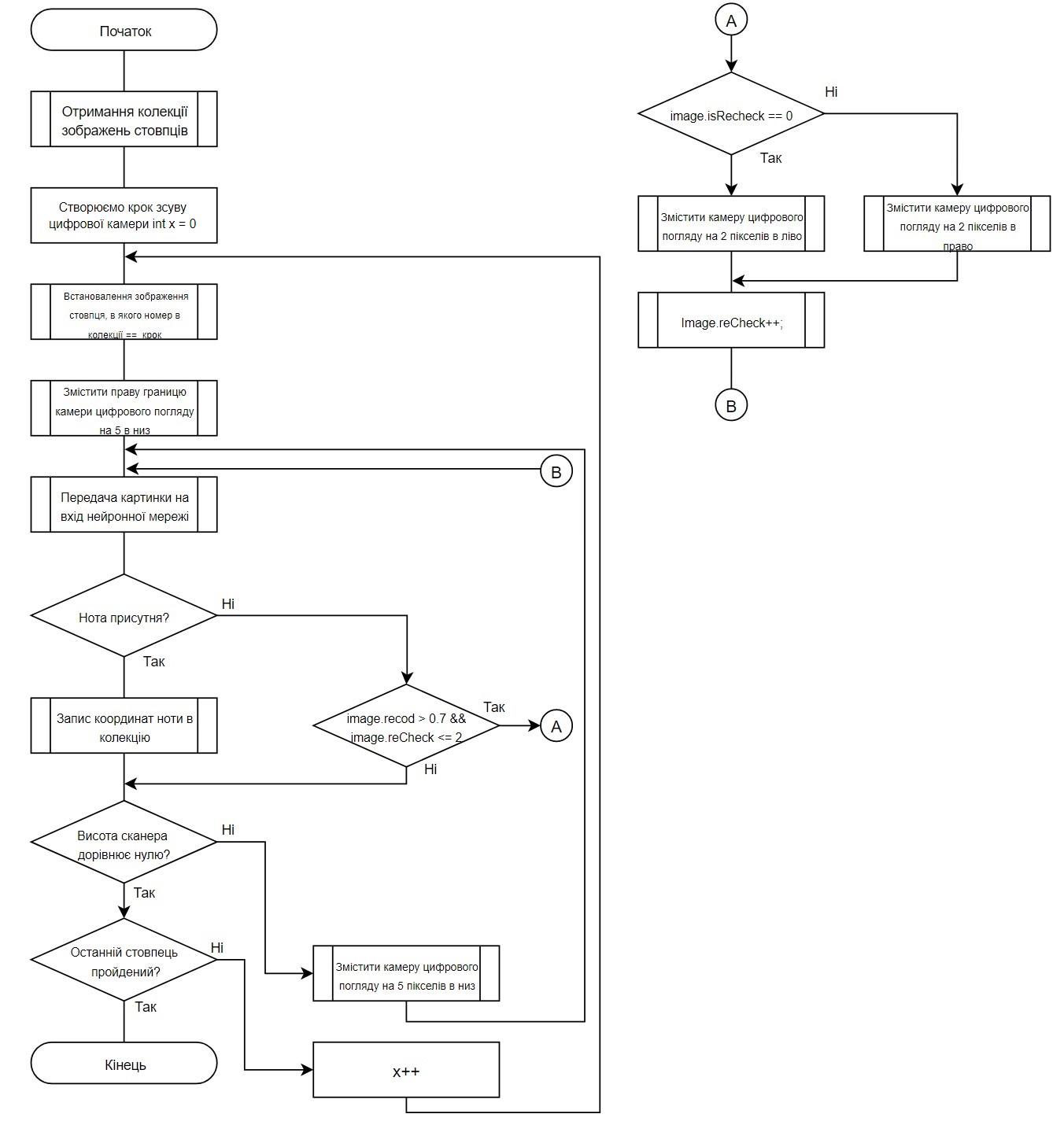


Рисунок 2.8 – Блок схема алгоритму нейронної мережі, яка розпізнає наявність музичних нот

Опис блок схеми алгоритму нейронної мережі, яка розпізнає наявність музичних нот:

1. Отримання колекції зображень стовпців.
2. Створюється кроку зсуву камери х = 0.
3. Встановлення зображення ряду, в якого номер в колекції дорівнює номеру кроку.
4. Змістити праву границю камери цифрового погляду на 5 пікселів низ.
5. Передача картинки на вхід нейронної мережі.
6. Нота присутня? Якщо так переходимо на пункт 13.
7. Рівень розпізнання зображення більший ніж 0.7 та картинка перевірялась менше 3х раз? Якщо ні, переходимо до пункту 14.
8. Картинка розпізнається перший раз? Якщо ні, пропускаємо пункт 9.
9. Зміщуємо камеру цифрового погляду на 2 пікселя в ліво . Пропускаємо пункт 10.
10. Зміщуємо камеру цифрового погляду на 2 пікселя в право.
11. Збільшуємо кількість разів розпізнавання картинки на 1.
12. Переходимо до пункту 5.
13. Запис координат ноти в колекцію.
14. Висота сканеру дорівнює нулю? Якщо ні переходимо на пункт 15.
15. Змістити камеру цифрового погляду на 5 пікселів в низ. Переходимо до пункту 5.
16. Останній стовпець пройдений? Якщо ні переходимо до пункту 17.
17. Збільшуємо крок на 1 . Переходимо до пункту 3. (Здійснюється перехід до наступного ряду).
18. Зберігання всіх музичних стовпців як зображень.

На рис. 2.9 зображена блок – схема алгоритму нейронної мережі, яка розпізнає наявність пониження чи підвищення ноти.

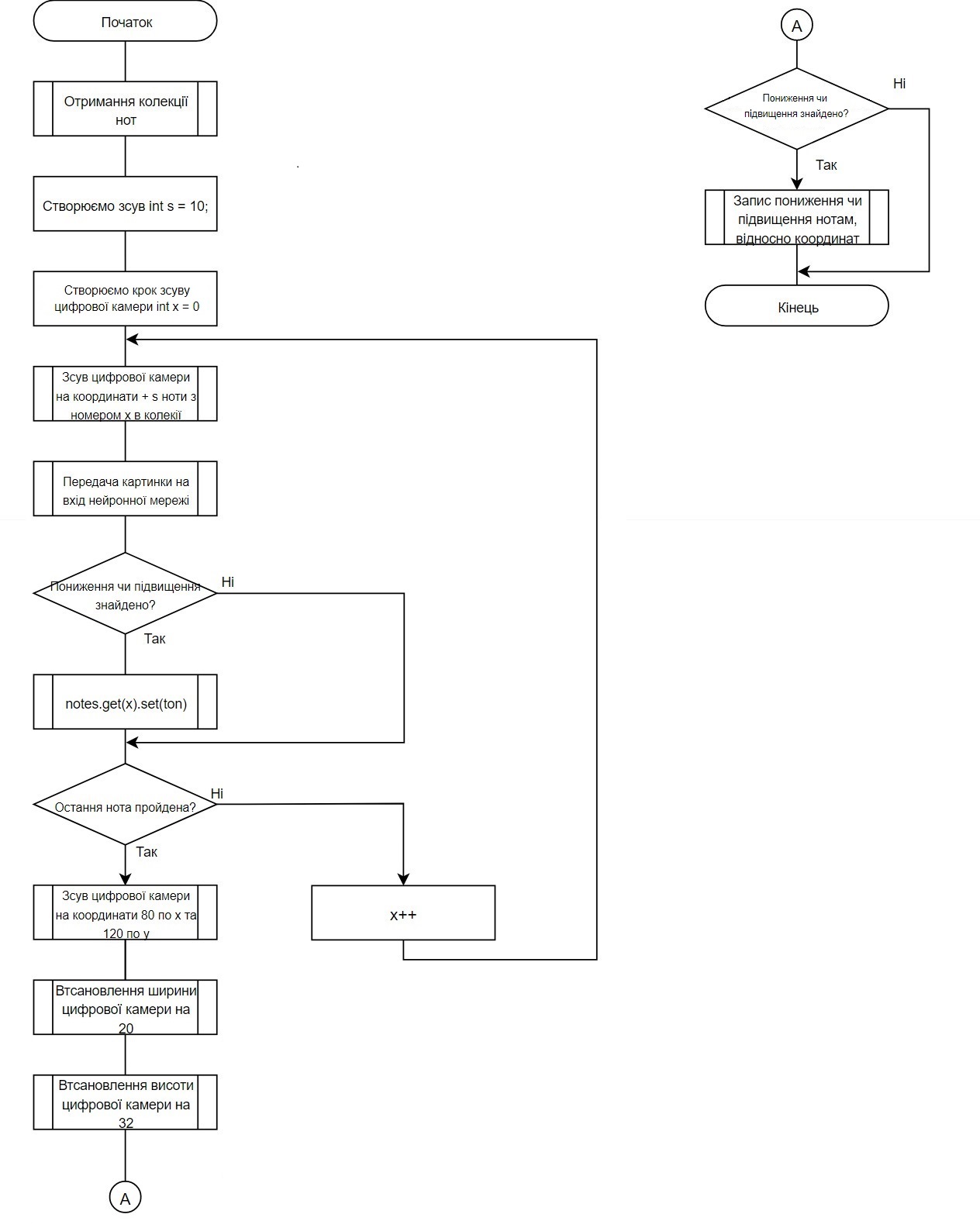


Рисунок 2.9 – Блок схема алгоритму нейронної мережі, яка розпізнає наявність пониження чи підвищення ноти

Опис блок схеми алгоритму нейронної мережі, яка розпізнає наявність пониження чи підвищення ноти:

1. Отримання колекції нот.
2. Створюємо зсувs = 0, який потрібен щоб зсунути цифрову камеру на координати вказання пониження чи підвищення ноти.
3. Створюється кроку зсуву цифрової камери х = 0.
4. Зсув цифрової камери на координати ноти з номером кроку в колекції + зсув s.
5. Передача картинки на вхід нейронної мережі.
6. Пониження чи підвищення знайдено? Якщо ні, пропускаємо пункт 7.
7. Запис пониження чи підвищення ноті із номером в колекції який дорівнює кроку х.
8. Останнянота пройдена? Якщо ні переходимо до пункту 9.
9. Збільшуємо крок на 1 . Переходимо до пункту 4. (Здійснюється перехід до наступного ряду).
10. Зсув цифрової камери на координати 80 по х та 120 по у.
11. Встановлення ширини цифрової камери на 20.
12. Встановлення висоти цифрової камери на 32.
13. Пониження чи підвищення нот знайдено? Якщо так, переходимо на пункт 14.
14. Запис пониження чи підвищення нотам, відносно координат.

Блок схема алгоритму бота, який генерує код із розпізнаних нот, та малює табулатуру:

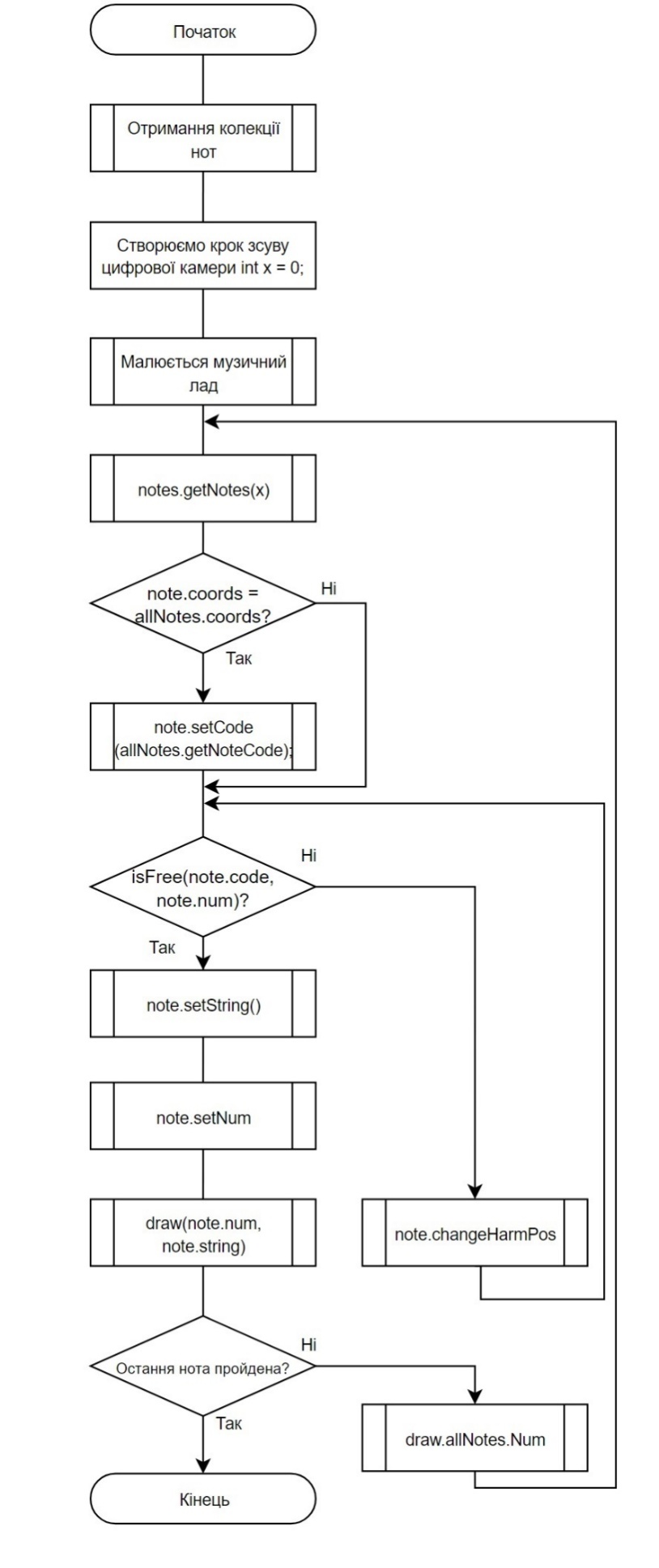


Рисунок 2.10 – Блок схема алгоритму бота, який генерує код із розпізнаних нот, та малює табулатуру

Опис блок схеми алгоритму бота, який генерує код із розпізнаних нот, та малює табулатуру.

1. Отримання колекції нот, які були розпізнані за допомогоючотирьох нейронних мереж, і в яких записано код ноти.
2. Створюється кроку зсуву цифрової камери х = 0.
3. Малюється музичний лад, на якому буде малюватись табулатура для розпізнаних нот.
4. Встановлення на розпізнавання ноти, в якої номер в колекції дорівнює номеру кроку х.
5. Координати ноти знаходяться в колекції координат всіх можливих нот? Якщо ні, пропускаємо пункт 6.
6. Запис ноті коду ноти з так такими ж координатами.
7. Можлива струна на якій грається нота зайнята? Якщо ні, пропускаємо пункт 8.
8. Змінити тон ноти. Переходимо до пункту 7.
9. Задати струну ноті.
10. Записати лад струні.
11. Намалювати ноту.
12. Остання нота пройдена? Якщо ні переходимо до пункту 4.

Після закінчення виконання алгоритму, який генерує код із розпізнаних нот, та малює табулатуру. Згенерована по розпізнаним нотам табулатура виводиться на екран комп’ютера користувача.

Щоб розпізнати інші ноти, потрібно завантажити їх в програму та запустити розпізнавання ще раз.

## Структури нейронних мереж

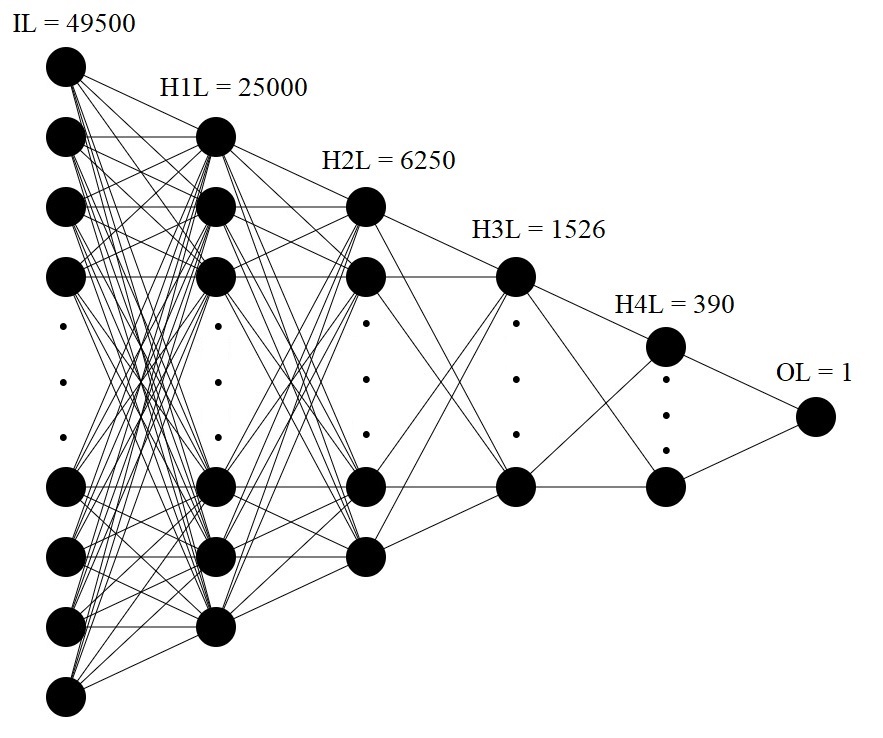
Розглянемо структуру нейронної мережі для розпізнавання рядів[3], яка зображена на рис. 2.11.

Рисунок 2.11 – Структура нейронної мережі для розпізнавання рядів

Структуранейронної мережі для розпізнавання рядів складається із:

* OL – вхідний шар, у якому 49500 нейронів.
* H1L – схований перший шар, у якому 25000 нейронів.
* H2L – схований другий шар, у якому 6250 нейронів.
* H3L – схований третій шар, у якому 1526 нейронів.
* H4L – схований четвертий шар, у якому 390 нейронів.
* OL–вихідний шар, у якому один нейрон.

Розглянемо структуру нейронної мережі для розпізнавання стовпців, яка зображена на рис 2.12.

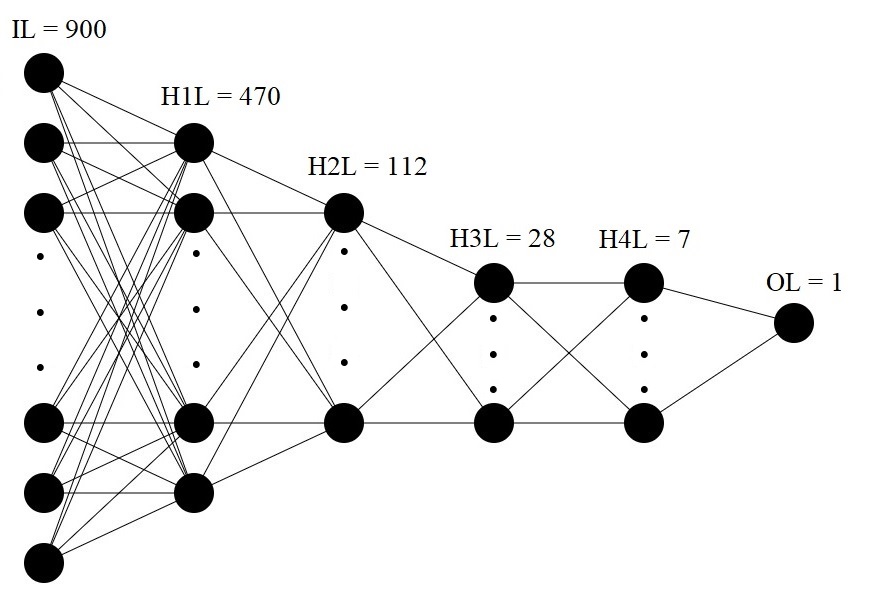


Рисунок 2.12 – нейронної мережі для розпізнавання стовпців

Структура нейронної мережі для розпізнавання стовпців складається із:

* OL – вхідний шар, у якому 900 нейронів.
* H1L – схований перший шар, у якому 470 нейронів.
* H2L – схований другий шар, у якому 112 нейронів.
* H3L – схований третій шар, у якому 28 нейронів.
* H4L – схований четвертий шар, у якому 7 нейронів.
* OL–вихідний шар, у якому один нейрон.

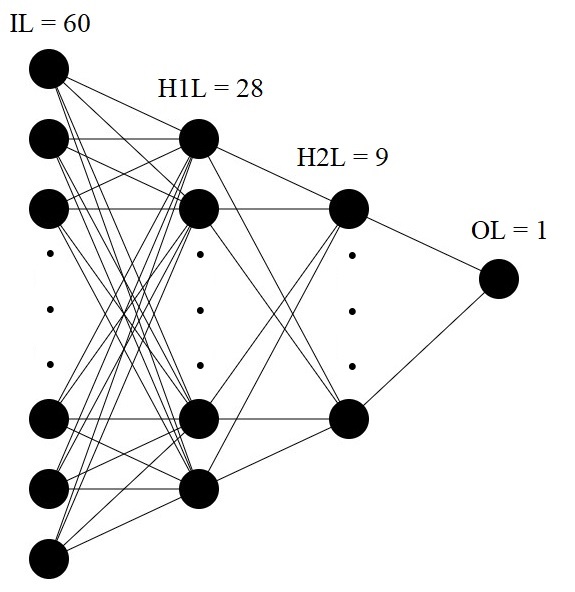
Розглянемо структуру нейронної мережі для розпізнавання наявності нот. 

Рисунок 2.13 – Структура нейронної мережі для розпізнавання наявності нот

Структура нейронної мережі для розпізнавання наявності нот складається із:

* OL – вхідний шар, у якому 60 нейронів.
* H1L – схований перший шар, у якому 28 нейронів.
* H2L – схований другий шар, у якому 9 нейронів.
* OL– вихідний шар, у якому один нейрон.

Розглянемо структуру нейронної мережі для розпізнавання наявності пониження чи підвищення ноти:

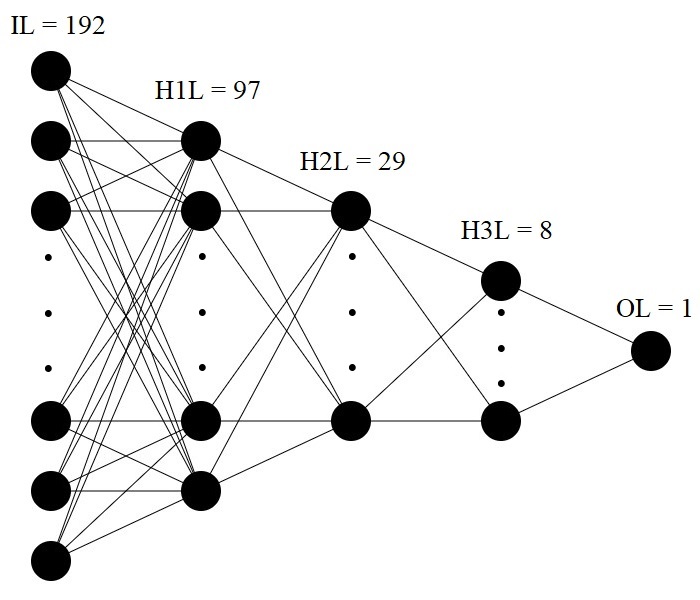


Рисунок 2.14 – Структура нейронної мережі для розпізнавання наявності пониження чи підвищення ноти

Структура нейронної мережі для розпізнавання наявності пониження чи підвищення ноти складається із:

* OL – вхідний шар, у якому 192 нейрона.
* H1L – схований перший шар, у якому 97 нейронів.
* H2L – схований другий шар, у якому 29 нейронів.
* H3L – схований третій шар, у якому 8 нейронів.
* OL–вихідний шар, у якому один нейрон.

Розглянемо структуру нейронних мереж з точки зору математики [4].

Підрахунок ваги нейронів та зв’язків між ними проходить по формулі 2.1.

Де H – нейрон прихованого слою, – функція активації,I – нейрон вхідного слою, w – вага.

Існують такі функції активації:

* Лінійна функція – використовується коли потрібно протестувати нейронну мережу або передати дані без їх перетворень. Зображена лінійна функція на рис. 2.15.

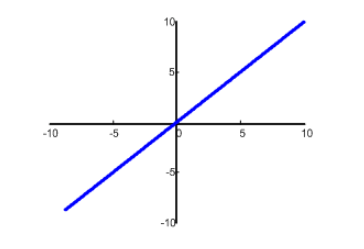


Рисунок 2.15 –Графік лінійної функції

* Сигмоїда – функція для нормалізації даних в діапазоні значень [0;1]. Зображена сигмоїда на рис 2.16.

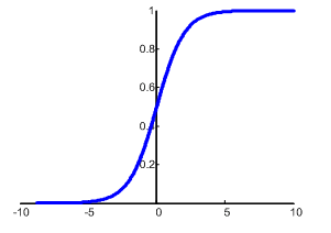


Рисунок 2.16 – графік функції сигмоїди

* Гіперболічний тангенс – використовується коли вхідні дані можуть заходити в міновий діапазон. Зображений гіперболічний тангенс на рис. 2.17.

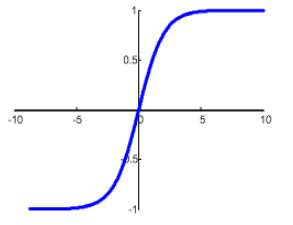


Рисунок 2.17 – графік функції гіперболічного тангенсу

Функція активації для чотирьох нейронних мереж – сигмоїда[5], одна з найрозповсюдженіший функцій. Вибір сигмоїди зумовлений тим, що на вхід нейронних мереж поступає чорно – біла картинка, пікселі якої мають діапазон яскравості [0; 255], тобто нам потрібно охоплювати тільки плюсовий діапазон чисел. Сигмоїда зображена на формулі 2.2.

Існують такі функції підрахування помилки нейронних мереж [6]:

* Функція MSE зображена на формулі 2.3.
* Функція Root MSE зображена на формулі 2.4.
* Функція Arctan зображена на формулі 2.5.

Функція виявлення похибки для обох нейронних мереж – MSE. Серед функцій конкурентів, а саме RootMSE, яка понижує похибку, та Arctan, яка підвищує похибку, саме MSE тримає золоту середину та найймовірнішу похибку. ФункціяMSE зображена на формулі 2.6.

Методом навчання нейронних мереж обрано метод зворотного поширення [7]. Розглянемо алгоритм даного методу:

1. Підраховуємо δ (дельту) для кожного із нейронів за формулою 2.7.

Для вихідних нейронів, так як в них немає вихідних зв’язків формула 2.8, де – бажаний результат, – отриманий результат, похідна.

Для нейронів прихованого слою формула 2.9, де – сума ваг, помножених на отримані дельти.

Де OUT – вихідне значення нейрону.

Розрахунок похідної.

1. Міняємо ваги для всіх зв’язків, які виходять із нейронів, на основі вирахуваної δ
2. Шукаємо градієнт для кожного зв’язку за формулою 2.10:

Де точка А – початок зв’язку, точка В – кінець зв’язку.

1. Проводимо розрахунок та оновлення ваг для зв’язків за формулою 2.11:

Де E – швидкість навчання, – момент, – попередня вага зв’язку.

## Алгоритм видання струнита номеру ладу визначеній ноті

Ноти длягітари можна позначити таким чином:

Таблиця 2.1 – Можливі тони та їх місце гри для ноти Мі

|  |  |  |
| --- | --- | --- |
| Тони ноти | Лад тону | Струна |
| 0 | 0 | 6 |
| 1 | 12 | 6 |
| 1 | 7 | 5 |
| 1 | 2 | 4 |
| 2 | 9 | 3 |
| 2 | 5 | 2 |
| 2 | 0 | 1 |
| 3 | 11 | 1 |

Таблиця 2.2 – Можливі тони та їх місце гри для ноти Фа

|  |  |  |
| --- | --- | --- |
| Тони ноти | Лад тону | Струна |
| 0 | 1 | 6 |
| 1 | 8 | 5 |
| 1 | 3 | 4 |
| 2 | 10 | 3 |
| 2 | 6 | 2 |
| 2 | 1 | 1 |
| 3 | 1 | 12 |

Таблиця 2.3 – Можливі тони та їх місце гри для ноти Соль

|  |  |  |
| --- | --- | --- |
| Тони ноти | Лад тону | Струна |
| 0 | 3 | 6 |
| 1 | 10 | 5 |
| 1 | 5 | 4 |
| 1 | 0 | 3 |
| 2 | 12 | 3 |

Продовження таблиці 2.3

|  |  |  |
| --- | --- | --- |
| Тони ноти | Лад тону | Струна |
| 2 | 8 | 2 |
| 2 | 1 | 3 |

Таблиця 2.4 – Можливі тони та їх місце гри для ноти Ля

|  |  |  |
| --- | --- | --- |
| Тони ноти | Лад тону | Струна |
| 0 | 5 | 6 |
| 0 | 0 | 5 |
| 1 | 12 | 5 |
| 1 | 7 | 4 |
| 1 | 2 | 3 |
| 2 | 10 | 2 |
| 2 | 5 | 1 |

Таблиця 2.5 – Можливі тони та їх місце гри для ноти Сі

|  |  |  |
| --- | --- | --- |
| Тони ноти | Лад тону | Струна |
| 0 | 7 | 6 |
| 0 | 2 | 5 |
| 1 | 9 | 4 |
| 1 | 4 | 3 |
| 1 | 0 | 2 |
| 2 | 12 | 2 |
| 2 | 7 | 1 |

Таблиця 2.6 – Можливі тони та їх місце гри для ноти До

|  |  |  |
| --- | --- | --- |
| Тони ноти | Лад тону | Струна |
| 0 | 8 | 6 |
| 0 | 3 | 5 |

Продовження таблиці 2.6

|  |  |  |
| --- | --- | --- |
| Тони ноти | Лад тону | Струна |
| 1 | 10 | 4 |
| 1 | 5 | 3 |
| 1 | 1 | 2 |
| 2 | 10 | 1 |

Таблиця 2.7 – Можливі тони та їх місце гри для ноти Ре

|  |  |  |
| --- | --- | --- |
| Тони ноти | Лад тону | Струна |
| 0 | 10 | 6 |
| 0 | 5 | 5 |
| 0 | 3 | 4 |
| 1 | 12 | 4 |
| 2 | 7 | 3 |
| 2 | 3 | 2 |
| 2 | 11 | 1 |

Щоб видати струну та номер ладу ноті, після генерації коду потрібно йти за наступним алгоритмом:

1. Пріоритет розпізнавання нот, якщо в стовбцю знайдено більше одної ноти, йде з верху в низ.
2. Визначити всі можливі місця гри ноти, відносно її тону, на інструменті (всі можливі струни й лади на них).
3. Відсортувати по всі знайдені позиції гри ноти по ладам у порядку зростання.
4. Відбувається цикл по знайденим позиціям. Вибирається менший лад, в якого не має перекриття.

# СПЕЦИФІКАЦІЇВИМОГ ДО СИСТЕМИ РОЗПІЗНАВАННЯ МУЗИЧНИХ НОТ

# Варіанти використання системи розпізнавання нот

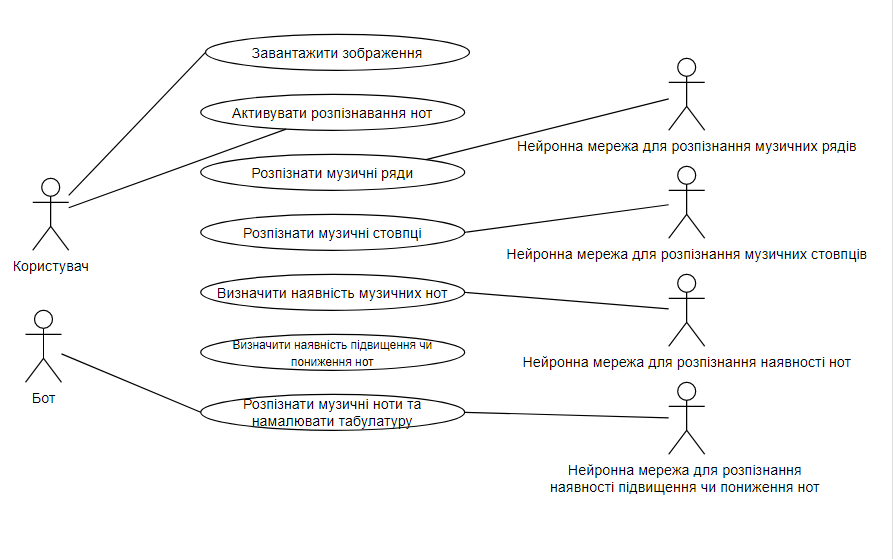
1.  Опис діаграм прецедентів зображені на рис. 3.1.

Рисунок 3.1 – Опис діаграми прецедентів

1. Опис сценаріїв використаннята будування схем життєвого циклу до них:
2. Завантажити зображення:
3. Користувач натискає кнопку «Завантажити зображення». Програма відкриває файлову систему.
4. Користувач обирає бажане зображення. Програма аналізує розмірність зображення.
5. Програма підтверджує розмірність зображення та завантажує його.

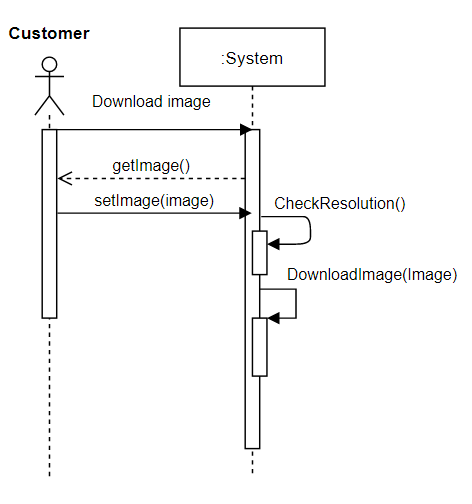
1.1) Схема життєвого циклу для прецеденту «Завантажити зображення»зображена на рис 3.2.

Рисунок 3.2 – Схема життєвого циклу для прецеденту «Завантажити зображення»

1. Активувати розпізнавання нот:
2. Користувач натискає кнопку «Розпізнати ноти». Система перевіряє наявність завантаженого зображення.
3. Система підтверджує наявність зображення та запускає прецедент «Знайти музичні ряди».

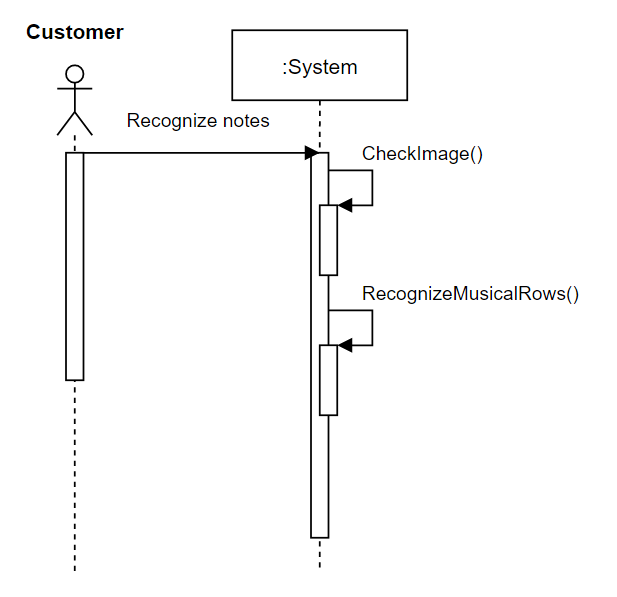
2.1) Схема життєвого циклу для прецеденту «Розпізнати ноти» зображена на рис 3.3.

Рисунок 3.3 – Схема життєвого циклу для прецеденту «Розпізнати ноти»

1. Знайти музичні ряди:
2. Програма подає завантажене зображення на вхід Нейронної мережі для розпізнавання музичних рядів (НМ1). НМ1 завантажує зображення.
3. НМ1 аналізує зображення проходячись по ньому зверху вниз. HM1 знаходить музичні ряди.
4. НМ1 зберігає музичні ряди я зображення та дає сигнал про це Системі. Система запускає прецедент «Знайти музичні стовпці».

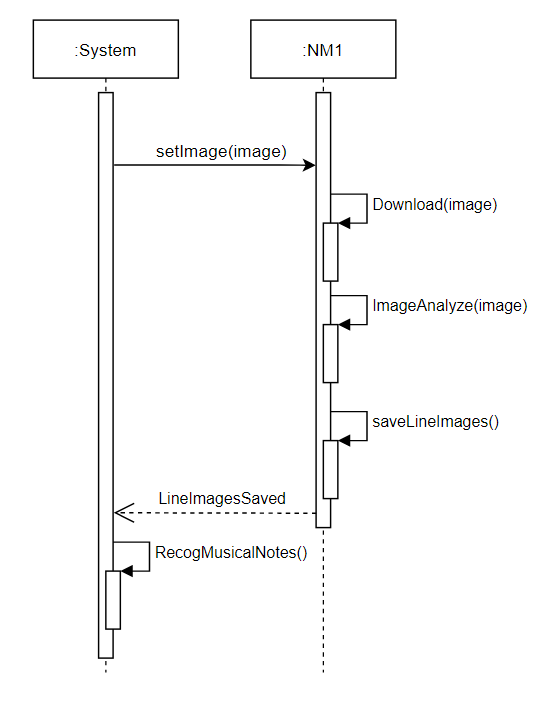
3.1) Схема життєвого циклу для прецеденту «Знайти музичні ряди» зображена на рис 3.4.

Рисунок 3.4 – Схема життєвого циклу для прецеденту «Знайти музичні ряди»

1. Знайти музичні стовпці:
2. НМ1 передає зображення музичних рядів Нейронній мережі для розпізнавання музичних стовпців (НМ2). НМ2 завантажує зображення зі музичними рядами.
3. НМ2 проходиться по зображенням музичних рядів. НМ2 знаходить музичні стовпці.
4. НМ2 зберігає музичніряди як зображення, та запускає варіант використання «Визначити наявність музичних нот».

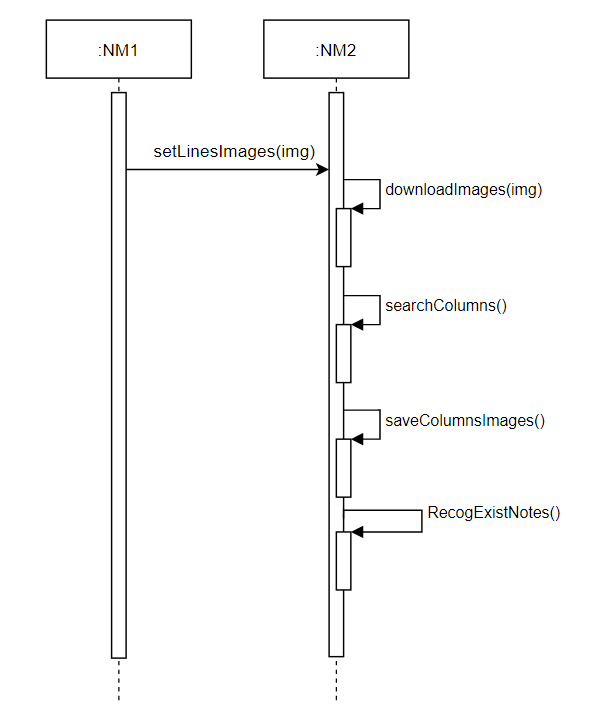
4.1) Схема життєвого циклу для прецеденту «Знайти музичні стовпці» зображена на рис. 3.5.

Рисунок 3.5 – Схема життєвого циклу для прецеденту «Знайти музичні стовпці»

1. Визначити наявність музичних нот:
   * + 1. НМ2 передає зображення музичних стовпців Нейронній мережі для розпізнавання наявності музичних нот (НМ3). НМ3 завантажує зображення зі музичними стовпцями.
       2. НМ3 проходиться по зображенням музичних стовпців. НМ3 знаходить музичні ноти.
       3. НМ3 зберігає координати музичних нот, та запускає варіант використання «Визначення наявності підвищення чи пониження нот».

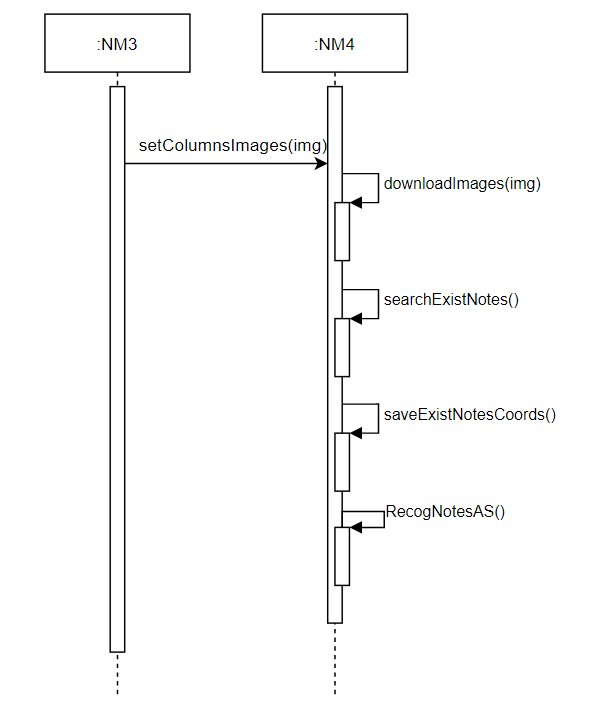
5.1) Схема життєвого циклу для прецеденту «Визначити наявність музичних нот» зображена на рис. 3.6.

Рисунок 3.6 – Схема життєвого циклу для прецеденту «Визначити наявність музичних нот»

1. Визначити наявність підвищення чи пониження нот:
   * + 1. НМ3 передає координати музичних нот Нейронній мережі для розпізнавання наявності підвищення чи пониження музичних нот (НМ4). НМ4 завантажує координати із музичними нотами.
       2. НМ4 проходиться координатам музичних нот плюс зсув. НМ4 знаходить музичні ноти.
       3. НМ4 зберігає пониження чи підвищення ноти відносно координат, та запускає варіант використання «Розпізнати музичні ноти»

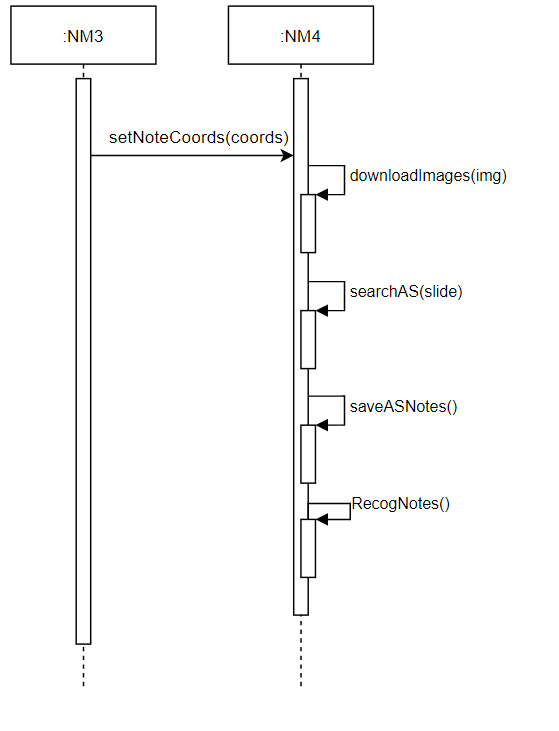
6.1) Схема життєвого циклу для прецеденту «Визначити наявність підвищення чи пониження нот» зображена на рис. 3.7.

Рисунок 3.7 – Схема життєвого циклу для прецеденту «Визначити наявність підвищення чи пониження нот»

1. Розпізнати музичні ноти та намалювати табулатуру:
   * + 1. НМ4 передає координати музичнихвідносно музичного ладу Ботові для розпізнавання нот та малювання табулатури.Бот завантажує координати із музичними нотами.
       2. Бот проходиться координатам музичних нот. Бот порівнює координати розпізнаних музичних нот із координатами всіх можливих музичних нот.
       3. Бот знаходить спільні координатирозпізнаних музичних нот. Бот генерує розпізнаним музичним нотам код, відносно їх координат.
       4. Бот проходиться по розпізнаним нотам та видає їм номера (лад, на якому грається нота)то номер струни, відносно їх координат, наявності пониження та підвищення та відсутності чи наявності перекриття та зручності гри.
       5. Бот малює лінії табулатури. Бот малює цифри розпізнаних нот відносно координат виданих їм струн.

7.1) Схема життєвого циклу для прецеденту «Розпізнати музичні ноти та намалювати табулатуру» зображена на рис. 3.8.

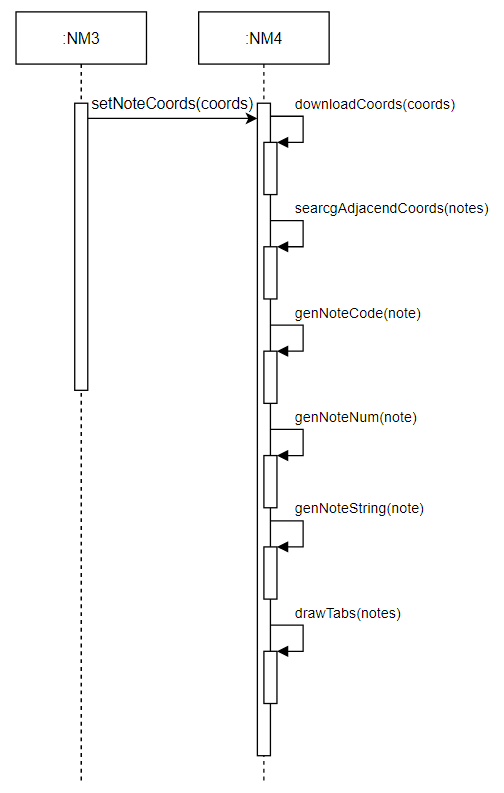


Рисунок 3.8 – Схема життєвого циклу для прецеденту «Розпізнати музичні ноти та намалювати табулатуру»

Альтернативні сценарії

* 1. Завантажити зображення

1. Користувач натискає кнопку «Завантажити зображення». Програма відкриває файлову систему.
2. Користувач обирає бажане зображення. Програма аналізує розмірність зображення.
3. Програма не підтверджує розмірність зображення та виводить помилку «Занадто мала розмірність зображення».
4. Користувач обирає інше зображення. Програма аналізує розмірність зображення.
5. Програма підтверджує розмірність зображення та завантажує його.

3.1) Знайти музичні ряди

1. Програма подає завантажене зображення на вхід Нейронної мережі 1 (НМ1). НМ1 завантажує зображення.
2. НМ1 аналізує зображення проходячись по ньому зверху вниз. HM1 знаходить музичні ряди.
3. НМ1 не фіксує координати музичних рядів, виводить помилку «На даному зображені не має нот, виберіть інше зображення». Запускається прецедент «Завантажити зображення»
4. Специфікація вимог користувача

Основний успішний сценарій:

1. Користувач натискає кнопку «Завантажити зображення», обирає зображення. Система підтверджує розмірність зображення та завантажує його.
2. Користувач натискає кнопку «Розпізнати ноти». Система передає зображення НМ1 та запускає прецедент «Розпізнати музичні ряди».
3. НМ1 отримує зображення. НМ1 знаходить музичні ряди, зберігає їх як зображення та передає їх НМ2. Запускається прецедент «Розпізнати музичні стовпці».
4. НМ2 отримує зображення музичних рядів, та знаходить по ним музичні стовпці, та зберігає їх як зображення. НМ2 передає зображення музичних стовпців НМ3 та запускає прецедент «Визначити наявність музичних нот».
5. НМ3 отримує зображення музичних стовпців, та знаходить по ним ноти, зберігає координати. НМ3 передає координати нот НМ4 та запускає прецедент «Визначити наявність підвищення чи пониження нот».
6. НМ4 отримує координати нот, визначає наявність пониження чи підвищення, та записує його відносно координат на музичному ладу. НМ4 запускає прецедент «Розпізнати музичні ноти та намалювати табулатуру»
7. Бот отримує розпізнані ноти відносно їх координат код (номер ладу на якому вони граються). Наоснові колу створює табулатуру, малює її та виводить на екран.

# 3.2 Нефункціональні вимоги до системи розпізнавання нот

1. Інтерфейсами для користувача слугує:

* Меню
* Вікно вибору зображення
* Вікно з відображенням вибраного зображення
* Вікно відображення табулатури

Данні інтерфейси виконані в класичному стилі JavaFX.

1. Послідовність полів елементів:

* Меню
* Вікно вибору зображення
* Вікно відображення табулатури

1. Керування

Керування проводиться за допомогою кнопки на мишці «leftclick». Сповіщення відображаються по центру програми.

1. Атрибути якості:

* Максимальний час відклику програми за запит користувача менше 0.05 секунди.
* Максимальний час оброблення зображення не більше 3 секунди.
* Максимальний час малювання табулатури не більше 1 секунду.

# ПРОЕКТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ МУЗИЧНИХ НОТ

# Проектування архітектури системи

За архітектурну основу був взятий шаблон MVC, який здебільшого використовується для написання програм з графічним інтерфейсом.

Короткий опис шаблону:

* M –Model. В даній частині зберігається «тіло системи»
* V – View. Дана частина зберігає функціонал для відображення графічного інтерфейсу.
* C–Control – відповідає за керування системою

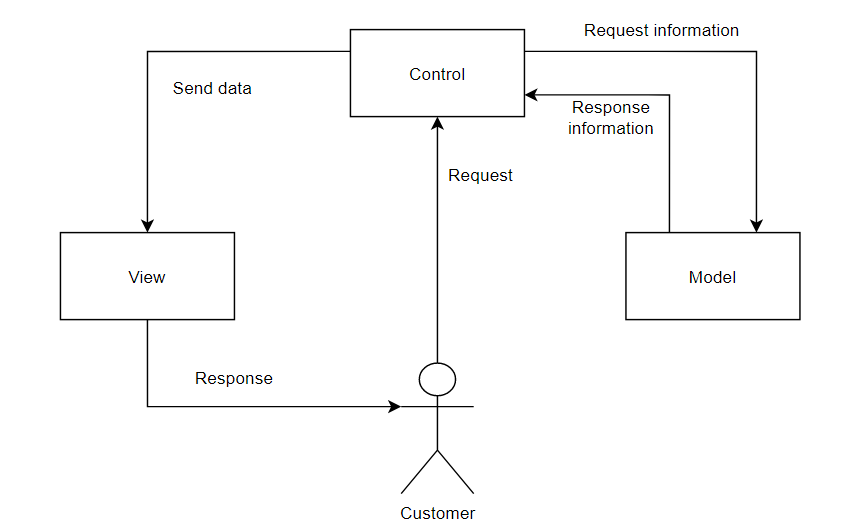
Структура системи з шаблоном MVC[8]зображена на рис. 4.1.

Рисунок 4.1 – Архітектура системи із шаблоном MVC

# Структура та організація класів

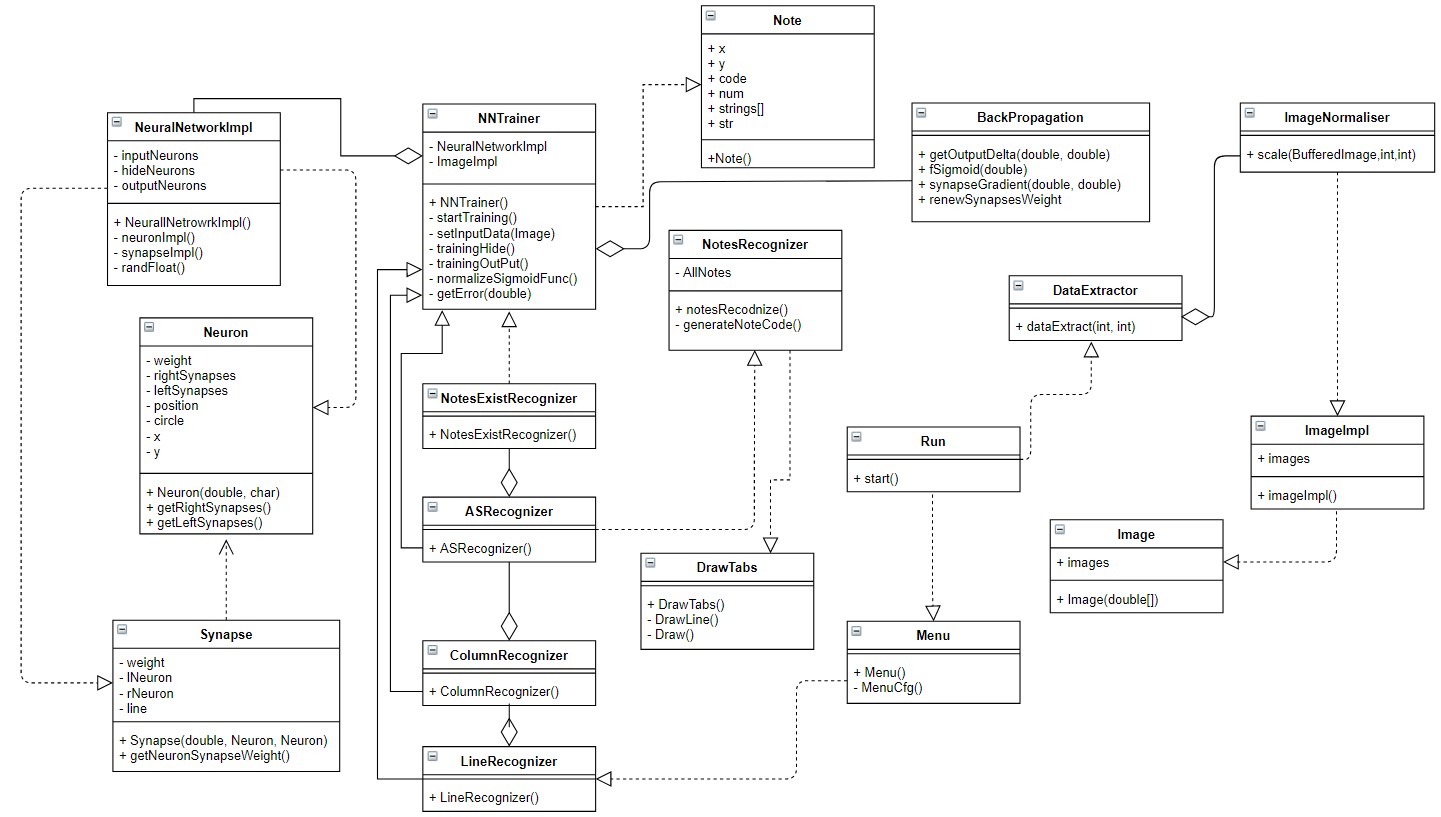
Діаграма реалізації класів, яка описує структуру проекту в стилі UML[9]представлена на рис 4.2.

Рисунок 4.2 – Діаграма організації класів

Опис діаграми реалізації класів[10].

1. Neuron

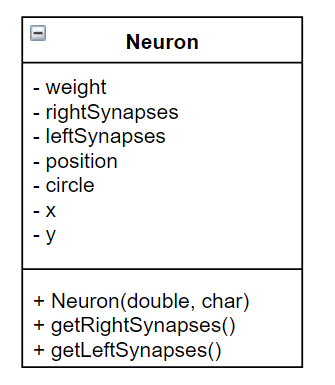
Поля класу зображені на рис 4.3.

Рисунок 4.3 – Поля класу Neuron

Опис полів класу:

Методи:

* Neuron(double,char) – конструктор класу.
* getRightSynapses() – отримати всі зв’язки справа.
* getLeftSynapses() – отримати всі зв’язки зліва.

Змінні:

* weight – вага нейрону.
* rightSynapses- зв’язки зліва.
* leftSynapses - зв’язки справа.
* Position – позиція нейрона в шарах нейронної мережі.
* Circle – радіус нейрону.
* x – позиція нейрону по осі х.
* y – позиція нейрону по осі у.

Роль класу:

Описує сутність об’єкту «Нейрон»

1. Synapse

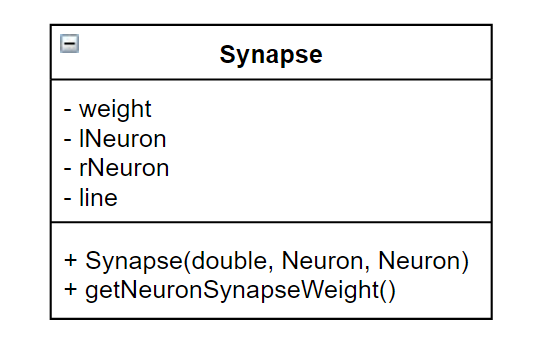
Поля класу зображені на рис. 4.4

Рисунок 4.4 – Поля класу Synapse

Опис полів класу:

Методи:

* Synapse(double, Neuron, Neuron) – конструктор класу.
* getNeuronSynapseWeight – зміна ваги нейрону та зв’язку.

Змінні:

* weight – вага зв’язку.
* lNeuron – нейронзліва.
* rNeuron – нейрон справа.
* line – товщина лінії зв’язку

Роль класу:

Описує сутність об’єкту «Зв’язок»

1. Menu

Поля класу зображені на рис. 4.5.

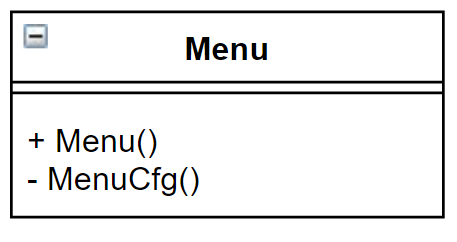


Рисунок 4.5 – Поля класу Menu

Опис полів класу:

Методи:

* Menu() – малювання інтерфейсу програми.
* MenuCfg() – налаштування меню.

Роль класу:

Малює графічний інтерфейс для користувача.

1. NeuralNetowrkImpl

Поля класу зображені на рис. 4.6.

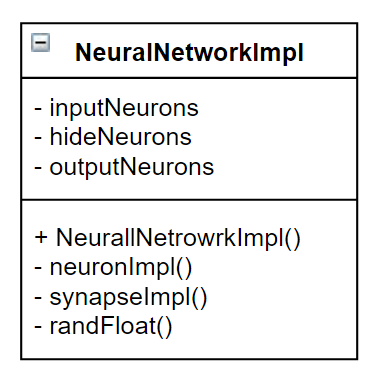


Рисунок 4.6 – Поля класу NeuralNetworkImpl

Опис полів класу:

Методи:

* NeuralNetworkImpl() – конструктор класу.
* neuronImpl() – створення нейронів.
* synapseImpl() – створення зв’язків між нейронами.
* randFloat() – метод, що задає випадкові значення вагам нейронів та зв’язків при їх створенні.

Змінні:

* inputNeurons – шар вхіднихнейронів
* hideNeurons – шар схованих нейронів
* outputNeurons – шар вихідних нейронів

Роль класу:

Створює нейронну мережу, реалізуючи нейрони та зв’язки між ними.

1. NNTrainer

Поля класу зображені на рис. 4.7.

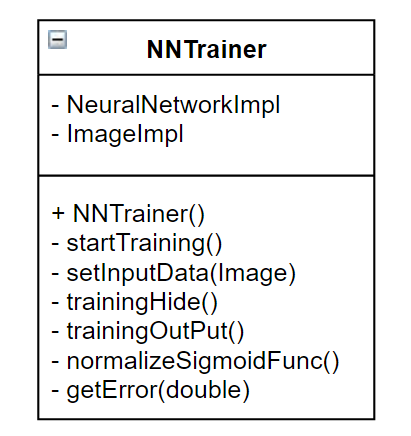


Рисунок 4.7 – Поля класу NNTrainer

Опис полів класу:

Методи:

* NNTrainer() – конструктор класу.
* startTraining() – запускає тренування нейронної мережі.
* setInputData(Image) – метод для завантаження вхідних даних для вхідного шару нейронної мережі.
* trainingHide() – тренування шару схованих нейронів.
* trainingOutPut() – тренування шару вихідних нейронів.
* normalizeSigmoidFunc() – функція нормалізації даних.
* getError() – функція для отримання похибки нейронної мережі.

Змінні:

* NeuralNetworkImpl – реалізує нейронну мережу.
* ImageImpl – реалізує конвертацію зображення у вхідні дані.

Роль класу:

Реалізує та тренує нейронну мережу.

1. BackPropagation

Поля класу зображені на рис. 4.8.

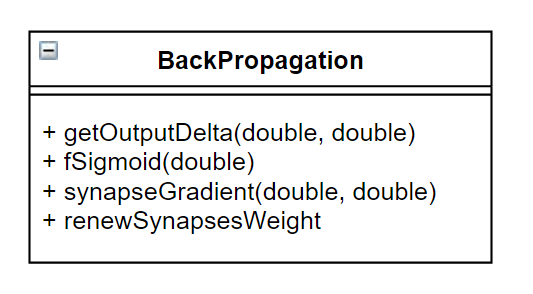


Рисунок 4.8 – Поля класу BackPropagation

Опис полів класу:

Методи:

* getOutputDelta(double, double) – вираховує дельту для зв’язків.
* fSigmoid(double) – адаптує дельту під функцію активації.
* synapseGradient(double, double) – вираховує зміщення ваг зв’язків.
* renewSynapseWeight – зміщує ваги зв’язків між нейронами.

Роль класу:

Виконує роль «серця» нейронної мережі та здійснює перерахунок ваг в середині неї.

1. Image

Поля класу зображені на рис. 4.9.

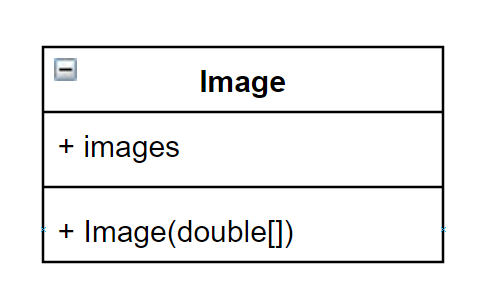


Рисунок 4.9 – Поля класу Image

Опис полів класу:

Методи:

* Image(double[]) – конструктор класу.

Змінні:

* Images – вхідні зображення.

Роль класу:

Описує сутність класу «Зображення»

1. ImageImpl

Поля класу зображені на рис. 4.10.

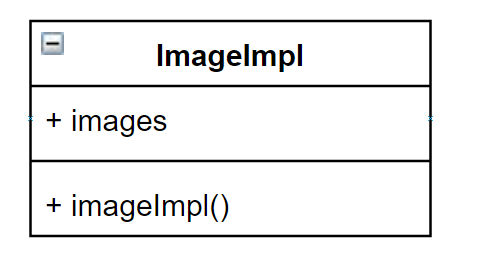


Рисунок 4.10 – Поля класу ImageImpl

Опис полів класу:

Методи:

* ImageImpl() – конструктор класу.

Змінні:

* images – вхідні зображення.

Роль класу:

Реалізує клас зображення.

1. DataExtractor

Поля класу зображені на рис. 4.11.

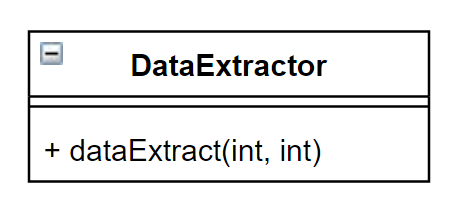


Рисунок 4.11 – Поля класу DataExtractor

Опис полів класу:

Методи:

* dataExtract(int, int) – завантажує вхідні дані із вказаного місця на жорсткому диску комп’ютера.

Роль класу:

Виконує роль завантажувача даних для нейронної мереж.

1. ImageNormaliser

Поля класу зображені на рис. 4.12.

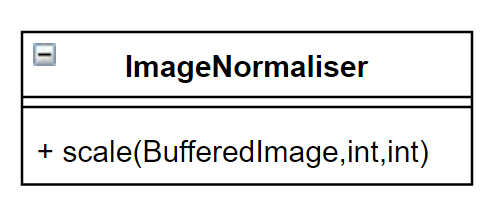


Рисунок 4.12 – Поля класу ImageNormaliser

Опис полів класу:

Методи:

* scale(BufferedImage, int, int) – нормалізує вхідне зображення.

Роль класу:

Нормалізація вхідних зображень.

1. Run

Поля класу зображені на рис. 4.13.

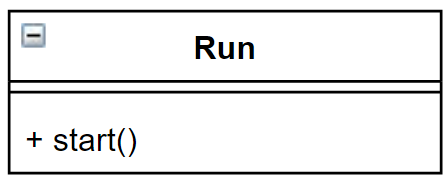


Рисунок 4.13 – Поля класу Run

Опис полів класу:

Методи:

* start() – конструктор класу. Запукає програму.

Роль класу:

Запуск програми.

1. LineRecognizer

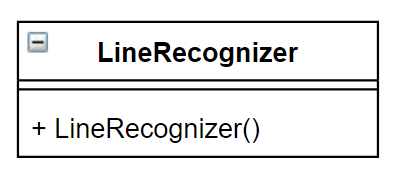
Поля класузображені на рис. 4.14.

Рисунок 4.14 – Поля класу LineRecognizer

Опис полів класу:

Методи:

* LineRecognizer() – конструктор класу. Запускає розпізнавання музичних рядів.

Роль класу:

Розпізнавання музичних рядів.

1. ColumnRecognizer

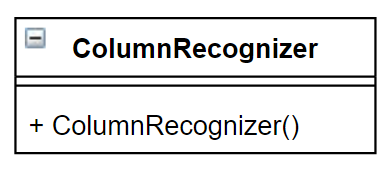
Поля класу зображені на рис. 4.15.

Рисунок 4.15 – Поля класу ColumnRecognizer

Опис полів класу:

Методи:

* ColumnRecognizer() – конструктор класу. Запускає розпізнавання музичних стовпців.

Роль класу:

Розпізнавання музичних стовпців.

1. ASRecognizer

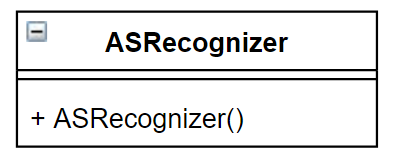
Поля класу зображені на рис. 4.16.

Рисунок 4.16 – Поля класу ASRecognizer

Опис полів класу:

Методи:

* ASRecognizer() – конструктор класу. Запускає розпізнавання наявності пониження чи підвищення у розпізнаних нот.

Роль класу:

Розпізнавання підвищення чи пониження у нот.

1. NoteExistRecognizer

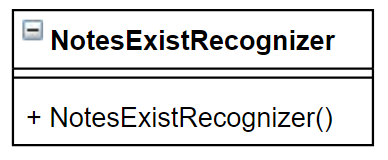
Поля класузображені на рис. 4.17.

Рисунок 4.17 – Поля класу NotesExistRecognizer

Опис полів класу:

Методи:

* NotesExistRecognizer() – конструктор класу. Запускає розпізнавання наявності нот у музичних рядах.

Роль класу:

Розпізнавання наявності нот у музичних рядах.

1. NoteRecognizer

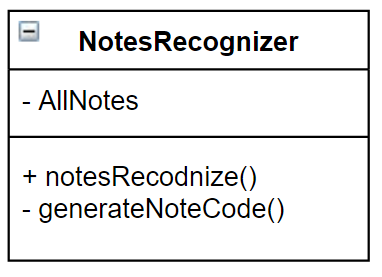
Поля класузображені на рис. 4.18.

Рисунок 4.18 – Поля класу NotesRecognizer

Опис полів класу:

Методи:

* notesRecognizer() – конструктор класу. Запускає розпізнавання нот у музичних рядах.
* generateNoteCode() – генерує код ноти.

Змінні:

* AllNotes – колекція, яка зберігає в собі всі можливі позиції всіх можливих нот.

Роль класу:

Розпізнавання ноти та генерація для неї коду.

1. DrawTabs

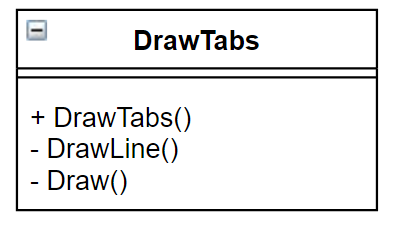
Поля класу зображені на рис. 4.19.

Рисунок 4.19 – Поля класу DrawTabs

Опис полів класу:

Методи:

* DrawTabs() – конструктор класу.
* DrawLine() – малює музичний лад.
* Draw() - малює табулатуру.

Роль класу:

Малювання табулатури по розпізнаним нотам.

1. Note

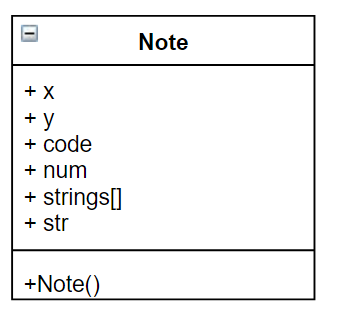
Поля класу зображені на рис. 4.20.

Рисунок 4.20 – Поля класу Note

Опис полів класу:

Методи:

* Nore() – конструктор класу.

Змінні:

* х – позиція ноти по координаті х.
* у – позиція ноти по координаті у.
* code – код ноти.
* num – лад ноти.
* strings[] – показник перекриття струн.
* str – струна ноти.

Роль класу:

Слугує для збереження інформації про ноти на різних стадіях їх розпізнання.

# ПРОГРАМНА РЕАЛІЗАЦІЯСИСТЕМИ РОЗПІЗНАВАННЯ МУЗИЧНИХ НОТ

# Особливості створення структури нейронноїмережівідносно мови програмування Java

Для створення структури нейронної мережі було створено та реалізовано дві сутності:

* Neuron – нейрон, який зберігає в собі інформацію (його вагу), яка являється одним із компонентів для прийняття рішень нейронною мережею.

Параметри нейрона реалізовано наступним шляхом:

* Вага–змінна з типом double.
* Дельта вага - змінна з типом double.
* Зв’язки, що входять в нейрон – колекція, з типом класу Synapse.
* Вихідні зв’язки – колекція, з типом класу Synapse.
* Synapse – зв’язок між нейронами, який відповідає за розподілення впливу нейронів на кінцеву відповідь.

Параметри зв’язку реалізовано наступним шляхом:

* Вага – змінна з типом double.
* Дельта вага– змінна з типом double.
* Нейрон справа (rightSynapses) – об’єкт класу Neuron.
* Нейрон зліва (leftSynapses)– об’єкт класу Neuron.
* Градієнт– змінна з типом double.
* Минула вага– змінна з типом double.
* NeuralNetworlImpl – клас, який створює нейронну мережу по наступним крокам:
* Створюється колекції вхідного, схованих та вихідного шару типу Neuron.
* Відбувається цикл реалізації колекції вихідного типу:
* Створюється нейрон.
* Створюється колекція зв’язків що виходять.
* Задається випадкова вага в радіусі [0 – 0,5].
* Відбувається цикл реалізацій колекції схованого типу:
* Створюється нейрон.
* Створюється колекція зв’язків, що входять.
* Створюється колекція зв’язків що виходять.
* Задається випадкова вага в радіусі [0 – 0,5].
* Відбувається цикл реалізацій колекції схованого типу:
* Створюється нейрон.
* Створюється колекція зв’язків, що входять.
* Задається випадкова вага в радіусі [0 – 0,5].
* Відбувається з`єднання нейронів.
* Цикл поєднання першого шару із n – м шаром:
* З`єднання вихідними зв’язками кожного нейрона першого схованого шару із кожним нейроном n– го шару шляхом створення зв’язків типу Synapseта додавання їх в колекцію rightSynapses.
* З`єднання вхідними зв’язками кожного нейрона n– го шару із кожним нейроном вхідного шару шляхом створення зв’язків типу Synapseта додавання їх в колекцію leftSynapses.

Для створення навчальної вибірки були реалізовані наступні сутності:

* Image–сутність, яка являє собою зображення, а саме його представлення у масиві чорно – білих пікселів із типом double.
* ImageNormalizer – сутність, яка приводить вхідне зображення до розмірів – 640 по ширині, 925 по довготі.
* DataExtractor – клас, який зчитує зображення, та переводить його в масив пікселів із типом double. Шлях переводу пікселів в числа відбувається наступним шляхом:

Створюється масивпікселів типу double із розмірністю ширини зображення, помноженої на довготу зображення.

Відбувається подвійний цикл по широті і довготі зображення, в якому:

* Створюється буфер із типом int, в який зчитується піксель картинки, з-за допомогою бібліотеки ImageIO.
* На основі буфери створюється об’єкт класу Color з параметром alpha = true.
* З об’єкту класу Color виділяється червоний пігмент та записується в масив пікселів.
* Записана інформаціє в соту масиву ділиться на 1000, та перезаписується.
* ImageImpl – клас, який завантажує данні із вказаної папки, та конвертує їх в масив чисел за допомогою класів ImageтаDataExtractor.

# Особливості алгоритму розпізнавання музичних рядіввідносно мови програмування Java

Для створення алгоритму, який буде розпізнавати музичні ряди потрібно:

* ImageImpl– клас, який відповідає за алгоритм подачі вхідних зображень.

Процес подачі зображення відбувається наступним шляхом:

* Створюєтьсяоб’єкт класу BufferedImage.
* Відбувається цикл по довготі зображення із кроком в 75 пікселів.
* Зображення записується об’єкт класу BufferedImage.
* Об’єкт класу BufferedImage обрізає зображення по його довготі з верху на пройдену відстань по довготі а з низу на пройдену відстань + 75.
* Зображення записується в папку для розпізнавання рядів.
* Нейрона мережа, яка була описана в пункті 5.1.
* LineRecognizer – клас, який відповідає за алгоритм фіксування розпізнаних музичних рядів.

Фіксація рядів виконується наступним чином:

* Приймається колекція типу Image, яка пройшла нейронну мережу по розпізнаванню рядків.
* Відбувається цикл по отриманій колекції.
* Якщо поле 0.7 <Image.isRecog< 0.9 відбувається зміщення зображення по довготі на 5 пікселів у верх та подача його на вхід нейронній мережі.
* Якщо поле 0.7 <Image.isRecog< 0.9 та це друга ітерація повторної перевірки зображення, то відбувається зміщення зображення по довготі на 5 пікселів у низ та подача його на вхід нейронній мережі.
* Якщо поле Image.isRecog> 0.9 - створюється об’єкт класу BufferedImage в який записується дане зображення
* Зображення записується в папку для розпізнавання стовпців.

# Створення алгоритму розпізнавання музичних стовпціввідносно мови програмування Java

Для створення алгоритму, який буде розпізнавати музичні стовпців потрібно:

* ImageImpl– клас, який відповідає за алгоритм подачі вхідних зображень.

Процес подачі зображення відбувається наступним шляхом:

* Створюєтьсяоб’єкт класу BufferedImage.
* Відбувається цикл по широті зображення із кроком в 12 пікселя.
* Зображення записується об’єкт класу BufferedImage.
* Об’єкт класу BufferedImage обрізає зображення по його широті справа на пройдену відстань по довготі а зліва на пройдену відстань + 12.
* Зображення записується в папку для розпізнавання стовпців.
* Нейрона мережа, яка була описана в пункті 5.1.
* ColumnRecognizer – клас, який відповідає за алгоритм фіксування розпізнаних музичних стовпців.

Фіксація стовпців виконується наступним чином:

* Приймається колекція типу Image, яка пройшла нейронну мережу по розпізнаванню рядків.
* Відбувається цикл по отриманій колекції.
* Якщо поле 0.7 <Image.isRecog< 0.9 відбувається зміщення зображення по широті на 3 пікселів у верх та подача його на вхід нейронній мережі.
* Якщо поле 0.7 <Image.isRecog< 0.9 та це друга ітерація повторної перевірки зображення, то відбувається зміщення зображення по широті на 3 пікселів у низ та подача його на вхід нейронній мережі.
* Якщо поле Image.isRecog> 0.9 - створюється об’єкт класу BufferedImage в який записується дане зображення
* Зображення записується в папку для розпізнавання наявності музичних нот.

# Створення алгоритму для розпізнавання наявності нот у музичних стовпцяхвідносно мови програмування Java

Для створення алгоритму, який буде розпізнавати наявність нот в музичних стовпцях потрібно:

* ImageImpl – клас, який відповідає за алгоритм подачі вхідних зображень.

Процес подачі зображення відбувається наступним шляхом:

* Створюєтьсяоб’єкт класу BufferedImage.
* Відбувається цикл по широті зображення із кроком в 6 пікселів.
* Зображення записується об’єкт класу BufferedImage.
* Об’єкт класу BufferedImage обрізає зображення по його широті справа на пройдену відстань по довготі а зліва на пройдену відстань + 5.
* Зображення записується в папку для розпізнавання наявності нот.
* Нейрона мережа, яка була описана в пункті 5.1.
* NotesExistRecognizer – клас, який відповідає за алгоритм фіксування розпізнаних музичних нот.

Фіксація нот виконується наступним чином:

* Створюється колекція recNotesнот з типом Note.
* Приймається колекція типу Image, яка пройшла нейронну мережу по розпізнаванню рядків.
* Відбувається цикл по отриманій колекції.
* Якщо поле 0.7 <Image.isRecog< 0.9 відбувається зміщення зображення по широті на 2 пікселя у верх та подача його на вхід нейронній мережі.
* Якщо поле 0.7 <Image.isRecog< 0.9 та це друга ітерація повторної перевірки зображення, то відбувається зміщення зображення по довготі на 2 пікселя у низ та подача його на вхід нейронній мережі.
* Якщо поле Image.isRecog> 0.9 - створюється об’єкт класу Note, в який записуються координати ноти відносно основного зображення. Створений об’єкт додається в колекцію нот.

# Створення алгоритму для розпізнавання пониження чи підвищення нотивідносно мови програмування Java

Для створення алгоритму, який буде розпізнавати підвищення чи пониження нот:

* notes – колекція, яка в якій зберігаються всі розпізнані ноти.

Процес подачі зображення відбувається наступним шляхом:

* Створюєтьсяоб’єкт класу BufferedImage.
* Відбувається цикл по кількості розпізнаних нот.
* З головного зображення вирізається фрагмент по координатамimage.y,image.x– 12, по широті – 12, по довготі – 16. Вирізаний фрагмент записується об’єкт класу BufferedImage.
* Зображення записується в папку для розпізнавання підвищення чи пониження ноти.
* Нейрона мережа, яка була описана в пункті 5.1.

Фіксація підвищених чи понижених знаків виконується наступним чином:

* Створюється подвійний масив даних із типом int.
* Приймається колекція типу Image, яка пройшла нейронну мережу по розпізнаванню рядків.
* Відбувається цикл по отриманій колекції та по розпізнаним нотам.
* Записується розпізнаний знак, або його відсутність до розпізнаної ноти.

# Створення алгоритму для розпізнавання нот та малювання табулатуривідносно мови програмування Java

Для створення алгоритму, який буде розпізнавати ноти та малювати табулатуру:

* AllNotes – клас, в якому записана координати кожної ноти відносно нотного стану
* notes–колекція, яка в якій зберігаються всі розпізнані ноти.
* TabsDraw – клас, який відповідає за рисування нот

Розпізнання нот виконується наступним чином:

* Відбувається цикл по колекції розпізнаних нот та по можливим їх координатам.
* Якщо note.coords == allNotes.coords (допускається відхилення в 3 пікселя) – ноті присвоюється код allNotes.getNoteCode.
* Відбувається цикл, по можливим струнам та ладам, на яких може гратися нота відносно присвоєного їй коду.
* Ноті присвоюєтеся найменший лад із відсутнім перекриттям струни.

Малювання табулатури відбувається наступним чином

* notePainter– малює нотний стан.
* Відбувається цикл по колекції розпізнаних нот.
* tabsDrawотримує код ноти її координати.
* tabsDrawрисує цифру, вичитуючи підвищення чи пониження ноти (лад, на якому грається нота) на нотному стані.

# Приклад використання програми

Щоб розпочати розпізнавання нот потрібно виконати пункти:

1. Натиснути на кнопку «Завантажити зображення», як вказано на рис. 5.1

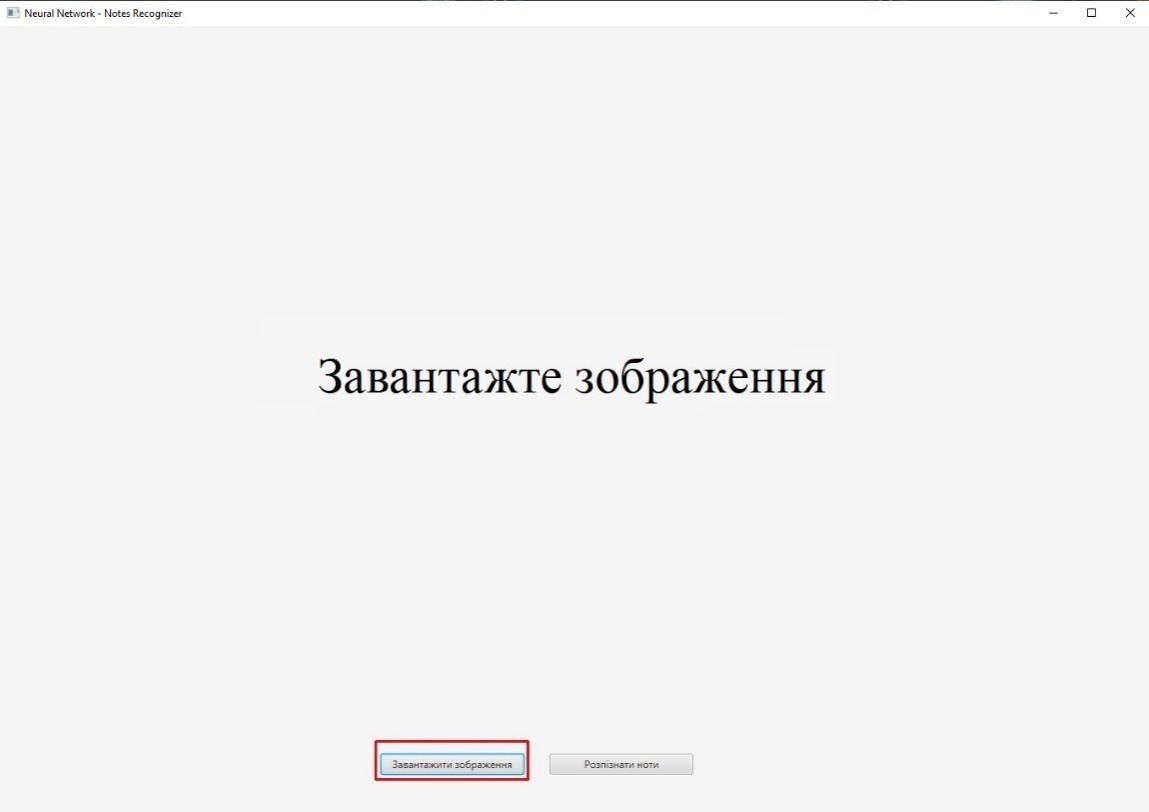


Рисунок 5.1 – Завантажити зображення

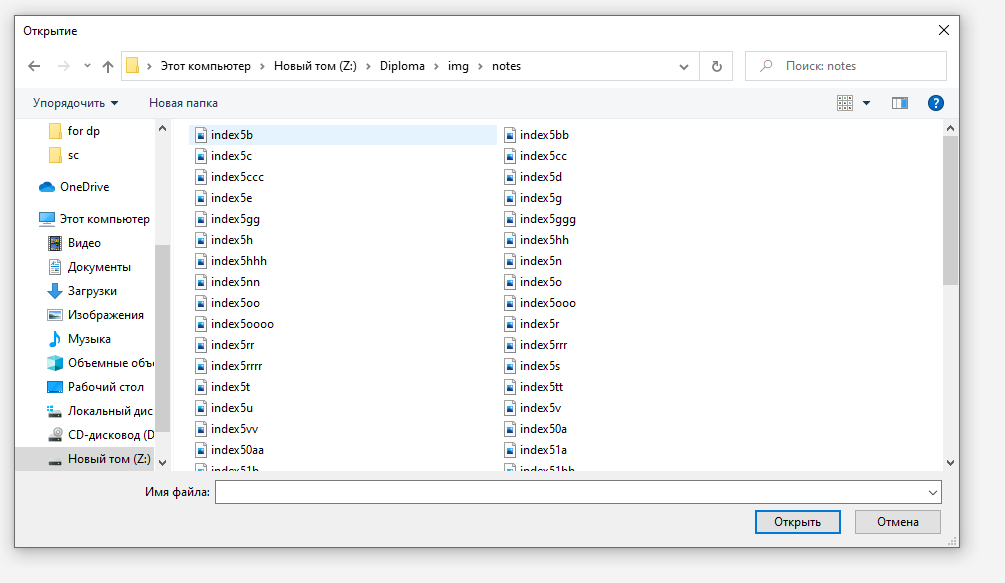
1. Обрати зображення, на якому потрібно розпізнати ноти, як вказано на рис. 5.2.

Рисунок 5.2 – Обрати зображення

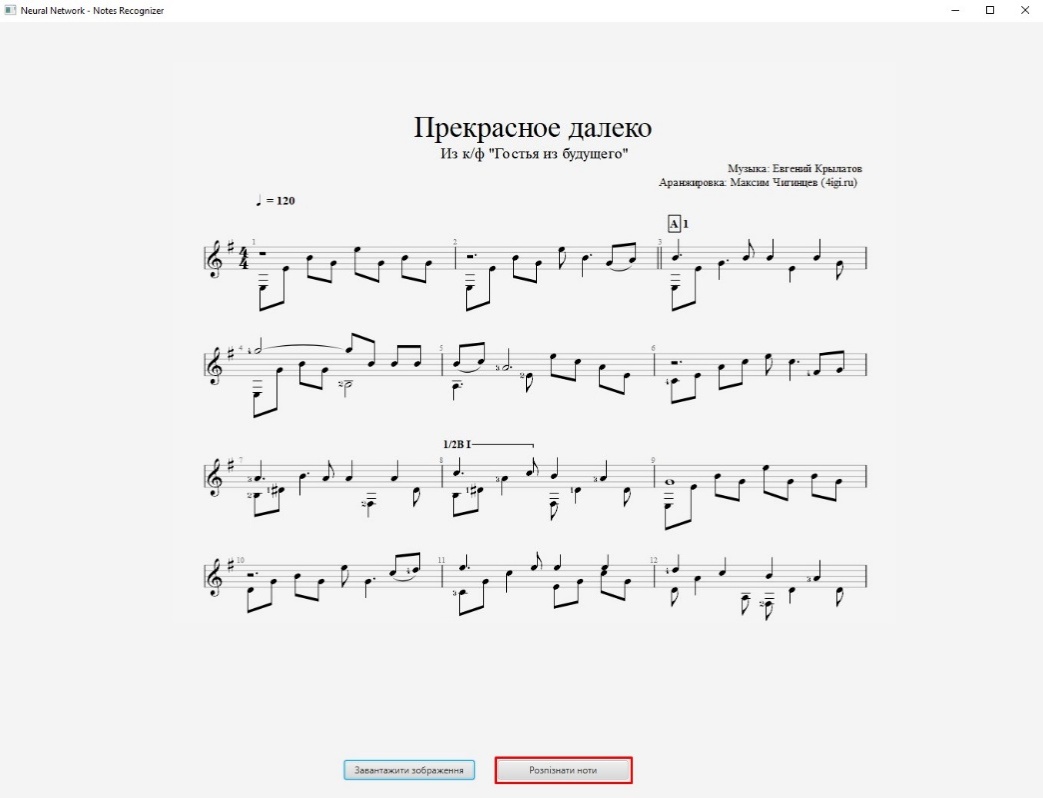
1. Натиснути кнопку «Розпізнати ноти», як вказано на рис. 5.3.

Рисунок 5.3 – Початок розпізнавання нот

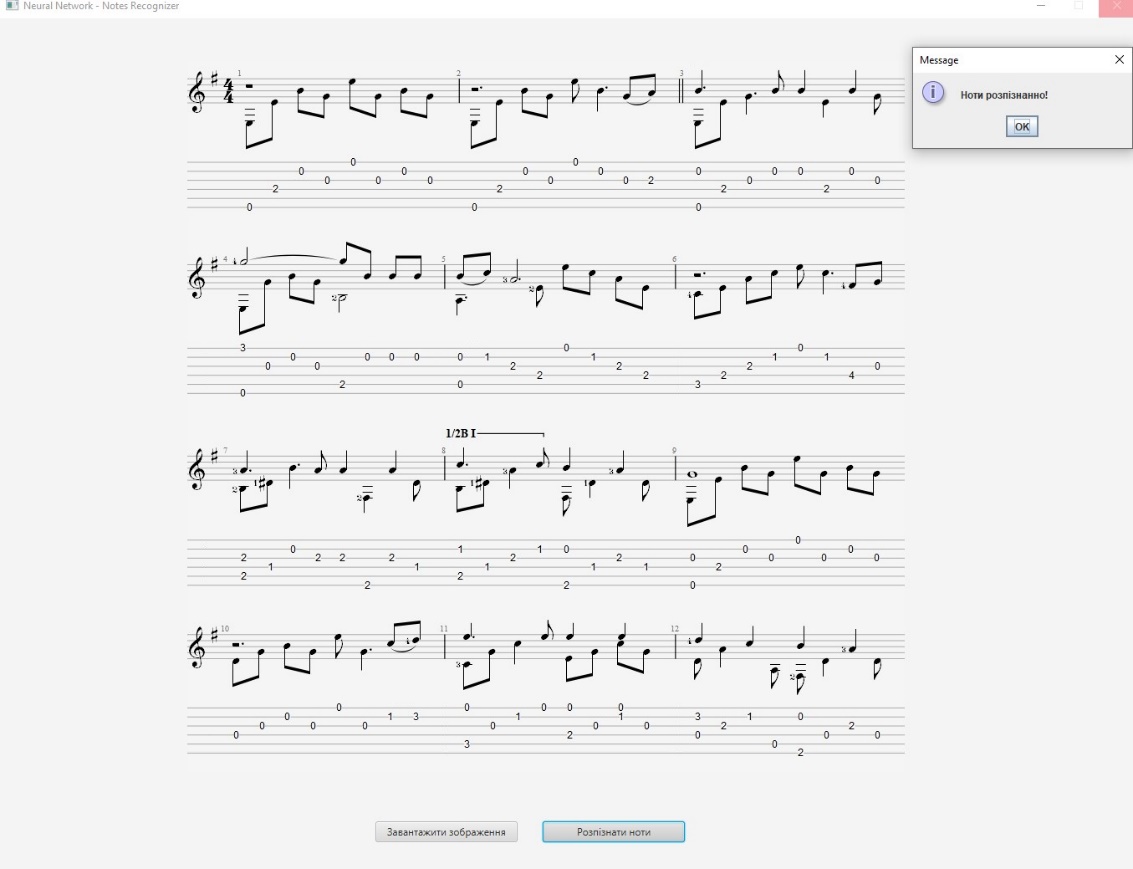
1. Ноти розпізнано, як продемонстровано на рис. 5.4.

Рисунок 5.4 – Намальована табулатура по нотам

# ВИЗНАЧЕННЯ ВЛАСТИВОСТЕЙ СИСТЕМИ РОЗПІЗНАВАННЯ МУЗИЧНИХ НОТ

# Оцінювання точності результатів роботи системи розпізнавання музичних нот

Було проведено тестове розпізнання 10 сторінок музичних нот. Розглянемо результати тестування чотирьох нейронних мереж, які були наведені у табл. 6.1.

Таблиця 6.1 – Результати проведеного тестування

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Сторінка з нотами | Точність розпізнання рядів | Точність розпізнання стовпців | Точність розпізнання наявності нот | Точність розпізнання наявності підвищення чи пониження нот |
| 1 | 95.6% | 92.4% | 95.6% | 99.8% |
| 2 | 99.3% | 91.1% | 94.7% | 95.2% |
| 3 | 98.4% | 93.4% | 97.2% | 99.4% |
| 4 | 96.7% | 95.8% | 93.2% | 96.7% |
| 5 | 95.2% | 97.2% | 94.9% | 97.5% |
| 6 | 96.9% | 96.0% | 95.4% | 98.5% |
| 7 | 97.0% | 93.7% | 96.7% | 97.4% |
| 8 | 98.1% | 93.5% | 94.2% | 95.9% |
| 9 | 97.5% | 95.6% | 95.1% | 95.3% |
| 10 | 97.4% | 94.2% | 95.7% | 97.7% |

Розрахуємо середнє значення точності нейронних мереж за формулою:

* Нейронна мережа для розпізнавання музичних рядів – найважливіша частина системи для розпізнавання нот, так як від неї залежить точність послідуючих нейронних мереж. Розрахуємо для неї точність:
* Розрахуємо точність відповідей для нейронної мережі для розпізнавання музичних стовпців:
* Розрахуємо точність відповідей для нейронної мережі для розпізнавання наявності нот в музичних стовпцях:
* Розрахуємо точність відповідей для нейронної мережі для розпізнання підвищення чи пониження нот:

# Експериментальне визначення часу переходу від теорії до практичної гри музикантів – новачків

Був проведений експеримент по освоєнню навику розпізнавання музичних нот на ладах гітари, у якому запроваджено тестування програми тривалістю в неділю, у якому прийняло участь 10 музикантів новачків. Перших п’ять музикантів займались самостійно а других п’ять із програмою для графічного розпізнавання музичних нот. Перед тестуванням кожнен із його учасників оцінив свої знання із знання находження нот на гітарі по десятибальній шкалі. Результати опиту зображені на рис. 6.1.

Рисунок 6.1 – Діаграма результатів опитування перед проводженням експерименту

Після завершення тестування було проведене повторне опитування, результати якого описує рисунок 6.2.

Рисунок 6.2 – Діаграма результатів опитування після проводженням експерименту

За результатами опитування маємо що перша група, яка займалась самостійно підвищила свої вміння до розпізнавання нот на 10 – 20%. Друга група, яка працювала з програмою, підвищила свої навички по розпізнаванню нот на 20 – 40%, що являється набагато більше, ніж у першої групи.

# ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Охорона праці це – система правових, соціально – економічних, організаційно – технічних, санітарно – гігієнічних, лікувально – профілактичних заходів та засобів, направлених на зберігання життя та здоров’я і працездатності працівника в процесі трудової діяльності.

Оскільки диплом присвячено розробці програми «Програмне забезпечення для розпізнавання графічного зображення музичних нот методами машинного навчання» тому для розгляду питань охорони праці оберемо робоче місце головного менеджера проекту з розроблення програмного забезпечення.

Охорона праці. Організація та управління.

Працівник інженер – програміст на своєму робочому місті інженера – програміста виконує свою діяльність на фірмі «STARKIND» отже розглянемо організацію та управління охороною праці на предмет управління охороною праці.

У STARKIND є система управління охороною праці[11] (СУОП) — це підсистема, яка контролює показники безпеки, аналізує стан ОП та забезпечує підготовку, прийняття та реалізацію рішень, направлених на зберігання життя та здоров’я працівника.

До основних функцій управління охороною праці належать:

* прогнозування і планування робіт, їх фінансування;
* організація та координація робіт;
* облік показників, аналіз та оцінка стану умов і безпеки праці;
* контроль за станом охорони праці та функціонуванням СУОП;
* стимулювання діяльності з охорони праці.

Контроль здійснюється відповідними уповноваженими особами, згідно наведеними нормативними актами.

Нормативно - правові акти з охорони праці (НПАОП):

* НПАОП 0.00-1.02-08 - Правила будови і безпечної експлуатації ліфтів - 01.09.2008 наказ № 190.
* НПАОП 0.00-1.04-07 - Правила вибору та застосування засобів індивідуального захисту органів дихання - 28.12.2007 наказ № 331.
* НПАОП 0.00-1.25-10 - Правила улаштування та експлуатації загальнопромислових вогнеперепинувачів27.07.2010 наказ № 138.
* НПАОП 0.00-1.78-17Правила охорони праці в архівних установах - 18.04.2017 наказ № 634

Служба охорони праці [12] в STARKIND існує та виконує наступні функції:

* Забезпечення фахової підтримки рішень роботодавця з питань охорони праці;
* забезпечення безпеки виробничих процесів, устаткування, будівель і споруд;
* забезпечення працівників засобами індивідуального та колективного захисту;
* професійної підготовки і підвищення кваліфікації працівників з питань охорони праці, пропаганди безпечних методів праці;
* вибору оптимальних режимів праці й відпочинку працівників;інформування та надання роз'яснень працівникам підприємства з питань охорони праці.

Керівником є уповноважений інженер з охорони праці, який виконує свої обов’язки згідно своєї посади, а саме первинний інструктаж з охорони праці та навчання з охорони праці.

Навчання з охорони праці [13] у STARKIND виконується згідно нормативним вимогам та встановленому плану зі всіма співробітниками фірми, крім того, медогляди здійснюються щорічно. Контроль покладений на інженера з охорони праці.

Видача засобів індивідуального захисту здійснюється начальниками відділів за необхідністю кожному співробітнику.

В робочій зоні є кімната в якій зберігаються протигази на випадок пожежі.

Розслідування нещасних випадків здійснюється за необхідністю комісією, з розслідувань.

Розрахунки з питання охорони праці та безпеки у надзвичайних ситуаціях наведені у додатку В[14].

**Висновки**

Розглянуто організацію та управління охороною праці на робочому місці інженера – програміста.

Проаналізовано робочі умови працівника, що забезпечують для них комфорт

Розраховано індивідуальні завдання на тему занулення робочого місця та розрахунок по безпеці при надзвичайній ситуації – пожежа.

Запропоновано заходи з охорони праці, які спрямовані на покращення занулення на робочому місці працівника шляхом перевірки провідників.

# ВИСНОВКИ

У ході дипломного проектування розроблено програмне забезпечення, за допомогою якого можна доволі точно розпізнати ноти та згенерувати табулатуру до них. Розроблений гнучкий алгоритм, який побудований за допомогою методів машинного навчання для розпізнання музичних нот. Складовою алгоритму є чотири нейронних мережі, які дають змогу сильно зменшити навчальну вибірку та період тренування відносно аналогів.

Було проаналізовано та описано процес розпізнання музичних нот, описано блок – схеми алгоритмів до кожної із нейронних мереж, спроектовано та описано систему, до якої наведено UML – діаграму, обрано технології розробки, створено програмне забезпечення, описано всі програмні класи та їх поля, протестовано програмне забезпечення.

Було проведено експеримент із по освоєнню навику розпізнавання музичних нот на ладах гітари. В ньому приймало участь 2 групи, перша займалась самостійно, друга із програмою для розпізнавання музичних нот. Перед експериментом та після нього було проведено опитування, щодо навичок обох груп, за результатами якого з’ясувалось що група, яка навчалась із програмою прогресувала на 10 – 20% швидше ніж група, яка навчалась самостійно.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Хайкін С. Нейронні мережі [Текст]/ Хайкін С. Cпб.: Діалектрика 2020 – 1104с.
2. DiMarzio J.F. QuickStartGuideto JavaFX [Текст]/ DiMarzio J.F. Cпб.: McGraw-HillEducation2014 – 325с.
3. Арнис А. Нейронные сети [Електронний ресурс]/ Арнис А. – 2016 – 50с.
4. Річард С., Саттон Ендрю, Дж. БартоНавчання з підкріпленням[Електронний ресурс]/ Річард С., Саттон Ендрю, Дж. Барто. – 2020 – 552с.
5. Чару Аггарвал Нейронні мережі та глибоке навчання [Електронний ресурс] Чару Аггарвал Нейронні. 2020 – 752с.
6. ГрасДжоел Наука про дані [Текст]/ ГрасДжоелСпб.: БХВ 2016 – 336с.
7. ТарикРашик Створюємо нейронну мережу [Текст]/ Спб.: Діалектика – Вільямс 2020 – 272с.
8. Брюс Эккель. Философия Java [Текст] /Брюс Эккель -СПб.: Питер1998. – 1168c.
9. Мартін Фоулер UML Distilled[Текст] / Мартін Фоулер СПб.: Мічіган 2018. – 456c.
10. Кевін Уейн,РобертСеджвикАлгоритмы на Java [Електроний ресурс] / Кевін Уейн, Роберт Седжвик – 2018. – 848с.
11. ГОСТ 12.1.005-88 Система стандартов безопасности труда (ССБТ). Общие санитарно-гигиенические требования к воздуху рабочей зоны (с Изменением № 1). [На заміну ГОСТ 12.1.005-76; чинний від 1989-01-01]. М.: Стандартинформ, 2008.
12. Гогіташвілі Г.Г. Основи охорони праці: Навчальний посібник/ Г.Г. Гогіташвілі, В.М. Лапін. - Львів: Новий світ-2000, 2006. - 232 с.
13. Гогіташвілі Г.Г. Управління охороною праці та ризиками за міжнародними стандартами: Навчальний посібник/ Г. Г. Гогіташвілі, Є. -Т. Карчевські, В. М. Лапін. - К.: Знання, 2007. - 367 с
14. Про затвердження Правил пожежної безпеки в Україні : наказ Міністерства внутрішніх справ України від 30.12.2014 р. № 1417. Офіційний вісник України. 2015 р. № 26. С. 91, Ст. 767.