СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра програмування та математики

ПОЯСНЮВАЛЬНА ЗАПИСКА
до кваліфікаційної випускної роботи

освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)
спеціалізація „Інженерія програмного забезпечення”

на тему „Програмна реалізація елементів гри з використанням методів штучного інтелекту”

Виконав: студент групи ІПЗ-16д _____ Є.В. Панфілов
(підпис) (ініціали і прізвище)

Керівник _____ В.О. Лифар
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ В.О. Лифар
(підпис) (ініціали і прізвище)

Рецензент_____

Сєвєродонецьк – 2020

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра програмування та математики

Освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)
спеціалізація „Інженерія програмного забезпечення”
(назва спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____ Лифар В.О.
“____” _____2020 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Панфілов Євген Віталійович
(прізвище, ім’я, по батькові)

1.Тема роботи Програмна реалізація елементів гри з використанням методів штучного інтелекту. _____
Керівник роботи доцент, д.т.н. Лифар Володимир Олексійович,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від “___”_____20__року №___

2. Строк подання студентом роботи 20 травня 2020 р. _____

3. Вихідні дані до роботи Об'єктом даної роботи є процес розробки гри з використанням штучного інтелекту. _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Аналітичний огляд, з висвітленням наступних питань: алгоритми, гени, схрещування, нащадки, нейронна система її недоліки та переваги. Основна частина, в якої висвітлити: збір вимог та проектування, етап даталогічного проектування. Висновки. Перелік використаних джерел. _____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників)

_____

_____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання__30 березеня 2020 року._____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	30.03.20	
2	Укладання і погодження з керівником плану і етапів виконання роботи	06.04.20	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	13.04.20	
4	Аналіз шляхів виконання завдання. Вибір і погодження керівником оптимального шляху	20.04.20	
5	Укладання та тестування програмного продукту	27.04.20	
6	Укладання, оформлення та погодження пояснювальної записки з керівником	04.05.20	
7	Здача готової пояснювальної записки на кафедру	12.05.20	
8	Укладання доповіді і презентації	30.05.20	

Студент _____ ___Є.В. Панфілов _____
(підпис) (ініціали і прізвище)

Керівник роботи _____ ____В.О. Лифар_____
(підпис) (ініціали і прізвище)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІПЗ-16д Панфілов Є.В.

Науковий керівник

Доцент, д.т.н. _____ Лифар В.О.

Оцінка наукового керівника: _____

Рецензент _____

ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК

_____ Лифар В.О.

підпис

РЕФЕРАТ

Робота містить: 46 сторінок основного тексту, 24 сторінок додатків, 23 рисунка, 11 використаних джерел.

Метою випускної кваліфікаційної роботи є вивчення особливостей роботи штучного інтелекту, методів його реалізації, оптимальних рішень, в плані програмування і підбору продуктивних інструментів.

Була проведена робота яка включала в себе створення штучного інтелекту, знаходження его мінусів і плюсів, визначення найбільш вдалого прикладу для проекту, так само вибір актуальної мови програмування і нових технологій потрібних для реалізації даної роботи.

В результаті виконаної роботи, була реалізована гра, яка включає в себе штучний інтелект, який може виконувати будь-які завдання які користувач йому поставить, виконувати на ідеальному рівні, без помилково.

Даний приклад демонструє те, що така складна і велика тема як штучний інтелект, може застосовуватися в будь-яких сферах, так само і в іграх.

Система реалізована відповідно всім вимогам технічного завдання.

Зроблено детальний опис процесу розробки штучного інтелекту, а також, продемонстрована робота готового прикладу гри.

ЗМІСТ

ВСТУП

Актуальність. Штучний інтелект - наука і технологія створення інтелектуальних машин, особливо інтелектуальних комп'ютерних програм. Штучний інтелект надає засіб і випробувальну модель для теорій інтелекту: ці теорії можуть бути сформульовані мовою комп'ютерних програм, а потім - випробувані. На сьогоднішній день існує безліч алгоритмів і підходів до створення ігрового штучного інтелекту. Так, ШІ може бути заснований на використанні Булевої алгебри та обчислення предикатів, в якому вона розширена за рахунок введення предметних символів, відносин між ними, кванторів існування та загальності. Іншим поширеним підходом до побудови ШІ є структурний, який передбачає моделювання структури людського мозку.

Об’єкт дослідження: процес розробки елементів гри з використанням штучного інтелекту.

Предмет дослідження: гра побудована на основі штучного інтелекту.

Мета дослідження: розробка ідеальної нейронної системи на прикладі сучасної гри.

Задачі дослідження:

1. Ознайомитися з принципом створення штучного інтелекту, його можливостей і актуальності застосування у даному проекту.
2. Сформувати список актуальних та потрібних технологій і мов, для розробки власного штучного інтелекту.
3. Розробити гру з використанням штучного інтелекту та нейронних мереж.

1 АНАЛІТИЧНИЙ ОГЛЯД

Штучний інтелект (ШІ, англ. Artificial intelligence, AI) - наука і технологія створення інтелектуальних машин, особливо інтелектуальних комп'ютерних програм. ШІ пов'язаний саме з таким завданням використання комп'ютерів для розуміння людського інтелекту, але не обов'язково обмежується біологічно правдоподібними методами.

Основні властивості ШІ – це розуміння мови, навчання і здатність мислити і, що важливо, діяти.

ШІ - комплекс споріднених технологій і процесів, що розвиваються якісно і швидко, наприклад:

- обробка тексту на природній мові
- машинне навчання
- експертні системи
- віртуальні агенти (чат-боти і віртуальні помічники)
- системи рекомендацій

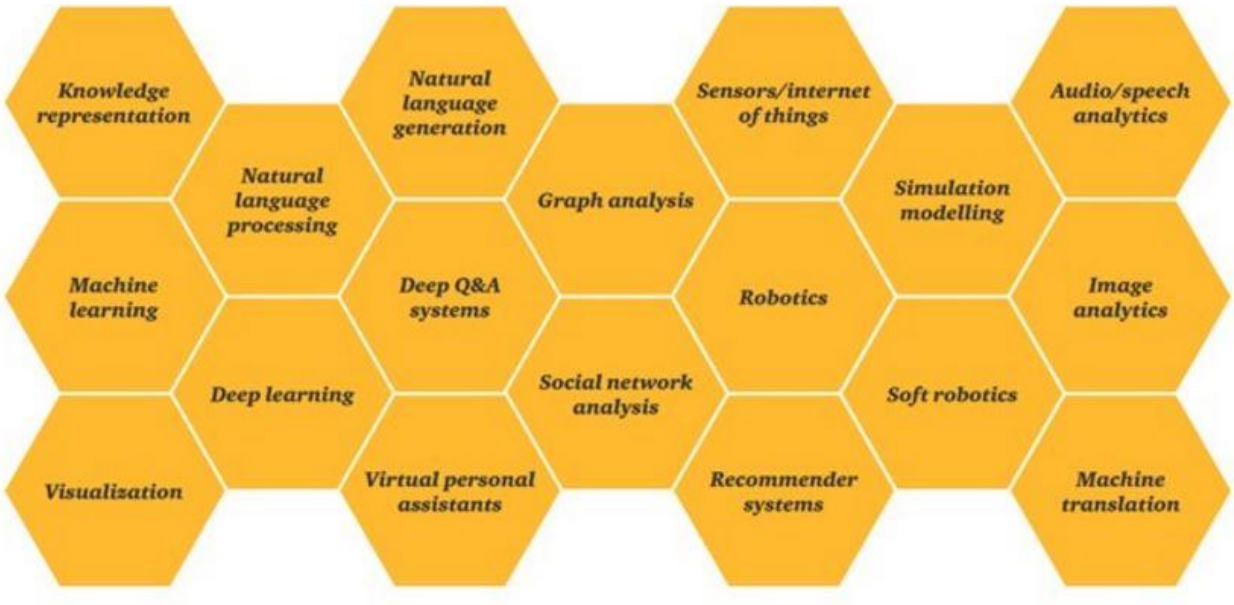

Рисунок 1. Технологічні напрямки ШІ.

1.1. Методи ШІ: NLP, CV, Date Science

Природна мова (NLP) Мовні технології:

- тексти: розпізнають, автоматично переводять
- мова: розпізнають, генерують

Комп'ютерний зір (CV):

- знаходять, відстежують, класифікують, ідентифікують об'єкти
- отримує дані з зображень
- аналізують отриману інформацію

В подальшому (CV) використовується для: розпізнавання об'єктів, відео аналітики, опису змісту зображень і відео, розпізнавання жестів і рукописного введення, інтелектуальної обробки зображень

Аналіз даних (Date Science):

- витягають знання
- знаходять закономірності в даних
- прогнозують

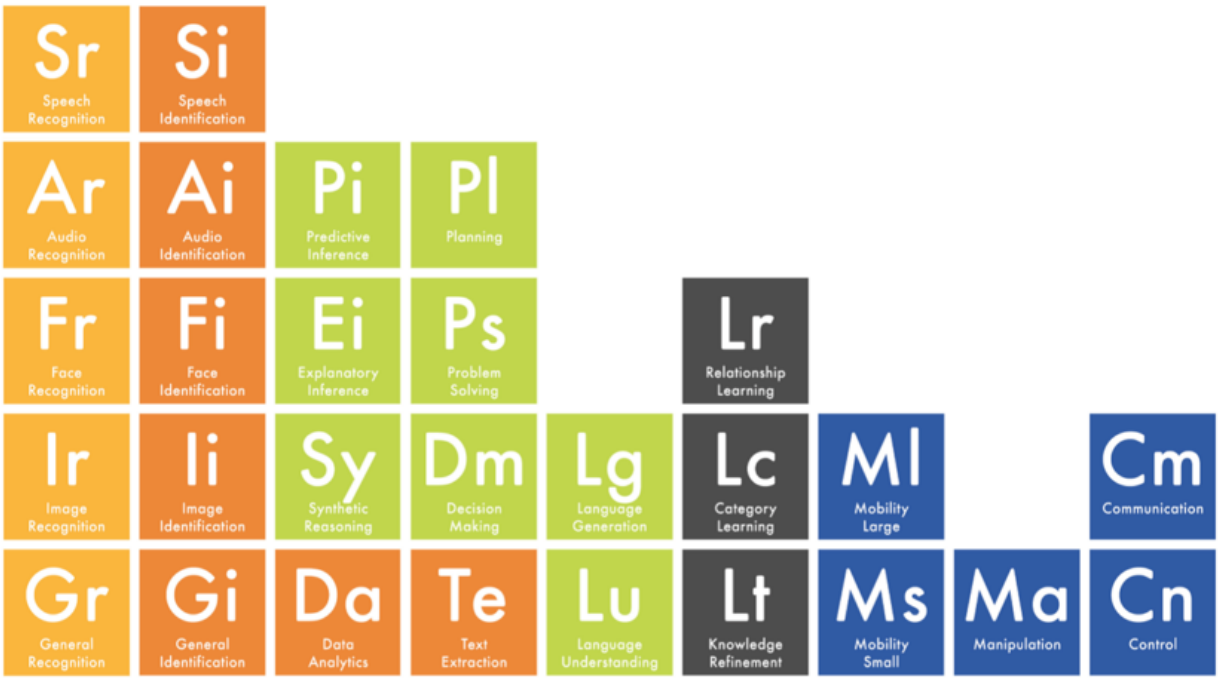

Рисунок 2. Методи ШІ.

1.2. Вплив штучного інтелекту на світ

Британський вчений Стівен Хокінг часто висловлювався про розвиток штучного інтелекту (ШІ) як про реальну причину можливого знищення людського роду.

У квітні 2017 Стівен Хокінг в ході відеоконференції в Пекіні, що відбулася в рамках Глобальної конференції мобільного інтернету, заявив:

"*Розвиток штучного інтелекту може стати як найбільш позитивним, так і найстрашнішим фактором для людства. Ми повинні усвідомлювати небезпеку, яку він собою являє"*

Як розповів учений у своєму інтерв'ю виданню Wired в кінці листопада 2017 року, він побоюється того, що ШІ може в цілому замінити людей.

За словами самого Хокінга, люди можуть створити дуже потужний штучний інтелект, який буде надзвичайно гарний у досягненні своїх цілей. І якщо ці цілі не будуть збігатися з людськими, то у людей будуть проблеми, вважає вчений. [1]

Ілон Маск також має свою думку на тему ШІ.

Американський мільярдер раз у раз попереджає про те, що людям варто бути вкрай обережними з цією технологією. Глава Tesla і SpaceX переконаний, що вченим не вдасться зробити роботів безпечними.

«Вважаю, що штучний інтелект рано чи пізно вб'є всіх нас», - заявив Маск.

Шанс вижити у людській цивілізації мінімальний - лише 5-10%, вважає він. При цьому горезвісне повстання машин, за його словами, має відбутися вже в найближче десятиліття.

Маск вважає, що основна небезпека криється в тому, що вчені створюють техніку, яка здатна самостійно вчитися, вдосконалюватися без

участі людини, вирішувати завдання і реагувати, але жодну машину не можна навчити виявляти емоції, керуватися мораллю, робити виключення в залежності, наприклад, від каяття протилежної сторони конфлікту. Людські почуття не підвладні роботам, і в цьому криється проблема - ніхто не знає, до яких умовиводам в кінцевому рахунку прийдуть машини. Таким чином, робить висновок Маск, внаслідок неконтрольованості ШІ буде страшніше ядерної зброї.

Єдине позитивне рішення проблеми Маск бачить в безпосередньому поєднанні з комп'ютерами - а це якраз те, над чим і працює, власне кажучи, що належить йому компанія Neuralink. Нейронні мережива дозволять людям еволюціонувати разом з машинами, зберігаючи лідерство в гонці розвитку з штучним інтелектом, вважає Маск.

Цікаво, що, незважаючи на апокаліптичні прогнози, інженер згоден з тим, що нині технології ШІ надають людству неоціненну послугу: наприклад, при діагностуванні раку і виявленні суїцидальної поведінки. Ці розробки потенційно можуть поліпшити якість життя кожного жителя Землі, тому сфера штучного інтелекту, каже Маск, потребує не в забороні, а в регулюванні. [2]

Вплив штучного інтелекту на ринок праці.

Компанії освоюють ШІ-систему яка не допускає витік кадрів.

У вересні 2019 року стало відомо про те, що група компаній Recruit Holdings, що спеціалізується на наданні комплексних послуг в області кадрових ресурсів, почала освоювати ШІ-систему, яка допомагає не допустити витік кадрів. Штучний інтелект завчасно виявляє співробітників з наміром звільнитися і оповіщає роботодавця, що дає час втрутитися в ситуацію і запобігти відходу працівника. [3]

Компанії витрачають мільйони на захмарні зарплати ШІ-фахівців.

Зарплати фахівців в області штучного інтелекту (ШІ) продовжують зростати до захмарних висот. Про це свідчать дані з податкової декларації

компанії OpenAI - некомерційної організації, зайнятої дослідженнями штучного інтелекту, яку в 2015 році заснував Ілон Маск спільно з ще декількома інвесторами.

Як випливає з документа, витяги з якого наводить The New York Times, в 2016 році OpenAI заплатила своєму головному гуру в ШІ-сфері Іллі Суцкеверу (Ilya Sutskever) $ 1,9 млн. Ще один провідний науковий співробітник компанії - Іан Гудфеллоу (Ian Goodfellow) отримав більше $ 800 тисяч, хоча він влаштувався на роботу в OpenAI тільки в березні 2016 року. Обидва фахівці - вихідці з Google. Крім того, інженеру-робототехніку Пітеру Аббілу (Pieter Abbeel), який раніше працював в університеті Каліфорнії і перейшов в OpenAI лише в червні 2016- го, виплатили $ 425 тис. Всі зазначені суми включають бонуси, отримані співробітниками при підписанні контракту.

В цілому за свій перший рік роботи компанія OpenAI витратила близько $ 11 млн, з яких понад $ 7 млн затрачено на зарплати та інші види винагороди співробітникам. У 2016 році в штаті компанії значилося 52 людини.

Оклади провідних ШІ-фахівців злетіли до небес через гострий дефіцит кваліфікованих кадрів у цій галузі. За оцінками незалежної канадської ШІ-лабораторії Element AI, в світі налічується всього лише близько 22 тисяч програмістів з науковим ступенем, здатних розробляти системи штучного інтелекту.

Опубліковані OpenAI цифри дають уявлення про те, скільки готові платити компанії цінних ШІ-кадрам. Разом з тим OpenAI - це некомерційна організація, так що пропонований нею рівень зарплати може бути далеко не найвищим.

Співробітники провідних технологічних компаній і люди, які одержували пропозиції про роботу від IT-гігантів, повідомили NYT, що з урахуванням заохочень у вигляді опціонів на акції розмір винагороди у провідних ШІ-фахівців вимірюється кількома мільйонами. Навіть початківці співробітники з нульовим або невеликим практичним досвідом роботи в сфері

штучного інтелекту можуть розраховувати на річну зарплату від $ 300 до $ 500 тисяч з урахуванням окладу і премій у вигляді акцій.

В якості ще одного прикладу, що ілюструє рівень заробітку експертів з сфері технологій штучного інтелекту, таких як нейронні мережі, машинне навчання і т.д., журналісти привели лондонську ШІ-лабораторію DeepMind, з 2014 року належить Google. Відповідно до річного фінансового звіту цієї компанії, в 2016-му видатки на 400 співробітників фірми склали $ 138 млн, тобто в середньому на кожного співробітника, включаючи дослідний і інший персонал, довелося по $ 345 тисяч.

Захмарні зарплати деяких іменитих ШІ-фахівців відображають не тільки високий рівень їх компетенції, але ще і авторитет в своєму професійному співтоваристві, а також здатність залучити інших дослідників.

«Беручи на роботу "зірок", ви отримуєте заодно і всіх, кого вони за собою залучать. Ви платите за популярність і увагу до цих людей», - каже Кріс Ніколсон (Chris Nicholson), генеральний директор і співзасновник ШІ-стартапу Skymind.

ШІ створить більше робочих місць, ніж вб'є.

На початку жовтня 2017 року аналітична компанія Gartner склала прогноз щодо впливу штучного інтелекту (ШІ) на ринок праці. Експерти вважають, що завдяки роботам буде створено більше робочих місць, ніж ліквідовано. Самі люди дуже бояться автоматизації.

У Gartner очікують, що до 2021 року буде автоматизовано близько 1,8 млн робочих місць, а також створено 2,3 млн нових позицій, на яких зможуть працювати люди. Таким чином, чистий приріст живої робочої сили складе близько 500 тис. чоловік.

Ступінь впливу ШІ на ситуацію з кадрами буде залежати від галузі: в деяких з них чисте число робочих місць буде знижуватися кілька років, а в

інших - наприклад, в охороні здоров'я та освіті - це кількість ніколи не буде скорочуватися, кажуть аналітики.

Вони не роз'яснили методику, по якій був складений такий прогноз. Основне припущення експертів полягає в тому, що штучний інтелект скоріше не замінить людей, а буде доповнювати їх професійну діяльність, роблячи її більш швидкий, ефективний і продуктивніше.

За словами дослідників, «розумні» машини поступово перетворюються в «супертехнологію», їм під силу стає виконання найрізноманітніших видів праці - як фізичного, так і інтелектуального. Когнітивні можливості програмного забезпечення розширюються на багато областей інтелектуальної діяльності, наприклад, на фінансову аналітику, медичну діагностику та аналіз даних будь-якого виду.

Безпілотні автомобілі, роботи-няні, автоматизовані системи прийому людей на роботу і інші ШІ-технології лякають людей масовими звільненнями. У жовтні 2017 року дослідницька компанія Pew Research опублікувала результати опитування серед американців і з'ясувала, що 70% населення США боїться автоматизації і втрати робочих місць.

Згідно з прогнозами Gartner, до 2021 року більше половини компаній будуть витрачати більше грошей на розробку і впровадження чат-ботів, ніж на створення мобільного програмного забезпечення.

В ООН попереджають, що, якщо суспільство не встигне адаптуватися до нових технологій, то штучний інтелект порушить стабільність і призведе до непередбачуваних економічних та політичних наслідків. [4]

1.3. Сфери застосування ШІ

Сфери застосування ШІ досить широкі і охоплюють як звичні слуху технології, так і з'являються нові напрямки, далекі від масового застосування, інакше кажучи, це весь спектр рішень, від пилососів до космічних станцій. Можна розділити всі їх різноманітність за критерієм ключових точок розвитку.

Рисунок 3. Основні комерційні сфери застосування технологій ШІ

Аналітика Big Data.

З практичною роботою аналітичних алгоритмів Big Data стикається сьогодні практично кожна людина. IT-гіганти типу Facebook, Google, Яндекс протягом ряду років накопичують дані про користувачів, патерни поведінки, пошукові запити і т.д. Нікого не дивує, що розмова на домашній кухні в присутності розумної колонки або голосового помічника завершується появою відповідної реклами в поштовій програмі. Розумні алгоритми допомагають перекладати тексти і, треба сказати, справляються з цим завданням все краще.

Корпоративний сектор, який не має доступу до таких обсягів накопичених даних, сьогодні тільки шукає найбільш ефективні підходи використання великих даних. Так, банки покладають великі надії на OpenAPI і можливості, що відкриваються з підключенням баз даних партнерів. У цій частині корпоративного сектору ще тільки належить пройти великий шлях, щоб набути досвіду створення таких клієнтських рекомендаційних сервісів, які будуть не дратувати людини, а забезпечувати йому реальну допомогу в тій чи іншій життєвій ситуації.

Медична діагностика.

Компанії «ТехЛАБ» і «Нетріка» в лютому завершили впровадження підсистеми «онкодопомоги». Головне її призначення - ефективна маршрутизація онкопацієнтів, а також контроль термінів, обсягів і результатів наданої їм медичної допомоги. У перекладі на звичайну мову це означає, що розумна комп'ютерна система, яка має в своєму розпорядженні велику базу клінічних даних та історій хвороби пацієнтів із злоякісними новоутвореннями або підозрами на них, надаватиме рекомендації щодо їх лікування з урахуванням індивідуальної клінічної картини, забезпечувати спадкоємність лікування, а також контролювати дотримання встановленого маршруту лікування.

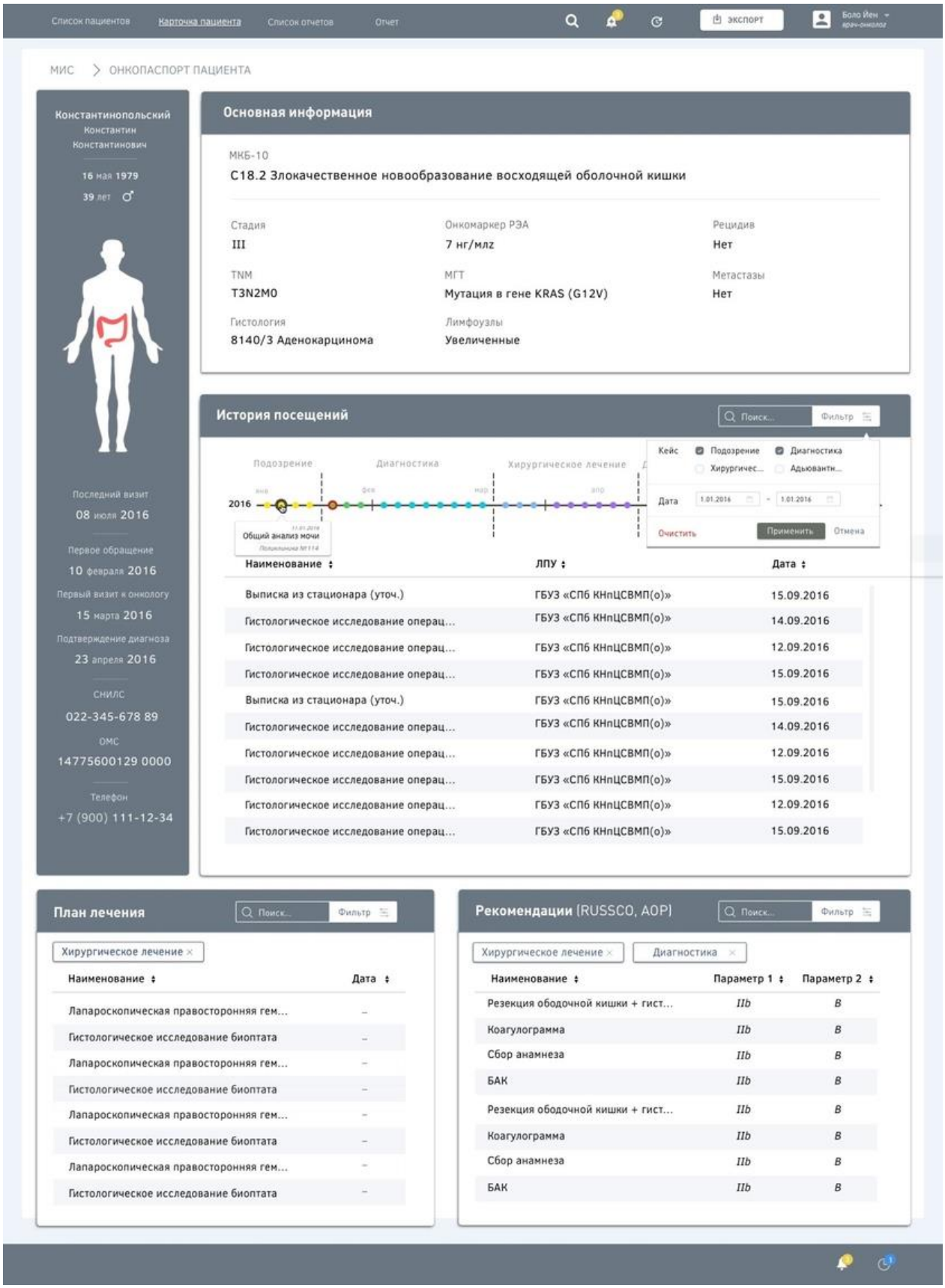

Рисунок 4. Медична картка пацієнта с рекомендаціями ШІ.

Штучний інтелект в музиці і живопису.

27 березня 2019 року з'явилася інформація про те, що Warner Music уклала перший в історії контракт з виконавцем-алгоритмом Endel, що створює музичні композиції під настрій користувача в поточний момент. За умовами контракту, протягом року нейросеть Endel випустить 20 унікальних альбомів. На березень 2017 року п'ять альбомів вже доступні для скачування в iTunes, при цьому всі альбоми створені, як виражаються розробники, «натисненням однієї кнопки».

За твердженням розробників алгоритму, Endel адаптується до настрою користувача і допомагає йому в залежності від поставлених завдань - музика нейромережі допомагає займатися спортом, працювати, засипати або медитувати. При цьому ШІ сам визначає, що потрібно людині в даний момент, аналізуючи безліч параметрів: час доби, геолокацію, погоду за вікном, пульс і частоту биття серця людини. [5]

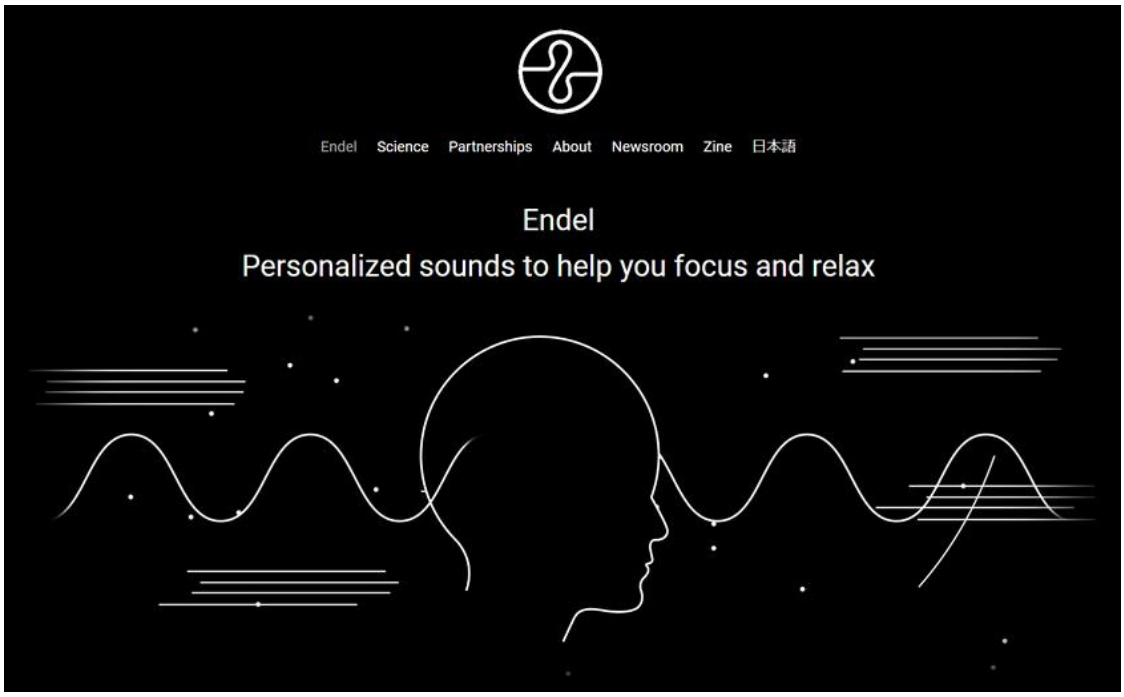

Рис. 5. Endel музика за допомогою ШІ.

Живопис за допомогою штучного інтелекту Google.

У 2015 році команда Google тестувала нейронні мережі на предмет можливості самостійно створювати зображення. Тоді штучний інтелект

навчали на прикладі великої кількості різних картинок. Однак, коли машину «попросили» самостійно що-небудь зобразити, то виявилося, що вона інтерпретує навколишній світ трохи дивно. Наприклад, на завдання намалювати гантелі, розробники отримали зображення, в якому метал був з'єднаний людськими руками. Ймовірно, сталося це через те, що на етапі навчання аналізовані картинки з гантелями містили руки, і нейронна мережа невірно це інтерпретувала.

26 лютого 2016 року в Сан-Франциско на спеціальному аукціоні представники Google виручили з психоделічних картин, написаних штучним інтелектом, близько $ 98 тис. Дані кошти були пожертвувані на доброчинність. Одна з найбільш вдалих картин машини представлена нижче.

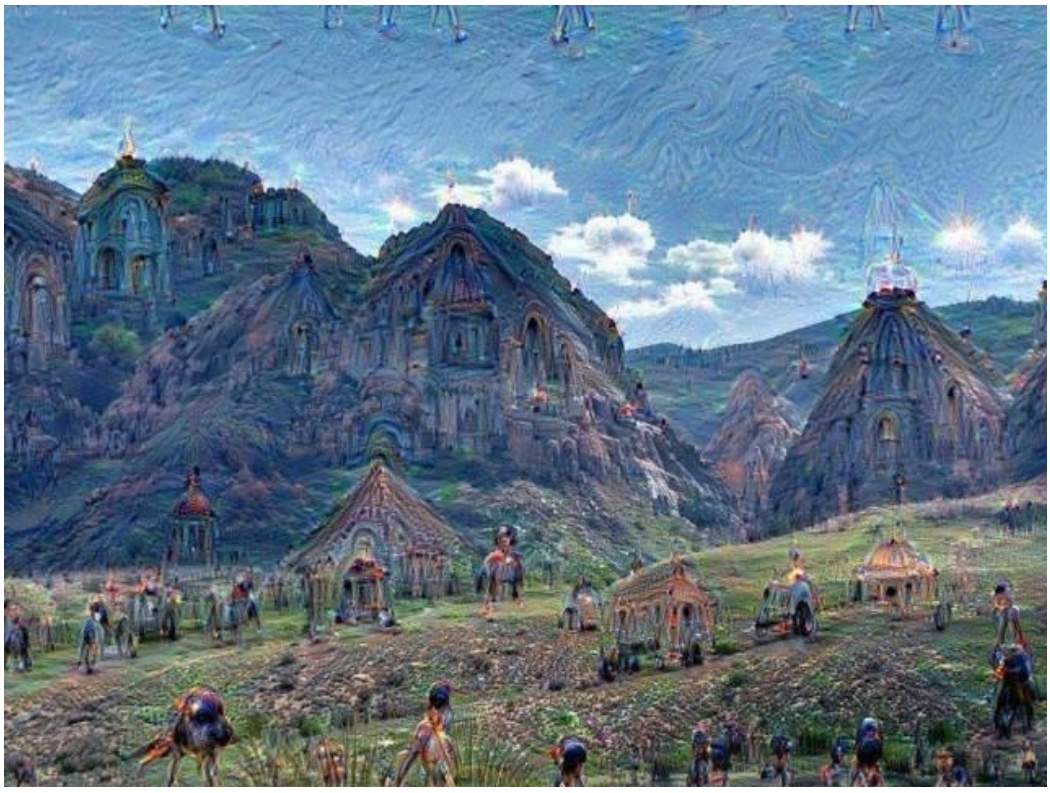

Рисунок 6. Картина, намальована штучним інтелектом Google

Штучний інтелект та контроль коронавірусу COVID-19.

Штучний інтелект (ШІ) використовується як інструмент підтримки боротьби з вірусною пандемією, що вразила весь світ з початку 2020 року. Преса і наукове співтовариство поділяють великі надії на те, що наука про дані

і ШІ можуть бути використані для боротьби з коронавірусом і "заповнити прогалини", залишені наукою.

Китай, перший епіцентр цієї хвороби і відомий своїми технологічними досягненнями в галузі ШІ, спробував використовувати це в своїх інтересах. Його застосування включало в себе підтримку заходів щодо обмеження пересування населення, прогнозування еволюції спалахів захворювань і проведення досліджень для розробки вакцини або лікування. Що стосується останнього аспекту, штучний інтелект використовується для прискорення секвенування геномів, більш швидкої постановки діагнозу, проведення аналізів за допомогою сканерів або, рідше, для керування роботами для технічного обслуговування і доставки.

Перше застосування штучного інтелекту, очікуване в умовах кризи в галузі охорони здоров'я, безсумнівно, полягає в наданні допомоги дослідникам в пошуку вакцини, здатної захистити осіб, які доглядають за хворими, і стримати пандемію. Біомедицина і наукові дослідження спираються на велику кількість методів, серед яких різні прикладні комп'ютерні науки і статистика вже давно вносять свій внесок. Тому використання штучного інтелекту є частиною цієї спадкоємності.

Прогнози структури вірусу, створеного ШІ, вже врятували вчених від багатомісячних експериментів. ШІ надав суттєву підтримку в цьому сенсі, навіть якщо вона обмежена через так званих "безперервних" правил і нескінченних комбінацій для вивчення згортання білків. Американський стартап Moderna відзначився своїм володінням біотехнологією, заснованої на мессенджерной рибонуклеїнової кислоти (мРНК), для якої вивчення згортання білка є необхідним. Завдяки підтримці біоінформатики, невід'ємною частиною якої є ШІ, їй вдалося значно скоротити час, необхідний для розробки прототипу вакцини, яка може бути протестована на людину.

Команда дослідників, що працюють з Бостонської дитячої лікарнею, також розробила ШІ, щоб відслідковувати розповсюдження коронавируса.

Система під назвою HealthMap об'єднує дані пошуків в системі Google, з соціальних мережах і блогах, а також з дискусійних форумів: джерел інформації, які епідеміологи зазвичай не використовують, але які корисні для виявлення перших ознак спалаху і оцінки заходів у відповідь з боку громадськості.

Зі свого боку, дві китайські компанії розробили програмне забезпечення для діагностики коронавірусів на основі ШІ. Стартап Infervision, розташований в Пекіні, навчив своє програмне забезпечення виявлення проблем з легкими використовуючи результати комп'ютерної томографії (КТ). Спочатку використовується для діагностики раку легенів, це програмне забезпечення може також виявляти пневмонію, пов'язану з респіраторними захворюваннями, такими як коронавірус. Щонайменше 34 китайські лікарні, як повідомляється, використовували цю технологію, щоб допомогти їм обстежити 32 000 передбачуваних випадків. [6]

2 МОДЕЛЬ ПРОЕКТУ ТА ПОСТАНОВКА ЗАДАЧІ

В основі реалізації проекту «Застосування ШІ на прикладі гри», лежить алгоритм еволюціонуючіх нейронних мереж зростаючих топологій, або скорочено NEAT, а також мова програмування Python.

Алгоритм NEAT.

Еволюціонующі нейронні мережі зростаючих топологій.

Нейронні мережі як правило використовуються на завданнях кластеризації і розпізнавання образів. І там вони показують дійсно вражаючі результати. Але в задачах розпізнавання образів структура нейронної мережі (її топологія) як правило задається заздалегідь. Навчанням ж такої мережі є налагодження ваг між заздалегідь певними верствами. Сам вибір топології мережі є дуже нетривіальним завданням, якє виникає ще задовго до навчання самої мережі.

Тому постало завдання, щоб мережа могла не тільки навчатися, а й сама налаштовувати свою топологію, створювати / видаляти вузли та зв'язки. Одним з таких алгоритмів є алгоритм NEAT.

Алгоритм.

Алгоритм NEAT - еволюційний алгоритм. Він дозволяє використовувати генетичні алгоритми для визначення кращої і мінімально необхідної топології нейронної мережі. У цьому алгоритмі нейронна мережа являє собою орієнтований граф. Нейрони можуть бути трьох видів - сенсори (вхідні), приховані, і вихідні. Зв'язки містять в собі номери нейронів, з якими вони пов'язані, ваги і порядковий номер. Кожен зв'язок може бути в двох станах: активному і неактивному.

Ген.

Набори (вектори) нейронів і зв'язків являють собою "Ген", яким оперують генетичний алгоритм.

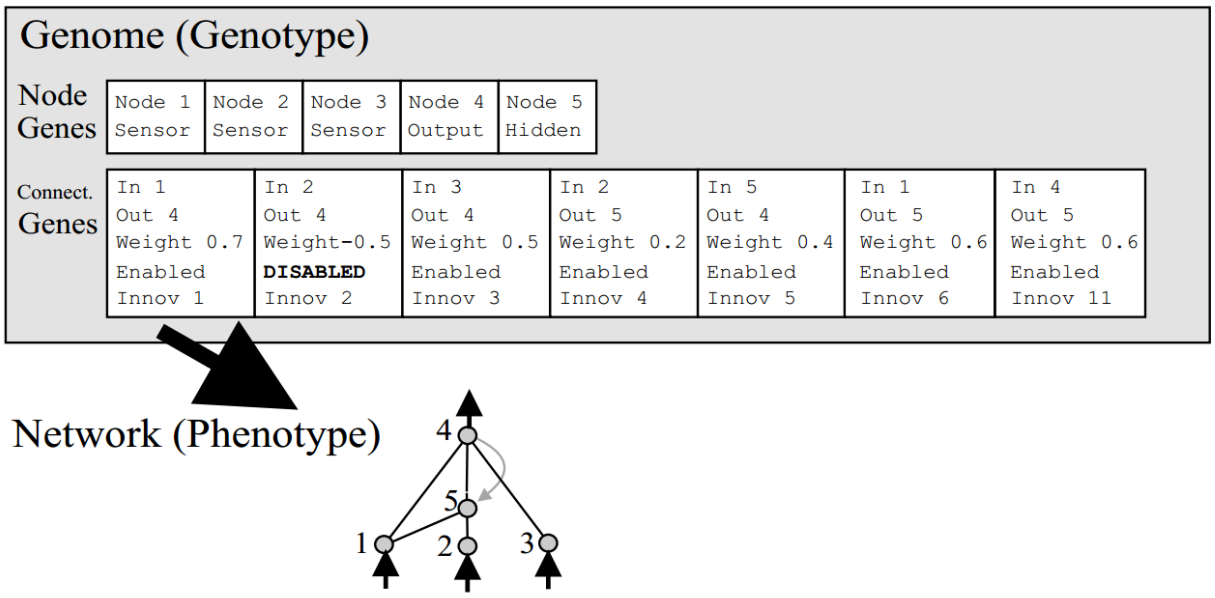

Рисунок 6. Приклад генотипу.

В наведеному вище прикладі прямого шляху з 2 в 4 немає, тому цей зв'язок позначена неактивною. Такі зв'язки не викидаються, а зберігаються.

Історичні маркери.

Кожен зв'язок має свій унікальний маркер (innovation number). Новий маркер може бути створений тільки при мутації. Це по суті ID зв'язку, але крім усього іншого він дозволяє відстежити як "вік" самої мережі, так і які вже мутації з нею відбувалися. Крім того це дозволяє значно спростити схрещування.

Мутація.

Мутації бувають двох видів. При першій мутації може додасться зв'язок до вже існуючих нейронам:

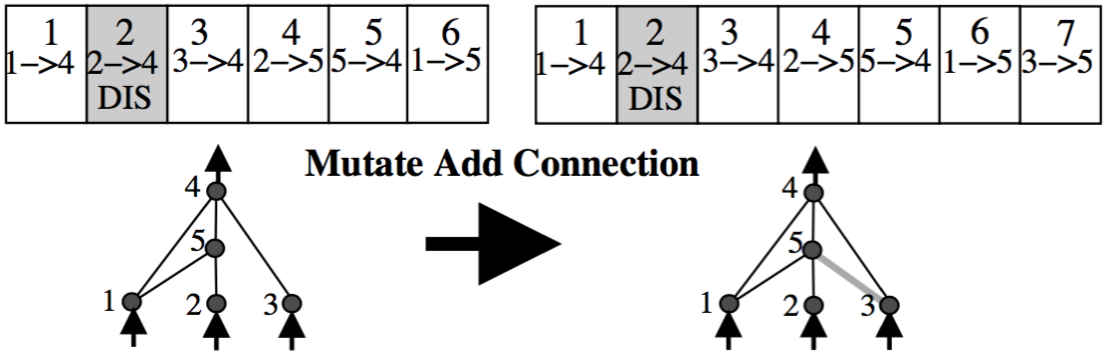

Рисунок 7. Додавання мутацій. Новий зв'язок 3-5.

При другому типі мутації створюється новий нейрон, на місці вже існуючого зв'язку між двома нейронами. При цьому старий зв'язок стає неактивним, і створюються два нових:

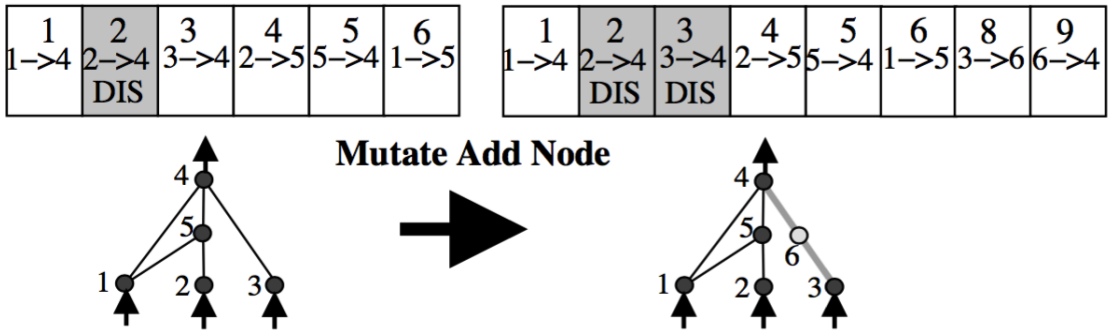

Рисунок 8. Два нових зв'язка (3-6) та (6-4). Зв'язок (3-4) став неактивним.

В обох випадках новим зв'язкам буде присвоєно новий історичний маркер (унікальне число).

Схрещування.

Саме ці ж історичні маркери використовуються при схрещуванні щоб зрозуміти як змішати два гена потрібно всього-лише в ген нащадок внести всі унікальні історичні маркери обох батьків. Якщо зв'язок неактивний у одного з батьків, то у нащадка він також буде не активним. Якщо зв'язок неактивний у обох батьків, то у нього є шанс мутувати і стати активним у нащадка.

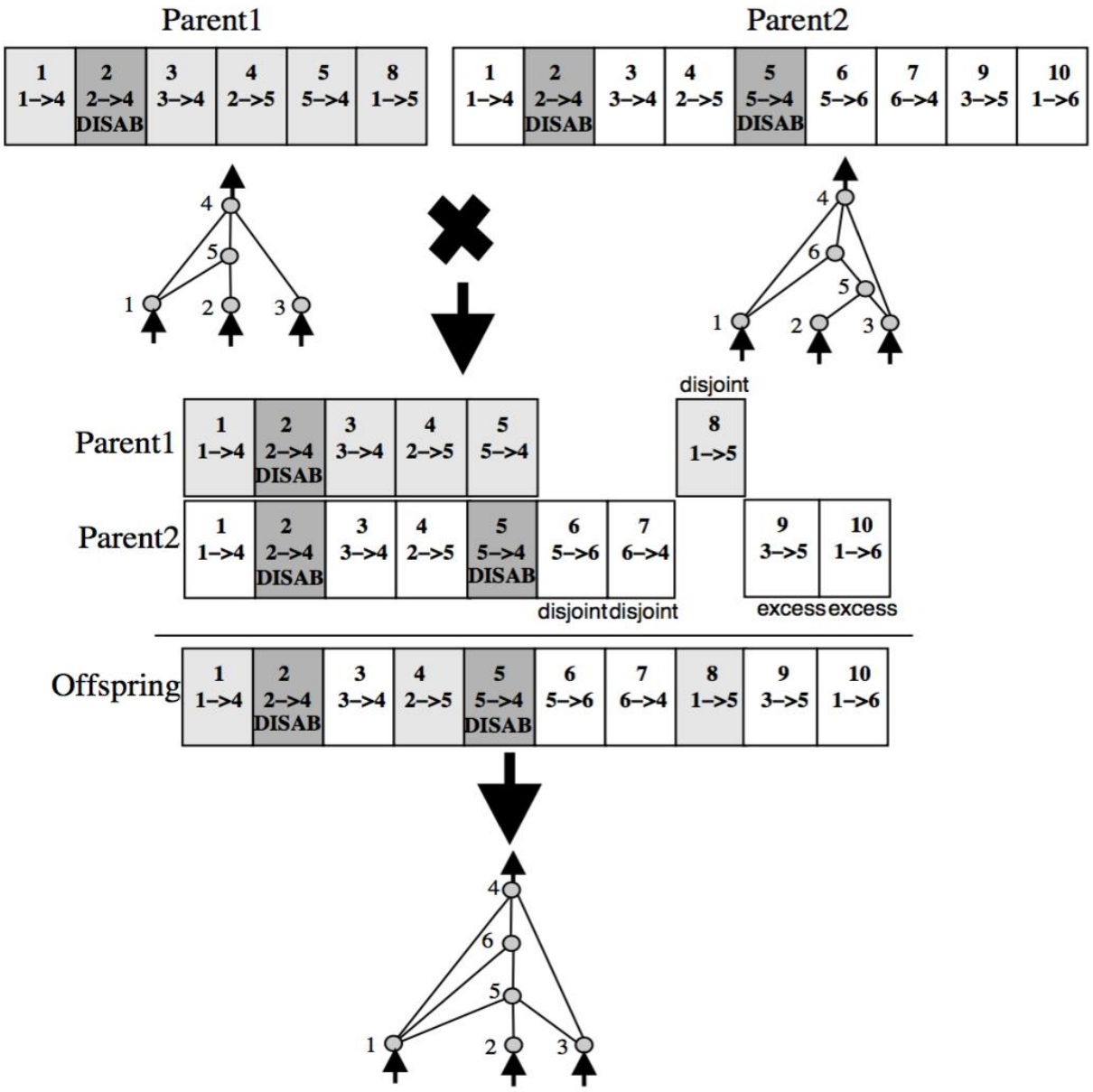

Рисунок 9. Схрещування генів двох батьків.

Генетичний алгоритм.

Далі, як і в будь-якому генетичному алгоритмі створюється початкова популяція. Спочатку кожна особина складається тільки з пронумерованих вхідних і вихідних нейронів. Кожен вихідний нейрон пов'язаний з вихідним. Тобто перед нами класичний одношаровий персептрон:

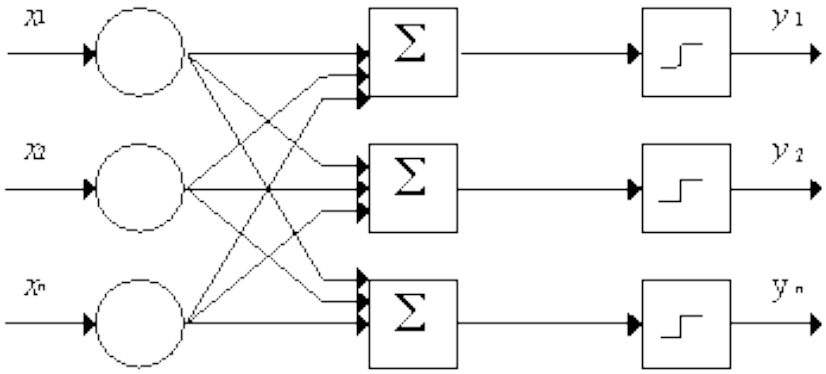

Рисунок 10. Одношаровий персептрон, де x (1..n) – входи, y (1..m) – виходи.

Далі до кожної особини застосовується фітнес-функція. Вона визначає те, наскільки дана особина підходить для виконання покладених на неї функцій. На входи подаються дані, і порівнюються з очікуваними виходами. Або проводиться симуляція, за допомогою ігор, фізичних движків, або фізичних стимуляторів роботів.

Ця фітнес функція ранжує всю популяцію. Ті, особини, які опинилися краще за інших матимуть великі ймовірності на "продовження роду", але навіть гірші мають невеликий статистичний шанс. Це допомагає уникнути скочування популяції в локальний мінімум / максимум.

Далі алгоритм відбирає особин з урахуванням отриманих ймовірностей і відбувається "схрещування". Саме тут алгоритм NEAT дуже гарний, бо має прості правила "двостатевого" схрещування генів. Найчастіше саме схрещування - найскладніша частина генетичних алгоритмів.

За допомогою схрещування генерується повністю нова популяція (в деяких версіях беруться кілька особин колишнього покоління і додаються в нове покоління). Після чого відбувається мутація - або значення ваг з певною ймовірністю змінюються, або, як в алгоритмі NEAT, випадковим чином додаються нові зв'язки / нейрони. Після чого цикл повторюється. Таких циклів / популяцій може бути дуже багато.

Плюсом отриманої нейронної мережі є те, що її топологія генерується на льоту, і швидше за все буде значно складніше, ніж статично задані.

З мінусів - така мережа буде заточена на певний вид завдань, і не матиме надмірності. Це може зменшити її пластичність, що негативно позначиться на сприйнятті невідомих вхідних даних. [7]

NEAT - це тільки один з алгоритмів, для побудови зростаючих нейронних мереж. Іншими широко відомими алгоритмами є HyperNEAT (варіація цього-ж алгоритму) і SOINN.

Мова програмування Python.

Python - високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читання коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Python підтримує структурне, об'єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване програмування. Основні архітектурні риси - динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень, високорівневі структури даних. Підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети.

Еталонної реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ [8]. Він поширюється під вільною ліцензією Python Software Foundation License, що дозволяє використовувати його без обмежень в будь-яких додатках, включаючи пропрієтарні.

Python - активно розвивається мова програмування, нові версії з додаванням / зміною мовних властивостей виходять приблизно раз в два з половиною роки. Язик не піддавався офіційній стандартизації, роль стандарту де-факто виконує CPython, що розробляється під контролем автора мови. На даний момент Python займає третє місце в рейтингу TIOBE з показником 8,5%.

Особливості Python.

Python, як і будь-яка інша мова програмування, має свої відмінні риси. Отже, можна виділити наступні:

- Кросплатформеність. Python – це інтерпретована мова, його інтерпретатори існують для багатьох платформ. Тому з запуском його на будь-який ОС не повинно виникнути проблем.
- З Python доступна величезна кількість сервісів, середовищ розробки, і фреймворків.
- Можливість підключити бібліотеки, написані на C. Це дозволяє підвищити ефективність, поліпшити швидкість роботи.
- Наявність самих різних джерел інформації про Python. Не важно буде знайти відповідь на питання, які виникають, так існую багато безкоштовної літератури, навчальних відео-посібників, готових початкових кодів та шаблонів для роботи у відкритому доступі.

Переваги Python щодо інших мов програмування.

Python легко конкурує з іншими мовами програмування, так як має безліч переваг. По-перше, це зрозумілий і простий синтаксис. Особливо добре він для новачків. Можна створити цікаві програми, і при цьому не доведеться сидіти тижнями, вивчаючи складний синтаксис.

Динамічна типізація - це одне з головних достоїнств мови Python. Для новачків це можливість спростити написання коду і уникнути безлічі фатальних помилок і багів в роботі. Також в Python немає операторних дужок, з розставленими яких найчастіше виникають складності.

За швидкістю виконання програм, коли це стосується великих повномасштабних проектів, Python, звичайно ж, не лідер. Тут мінусом є і автоматичне керування пам'яттю, і повна динамічна типізація. Python поступається значно таких мов як Java, C, C ++, але і в той же час з легкістю дає фору JavaScript, Ruby, PHP. Підключення бібліотек, написаних на C і

можливість попередньої компіляції коду в байт-код - все це дозволяє поліпшити швидкодію.

В кінці хочеться відзначити, що Python - це мова програмування, затребувана сьогодні і з великим потенціалом в майбутнє. Сьогодні ринок праці потребує кваліфікованих фахівців зі знаннями Python.

3 ВИБІР МЕТОДІВ ТА РЕАЛІЗАЦІЯ ПОСТАВЛЕНИХ ЗАДАЧ

Для реалізації даного проекту на мові програмування Python, нам знадобиться декілька бібліотек, а також знання для роботи з ними. Таким чином нам будуть потрібні наступні модулі і бібліотеки:

- модуль OS
- бібліотека PyGame
- модуль time
- модуль pickle
- модуль random
- бібліотека NEAT
- бібліотека visualize

Для того щоб підключити модуль або бібліотеку для проекту на Python, потрібно написати команду **import** + назва потрібного модуля. Більш детальніше про кожен з них.

OS (Operating System)

Модуль OS в Python - це бібліотека функцій для роботи з операційною системою. Методи, включені в неї дозволяють визначати тип операційної системи, отримувати доступ до змінних оточення, управляти директоріями і файлами:

- перевірка існування об’єкта по заданому шляху
- визначення розміру в байтах
- видалення
- перейменування та ін.

При виклику функцій OS необхідно враховувати, що деякі з них можуть не підтримуватися поточною ОС.

PyGame (Python Game)

PyGame- це бібліотека модулів для мови Python, створена для розробки 2D ігор. Також існує поняття "ігрового движка" як програмного середовища для розробки ігор. За своїм призначенням Pygame можна вважати ігровим движком. У той же час, з точки зору класифікації програмного забезпечення, Pygame є API для Пітона до API бібліотеки SDL. [9]

API - це інтерфейс (в основному набір функцій і класів) для прикладного (часто більш високорівневого) програмування, який надає, наприклад, та чи інша бібліотека. SDL - це бібліотека, яка працює з мультимедійними пристроями комп'ютера.

Ігровий цикл

У серці кожної гри лежить цикл, який прийнято називати «ігровим циклом». Він запускається знову і знову, роблячи все, щоб працювала гра. Кожен цикл в грі називається кадром. У кожному кадрі відбувається маса речей, але їх можна розбити на три категорії:

1. Обробка введення (події)

   Йдеться про все, що відбувається поза грою - ті події, на які вона повинна реагувати. Це можуть бути натискання клавіш на клавіатурі, кліки мишею і так далі.
2. Оновлення гри

   Зміна всього, що повинно змінюватися протягом одного кадру. Приклад - якщо персонаж в повітрі, гравітація повинна потягнути його вниз. Якщо два об’єкти зустрічаються на великій швидкості, вони повинні вибухнути.
3. Візуалізація (промальовування)

   У цьому кроці все виводиться на екран: фони, персонажі, меню. Все, що гравець повинен бачити, з’являється на екрані в потрібному місці.

Час (FPS)

Ще один важливий аспект ігрового циклу - швидкість його роботи. Існує такий термін як FPS, який розшифровується як Frames Per Second (або кадри в секунду). Він вказує на те, скільки разів цикл повинен повторитися за одну секунду. Це важливо, щоб гра не була надто повільною або швидкою. Важливо і те, щоб гра не працювала з різною швидкістю на різних ПК. Якщо персонажу необхідно 10 секунд на те, щоб перетнути екран, ці 10 секунд повинні бути незмінними для всіх комп'ютерів.

Модуль time

Робота методів з модуля time ґрунтується на загальноприйнятій системі опису часу, яка справедлива для Unix, а також для POSIX-сумісних ОС. Згідно з її концепції, даний час є у вигляді звичайного речового значення в секундах, що пройшли з моменту початку епохи і до сьогоднішнього дня. Відправною точкою для часу вважається 1 січня 1970 року, коли лічильник секунд мав повністю нульове значення.

З тих пір це число постійно зростає, дозволяючи людям працювати з різними видами дат в максимально точному поданні. У разі необхідності секунди переводяться в години, дні, місяці і роки, надаючи користувачу зручний вид відображення часу. Щоб побачити кількість секунд, що пройшли з моменту початку епохи, досить викликати метод **time ()**.

Модуль pickle

Модуль pickle надає функції і класи для серіалізації і десеріалізації об'єктів. У програмуванні під серіалізацією розуміють перетворення будь-яких даних в набір байтів, який потім зазвичай зберігають в файл або передають по мережі. Десеріалізація - це відновлення об'єктів з їх байтових уявлень.

Часто серіалізація використовується для збереження призначених для користувача даних між різними сесіями роботи програми, зазвичай гри. Більш

простий приклад - ви працюєте в інтерактивному режимі і створили список або словник, який будете використовувати в наступний раз. Для цього за допомогою функції **dump ()** модуля pickle ви зберігаєте об'єкт в файл, а з допомогу **load ()** відновлюєте в наступний раз.

Модуль random надає функції для генерації випадкових чисел, букв, випадкового вибору елементів послідовності.

Бібліотека NEAT

NEAT (NeuroEvolution of Augmenting Topologies) - це еволюційний алгоритм, який створює штучні нейронні мережі.

У поточній реалізації NEAT-Python підтримується популяція окремих геномів. Кожен геном містить два набори генів, які описують, як побудувати штучну нейронну мережу:

- Вузлові гени, в кожному з яких зазначено один нейрон.
- Сполучні гени, кожен з яких задає один зв’язок між нейронами.

Щоб знайти рішення проблеми, користувач повинен надати фітнес-функцію, яка обчислює одне дійсне число, яке вказує якість окремого генома: краща здатність вирішувати проблему означає більш високий бал. Алгоритм просувається через вказане користувачем кількість поколінь, причому кожне покоління створюється шляхом розмноження і мутації найбільш підходящих особин попереднього покоління.

Операції з відтворення та мутації можуть додавати вузли та / або з'єднання до геномів, так як алгоритм, який протікає в геномах (і нейронні мережі, які вони виробляють), можуть ставати все більш складними. Коли досягнуто встановленої кількості поколінь або коли принаймні один індивід (для функції критерію придатності max; інші налаштовуються) перевищує визначений користувачем поріг, алгоритм припиняється.

Одна складність цього налаштування полягає у здійсненні кросовера - NEAT обробляє це шляхом відстеження походження вузлів, з ідентифікаційним номером (нові, більш високі числа генеруються для кожного додаткового вузла). Ті, що походять від спільного предка (які є гомологічними), узгоджуються для схрещування, а з'єднання узгоджуються, якщо вузли, які вони з'єднують, мають спільне походження. [10]

Visualize це додатковий допоміжний модуль для бібліотеки NEAT

Flappy Bird як приклад використовування нейронної мережі.

Розробка прикладу штучної нейронної мережі, буде проводиться на прикладі гри Flappy Bird.

Flappy Bird - гра для мобільних пристроїв, розроблена в'єтнамським розробником Донг Нгуєном, в якій гравець з допомогою торкань екрана повинен контролювати політ птаха між рядами зелених труб, не зачіпаючи їх. Була реалізована на платформах iOS і Android [11].

Flappy Bird має ігровий процес за участю 2D-графіки. Мета гри полягає в управлінні польотом птаха, яка безперервно пересувається між рядами зелених труб. При зіткненні з ними відбувається завершення гри. Управління проводиться дотиком екрану, при якому птах робить невеликий ривок вгору. При відсутності ривків птах падає через сили тяжіння, і гру буде теж завершено. Бали набираються при кожному успішному перельоті між двома трубами. Геймплей не має змін протягом всієї гри.

Штучна нейронна мережа - це підмножина алгоритму машинного навчання. За основу в ній взято структура і функції біологічних нейронних мереж. Ці мережі створені з безлічі нейронів, що передають сигнали один одному.

Стандартна штучна нейронна мережа складається з шару вхідних даних, одного або декількох прихованих шарів і шару вихідних даних. У кожному шарі є кілька нейронів. Нейрони вхідних і вихідних даних приєднані безпосередньо до зовнішнього середовища. Приховані нейрони з'єднуються між ними.

У цьому проекті кожен об'єкт (птах) має власну нейронну мережу, яка використовується в якості ШІ-мозку для проходження гри. Вона складається з наступних двох шарів:

1. шар вхідних даних з трьома нейронами представляє те, що бачить птах:
   - позиція птаха до найближчого проміжку
   - відстань до верхнього краю труби
   - відстань до нижнього краю труби
2. шар вихідних даних з одним нейроном, що створює дію:

- якщо вихідні дані> 0,5, то зробити стрибок, в іншому випадку не робити нічого

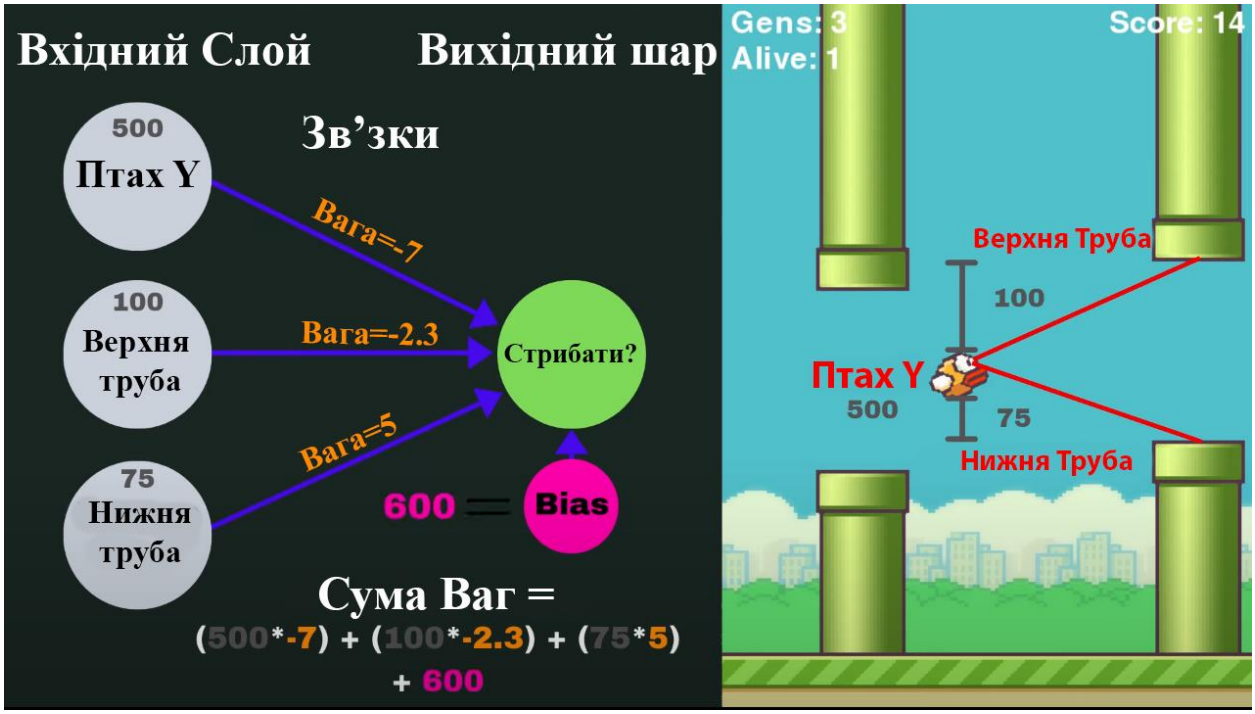

Рисунок 11. Нейронна мережа Flappy Bird.

Нейрон зміщення або bias нейрон - це третій вид нейронів, що використовується в більшості нейромереж. Особливість цього типу нейронів полягає в тому, що його вхід і вихід завжди дорівнюють 1 і вони ніколи не мають вхідних синапсів. Нейрони зміщення можуть, або бути присутнім в нейронної мережі по одному на шарі, або повністю відсутні, 50/50 бути не може. З'єднання у нейронів зміщення такі ж, як у звичайних нейронів - з усіма нейронами наступного рівня, за винятком того, що синапсів між двома bias нейронами бути не може. Отже, їх можна розміщувати на вхідному шарі і всіх прихованих шарах, але ніяк не на вихідному шарі, так як їм просто нема з чим буде формувати зв'язок.

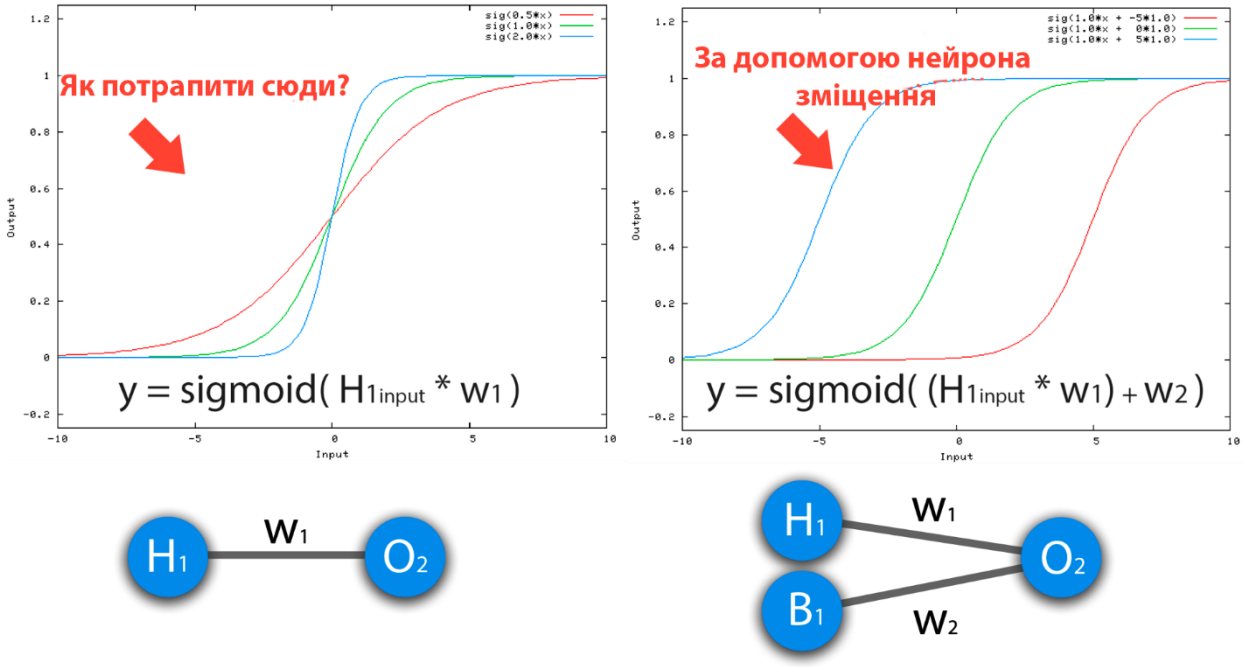

Рисунок 12. Зміщення за допомогою bias.

Нейрон зміщення потрібен для того, щоб мати можливість отримувати вихідний результат, шляхом зсуву графіка функції активації вправо або вліво.

Запускаємо початкову популяцію.

Рисунок 13. Початкова популяція.

Кожен птах у популяції має свою нейронну мережу.

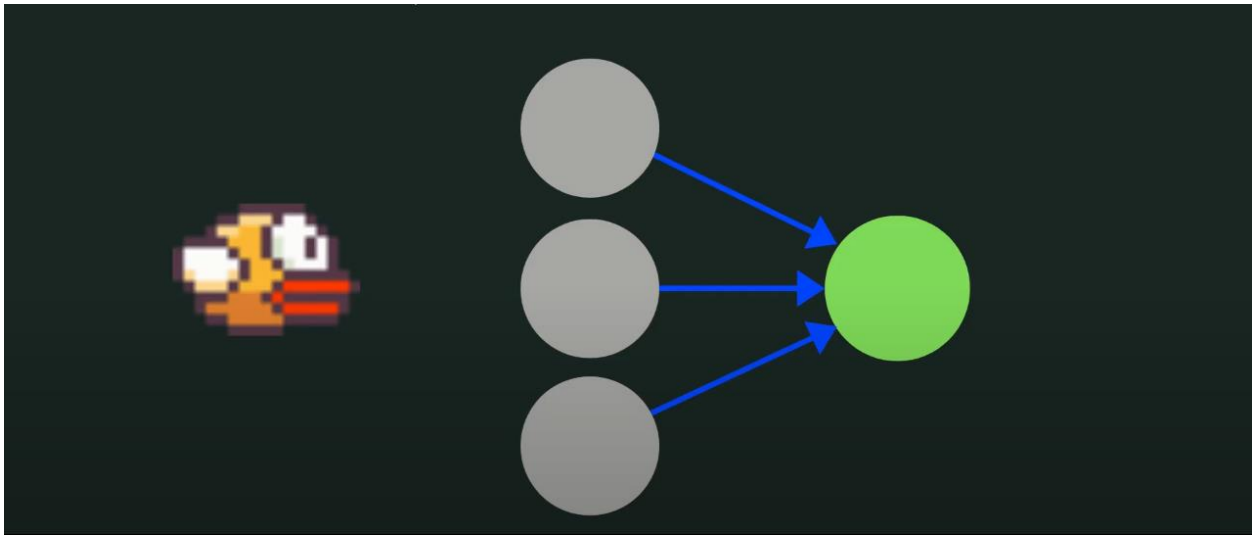

Рисунок 14. Нейронна мережа птахи.

Ми навчаємо кожну з цих мереж і оцінюємо їх придатність.

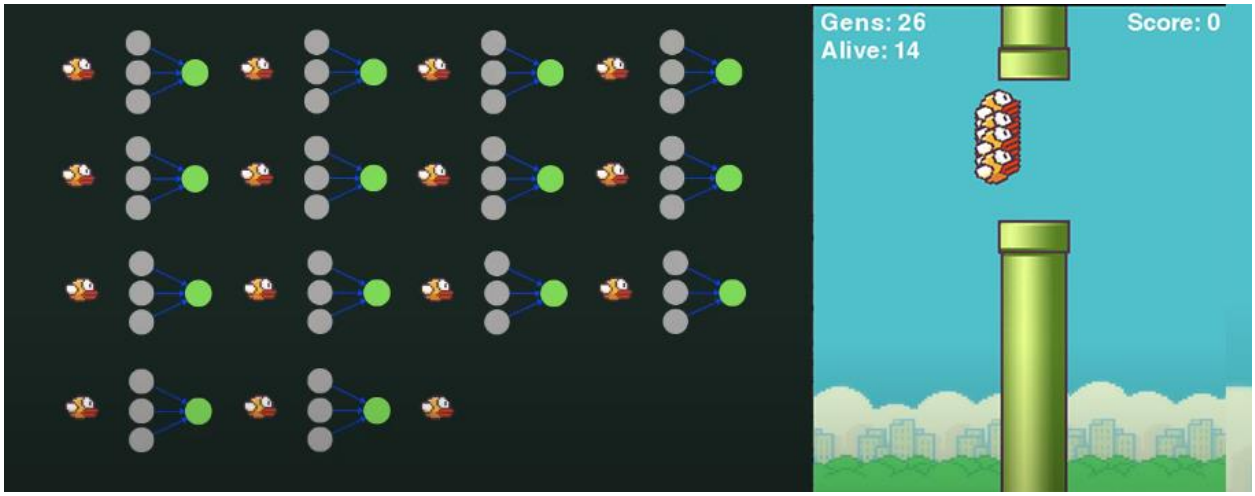

Рисунок 15. Багато птахів – багато нейронних мереж.

Найкращі з попередньої популяції, мутують та створюють нові мережі.

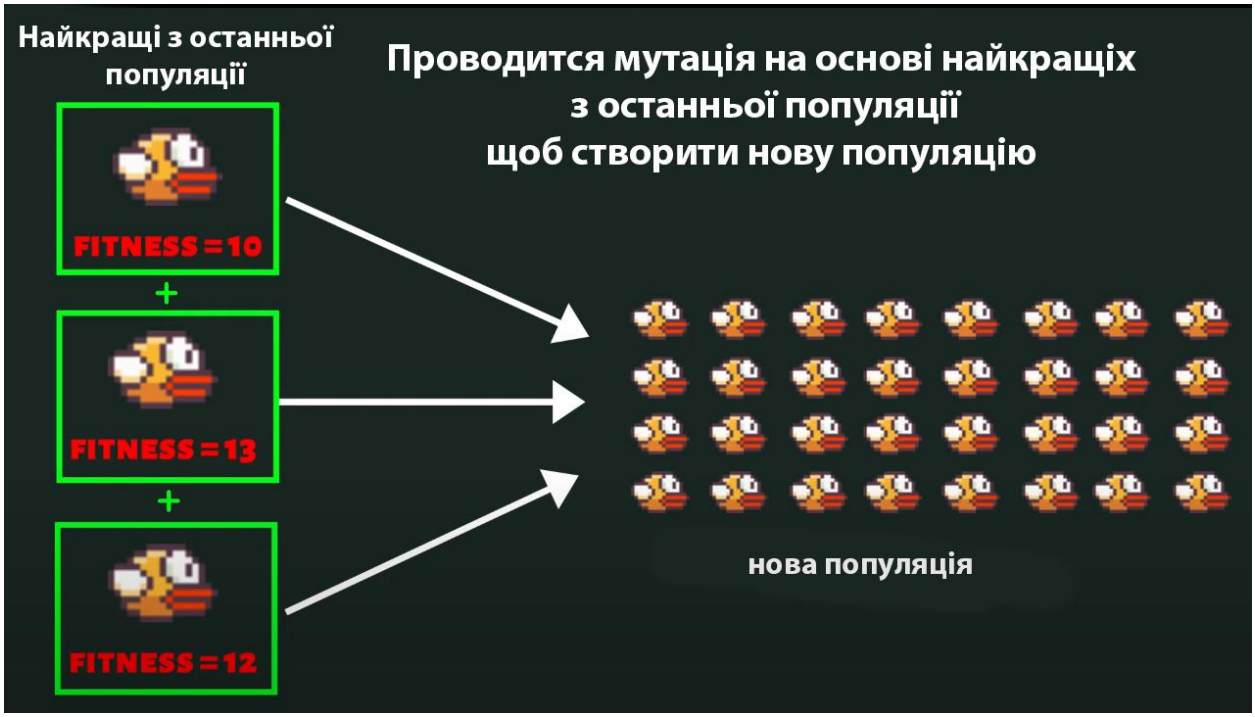

Рисунок 16. Мутація нових нейронних мереж.

Таким чином мутація продовжується до тих пір, поки не вийде ідеальна популяція яка буду ідеально виконувати поставлене завдання.

Генетичний алгоритм

Коли ми обговорювали алгоритм машинного навчання, то говорили, що для навчання і вдосконалення нейронних мереж використовується генетичний алгоритм.

Генетичний алгоритм - це техніка оптимізації на основі пошуку, яка копіює природний відбір і генетику. У ній використовується таке ж поєднання відбору, кросинговеру і мутації для зміни вихідної випадкової популяції.

Ось основні етапи реалізації нашого генетичного алгоритму:

1. створюємо вихідну популяцію з 10 об'єктів (птахів) з випадковими нейронними мережами
2. даємо всім об'єктам грати одночасно з використанням їх власних нейронних мереж

3. у кожного об'єкта обчислюємо його функцію пристосованості для оцінки його якості (докладніше див. у розділі Функція пристосованості)
4. після смерті всіх об'єктів оцінюємо поточне покоління для створення нового за допомогою генетичних операторів
5. повертаємося до етапу 2

Функція пристосованості

Оскільки ми хочемо, щоб популяція еволюціонувала з найкращих об'єктів, нам необхідно визначити функцію пристосованості. У загальному випадку, функція пристосованості - це метрика, що вимірює якість об'єкта. Якщо у нас буде метрика якості кожного птаха, ми зможемо вибрати найбільш пристосовані об'єкти і використовувати їх для відтворення наступного покоління.

У цьому проекті ми винагороджуємо птицю в прямій залежності від зробленого відстані. Крім того, ми караємо її за поточним відстані до найближчого проміжку. Таким чином ми зможемо розрізняти птахів, які пролетіли однакова відстань.

Підіб'ємо підсумок: наша функція пристосованості - це різниця між загальною відстанню, пророблених птахом, і поточним відстанню до найближчого проміжку.

Стратегія заміни

На додаток до генетичного алгоритму, ось етапи застосування природної еволюції до вмираючого поколінню. Виживають кращі об'єкти, а їхні нащадки заміняють найгірші об'єкти наступним чином:

1. сортуємо об'єкти поточного покоління по їх рівню пристосованості
2. вибираємо чотири кращих об'єкта (переможців) і передаємо їх безпосередньо в наступне покоління

3. створюємо одного нащадка як результат кросинговеру між двома найкращими переможцями
4. створюємо трьох нащадків як результати кросинговеру двох випадкових переможців
5. створюємо двох нащадків як прямі копії двох випадкових переможців
6. застосовуємо до кожного нащадку випадкові мутації, щоб додати варіативності

Проект дипломної роботи складається з наступних файлів:

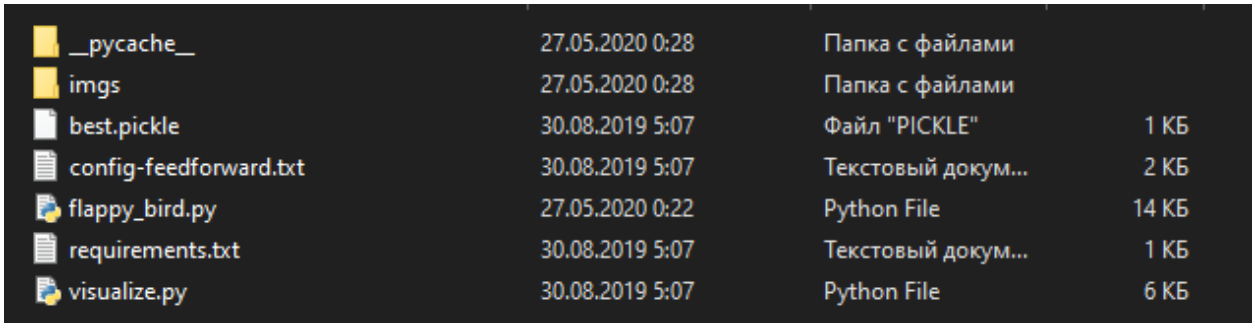

Рисунок 17. Папка дипломної роботи.

- папка imgs містить візуальне оформлення гри
- файл best.pickle містить данні о найкращій популяції
- flappy_bird.py основний файл проекту
- requirements.txt список потрібних для установки модулів та бібліотек
- visualize.py допоміжний файл проекту

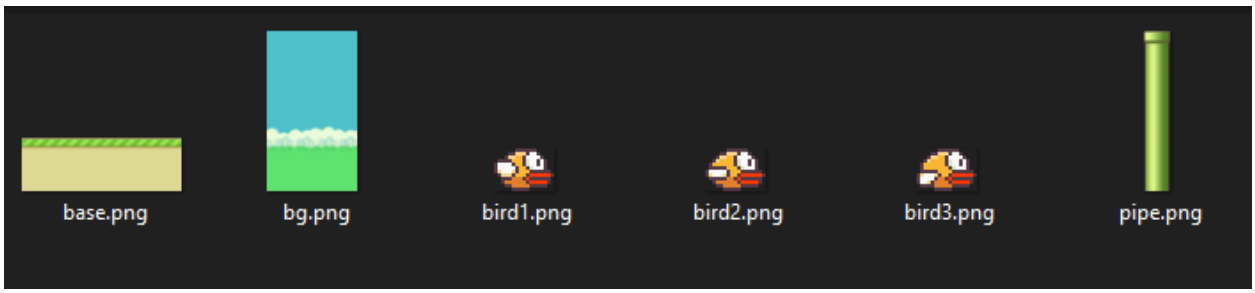

Рисунок 18. Візуальне оформлення гри.

Імпортуємо потрібні нам бібліотеки та модулі:

```
import pygame
import random
import os
import time
import neat
import visualize
import pickle
```

Рисунок 19. Підключення бібліотек та модулів.

Створюємо вікно ігри з параметрами (600 px, 800px), а також підключаємо зображення:

```
pygame.font.init()  # init font

WIN_WIDTH = 600
WIN_HEIGHT = 800
FLOOR = 730
STAT_FONT = pygame.font.SysFont("comicsans", 50)
END_FONT = pygame.font.SysFont("comicsans", 70)
DRAW_LINES = False

WIN = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT))
pygame.display.set_caption("Flappy Bird")

pipe_img = pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","pipe.png")).convert_alpha())
bg_img = pygame.transform.scale(pygame.image.load(os.path.join("imgs","bg.png")).convert_alpha(), (600, 900))
bird_images = [pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","bird" + str(x) + ".png"))) for x in range(1,4)]
base_img = pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","base.png")).convert_alpha())
```

Рисунок 19.1 Створення вікна ігри.

Створюємо клас «Птах» з усіма параметрами потрібними йому, надаємо можливість стрибати, пересуватися та інше.

```
class Bird:
    """
    Bird class representing the flappy bird
    """
    MAX_ROTATION = 25
    IMGS = bird_images
    ROT_VEL = 20
    ANIMATION_TIME = 5

    def __init__(self, x, y):
        """
        Initialize the object
        :param x: starting x pos (int)
        :param y: starting y pos (int)
        :return: None
        """
        self.x = x
        self.y = y
        self.tilt = 0  # degrees to tilt
        self.tick_count = 0
        self.vel = 0
        self.height = self.y
        self.img_count = 0
        self.img = self.IMGS[0]

    def jump(self):
        """
        make the bird jump
        :return: None
        """
        self.vel = -10.5
        self.tick_count = 0
        self.height = self.y
```

Рисунок 19.2. Клас «Птах».

Також нам потрібен клас який буде описувати «труби» через які птах повинен пролітати.

```
class Pipe():
    """
    represents a pipe object
    """
    GAP = 200
    VEL = 5

    def __init__(self, x):
        """
        initialize pipe object
        :param x: int
        :param y: int
        :return" None
        """
        self.x = x
        self.height = 0

        # where the top and bottom of the pipe is
        self.top = 0
        self.bottom = 0

        self.PIPE_TOP = pygame.transform.flip(pipe_img, False, True)
        self.PIPE_BOTTOM = pipe_img

        self.passed = False

        self.set_height()

    def set_height(self):
        """
        set the height of the pipe, from the top of the screen
        :return: None
        """
        self.height = random.randrange(50, 450)
        self.top = self.height - self.PIPE_TOP.get_height()
        self.bottom = self.height + self.GAP

    def move(self):
        """
        move pipe based on vel
        :return: None
        """
        self.x -= self.VEL
```

Рисунок 19.3. Клас «Pipe»

Функція яка запускає NEAT алгоритм, з параметрами указаними у файлі конфігурації.

```
def run(config_file):
    """
    runs the NEAT algorithm to train a neural network to play flappy bird.
    :param config_file: location of config file
    :return: None
    """
    config = neat.config.Config(neat.DefaultGenome, neat.DefaultReproduction,
                                neat.DefaultSpeciesSet, neat.DefaultStagnation,
                                config_file)

    # Create the population, which is the top-level object for a NEAT run.
    p = neat.Population(config)

    # Add a stdout reporter to show progress in the terminal.
    p.add_reporter(neat.StdOutReporter(True))
    stats = neat.StatisticsReporter()
    p.add_reporter(stats)
    #p.add_reporter(neat.Checkpointer(5))

    # Run for up to 50 generations.
    winner = p.run(eval_genomes, 50)

    # show final stats
    print('\nBest genome:\n{!s}'.format(winner))
```

Рисунок 19.4. Частина функції NEAT алгоритму.

ВИСНОВКИ

Нейронні мережі являють собою нову і дуже перспективну обчислювальну технологію, що дає нові підходи до дослідження динамічних задач у фінансовій області. Спочатку нейронні мережі відкрили нові можливості в області розпізнавання образів, потім до цього додалися статистичні і засновані на методах штучного інтелекту засоби підтримки прийняття рішень і вирішення завдань у сфері фінансів та інших.

В процесі виконання дипломної роботи було розглянуто питання створення штучного інтелекту, вибору найбільш вдалої мови програмування, яка найбільше підходить під реалізацію подібних проектів.

У цьому проекті ми успішно реалізували ШІ-робота для навчання грі у Flappy Bird. В результаті декількох ітерацій ми можемо отримати майже невразливого гравця. Для досягнення цієї мети ми використовували два підходи до алгоритмів машинного навчання: штучні нейронні мережі і генетичний алгоритм.

На етапах створення штучного інтелекту, для гри, так само були поміченими і недоліки даного методу, які в подальшому можна виправити, якщо використовувати більш нові методи вирішення подібних проблем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Стівен Хокінг об ШІ – tadviser.ru [Електронний ресурс] - Режим доступу: - http://www.tadviser.ru/index.php
2. Ілон Маск об ШІ - itc.ua [Електронний ресурс] - Режим доступу: https://itc.ua/blogs/ilon-mask-iskusstvennyiy-intellekt-rano-ili-pozdno-prikonchit-vseh-nas/
3. Recruit Holdings– tadviser.ru [Електронний ресурс] - Режим доступу: http://www.tadviser.ru/index.php/Company:F:Recruit_Holdings
4. Вплив штучного інтелекту на ринок праці – tadviser.ru [Електронний ресурс] - Режим доступу: http://www.tadviser.ru/index.php/IIonwork
5. Штучний інтелект і створення музики- tadviser.ru [Електронний ресурс] - http://www.tadviser.ru/index.php/musicandai
6. ШІ та контроль коронавируса Ковід-19- coe.int [Електронний ресурс] - Режим доступу: https://www.coe.int/en/web/artificial-intelligence/ii-i-kontrol-koronavirusa-kovid-19
7. Алгоритм NEAT- nut-code-monkey.blogspot.com [Електронний ресурс] - Режим доступу: https://nut-code-monkey.blogspot.com/2016/04/NEAT-algorithm.html
8. CPython та Python – python.org [Електронний ресурс] - Режим доступу: https://www.python.org/about/
9. PyGame бібліотека для ігор - pygame.org [Електронний ресурс] - Режим доступу: https://www.pygame.org/news2/
10. NEAT алгоритм – neat-python.readthedocs.io [Електронний ресурс] - Режим доступу: https://neat-python.readthedocs.io/en/latest/neat_overview.html
11. FLAPPY BIRD це - wikipedia.org [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Flappy_Bird

ДОДАТКИ

ДОДАТОК А. ЗМІСТ ОСНОВНОГО ФАЙЛУ ПРОЕКТУ

```
import pygame
import random
import os
import time
import neat
import visualize
import pickle
pygame.font.init()  # init font

WIN_WIDTH = 600
WIN_HEIGHT = 800
FLOOR = 730
STAT_FONT = pygame.font.SysFont("comicsans", 50)
END_FONT = pygame.font.SysFont("comicsans", 70)
DRAW_LINES = False

WIN = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT))
pygame.display.set_caption("Flappy Bird")

pipe_img =
pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","pipe.png")).convert_a
lpha())
bg_img =
pygame.transform.scale(pygame.image.load(os.path.join("imgs","bg.png")).convert_alpha
(), (600, 900))
bird_images = [pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","bird" +
str(x) + ".png"))) for x in range(1,4)]
base_img =
pygame.transform.scale2x(pygame.image.load(os.path.join("imgs","base.png")).convert_a
lpha())
```

```
gen = 0


class Bird:
    """
    Bird class representing the flappy bird
    """
    MAX_ROTATION = 25
    IMGS = bird_images
    ROT_VEL = 20
    ANIMATION_TIME = 5

    def __init__(self, x, y):
        """
        Initialize the object
        :param x: starting x pos (int)
        :param y: starting y pos (int)
        :return: None
        """
        self.x = x
        self.y = y
        self.tilt = 0  # degrees to tilt
        self.tick_count = 0
        self.vel = 0
        self.height = self.y
        self.img_count = 0
        self.img = self.IMGS[0]

    def jump(self):
        """
        make the bird jump
        :return: None
        """
```

```
        self.vel = -10.5
        self.tick_count = 0
        self.height = self.y

    def move(self):
        """
        make the bird move
        :return: None
        """
        self.tick_count += 1

        # for downward acceleration
        displacement = self.vel*(self.tick_count) + 0.5*(3)*(self.tick_count)**2  # calculate
displacement

        # terminal velocity
        if displacement >= 16:
            displacement = (displacement/abs(displacement)) * 16

        if displacement < 0:
            displacement -= 2

        self.y = self.y + displacement

        if displacement < 0 or self.y < self.height + 50:  # tilt up
            if self.tilt < self.MAX_ROTATION:
                self.tilt = self.MAX_ROTATION
        else:  # tilt down
            if self.tilt > -90:
                self.tilt -= self.ROT_VEL

    def draw(self, win):
```

```
        """
        draw the bird
        :param win: pygame window or surface
        :return: None
        """
        self.img_count += 1

        # For animation of bird, loop through three images
        if self.img_count <= self.ANIMATION_TIME:
            self.img = self.IMGS[0]
        elif self.img_count <= self.ANIMATION_TIME*2:
            self.img = self.IMGS[1]
        elif self.img_count <= self.ANIMATION_TIME*3:
            self.img = self.IMGS[2]
        elif self.img_count <= self.ANIMATION_TIME*4:
            self.img = self.IMGS[1]
        elif self.img_count == self.ANIMATION_TIME*4 + 1:
            self.img = self.IMGS[0]
            self.img_count = 0

        # so when bird is nose diving it isn't flapping
        if self.tilt <= -80:
            self.img = self.IMGS[1]
            self.img_count = self.ANIMATION_TIME*2


        # tilt the bird
        blitRotateCenter(win, self.img, (self.x, self.y), self.tilt)

    def get_mask(self):
        """
        gets the mask for the current image of the bird
```

```
        :return: None
        """
        return pygame.mask.from_surface(self.img)


class Pipe():
    """
    represents a pipe object
    """
    GAP = 200
    VEL = 5

    def __init__(self, x):
        """
        initialize pipe object
        :param x: int
        :param y: int
        :return" None
        """
        self.x = x
        self.height = 0

        # where the top and bottom of the pipe is
        self.top = 0
        self.bottom = 0

        self.PIPE_TOP = pygame.transform.flip(pipe_img, False, True)
        self.PIPE_BOTTOM = pipe_img

        self.passed = False

        self.set_height()
```

```
def set_height(self):
    """
    set the height of the pipe, from the top of the screen
    :return: None
    """
    self.height = random.randrange(50, 450)
    self.top = self.height - self.PIPE_TOP.get_height()
    self.bottom = self.height + self.GAP

def move(self):
    """
    move pipe based on vel
    :return: None
    """
    self.x -= self.VEL

def draw(self, win):
    """
    draw both the top and bottom of the pipe
    :param win: pygame window/surface
    :return: None
    """
    # draw top
    win.blit(self.PIPE_TOP, (self.x, self.top))
    # draw bottom
    win.blit(self.PIPE_BOTTOM, (self.x, self.bottom))


def collide(self, bird, win):
    """
    returns if a point is colliding with the pipe
```

```
        :param bird: Bird object
        :return: Bool
        """
        bird_mask = bird.get_mask()
        top_mask = pygame.mask.from_surface(self.PIPE_TOP)
        bottom_mask = pygame.mask.from_surface(self.PIPE_BOTTOM)
        top_offset = (self.x - bird.x, self.top - round(bird.y))
        bottom_offset = (self.x - bird.x, self.bottom - round(bird.y))

        b_point = bird_mask.overlap(bottom_mask, bottom_offset)
        t_point = bird_mask.overlap(top_mask,top_offset)

        if b_point or t_point:
            return True

        return False

class Base:
    """
    Represnts the moving floor of the game
    """
    VEL = 5
    WIDTH = base_img.get_width()
    IMG = base_img

    def __init__(self, y):
        """
        Initialize the object
        :param y: int
        :return: None
        """
        self.y = y
```

```
        self.x1 = 0
        self.x2 = self.WIDTH

    def move(self):
        """
        move floor so it looks like its scrolling
        :return: None
        """
        self.x1 -= self.VEL
        self.x2 -= self.VEL
        if self.x1 + self.WIDTH < 0:
            self.x1 = self.x2 + self.WIDTH

        if self.x2 + self.WIDTH < 0:
            self.x2 = self.x1 + self.WIDTH

    def draw(self, win):
        """
        Draw the floor. This is two images that move together.
        :param win: the pygame surface/window
        :return: None
        """
        win.blit(self.IMG, (self.x1, self.y))
        win.blit(self.IMG, (self.x2, self.y))


def blitRotateCenter(surf, image, topleft, angle):
    """
    Rotate a surface and blit it to the window
    :param surf: the surface to blit to
    :param image: the image surface to rotate
    :param topLeft: the top left position of the image
```

```
    :param angle: a float value for angle
    :return: None
    """
    rotated_image = pygame.transform.rotate(image, angle)
    new_rect = rotated_image.get_rect(center = image.get_rect(topleft = topleft).center)

    surf.blit(rotated_image, new_rect.topleft)

def draw_window(win, birds, pipes, base, score, gen, pipe_ind):
    """
    draws the windows for the main game loop
    :param win: pygame window surface
    :param bird: a Bird object
    :param pipes: List of pipes
    :param score: score of the game (int)
    :param gen: current generation
    :param pipe_ind: index of closest pipe
    :return: None
    """
    if gen == 0:
        gen = 1
    win.blit(bg_img, (0,0))

    for pipe in pipes:
        pipe.draw(win)

    base.draw(win)
    for bird in birds:
        # draw lines from bird to pipe
        if DRAW_LINES:
            try:
```

```
            pygame.draw.line(win, (255,0,0), (bird.x+bird.img.get_width()/2, bird.y +
bird.img.get_height()/2), (pipes[pipe_ind].x + pipes[pipe_ind].PIPE_TOP.get_width()/2,
pipes[pipe_ind].height), 5)
            pygame.draw.line(win, (255,0,0), (bird.x+bird.img.get_width()/2, bird.y +
bird.img.get_height()/2), (pipes[pipe_ind].x +
pipes[pipe_ind].PIPE_BOTTOM.get_width()/2, pipes[pipe_ind].bottom), 5)
        except:
            pass
    # draw bird
    bird.draw(win)

  # score
  score_label = STAT_FONT.render("Score: " + str(score),1,(255,255,255))
  win.blit(score_label, (WIN_WIDTH - score_label.get_width() - 15, 10))

  # generations
  score_label = STAT_FONT.render("Gens: " + str(gen-1),1,(255,255,255))
  win.blit(score_label, (10, 10))

  # alive
  score_label = STAT_FONT.render("Alive: " + str(len(birds)),1,(255,255,255))
  win.blit(score_label, (10, 50))

  pygame.display.update()



def eval_genomes(genomes, config):
  """
  runs the simulation of the current population of
  birds and sets their fitness based on the distance they
  reach in the game.
  """
```

```
global WIN, gen
win = WIN
gen += 1

# start by creating lists holding the genome itself, the
# neural network associated with the genome and the
# bird object that uses that network to play
nets = []
birds = []
ge = []
for genome_id, genome in genomes:
    genome.fitness = 0  # start with fitness level of 0
    net = neat.nn.FeedForwardNetwork.create(genome, config)
    nets.append(net)
    birds.append(Bird(230,350))
    ge.append(genome)

base = Base(FLOOR)
pipes = [Pipe(700)]
score = 0

clock = pygame.time.Clock()

run = True
while run and len(birds) > 0:
    clock.tick(30)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
            pygame.quit()
            quit()
```

```
            break

        pipe_ind = 0
        if len(birds) > 0:
            if len(pipes) > 1 and birds[0].x > pipes[0].x + pipes[0].PIPE_TOP.get_width():  # determine whether to use the first or second
                pipe_ind = 1                                 # pipe on the screen for neural network input

        for x, bird in enumerate(birds):  # give each bird a fitness of 0.1 for each frame it stays alive
            ge[x].fitness += 0.1
            bird.move()

            # send bird location, top pipe location and bottom pipe location and determine from network whether to jump or not
            output = nets[birds.index(bird)].activate((bird.y, abs(bird.y - pipes[pipe_ind].height), abs(bird.y - pipes[pipe_ind].bottom)))

            if output[0] > 0.5:  # we use a tanh activation function so result will be between -1 and 1. if over 0.5 jump
                bird.jump()

        base.move()

        rem = []
        add_pipe = False
        for pipe in pipes:
            pipe.move()
            # check for collision
            for bird in birds:
                if pipe.collide(bird, win):
```

```
            ge[birds.index(bird)].fitness -= 1
            nets.pop(birds.index(bird))
            ge.pop(birds.index(bird))
            birds.pop(birds.index(bird))

        if pipe.x + pipe.PIPE_TOP.get_width() < 0:
            rem.append(pipe)

        if not pipe.passed and pipe.x < bird.x:
            pipe.passed = True
            add_pipe = True

    if add_pipe:
        score += 1
        # can add this line to give more reward for passing through a pipe (not required)
        for genome in ge:
            genome.fitness += 5
        pipes.append(Pipe(WIN_WIDTH))

    for r in rem:
        pipes.remove(r)

    for bird in birds:
        if bird.y + bird.img.get_height() - 10 >= FLOOR or bird.y < -50:
            nets.pop(birds.index(bird))
            ge.pop(birds.index(bird))
            birds.pop(birds.index(bird))

    draw_window(WIN, birds, pipes, base, score, gen, pipe_ind)

    # break if score gets large enough
    '''if score > 20:
```

```
            pickle.dump(nets[0],open("best.pickle", "wb"))
            break'''


def run(config_file):
    """
    runs the NEAT algorithm to train a neural network to play flappy bird.
    :param config_file: location of config file
    :return: None
    """
    config = neat.config.Config(neat.DefaultGenome, neat.DefaultReproduction,
                         neat.DefaultSpeciesSet, neat.DefaultStagnation,
                         config_file)

    # Create the population, which is the top-level object for a NEAT run.
    p = neat.Population(config)

    # Add a stdout reporter to show progress in the terminal.
    p.add_reporter(neat.StdOutReporter(True))
    stats = neat.StatisticsReporter()
    p.add_reporter(stats)
    #p.add_reporter(neat.Checkpointer(5))

    # Run for up to 50 generations.
    winner = p.run(eval_genomes, 50)

    # show final stats
    print('\nBest genome:\n{!s}'.format(winner))


if __name__ == '__main__':
    # Determine path to configuration file. This path manipulation is
```

```
    # here so that the script will run successfully regardless of the
    # current working directory.
    local_dir = os.path.dirname(__file__)
    config_path = os.path.join(local_dir, 'config-feedforward.txt')
    run(config_path)
```

ДОДАТОК Б. ЗМІСТ ДОПОМІЖНОГО ФАЙЛУ ПРОЕКТУ

```
from __future__ import print_function


import copy

import warnings

import graphviz

import matplotlib.pyplot as plt

import numpy as np




def plot_stats(statistics, ylog=False, view=False, filename='avg_fitness.svg'):

    """ Plots the population's average and best fitness. """

    if plt is None:

        warnings.warn("This display is not available due to a missing optional dependency (matplotlib)")

        return


    generation = range(len(statistics.most_fit_genomes))

    best_fitness = [c.fitness for c in statistics.most_fit_genomes]

    avg_fitness = np.array(statistics.get_fitness_mean())

    stdev_fitness = np.array(statistics.get_fitness_stdev())
```

```
    plt.plot(generation, avg_fitness, 'b-', label="average")
    plt.plot(generation, avg_fitness - stdev_fitness, 'g-.', label="-1 sd")
    plt.plot(generation, avg_fitness + stdev_fitness, 'g-.', label="+1 sd")
    plt.plot(generation, best_fitness, 'r-', label="best")

    plt.title("Population's average and best fitness")
    plt.xlabel("Generations")
    plt.ylabel("Fitness")
    plt.grid()
    plt.legend(loc="best")
    if ylog:
        plt.gca().set_yscale('symlog')

    plt.savefig(filename)
    if view:
        plt.show()

    plt.close()


def plot_spikes(spikes, view=False, filename=None, title=None):
    """ Plots the trains for a single spiking neuron. """
    t_values = [t for t, I, v, u, f in spikes]
    v_values = [v for t, I, v, u, f in spikes]
```

```
u_values = [u for t, I, v, u, f in spikes]

I_values = [I for t, I, v, u, f in spikes]

f_values = [f for t, I, v, u, f in spikes]


fig = plt.figure()

plt.subplot(4, 1, 1)

plt.ylabel("Potential (mv)")

plt.xlabel("Time (in ms)")

plt.grid()

plt.plot(t_values, v_values, "g-")


if title is None:

   plt.title("Izhikevich's spiking neuron model")

else:

   plt.title("Izhikevich's spiking neuron model ({0!s})".format(title))


plt.subplot(4, 1, 2)

plt.ylabel("Fired")

plt.xlabel("Time (in ms)")

plt.grid()

plt.plot(t_values, f_values, "r-")


plt.subplot(4, 1, 3)

plt.ylabel("Recovery (u)")

plt.xlabel("Time (in ms)")
```

```
    plt.grid()
    plt.plot(t_values, u_values, "r-")

    plt.subplot(4, 1, 4)
    plt.ylabel("Current (I)")
    plt.xlabel("Time (in ms)")
    plt.grid()
    plt.plot(t_values, I_values, "r-o")

    if filename is not None:
        plt.savefig(filename)

    if view:
        plt.show()
        plt.close()
        fig = None

    return fig


def plot_species(statistics, view=False, filename='speciation.svg'):
    """ Visualizes speciation throughout evolution. """
    if plt is None:
        warnings.warn("This display is not available due to a missing optional dependency
(matplotlib)")
```

```
    return

    species_sizes = statistics.get_species_sizes()
    num_generations = len(species_sizes)
    curves = np.array(species_sizes).T

    fig, ax = plt.subplots()
    ax.stackplot(range(num_generations), *curves)

    plt.title("Speciation")
    plt.ylabel("Size per Species")
    plt.xlabel("Generations")

    plt.savefig(filename)

    if view:
        plt.show()

    plt.close()


def draw_net(config, genome, view=False, filename=None, node_names=None,
show_disabled=True, prune_unused=False,
             node_colors=None, fmt='svg'):
    """ Receives a genome and draws a neural network with arbitrary topology. """
```

```
    # Attributes for network nodes.

    if graphviz is None:

        warnings.warn("This display is not available due to a missing optional dependency
(graphviz)")

        return


    if node_names is None:

        node_names = {}


    assert type(node_names) is dict


    if node_colors is None:

        node_colors = {}


    assert type(node_colors) is dict


    node_attrs = {

        'shape': 'circle',

        'fontsize': '9',

        'height': '0.2',

        'width': '0.2'}


    dot = graphviz.Digraph(format=fmt, node_attr=node_attrs)


    inputs = set()
```

```
for k in config.genome_config.input_keys:
    inputs.add(k)
    name = node_names.get(k, str(k))
    input_attrs = {'style': 'filled', 'shape': 'box', 'fillcolor': node_colors.get(k, 'lightgray')}
    dot.node(name, _attributes=input_attrs)

outputs = set()
for k in config.genome_config.output_keys:
    outputs.add(k)
    name = node_names.get(k, str(k))
    node_attrs = {'style': 'filled', 'fillcolor': node_colors.get(k, 'lightblue')}

    dot.node(name, _attributes=node_attrs)

if prune_unused:
    connections = set()
    for cg in genome.connections.values():
        if cg.enabled or show_disabled:
            connections.add((cg.in_node_id, cg.out_node_id))

    used_nodes = copy.copy(outputs)
    pending = copy.copy(outputs)
    while pending:
        new_pending = set()
        for a, b in connections:
```

```
            if b in pending and a not in used_nodes:
                new_pending.add(a)
                used_nodes.add(a)
        pending = new_pending
else:
    used_nodes = set(genome.nodes.keys())

for n in used_nodes:
    if n in inputs or n in outputs:
        continue

    attrs = {'style': 'filled',
             'fillcolor': node_colors.get(n, 'white')}
    dot.node(str(n), _attributes=attrs)

for cg in genome.connections.values():
    if cg.enabled or show_disabled:
        #if cg.input not in used_nodes or cg.output not in used_nodes:
        #    continue
        input, output = cg.key
        a = node_names.get(input, str(input))
        b = node_names.get(output, str(output))
        style = 'solid' if cg.enabled else 'dotted'
        color = 'green' if cg.weight > 0 else 'red'
        width = str(0.1 + abs(cg.weight / 5.0))
```

```
        dot.edge(a, b, _attributes={'style': style, 'color': color, 'penwidth': width})


dot.render(filename, view=view)


return dot
```