МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Аліна САВЧЕНКО

"____"_____2021 р.

# ДИПЛОМНА РОБОТА

**(ПОЯСНЮВАЛЬНА ЗАПИСКА)**

*ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ*

**"МАГІСТРА"**

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

**Тема: "Штучна нейронна система з використанням генетичних алгоритмів для її тренування"**

**Виконавець:** Саранча Роман Михайлович

**Керівник:** професор Зіатдінов Юрій Кашафович

**Нормоконтролер:** _____ Ігор РАЙЧЕВ

**Київ - 2021**

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра Комп'ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології".

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Аліна САВЧЕНКО
«____»_____2021р.

# ЗАВДАННЯ

**на виконання дипломної роботи студента**

Саранчі Романа Михайловича
(прізвище, ім'я, по батькові)

**1. Тема роботи:** «Штучна нейронна система з використання з використанням генетичних алгоритмів для її тренування» затверджена наказом ректора від 12.10.2021 за № 2228/ст.

**2. Термін виконання роботи:** з 12.10.2021 по 31.12.2021.

**3. Вихідні дані до роботи:** розробка алгоритмів для тренування нейронної системи

**4. Зміст пояснювальної записки**: вступ, аналітичний огляд і постановка завдання, реалізація завдання, аналіз виконаного завдання та його характеристики.

**5. Перелік обов'язкового ілюстративного матеріалу:** архітектура нейронної мережі, блок-схема алгоритму створення архітектури ЗНМ, інтерфейс створеного додатку.

**6. Календарний план-графік**

| № п/п | Етапи виконання дипломної роботи | Термін виконання етапів | Примітка |
|---|---|---|---|
| 1. | Аналіз літератури за темою дипломного проекту | 13.10.21 – 15.10.21р. | |
| 2. | Розробка та затвердження плану дипломного проекту | 16.10.21 – 19.10.21р. | |
| 3. | Проведення консультації з науковим керівником щодо створення першого розділу | 20.10.21 – 23.10.21р. | |
| 4. | Розробка навчання нейронної мережі | 24.10.21 – 31.10.21р. | |
| 5. | Підбирання алгоритму тренування | 01.11.21 – 15.11.21р. | |
| 6. | Опис інтерфейсу створеної програми | 16.11.21 – 29.11.21р. | |
| 7. | Висновки та оформлення пояснювальної записки дипломного проекту | 30.11.21 – 10.12.21р. | |

**7. Дата видачі завдання:** 12.10.2021р.

**Керівник дипломної роботи** _____ Юрій ЗІАТДІНОВ
(підпис керівника)

**Завдання прийняв до виконання** _____ Роман САРАНЧА

# РЕФЕРАТ

Пояснювальна записка до дипломної роботи "Використання генетичних алгоритмів для тренування штучних нейронних мереж": 70 сторінок, 19 рисунків, 11 бібліографічних посилань та інформаційних джерел.

***Мета роботи:*** покращення способів розробки та тренування нейронних мереж для подальшого їх використання у якості штучного інтелекту.

***Предмет дослідження:*** автоматизована система навчання нейронних мереж на основі генетичних алгоритмів.

***Об'єкт дослідження:*** процес навчання нейронної мережі.

***Ключові слова:*** ГЕНЕТИЧНІ АЛГОРИТМИ, НЕЙРОННІ МЕРЕЖІ, НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ, ШТУЧНИЙ ІНТЕЛЕКТ.

**ЗМІСТ**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

БД – база даних

ШНМ – штучна нейронна мережа

ЗНМ – згорткова нейронна мережа

СУБД – система управління базами даних

ЕОМ – електронна обчислювальна машина

АРМ – автоматизоване робоче місце

ПК – персональний комп’ютер

UI – інтерфейс користувача

# ВСТУП

Одними з найважливіших областей досліджень і розробок сучасної кібернетики є області машинного навчання та штучного інтелекту, які широко використовуються для вирішення досить великого спектру задач: від прогнозування і класифікації образів до більш складних таких як системи прийняття рішень.

В дипломному проекті досліджується використання звичайних багатошарових нейронних мереж в контексті їх навчання за допомогою генетичних алгоритмів для моделювання поведінки мікроорганізмів в замкнутому середовищі. Такі задачі досить важко вирішувати за допомогою звичайних “не гнучких” алгоритмів, так як вони можуть потребувати коригування параметрів, а бо навіть повної заміни алгоритму в залежності від параметрів моделюмого середовища і тому дана робота ставить за ціль розробити евристичний нейроеволюційний алгоритм для вирішення даного класу проблем моделювання.

# РОЗДІЛ 1
# НЕЙРОННІ МЕРЕЖІ

## 1.1. Штучна нейронна мережа

Штучні нейронні мережі (ШНМ) - математичні моделі, а також їх програмні або апаратні реалізації, побудовані за принципом організації та функціонування біологічних нейронних мереж - мереж нервових клітин живого організму. Це поняття виникло щодо процесів, які у мозку, і за спробі змоделювати ці процеси. Першою такою спробою були нейронні мережі Маккалока та Піттса. Згодом, після розробки алгоритмів навчання, одержувані моделі стали використовувати у практичних цілях: у задачух прогнозування, для розпізнавання образів, у задачух управління та інших.

ШНМ є системою з'єднаних і взаємодіючих між собою простих процесорів (штучних нейронів). Такі процесори зазвичай досить прості, особливо у порівнянні з процесорами, які використовуються в персональних комп'ютерах. Кожен процесор подібної мережі має справу лише з сигналами, які він періодично отримує, та сигналами, які він періодично надсилає іншим процесорам. Проте, будучи з'єднаними в досить велику мережу з керованою взаємодією, такі локально прості процесори разом здатні виконувати досить складні задачу.

Штучні нейронні мережі індуковані біологією, оскільки складаються з елементів, функціональні можливості яких аналогічні більшості елементарних функцій біологічного нейрона. Ці елементи потім організуються за способом, який може відповідати (або відповідати) анатомії мозку. Всупереч на таку поверхневу подібність, штучні нейронні мережі демонструють дивовижне число властивостей мозку.

Наприклад, вони навчаються на основі досвіду, узагальнюють попередні прецеденти на нові випадки і витягують суттєві властивості з інформації, що надходить, що містить зайві дані.

Всупереч на таку функціональну схожість, навіть найоптимістичніший їх захисник не припустить, що незабаром штучні нейронні мережі дублюватимуть функції людського мозку. Реальний «інтелект», що демонструється найскладнішими нейронними мережами, знаходиться нижче рівня дощового хробака, і ентузіазм має бути помірним відповідно до сучасних реалій. Однак і було б неправильним ігнорувати дивовижну подібність у функціонуванні деяких нейронних мереж з людським мозком. Ці можливості, хоч би як вони були обмежені сьогодні, наводять на думку, що глибоке проникнення в людський інтелект, а також безліч революційних додатків, можуть бути не за горами.

***Навчання***

Штучні нейронні мережі можуть змінювати свою поведінку залежно від зовнішнього середовища. Цей фактор більшою мірою, ніж будь-який інший, відповідальний за той інтерес, який вони викликають. Після пред'явлення вхідних сигналів (можливо разом з необхідними виходами) вони самоналаштовуються, щоб забезпечувати необхідну реакцію. Було розроблено безліч навчальних алгоритмів, кожен зі своїми сильними та слабкими сторонами.

***Узагальнення***

Відгук мережі після навчання може бути певною мірою нечутливий до невеликих змін вхідних сигналів. Ця внутрішньо властива здатність бачити образ крізь шум та спотворення життєво важлива для розпізнавання образів у реальному світі. Вона дозволяє подолати вимогу суворої точності, що пред'являється звичайним комп'ютером, і відкриває шлях до системи, яка може мати справу з тим недосконалим світом, в якому ми живемо. Важливо, що штучна нейронна мережа робить узагальнення автоматично завдяки структурі, а чи не з допомогою використання «людського інтелекту» у вигляді спеціально написаних комп'ютерних програм.

***Абстрагування***

Деякі зі штучних нейронних мереж мають здатність отримувати сутність із вхідних сигналів. Наприклад, мережа може бути навчена на послідовність викривлених версій літери «А». Після відповідного навчання подання такого спотвореного прикладу призведе до того, що мережа породить букву досконалої форми. У певному сенсі вона навчиться породжувати те, що ніколи не бачила.

Ця здатність видобувати ідеальне з недосконалих входів ставить цікаві філософські питання. Вона нагадує концепцію ідеалів, висунуту Платоном у його «Республике». У всякому разі, здатність видобувати ідеальні прототипи є у людей дуже цінною якістю.

***Застосовність***

Штучні нейронні мережі є панацеєю. Вони, мабуть, не годяться для виконання таких завдань, як нарахування заробітної плати. Схоже, однак, що їм надаватиметься перевага у великому класі завдань розпізнавання образів, з якими погано або взагалі не справляються звичайні комп'ютери.

Коло завдань, на вирішення яких використовуються нейронні мережі, багато в чому збігається із задачуми, розв'язуваними традиційними статистичними методами. Тому вкажемо переваги нейромереж перед кількома класичними методами статистики.

У порівнянні з лінійними методами статистики (лінійна регресія, авторегресія, лінійний дискримінант), нейронні мережі дозволяють ефективно будувати нелінійні залежності, які більш точно описують набори даних. З нелінійних методів класичної статистики поширений, мабуть, лише байесовський класифікатор, що будує квадратичну поверхню, що розділяє - нейронна мережа ж може побудувати поверхню вищого порядку. Висока нелінійність розділяючої поверхні наївного байєсівського класифікатора (він не використовує кваріаційні матриці класів, як класичний байєс, а аналізує локальні щільності ймовірності) вимагає значного сумарного числа прикладів для можливості оцінювання ймовірностей при кожному поєднанні інтервалів значень змінних − нейронна

мережа не фрагментуючи її, що підвищує адекватність налаштування нейронної мережі.

При побудові нелінійних моделей (наприклад, поліноміальних) у статистичних програмах зазвичай потрібне ручне введення-опис моделі у символьному вигляді з точністю до значень параметрів: при N=10 незалежних змінних поліном другого ступеня міститиме N*(N-1)/2=45 коефіцієнтів при попарних творах змінних, 10 за самих змінних, 10 за квадратів значень змінних, тобто. 65 (66 з урахуванням неоднорідного доданку) коефіцієнтів. При двадцяти змінних у вираз увійде 231 доданок. Вводити такі довгі формули довго, великий ризик помилки. Нейронна мережа створюється шляхом вказівки виду структури, числа шарів і числа нейронів у кожному шарі, що набагато швидше. А алгоритми побудови нейромереж, що ростуть, і зовсім не вимагають початкового задачу розміру нейронної мережі. Альтернативою нейронної мережі при побудові складних нелінійних моделей є лише метод групового обліку аргументів.

Для стиснення та візуалізації даних у статистиці розроблено метод лінійних основних компонентів. Нейросети-автоасоціатори дозволяють ефективніше стискати дані за рахунок побудови нелінійних відображень та візуалізувати дані у просторі меншої кількості нелінійних головних компонентів.

У порівнянні з методами непараметричної статистики, нейронна мережа з радіальними базовими властивістьми дозволяє скорочувати кількість ядер, оптимізувати координати та розмитість кожного ядра. Це дозволяє за збереження парадигми локальної ядерної апроксимації прискорювати подальший процес прийняття рішення.

Під час навчання нейронної мережі замість критерію якості як найменших квадратів можна використовувати робастные критерії, додатково вести оптимізацію та інших властивостей нейронної мережі (наприклад, додаючи критерії регуляризації рішення чи оптимізації структури нейронної мережі). Алгоритми навчання нейронної мережі у своїй залишаються незмінними.

Необхідність вирішення прямої та зворотної задач зазвичай вимагає побудови двох моделей. При використанні нейронних мереж можна обійтися однією мережею, навченою вирішувати пряме задачу.

### 1.2. Архітектура ШНМ

У більшості архітектур ШНМ функції активації нейронів є статичними, а вагові коефіцієнти штучних синапсів є змінними параметрами мережі, що коригуються при тренуванні. Буває так, що певні входи конкретних нейронів є зовнішніми входами всієї мережі, а деякі виходи нейронів, відповідно, виходами мережі.

У повнозв'язних нейронних мережах кожен нейрон передає свій вихідний сигнал іншим нейронам, у тому числі й самому собі. Усі вхідні сигнали подаються всім нейронам. Вихідними сигналами мережі можуть бути всі або деякі вихідні сигнали нейронів після кількох тактів функціонування мережі. У багатошарових (шарових) нейронних мережах нейрони поєднуються в шари. Шар містить сукупність нейронів із єдиними вхідними сигналами.

Число нейронів у шарі може бути будь-яким і не залежить від кількості нейронів в інших шарах. У випадку мережа складається з шарів, пронумерованих зліва направо. Зовнішні вхідні сигнали подаються на входи нейронів вхідного шару (його часто нумерують як нульовий), а виходами мережі є вихідні сигнали останнього шару. Крім вхідного і вихідного шарів багатошарової нейронної мережі є один або кілька прихованих шарів. Зв'язки від виходів нейронів деякого шару q до входів нейронів наступного шару (q+1) називаються послідовними (рис. 1.1.)

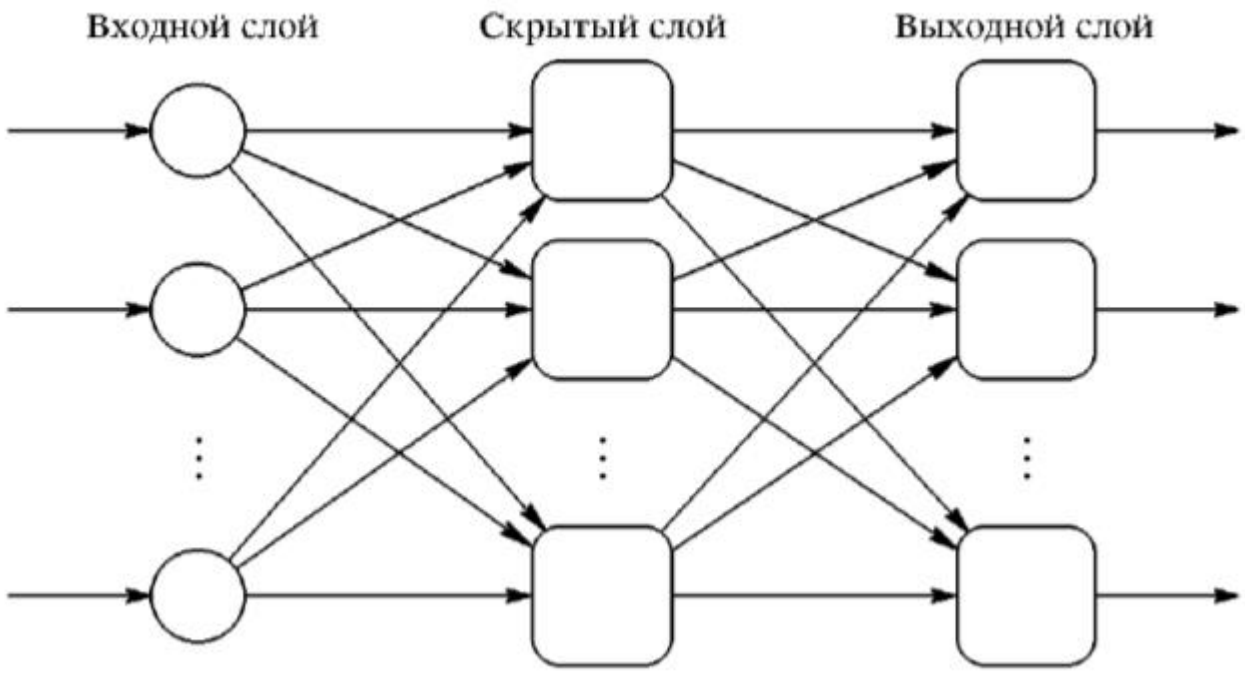


Рис. 1.1. Типова архітектура нейронної мережі

У найбільш поширеному сімействі мереж першого класу, званих багатошаровим перцептроном, нейрони розташовані шарами і мають односпрямовані зв'язки між шарами. Мережі прямого поширення є статичними тому, що у заданий вхід вони виробляють одну сукупність вихідних значень, які залежать від попереднього стану мережі. Рекурентні мережі є динамічними, тому що через зворотні зв'язки в них модифікуються входи нейронів, що призводить до зміни стану мережі.

### 1.3. Види багатошарових штучних нейронних мереж

Монотонні. Це окремий випадок шаруватих мереж з додатковими умовами на зв'язки та нейрони. Кожен шар, крім останнього (вихідного), розбитий на два блоки: збуджуючий і гальмуючий. Зв'язки між блоками теж поділяються на гальмівні та збуджуючі. Якщо від нейронів блоку до нейронів блоку ведуть лише збуджуючі зв'язки, це означає, що будь-який вихідний сигнал блоку є монотонною незнищувальною функцією будь-якого вихідного сигналу блоку.

Якщо ж ці зв'язки лише гальмують, будь-який вихідний сигнал блоку є функцією, що не зростає, будь-якого вихідного сигналу блоку. Для нейронів монотонних мереж потрібна монотонна залежність вихідного сигналу нейрона від параметрів вхідних сигналів.

Мережі без зворотного зв'язку. У таких мережах нейрони вхідного шару отримують вхідні сигнали, перетворюють їх і передають нейронам першого прихованого шару, і так аж до вихідного, що видає сигнали інтерпретатора і користувача. Якщо не обумовлено неприємне, то кожен вихідний сигнал q-го шару подається на вхід всіх нейронів (q+1)-го шару; однак можливий варіант з'єднання q-го шару з довільним шаром.

Серед багатошарових мереж без зворотних зв'язків розрізняють повнозв'язні (вихід кожного нейрона q-го шару пов'язаний з входом кожного нейрона (q+1)-го шару) та частково повнозв'язні.

Прикладом статичних архітектур ШНМ може бути:

- Перцептрон
- Нейронна мережа Кохонена
- Когнітрон
- Згорткова нейронна мережа
- Звичайний неокогнітрон

Як протилежність статичним, відповідно мають місце динамічні версії ШНМ, що мають рекурентну топологію, що включають в себе механізм зворотних зв'язків, завдяки чому стан структури в кожний наступний момент часу T залежить від минулого стану. Рекурентні ШНМ зазвичай є одним із видів перцептрона. Динамічні рекурентні ШНМ з зворотними зв'язками:

- мережа Хопфілда
- мережа Коско
- мережа Джордана
- мережа Елмана

Структура перцептрона будується на основі сенсорних даних, які поступають на вхід мережі. Головними сутностями є: S-елементи, асоціативні елементи - A-

елементи, і елементи, що реагують на виході - R-елементи. S-елементи пов'язані з A-елементом оргнанізовують “асоціацію”, і A-елемент вмикається відразу після того, як лише певне число сигналів від S-елементів поступає на вхід. A-елемент направляє сигнал з урахуванням вагових коефіцієнтів на R-елемент що робить зважену суму сигналів, і дивлячись на те, чи перевищує зважена сума певний поріг, R-елемент створює рішення роботи перцептрону (Рис.1.2.). Перцептрони, що мають багато шарів будуються з додаванням прихованих шарів із A-елементів, розташованими між S-елементами і R-елементами.

Ступінь складності проблем, що можна вирішити за допомогою багатошарового перцептрона, є найвищою для класу перцептронів. Навчання елементарного і багатошарового перцептрона заключається в ітеративному коригуванні вагових коефіцієнтів зв'язків між шарами A і R. Перцептрон здатний працювати в режимі розпізнавання або узагальнення.

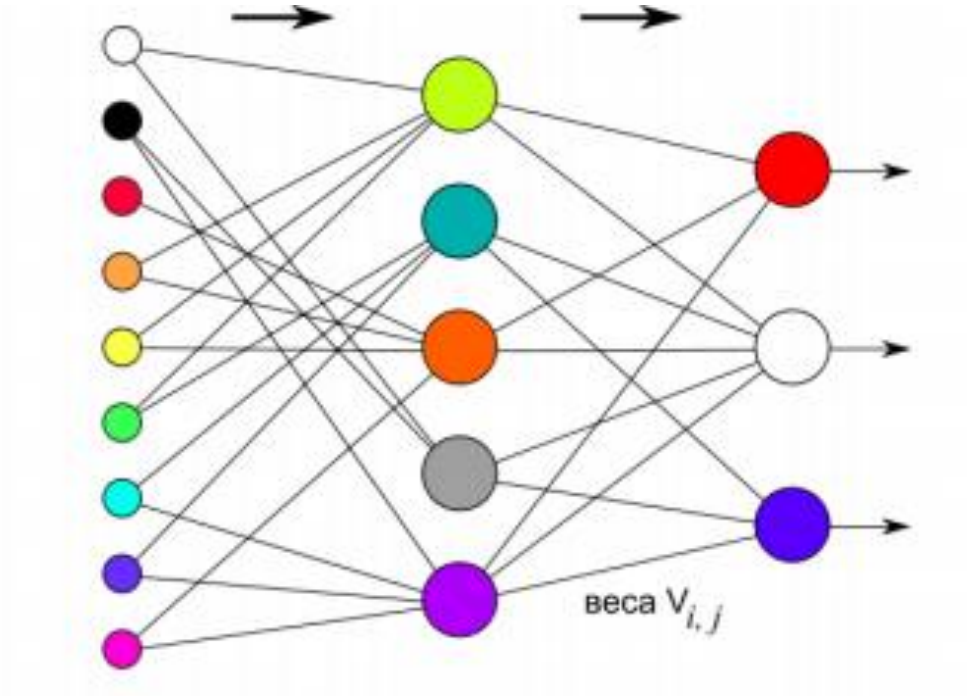


Рис.1.2. Перцептрон

В простому перцептроні нейрони, що є вхідними мають зв’язки напряму з вихідними нейронами, S <--> A зв’язки існують за принципом прямої

відповідності. Перцептрон у якого всього 1 шар є просто окремим випадком елементарного перцептрона, але має велику кількість обмежень, наприклад, неможливість вирішувати певні класи задач такі як властивість “Виключного АБО”.

### 1.4. Топологі згорткових нейронних мереж

Однією з найважливіших проблем побудови систем розпізнавання образів на основі нейронних мереж є те, що виникає необхідність підбору значень ряду параметрів, що мають віддалене відношення до даної предметної області - параметрів, що стосуються самої нейронної мережі: кількість шарів, кількість нейронів у кожному конкретному шарі, вид функції активації нейрону.

Це призводить до того, що в процесі побудови системи доводиться шукати не лише найбільш оптимальну модель даної предметної області, але і найбільш оптимальне поєднання параметрів нейронної мережі - будувати та навчати безліч мереж, що відповідають певній моделі та певним параметрам. Процес навчання нейронної мережі досить тривалий, а ефективного методу, що дозволяє визначити оптимальні параметри мережі без її попереднього навчання - ні, отже, пошук оптимальних параметрів ведеться «на сліпу», займає тривалий час, призводить до дорожнечі систем на нейронних мережах.

Проведене моделювання різних класичних нейромережевих структур (багатошарові персептрони, радіально-базисні нейронні мережі, карти Кохонена) показало, що застосування даних нейромережевих архітектур до цього задачу є неефективним, а саме:

- дані нейронні мережі не забезпечують необхідний рівень надійності визначення особи. Під час проведення моделювання рівень правильно класифікованих зображень не перевищував 70%;
- є дуже громіздкими та повільними для застосування до задачу виділення сюжетної частини для систем відеоконтролю та відеоспостереження;

- не мають інваріантності до змін різних зовнішніх умов (умов зйомки, спотворень, якості зображення).

Згорткові нейронні мережі (ЗНМ) схожі на мережі прямого поширення типу персептрон: нейрони даного типу також мають ваги, що навчаються, і параметр зсуву. Однак ЗНМ беруть до уваги те, що вхідні дані являють собою зображення. Маючи дану інформацію, можна закодувати певні властивості в архітектуру мережі, що дозволяє значно зменшити кількість параметрів нейронної мережі.

Виходячи з цього та успішного застосування згорткових нейронних мереж до вирішення даної задачі в роботі [22], було прийнято рішення, що найбільш оптимальним буде конфігурація та подальше вдосконалення запропонованої в даній роботі нейронної мережі.

Згортковим нейронним мережам властиво мати багатовимірні шари (в основному використовуються двомірні, наприклад, в мережах, які опрацюють зображення, і тривимірні, наприклад, додавання декількох колірних каналів для зображення) декількох типів:

- « Вхідний шар: вхідне зображення, включаючи кілька колірних каналів.
- Згортковий шар (*Convolution*): всі нейрони шару, на відміну від перцептрона, пов'язані лише з частиною нейронів попереднього шару.
- Шар субдискретизації (*Pooling, Subsampling*): виділення найбільш значущих ознак попереднього шару і значне скорочення розмірності наступних шарів мережі.
- Повнозв'язний шар (*Fully-connected*): являє собою прихований шар штучної нейронної мережі типу перцептрон» [23. с.45].

Також, після згорткового шару і повнозв'язного шару може застосовуватися властивість активації нейрона, яка перетворює сигнал нейрона і формує вихідний сигнал.

Нижче всі перераховані типи шарів описуються більш детально. Вхідний шар. Найчастіше у задачах розпізнавання образів та патернів у двомірному векторі вхідний шар подається у вигляді 3-вимірної сітки розміри якої залежать від розміру вхідного вектору - формула (1.1).

$$In = W * H * D \quad (1.1)$$

де $In$ - розмірність вхідного шару мережі;

$W$ - ширина вхідного вектору; $H$ - висота вхідного вектору;

$D$ - глибина або кількість колірних каналів зображення.

Згортковий шар. Важливим компонентом штучної нейронної мережі є згортковий шар. Він необхідний для класифікації певних ознак зображення та їх трансформації, і які, після чого, на наступних шарах, використовуються щоб отримати складніші ознаки для визначення класу розпізнаного об'єкта.

Важливою ознакою такого згорткового шару є фільтри - що являють собою 2-мірні чи 3-вимірні матриці ваг зв'язків нейронів попереднього шару з нейронами згорткового шару. Операція згортування – процес отримання вихідного сигналу нейрона згорткового шару, який має відповідні подібності з операцією фільтрації зображення: кожне значення сигналу нейрона попереднього шару, розташованого в певній галузі, відповідної ядру фільтра, множиться на відповідне значення ядра фільтра (в ЗНМ значення ядра фільтра називаються вагами зв'язків нейронів згорткового шару: $w_{00}$ , $w_{01}$, ... $w_{(F-1)}$, $w_{(F)}$, де F - розмір квадратного фільтра); а результатом фільтрації є сума всіх отриманих творів [5]. Саме тому вони носять назву фільтра. На рис. 1.3 показано процес створення сигналів нейронів згорткового шару.

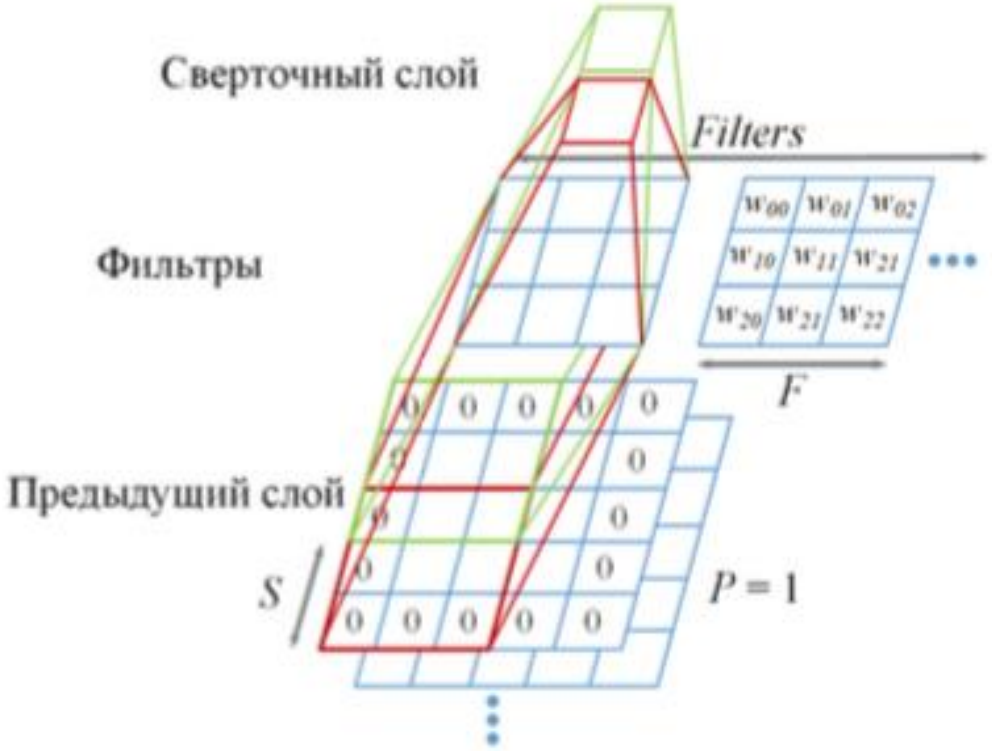


Рис. 1.3. Створення згортального шару згортувальної нейронної мережі

Не менш важливими параметрами згортального шару мережі є тип і кількість використовуваних фільтрів. Тип фільтра вказує для виділення яких ознак зображення він використовується. Наприклад, це може бути виділення прямих ліній під певним кутом (в процесі навчання мережі ваги фільтрів змінюються, впливаючи на характер виділяються ними ознак). Як правило, використовуються квадратні фільтри, розміром F. На рис. 1.3 filters показано як кількість використовуваних фільтрів визначає глибину сверточного шару.

Також згортковий шар має і інші параметри:

S - зміщення (stride) - показує на скільки позицій зміщується фільтр для створення сигналу наступного нейрона згорткового шару.

P - додавання нулів (padding) - додавання нейронів з нульовими значеннями, які не мають негативний вплив на створення сигналу згорткового шару, по краях попереднього шару для регулювання розмірності згорткового шару мережі.

Розмірність згорткового шару формується його параметрами і розмірами попереднього шару мережі, відповідно до формул (2) і (3).

$$W = \frac{W_p - F + P*2}{S} + 1, \quad (1.2)$$

$$H = \frac{H_p - F + P*2}{S} + 1 \quad (1.3)$$

де $W$ і $H$ - ширина і висота згорткового шару відповідно;

$W_p$ і $H_p$ - ширина і висота попереднього шару відповідно;

$F$ - розмір квадратного фільтру сверточного шару мережі;

$P$ - кількість доданих нулів по краях попереднього шару;

$S$ - зміщення фільтра.

Для створення результуючого сигналу нейронів згорткового шару в ЗНМ використовуються функції активації нейрона. Дані функції за певними правилами перетворять сигнали нейронів. У ЗНМ найчастіше використовуються три види функцій активації: порогова - формула (1.4), сігмоідні - формула (1.5) і тангенс - формула (1.6).

$$y = max(0, x), \quad (1.4)$$

$$y = \frac{1}{1+e^{-x}}, \quad (1.5)$$

$$y = tg(x), \quad (1.6)$$

Шар субдискретизації. Даний шар згорткової нейронної мережі використовується для скорочення розмірності даних з метою зменшення ймовірності швидкого перенавчання, а також для скорочення обчислювальних витрат і витрат пам'яті. Зазвичай даний шар використовується після проведення операції згортки і перетворює сигнали сверточного шару, виділяючи найбільш значущі, за певними критеріями.

В даному шарі використовується вікно певного розміру (U) для відображення областей нейронів минулого шару, які будуть пов'язані з нейроном шару Субдискретизація. Наступним кроком, за відповідними правилами вибирається і обчислюється один сигнал нейрона шару Субдискретизація. Поширеними та

типовими способами створення сигналу даного шару - це або шляхом знаходження максимального значення серед сигналів попереднього шару (max), що знаходяться в межах вікна Субдискретизація; або шляхом розрахунку середнього значення (avg). Процес створення сигналів шару Субдискретизація зображена на рис. 1.4.

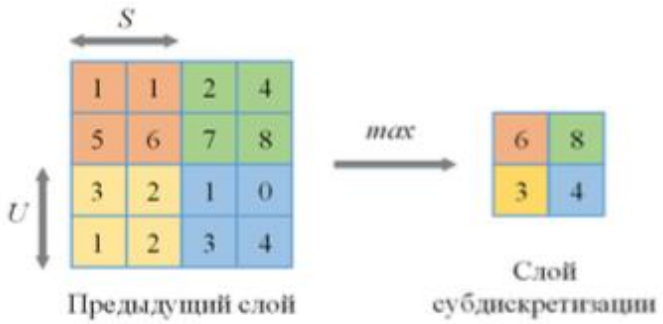


Рис.1.4. Створення шару Субдискретизація за допомогою операції max

Розмірність шару Субдискретизація обраховується його параметрами і розмірами попереднього шару мережі, відповідно до формул (1.7) і (1.8).

$$W = \frac{(W_p - U)}{S} + 1, \qquad (1.7)$$

$$H = \frac{(H_p - U)}{S} + 1, \qquad (1.8)$$

де $W$ і $H$ - ширина і висота шару Субдискретизація відповідно;

$W_p$ і $H_p$ - ширина і висота попереднього шару відповідно;

$U$ - розмір квадратного вікна шару Субдискретизація мережі;

$S$ - зміщення вікна шару Субдискретизація.

Одним із основних параметрів шару Субдискретизація є розмір вікна U. Вважається, що чим він більший, тим сильніше руйнівну дію операції Субдискретизація помічені на згортковому шарі ознаки, чим він менший, тим сильніше перенавчання мережі.

Повнозв'язний шар. Даний шар є одновимірним і в ньому кожен нейрон пов'язаний з кожним нейроном попереднього шару на всіх рівнях, якщо у попереднього шару присутній параметр глибини.

Основне призначення даного шару - перетворення сигналів, отриманих на згорткових рівнях мережі до одновимірного увазі і виділення ознак на одновимірному рівні. Даний шар також може використовуватися в якості останнього (вихідного) шару ЗНМ, результатом якого є ймовірність приналежності вхідного зображення певного класу. Схема даного шару представлена на рис. 1.5.

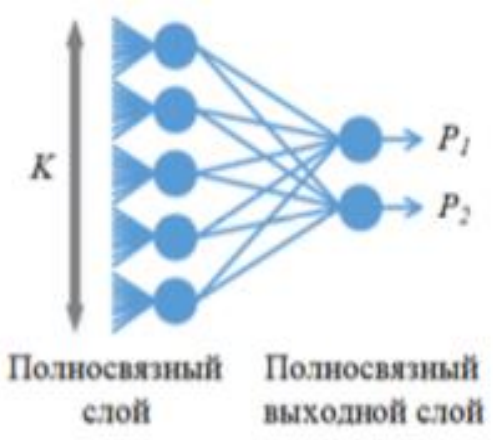


Рис. 1.5. Два повнозв'язних шара згорткової нейронної мережі, один з яких використовується в якості вихідного шару

Головним та єдиним параметром шару є кількість нейронів K. Зазвичай він вибирається відповідно до розміру попереднього шару і кількості класів (необхідної кількості виходів) мережі.

Насправді, виконується операція згортки: значення пікселів зображення поелементно множаться на ядро згортки, тобто. значення ваг нейрона, а потім отримані твори сумуються. Результат є значення даного нейрона.

Параметри згорткового шару:

K – кількість фільтрів;

F – розмір фільтра;

S – крок;

P – кількість заповнень нулями.

Розмір вихідних даних W2×H2×D2 обчислюється так:

W2 = (W1 - F + 2P) / S + 1,

H2 = (H1 - F + 2P) / S + 1,

D2 = K,

де W1 – ширина, H1 – висота, D1 – глибина вхідних даних.

### 1.5. Навчання згорткової нейронної мережі

Здатність до навчання є основною властивістю мозку. У контексті ШНМ процес навчання може розглядатися як налаштування архітектури та ваги зв'язків для ефективного виконання спеціального задачу. Зазвичай нейронна мережа має налаштувати вагу зв'язків з наявною навчальною вибіркою. Функціонування мережі покращується в міру ітеративного налаштування вагових коефіцієнтів. Властивість мережі учнів з прикладів робить їх привабливішими проти системами, які слідують певної системі правил функціонування, сформульованої експертами.

Для конструювання процесу навчання насамперед необхідно мати модель зовнішнього середовища, в якому функціонує нейронна мережа – знати доступну для мережі інформацію. Ця модель визначає парадигму навчання. По-друге, необхідно поняття, як модифікувати вагові параметри мережі – які правила навчання керують процесом налаштування. Алгоритм навчання означає процедуру, в якій використовуються правила навчання для ваги.

Є три парадигми навчання: "з учителем", "без учителя" (самонавчання) та змішана. У першому випадку нейронна мережа має правильні відповіді (виходи мережі) на кожен вхідний приклад. Терези налаштовуються так, щоб мережа виробляла відповіді якомога ближчі до відомих правильних відповідей. Посилений варіант навчання з учителем припускає, що відома лише критична

оцінка правильності виходу нейронної мережі, але не самі правильні значення виходу. Навчання без учителя не вимагає знання правильних відповідей на кожен приклад навчальної вибірки. І тут розкривається внутрішня структура даних чи кореляції між зразками у системі даних, що дозволяє розподілити зразки за категоріями. При змішаному навчанні частина тяжкостей визначається за допомогою навчання з учителем, тоді як решта виходить за допомогою самонавчання.

Теорія навчання розглядає три фундаментальні властивості, пов'язані з навчанням за прикладами: ємність, складність зразків та обчислювальна складність. Під ємністю розуміється, скільки зразків може запам'ятати мережу, і які функції та межі прийняття рішень можуть бути на ній сформовані. Складність зразків визначає кількість навчальних прикладів, необхідних досягнення здібності до узагальнення. Занадто мала кількість прикладів може викликати "перенавченість" мережі, коли вона добре функціонує на прикладах навчальної вибірки, але погано - на тестових прикладах, підпорядкованих тому ж статистичного розподілу. Відомі 4 основних типи правил навчання: корекція помилково, машина Больцмана, правило Хебба та навчання методом змагання.

Правило корекції помилково. При навчанні з учителем кожного вхідного прикладу заданий бажаний вихід d. Реальний вихід мережі може не співпадати з бажаним. Принцип корекції помилково під час навчання заключається у використанні сигналу (d-y) для модифікації ваг, що забезпечує поступове зменшення помилки. Навчання має місце лише у випадку, коли перцептрон помиляється. Відомі різні модифікації цього алгоритму навчання.

Навчання Больцмана. Є стохастичним правилом навчання, яке випливає з інформаційних теоретичних і термодинамічних принципів. Метою навчання Больцмана є така настройка вагових коефіцієнтів, коли стану видимих нейронів задовольняють бажаному розподілу ймовірностей. Навчання Больцмана може розглядатися як спеціальний випадок корекції помилково, в якому під помилкою розуміється розбіжність Кореляцій станів у двох режимах.

Правило Хебба. Найстарішим навчальним правилом є постулат навчання Хебба. Хебб спирався на такі нейрофізіологічні спостереження: якщо нейрони з обох боків синапсу активізуються одночасно і регулярно, сила синаптичного зв'язку зростає. Важливою особливістю цього правила є те, що зміна синаптичної ваги залежить лише від активності нейронів, пов'язаних із цим синапсом. Це значно спрощує ланцюга навчання у реалізації VLSI.

Навчання методом змагання. На відміну від навчання Хебба, в якому безліч вихідних нейронів можуть збуджуватися одночасно, при змаганні навчання вихідні нейрони змагаються між собою за активізацію. Це явище відоме як правило "переможець бере все". Подібне навчання має місце у біологічних нейронних мережах. Навчання за допомогою змагання дозволяє кластеризувати вхідні дані: подібні приклади групуються мережею відповідно до кореляцій і є одним елементом.

При навчанні модифікуються лише ваги нейрона, що "переміг". Ефект цього правила досягається за рахунок такої зміни збереженого в мережі зразка (вектора ваг зв'язків нейрона, що переміг), при якому він стає трохи ближче до вхідного прикладу. На рис. 3 дано геометричну ілюстрацію навчання методом змагання. Вхідні вектори нормалізовані та представлені точками на поверхні сфери. Вектори ваги для трьох нейронів ініціалізовані випадковими значеннями. Їх початкові та кінцеві значення після навчання відзначені X на рис. 3а та 3б відповідно. Кожна з трьох груп прикладів виявлена одним із вихідних нейронів, чий ваговий вектор налаштувався на центр тяжкості виявленої групи.

Навчанням штучної нейронної мережі називається процес підстроювання значень ваг зв'язків між нейронами мережі. У ЗНМ використовується навчання з учителем, що припускає використання навчальної вибірки для порівняння вихідного сигналу мережі з еталонним значенням навчальної вибірки, розрахунок помилки і підстроювання ваг зв'язків мережі з метою зменшення значення цієї помилки.

У ЗНМ використовується алгоритм градієнтного спуску або зворотного поширення помилки і його модифікації. Для повнозв'язних і згорткових шарів

розраховується значення помилки вихідного сигналу мережі за загальною формулою (9).

$$E(\omega) = \frac{1}{2}\sum_{i,k} \quad (f_{ik} - y_{ik})^2, \qquad (1.9)$$

де *E (w)* - властивість помилки мережі;

$f_{ik}$ - значення вихідного сигналу k-го нейрона мережі при подачі і-го зразка навчальної вибірки;

$y_{ik}$ - очікуване значення вихідного сигналу k-го нейрона мережі при подачі і-го зразка навчальної вибірки.

Навчання мережі спрямовано на мінімізацію функції помилки. Значення функції помилки для шару мережі визначається за наступною формулою (10).

$$\delta_i^{(q)} = (f_{ik}^{(q)}(S))' \sum_j \quad w_{ij}\delta_j^{(q+1)} \qquad (1.10)$$

де $\delta_i^{(q)}$ - помилка і-го нейрона в шарі q;

$(f_{ik}^{(q)}(S))'$- похідна функції активації і-го нейрона шару q для k- го зразка навчальної вибірки;

*S* - сигнал, що подається на вхід і-го нейрона мережі;

$w_{ij}$ - вага зв'язку, що з'єднує і-ий нейрон шару q та j-ий нейрон шару $(q + 1)$,

$\delta_j^{(q+1)}$ - помилка j-го нейрона в шарі $(q + 1)$.

На основі розрахованої помилки нейрона визначається зміна ваги зв'язку цього нейрона за формулою (1.11)

$$\Delta w_{ij}^{(q)} = -\eta\delta_j x_i, \qquad (1.11)$$

де $\Delta w_{ij}^{(q)}$ - значення зміни ваги зв'язку, що з'єднує і-ий нейрон шару q і j-ий нейрон шару $(q + 1)$;

$\eta$ - значення, що визначає швидкість навчання, j - значення помилки j-го нейрона в шарі q;

$x_i$ - значення i-го вхідного сигналу *[7]*.

У згорткових шарах ЗНМ вищенаведені формули використовуються для настройки ваг зв'язків для кожного фільтра.

Для шарів субдискретизації розрахунок помилки не проводиться, тому що дані шари не беруть участі в навчанні мережі. При використанні функції шару субдискретизації max (вибір максимального значення) помилка з шару мережі, розташованого після цього шару, відразу переміщається на згортковий шар мережі, що передує даному шару субдискретизації, на вихідне значення згорткового шару, яке відповідає «виграв» значенням субдискретизації шару. При використанні функції шару субдискретизації avg (розрахунок середнього значення) помилка з шару мережі, розташованого після цього шару, ділиться на кількість елементів вікна шару і переноситься на всі значення згорткового шару, що передує даному.

### 1.6. Алгоритм створення ефективної архітектури згорткової мережі

Задачу побудови архітектури згорткової нейронної мережі для класифікації вхідного кольорового зображення зводиться до «згортання» вхідного шару мережі до верствам з найменшими розмірами, наприклад, 2 x 2 x D або 1 x 1 x D, де D - глибина шару, і подальшого перетворення до C одновимірним сигналам ймовірності приналежності зразка, що надійшов на вхід мережі, до одного з C класів.

Пропонований підхід являє собою алгоритм, який регламентує вибір параметрів архітектури ЗНМ виходячи з основних характеристик вхідних даних на кожному етапі послідовного створення шарів мережі. Алгоритм передбачає опціональне введення основних параметрів мережі і випадкове задачу деяких значень параметрів, що означає, що результатами алгоритму в результаті його виконання для різних вхідних даних може бути набір архітектур ЗНМ, кожна з

яких буде ефективною і буде має високі показники точності класифікації на вихідних даних .

В описі алгоритму прийняті наступні позначення:

N - розмір квадратного попереднього шару мережі (для першого згорткового шару є розміром вхідного шару).

D - глибина вхідного шару мережі (кількість колірних каналів зображення)

C - кількість класів, належність до яких визначається класифікатором.

P - кількість рядків і стовпців, що містять нулі, що додаються до межами шару, що передує згортковому шару (параметр архітектури згорткового шару).

S - зміщення між фільтрами при формуванні сигналів нейронів згорткового шару / зсув між вікнами при формуванні сигналів шару Субдискретизація.

F - розмір квадратних фільтрів згорткового шару, причому F може приймати значення 3, 5 або 7.

Filters - глибина згорткового шару (кількість фільтрів).

Filtersp - глибина попереднього згорткового шару (глобальний параметр циклу створення шарів мережі).

AFc - властивість активації нейронів згорткового шару.

U - розмір квадратного вікна для шару Субдискретизація.

Subf - тип функції шару Субдискретизація (max - максимум, або avg - розрахунок середнього).

K - кількість нейронів в повнозв'язну шарі.

AFfc - властивість активації повнозв'язного шару

Рівень мережі - актуальна послідовність з n згорткових шарів і m шарів Субдискретизація, причому для рівня дані змінні приймають цілочисельні значення з діапазонів: n [1; 2], m [0; 1].

d - глибина мережі - кількість рівнів мережі (глобальний параметр циклу створення шарів мережі).

Текстовий опис етапів алгоритму створення архітектури ЗНМ

1. Опціональне введення основних параметрів алгоритму: F - ніж великі особливості необхідно детектувати на зразках даних, тим більше рекомендується

ставити значення F (за замовчуванням F не ставить), параметри рівня мережі: n, m - чим складніше зображення, тим більше рекомендується ставити значення n ( за замовчуванням n = 1, m = 1).

2. Створення параметрів вхідного шару мережі: введення розміру вхідного зображення, що має квадратну форму, зі значенням розміру боку, рівним N, причому N має багаторазово ділитися на 2 аж до однозначних чисел. Введення глибини вхідного шару D, зазвичай рівного кількості колірних каналів зображення. Введення значення кількості класів C.

3. Ініціалізація глобальних змінних алгоритму: d = 0, Filtersp = 0.

4. Ініціалізація змінної циклу створення згорткових шарів in = 0.

5. Якщо значення in не дорівнює n, то проводиться створення параметрів згорткового шару, інакше проводиться перехід до пункту 9. F вибирається залежно від N за формулою (1.12). Filters вибирається залежно від значень параметрів F і d по формулі (1.13). Після вибору значення F, проводиться операція Filtersp = Filters.

а.) Якщо in = 0, то проводиться створення першого згорткового шару, інакше перехід до пункту 5.б: P і S розраховуються шляхом вирішення системи рівнянь - формула (1.14), отриманої з формул (2) і (3) для шарів в яких ширина і висота рівні N і розмір попереднього шару дорівнює розміру згорткового шару.

$$F = \begin{cases} 7, if\,(N \geq 64) \\ 5, if\,(32 \leq N < 64) \\ 3, if\,(N < 32) \end{cases} \qquad (1.12)$$

$$\begin{cases} S = (N - F + P * 2)/(N - 1) \\ P = ((N - 1)S - N + F)/2 \end{cases} \qquad 1.13)$$

$$FIlters = \begin{cases} 8, if\,(F = 7 \,\&\&\, d = 0) \\ 16, if\,(F = 5 \,\&\&\, d = 0) \\ 24, if\,(F = 3 \,\&\&\, d = 0) \\ Filters_p \,||\, 1{,}25 * Filters_p, if\,(d > 0) \end{cases} \qquad (1.14)$$

б.) Проводиться створення параметрів інших згорткових шарів: P вибирається мінімальним цілочисельним з відрізка [0; F] таким чином, щоб розміри згорткового шару були цілочисельними відповідно до формул (2) і (3). S вибирається мінімальним цілочисельним з відрізка [1; F] таким чином, щоб розміри згорткового шару були цілочисельними відповідно до формул (2) і (3).

6. Розрахунок N для сформованого згорткового шару за формулою (15), отриманої з формул (1.2) і (1.3), де ширина і висота зображення рівні.

$$N = \frac{(N_p F + P * 2)}{S} + 1 \qquad (1.15)$$

де *Np* - розмір квадратного попереднього шару мережі.

7. Вибір в якості функції активації нейронів згорткового шару порогової функції, згідно з формулою (1.16).

$$AF_c = max(0, x), \qquad (1.16)$$

де *x* - сигнал нейрона згорткового шару, отриманий в результаті фільтрації.

8. *in = in + 1* та перехід до пункту 5.

9. Якщо *m = 1* і *N > 1*, то проводиться створення параметрів шару Субдискретизація, інакше перехід до пункту 11: U вибирається рівним значенню 2. S вибирається рівним U. Subf вибирається як властивість max - вибір максимального значення.

10. Розрахунок N для сформованого шару субдискретизації мережі за формулою (1.17), отриманої з формул (1.7) і (1.8), де ширина і висота зображення рівні.

$$N = \frac{N_p - U}{S} + 1, \qquad (1.17)$$

де *Np* - розмір квадратного попереднього шару мережі.

11. *d = d + 1*.

12. Ухвалення рішення про створення наступного рівня мережі: якщо N >= 3, то проводиться створення нового рівня мережі - перехід до пункту 4, інакше проводиться перехід до пункту 13.

13. Створення параметрів повнозв'язного шару мережі: K вибирається рівним C. Як функції активації нейронів повнозв'язного шару використовується сигмоїдна властивість згідно з формулою (1.18).

$$AF_{fc} = \frac{1}{1 + e^{-x}}, \qquad (1.18)$$

де *x* - сигнал нейрона повнозв'язного шару, отриманий в результаті зваженого підсумовування вхідних сигналів нейрона.

14. Створення вихідного шару мережі, призначеного для розрахунку ймовірностей приналежності вхідного зображення представленим класів, кількість яких одно C.

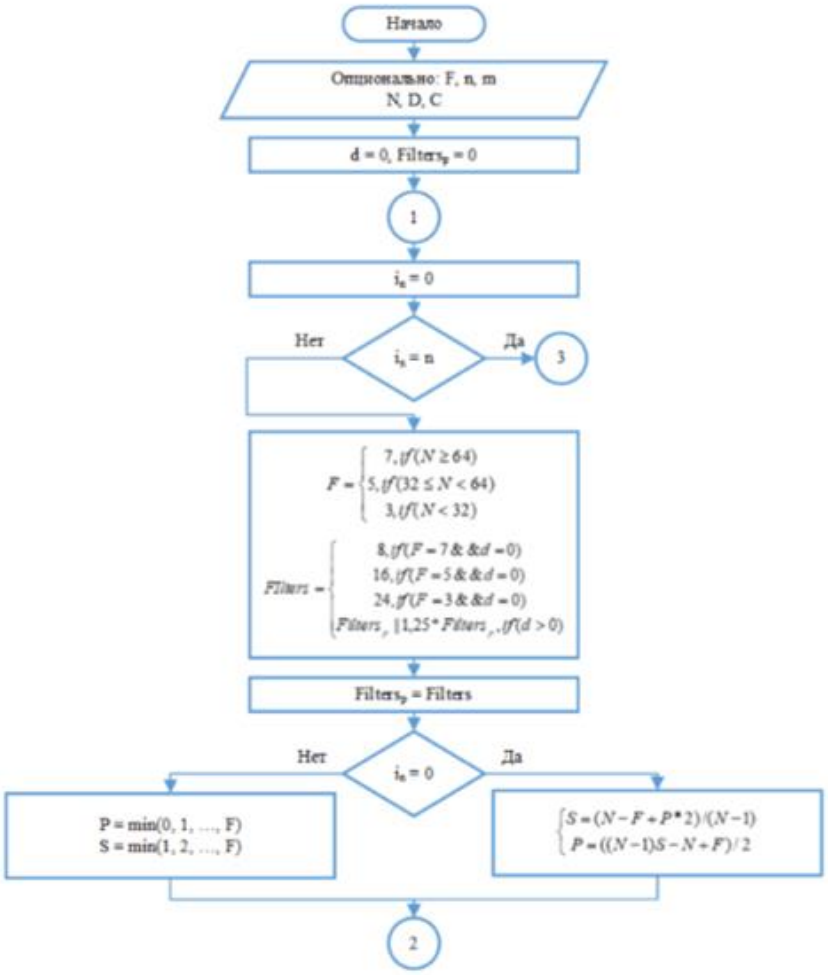


Рис. 1.6. Блок-схема алгоритму створення архітектури ЗНМ

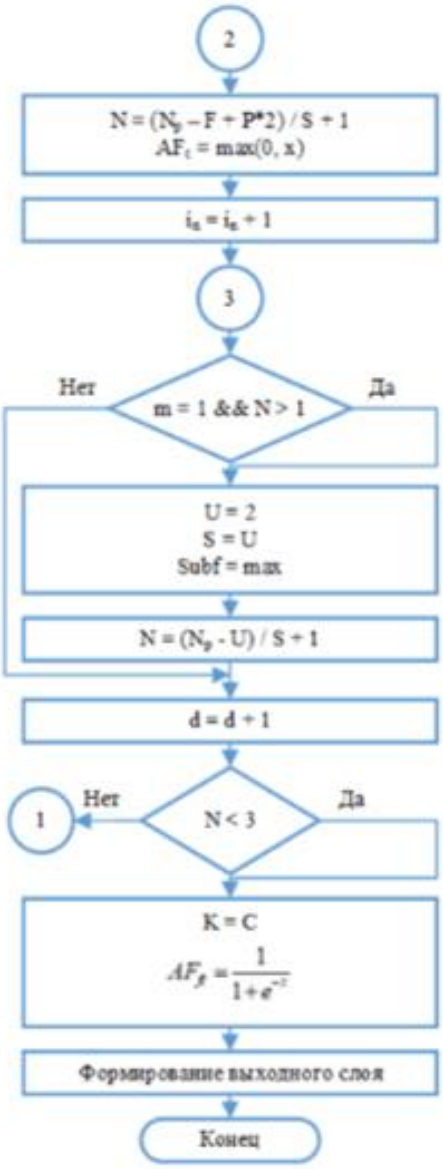


Рис. 1.7. Блок-схема алгоритму створення архітектури ЗНМ (продовження)

# РОЗДІЛ 2
# НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

Навчання нейронної мережі - це процес, у якому параметри нейронної мережі налаштовуються за допомогою моделювання середовища, в яке ця мережа вбудована. Тип навчання визначається способом підстроювання параметрів. Розрізняють алгоритми навчання з учителем та без вчителя. Процес навчання з учителем є пред'явленням мережі вибірки навчальних прикладів. Кожен зразок подається на входи мережі, потім проходить обробку всередині структури НМ, обчислюється вихідний сигнал мережі, який порівнюється з відповідним значенням цільового вектора, що являє собою необхідний вихід мережі.

Навчання нейронної мережі- це процес, у якому параметри нейронної мережі налаштовуються за допомогою моделювання середовища, в яке ця мережа вбудована. Тип навчання визначається способом підстроювання параметрів.

При навчанні без вчителя навчальна множина складається лише з вхідних векторів. Навчальний алгоритм підлаштовує ваги мережі те щоб узгоджувалися вихідні вектори, тобто. щоб пред'явлення досить близьких вхідних векторів давало однакові виходи.

Процес навчання, отже, виділяє статистичні властивості навчальної множини та групує подібні вектори до класів. Пред'явлення на вхід вектора з даного класу дасть певний вихідний вектор, але до навчання неможливо передбачити, який буде вироблятися даним класом вхідних векторів. Отже, виходи такої мережі мають трансформуватися на деяку зрозумілу форму, зумовлену процесом навчання. Це не є серйозною проблемою. Зазвичай не складно ідентифікувати зв'язок між входом та виходом, встановлений мережею. Для навчання нейронних мереж без вчителя застосовуються сигнальні методи навчання Хебба і Ойа.

Розрізняють алгоритми навчання з учителем та без учителя. Процес навчання з учителем є пред'явленням мережі вибірки навчальних прикладів. Кожний зразок подається на входи мережі, потім проходити обробку всередині структури НМ, обчислюється вихідний сигнал мережі, який порівнюється з відповідним значенням цільового вектора, що являє собою необхідний вихід мережі.

Навчання нейронних мереж ділиться на категорії: навчання з учителем та навчання без вчителя.

Для навчання з учителем необхідно розмітити дані, що є парою значень. Перше значення кожної пари – вхідні дані, друге значення – еталон. Еталон - це ідеальні значення нейронної мережі, результат, якого вона прагне. Дані подаються на вхід нейронної мережі, потім результати виходу мережі звіряються з стандартом.

Для того, щоб знайти відхилення від необхідного результату, вводиться властивість помилки, яка є цільовою функцією, що мінімізується:

$$E(w) = \frac{1}{2}\sum_{j\,k}(y_{jk} - d_{jk})^2$$

де, $y$ – реальне вихідне значення нейрона j вихідного шару при подачі на $j_k$ вхід значення $k$, а $d$ – необхідне вихідне значення цього нейрона. $j_k$ У процесі навчання нейромережі мінімізація помилки відбувається шляхом оновлення ваг кожного шару нейромережі.

Оновлене значення розраховується за такою формулою:

$$\Delta w_{ij}^{q} = -n\frac{\delta E}{\delta w_{ij}}$$

де $w_{ij}$ ваговий коефіцієнт, який з'єднує $i$ - тий нейрон шару $q$ - $1$ $j$ - тим нейроном шару $q$, $n$ - коефіцієнт швидкості навчання, $0 < n < 1$.

Математично процес навчання можна описати в такий спосіб. У процесі функціонування нейронна мережа формує вихідний сигнал $Y$, реалізуючи певну функцію $Y = G(X)$. Якщо архітектура мережі задана, вид функції $G$ визначається

значеннями синаптичних ваг і зміщеної мережі. Нехай рішенням деякої задачі є властивість $Y = F(X)$, задана параметрами вхідних-вихідних даних $(X_1, Y_1)$, $(X_2, Y_2)$, …, $(X_N, Y_N)$, для яких $Y_k = F(X_k)$ $(k = 1, 2, …, N)$.

### 2.1. Навчання з вчителем

Під час навчання ШНМ з учителем кожному елементу з навчальної вибірки відповідає вектор, який певним чином характеризує лише однозначно правильну відповідь, що подається відразу на вихід мережі в обхід всієї її архітектури. Наступним кроком після отримання власного результату мережі є порівняння алгоритмом результуючого вектора з правильною відповіддю.На основі отриманих результатів відбувається корекція подальша помилки. Класичне правило навчання з учителем коротко можна описати як правило корекції помилки, або зворотній метод поширення помилок.

### 2.2. Навчання без вчителя

Навчання без вчителя в разі ШНМ відбувається максимально природно в процесі навчання, коли автоматична настройка параметрів мережею веде до появи однакових результатів її діяльності при доволі близьких вхідних значеннях, що на практиці можна порівняти зі зниженням розмірності даних в результаті ітераційного методу головних компонент. Правило Хеба, засноване на гіпотезі про посилення зв'язків між біологічними нейронами в разі їх одночасного збудження, і методи навчання при змаганні нейронів шляхом порівняння інтенсивності їх реакції є класичними прикладами методів навчання без учителя.

З теореми про відображенні практично будь-якої функції за допомогою багатошарової нейромережі випливає, що нейронна мережа, що навчається нами, в принципі здатна сама підлаштуватися під будь-які дані з метою мінімізації сумарної квадратичної помилки. Щоб цього не відбувалося, під час навчання нейромереж використовують наступний спосіб перевірки мережі. Для цього

навчальну вибірку ще перед початком навчання розбивають випадково на дві підвиборки: навчальну та тестову. Навчальну вибірку використовують власне процесу навчання, у своїй змінюються ваги нейронів. А тестову використовують у процесі навчання для перевірки на ній сумарної квадратичної помилки, але при цьому не відбувається зміна ваги. Якщо нейромережа показує поліпшення апроксимації і на навчальній, і на тестовій вибірках, навчання мережі відбувається в правильному напрямку. Інакше може знижуватись помилка на навчальній вибірці, але відбувається її збільшення на тестовій. Останнє означає, що мережа "перенавчилася" і вже не може бути використана для прогнозування чи класифікації. У цьому випадку трохи змінюються ваги нейронів, щоб вивести мережу з околиці локального мінімуму помилки.

### 2.3. Метод зворотного поширення помилки (Backpropagation)

Даний метод є класичним методом навчання з учителем, заснований на правилі корекції помилково і був розроблений як метод навчання багатошарового перцептрона. Головна ідея методу заключається в поширенні сигналів помилки вже після її обрахування на виході мережі в напрямку, зворотному прямому поширенні сигналів під час звичайного обчислювального процесу, від виходів мережі до її входів. При зворотному проході синаптичні ваги налаштовуються з метою мінімізації помилки.

$$E(\{\omega_{i,j}\}) = \frac{1}{2}\sum_{k \in OUT} (t_k - o_k)^2 \qquad (2.1)$$

де $E$ - властивість помилки;

$\omega_{i,i}$- синаптична вага між нейронами $i$, $j$;

$OUT$ - безліч виходів мережі;

$t_k$ - правильні відповіді мережі, $k \in$ OUT;

$o_k$ - *вихід $k$-го нейрона.*

Якщо коротко, тут реалізується стохастический градієнтний спуск, відбувається рух в багатовимірному просторі ваг до мінімуму помилки в сторону, протилежну градієнту. Для кожної групи правильних відповідей, відбувається налаштування синаптичних ваг шляхом обрахування:

$$\Delta\omega_{i,j} = -\eta \frac{dE}{d\omega_{i,j}}, \qquad (2.2)$$

де $0 < \eta < 1$ - множник швидкості руху.

В процесі навчання циклічно розв'язується однокритеріальна задача оптимізації. При можливості реалізації методу передавальна властивість нейронів повинна бути диференційована. Так як метод є модифікацією класичного методу градієнтного спуску, варто розглядати як градієнтний спуск по поверхні помилки.

Спосіб зворотного поширення помилки можна поділити на 4 окремі блоки: пряме поширення, функцію втрати, зворотне поширення і оновлення ваги. Під час прямого поширення, береться тренувальне зображення, це матриця N x N x 3 - і пропускається через всю мережу.

Тобто, у першому навчальному прикладі, так як всі ваги або значення фільтра були ініційовані випадковим чином, вихідним значенням буде щось на зразок [.0 .0 .0 .0 .0 .0 .0 .0 .0 .0], тобто таке значення, яке не дасть переваги якомусь певному числу. Мережа з такими вагами не може знайти властивості базового рівня і не може обґрунтовано визначити клас зображення. Це веде до функції втрати. Припустимо, перше навчальне зображення - це цифра 3. Ярликом зображення буде [0 0 0 1 0 0 0 0 0 0]. Властивість втрати може бути виражена по-різному, але часто використовується СКО (середньоквадратична помилка), це 1/2 помножити на (реальність - передбачення) в квадраті:

$$E = \sum 1/2(target\text{-}output)^2 \qquad (2.3)$$

Ми хочемо досягти того, щоб прогнозований ярлик (висновок згорткового шару) був таким же, як ярлик навчального зображення (це означає, що мережа зробила правильне припущення). Для того, щоб отримати такий результат, нам необхідно привести до мінімуму кількість втрат, яке у нас є. Візуалізуючи це як задачу оптимізації з математичного аналізу, нам необхідно з'ясувати, які входи найбезпосереднішим чином провокували втрат (або помилок) мережі. Задачу мінімізації втрат - налаштувати ваги так, щоб знизити втрату. Тобто, візуально нам необхідно підійти до найнижчій точці чаше-подібного об'єкта.

Всупереч на широке успішне застосування методу, він не позбавлений ряду недоліків, наприклад, таких як дуже низька швидкість навчання і принципова невдача навчання мережі (наприклад, нескінченна навчання), можлива по ряду причин, серед яких параліч мережі, проблема локальних мінімумів і проблема розміру кроку.

Усі ці недоліки вирішуються різними конфігураціями методу, наприклад, конфігурація RProp (Resilient Propagation) значно пришвидшує процес навчання завдяки обчисленню лише знаків приватних похідних для підстроювання вагових коефіцієнтів, замість обрахування повних похідних. Наприклад, RProp використовує метод навчання по епохах. Існують інші методи, що прискорюють процес навчання: QuickProp, метод сполучених градієнтів, метод Левенберга — Марквардта.

# РОЗДІЛ 3
# ГЕНЕТИЧНІ АЛГОРИТМИ

Генетичні алгоритми в даний час широко використовуються для інтелектуальної обробки даних та вирішення задач оптимізації та пошуку. Вони успішно використовуються для вирішення низки економічно значущих завдань у бізнесі та інженерних розробках. Фінансові компанії широко використовують генетичні алгоритми прогнозування розвитку фінансових ринків.

Генетичні алгоритми виникли в результаті спостереження та спроб копіювання природних процесів, що відбуваються у світі живих організмів, зокрема, розвитку та пов'язаної з нею селекції (природного відбору) популяцій живих істот.

Ідею генетичних алгоритмів висловив Джон Холланд у 1975 році. Він зацікавився властивостями процесів природної розвитку, зокрема фактом, що еволюціонують хромосоми, а чи не самі живі істоти. Холланд був упевнений у можливості скласти та реалізувати у вигляді комп'ютерної програми алгоритм, який вирішуватиме складні завдання так, як це робить природа – шляхом розвитку. Тому він почав працювати над алгоритмами, що оперували послідовностями двійкових цифр (одиниць і нулів), що отримали назву хромосом. Ці алгоритми імітували еволюційні процеси у поколіннях таких хромосом. Так само, як і в природі, генетичні алгоритми здійснювали пошук "хороших" хромосом без використання будь-якої інформації про характер вирішуваного завдання. Потрібна була тільки якась оцінка кожної хромосоми, що відображає її пристосованість. [7] Генетичні алгоритми застосовуються розробки програмного забезпечення, в системах штучного інтелекту, оптимізації, штучних нейронних мережах та інших галузях знань. Слід зазначити, що з допомогою вирішуються завдання, котрим раніше використовувалися лише нейронні мережі.

У цьому випадку генетичні алгоритми виступають просто в ролі незалежного від нейронних мереж альтернативного методу, призначеного для вирішення того ж завдання. Генетичні алгоритми часто використовуються разом із нейронними мережами. Вони можуть підтримувати нейронні мережі або навпаки або обидва методи взаємодіють у рамках гібридної системи, призначеної для вирішення конкретного завдання. Генетичні алгоритми також застосовуються разом із нечіткими системами.

Генетичні алгоритми – адаптивні методи пошуку, які останнім часом часто використовуються для вирішення задач функціональної оптимізації. Вони засновані на генетичних процесах біологічних організмів: біологічні популяції розвиваються протягом кількох поколінь, підкоряючись законам природного відбору і за принципом "виживає найпристосованіший", відкритий Чарльзом Дарвіном. Наслідуючи цього процесу генетичні алгоритми здатні "розвивати" вирішення реальних завдань, якщо ті відповідним чином закодовані.

Основний закон наслідування інтуїтивно зрозумілий кожному - він у тому, що нащадки схожі батьків. Зокрема, нащадки більш пристосованих батьків будуть, найімовірніше, одними з найпристосованіших особин у своєму поколінні. З біології ми знаємо, що будь-який організм може бути представлений своїм фенотипом, який фактично визначає, чим є об'єкт у реальному світі, та генотипом, який містить всю інформацію про об'єкт на рівні хромосомного набору. У цьому кожен ген, тобто елемент інформації генотипу, має свій відбиток у фенотипі. Таким чином, для вирішення завдань нам необхідно представити кожну ознаку об'єкта у формі, що підходить для використання в генетичному алгоритмі. Все подальше функціонування механізмів генетичного алгоритму проводиться на рівні генотипу, дозволяючи обійтися без інформації про внутрішню структуру об'єкта, що і зумовлює його широке застосування в різних завданнях. [6]

Генетичні алгоритми, це велике різноманіття евристичних алгоритмів що найчастіше використовують для підбору параметрів, коефіцієнтів, оптимальних рішень для різного рівня задач моделювання при цьому користуючись методами та механізмами схожими на процес природного відбору в природі. Входить до

класу еволюційних методів, якими обчислюють оптимізацію з використанням методів схожих на процес розвитку: селекція, мутації і поєднання. Важливою властивістю, що присутня у генетичних алгоритмах є обов’язковість шагу поєднання, який слугує оператором рекомбінації кандидатів, що дуже схоже на роль поєднання в звичайній природі.

### 3.1. Узагальнений опис алгоритму

У найчастішому різновиді генетичного алгоритму для представлення генотипу об'єкта застосовуються бітові рядки. При цьому кожному атрибуту об'єкта у фенотипі відповідає один ген генотипу об'єкта. Ген являє собою бітовий рядок, найчастіше фіксованої довжини, яка є значенням цієї ознаки.

Для кодування таких ознак можна використовувати найпростіший варіант – бітове значення цієї ознаки. Тоді буде дуже просто використовувати ген певної довжини, достатньої уявлення всіх можливих значень такої ознаки.

Таким чином, для того щоб визначити фенотип об'єкта (тобто значення ознак, що описують об'єкт) нам необхідно тільки знати значення генів, відповідним цим ознаками, тобто генотип об'єкта. У цьому сукупність генів, що описують генотип об'єкта, є хромосому. У деяких реалізаціях її також називають особиною. У реалізації генетичного алгоритму хромосома є бітовий рядок фіксованої довжини. У цьому кожному ділянці рядки відповідає ген. Довжина генів усередині хромосоми може бути однаковою або різною. Найчастіше застосовують гени однакової довжини. [9]

Генетичні алгоритми працюють із сукупністю "особин" – населенням, кожна з яких представляє можливе вирішення цієї проблеми. Кожна особина оцінюється мірою її "пристосованості" відповідно до того, наскільки "добре" відповідне їй вирішення завдання. У природі це еквівалентно оцінці того, наскільки ефективним є організм при конкуренції за ресурси. Найбільш пристосовані особини набувають можливість "відтворювати" потомство за допомогою "перехресного поєднання" з іншими особами популяції. Це призводить до появи

нових особин, які поєднують деякі властивості, успадковані ними від батьків. Найменш пристосовані особини з меншою ймовірністю зможуть відтворити нащадків, так що ті властивості, які вони мали, будуть поступово зникати з популяції в процесі розвитку. Іноді відбуваються мутації, чи спонтанні зміни у генах.[13]

Таким чином, з покоління до покоління, хороші властивості поширюються по всій популяції. Поєднання найбільш пристосованих особин призводить до того, що досліджуються найперспективніші ділянки простору пошуку. Зрештою населення буде сходитися до раціонального вирішення завдання. Перевага генетичних алгоритмів у тому, що він знаходить приблизні оптимальні рішення щодо короткий час.

Генетичний алгоритм складається з наступних компонентів:

- Хромосоми. Вирішення розглянутої проблеми. Складається із генів.
- Початкова популяція хромосом.
- Набір операторів для генерації нових рішень із попередньої популяції.
- Цільова властивість для оцінки пристосованості рішень. [11]

Стандартні оператори для всіх типів генетичних алгоритмів це: поєднання, мутація та селекція.

Як відомо теоретично розвитку важливу роль грає те, як ознаки батьків передаються нащадкам. У генетичних алгоритмах за передачу ознак батьків нащадкам відповідає оператор, який називається поєднання (його також називають кросовер або кросинговер). Цей оператор визначає передачу ознак батьків нащадкам.

Чинний він так:

1. З популяції вибираються дві особи, які будуть батьками;
2. Визначається (зазвичай випадковим чином) точка розриву;
3. Нащадок визначається як конкатенація частини першого та другого з батьків.

Таким чином, оператор поєднання здійснює обмін частинами хромосом між двома хромосомами в популяції, тобто створює структуру, засновану на двох

структурах - заміною однієї частини першої структури на ту ж область в другій. Потім з ймовірністю 0.5 визначається одна з результуючих хромосом як нащадок.

Наступний генетичний оператор призначений для того, щоб підтримувати різноманітність особин у популяції. Він називається мутацією.

При використанні даного оператора кожен біт у хромосомі з певною ймовірністю інвертується. Крім того, використовується ще й так званий оператор інверсії, який полягає в тому, що хромосома поділяється на дві частини, а потім вони змінюються місцями.

Задача формується так, щоб її рішення було представлено у вигляді вектора. Цей вектор називається генотипом, де ген (склада генотипу) може бути певним бітом або якоюсь іншою величиною в залежності від задачі, що вирішується. У канонічних варіантах реалізацій генетичних алгоритмів генотип завжди має одну й ту саму довжину, тобто вона є фіксованою. Але варто відмітити, що в сучасних версіях часто немає ніякого обмеження по довжині генотипу.

Для першої популяції, як правило, створюється велика кількість випадкових генотипів. Їх оцінка відбувається за допомогою так званої функції пристосування (або англійською *fitness*), яка визначає наскільки ефективним є генотип. Важливим критерієм вибору функції пристосування є її рельєф - необхідно, щоб він був досить “гладким”.

Після запуску моделі з кінцевої кількості рішень вибираються “переможці” за значенням функції пристосування у кожного генотипа. До цих рішень застосовуються оператори поєднання і мутації і як результат отримуються нові набори рішень. Для них в свою чергу теж вираховується значення функції пристосування, і проводиться відбір рішень в наступне покоління, що показали найкращі результати.

Цей алгоритм є ітеративним і таким чином відтворюється еволюційний процес, що може тривати певну кількість поколінь в залежності від заданих критеріїв його роботи. Одними із видів критеріїв є:

- глобальний оптимум знайдено
- досягнуто граничну кількість поколінь

- завершення по тайм-ауту

Генетичні алгоритми використовують в більшій мірі для знаходження апроксимованих рішень в багатовимірних системах і просторах.

Тому зазвичай виділяються слідуючі кроки для загального генетичного алгоритму:

1) Визначити *fitness* функцію для всіх генотипів
2) Згенерувати першу популяцію генотипів
3) Провести операцію поєднання
4) Випадкові мутації
5) Обрахування *fitness* функції
6) Селекція та створення наступного набору генотипів

### 3.2. Опис основних операцій генетичних алгоритмів

Для функціонування генетичного алгоритму достатньо цих двох генетичних операторів, але практично застосовують ще деякі додаткові оператори чи модифікації цих двох операторів. Наприклад, кросовер може бути не однокрапковий (з однією точкою розриву), а багатоточковий, коли формується кілька точок розриву (найчастіше дві). Крім того, в деяких реалізаціях алгоритму оператор мутації є інверсією лише одного випадково обраного біта хромосоми.

Оператор селекції здійснює відбір хромосом відповідно до значень їх функції пристосованості.

Найбільш ефективні два механізми відбору - елітний відбір та відбір з витісненням.

Ідея елітного відбору заснована на побудові нової популяції лише з кращих особин репродукційної групи, що об'єднує у собі батьків, їхніх нащадків та мутантів. В основному це пояснюють потенційною небезпекою передчасної збіжності, віддаючи перевагу пропорційному добору.

Швидка збіжність, що забезпечується елітним відбором, можливо, коли це необхідно, з успіхом компенсована відповідним методом вибору батьківських пар, наприклад, аутбридингом. Саме така комбінація "аутбридинг – елітний відбір" є однією з найефективніших. [3]

Другий спосіб – це відбір витісненням. Чи буде особина з репродукційної групи заноситись у популяцію нового покоління, визначається як величиною її пристосованості, а й тим, чи є у формованої популяції наступного покоління особина з аналогічним хромосомним набором. З усіх особин з однаковими генотипами перевага спочатку, звичайно ж, надається тим, чия пристосованість вища.

Таким чином, досягаються дві мети: по-перше, не втрачаються кращі знайдені рішення, що мають різні хромосомні набори, а по-друге, у популяції постійно підтримується достатня генетична різноманітність. [8]

Однією з основних операцій, що виконується під час роботи генетичного алгоритму є операція операція селекції. Під селекцією мається на увазі те, що після певної кількості моделювань вибираються найбільш ефективні генотипи серед усіх, що є в системі.

Ця операція допомагає прискорювати моделювання і швидше знаходити вірне або близьке до вірного рішення за рахунок відкидання найменш ефективних генотипів.

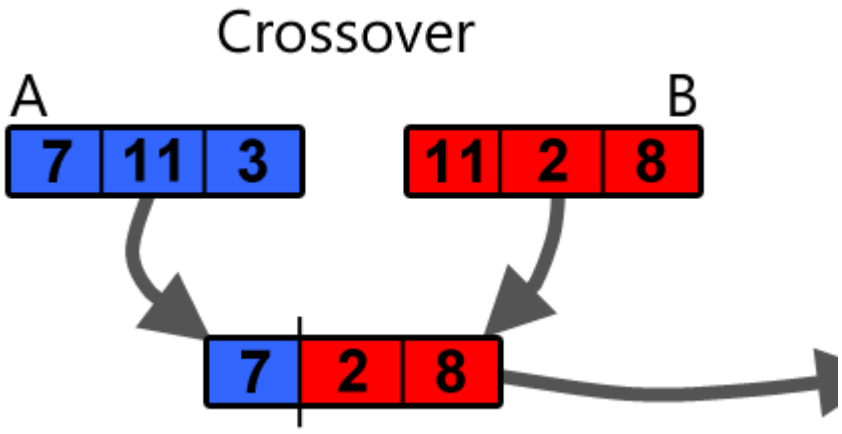

Рис. 3.1. Приклад механізму поєднання на випадковому генотипі

Іншою важливою операцією являється операція поєднання, яка часто виконується після селективного відбору найефективніших генотипів - це дає змогу отримати новий генотип, що заснований на декількох найефективніших. Операція поєднання може призводити до створення як і одного, так і декількох нових генотипів.

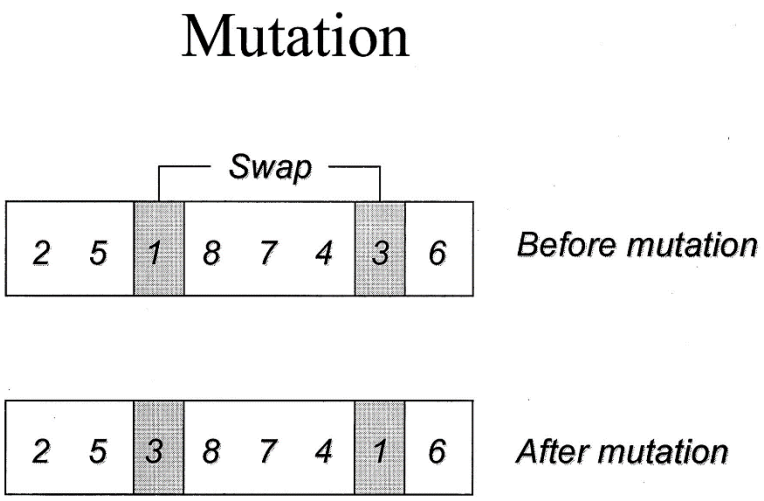


Рис. 3.2. Приклад механізму мутації на випадковому генотипі

Третьою операцією є операція мутацій, що випадково приміняється до випадкових генів у генотипі. Цей евристичний елемент дає змогу генерувати нові випадкові генотипи, які потенційно можуть припривести до отримання більш ефективних варіацій генотипів. Мутації можуть являти собою будь-які випадкові зміни у генотипі, наприклад, зміна генів місцями, або множення певних генів на випадкові коефіцієнти.

Майже кожна реалізація генетичних алгоритмів включає в себе усі три операції, що максимізує його результат.

### 3.3. Тренування нейронних мереж за допомогою генетичних алгоритмів

Генетичні алгоритми також досить добре вирішують задачі для тренування нейронних мереж, знаходження оптимальних топологій та оптимізації існуючих вагових коефіцієнтів.

Такі алгоритми можуть дозволяти створювати повністю працюючі рішення для нейромереж маючи у розпорядженні лише сам набір даних для навчання, і, що найголовніше, не потребує від користувача глубоких пізнань у сфері штучного інтелекту.

Хоча необхідно завжди пам’ятати, що при навчанні мережі з вчителем на статичному наборі інформації за допомогою більш класичних алгоритмів таких як алгоритм зворотного поширення помилки часто можна досягти навіть кращих результатів ніж за допомогою нейроеволюційного варіанту.

Нейроеволюційні алгоритми краще підходять для випадків, коли методи, що включають в себе градієнтний спуск не можуть вирішити задачу якісно. Наприклад, задачі навчання мережі для орієнтації в середовищі, що динамічно змінюється або знаходження стратегій поведінки “агентів” у середовищі. Важливим фактором є те, що при такому моделюванні значення необхідних виходів нейронної мережі вже відомі. Таким чином якість роботи мережі вимірюють за допомогою цільової функції.

Однією з переваг даних алгоритмів є те, що вони слабо залежать від конкретного виду задачі і їх можна приміняти для вирішення цілого спектру різних проблем.

Генотип нейронної мережі можна закодувати наступними способами:

- За допомогою вагових коеф. зв’язків
- За допомогою інформації про конфігурацію нейронів
- За допомогою кодування шарів

Існує відома проблема, що виникає, наприклад, при вирішенні задачі пошуку оптимальної топології мережі, що називається проблемою конкуруючих результатів (або рішень). Проблема заключається в тому, що мережу можна

представити у генотипі зовсім різними способами і це не дає інколи змогу провести операцію поєднання належним чином.

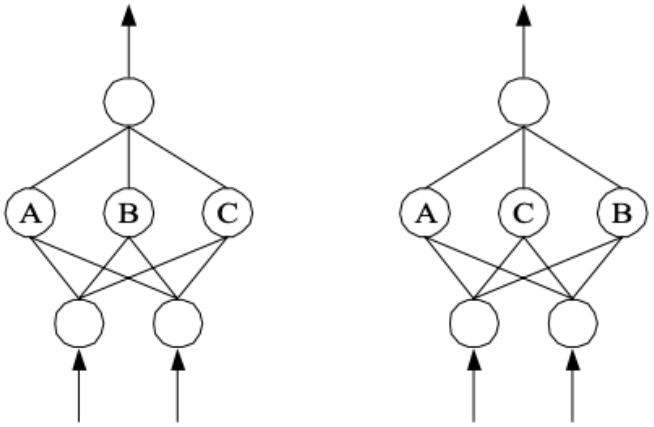


Рис. 3.3. Приклад проблеми конкуруючих результатів

Якщо генотипи відповідають мережам, що зображені на рис.2.3, і вагові коефіцієнти у нейронів у схованому шарі відповідають один одному, то при схрещуванні результуючий генотип мережі буде включати в себе наступні конфігурації ABB та ACC.

Дану проблему можна вирішити декількома способами: наприклад, за допомогою більш “розумного” алгоритму поєднання, що буде ідентифікувати однакові зв’язки у нейронах схованого шару і потім формувати новий генотип враховуючи ці дані. Іншим варіантом, буду нумерування кожної ітерації і вираховування унікального індексу для усіх зв’язків, що дасть змогу знаходити аналогічні топології.

Також очевидно для різних варіантів архітектур нейронних мереж необхідні свої оператори поєднання та мутацій, щоб забезпечити переміщення інформації від “батьків” до їх нащадків. В незалежності від кількості отриманих нащадків, то отримані генотипи повинні включати в себе генетичну інформацію про кожного з батьків, тобто не повинен відбуватися процес втрати інформації отриманого в результаті розвитку моделі.

Перший підхід найпростіший – це випадковий вибір батьківської пари, коли обидві особини, які складуть батьківську пару, випадково вибираються з усієї популяції, причому будь-яка особина може стати членом кількох пар. Незважаючи на простоту, такий підхід є універсальним для вирішення різних класів завдань. Однак він досить критичний до чисельності популяції, оскільки ефективність алгоритму, що реалізує такий підхід, знижується зі зростанням чисельності популяції.

Другий спосіб вибору особин у батьківську пару – так званий селективний. Його суть полягає в тому, що "батьками" можуть стати ті особи, значення пристосованості яких не менше середнього значення пристосованості по популяції, при рівній ймовірності таких кандидатів скласти шлюбну пару. Такий підхід забезпечує більш швидку збіжність алгоритму. Однак через швидку збіжність селективний вибір батьківської пари не підходить тоді, коли ставиться завдання визначення кількох екстремумів, оскільки таких завдань алгоритм, зазвичай, швидко сходиться одного з рішень.

Інші два способи формування батьківської пари – інбридинг та аутбридинг. Обидва ці методи побудовані на формуванні пари на основі близької та далекої "спорідненості" відповідно. Під "спорідненістю" тут розуміється відстань між членами популяції як у сенсі геометричної відстані особин у просторі параметрів. У зв'язку з цим розрізняють генотипний та фенотипний (або географічний) інбридинг та аутбридинг. Під інбридингом розуміється такий метод, коли перший член пари вибирається випадково, а другим з більшою ймовірністю буде максимально близька до нього особина. Аутбридинг, навпаки, формує шлюбні пари з максимально далеких особин. Використання генетичних інбридингу та аутбридингу виявилося ефективнішим порівняно з географічним для всіх тестових функцій при різних параметрах алгоритму. Найбільш корисним є застосування обох представлених методів для багатоекстремальних завдань. Однак ці два способи по-різному впливають на поведінку генетичного алгоритму. Так інбридинг можна охарактеризувати властивістю концентрації пошуку в локальних вузлах, що фактично призводить до розбиття популяції на окремі

локальні групи навколо підозрілих на екстремум ділянок ландшафту, а аутбридинг спрямований на попередження збіжності алгоритму до вже знайдених рішень, змушуючи алгоритм переглядати нові. [10]

# РОЗДІЛ 4
# ПРЕЗЕНТАЦІЯ ТА ОПИС СТВОРЕНОГО ДОДАТКУ

Результатом роботи над даним дипломним проектом є створення додатку, основна мета якого заключається у моделюванні середовища з агентами (або “мікроорганізмами”) та їжею, що випадковим чином розподілена по всьому полю.

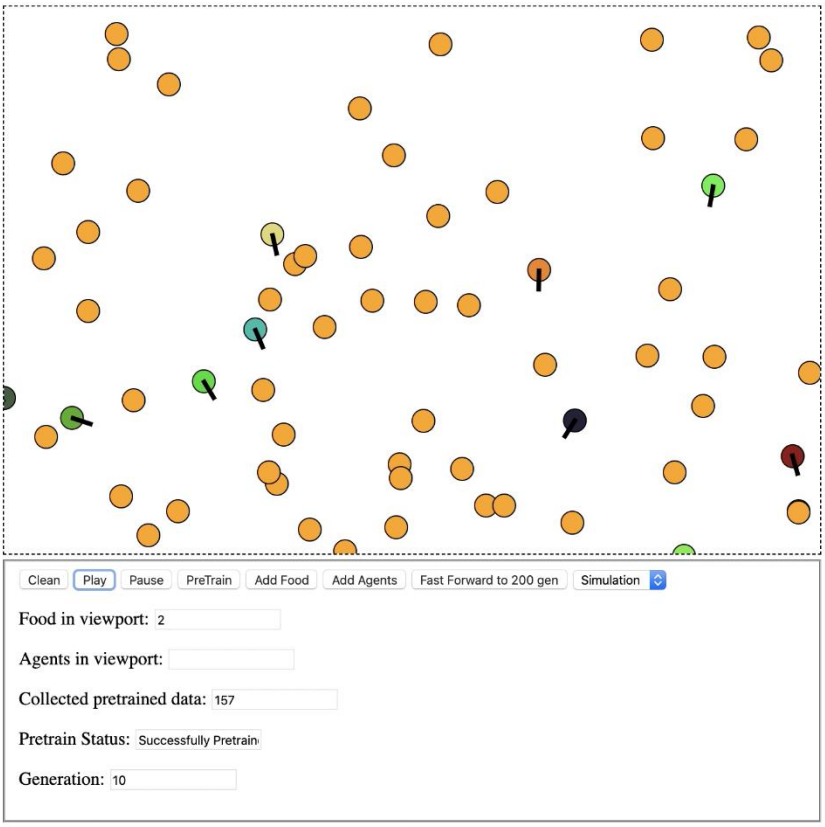


Рис. 4.1. Інтерфейс створеного додатку

Сам проект являє собою web-додаток створений за допомогою HTML та TypeScript. TypeScript - це сучасна мова програмування створена компанією Microsoft, що являє собою розширену версію мови JavaScript зі статичною типізацією.

### 4.1. Опис інтерфейсу створеної програми

Інтерфейс програми складається з декількох головних блоків:

1) “Ігрове поле” - розміщується у верхній частині інтерфейсу користувача. Поле наповнене агентами, що підпорядковуються нейронними мережами, а також їжу за якою будуть вчитися здобувати агенти.

2) Кнопки для контролю моделювання і перемикання між різними режимами роботи.

3) Також важлива іІнформація для відладки та статистика буде розміщена у нижній частині інтерфейсу.

Програма містить 3 режими роботи: “*Дотренування*”, “*Симуляція*”, “Пришвидшена симуляція”.

Режим “*Дотренування*” потрібний для того, щоб вказати вагові коефіцієнти для нейронної мережі, а також зазначити що буде використана для генерування першого покоління агентів. Також програма надає доступ навчити на прикладі того, як управляє агентом користувач навчити її базовим реакціям на навколишнє середовище. Управління агентом можливе за допомогою класичних для комп’ютерних ігор клавіш W, A, S, D - які дозволяють керувати прискоренням і кутом нахилу агента.

Для програми доступна можливість завантажити вже натренований *serialized* в форматі JSON екземпляр нейронної мережі і сформувати на її основі перших агентів.

Доступний ще один режим “*Симуляція*”, який для початку формує стартове покоління агентів і запускає процес моделювання, який можна побачити на ігровому полі у реальному часі.

Сам процес навчання або розвитку буде тривати певний час, тому додали ще третій режим роботи “*Пришвидшена симуляція*”, який надає можливість виконати чимало симуляцій без відображення на ігровому полі, що в свою чергу дозволяє швидше аналізувати чи стають ефективнішими агенти з покоління у покоління.

“*Ігрове поле*” реалізовано за допомогою svg елементів, які керуються NPM пакетом Two.JS, який є обгорткою високого рівня для маніпуляції векторними елементами та їх анімаціями.

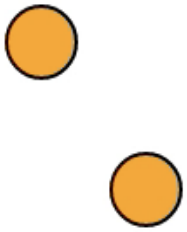

Рис.. 4.2. Вигляд їжі у середовищі

Їжа, що розподілена випадковим чином по середовищу виглядає як коло помаранчевого кольору.

Рис. 4.3. Вигляд агента, що моделюється

Агенти на ігровому полі виглядають як зображено на рис. 4.3 - вони складаються з основного “тіла”, а також вектору напрямку руху який позначений чорною рискою. Колір агента відповідає відповідає його геному, це зроблено для ідентифікації мутацій - тобто якщо користувач програми бачить, що агенти різного кольору, то це буде означати, що конфігурація нейронних мереж у цих агентів різна (і навпаки).

## 4.2. Архітектура нейронної мережі та опис генетичного алгоритму

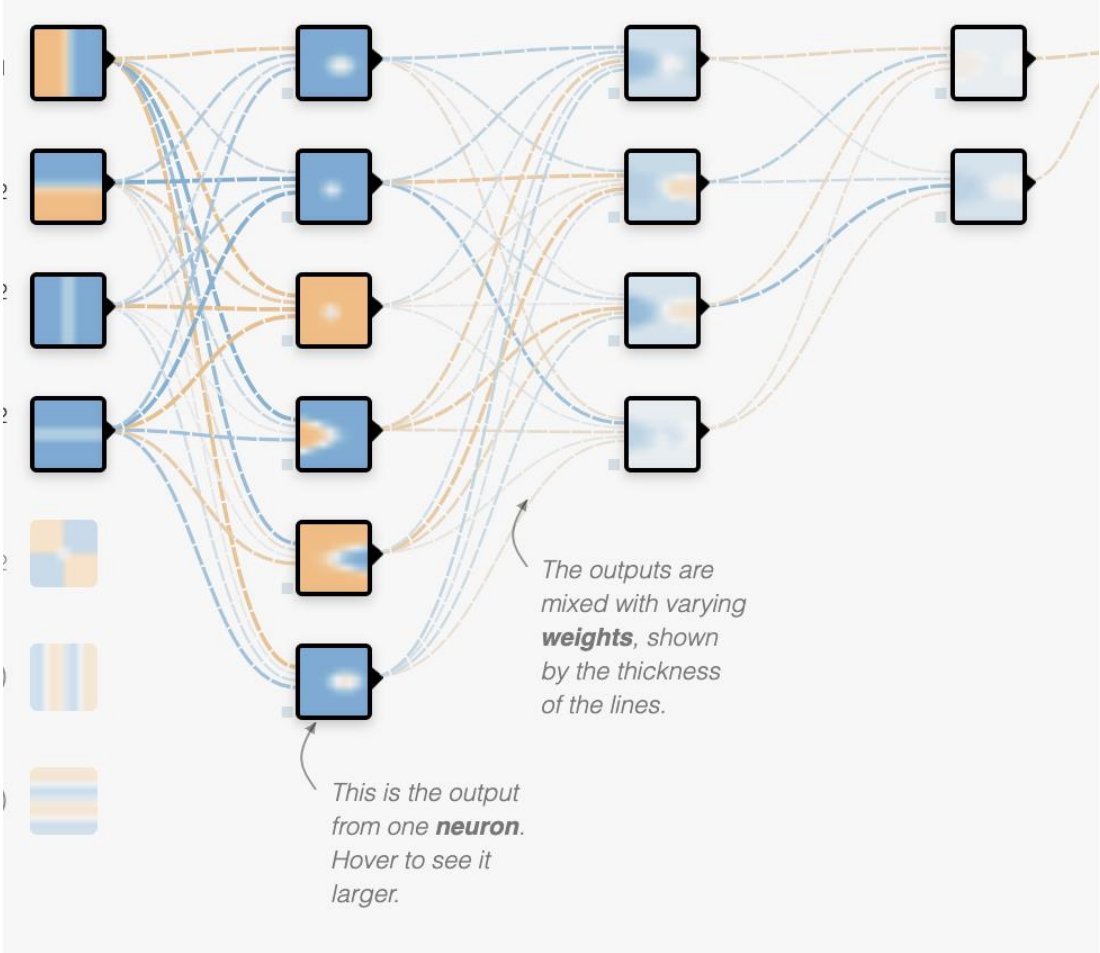


Рис. 4.4. Схема нейронів і зв’язків

Архітектура нейронної мережі (рис. 4.4), що моделювалася - це звичайний багатошаровий перецептрон з:

- 4 вхідними нейронами
- 6 нейронами в першому прихованому шарі
- 4 нейронами в другому прихованому шарі
- 2 вихідними нейронами

Для реалізації нейронних мереж була використана бібліотека під назвою Brain.JS. Вона реалізує цілий ряд алгоритмів навчання та дозволяє

експериментувати з різними типами  нейронних мереж, а також серіалізовувати і десеріалізовувати натреновані моделі.

```
// provide optional config object (or undefined). Defaults shown.
const config = {
  binaryThresh: 0.5,
  hiddenLayers: [3], // array of ints for the sizes of the hidden layers in the network
  activation: 'sigmoid', // supported activation types: ['sigmoid', 'relu', 'leaky-relu', 'tanh'],
  leakyReluAlpha: 0.01, // supported for activation type 'leaky-relu'
}

// create a simple feed forward neural network with backpropagation
const net = new brain.NeuralNetwork(config)

net.train([
  { input: [0, 0], output: [0] },
  { input: [0, 1], output: [1] },
  { input: [1, 0], output: [1] },
  { input: [1, 1], output: [0] },
])

const output = net.run([1, 0]) // [0.987]
```

Рис. 4.5. Приклад опису найпростішої мережі

У якості активаційної функції була використана лінійна активаційна властивість ReLU

(4.1) $$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geqslant 0 \end{cases}$$

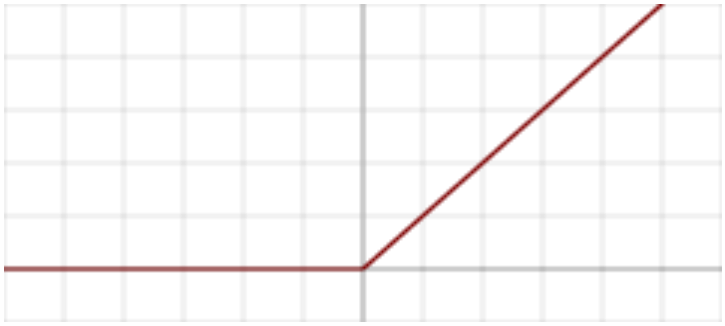

Рис. 4.5. Приклад опису найпростішої мережі

Дана властивість має графік як зображено на рис.4.6., вона є лінійною. Також можна використовувати і інші типи нейронних мереж, для цього необхідно змінити конфігурацію у файлі src/game - доступними варіантами функцій є:

- Sigmoid, класична сигмоїдальна активаційна властивість
- ReLU, властивість за замовчуванням
- Leaky-ReLU
- Tanh

Нейронна мережа управляє поведінкою агента в середовищі сприймаючи 4 вхідних сигнали і видаючи 2 сигнали на вихід. На вхід подаються наступні змінні: поточна швидкість, поточний кут повороту відносно горизонтальної осі, та відсотки кількості їжі і інших агентів, що має у полі зору сам агент.

Сигнали, що отримуються на вихід, це $\Delta \boldsymbol{v}$ та $\Delta \boldsymbol{\gamma}$ - сигнал зміни швидкості та кута повороту агента.

Варто відмітити, основним моментом, при тренуванні, або навіть подачі будь-яких вході до нейронної мережі є нормалізація вхідного вектору даних. Це необхідно для того, щоб дати можливість нейронній мережі оперувати в контексті обмежень активаційних функцій нейронів. Усі входи та виходи є числами, що відповідають проміжку [0; 1] і для того щоб конвертувати від'ємні та інші значення за межами діапазону було використано метод *min-max normalization* формула *(4.2)*.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad (4.2)$$

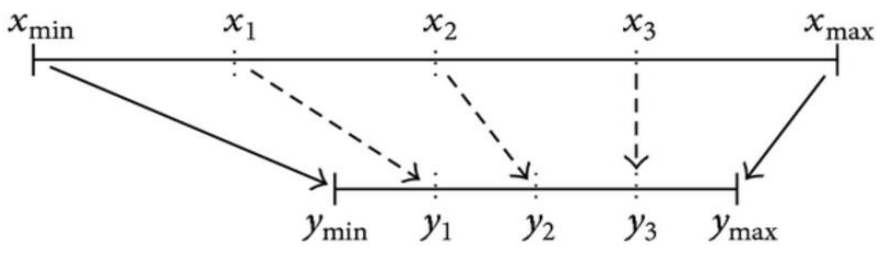


Рис. 4.6. Ілюстрація алгоритму нормалізації

Нейронна мережа тренується в режимі “Претренування” досить поверхнево, але таким чином, щоб досягти будь-якої працюючої моделі мережі для Управління агентами. Закладено поріг 30000 ітерацій для зупинки тренування, якщо метод тренування не дає належних результатів. Після початкового тренування нейронна мережа використовується для створення першої популяції агентів, де вона випадково може включати в себе мутації.

### 4.3. Архітектура агентів

Агент у розробленій програмні це модель штучного мікроорганізма, основною еволюційною задачею якого є навчитися розпізнавати їжу та “з’їдати” їжу, що розподілена випадковим чином по усій поверхні середовища.

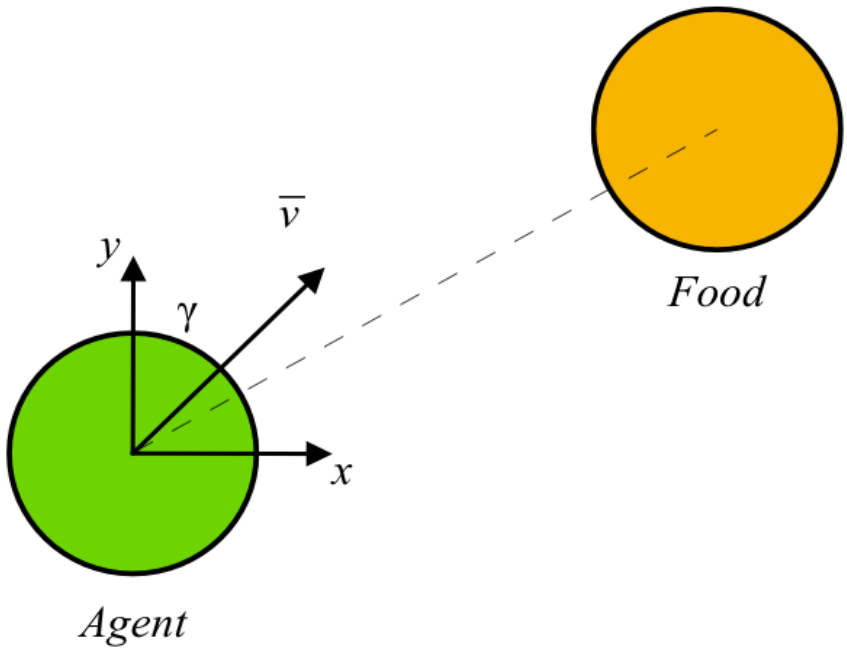


Рис. 4.6. Ілюстрація моделі агента

Агент має наступні властивості:

1) *V* - швидкість, що може бути від 0 до 5
2) *(X,Y)* - вектор позиції на полі

*3)* *S* - вектор зору (співпадає з вектором швидкості)

4) *network* - нейронна мережа, що виступає генотипом агента

5) eaten - кількість з'їденої їжі

Під час розвитку нейронна мережа отримуватиме в режимі реального часу сигнали від основних параметрів агента - швидкості, куту нахилу відносно горизонтальної осі, і інформації про те, що знаходиться попереду агента.

Коли агент стикається з їжею і "торкається" її, то елемент їжі зникає з ігрового поля, а параметр eaten в екземплярі Agent інкрементується. Таким чином зберігається інформація про те, скільки їжі, кожен агент зумів знайти за час моделювання одного покоління.

Виявлення колізій агентів і їжі приведене до простої задачі рахування відстані між центром кола агента і центром елемента їжі за допомогою наступної формули:

$$distance(O_{food}, O_{agent}) = \sqrt{(x_f - x_a)^2 + (y_f - y_a)^2} \quad (4.3)$$

Після цього разрахунку програма перевіряє чи менше відстань між об'єктами ніж сума радіусів кіл агента і їжі: якщо менше або дорівнює, то значить агент і їжа перетинаються.

Параметр eaten є fitness показником для кожного агента. Після того як вся їжа на ігровому полі закінчується, то програма зупиняє покоління і обирає 2-х найбільш успішних агентів, тобто агентів, що знайшли найбільше їжі. І на основі них генерує нове покоління за допомогою операції поєднання.

### 4.4. Еволюція агентів

Операція поєднання відбувається над двома агентами через випадкове поєднання вагових коефіцієнтів обох нейронних мереж. При чому це поєднання є медіанним і нові вагові коефіцієнти результуючої мережі будуть вираховуватися за формулою:

$$median(a,b) = \frac{max(a,b) - min(a,b)}{2} + min(a,b) \quad (4.4)$$

Результуюча мережа є усередненою версією двох найкращих агентів з покоління. Так як це не гарантує, що така мережа буде ефективною, то алгоритм додає у наступне покоління батьків, щоб переконатися у тому, що показники ефективності отриманої мережі не деградують.

Мутації у свою чергу побудовані на алгоритмі з використанням випадковості для прийняття рішення щодо того, чи необхідно робити мутацію конкретного значення в генотипі агента.

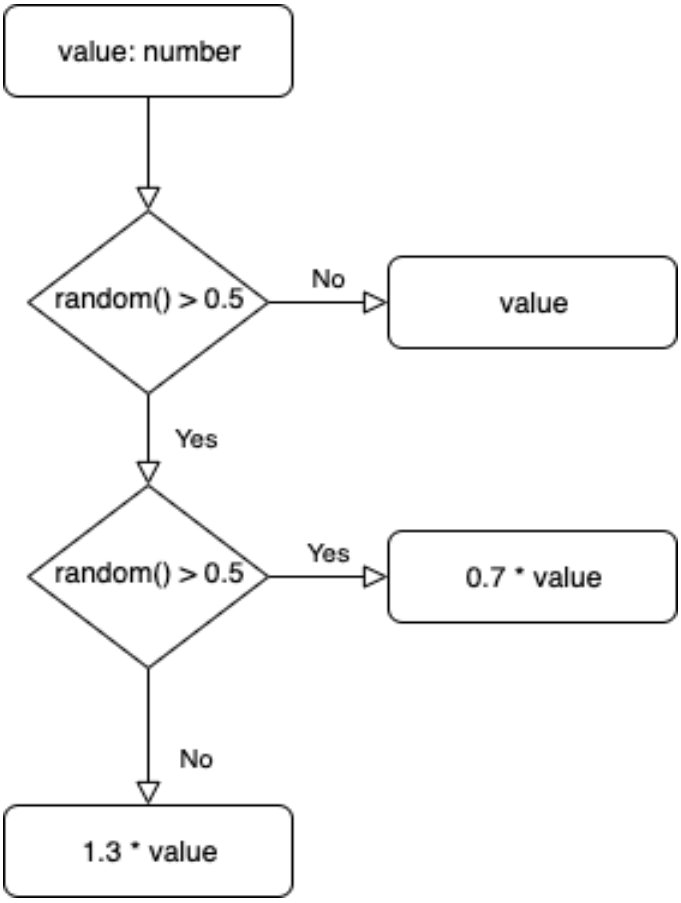


Рис. 4.7. Блок-схема алгоритму мутації

Мутації відбуваються через повний перебір вагових коефіцієнтів мережі та випадкового застосування функції мутації на них. Таким чином кожен агент

нового покоління генерується на базі отриманої після поєднання мережі і її випадкових мутацій, унікальних для кожного агента.

Для покращення візуальної ідентифікації унікальних генотипів у модельованому середовищі для кожного генотипу вираховується хеш за допомогою односторонньої хеш-функції на основі якого задається колір агента. Отже якщо агенти мають однаковий колір, то вони мають однаковий геном, і, навпаки, якщо колір хоч трохи відрізняється - геноми можуть бути абсолютно різні. Це зв’язано з особливістю хеш-функцій: якщо змінити хоч один біт вхідної інформації, то її хеш буде повністю відрізнятися від хешу початкового набору вхідних даних.

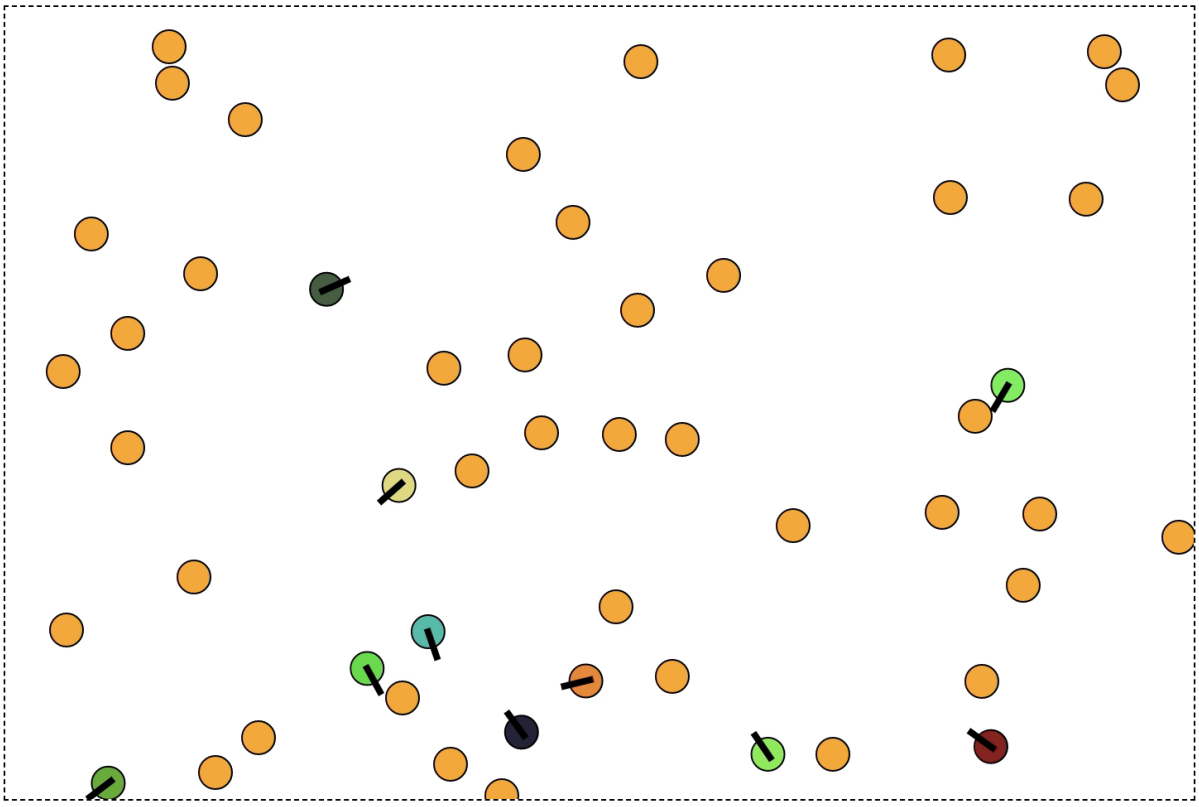

Рис. 4.8. Агенти з різними генотипами

На рис. 4.8 показано приклад популяції агентів з різними геномами - вони мають кардинально різні кольори (крім двох зелених).

### 4.5. Структура розробленої програми

Програма була розроблена за допомогою сучасної мови програмування TypeScript, що має підтримку парадигм об’єктно-орієнтовного, функціонального і імперативного програмування. Особливістю мови є те, що ця мова дуже схожа на останні версії мови JavaScript, але при цьому має статичну типізацію, що дозволяє валідувати написаний код ще під час його написання. Також TypeScript дає можливість налаштовувати строгість компілятора, що гарно для розробки прототипів.

Структура файлів проекту виглядає наступним чином:

```
├── index.html
├── package-lock.json
├── package.json
├── src
│   ├── agent.ts
│   ├── food.ts
│   ├── game.ts
│   ├── index.ts
│   ├── movement.ts
│   ├── network.ts
│   ├── types.ts
│   └── utils.ts
└── tsconfig.json
```

Головним файлом з якого починається запуск програми є *index.ts*, це канонічна назва ініціючих файлів ще з мови JavaScript.

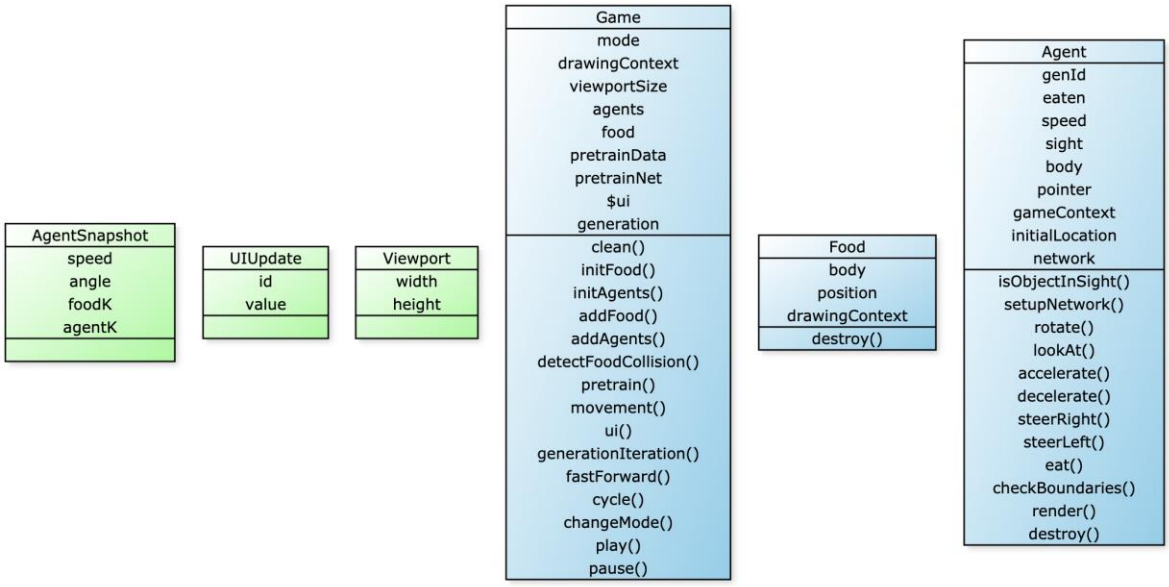


Рис. 4.9. UML діаграма основних сутностей

Увесь UI або користувацький інтерфейс було вирішено зробити за допомогою стандартного набору елементів, що представляє браузер. Для оновлення інтерфейсу використовується RxJS Observable об’єкт типу Subject під назвою $ui. В нього передаються відповідно назва UI-елементу і новий стан або значення, після чого listener з *index.js* реагує на зміни і оновлює відповідні елементи у Document Object Model сторінки. Це приклад так званої “реактивної” (від *FRP functional react programming)* реакції користувацького інтерфейсу на зміни стану програми.

Програма запускається і компілюється за допомогою спеціального інструменту під назвою *Parcel.* Цей інструмент потребує мінімум конфігурації для налаштування базового проекту. Для його встановлення в проект необхідно виконати наступну CLI команду:

**$ npm i [-g] parcel**

Для старту серверу в терміналі необхідно виконати наступну команду:

***$ npm run dev***

Після чого буде запущено спочатку збирання проекту, а потім буде піднятий сервер для віддачі актуальної версії програми на тестовому TCP порті під номером 1234. Приклад запущеного серверу для розробки знахдиться на рис. 4.10.

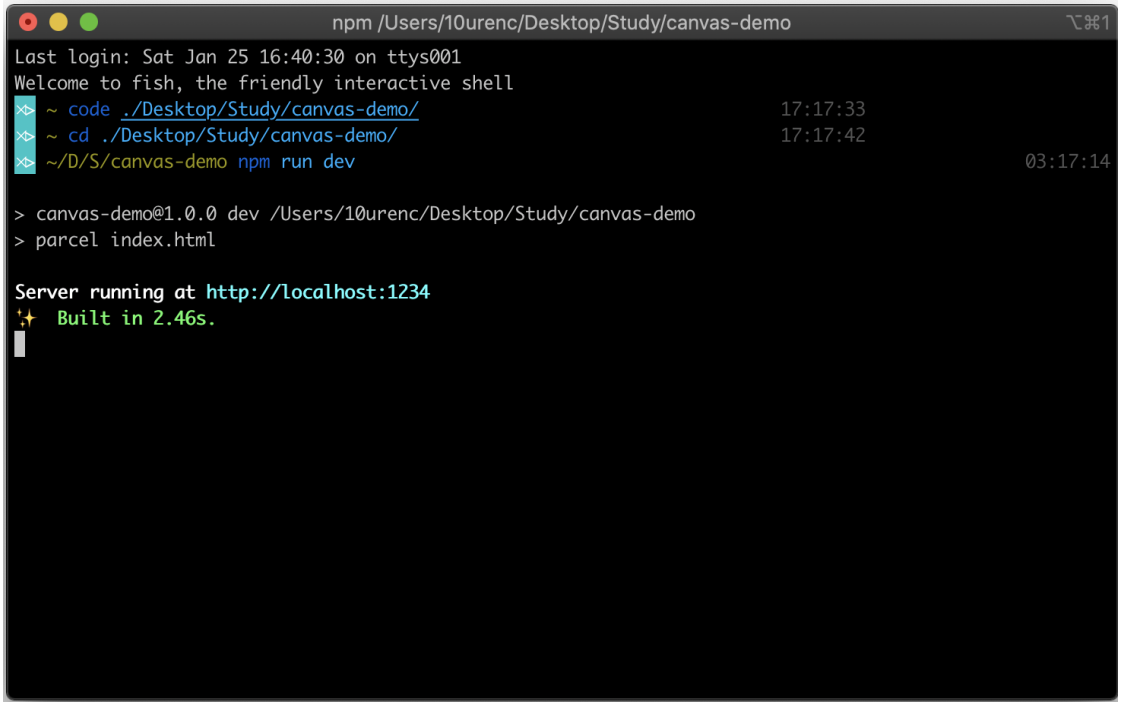


Рис. 4.10. Вікно консолі з сервером

## 4.6. Результати спостережень за моделлю

У результаті моделювання середовища з мікроорганізмами-агентами та елементами їжі була проведена низка експериментів та як результат було знайдено декілька конфігурацій, що задовольняють умови дослідження: знайти оптимальну матрицю ваг нейронної мережі агента, щоб агент якнайефективніше знаходив та “поглинав” їжу в модельованому просторі.

Одна з найкращих варіацій поколінь, що була отримана за допомогою програми повністю знищила всю їжу в середовищі розміром 720 на 480 за 8 секунд, що є достатньо високим результатом відносно перших поколінь, у яких

середній час проходження однієї ітерації був близько 2 хвилини 35 секунд - що є значно гіршим показником.

З кожним поколінням агентів видно як зберігається генетична інформація про минулі покоління. Хоча нейронна мережа на має ніякого стану у реальному часі і немає реалізації пам’яті, то генитична пам’ять що можна спостерігати являє собою вагові коефіцієнти і Bias-значення нейронної мережі.

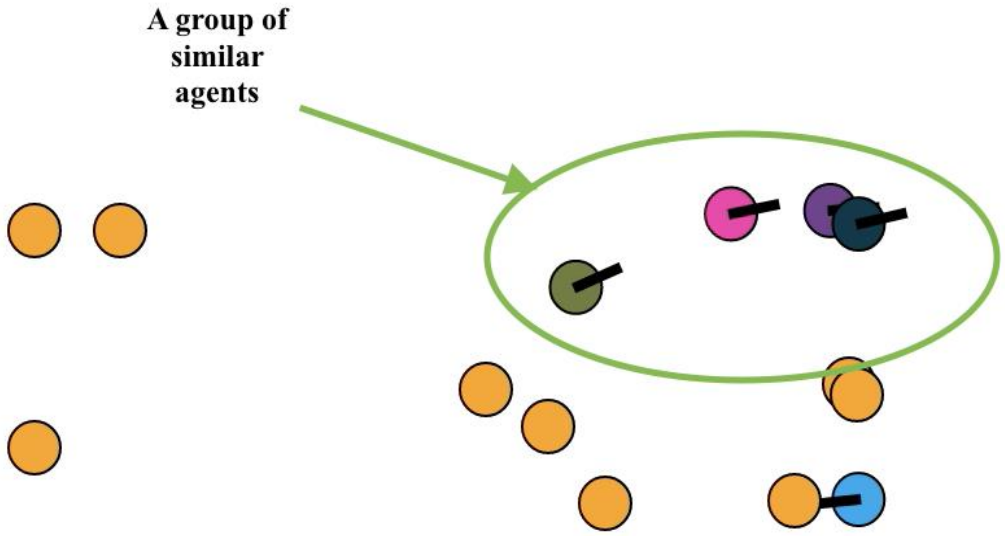


Рис. 4.11. Агенти зі схожими генотипами, що об’єднуються в одну групу

Прикладом генетичної пам’яті, що відстежується протягом багатьох симуляцій підряд є об’єднання агентів в групи. Таке трапляється досить часто майже в кожній симуляції, часто в незалежності від коригування конфігурації процесу розвитку. Цей ефект зумовлений тим, що агентам на вхід нейронної мережі подається також інформація про агентів, що знаходяться в їх полі зору і вони мають змогу базувати свої рішення ще на цій інформації.

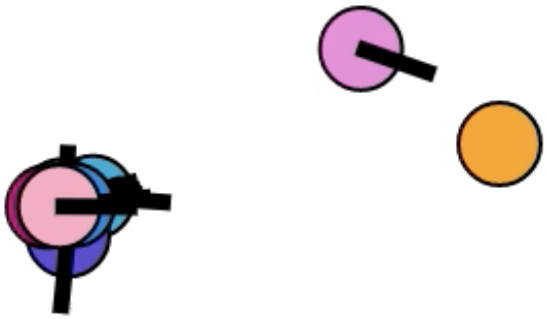

Рисунок 4.12. Приклад успішної мутації

Прикладом вдачної мутації є агент рожевого кольору, що зображений на рис. 4.11 - мутація призвела до того, що він став швидшим за інших агентів цього покоління. Схожим чином з’являться якісні мутації що стосуються загальної поведінки нових агентів.

Отже, за допомогою даного програмного продукту було розроблено модель мікроорагнізму у вигляді агента, що управляє ться штучною нейронною мережею і який цілком успішно був запрограмований орієнтуватися і взаємодіяти з модельованим середовищем. Поточні розробки можна використовувати для програмування і оптимізації різного роду нейронних мереж для вирішення великого спектру задач. Пропонується використовувати даний алгоритм для тренування і розробки штучного інтелекту, що може використовуватися, наприклад, в комп’ютерних іграх, а також різних наукових моделюваннях таким чином зменшуючи необхідний ресурс для створення комплексних моделей.

# ВИСНОВКИ

В даній роботі було досліджено існуючі методи створення штучних нейронних мереж, також була зроблена порівняльна характеристика градієнтних і евристичних алгоритмів навчання таких як нейроеволюційні алгоритми. У ході аналізу зроблені висновки щодо класу задач для яких краще підходить кожен тип тренування.

Штучні нейронні мережі є важливим розширенням поняття обчислення. Вони обіцяють створення автоматів, що виконують функції, що були раніше винятковою прерогативою людини. Машини можуть виконувати нудні, монотонні та небезпечні завдання, і з розвитком технології виникнуть нові програми.

Теорія штучних нейронних мереж розвивається стрімко, але нині вона недостатня, щоб бути опорою найбільш оптимістичних проектів. У ретроспективі видно, що теорія розвивалася швидше, ніж пророкували песимісти, але повільніше, ніж сподівалися оптимісти, – типова ситуація. Сьогоднішній вибух інтересу привернув до нейронних мереж тисячі дослідників. Резонно очікувати швидкого зростання нашого розуміння штучних нейронних мереж, що веде до більш досконалих мережевих парадигм та безлічі прикладних можливостей.

Розроблено нейроеволюційний алгоритм для створення найбільш ефективної архітектури та конфігурації вагових коефіцієнтів нейронної мережі, що забезпечує повноцінне моделювання еволюційної системи з агентами-мікроорганізмами та вирішує задачу оптимізації поведінки агентів у середовищі навчаючи їх відповідним реакціями на навколишнє середовище.

На базі даного алгоритму також був створений веб-додаток який дозволяє безпосередньо у браузері користувача відтворити моделювання і зберегти отриману модель для подальшого використання.

При дослідженні питання застосування генетичних алгоритмів на вирішення завдань оптимізації, ми розглянули достатню кількість аспектів цієї теми. По-перше, з'ясували загальну теоретичну інформацію, дізналися хто, коли й навіщо

вигадав генетичні алгоритми, що вони собою представляють, і яку мають аналогію з природним відбором у природі. Також ми дізналися про принцип роботи генетичних алгоритмів і те, за допомогою яких основних операторів вона здійснюється.

Також було з'ясовано які завдання можна вирішувати за допомогою генетичних алгоритмів. Серед таких завдань є особливий клас – оптимізаційні.

Було розглянуто шляхи вирішення оптимізаційних завдань. Крім генетичних алгоритмів використовуються також метод перебору та градієнтний спуск. Генетичні алгоритми хороші тим, що поєднують у собі два цим традиційні методи.

Ідея, що світ можна розглядати як у термінах об'єктів, так і подій, була відома ще в давнину. За словами Декарта, люди мають об'єктно-орієнтований погляд на світ. Об'єктний підхід одна із сучасних методів реалізації програмних систем. Він дозволяє застосовувати об'єктну орієнтацію на вирішення всього кола проблем, що з складними системами. Об'єктний підхід є концептуальною основою об'єктно-орієнтованого проектування, яке використовує як метод об'єктно-орієнтований аналіз, а як інструмент для реалізації об'єктно-орієнтоване програмування.

Найбільш показовою є ефективність застосування об'єктного підходу для великих програмних систем, зі складним характером взаємодії значної кількості елементів. Дослідженню цих питань і присвячено цю курсову роботу. Мета даної курсової роботи – детальне проектування та реалізація системи, що реалізує процеси створення та взаємодії групи об'єктів. Як реалізована система для реалізації була обрана штучна нейронна мережа. Вона є об'єктом, що складається з об'єктів – шарів. У свою чергу, кожен шар складається з певної кількості елементарних об'єктів – нейронів.

## СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Networking concepts and Netware [Electronic resource]. – 2010. – Access mode: https://cds.cern.ch/record/1611666/(lastaccess: 07.05.20). – Title from the screen.
2. Writing Information Security Policies [Electronic resource]. – 2001. – Access mode: https://dl.acm.org/doi/book/10.5555/515920/(lastaccess: 09.05.20). – Title from the screen.
3. Cisco LAN Switching [Electronic resource]. – 2018. – Access mode: https://www.cisco.com/c/en/us/tech/lan-switching/index.html/(lastaccess: 10.05.20). – Title from the screen.
4. Cisco campus network design guide [Electronic resource]. – 2019. – Access mode: https://www.cisco.com/c/en/us/solutions/design-zone/networking-design-guides/campus-wired-wireless.html/(lastaccess: 12.05.20). – Title from the screen.
5. Network Security for dummies [Electronic resource]. – 2011. – Access mode: https://www.dummies.com/computers/computer-networking/network-security/(lastaccess: 14.05.20). – Title from the screen.
6. Network Security Bible [Electronic resource]. – 2011. – Access mode: https://www.oreilly.com/library/view/network-security-bible/9780470502495/(lastaccess: 18.05.20). – Title from the screen.
7. Network topology [Electronic resource]. – 2020. – Access mode: https://simple.wikipedia.org/wiki/Network_topology#Daisy_chain/(lastaccess: 21.05.20). – Title from the screen.
8. TCP/IP definition [Electronic resource]. – 2012. – Access mode: https://searchnetworking.techtarget.com/definition/TCP-IP/(lastaccess: 26.05.20). – Title from the screen.
9. Firewall [Electronic resource]. – 2012. – Access mode: http://www.tech-faq.com/firewall.html/(lastaccess: 28.05.20). – Title from the screen.

10. Virus [Electronic resource]. – 2012. – Access mode: https://www.webopedia.com/TERM/V/virus.html/(lastaccess: 01.06.20). – Title from the screen.
11. Hardware [Electronic resource]. – 2010. – Access mode: http://www.fcit.usf.edu/network/chap3/chap3.htm/(lastaccess: 04.06.20). – Title from the screen.