Хмельницький національний університет
Факультет програмування та комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерних наук та інформаційних технологій

# ДИПЛОМНА РОБОТА
# МАГІСТРА

на тему *Застосування штучного інтелекту для класифікації продуктів харчування*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Виконав: студент *2* курсу, група *КНм-19-1* _____ *А.О. Шевченко*
Підпис Ініціали, прізвище

Керівник: *к.ф-м.н., доцент кафедри КНІТ* _____ *В.Ц. Міхалевський*
Підпис Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КНІТ* _____ *Р.О. Багрій*
Підпис Ініціали, прізвище

До захисту допускаю:
*Зав. кафедри КНІТ, д.т.н., професор* _____ *О.В. Бармак*
Підпис Ініціали, прізвище
7 12 2020 р.

Хмельницький 2020

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Факультет *програмування та комп'ютерних і телекомунікаційних систем*
Кафедра *комп'ютерних наук та інформаційних технологій*
Освітній ступінь *магістр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук та інформаційних технологій*

_____
(підпис)
*д.т.н., професор О.В. Бармак*
«*7*» *09* *2020* року

**ЗАВДАННЯ**
**НА ДИПЛОМНУ РОБОТУ МАГІСТРА**

1. Тема дипломної роботи магістра: *«Застосування штучного інтелекту для класифікації продуктів харчування»*

2. Завдання видано студентці *Шевченко Артем Олександрович*
(прізвище, ім'я, по батькові)

3. Керівник роботи *к.ф-м.н., доцент Міхалевський Віталій Цезарійович*
(прізвище, ім'я, по батькові)

4. Затверджені наказом університету від «*9*» *09* *2020* р. № *22*

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

*Мета роботи* – розробка методу класифікації продуктів харчування із використанням підходів штучного інтелекту. Об'єктом дослідження є порівняння методів класифікації продуктів харчування з використання підходів машинного навчання. Предметом дослідження є набори даних, структуровані та класифіковані для навчанням машинної системи класифікації і перевірки її ефективності.

# Реферат

Дипломна робота магістра присвячена використанню штучного інтелекту для класифікації продуктів харчування.

**Актуальність теми.** В магістерській роботі було проведено дослідження з пошуку методів та аналізу підходів класифікації зображень з використання штучного інтелекту. Необхідність роботи полягає в тому, що найбільш важливим напрямком, який активно розбудовується, штучного інтелекту є штучні нейронні мережі, що працюють за принципом біологічних нейронних мереж, і являють собою систему взаємодіючих штучних нейронів. Серед основних областей застосування штучної нейронної мережі можна виділити: розпізнавання об'єктів, обробка звукової й текстової інформації, прогнозування, прийняття розв'язків, оптимізація, аналіз даних.

**Метою дослідження** є розробка методу класифікації зображень з використанням штучного інтелекту.

Для досягнення зазначеної мети поставлені наступні **задачі**:

– показати, що використання сучасних методів машинного навчання дозволяє на прийняйному рівні проводиди ефективну класифікацію;

– провести дослідження факторів впливу на кінцевий результат розпізнання;

– провести порівняльні дослідження ефективності застосування відомих методів класифікації.

При цьому передбачається розв'язок таких **підзадач:**

– попередня обробки даних у вигляді зображень;

– побудова кінцівих даних після попередньої обробки;

– розробка структури нейронної мережі із застосуванням в предметній області;

– вибір моделей моделей машинного навчання, які ефективно працюють з даними предметної області;

– тестування методів та підходів з визначенням якісних та кількістних характеристик;

– програмна реалізація системи класифікації продуктів харчування.

**Об'єктом дослідження** є область сервісних послуг харчування та порівняння методів класифікації продуктів харчування з використання підходів машинного навчання.

**Предметом дослідження** є набори даних, отримані із предметної області послуг харчування та дослідження, структуровані та класифіковані для навчання машинної системи класифікації.

Автоматичне розпізнавання продуктів харчування на основі зображень є особливо складним завданням. Традиційні підходи до аналізу зображень досягли низької точності класифікації в минулому, тоді як підходи глибокого навчання дозволили ідентифікувати типи продуктів харчування. Вміст харчових страв, є деформованими предметами, як правило, включаючи складну семантику, що робить завдання визначення їх структури дуже складним. Методи глибокого навчання вже показали дуже перспективні результати в таких викликах, тому це дослідження присвячено презентації деяких популярних підходів і методів, що застосовуються в розпізнавання продуктів харчування на основі зображень. Основні напрямки рішень є навчання з нуля, навчання відібраних даних, дані особливої групи для поставленого завдання, які перевіряються і порівнюються, щоб виявити притаманні сильні та слабкі сторони.

**Достовірність** результатів забезпечується проведенням тестування розробленої системи. Також достовірність підтверджується системним аналізом ефективності.

Розпізнавання продуктів харчування на основі зображень є складним завданням, яке цікаве для наукових спільнот комп'ютерного зору та штучного інтелекту за останнє десятиліття. Звичайно, що блюдо з їжі може містити різноманітні інгредієнти, які можуть відрізнятися за формами і розмірами в залежності від регіональних традицій і звичок місцевих громад. Наприклад, сьогодні відомо по всьому світу, що салат з помідорами, оливами і цибулею

відноситься до середземноморської дієти. Таким чином, знаходження відмінності серед грецького або ізраїльського салату може бути важким завданням навіть для людини. Те, як інгредієнти нарізані і поміщені на тарілку, може бути головним параметром для відмінності. Однак, пропонована система дає прийнятні результати та дає змогу розпізнати основні забраження, які типові для визначеної області.

Це дослідження було зосереджено на особливо цікавій і складній проблемі розпізнавання продуктів харчування на основі зображень і спиралися на основні напрямки машинного навчання, щоб виявити сильні і слабкі сторони кожного підходу і виділити можливості, які пропонуються, та притаманні недоліки як з підходами, так і передумовами. Крім того, оскільки підходи до машинного навчання є необробленими даними, були представлені популярні великі набори даних для розпізнавання харчових продуктів і виявлено необхідність ще кращих наборів даних з більш конкретними та всесвітньо відомими категоріями продуктів харчування. У дослідженні наголошувалось на необхідності маркування харчових інгредієнтів, що дозволило б розширити можливість системи для досягнення кращих показників і розширення можливостей більш корисних реальних додатків (наприклад, в харчуванні та туризмі). В цілому, дослідження дає хороші результати в сфері розпізнавання продуктів харчування з підходами машинного навчання, і пропонує ряд майбутніх напрямків дослідження.

**Наукова новизна одержаних результатів**. В результаті проведеної роботи були отримані такі результати.

– Набули подальшого розвитку існуючі методи дослідження систем прогнозування та розпізнання продуктів харчування та інгредієнтів за їхніми зображеннями.

– Запропоновано модель глибокого навчання та модернізовано доступну платформу прогнозування змісту на основі зображень.

– Розроблено правила, засновані на контекстних факторах, що розвивають аналогічні правила поділу харчових продуктів, враховуючи

походження, звички, культуру тощо, які можуть допомогти в цільових прогнозах на основі географічних місць, особистих уподобань тощо.

**Практична значимість** дослідження полягає в тому, що розглянуті набори даних для розпізнавання та методи з їх класифікації є прикладними та практичними даними, які були взяті із повсякденного використання та є актуальними в прикладному значенні щодо практичного застосування.

З розвитком технологій і істотним збільшенням потужності обчислювальних систем, стали актуальними задачі автоматизації процесів і створення штучного інтелекту. Головною метою штучного інтелекту є розв'язок задач, що виникають на практиці. Дослідження доповнено базовим матеріалом, розділом, присвяченим відповідним наборам даних, які мають вирішальне значення у світлі прийнятих емпіричних підходів, та деякими заключними зауваженнями, які підкресляють майбутні напрямки.

В даний час практики глибокого навчання, вибір для розробки нових моделей або застосування існуючих, досить різні і іноді, думки різко розділені. Таким чином, розробка нової моделі глибокого навчання (архітектури), застосованої до нового домену (проблема - завдання) є складним і трудомістким процесом, який вимагає програмування та математичних навичок за межами середнього розуміння, поряд з розумінням data-driven (емпіричного) модельного навчання. З іншого боку, передача навчання з точним налаштуванням повинна бути прийнята, коли потрібне негайне рішення. Такий підхід менш вимогливий до технічних навичок і теоретичних знань, які могли б бути в основному обмежені розробкою простого методу класифікації. Крім того, сьогодні існує ряд онлайн-платформ, які дозволяють швидко розвивати нові моделі на основі власних технологій, які в основному виступають в якості чорних скриньок для розробників. Ці платформи забезпечують альтернативу і, як правило, системи машинного навчання попередньо навчені з великим наборам даних, що охоплюють широкий спектр різних класів, готових забезпечити вирішення найрізноманітніших проблем, за допомогою інтерфейсів прикладного програмування (API).

У роботі були розглянуті теоретичні та практичні основи нейронних мереж. У ході багаторазових спроб навчання штучної нейронної мережі, для реалізації даної задачі було вирішено використовувати поширені бібліотеки, що дозволяє максимально швидко та просто створювати нейронні мережі різних типів і архітектур.

Нейронні мережі займають лідируючі місця в області штучного інтелекту. Завдяки ним, багато задач можуть бути вирішені без конструювання алгоритмів. Але для того, щоб нейронні мережі могли функціонувати хоча б приблизно як біологічні, їм треба буде пройти дуже довгий процес еволюції.

**Апробація дипломної роботи.** Основні положення і результати роботи опубліковані в збірнику наукових праць – Шевченко А. О. Застосування штучного інтелекту для класифікації продуктів харчування / А.О. Шевченко, В.Ц. Міхалевський // Збірник наукових праць за матеріалами XII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук - 2020». - Хмельницький, 2020. – С.357-358.

Основні наукові та практичні результати доповідалися на конференції: доповідь на тему «Застосування штучного інтелекту для класифікації продуктів харчування» на XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» (Хмельницький, 9-10 листопада 2020 року, Хмельницький національний університет).

**Структура та обсяг роботи**. Дипломна робота магістра складається з завдання, реферату, змісту, вступу, 4 розділів, висновків, переліку посилань із 44 найменувань та додатків. Загальний обсяг дипломної роботи магістра становить 90 сторінок, з них 82 сторінок основного тексту та 8 сторінок додатків, в роботі наведено 28 рисунків та 3 таблиці.

**Ключові слова:** нейронна мережа, згорткова нейронна мережа, класифікація, глибоке навчання.

# Зміст

Перелік скорочень ..... 4
Вступ ..... 5
Розділ 1
Аналіз сучасного стану розпізнавання продуктів харчування ..... 10
1.1 Описа предметної області ..... 10
1.2 Структура дослідженнь ..... 13
1.3 Популярні фреймворки ..... 13
1.4 Постановка задачі ..... 17
Висновки до розділу 1 ..... 18
Розділ 2
Формування та структура мережі ..... 20
2.1 Набори даних зображень ..... 20
2.2 Розробки архітектури мережі ..... 23
2.3 Трансферне навчання та точне налаштування ..... 26
2.4 Платформи глибокого навчання ..... 27
2.5 Порівняння платформ глибокого навчання ..... 29
Висновки до розділу 2 ..... 32
Розділ 3
Розробка мережі розпізнавання ..... 34
3.1 Введення в штучні нейронні мережі ..... 34
3.2 Проблеми, розв'язувані в рамках штучної нейронної мережі ..... 34
3.3 Структура штучного нейрона ..... 36
3.4 Архітектура нейронної мережі ..... 37
3.5 Навчання нейронних мереж ..... 41
3.6 Алгоритм зворотного поширення помилки ..... 44
3.7 Деякі проблеми штучної нейронної мережі ..... 47
3.8 Створення нейронних мереж ..... 48
3.9 Програмна реалізація нейронної мережі ..... 50

Висновки до розділу 3 ..... 52
Розділ 4
Дослідження ефективности застосування мережі ..... 53
4.1 Формування нейронної мережі ..... 53
4.2 Завантаження та попередня обробка набору даних ..... 55
4.3 Вектор двійкових функцій ..... 62
4.4 Навчання системи ..... 66
4.5 Оцінка моделі ..... 68
Висновки до розділу 4 ..... 75
Загальні висновки ..... 77
Перелік посилань ..... 79
Додатки

## Перелік скорочень

| Скорочення, термін, позначення | Пояснення |
|---|---|
| АІС | Автоматизована інформаційна система |
| БД | База даних |
| ДРМ | Дипломна робота магістра |
| ЗНМ | Згорткова нейронна мережа |
| ШІ | Штучний інтелект |

# Вступ

З розвитком технологій і істотним збільшенням потужності обчислювальних систем, стали актуальними задачі автоматизації процесів і створення штучного інтелекту. Головною метою штучного інтелекту є розв'язок задач, що виникають на практиці.

Одним з напрямків, що активно розбудовуються, штучного інтелекту є штучні нейронні мережі, що працюють за принципом біологічних нейронних мереж, що являють собою систему взаємодіючих штучних нейронів. Серед основних областей застосування штучної нейронної мережі можна виділити: розпізнавання об'єктів, обробка звукової й текстової інформації, прогнозування, прийняття розв'язків, оптимізація, аналіз даних.

У даній роботі розглянута програмна реалізація згорткової нейронної мережі, задачею якої є розпізнавання певних об'єктів.

**Актуальність теми.** В магістерській роботі було проведено дослідження з пошуку методів та аналізу підходів класифікації зображень з використання штучного інтелекту. Необхідність роботи полягає в тому, що найбільш важливим напрямком, який активно розбудовується, штучного інтелекту є штучні нейронні мережі, що працюють за принципом біологічних нейронних мереж, і являють собою систему взаємодіючих штучних нейронів. Серед основних областей застосування штучної нейронної мережі можна виділити: розпізнавання об'єктів, обробка звукової й текстової інформації, прогнозування, прийняття розв'язків, оптимізація, аналіз даних.

**Метою дослідження** є розробка методу класифікації зображень з використанням штучного інтелекту.

Для досягнення зазначеної мети поставлені наступні **задачі**:

– показати, що використання сучаснихметодів машинного навчання дозволяє на прийняйному рівні проводиди ефективну класифікацію;

– провести дослідження факторів впливу на кінцевий результат розпізнання;

– провести порівняльні дослідження ефективності застосування відомих методів класифікації.

При цьому передбачається розв'язок таких **підзадач:**

– попередня обробки даних у виглядізображень;

– побудова кінцівих даних після попередньої обробки;

– розробка структури нейронної мережі із застсуванням в предметній області;

– вибір моделей моделей машинного навчання які ефективно працюють з даними предметної області;

– тестування методів та підходів з визначенням якісних та кількістних характеристик;

– програмна реалізація системи класифікації продуктів харчування.

**Об'єктом дослідження** є область сервісних послуг харчування та порівняння методів класифікації продуктів харчування з використання підходів машинного навчання.

**Предметом дослідження** є набори даних, отримані із предметної області послуг харчування та дослідження, структуровані та класифіковані для навчанням машинної системи класифікації.

Автоматичне розпізнавання продуктів харчування на основі зображень є особливо складним завданням. Традиційні підходи до аналізу зображень досягли низької точності класифікації в минулому, тоді як підходи глибокого навчання дозволили ідентифікувати типи продуктів харчування. Вміст харчових страв, є деформованими предметами, як правило, включаючи складну семантику, що робить завдання визначення їх структури дуже складним. Методи глибокого навчання вже показали дуже перспективні результати в таких викликах, тому цей дослідження присвячено презентації деяких популярних підходів і методів, що застосовуються в розпізнавання продуктів харчування на основі зображень.

Основні напрямки рішень, є навчання з нуля, навчання відібраних даних, дані особливої групи для поставленого завдання, які перевіряються і порівнюються, щоб виявити притаманні сильні та слабкі сторони.

**Достовірність** результатів забезпечується проведенням тестування розробленої системи. Також достовірність підтверджується системним аналізом ефективності.

Розпізнавання продуктів харчування на основі зображень є складним завданням, яке цікаве для наукових спільнот комп'ютерного зору та штучного інтелекту за останнє десятиліття. Звичайно, що блюдо з їжі може містити різноманітні інгредієнти, які можуть відрізнятися за формами і розмірами в залежності від регіональних традицій і звичок місцевих громад. Наприклад, сьогодні відомо по всьому світу, що салат з помідорами, оливами і цибулею відноситься до середземноморської дієти. Таким чином, знаходження відмінності серед грецького або ізраїльського салату може бути важким завданням навіть для людини. Те, як інгредієнти нарізані і поміщені на тарілку, може бути головним параметром для відмінності. Однак пропонована система дає прийнятні результати та дає змогу розпізнати основні забраження, які типові для визначеної області.

Це дослідження було зосереджено на особливо цікавій і складній проблемі розпізнавання продуктів харчування на основі зображень і спиралися на основні напрямки машинного навчання, щоб виявити сильні і слабкі сторони кожного підходу і виділити можливості, які пропонуються, та притаманні недоліки як з підходами, так і передумовами. Крім того, оскільки підходи до машинного навчання є необробленими даними, були представлені популярні великі набори даних для розпізнавання харчових продуктів і виявлено необхідність ще кращих наборів даних з більш конкретними та всесвітньо відомими категоріями продуктів харчування. У дослідженні наголошувалось на необхідності маркування харчових інгредієнтів, що дозволило б розширити можливість системи для досягнення кращих показників і розширення можливостей більш корисних реальних додатків

(наприклад, в харчуванні та туризмі). В цілому, дослідження дає хороші результати в сфері розпізнавання продуктів харчування з підходами машинного навчання, і пропонує ряд майбутніх напрямків дослідження.

**Наукова новизна одержаних результатів**. В результаті проведеної роботи були отримані такі результати.

– Набули подальшого розвитку існуючі методи дослідження систем прогнозування та розпізнання продуктів харчування та інгредієнтів за їхніми зображеннями.

– Запропоновано модель глибокого навчання та модернізовано доступну платформу прогнозування змісту на основі зображень.

– Розроблено правила, засновані на контекстних факторах, що розвивають аналогічні правила поділу харчових продуктів, враховуючи походження, звички, культуру тощо, які можуть допомогти в цільових прогнозах на основі географічних місць, особистих уподобань тощо.

**Практична значимість** дослідження полягає в тому, що розглянуті набори даних для розпізнавання та методи з їх класифікації є прикладними та практичними даними, які були взяті із повсякденного використання та є актуальними в прикладному значенні щодо практичного застосування.

З розвитком технологій і істотним збільшенням потужності обчислювальних систем, стали актуальними задачі автоматизації процесів і створення штучного інтелекту. Головною метою штучного інтелекту є розв'язок задач, що виникають на практиці. Дослідження доповнено базовим матеріалом, розділом, присвяченим відповідним наборам даних, які мають вирішальне значення у світлі прийнятих емпіричних підходів, та деякими заключними зауваженнями, які підкресляють майбутні напрямки.

В даний час практики глибокого навчання вибір для розробки нових моделей або застосування існуючих, досить різні і іноді, думки різко розділені. Таким чином, розробка нової моделі глибокого навчання (архітектури), застосованої до нового домену (проблема - завдання) є складним і трудомістким

процесом, який вимагає програмування та математичних навичок за межами середнього розуміння, поряд з розумінням data-driven (емпіричного) модельного навчання. З іншого боку, передача навчання з точним налаштуванням повинна бути прийнята, коли потрібне негайне рішення. Такий підхід менш вимогливий до технічних навичок і теоретичних знань, які могли б бути в основному обмежені розробкою простого методу класифікації. Крім того, сьогодні існує ряд онлайн-платформ, які дозволяють швидко розвивати нові моделі на основі власних технологій, які в основному виступають в якості чорних скриньок для розробників. Ці платформи забезпечують альтернативу і, як правило, системи машинного навчання попередньо навчені з великим наборам даних, що охоплюють широкий спектр різних класів, готових забезпечити вирішення найрізноманітніших проблем, за допомогою інтерфейсів прикладного програмування (API).

У роботі були розглянуті теоретичні та практичні основи нейронних мереж. У ході багаторазових спроб навчання штучної нейронної мережі, для реалізації даної задачі було вирішено використовувати поширені бібліотеки, що дозволяє максимально швидко та просто створювати нейронні мережі різних типів і архітектур.

Нейронні мережі займають лідируючі місця в області штучного інтелекту. Завдяки ним, багато задач можуть бути вирішені без конструювання алгоритмів. Але для того, щоб нейронні мережі могли функціонувати хоча б приблизно як біологічні, їм треба буде пройти дуже довгий процес еволюції.

# Розділ 1
# Аналіз сучасного стану розпізнавання продуктів харчування

## 1.1 Опис предметної області

Поява технології глибокого навчання розширила багато наукових сфер, надаючи передові методи для кращого прогнозування та розпізнавання об'єктів на основі зображень або відео. В інформатиці, особливо в комп'ютерному зорі та штучному інтелекті, класифікація зображень є важливим завданням, з багатьма останніми досягненнями, що надходять від розпізнавання об'єктів з підходами глибокого навчання.

Їжа, як важлива частина повсякденного життя у всіх культурах, є особливим викликом у сфері класифікації зображень, завдяки своїй візуальній складності, а також семантичній складності, що випливає з варіації змішування різних інгредієнтів як практикується регіональними громадами. Було доведено, що цей виклик має багато складних аспектів [2]. Велика кількість харчових зображень, наданих соціальними мережами, присвячена сайтам обміну фотографіями, мобільним додаткам і потужним пошуковим системам [3] вважається простим способом розробки нових наборів даних для наукових цілей. Наукова спільнота припускає, що автоматичне розпізнавання (класифікація) страв не тільки допоможе людям без особливих зусиль організувати свої величезні фотодобірки і допоможе онлайн-репозиторіям фотографій зробити їх вміст більш доступним, а також допоможе оцінити і відстежувати щоденні харчові звички споживання калорій і планувати стратегії харчування навіть за межами обмеженого клінічного середовища [4].

Незважаючи на десятки додатків, алгоритмів і систем, проблема розпізнавання страв (їжі) і їх інгредієнтів не була повністю вирішена спільнотами машинного навчання і комп'ютерного зору [5]. Це пов'язано з відсутністю характерного просторового розташування харчових інгредієнтів, як правило, знайдених на зображеннях сцен або об'єктів. Наприклад, зображення відкритої

сцени, як правило, може бути розкладається на (a) наземне місце, (b) горизонт, (c) ліс, і (d) небо. Візерунки, такі як ці, не можуть бути знайдені в харчових зображеннях. Харчові інгредієнти, як і ті, що містяться в салаті, є сумішами, які часто приходять в різних формах і розмірах, в залежності від регіональних практик і культурних звичок. Слід підкреслити, що характер страв часто визначається різними кольорами, формами і текстурами різних інгредієнтів [6]. Тим не менш, види особливостей, які описують їжу в більшості випадків, легко впізнаються людьми на одному зображенні, незалежно від геометричних варіацій інгредієнтів. Отже, можна вважати, що розпізнавання продуктів харчування є специфічною і дуже складною проблемою класифікації, що вимагає розробки моделей, здатних використовувати місцеву інформацію (функції) всередині зображень, поряд зі складною семантики вищого рівня. Визначення продуктів харчування як і раніше є складною проблемою, яка привертає інтерес наукової спільноти.

Розпізнавання продуктів харчування роззвивалося з підвищенням обчислювальної продуктивності, комп'ютерного зору та машинного навчання за останнє десятиліття. Matsuda та ін., в 2012 році, досягли точності оцінки 55,8% для декількох елементів харчових зображень і 68,9% для однокомпонентних харчових зображень, використовуючи два різних методи [8]. Перший метод був заснований на моделі деформованої частини Felzenszwalb, а другий був заснований на функції злиття. У 2014 році була розроблена одна з перших робіт, яка застосовала глибоке навчання [9]. Дослідники досягли точності класифікації 72,26% з попередньо підготовленою (трансферною навчанням) моделлю, подібною до AlexNet [10]. У той же період Bossard та інші досягла 50,76% точності класифікації з використанням випадкового лісу на наборі даних Food101 [11]. Дослідники відзначили, що випадковий лісовий метод не може перевершити підходи глибокого навчання. Аналогічні висновки зробили Кагая та ін., які повідомили про точність класифікації 73,70% з використанням глибоких згортків

нейронних мереж (CNN) [12]. У наступному році глибокі CNN в поєднанні з трансфертним навчанням досягли 78,77%.

Christodoulidis та інші представили новий метод на глибоких CNN досягнення точності 84,90% на користувальницькому наборі даних [12]. У 2016 році Singla та інші прийняли архітектуру GoogLeNet [13] та із попередньо навченою моделлю досягла точності 83,60% [14]. Lui та інші, в 2016 році, розробили DeepFood і досягли аналогічних результатів, використовуючи оптимізовані методи згортки в модифікованій версії Inception [11]. Дослідники повідомили про точність 76,30% на наборі даних UEC-Food100, точність 54,70% на наборі даних UEC-Food256 і точність 77,40% на наборі даних Food-101. Hassannejad та інші набрали точність 81,45% на наборі даних UEC-Food100, 76,17% на набірі даних UEC-Food256 і 88,28% на наборі даних Food-101 [6], використовуючи архітектуру розпізнавання зображень Google під назвою Inception V3 [16].

У 2017 році Ciocca та інші представили новий метод, який поєднував методи сегментації та розпізнавання на основі глибоких CNN на новому наборі даних, що досягає 78,30% точності [15]. Mezgec та інші (2017) прийняла модифікацію популярного AlexNet і представила NutriNet, який був навчений зображенням, придбаним за допомогою веб-пошукових систем. Вони досягли показників класифікації 86,72% понад 520 класів продуктів харчування та напоїв. NutriNet використовує менше параметрів у порівнянні з оригінальною структурою AlexNet. У 2018 році Ciocca та інші представила новий набір даних, який був придбаний в результаті злиття інших наборів даних [17]. Вони протестували кілька популярних архітектур на наборі даних, включаючи залишкову мережу [18] з 50 шарами, встановивши останню як довідкову архітектуру. Як бачимо, багато різних дослідницьких груп енергійно працювали над темою розпізнавання продуктів харчування, використовуючи найрізноманітніші методи і прийоми. В цілому, найбільш успішними методами є варіації підходів глибокого навчання.

### 1.2 Структура досліджень

Цей розділ служить меті представлення можливих шляхів для того, щоб розробити техніку машинного навчання для складного завдання розпізнавання харчових продуктів. Він починається з впровадження загальних концепцій глибокого навчання для встановлення базового рівня розуміння та окреслення найпопулярніших глибоких навчальних рамок, доступних для проектування та розробки нових моделей. Далі представлені три можливі підходи до розробки, включаючи нову архітектуру, кілька адаптацій для передачі навчання та найрезентативніші платформи прогнозування контенту. Оскільки емпіричне навчання, що використовується в глибокому навчанні, є вимогливим до даних, популярні набори даних зображень для розпізнавання харчових продуктів представлені в окремому підрозділі. Порівняльне дослідження представлених рішень слідує, і завершується наданням інформації щодо розробки моделі глибокого навчання, яка має завдання вирішити проблему розпізнавання харчових продуктів, а також переліку майбутніх викликів у відповідній науковій сфері.

### 1.3 Популярні фреймворки

Глибока структура навчання - це платформа, що містить інтерфейси, бібліотеки, інструменти та додатки, що дозволяє розробникам і вченим легко будувати і розгортати складні моделі глибокого навчання, не безпосередньо залучаючись до розробки своїх архітектур і обробки математики. Крім того, фреймворки пропонують будівельні блоки для проектування, навчання та перевірки глибоких моделей нейронної мережі, через інтерфейси програмування високого рівня. Більшість з цих фреймворку покладаються на прискорені графічні рішення, такі як CuDNN, щоб забезпечити прискорене навчання. Деякі ключові особливості хорошої глибокої навчальної бази є:

– оптимізація продуктивності;

– графічний процесор;

– хороша документація та хороша підтримка громади;

– легко кодувати;

– паралелізовані процеси;

– готові до використання глибокі моделі/архітектури (додатки);

– попередньо навчені моделі;

– кілька готових до використання функції (наприклад, оптимізатори, функції активації, шари і т.д.)

– популярність;

– продовжується пвдтримка та оновлення.

На основі цих ключових особливостей виділяються шість глибоких навчальних рамок і представлені в цьому розділі. Рисунок 1.1 представляє простий ієрархічний погляд на ці рамки, підкреслюючи, як прискорюється всі вони графічного процесора. Слід підкреслити, що тільки фреймворк Keras можна розглядати як API високого рівня, розроблений над іншими бібліотеками, в основному зосереджений на забезпеченні зручного і легкого для розширення середовища (фреймворку).

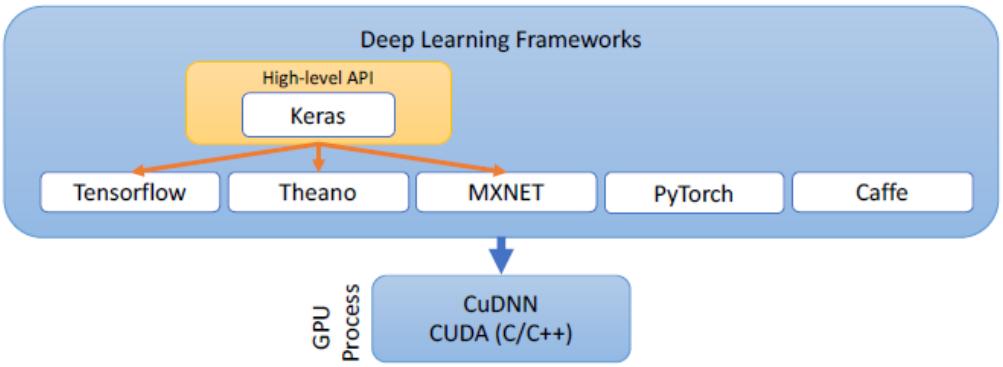


Рисунок 1.1 – Найпопулярніші фреймворки [4]

Caffe3 [5] є найстарішою і найпопулярнішою глибокою навчальною базою, розробленою Янцин Цзя в BAIR (Berkeley Artificial Intelligence Research). Caffe

підтримує пошаровий глибокий розвиток нейронної мережі. Хоча новий шар може бути розроблений мовою програмування Python, він вважається менш ефективним на відміну від шару, написаного в C++ CUDA. Caffe, крім Python API, також забезпечує інтерфейс MATLAB. Через кілька років після свого першого релізу, його хрещений батько (Yangqing Jia) і його команда в Facebook розробили рамки Caffe наступного покоління в 2018 році під назвою Caffe2. Ця версія була покращена в напрямку розробки більш зручних для користувача рамок. Крім того, він надає API, який підтримує мобільні додатки. Крім того, Caffe2 підтримує формат Open Neural Network Exchange (ONNX), що дозволяє легко інтеграцію моделей з іншими фреймворками, такими як MXNet, PyTorch і т.д. Фактично з 02 травня 2018 року Caffe2 був інтегрований в PyTorch [12].

TensorFlow4 (Abadi, та інші, 2016) - це фреймворк з відкритим вихідним кодом, розроблений і підтримується командою машинного навчання в Google. Його розробка та дизайн базуються на числових обчисленнях з використанням графіків потоку даних, що зосереджуються на масштабних розподілених моделях навчання. Він складається з більш ніж двохсот стандартних операцій, включаючи різні математичні операції, багатовимірні маніпуляції масивами, управління потоком і багато іншого, написане мовою програмування C++. Ця структура прискорюється графічним процесором, а також може працювати в процесорах і мобільних пристроях. З цієї причини його можна розглядати як основу для використання як в наукових дослідженнях, так і в розробці продукту. Його характерним недоліком є те, що він вимагає низькорівневого API і його використання здається досить складним для недосвідчених користувачів. Тим не менш, він був інтегрований в кілька високорівневих API, таких як Keras, Estimator і Eager.

Keras5 [7] - це нейронна мережева бібліотека з відкритим кодом, що працює на деяких популярних фреймворках, таких як TensorFlow і Theano. Він вважається API високого рівня (інтерфейс), написаний на Python, а не автономною рамкою машинного навчання, зосереджуючись на його розширення, його

модульності та зручному середовищі. Це вимагає мінімальних знань і технічних навичок. З 2017 року команда TensorFlow включає Keras в основну бібліотеку TensorFlow.

Таблиця 1.1 – Найбільш популярні фреймворки [5]

| **Назва** | Вихід | Програмний інтерфейс | Цитування |
|---|---|---|---|
| **Caffe** | 2014 | Python, MATLAB, C++ | 11990 |
| **TensorFlow** | 2016 | Python, C++[a], Java [a], Go [a] | 9250 |
| **Keras** | 2015 | Python | 5399 |
| **PyTorch** | 2017 | Python, ONNX, C++ | 2718 |
| **MXNET** | 2015 | Python, C++, R, Java, .. [b] | 1077 |
| **Theano** | 2007 | Python, C++ | 572 |

PyTorch6 (Adam Paszke, 2017) — це прискорена графічна структура глибокого навчання, написана в поєднанні мов програмування Python, C і CUDA. Він був розроблений дослідницькою групою Facebook зі ШІ. PyTorch молодший за TensorFlow, але він швидко виріс у популярності, завдяки своїй гнучкості та легкості розвитку, коли мова йде про складні архітектури. Крім того, це дозволяє настроювання, що TensorFlow не. Фреймворк використовується як науковими, так і промисловими спільнотами, наприклад, Uber використовує PyTorch для своїх додатків. PyTorch підтримується деякими з найбільш високотехнологічних компаній, таких як Facebook, Twitter і NVIDIA. Слід підкреслити, що PyTorch підтримує моделі ONNX, що дозволяє легко обмінюватися моделями між різними фреймворками.

Theano7 (TheanoTeam, 2016) був вперше випущений в 2007 році і був розроблений в Монреальському університеті. Це друга найстаріша значна структура глибокого навчання Python. Теано компілює математичні вирази за допомогою NumPy на Python і деяких ефективних рідних бібліотек, таких як BLAS для запуску експериментів у декількох графічних процесорах та

процесорах. У листопаді 2017 року Теано припинив активну модернізацію, але вона підтримується безперервно.

MXNet8 (Чень та ін., 2015) інкубується Apache і використовується Amazon. Це п'ята за популярністю бібліотека глибокого навчання. Однією з важливих переваг MXNet є його портативність, його легкий розвиток і його масштабованість, що дозволяє ефективні адаптації в різних комп'ютерних системах з декількома графічними процесорами. Цей ключовий момент робить MXNet дуже потужним і корисним для підприємств. Також його портативність підтримує широкий спектр розумних пристроїв, таких як мобільні пристрої (з використанням об'єднання), пристрої IoT (підтримка AWS Greengrass) і Serverless (з використанням AWS Lambda). Крім того, він хмарний і безпосередньо сумісний з різними хмарними сервісами. MXNet підтримує Keras як інтерфейс верхнього шару, що дозволяє легко і швидко прототифікувати. MXNet підтримує моделі формату ONNX.

Є кілька комплексних досліджень [6, 7], які представляють переваги та недоліки різних бібліотек, орієнтованих на машинне навчання. Слід зазначити, що більшість фреймворків мають науково задокументовану присутність.

Всі вони в основному зосереджені на глибокому навчанні нейронних мереж. Хоча Теано є найстарішим фреймворк, він не є найпопулярнішим. Найпопулярнішою основою в науковому співтоваристві є Caffe. Крім того, інтерфейс програмування Python є загальним для всіх фреймворків. Це показало широке опитування, яке розглядає практично всі рамки машинного навчання та бібліотеки, опубліковане в 2019 році Nguyen та іншими.

### 1.4 Постановка задачі

Метою дослідження є розробка методу класифікації зображень з використанням штучного інтелекту.

Для досягнення зазначеної мети поставлені наступні задачі:

– показати, що використання сучаснихметодів машинного навчання дозволяє на прийняйному рівні проводиди ефективну класифікацію ;

– провести дослідження факторів впливу на кінцевий результат розпізнання;

– провести порівняльні дослідження ефективності застосування відомих методів класифікації.

При цьому передбачається розв'язок таких підзадач:

– попередня обробки даних у виглядізображень;

– побудова кінцівих даних після попередньої обробки;

– розробка структури нейронної мережі із застсуванням в предметній області;

– вибір моделей моделей машинного навчання які ефективно працюють з даними предметної області;

– тестування методів та підходів з визначенням якісних та кількістних характеристик;

– програмна реалізація системи класифікації продуктів харчування.

**Висновки до розділу 1**

Автоматичне розпізнавання продуктів харчування на основі зображень є особливо складним завданням. Традиційні підходи до аналізу зображень досягли низької точності класифікації в минулому, тоді як підходи глибокого навчання дозволили ідентифікувати типи продуктів харчування та їх інгредієнти. Вміст харчових страв, є деформованими предметами, як правило, включаючи складну семантику, що робить завдання визначення їх структури дуже складним. Методи глибокого навчання вже показали дуже перспективні результати в таких викликах, тому цей розділ присвячений презентації деяких популярних підходів і методів, що застосовуються в розпізнаванні продуктів харчування на основі зображень. Основні напрямки рішень, а особливо для поставленого завдання перевіряються і

порівнюються навчання з нуля, навчання трансформованих даних, щоб виявити притаманні сильні та слабкі сторони. Розділ доповнено базовим матеріалом, розділом, присвяченим відповідним наборам даних, які мають вирішальне значення у світлі прийнятих емпіричних підходів, та деякими заключними зауваженнями, які підкресляють майбутні напрямки.

# Розділ 2
# Формування та структура мережі

## 2.1 Набори даних зображень

Підходи глибокого навчання, особливо методи CNN, вимагають великих наборів даних для побудови ефективної моделі класифікації. В даний час, навіть якщо є сотні наборів даних доступні, з різним вмістом, набори даних, що зосереджені на вмісті (продукти харчування, напої, інгредієнти і т.д.) дуже обмежені і досить малі. У цьому розділі представляємо деякі з найпопулярніших і великих наборів даних, що зосереджуються на визнанні харчових продуктів, що використовуються в літературі. Таблиця 2.1 підсумовує ці набори даних і оцінює їх за кількістю зображень, які вони містять, і роком їх введення. Слід зазначити, що останні п'ять років працювали над розробкою більших наборів харчових даних, оскільки розпізнавання продуктів харчування та їх інгредієнтів є дуже цікавим і складною проблемою з багатьма реальними додатками.

Таблиця 2.1 – Найбільш популярні набори даних категорії їжа [10]

| Назва | Класи | Зображення | Рік | Опис |
|---|---|---|---|---|
| UECFood100 | 100 | 14,461 | 2012 | Більшість категорій популярні в Японії |
| UECFood256 | 256 | 31,651 | 2014 | |
| Food-101 | 101 | 101,000 | 2015 | Більшість загальних категорій |
| UMPCFood-101 | 101 | 100,000 | 2015 | |
| VireoFood-172 | 172 | 110,241 | 2016 | |
| Food524DB | 524 | 247,636 | 2017 | |
| Food-101N | 101 | 310,000 | 2018 | |

Підходи глибокого навчання в основному забезпечують прогноз для кожного зразка (зображення) на основі математичних ймовірностей. У більшості випадків оцінка прогнозу базується на порівнянні результату мережі (прогнозування) і справжньої мітки класу. Точність топ-1 - це сортування результатів прогнозування, яке показує, чи є цільовий вихід (клас) однаковим з верхнім класом (той, який має найвищу розраховану ймовірність) прогнозу, зробленого для зображення моделлю. Найвища точність може бути визначена як

$$\text{top-k} = \frac{1}{N}\sum_{i=1}^{N} 1\left[a_i \in C_i^k\right] \tag{2.1}$$

де $1[\cdot] \rightarrow \{0,1\}$ позначає функцію індикатора.

Якщо умова в квадратних дужках задоволена, що означає істина відповіді на питання $q_i$, і $C_i^k$ як кандидат, встановлений з top-k з найвищих подібних відповідей, то функція повертає 1 або 0 інакше.

Набір даних UECFood10010 був вперше представлений в 2012 році і містить 100 класів харчових зображень. На кожному зображенні продукт харчування знаходиться в рамках рамки розміру. Більшість категорій продуктів харчування (класів) є популярними японськими продуктами харчування. З цієї причини більшість класів харчування можуть бути не знайомі всім. Основною причиною розробки цього набору даних було впровадження практичної системи розпізнавання харчових продуктів для використання в Японії. У 2014 році Кавано і Янай використали цей набір даних, щоб набрати показник класифікації 59,6% для точності топ-1 і 82,9% для точності топ-5 відповідно. Ця ж команда покращила свої результати класифікації, досягнувши 72,26% в топ-1 точності і 92,00% в топ-5 точності при використанні Deep CNN попередньо навчений з набором даних ILSVRC201011.

Набір даних UECFood25612 складається з 256 класів харчових зображень. Структура класів і зображення схожі на UECFood100. Знову ж таки, більшість

категорій продуктів харчування орієнтовані на японські продукти харчування. Цей набір даних використовувався для розробки мобільного додатку для розпізнавання харчових продуктів. У 2018 році Мартінел, Форесті і Мікелоні досягли показників класифікації 83,15% за топ-1 точністю 95,45% за найвищу точність на цьому наборі даних.

Набір даних Food-10113 був представлений в 2014 році з лабораторії комп'ютерного зору, ETH Zurich, Швейцарія. Загалом набір даних має тисячу зображень для кожного зі 101 класів харчування (категорій), що складається з набору даних із 101 100 зображень. Для кожного класу передбачено 250 тестових зображень та 750 навчальних зображень. Всі зображення були переглянуті вручну, а навчальні зображення містять певний шум навмисно, наприклад, інтенсивне зсув кольору зображення та неправильні мітки. Всі зображення були перенакидовані, щоб мати максимальну довжину сторони 512 пікселів. У 2018 році Мартінел, Форесті і Мікелоні повідомили про 90,27% за топ-1 і 98,71% за топ-5 за набором даних food-101.

UPMC Food-101 був розроблений в 2015 році і розглядається як великий мультимодальний набір даних, що складався з 100 000 продуктів харчування, класифікованих в 101 категорії. Цей набір даних був зібраний з Інтернету і може розглядатися як "набір даних близнюків" для Food-101, оскільки вони поділяють ті ж класи 101, і вони мають приблизно однаковий розмір.

VireoFood-17214 - набір даних із зображеннями, що зображують типи продуктів харчування та інгредієнти. Цей набір даних складається з 110 241 харчових зображень зі 172 категорій. Всі зображення вручну анотуються відповідно до 353 інгредієнтів. Творці цього набору даних використовували MultiTaskDCNN і досягли показників класифікації 82,05% для точності топ-1 і 95,88% для точності топ-5.

Food524DB15 був представлений в 2017 році і вважається одним з найбільших загальнодоступних наборів даних про їжу, з 524 класами продуктів харчування і 247 636 зображеннями. Цей набір даних був розроблений шляхом

об'ємування класів продуктів харчування з існуючих наборів даних. Набір даних був побудований шляхом об'ємування чотирьох наборів порівняльних даних: VireoFood-172, Food-101, Food50 та модифікованої версії UECFood256. Творці цього набору даних використовували популярну архітектуру RestNet-50 і набрали 81,34% за точність топ-1 і 95,45% за точність топ-5. Модифікація цього набору даних була випущена в 2018 році як Food-475, ввівши невеликі відмінності.

Food-101N16 був представлений в 2018 році в CVPR від Microsoft AI & Research. Набір даних був розроблений для усунення шуму етикетки з мінімальним наглядом людини. Цей набір даних містить 310 000 зображень рецептів продуктів харчування, класифікованих у 101 класи (категорії). Food-101and Food-101N і UPMC Food-101 набори даних, поділяють ті ж 101 класи. Крім того, зображення Food-101N набагато шумніше. Kuang-Huei, Xiaodong і Linjun забезпечили класифікацію зображень 83,95% для точності топ-1 за допомогою моделі CleanNet.

Відзначимо, що розробка нових великих наборів даних зі всесвітньо відомими категоріями продуктів харчування, є дуже складним і трудомістким процесом. В даний час кілька дослідницьких груп зосереджуються на автоматизованих процесах розробки більших наборів даних. Автоматизовані процеси використовують потужні пошукові графічні системи, краудсорсинг і контент соціальних мереж для виконання завдання зі збору даних. Наприклад, Mezgec та інші (2017) використовувала пошукову систему зображень для розробки набору харчових даних і результат був досить цікавим.

### 2.2 Розробки архітектури мережі

У цьому розділі представлена модель CNN (названа PureFoodNet), яка була розроблена для цілей проекту в Греції, яка спрямований на регіональне визнання грецьких продуктів харчування для мобільних додатків туризму. Ця модель заснована на будівельних блоках і принципах (проста однорідна топологія)

архітектури VGG. Поширеною практикою для розробки нових архітектур (моделей) CNN є додавання згорткових шарів, поки модель не почне добре працювати і перевершити. Після цього процес налаштування наноситься на гіперпараметри моделі до тих пір, поки не з'явиться добре облягаюча, висока точність моделі. Цей процес є трудомістким і потребує величезних обчислювальних ресурсів. Наприклад, представлена нова модель PureFoodNet вимагала повного місяця, щоб бути розроблена і правильно налаштована на високотехнологічну обчислювальну систему.

Архітектура PureFoodNet зображена на рисунку 2.1. Він складається з трьох згортків і класифікаційного блоку. Перший блок містить два згорткових шари, кожен з яких має 128 фільтрів. Другий блок містить три згорткових шари по 256 фільтрів кожен, а третій блок містить три згорткових шари по 512 фільтрів кожен. Останній блок, який зазвичай називають "Верхні шари" або "Класифікаційні шари", містить щільний шар з нейронами 512 і провісником шару (щільний) зі 101 виходом. Для початкової розробки цієї моделі зверніть увагу, що вихід пророчого шару був розроблений відповідно до 101 класів (категорій продуктів харчування) набору даних Food101. Щоб уникнути переоснащення в моделі PureFoodNet, архітектура включає класичні механізми упорядкування, такі як відсів, нормалізація L2 та пакетна нормалізація шарів. Крім того, з метою Тюнінг PureFoodNet, метод ініціалізації ваги був використаний. Щоб вибрати швидку і точну підготовку, був використаний оптимізатор з імпульсом Несторова.

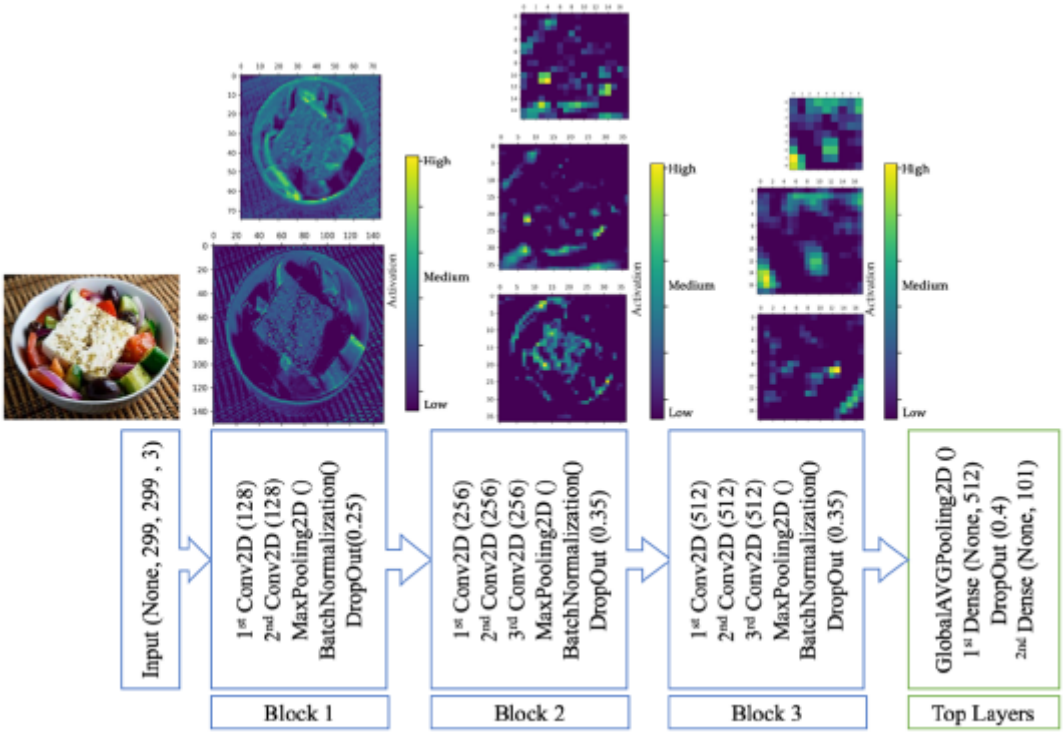


Рисунок 2.1 – Архітектура CNN для розпізнавання продуктів харчування [13]

При визнанні їжі і в багатьох інших випадках важливо знати, що вчиться кожен шар (або навіть кожен нейрон). Одним із способів зробити це є візуалізація значень шарів (локальні "знання"). Отже, питання невикористаних (мертвих) фільтрів можна легко вирішити. Цю проблему можна помітити, перевіривши, чи є деякі карти активації нульовими для багатьох різних входів, що є симптомом, викликаним високим рівнем навчання. Найбільш часто використовуваним методом є візуалізація активацій шарів під час перевалу вперед. Мережі, які використовують функції активації ReLU, як правило, мають активації, які виглядають занадто шумно (відносно blobby і щільні) на початку навчання.

Після кількох навчальних епох і після того, як мережа починає добре виконувати, візуалізація активацій стає більш інтуїтивно зрозумілою, розрідженою і локалізованою. На рисунку 2.1 над кожним блоком показано кілька активацій деяких індикативних шарів для того, щоб виділити інформацію, вивчену в кожному шарі. Зрозуміло, що нижній виглядає (Блок 1) в мережі, тим більше загальної інформації шар зберігає, на відміну від шарів високого рівня

(Блок 3), де більш конкретна/цільова інформація вивчається мережею. Це пов'язано з тим, що в багатьох зображеннях харчових страв існує самодостатність у вигляді повторюваних візерунків однакових форм по всій поверхні зображення. Саме тому очікується, що укладання зображення, викликане згорткових шарів, покращує локалізацію інгредієнтів, що, в свою чергу, призводить до кращого вивчення страв і поліпшення прогнозів.

### 2.3 Трансферне навчання та точне налаштування

Другий альтернативний шлях у підходах до глибокого розпізнавання продуктів харчування на основі навчання, спирається на використання популярних попередньо навчених моделей глибокого навчання, впровадження техніки трансферного навчання. Передача навчання в теорії глибокого навчання є популярною технікою машинного навчання, в якій модель, розроблена і навчена для конкретного завдання, повторно використані для іншого подібного завдання при деякій параметризації (тонка тюнінг). Інтуїтивно зрозуміле визначення трансферного навчання було надано Торрей і Shavlink в (2009): "Передача навчання є поліпшення навчання в новому завданні через передачу знань з пов'язаного шару, яке вже було вивчено". Точне налаштування - це процес, в якому попередньо навчена модель використовується без її верхніх шарів, які замінюються новими шарами, більш підходящими для нового завдання (набору даних). Наприклад, якщо слід використовувати попередньо навчену модель PureFoodNet, верхній шар слід видалити (тобто шари всередині зеленого прямокутника) і тільки згорткові шари повинні зберігатися з їх вагою. Нові верхні шари слід додавати відповідно до потреб нового завдання (нового набору даних), а кількість виходів у шарі прогнозування має бути пов'язана з кількістю нових класів. Це звичайна практика, зробити відповідну рекомбінацію верхніх шарів, переналаштування параметрів і безліч повторень для того, щоб домогтися бажаної продуктивності.

Через простоту використання трансферного навчання, в даний час дуже мало людей розробляють і тренують нові моделі CNN (архітектури) з нуля. Це пов'язано з тим, що важко мати набір даних достатнього розміру для конкретного завдання. Також, як уже згадувалося раніше, розробка нової моделі є складною і трудомісткою. Замість цього зручніше повторно використовувати модель, яка була навчена на дуже великому наборі даних, як-й ImageNet (Zeiler, 2013), який містить більше 14 мільйонів зображень, що охоплюють більше 20 000 категорій взагалі. Також зверніть увагу, що ImageNet містить безліч категорій із зображеннями, що зображують продукти або їстівні предмети в цілому, таким чином, попередньо навчена модель на наборі даних ImageNet, як очікується, буде досить ефективною і легко налаштована для підвищення продуктивності в додатках для розпізнавання продуктів харчування.

## 2.4 Платформи глибокого навчання

Необхідність розробки нової моделі або точного налаштування існуючої попередньо навченої моделі для конкретного завдання іноді під питанням, обумовлена існуванням третьої альтернативи в практиці глибокого навчання: потужних онлайн-платформ, які попередньо навчаються з великими наборами даних на широкому спектрі занять. У цьому розділі представлені деякі популярні платформи для розпізнавання харчових продуктів на основі зображень, а також оцінка їх ефективності та резюме їх переваг і недоліків. Ці платформи продовжують тренуватися на щоденній основі і, таким чином, покращуються з плином часу. Деякі з цих платформ належать до добре налагоджених технологічних підприємств, які отримують вигоду від послуг і додатків глибокого навчання. Ці підприємства пропонують можливість комерційного використання своїх інфраструктур науковим і промисловим громадам або у вигляді платної підписки, або при обмеженому безкоштовному пробному використанні, що, в більшості випадків, цілком достатньо для експериментів. Однак наголошується,

що навчені моделі, пропоновані цими платформами, працюють як «чорні скриньки», оскільки вони не надають достатньо подробиць про свій механізм прогнозування, архітектури тощо і їх параметризація обмежена. Всі ці платформи надають кілька інструментів і програмних інтерфейсів через API.

Vision AI17 (VAI) був розроблений Компанією Google як хмарний сервіс у 2016 році. Вона включає в себе кілька різних функцій виявлення, таких як оптичне розпізнавання символів (OCR), збільшений за допомогою механізму, що фокусується на розпізнавання рукописного тексту на декількох мовах, виявлення обличчя, помірний вміст (явний вміст, такий як дорослий, насильницький тощо), орієнтири та виявлення логотипів.

Clarifai18 (CAI) був розроблений в 2013 році для конкурсу візуального розпізнавання великого масштабу ImageNet (ILSVRC) і завоював топ-5 місць того року (Zeiler, 2013). Clarifai є єдиною платформою, яка надає варіанти щодо вибору моделі для прогнозування. Він пропонує досить цікавий вибір моделей, що застосовуються для різних цілей, таких як загальне розпізнавання зображень, ідентифікація обличчя, не безпечне для перегляду (NSFW) розпізнавання, виявлення кольорів, розпізнавання знаменитостей, розпізнавання продуктів харчування тощо.

Amazon Rekognition19 (AR) був вперше представлений в 2016 amazon як сервіс після придбання Orbeus, глибокого навчального стартапу. Amazon Rekognition забезпечує ідентифікацію об'єктів, людей, тексту, сцен і заходів, і він здатний виявити будь-який невідповідний вміст, з значними можливостями в виявленні та аналізі обличчя (емоції, віковий діапазон, відкриті очі, окуляри, волосся на обличчі і т.д.).

Комп'ютер Vision20 (CV) був розроблений корпорацією Майкрософт як частина когнітивних служб. Ця платформа пропонує виявлення кольору, обличчя, емоцій, знаменитостей, тексту та вмісту NSFW в зображеннях. Важливою особливістю Computer Vision є система розпізнавання, яка ідентифікує текст на 25 різних мовах, поряд з потужною системою інтерпретації контенту.

## 2.5 Порівняння платформ глибокого навчання

У наступних пунктах розгортається експеримент з наданням інтуїції щодо оцінки популярних платформ глибокого навчання в розпізнаванні продуктів харчування на основі зображень. Для експерименту було відібрано два різних тестових зображення зі складним вмістом.

A B

Рисунок 2.2 – Тестові зображення: (A) грецький салат і (B) Фруктовий десерт [7]

Обидва зображення містять кілька інгредієнтів. Зображення A (рис. 2.2 ліворуч) представляє грецький салат. Це зображення містить деякі помітні інгредієнти, такі як нарізані помідори, огірок, цибуля, сир (фета), оливки, кріп і петрушки. Зображення B (рисунок 2.2 праворуч) представляє фруктовий десерт, в якому є полуниця, чорниця, ківі, грейпфрути, апельсини і вершки в нижній частині чашки.

На рисунку 2.3 і на рисунку2.4 присутні результати прогнозування для двох тестових зображень відповідно всіма розглянутими платформами, з (а) зеленими смугами, що відповідають істинно позитивним (правильно прогнозованим) класам (з точки зору інгредієнтів, типу їжі тощо), (б) червоним смугам, що відповідають помилковим позитивним (неправильно прогнозованим)

класам і (в) синім смугам, що відповідають нейтральним, частково правильним або неважливим (до цього випадку дослідження) результатами.

Навіть поверхневе порівняння результатів, представлених на рисунку 2.3, чітко показує, що використовують малі, як в терміні інгредієнтів, так і з точки зору загальних результатів. VAI був дещо кращим, але все ж визначив деякі неіснуючі (або не мають відношення до контексту) класи. CAI виконується значно краще, але також повідомив про неіснуючі харчові інгредієнти в межах зображення. Судячи з усього, кількість правильно визначених класів у поєднанні з їх точністю, чітко оцінює CAI як найкращу платформу для зображення A. Слід зазначити, що CAI була єдиною платформою, яка визнала існування петрушки, яка була поза плитою (праве дно на зображенні A). VAI передбачив з досить високою точністю, що тип їжі може бути "Салат Капрезе" (64% точність) або ізраїльський салат (63% точність), і це явно вибагливі через високу схожість цих трьох салатів.

На рисунку 2.4, в якому показані результати на зображенні B, ще раз зрозуміло, що CV визначив найменшу кількість класів, включаючи два дуже загальних класи (Fresh, Food). Тим не менш, важливо також, що CV є єдиною платформою, яка помітила клас "Нарізаний" з дуже малою впевненістю. AR також не виконала добре, визнавши лише кілька правильних класів і з дуже низькою впевненістю.

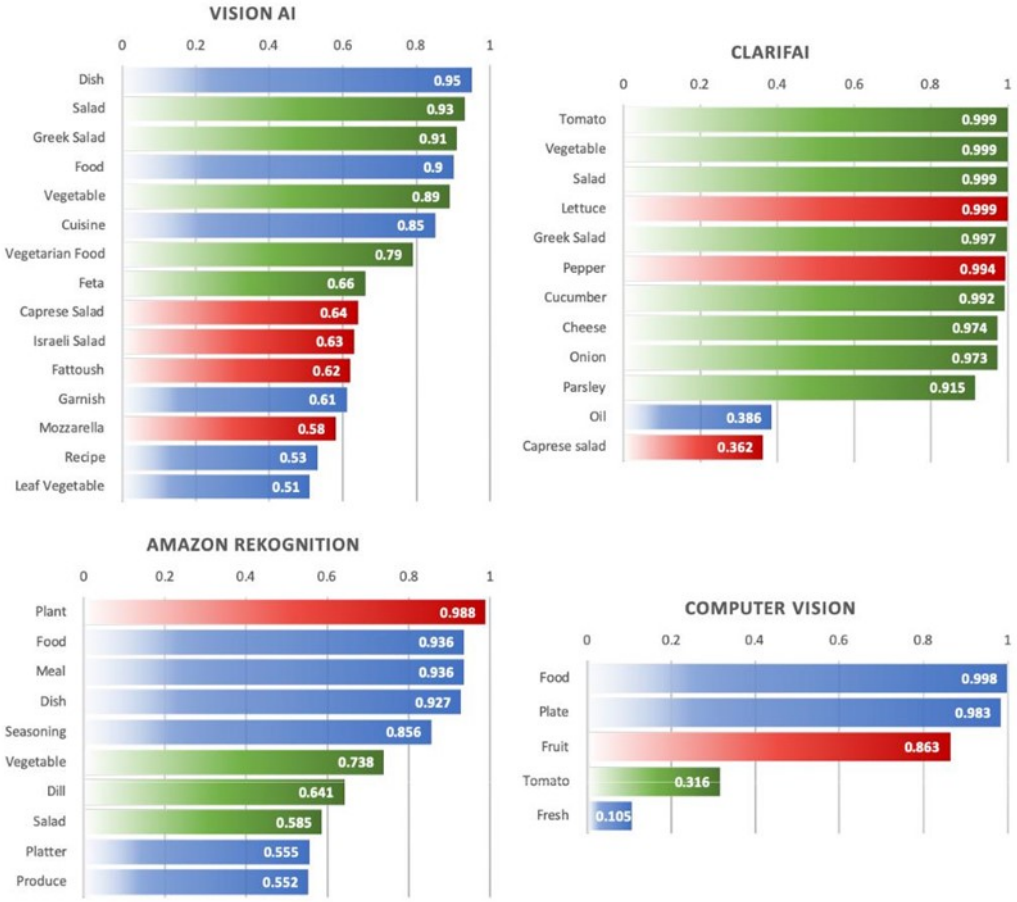


Рисунок 2.3 – Результати розпізнавання продуктів харчування А рецензованими платформами [8]

Зрозуміло, що CAI і VAI добре виступили в цьому випадку. Хоча CAI представив деякі помилкові позитивні (але все ще якось актуальні) призводить до високої впевненості в істинних позитивів і більше класів повідомили зробити його цінним інструментом для завдання. З іншого боку, VAI, здається, вдалося уникнути помилкових позитивних результатів, але кількість класів і довіра значно нижчі.

В цілому, з просто експерименту, CAI і VAI, здається, краще працювали серед протестованих платформ ідентифікації контенту в сфері розпізнавання харчових продуктів, з трохи кращими результатами, наданими CAI, вочевидь завдяки своїй спеціалізованій моделі з харчування. Що важливо з точки зору

розробника, так це те, що жодна з цих платформ не вимагає глибоких навичок програмування для розробки системи аналізу контенту зображення.

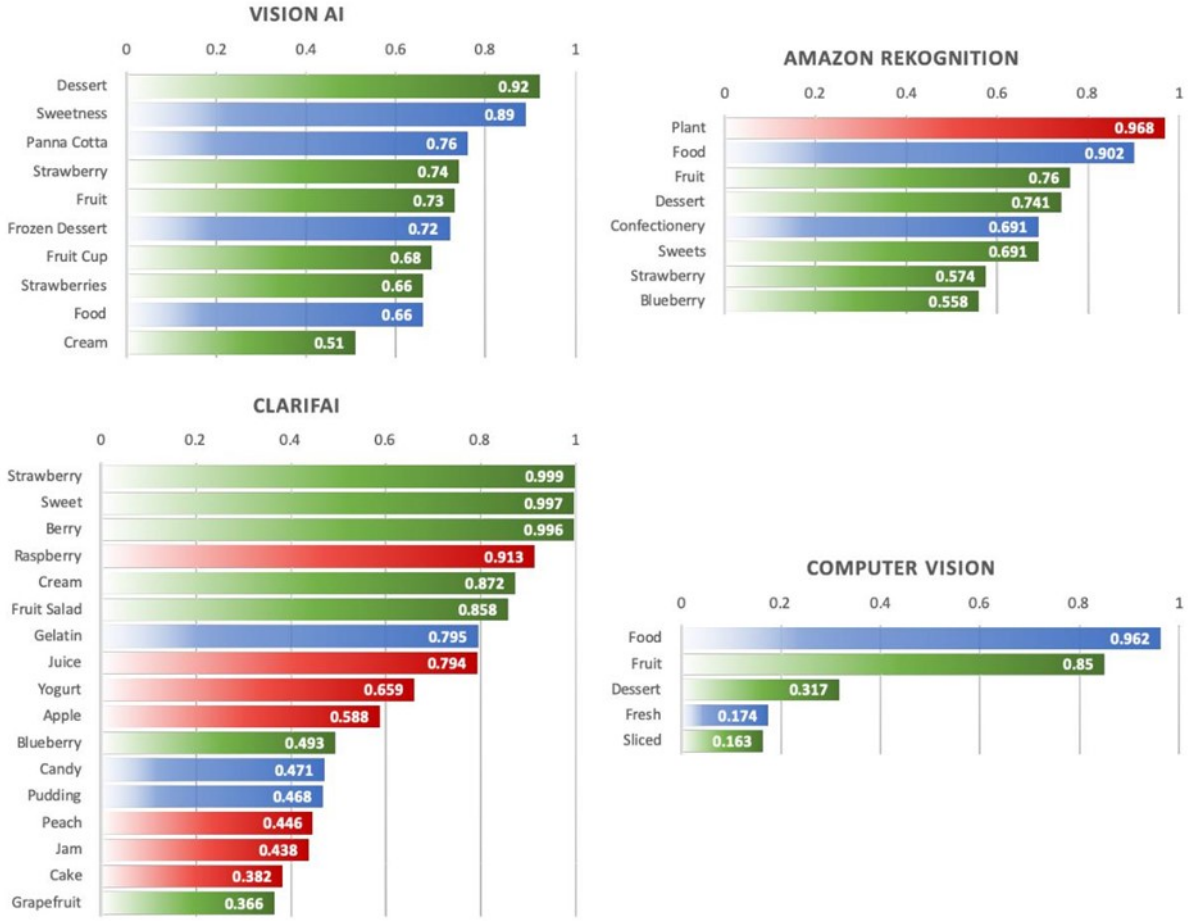


Рисунок 2.4 – Результати по визнанню харчових продуктів зображення В рецензованими платформами [10]

**Висновки до розділу 2**

В даний час практики глибокого навчання вибір для розробки нових моделей або застосування існуючих, досить різний і іноді, думки різко розділені. Таким чином, розробка нової моделі глибокого навчання (архітектури), застосованої до нового домену (проблема - завдання) є складним і трудомістким процесом, який вимагає програмування та математичних навичок за межами

середнього, поряд з розумінням data-driven (емпіричного) модельного навчання. З іншого боку, передача навчання з точним налаштуванням повинна бути прийнята, коли потрібне негайне рішення. Такий підхід менш вимогливий до технічних навичок і теоретичних знань, які могли б бути в основному обмежені розробкою простого методу класифікації. Крім того, сьогодні існує ряд онлайн-платформ, які дозволяють швидко розвивати нові моделі на основі власних технологій, які в основному виступають в якості чорних скриньок для розробників. Ці платформи забезпечують третю альтернативу і, як правило, системи машинного навчання попередньо навчені великим наборам даних, що охоплюють широкий спектр різних класів, готових забезпечити вирішення найрізноманітніших проблем, за допомогою інтерфейсів прикладного програмування (API).

# Розділ 3
# Розробка мережі розпізнавання

## 3.1 Введення в штучні нейронні мережі

Вважається, що теорія нейронних мереж як науковий напрямок вперше була позначена в класичній роботі Уорренна Мак-Коллока й Уолтера Питтса в 1943 р., у якій стверджувалося, що практично будь-яку логічну операцію можна реалізувати за допомогою простої нейронної мережі. Продовжуючи дослідження в цій області, в 1958 р. Френк Розенблатт винаходить одношаровий перцептрон, що є однієї з перших моделей нейронної мережі й демонструє його здатність вирішувати задачі класифікації. Однак, як виявилося пізніше, перцептрон мав серйозні обмеження. В 1969 році Марвін Чі публікує формальний доказ про нездатність перцептрона вирішувати деякі задачі, написавши про це книгу. Лише через 20 років, нейронні мережі стали активно використовуватися, що пов'язане з енергетичним підходом Джона Хопфілда та створенням алгоритму зворотного поширення помилки вперше запропонованого Вербосом і незалежно розробленого іншими авторами. Алгоритм одержує популярність завдяки Румельхарту в 1986 р., який активно використовується в наш час.

Із середини 80-х років теорія нейронних мереж одержала ще більший імпульс дослідження у зв'язку з появою високопродуктивних персональних комп'ютерів [5].

## 3.2 Проблеми, розв'язувані в рамках штучної нейронної мережі

На сьогоднішній момент, коло задач, які можуть бути вирішені з використанням нейронних мереж досить великий. До найпоширеніших можна віднести:

Розпізнавання образів і класифікація**.** Суть полягає в розпізнаванні вхідного образа (символів тексту, зображення, мовних сигналів і т.д.) і вказівці його приналежності до певного класу. Під час навчання нейронної мережі, на вхід їй подають вектор значень ознак образа, із вказівкою його класу. Після навчання, їй можна пред'являти невідомі раніше образи й одержувати відповідь про приналежність до певного класу. У такого роду задачах, установлюється відповідність між вихідним шаром штучної нейронної мережі і класом, який він представляє.

Задачі кластеризації. Задачі кластеризації, відомі так само як класифікація образів “без учителя”, не мають навчальної вибірки з мітками класів. Їхній принцип роботи заснований на виявленні закономірностей і подібності між образами та розміщення цих образів в один кластер або категорію.

Прогнозування. Здатність нейронних мереж до пророкування результатів обумовлена узагальненням і виділенням загальних залежностей між вхідними й вихідними даними. Після навчання, мережа здатна передбачити майбутнє значення деякої послідовності, ґрунтуючись на її попередніх змінах. Для того що б штучна нейронна мережа могла дати максимально точну відповідь, необхідно навчати її на даних, у яких дійсно прослідковується якась залежність. А якщо ні, то, прогнозування найімовірніше не дасть ніяких результатів.

Апроксимація функцій. Задача апроксимації штучної нейронної мережі, полягає в знаходженні оцінки невідомої функції, по її значеннях. Точність апроксимації залежить від вибору структури нейронної мережі.

Оптимізація. Способи оптимізації з використанням нейронних мереж, мають на увазі знаходження такого розв'язку, який задовольняє системі умов і мінімізує цільову функцію. Як приклад можна розглянути алгоритм, що реалізує якусь послідовність дій, який можна повністю замінити функціонуванням нейронної мережі, використовуючи правила, по яких він складений.

### 3.3 Структура штучного нейрона

Штучний нейрон є елементарною структурною одиницею штучної нейронної мережі, що й представляє із себе спрощену моделлю біологічного нейрона. На рисунку 3.1 зображений нейрон, входами якого можуть бути або вхідні дані, або виходи від такого ж нейрона. Входи з'єднані із гніздом нейрона S за допомогою синаптичних зв'язків. Кожний синапс має свою вагу, при передачі в гніздо нейрона вхідного параметра, він відповідно множиться на вагу, тобто $x_i * w_i$. Стан нейрона $S$, визначається у вигляді формули:

$$S = \sum_{i=1}^{N} x_i * w_i \tag{3.1}$$

Вихід нейрона, є функція його стану $y=f(S)$.

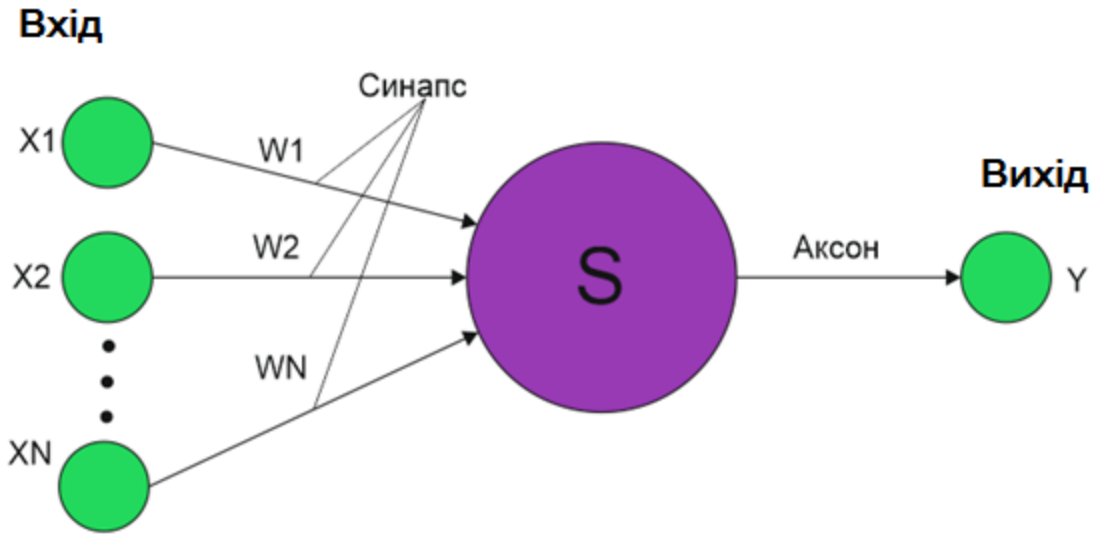


Рисунок 3.1 – Візуалізація штучного нейрона [7]

Функція $f$ називається функцією активації. Такі функції не можуть бути лінійними, оскільки нейронні мережі з лінійною функцією активації ефективні

тільки на одному рівні, незалежно від того, наскільки складна їхня структура. Однієї з найпоширеніших функцій є нелінійна функція з насиченням, названа логістичною функція або сигмоїда:

$$f(x) = \frac{1}{1 + e^{-ax}} \tag{3.2}$$

Параметр *a* відповідає за положистість функції. Чим він менше, тим більше пологим стає сигмоїда. Слід зазначити, що дана функція диференцюєма на всій осі абсцис, що є необхідною умовою в деяких алгоритмах навчання нейронних мереж. Крім того, вона має властивість підсилювати слабкі сигнали краще, чим більші, а також запобігає насиченню від більших сигналів тому що вони відповідають областям аргументів, де сигмоїда має пологий нахил (Рисунок 3.2) [2].

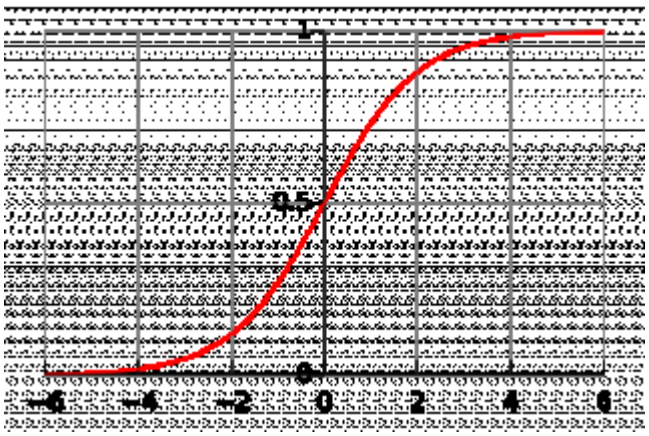


Рисунок 3.2 – Сигмоїда [2]

### 3.4 Архітектура нейронної мережі

Можна виділити кілька основних типів нейронних мереж.

Багатошарові мережі. У багатошарових мережах один або кілька нейронів поєднуються в шари. Шар – сукупність нейронів, на вхід яких подається той самий

загальний сигнал. У мережах такого типу, зовнішні вхідні дані подаються на входи нейронів першого шару, а вихідні дані є результатом останнього вихідного шару. Крім вхідного й вихідного шарів, у багатошарових мережах так само є присутнім один або кілька схованих шарів. Зв'язки виходів нейронів від деякого шару *i* до деякого шару *i+1* називають послідовним.

Повнозв’язні нейронні мережі. В повнозв’язній нейронній мережі кожний нейрон передає свій сигнал іншим нейронам.

Вихідними сигналами, можуть бути всі або деякі сигнали нейронів після декількох тактів функціонування. Усі вхідні сигнали подаються на вхід усім нейронам.

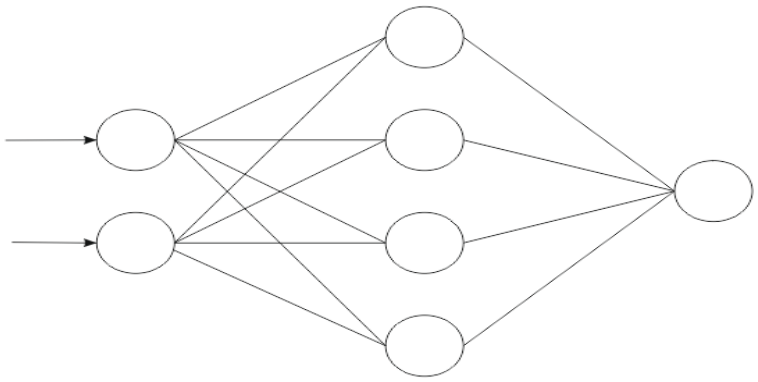

Рисунок 3.3 – Багатошарова нейронна мережа [7]

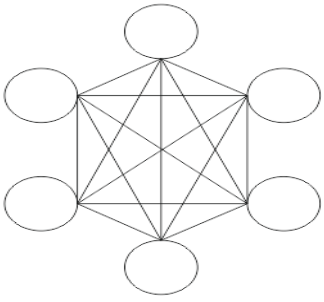

Рисунок 3.4 – Повнозв’язна нейронна мережа [7]

Кожна нейронна мережа має не тільки свою архітектуру, але й тип, який краще підходить для розв'язку конкретної задачі. Приміром, згорткові нейронні

мережі (convolutional neural network CNN) набагато краще справляється з розпізнаванням образів і проблемами комп'ютерного зору. Її відмінність від інших типів штучної нейронної мережі полягає в тому, що кожний фрагмент зображення множиться на матрицю (ядро) згортки поелементно, а результат підсумується й записується в аналогічну позицію вихідного зображення. Якщо кожний піксель зображення розглядався б окремо, це привело б мережу до швидкого перенавчання і її здатність розпізнавання образів була б точна тільки на навчальній вибірці.

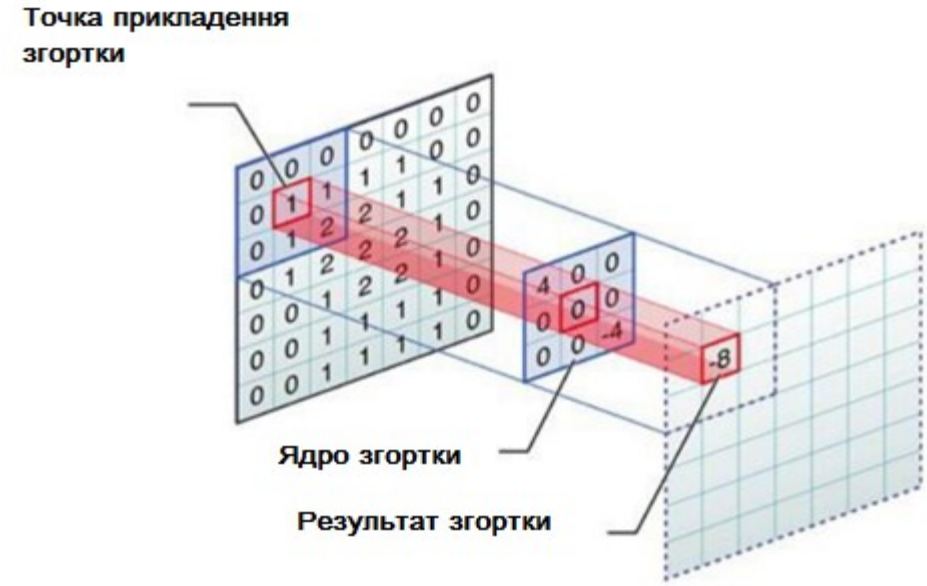


Рисунок 3.5 – Згорткова нейронна мережа [7]

Існують нейронні мережі, що так само розгортають (deconvolutional networks DN), так само називані зворотними графічними мережами, що і є зворотними до CNN. Їхня задача генерувати зображення по заданих ознаках. Приміром, при передачі мережі слова “кіт” вона повинна буде згенерувати зображення, схожі на котів.

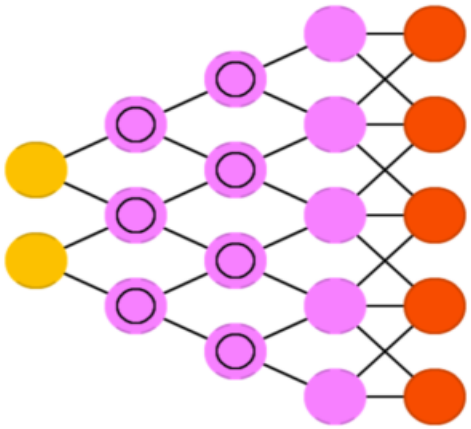

Рисунок 3.6 – Нейронна мережа, яка розгортається [7]

Для пророкування слів у пропозиції, пророкуванні наступного числа в послідовності, виділення головної думки тексту, генерації нової інформації схожої на дану використовуються рекуррентніе нейронні мережі (recurrent neural network RNN). В рекурентних нейронних мережах нейрони обмінюються інформацією між собою, приміром, у добавок до нового шматочка вхідних даних нейрон так само одержує інформацію про попередній стан мережі. Таким чином, у мережі реалізується так звана “пам'ять”, що принципово міняє характер її роботи й дозволяє аналізувати будь-які послідовності даних.

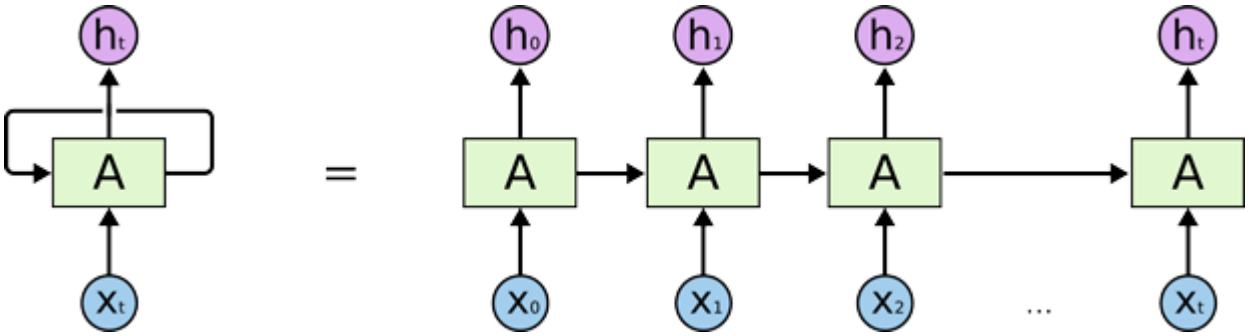


Рисунок 3.7 – Рекурентна нейронна мережа [8]

Найпростішими нейронними мережами, є мережі прямого поширення (feedforward neural network FFN), які взяті за основу для створення багатьох інших мереж. Дані мережі прямолінійні й передають інформацію від входу до виходу.

Нейрони кожного шару не зв'язані між собою, а сусідні шари звичайно повністю зв'язані. Вид FFN можна розглядати як багатошарову мережу.

### 3.5 Навчання нейронних мереж

Процес навчання нейронної мережі є необхідним для її здатності виконувати поставлені задачі. Цей процес може бути розглянутий як настроювання архітектури мережі й ваг зв'язків за допомогою моделювання середовища, у яке ця мережа вбудована. Властивість нейронної мережі навчатися на прикладах дозволяє спростити завдання умов для розв'язку конкретної задачі, у порівнянні із системами, які додержуються набору певних правил, складених експертами. Розділяють алгоритми навчання із учителем і без учителя.

Процес навчання із учителем має на увазі пред'явлення мережі вибірки навчальних прикладів. Кожний зразок подається на вхід нейронної мережі, потім проходить процедуру обробки всередині штучної нейронної мережі. Після обчислення вихідного сигналу штучна нейронна мережа порівнює отриманий результат з відповідним значенням цільового вектора, що представляє собою необхідний вихід мережі. Обчисливши помилку, відбувається зміна вагових коефіцієнтів зв'язків усередині мережі по обраному алгоритму. Ваги підбудовуються під кожний вектор доти, поки помилка по всьому масиву вхідних даних не досягнеться заданого рівня.

Навчання без учителя не вимагає знання правильної відповіді на кожний приклад навчальної вибірки. У цьому випадку розкривається внутрішня структура даних або кореляція в системі даних, що дозволяє розподілити образи по категоріях.

Залежно від розв'язуваної задачі в навчальній вибірці використовуються ті або інші типи даних і різні розмірності вхідних/вихідних сигналів. Вхідні дані прикладів навчальної вибірки - зображення, таблиці чисел, розподілу. Типи вхідних даних - бінарні (0 і 1), біполярні (-1 і 1) числа, цілі або дійсні числа з

деякого діапазону. Вихідні сигнали мережі - вектора цілих або дійсних чисел. Для розв'язку практичних задач часто потрібні навчальні вибірки великого об'єму. Через жорстко обмежений об'єм оперативної пам'яті комп'ютера розмістити в ній більші навчальні вибірки неможливо. Тому вибірка ділиться на сторінки - групи прикладів. У кожний момент часу лише одна сторінка прикладів розташовується в пам'яті комп'ютера, інші - на жорсткому диску. Сторінки послідовно завантажуються на пам’ять комп'ютера. Навчання мережі відбувається по всій сукупності сторінок прикладів, по всій навчальній вибірці.

У цей час відсутня універсальна методика побудови навчальних вибірок. Набір навчальних прикладів формується на розгляд користувача програми моделювання нейронних мереж індивідуально для кожної конкретної розв'язуваної задачі.

Якщо в ненавчену нейронну мережу ввести вхідний сигнал одного із прикладів навчальної вибірки, то вихідний сигнал мережі буде суттєво відрізнятися від бажаного вихідного сигналу, певного в навчальній вибірці. Функція помилки чисельно визначає подібність усіх поточних вихідних сигналів мережі й відповідних бажаних вихідних сигналів навчальної вибірки. Найпоширенішою функцією помилки є середньоквадратичне відхилення. Однак запропоновані й інші функції помилки.

Для навчання нейронних мереж можуть бути використані різні алгоритми. Можна виділити дві найбільші групи алгоритмів - градієнтні й стохастичні. Градієнтні алгоритми навчання мереж засновані на обчисленні часток похідних функції помилки по параметрах мережі. Серед градієнтних розрізняють алгоритми першого й другого порядків. У стохастичних алгоритмах пошук мінімуму функції помилки ведеться випадковим чином.

При навчанні мереж, як правило, використовується один із двох наступних критеріїв зупинки: зупинка при досягненні деякого малого значення функції помилки, зупинка у випадку успішного розв'язання всіх прикладів навчальної вибірки.

Перед навчанням виконується ініціалізація нейронної мережі, тобто присвоювання параметрам мережі деяких початкових значень. Як правило, ці початкові значення - деякі малі випадкові числа.

Для формування навчальних вибірок, ініціалізації й навчання в програмах моделювання нейронних мереж використовуються спеціальні процедури. Можливість використання багатосторінкового навчання є дуже важливою при розв'язанні практичних задач за допомогою нейронних мереж, що моделюються на звичайних комп'ютерах.

Навчання - це ітераційна процедура, яка при реалізації на звичайних комп'ютерах, вимагає значного часу. Алгоритми навчання суттєво відрізняються за швидкістю збіжності. Однієї з найважливіших характеристик програм для моделювання нейронних мереж є швидкість збіжності алгоритму (або алгоритмів) навчання, які реалізовані в програмі.

Теорія навчання розглядає три фундаментальні властивості, пов'язані з навчанням по прикладах: ємність, складність зразків і обчислювальна складність. Під ємністю розуміється, скільки зразків може запам'ятати мережа, і які функції й границі прийняття розв'язків можуть бути на ній сформовані. Складність зразків визначає число навчальних прикладів, необхідних для досягнення здатності мережі до узагальнення. Занадто мале число прикладів може викликати "перенавченість" мережі, коли вона добре функціонує на прикладах навчальної вибірки, але погано - на тестових прикладах, підлеглих тому ж статистичному розподілу. Відомі 3 основних типи правил навчання: корекція помилок, машина Больцмана та правило Хебба.

Правило корекції помилок. При навчанні із учителем для кожного вхідного прикладу заданий бажаний вихід d. Реальний вихід мережі у може не збігатися з бажаним. Принцип корекції помилок при навчанні полягає у використанні сигналу (d-y) для модифікації ваг, що забезпечує поступове зменшення помилки. Навчання має місце тільки у випадку, коли персептрон помиляється. Відомі різні модифікації цього алгоритму навчання.

Навчання Больцмана являє собою стохастичне правило навчання, яке випливає з інформаційних теоретичних і термодинамічних принципів. Метою навчання Больцмана є таке налаштування вагових коефіцієнтів, при якій стани видимих нейронів задовольняють бажаному розподілу ймовірностей. Навчання Больцмана може розглядатися як спеціальний випадок корекції помилок, в якому під помилкою розуміється розбіжність кореляцій станів у двох режимах.

Правило Хебба є самим старим навчальним правилом, або постулатом навчання Хебба. Хебб опирався на наступні нейрофізіологічні спостереження: якщо нейрони по обидва боки синапса активізуються одночасно й регулярно, то сила синаптичного зв'язку зростає. Важливою особливістю цього правила є те, що зміна синаптичної ваги залежить тільки від активності нейронів, які зв'язані даним синапсом. Це суттєво спрощує ланцюг навчання.

Для того, щоб перевірити навички, придбані нейронною мережею в процесі навчання, використовується імітація функціонування мережі. У мережу вводиться деякий сигнал, який, як правило, не збігається з жодним із вхідних сигналів прикладів навчальної вибірки. Далі аналізується вихідний сигнал мережі. Тестування навченої мережі може проводитися на одиночних вхідних сигналах, або на контрольній вибірці, яка має структуру, аналогічну навчальній вибірці.

### 3.6 Алгоритм зворотного поширення помилки

Алгоритм зворотного поширення помилки (back propagation) належить до методу навчання з корекцією помилок і, як правило, застосовується до багатошарових перцептронів. Це метод навчання "із учителем", при якому "наставник" навчає мережу, так як дитину навчають читати й писати. При навчанні на вхідний шар багаторазово подаються образи сигналів, розпізнаванню яких нейронна мережа повинна бути навчена, і коректуються ваги нейронів для досягнення бажаного вихідного сигналу. Для поліпшення якості розпізнавання

образу, що подається на вхідний шар, можуть бути злегка змінені (доданий шум і т.п.). Детально процедура навчання виглядає так:

1) Вибірка вхідних даних (множина образів, класифікованих учителем) розбивається на дві: навчальну й контролюючу послідовності. Звичайно навчальна послідовність містить більше образів, ніж перевірочна.

2) Проводиться ініціалізація всіх ваг, включаючи граничні, невеликими випадковими величинами (звичайно в діапазоні [-1; +1]). Це визначає початкову точку на поверхні помилок для методу градієнтів, позиція може виявитися вирішальною для збіжності мережі.

3) Проводиться прямий прохід мережі для першого образу з навчальної вибірки від вхідного шару через сховані шари до вихідного шару: кожний нейрон підсумовує добуток входів на ваги й видає результат функції активації, застосованої до цієї суми, на нейрони наступного рівня.

4) Обчислюється різниця між дійсним і бажаним вихідним значенням кожного нейрона вихідного шару. При їхній розбіжності має місце помилка в розпізнаванні (класифікації) образів.

5) Проводиться процедура зворотного поширення цих помилок по зв'язках від вихідних нейронів до вхідних і визначаються помилки для кожного нейрона. Розглянемо цю процедуру докладніше. Припустимо, вихідне значення нейронної мережі було 0.5, а бажане 0. Нехай помилка визначається по формулі:

$$E = y_i - d_i, \tag{3.3}$$

де E – помилка мережі, $y_i$ – бажане значення на i-ом виході нейронної мережі, $d_i$ – вихідне значення мережі на i-ом виході.

У цьому випадку, на виході помилка буде рівна E = 0.5-0 = 0.5. Якщо розглядати в якості функції активації сигмоїд, то розрахунки Δw, на який необхідно зрушити ваги синаптичних зв'язків передостаннього і останнього шару, буде проводитися по формулі:

$$\Delta w = E * sigmoid(x)dx \tag{3.4}$$

Тут sigmoid(x)dx – похідна функції активації, рівна sigmoid(x)(1-sigmoid(x)). Під x розуміється значення, яке було отримано шляхом підсумовування всіх вихідних значень нейронів передостаннього шару, відповідно помножених на wi. Завершальний крок корекції ваг синаптичних зв'язків між останнім і передостаннім шаром буде проводитися за наступною формулою:

$$w_i = w_i - m_i * \Delta w * l \tag{3.5}$$

де $m_i$ – вихідні значення і нейрона передостаннього шару;

$l$ – параметр відповідальний за швидкість навчання.

Після того як ваги пораховано, можемо порахувати помилку вихідного значення mi по формулі E = $w_i$*Δw. Знаючи помилку кожного нейрона на попередньому шарі, застосовуючи формули, які використовувалися вище, можемо знайти відхилення ваг і значення нейронів на попередніх шарах.

6) Знову проводиться прямий прохід мережі вже для чергового образу навчальної вибірки.

По завершенню навчальної фази мережа перевіряється за допомогою контролюючої послідовності, що містить образи, не пред'явлені раніше. ( При цьому не проводиться корекція ваг, а лише обчислюється помилка). У випадку, якщо якість роботи знайдена задовільним, мережа вважається готової до роботи. А якщо ні, то мережа зазнає повторного навчання, при якому можлива зміна деяких параметрів (початкові ваги, кількість нейронів у схованих шарах, додаткові навчальні образи й т.п.).

### 3.7 Деякі проблеми штучної нейронної мережі

Однієї з головних проблем нейронних мереж є необхідність у дуже великих об'ємах даних. Кілька років назад були продемонстровані алгоритми, здатні розпізнавати образи на зображеннях краще, ніж люди. Щоб перевершити людину, мережам довелося вивчити більше 1,2 мільйони зображень, у той час як дитина або тварина може навчитися визначати новий об'єкт після однієї побаченої світлини. Практично для будь-якої задачі розпізнавання образів, штучним нейронним мережам необхідно в сотні тисяч раз більше інформації, ніж людині.

Іншою проблемою є непристосованість нейронних мереж до мультизадачності. Сучасні алгоритми призначені тільки для розв'язку однієї задачі. Штучну нейронну мережу можна навчити розпізнавати собак або створювати музику. Але на сьогоднішній момент не існує таких мереж, які могли б виконувати обидві ці задачі. Якщо розглянути нейронну мережу з фіксованою кількістю шарів і нейронів у кожному шарі, то при постійному збільшенні, приміром, навчаючих образів, рано чи пізно нейронна мережа перестане піддаватися обробці. Або якщо вже навчену нейронну мережу розпізнавати особу почати навчати розпізнавати кішок, то вона почне забувати про обличчя для звільнення пам'яті для нової інформації.

В 2016 році був проведений експеримент. Дослідники прагнули визначити, на яку частину зображення “дивиться” нейронна мережа щоб виконати задачу. Вони показали нейронній мережі світлину спальні й запитали в неї “Що висить на вікнах?”. Замість того, щоб подивитися на вікна, штучна нейронна мережа стала дивитися на підлогу, після того на ліжко, із чого зробила висновок, що на вікні висять фіранки. Справа в тому, що неможливо подивитися “углиб” нейронних мереж, для того щоб зрозуміти, як вони працюють.

Так само, однієї із проблем нейронних мереж є її правильне проектування, приміром, невідомо скільки шарів необхідні для даної задачі, скільки потрібно вибрати елементів у кожному шарі, як мережа буде реагувати на дані, які не

включені в навчальну вибірку (яка здатність мережі до узагальнення) і який розмір навчальної вибірки необхідний для здатності мережі до узагальнення. Найчастіше дані параметри визначаються шляхом проб і помилок. Існуючі теоретичні результати дають лише слабке уявлення про те, якими повинні бути ці параметри.

### 3.8 Створення нейронних мереж

В останні кілька років спостерігається вибух інтересу до нейронних мереж. Усе більше й більше матеріалу про нейронні мережі на різних мовах можна знайти на просторах інтернету. Для спрощення створення штучної нейронної мережі стали створюватися різні фреймворки, у яких уже реалізована апаратна частина, відповідальна за обчислення функцій помилок, вистава нейронів і їх синаптических зв'язків між собою, створення архітектур нейронних мереж, їх типів і багато іншого. Більше того, стали з'являтися програмні засоби, що дозволяють створювати складні нейронні мережі без необхідності писати код.

Однією з лідируючих мов у створенні штучних нейронних мереж є скриптова мова python. Через свою простоту у використанні великої кількістю бібліотек для нейронних мереж, написаних для неї, вона становить найбільший інтерес у створенні штучної нейронної мережі.

Найбільш часто використовуваними бібліотеками є Tensorflow, розроблена компанією Google, Theano основним розроблювачем якої є група машинного навчання в Монреальському університеті і бібліотека Keras, що представляє із себе надбудову над одним з перерахованих вище фреймворків.

Як приклад, розглянемо програмну реалізацію нейронної мережі на Keras.

```
x = np.linspace(0,10,5000)
np.random.shuffle(x)
y = x*x

model = Sequential()
model.add(Dense(64, activation='sigmoid', input_dim=1))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer="sgd")

model.fit(x, y, nb_epoch=1000,verbose=2)
test = np.array([11,9])
pred = model.predict(test)
print(pred)
```

Рисунок 3.8 – Програмна реалізація мережі на Keras

У якості *x* в даному прикладі виступає numpy-масив, що складається з 5000 елементів від 0 до 10. Даний масив перемішується, після чого перебуває функція y=x*x. Дані масиви можна розглядати як вхідні й вихідні дані для нейронної мережі. За допомогою бібліотеки Keras, створюємо модель Sequential, у яку по черзі додаємо шари з 64 нейронів із сигмоїдною функцією активації і ще один шар, що складається з одного нейрона, який є вихідним. Для навчання мережі використовується функція fit, яка охоплює вектор вхідних параметрів (x), вектор вихідних параметрів (y), кількість епох, у цьому випадку 1000. На кожному кроці (епосі), помилка зменшується. Після навчання мережі, в якості тестування відправимо їй два вхідні значення 11 і 9. Як можемо помітити (Рисунок 3.9) отримані результати 114 і 81 не є точними. Усе тому, що для задачі екстраполяції більше підходять рекурентні нейронні мережі, в той час як із задачею апроксимації нейронна мережа впоралася з невеликою погрішністю. Використовуючи більш складні структури нейронної мережі, додавши більше епох навчання та шарів з нейронами, можна добитися більш точних результатів.

```
 - 0s - loss: 0.0042
Epoch 999/1000
 - 0s - loss: 0.0044
Epoch 1000/1000
 - 0s - loss: 0.0702
[[114.113846]
 [ 81.1832  ]]
```

Рисунок 3.9 – Навчання мережі та одержання тестових результатів [8]

Даний приклад, демонструє лише малу частину всіх можливостей бібліотеки Keras. Його інструментарій дозволяє так само створювати рекурентні та згорткові нейронні мережі.

### 3.9 Програмна реалізація нейронної мережі

Дано два зображення. Необхідно використовуючи нейронну мережу розпізнати на першому зображенні руки і людської особи, розпізнати руки і людські особи на другому зображенні, і визначити, чи була зміна позицій розпізнаваних об'єктів на першому зображенні з позиціями на другому зображенні відповідно на задану величину. Якщо позиція змінилася, сповістити про це в програмі.

Дану задачу можна розглядати як детектор руху заданих об'єктів і надалі поліпшити до захоплення зображення з камер і порівнянням двох наступних один за одним кадрів.

Для створення даного проекту планувалося використання Tensorflow object detection API. Для його функціонування, необхідно вибрати із уже наданих преднавчених мереж вибрати найбільш підходящу по швидкості й точності розпізнавання. Так само, попередньо необхідно створити датасет, що містить влучні вхідні зображення із позиціонуванням кожного розпізнаваного об'єкту. Для створення датасету використовувалася програма labelimg, за допомогою якої були створені xml-файли з розміткою рук і осіб людей на зображенні. Після чого, дані файли були перетворені за допомогою python-скрипта у файли запису,

використовувані Tensorflow object detection API. На вибір дана програма надає великий спектр нейронних мереж, які відрізняються своєї архітектурою. Як приклад, була використана згорткова мережа, з назвою “ssd_mobilenet_v2_coco”. Вона є найшвидшою із усіх наявних по швидкості, але самою неточною по розпізнаванню об'єктів. У більшості випадків її використовують для розпізнавання об'єктів у реальному часі на камерах мобільних телефонів. До даної мережі додається конфігураційний файл для її налаштування, де можна вказати такі параметри як швидкість навчання, функції активації нейронів, спосіб навчання нейронної мережі. Після налаштування через командний рядок був запущений процес навчання на даних, які були підготовлені раніше за допомогою labelimg. Незважаючи на те, що всі параметри для навчання були встановлені для мінімальної витрати ресурсів комп'ютера, процес навчання нейронної мережі по підрахунках міг би забрати тривалий час. На кожний крок навчання нейронної мережі в середньому йшло по 60 секунд. Для нормального функціонування нейронної мережі необхідно було зразкових 20000 кроків, що зайняло б приблизно 330 годин, тому, даний спосіб реалізації не підходив для виконання задачі.

Для створення програми буде використаний фреймворк Keras для Python, що дозволяє створити нейронну мережу й уникнути проблем з більшим періодом її навчання.

Будь-яке зображення може бути презентовано у вигляді двомірного масиву, індексами якого є координати пікселів зображення, а їх значеннями масив, що складається з 3 елементів (R, G, B) в діапазоні 0-255. Таким чином, кожній точці зображення однозначно відповідає три числа, що визначають її колір.

Надалі, за допомогою бібліотеки Keras планується створити модель згорткової нейронної мережі Sequential, яка буде навчена на навчальній вибірці, створеної в програмі labelimg, яка спрощує роботу виділення об'єктів на зображенні. Після навчання нейронної мережі, за допомогою бібліотеки Opencv (бібліотека комп'ютерного зору) буде проводитися перетворення і завантаження

даних зображень в оперативну пам'ять комп'ютера. Зображення будуть передаватися на вхід навченої нейронної мережі, яка буде виділяти задані об'єкти. Дані об'єкти будуть зберігатися в масиві, кожний елемент якого буде мати позицію x і y, висоту й ширину h і w відповідно. Поміщаючи в масив у такий же спосіб дані другого зображення, усі позиції об'єктів будуть порівнюватися з позиціями першого зображення, а також буде враховуватися їхній клас. Якщо, поелементно порівнюючи позиції об'єктів на першому й другому зображенні, буде виконуватися умова:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \geq E\,, \tag{3.6}$$

де $x_1$, $y_1$ – координати об'єкта на першому зображенні;

$x_2$, $y_2$ – координати приблизно того ж об'єкта на другому зображенні;

E – максимальна величина відхилення об'єкта, то це буде означати, що об'єкт був зміщений.

## Висновки до розділу 3

У роботі були розглянуті теоретичні та практичні основи нейронних мереж. У ході багаторазових спроб навчання штучної нейронної мережі для реалізації даної задачі було вирішено використовувати елементи бібліотек, що дозволяє максимально швидко та просто створювати нейронні мережі різних типів і архітектур.

Нейронні мережі займають лідируючі місця в області штучного інтелекту. Завдяки ним, багато задач можуть бути вирішені без конструювання алгоритмів. Але для того, щоб нейронні мережі могли функціонувати хоча б приблизно як біологічні, їм треба буде пройти дуже довгий процес еволюції.

# Розділ 4

# Дослідження ефективности застосування мережі

## 4.1 Формування нейронної мережі

Згорткова нейронна мережа (CNN), застосовується в рамках ширшої області глибокого навчання, та стала революційною силою в додатках комп'ютерного зору, особливо в останні півтора десятиліття. Одним з основних випадків використання є класифікація зображень, наприклад, визначення того, чи є зображення з собакою або кішкою.

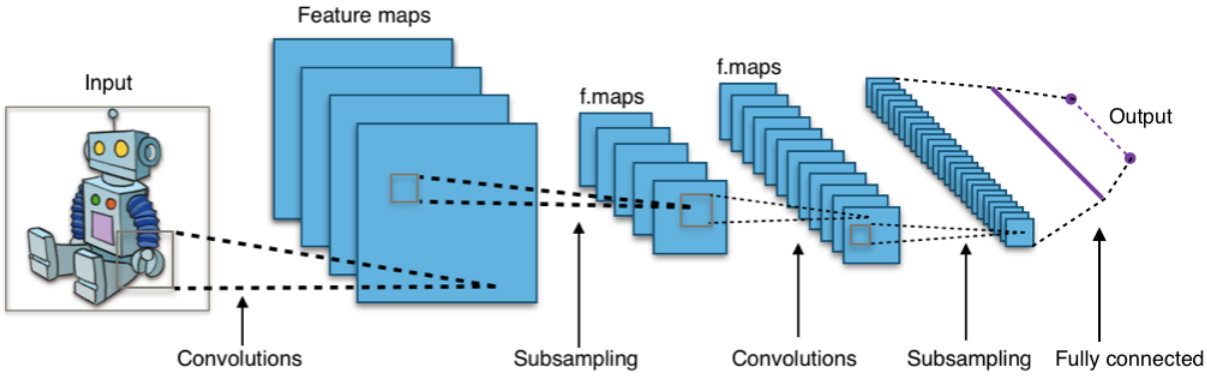


Рисунок 4.1 – Структура згорткової нейронної мережі [4]

Звичайно, не доведеться обмежуватися бінарним класифікатором; CNN можуть легко масштабувати до тисяч різних класів, як це бачимо у відомому наборі даних ImageNet 1000 класів, що використовується для порівняльного аналізу продуктивності алгоритму комп'ютерного зору.

За останні пару років ці передові методи почали ставати доступними для широкої спільноти розробників програмного забезпечення. Промислові пакети, такі як Tensorflow, дали нам ті ж будівельні блоки, які Google використовує для написання програм глибокого навчання для вбудованих/мобільних пристроїв для масштабованих кластерів у хмарі – без необхідності кодувати матричні операції

GPU, часткові похідні градієнти та стохастикові оптимізатори, які роблять можливими ефективні програми.

Перш за все, це зручні API, такі як Keras, які абстрагуються від деяких деталей нижнього рівня і дозволяють нам зосередитися на швидкій прототипації глибокого графіка обчислень навчання. Так само, ніби змішати і вирішувати Legos, щоб отримати бажаний результат.

Як вступний проект для вирішено використовувати попередньо навчений класифікатор зображень, який поставляється з Keras, і перекваліфікувати його на набір даних, який вважаю цікавим.

Створення потужних моделей класифікації зображень з використанням малих даних, і пов'язаний сценарій.

Побудовано систему з метою експериментів з Deep Learning. Ключовими компонентами є Nvidia Titan X Pascal w/12 GB пам'яті, 96 ГБ системної оперативної пам'яті, а також 12-ядра Intel Core i7. Він працює під керуванням 64-бітної Ubuntu 16.04 та використовує дистрибутив Anaconda Python. На жаль, ви не зможете стежити разом з ноутбуком у власній системі, якщо у неї буде достатньо оперативної пам'яті. У майбутньому варто дізнатись, як обробляти більше, ніж набори даних оперативної пам'яті у виконавця.

Після точного налаштування попередньо навченої моделі Google InceptionV3, можна досягти близько 82,03% Top-1 Точність на тестовому наборі за допомогою одного вектора на елемент. Використовуючи 10 культур на прикладі і беручи найбільш часті прогнозовані класи(и), змогли досягти 86,97% Top-1 точність і 97,42% Top-5 точність. InceptionV3: 88.28% Топ-1 точність з невідомими культурами. ResNet200: 90,14% Top-1 точність на наборі даних Food-101, доповнена 19 корейськими стравами. WISeR: 90.27% Топ-1 точність з 10-культурами.

### 4.2 Завантаження та попередня обробка набору даних

Імпортуємо всі пакети, необхідні для блокнота:

```
import matplotlib.pyplot as plt
import matplotlib.image as img
import numpy as np
from scipy.misc import imresize

%matplotlib inline

import os
from os import listdir
from os.path import isfile, join
import shutil
import stat
import collections
from collections import defaultdict

from ipywidgets import interact, interactive, fixed
import ipywidgets as widgets

import h5py
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
from keras.applications.inception_v3 import preprocess_input
from keras.models import load_model
Using TensorFlow backend.
```

Завантажити набір даних і запис його в папці блокнота. Це може бути простіше зробити в окремому вікні термінала.

```
# !wget http://data.vision.ee.ethz.ch/cvl/food-101.tar.gz
# !tar xzvf food-101.tar.gz
```

Подивимося, які продукти представлені тут:

```
apple_pie       eggs_benedict      onion_rings
```

```
baby_back_ribs        escargots               oysters
baklava               falafel           pad_thai
beef_carpaccio        filet_mignon      paella
beef_tartare          fish_and_chips          pancakes
beet_salad      foie_gras               panna_cotta
beignets        french_fries       peking_duck
bibimbap        french_onion_soup       pho
bread_pudding         french_toast      pizza
breakfast_burrito   fried_calamari     pork_chop
bruschetta      fried_rice              poutine
caesar_salad          frozen_yogurt     prime_rib
cannoli               garlic_bread      pulled_pork_sandwich
caprese_salad         gnocchi           ramen
carrot_cake     greek_salad             ravioli
ceviche               grilled_cheese_sandwich  red_velvet_cake
cheesecake      grilled_salmon          risotto
```

Подивимося на деякі випадкові зображення з кожного класу їжі. Можна клацнути правою кнопкою миші і відкрити зображення в новому вікні або зберегти його, щоб побачити його з більш високою роздільною здатністю.

```
root_dir = 'food-101/images/'
rows = 17
cols = 6
fig, ax = plt.subplots(rows, cols, frameon=False, figsize=(15, 25))
fig.suptitle('Random Image from Each Food Class', fontsize=20)
sorted_food_dirs = sorted(os.listdir(root_dir))
for i in range(rows):
    for j in range(cols):
        try:
            food_dir = sorted_food_dirs[i*cols + j]
        except:
            break
        all_files = os.listdir(os.path.join(root_dir, food_dir))
        rand_img = np.random.choice(all_files)
        img = plt.imread(os.path.join(root_dir, food_dir, rand_img))
        ax[i][j].imshow(img)
        ec = (0, .6, .1)
        fc = (0, .7, .2)
        ax[i][j].text(0, -20, food_dir, size=10, rotation=0,
```

```
                ha="left", va="top",
                bbox=dict(boxstyle="round", ec=ec, fc=fc))
plt.setp(ax, xticks=[], yticks=[])
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```


Рисунок 4.2 – Приклад набору зображення [8]

```
# Setup multiprocessing pool
# Do this early, as once images are loaded into memory there will be Errno 12
# http://stackoverflow.com/questions/14749897/python-multiprocessing-memory-usage
import multiprocessing as mp

num_processes = 6
pool = mp.Pool(processes=num_processes)

We need maps from class to index and vice versa, for proper label encoding and
pretty printing.

class_to_ix = {}
ix_to_class = {}
with open('food-101/meta/classes.txt', 'r') as txt:
    classes = [l.strip() for l in txt.readlines()]
```

```
    class_to_ix = dict(zip(classes, range(len(classes))))
    ix_to_class = dict(zip(range(len(classes)), classes))
    class_to_ix = {v: k for k, v in ix_to_class.items()}
sorted_class_to_ix = collections.OrderedDict(sorted(class_to_ix.items()))
```

Набір даних Food-101 має структуру. Якщо хочемо використовувати це для того, щоб порівняти наші показники класифікації з іншими реалізаціями.

```
# Only split files if haven't already
if not os.path.isdir('./food-101/test') and not os.path.isdir('./food-101/train'):

    def copytree(src, dst, symlinks = False, ignore = None):
        if not os.path.exists(dst):
            os.makedirs(dst)
            shutil.copystat(src, dst)
        lst = os.listdir(src)
        if ignore:
            excl = ignore(src, lst)
            lst = [x for x in lst if x not in excl]
        for item in lst:
            s = os.path.join(src, item)
            d = os.path.join(dst, item)
            if symlinks and os.path.islink(s):
                if os.path.lexists(d):
                    os.remove(d)
                os.symlink(os.readlink(s), d)
    def ignore_train(d, filenames):
        print(d)
        subdir = d.split('/')[-1]
        to_ignore = train_dir_files[subdir]
        return to_ignore

    def ignore_test(d, filenames):
        print(d)
        subdir = d.split('/')[-1]
        to_ignore = test_dir_files[subdir]
        return to_ignore

    copytree('food-101/images', 'food-101/test', ignore=ignore_train)
    copytree('food-101/images', 'food-101/train', ignore=ignore_test)
```

```
else:
    print('Train/Test files already copied into separate folders.')
Train/Test files already copied into separate folders.
```

Тепер готові завантажити навчання і тестування зображень в пам'ять. Після того, як все буде завантажено, буде задіяно близько 80 ГБ пам'яті.

Будь-які зображення, ширина або довжина яких менша за розмір, буде змінювати розмір. Тому можемо приймати належний розмір збораження під час збільшення зображення min_size.

```
# Load dataset images and resize to meet minimum width and height pixel size
def load_images(root, min_side=299):
    all_imgs = []
    all_classes = []
    resize_count = 0
    invalid_count = 0
    for i, subdir in enumerate(listdir(root)):
        imgs = listdir(join(root, subdir))
        class_ix = class_to_ix[subdir]
        print(i, class_ix, subdir)
        for img_name in imgs:
            img_arr = img.imread(join(root, subdir, img_name))
            img_arr_rs = img_arr
            try:
                w, h, _ = img_arr.shape
                if w < min_side:
                    wpercent = (min_side/float(w))
                    hsize = int((float(h)*float(wpercent)))
                    #print('new dims:', min_side, hsize)
                    img_arr_rs = imresize(img_arr, (min_side, hsize))
                    resize_count += 1
                elif h < min_side:
                    hpercent = (min_side/float(h))
                    wsize = int((float(w)*float(hpercent)))
```

Інструменти візуалізації.

```
@interact(n=(0, len(X_train)))
def show_pic(n):
    plt.imshow(X_train[n])
    print('class:', y_train[n], ix_to_class[y_train[n]])
```

```
class: 21 chocolate_cake
```

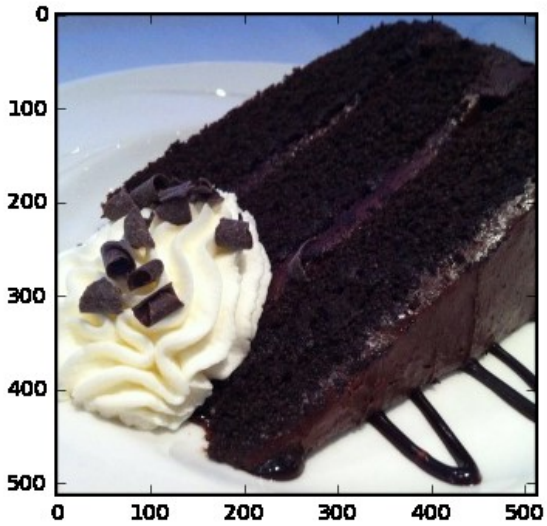


Рисунок 4.3 – Приклад візуалізованого зображення [8]

```
@interact(n=(0, len(X_test)))
def show_pic(n):
    plt.imshow(X_test[n])
    print('class:', y_test[n], ix_to_class[y_test[n]])
class: 21 chocolate_cake
```

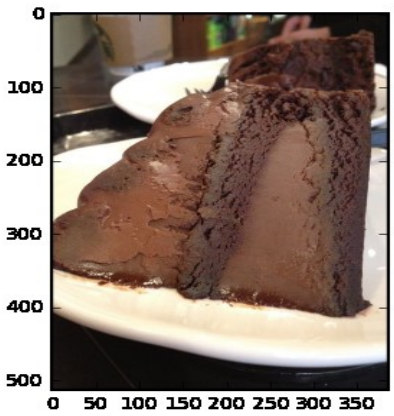


Рисунок 4.4 – Візуалізоване зображення [8]

```
    fig, axes = plt.subplots(nrows=nrows, ncols=ncols)
```

```
    fig.set_size_inches(12, 8)
    #fig.tight_layout()
    imgs = np.random.choice((y_train == n_class).nonzero()[0], nrows * ncols)
    for i, ax in enumerate(axes.flat):
        im = ax.imshow(X_train[imgs[i]])
        ax.set_axis_off()
        ax.title.set_visible(False)
        ax.xaxis.set_ticks([])
```

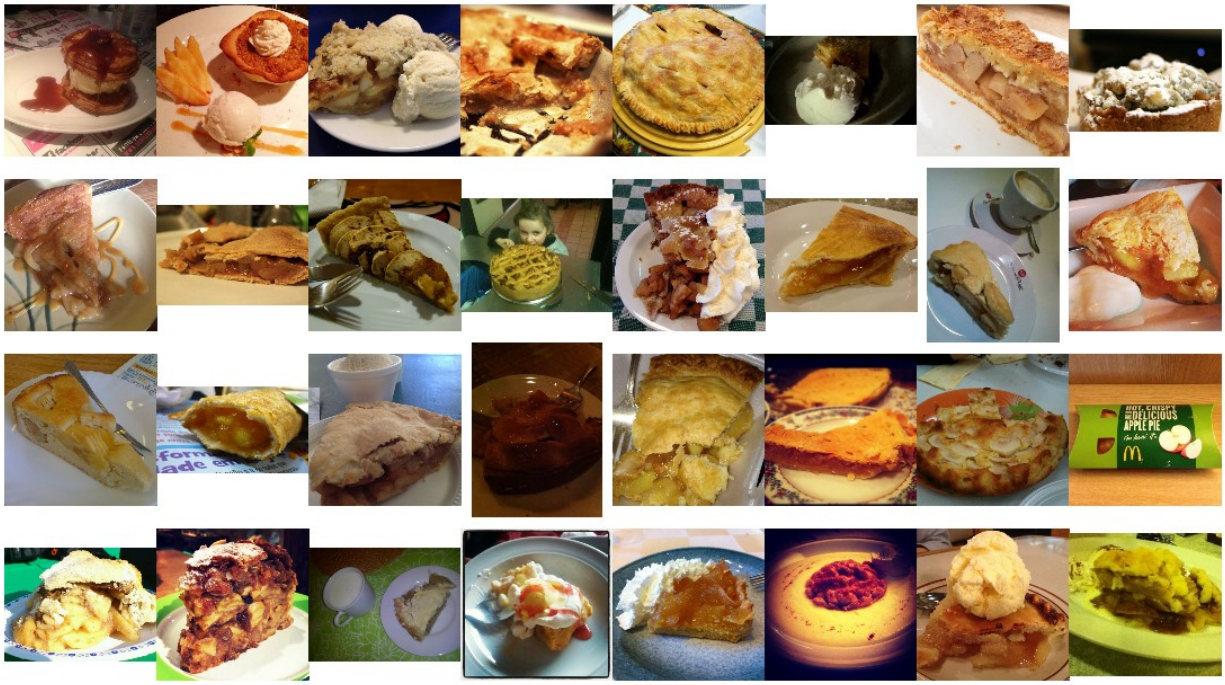

Рисунок 4.5 – Набір зображень [8]

```
def show_random_images_of_class(n_class=0):
    print(n_class)
    nrows = 4
    ncols = 8
    fig, axes = plt.subplots(nrows=nrows, ncols=ncols)
    fig.set_size_inches(12, 8)
    #fig.tight_layout()
    imgs = np.random.choice((y_test == n_class).nonzero()[0], nrows * ncols)
    for i, ax in enumerate(axes.flat):
        im = ax.imshow(X_test[imgs[i]])
        ax.set_axis_off()
        ax.title.set_visible(False)
        ax.xaxis.set_ticks([])
        ax.yaxis.set_ticks([])
```

```
        for spine in ax.spines.values():
            spine.set_visible(False)
    plt.subplots_adjust(left=0, wspace=0, hspace=0)
    plt.show()
```

### 4.3 Вектор двійкових функцій

Потрібно закодувати кожне значення етикетки, щоб створити вектор двійкових функцій, а не одну функцію, яка може взяти на себе значення n_classes.

```
from keras.utils.np_utils import to_categorical

n_classes = 101
y_train_cat = to_categorical(y_train, nb_classes=n_classes)
y_test_cat = to_categorical(y_test, nb_classes=n_classes)
from keras.applications.inception_v3 import InceptionV3
from keras.applications.inception_v3 import preprocess_input, decode_predictions
from keras.preprocessing import image
from keras.layers import Input

import tools.image_gen_extended as T
```

Потрібно мати більш потужний стек збільшення зображення, ніж той, який поставляється з Keras. Вдалося знайти цю модифіковану версію для використання в якості бази.

Додали додатковий pipeline, який дав можливість вказати додаткові зміни, такі як функції користувача, як обрізка і можливість використовувати препроцесор зображення Inception. Можливість застосовувати попередню проробку була динамічно необхідна, так як не було достатньо пам'яті, щоб зберегти всі навчальні набори. Змогли завантажити весь навчальний набір як float32suint8s.

Крім того, не повністю використовували ні графічний процесор, ні багатоявний процесор. За замовчуванням Python може використовувати лише одне ядро, тим самим обмежуючи кількість оброблених/доповнених зображень,

які можна відправити в графічному процесорі для навчання. Виходячи з деякого моніторингу ефективності в середньому використали лише невеликий відсоток ЦПУ. Включивши python, змогли отримати близько 50% використання процесора та використання графічного процесора 90% в multiprocessing pool.

Кінцевий результат полягає в тому, що кожна частина тренувань покращилась від 45 хвилин до 22 хвилин. Можна запустити графіки графічного процесора під час навчання в цьому блокноті.

На даний момент код має баги і вимагає перезапуску ядра Python кожного разу, коли навчання переривається вручну. Код досить складний разом, і деякі функції, такі як ті, які передбачають підготовку, відключені. Сподіваєся поліпшити цей ImageDataGenerator і випустити його в майбутньому.

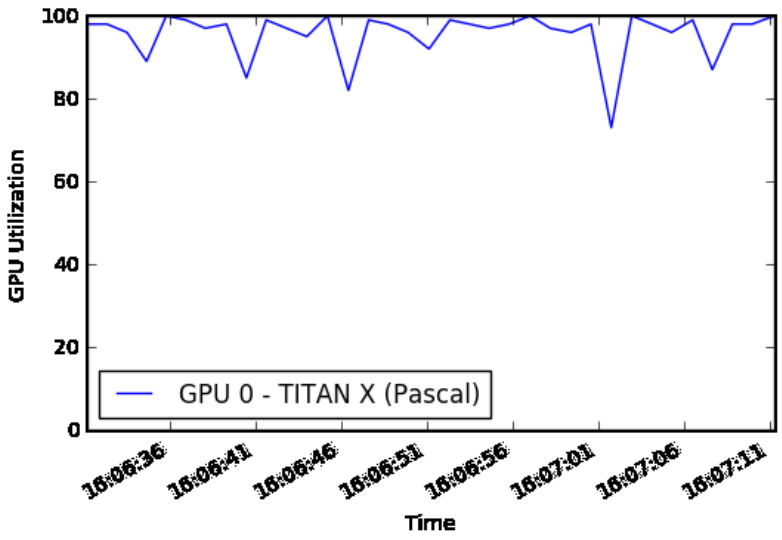


Рисунок 4.6 – Навантаження на процесор під час навчання [8]

```
# this is the augmentation configuration we will use for training
train_datagen = T.ImageDataGenerator(
    featurewise_center=False,  # set input mean to 0 over the dataset
    samplewise_center=False,  # set each sample mean to 0
    featurewise_std_normalization=False,  # divide inputs by std of the dataset
    samplewise_std_normalization=False,  # divide each input by its std
    zca_whitening=False,  # apply ZCA whitening
    rotation_range=0,  # randomly rotate images in the range (degrees, 0 to 180)
```

```
    width_shift_range=0.2,  # randomly shift images horizontally (fraction of
total width)
    height_shift_range=0.2,  # randomly shift images vertically (fraction of
total height)
    channel_shift_range=30,
    fill_mode='reflect')
train_datagen.config['random_crop_size'] = (299, 299)
train_datagen.set_pipeline([T.random_transform, T.random_crop,
T.preprocess_input])
train_generator = train_datagen.flow(X_train, y_train_cat, batch_size=64,
seed=11, pool=pool)
test_datagen = T.ImageDataGenerator()
test_datagen.config['random_crop_size'] = (299, 299)
test_datagen.set_pipeline([T.random_transform, T.random_crop,
T.preprocess_input])
test_generator = test_datagen.flow(X_test, y_test_cat, batch_size=64, seed=11,
pool=pool)
```

Бачимо, які зображення виходять з цих ImageDataGenerators:

```
def show_images(unprocess=True):
    for x in test_generator:
        fig, axes = plt.subplots(nrows=8, ncols=4)
        fig.set_size_inches(8, 8)
        page = 0
        page_size = 32
        start_i = page * page_size
        for i, ax in enumerate(axes.flat):
            img = x[0][i+start_i]
            if unprocess:
                im = ax.imshow( reverse_preprocess_input(img).astype('uint8') )
            else:
                im = ax.imshow(img)
            ax.set_axis_off()
            ax.title.set_visible(False)
            ax.xaxis.set_ticks([])
            ax.yaxis.set_ticks([])
            for spine in ax.spines.values():
```

Рисунок 4.7 – Зображення для навчання [8]

```
CPU times: user 1.54 s, sys: 524 ms, total: 2.06 s
Wall time: 2.24 s
%%time
show_images(unprocess=False)
```

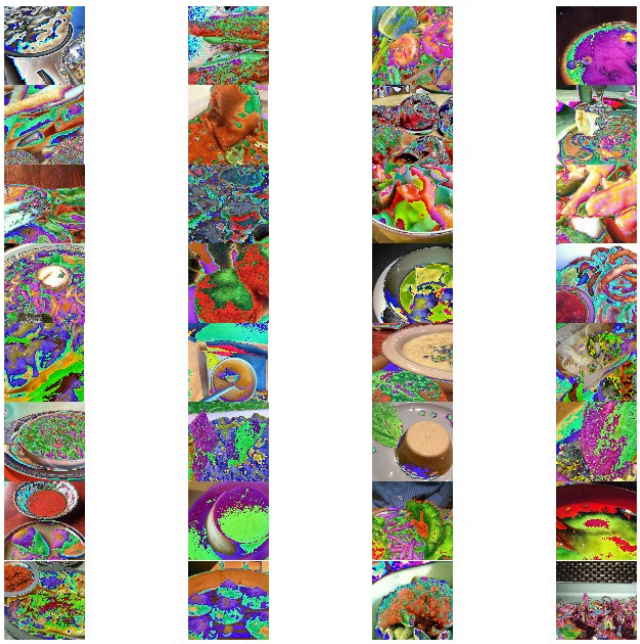

Рисунок 4.8 – Набір оброблених зображень [8]

### 4.4 Навчання системи

Будемо перекваліфікувати модель Google InceptionV3, попередньо підготовлену на ImageNet. Архітектура нейронної мережі показана нижче.

```
%%time
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D,
GlobalAveragePooling2D, AveragePooling2D
from keras.layers.normalization import BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, CSVLogger, LearningRateScheduler,
ReduceLROnPlateau
from keras.optimizers import SGD
from keras.regularizers import l2
import keras.backend as K
import math
```

На даний момент спостерігаємо до 81,65 Топ-1 точності на тестовому наборі. Можемо продовжувати тренувати модель з ще повільнішою швидкістю навчання, щоб побачити, чи поліпшується вона.

Початкові експерименти використовували більш сучасні оптимізатори, такі як Adam і AdaDelta, разом з більш високими показниками навчання. Певний час були нижче 80% точності, перш ніж вирішили слідувати літературі більш уважно і використовувати стохастичний градієнтний спуск (SGD) з швидким зниженням графіка навчання. Коли шукаємо через багатовизначну поверхню, іноді навчання відбувається повільніше та проходить довгий шлях.

Через деяку нестабільність з багатопроцесним кодом іноді потрібно перезапустити ноутбук, завантажити останню модель, потім продовжити навчання.

```
%%time
from keras.models import Sequential, Model, load_model
```

```
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D, ZeroPadding2D,
GlobalAveragePooling2D, AveragePooling2D
from keras.layers.normalization import BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, CSVLogger, LearningRateScheduler,
ReduceLROnPlateau
from keras.optimizers import SGD
from keras.regularizers import l2
import keras.backend as K
import math
```

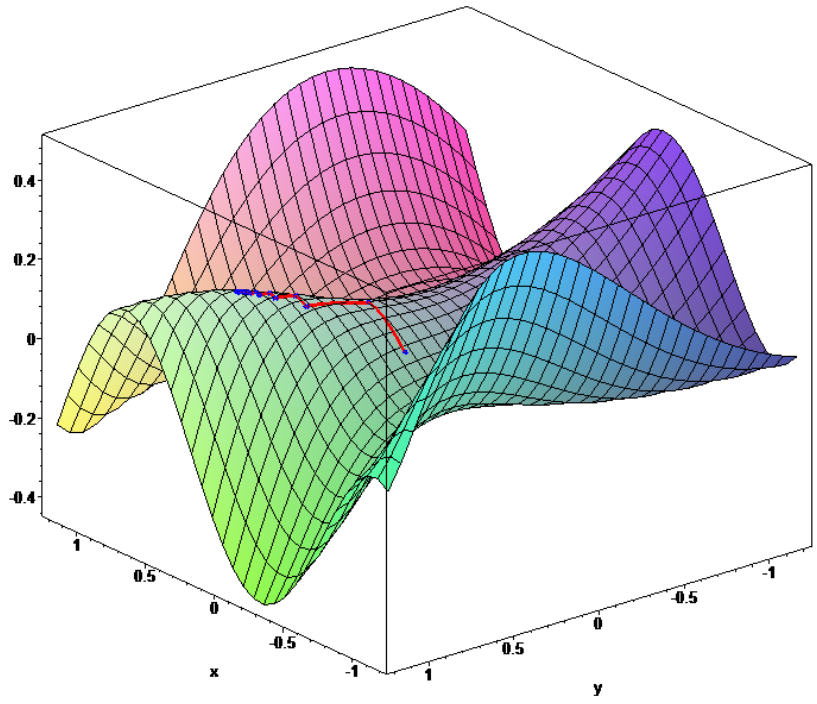


Рисунок 4.9 – Зображення градієнтного спуску [8]

```
model = load_model(filepath='./model4.29-0.69.hdf5')

opt = SGD(lr=.01, momentum=.9)
model.compile(optimizer=opt, loss='categorical_crossentropy',
metrics=['accuracy'])

checkpointer = ModelCheckpoint(filepath='model4b.{epoch:02d}-
{val_loss:.2f}.hdf5', verbose=1, save_best_only=True)
csv_logger = CSVLogger('model4b.log')
.0000032
```

```
lr_scheduler = LearningRateScheduler(schedule)

model.fit_generator(train_generator,
                    validation_data=test_generator,
                    nb_val_samples=X_test.shape[0],
                    samples_per_epoch=X_train.shape[0],
                    nb_epoch=32,
                    verbose=2,
                    callbacks=[lr_scheduler, csv_logger, checkpointer])
```

### 4.5 Оцінка моделі

На цей час ми повинні мати кілька навчених моделей, збережених на диску. Можемо пройти через них і використовувати функцію для завантаження моделі з найнижчою втратою / найвищою точністю.

```
#model = load_model(filepath='./model4.29-0.69.hdf5') # 86.8039 10-crop Top-1 test accuracy
model = load_model(filepath='./model4b.10-0.68.hdf5') # 86.9703
CPU times: user 36.4 s, sys: 1.11 s, total: 37.5 s
Wall time: 36.5 s
```

Також хочемо оцінити тестовий набір за допомогою декількох культур. Це може призвести до підвищення точності на 5% у порівнянні з оцінкою одного збору. Зазвичай використовуються такі напрями: верхній лівий, верхній правий, нижній лівий, нижній правий, центр. також беремо ті ж культури на зображенні, перевернутому зліва направо, створивши в цілому 10 культур.

Крім того, хочемо повернути прогнози top-N для кожного врожаю, щоб, наприклад, обчислити точність Top-5.

```
def center_crop(x, center_crop_size, **kwargs):
    centerw, centerh = x.shape[0]//2, x.shape[1]//2
    halfw, halfh = center_crop_size[0]//2, center_crop_size[1]//2
    return x[centerw-halfw:centerw+halfw+1,centerh-halfh:centerh+halfh+1, :]
def predict_10_crop(img, ix, top_n=5, plot=False, preprocess=True, debug=False):
```

```
    flipped_X = np.fliplr(img)
    crops = [
        img[:299,:299, :], # Upper Left
        img[:299, img.shape[1]-299:, :], # Upper Right
        img[img.shape[0]-299:, :299, :], # Lower Left
        img[img.shape[0]-299:, img.shape[1]-299:, :], # Lower Right
        center_crop(img, (299, 299)),

        flipped_X[:299,:299, :],
        flipped_X[:299, flipped_X.shape[1]-299:, :],
        flipped_X[flipped_X.shape[0]-299:, :299, :],
        flipped_X[flipped_X.shape[0]-299:, flipped_X.shape[1]-299:, :],
        center_crop(flipped_X, (299, 299))
    ]
        ax[1][4].imshow(crops[9])

    y_pred = model.predict(np.array(crops))
    preds = np.argmax(y_pred, axis=1)
    top_n_preds= np.argpartition(y_pred, -top_n)[:,-top_n:]
    if debug:
        print('Top-1 Predicted:', preds)
        print('Top-5 Predicted:', top_n_preds)
        print('True Label:', y_test[ix])
    return preds, top_n_preds
```

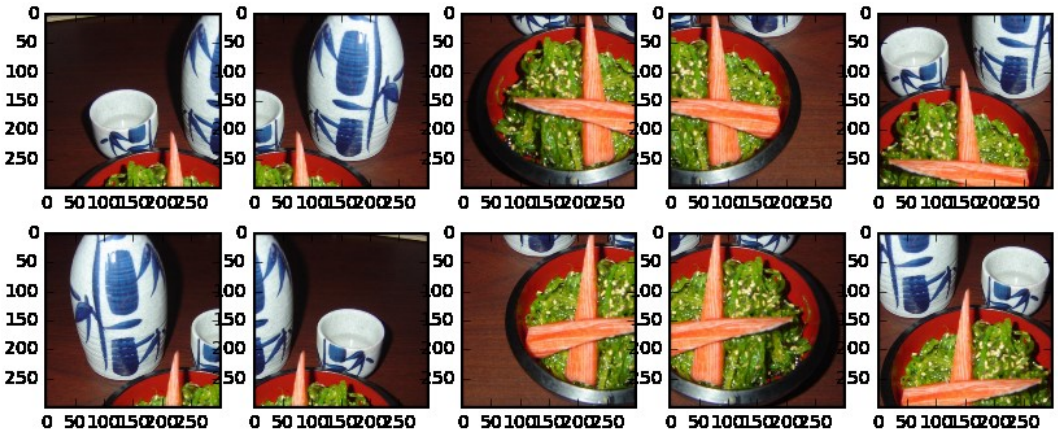

Рисунок 4.10 – Обробка зображення [8]

Також потрібно попередньо обробити зображення для моделі.

```
predict_10_crop(X_test[ix], ix, top_n=5, plot=True, preprocess=True, debug=True)
```

```
Top-1 Predicted: [51 51 88 88 88 51 51 88 88 88]
Top-5 Predicted: [[18 79 51 13 48]
 [48 79 11 55 51]
 [79 93 81 37 88]
 [51 86 93 81 88]
 [11 79 51 81 88]
 [19 79 51 56 13]
 [11 88 48 51 13]
 [37 93 86 88 81]
 [37 79 93 88 81]
 [84 81 11 79 88]]
True Label: 88
(array([51, 51, 88, 88, 88, 51, 51, 88, 88, 88]), array([[18, 79, 51, 13, 48],
        [48, 79, 11, 55, 51],
        [79, 93, 81, 37, 88],
        [51, 86, 93, 81, 88],
        [11, 79, 51, 81, 88],
```

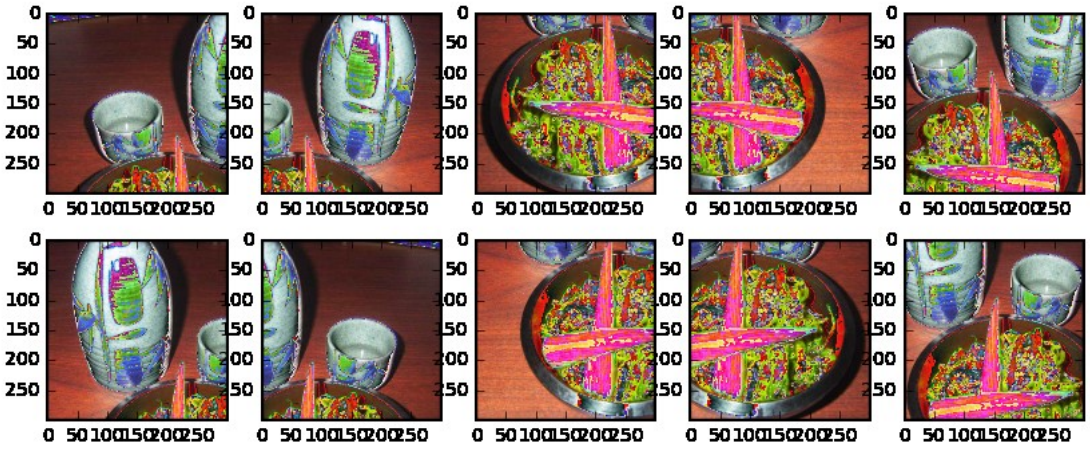

Рисунок 4.11 – Зображення після обробки [8]

Тепер створюємо посіви для кожного пункту в тестовому наборі і отримуємо прогнози. Це повільний процес на даний момент, оскільки ми не користуємося багатопроцесним або іншими видами паралелізму.

```
preds_10_crop = {}
for ix in range(len(X_test)):
    if ix % 1000 == 0:
        print(ix)
    preds_10_crop[ix] = predict_10_crop(X_test[ix], ix)
```

```
CPU times: user 50min 3s, sys: 5min 13s, total: 55min 16s
Wall time: 31min 28s
```

Тепер у нас є набір з 10 прогнозів для кожного зображення. Використовуючи гістограму, бачимо, як розподіляється # унікальних прогнозів для кожного зображення.

```
preds_uniq = {k: np.unique(v[0]) for k, v in preds_10_crop.items()}
preds_hist = np.array([len(x) for x in preds_uniq.values()])

plt.hist(preds_hist, bins=11)
plt.title('Number of unique predictions per image')
<matplotlib.text.Text at 0x7fe30c3daa20>
```

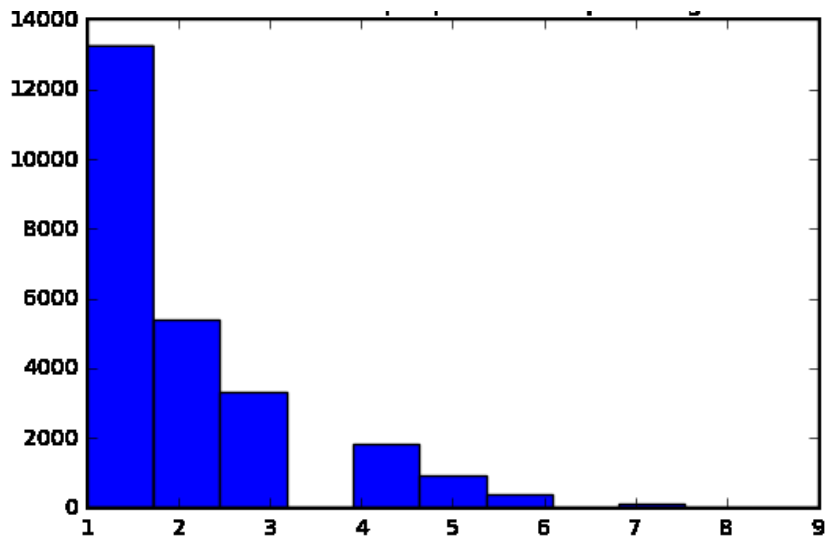


Рисунок 4.12 – Кількість унікальних передбачень по рисунку [8]

Створимо словник для співставлення індексу тестових елементів з його топ-1 / топ-5 прогнозів.

```
preds_top_1 = {k: collections.Counter(v[0]).most_common(1) for k, v in
preds_10_crop.items()}

top_5_per_ix = {k: collections.Counter(preds_10_crop[k][1].reshape(-
1)).most_common(5)
                for k, v in preds_10_crop.items()}
```

```
preds_top_5 = {k: [y[0] for y in v] for k, v in top_5_per_ix.items()}
%%time
right_counter = 0
for i in range(len(y_test)):
    guess, actual = preds_top_1[i][0][0], y_test[i]
    if guess == actual:
        right_counter += 1

print('Top-1 Accuracy, 10-Crop: {0:.2f}%'.format(right_counter / len(y_test) *
100))
Top-1 Accuracy, 10-Crop: 86.97%
CPU times: user 28 ms, sys: 0 ns, total: 28 ms
Wall time: 27.3 ms
%%time
top_5_counter = 0
for i in range(len(y_test)):
    guesses, actual = preds_top_5[i], y_test[i]
    if actual in guesses:
        top_5_counter += 1

print('Top-5 Accuracy, 10-Crop: {0:.2f}%'.format(top_5_counter / len(y_test) *
```


Рисунок 4.13 – Розпізнання зображення [8]

Матриця буде креслити кожну мітку класу і те, скільки разів вона була правильно позначена проти інших разів, коли вона була неправильно позначена як інший клас.

```
from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)
```

Подивимось, чи точність була послідовною у всіх класах, або чи було деякі класи набагато простіше / важче маркувати, ніж інші. За нашим сюжетом, кілька класів були аутсайдерами з точки зору того, що набагато складніше правильно маркувати.

```
corrects = collections.defaultdict(int)
incorrects = collections.defaultdict(int)
for (pred, actual) in zip(y_pred, y_test):
    if pred == actual:
        corrects[actual] += 1
    else:
        incorrects[actual] += 1

class_accuracies = {}
for ix in range(101):
    class_accuracies[ix] = corrects[ix]/250
```

```
plt.hist(list(class_accuracies.values()), bins=20)
plt.title('Accuracy by Class histogram')
```

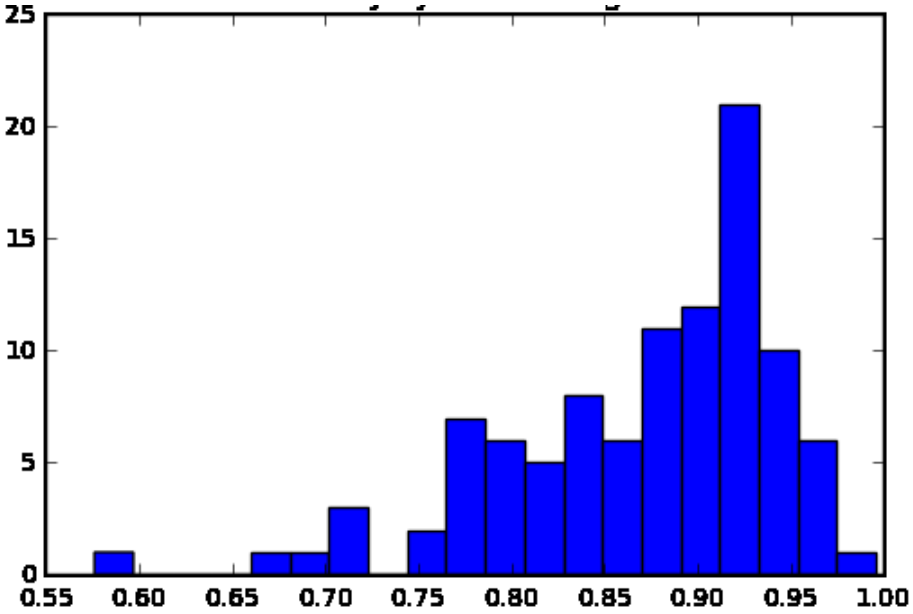


Рисунок 4.14 – Гістограмма точності за класами [8]

```
sorted_class_accuracies = sorted(class_accuracies.items(), key=lambda x: -x[1])
[(ix_to_class[c[0]], c[1]) for c in sorted_class_accuracies]
[('edamame', 0.996),
```

Інтерактивна класифікація Прогнозування з локального файла

```
pic_path = '/home/stratospark/Downloads/soup.jpg'
pic = img.imread(pic_path)
preds = predict_10_crop(np.array(pic), 0)[0]
best_pred = collections.Counter(preds).most_common(1)[0][0]
print(ix_to_class[best_pred])
plt.imshow(pic)
french_onion_soup
checkpointer = ModelCheckpoint(filepath='model4b.{epoch:02d}-
{val_loss:.2f}.hdf5', verbose=1, save_best_only=True)
csv_logger = CSVLogger('model4b.log')
corrects = collections.defaultdict(int)
incorrects = collections.defaultdict(int)
```


Рисунок 4.15 – Прогнозування із зображення в Інтернеті [8]

```
@interact
def predict_remote_image(url='http://themodelhouse.tv/wp-
content/uploads/2016/08/hummus.jpg'):
    with urllib.request.urlopen(url) as f:
        pic = plt.imread(f, format='jpg')
        preds = predict_10_crop(np.array(pic), 0)[0]
        best_pred = collections.Counter(preds).most_common(1)[0][0]
        print(ix_to_class[best_pred])
        plt.imshow(pic)
```

## Висновки до розділу 4

Розпізнавання продуктів харчування на основі зображень є складним завданням, яке в останнє десятиліття цікаве для наукових спільнот комп'ютерного зору та штучного інтелекту. Звичайно, що зображення може містити різноманітні інгредієнти, які можуть відрізнятися за формами і розмірами в залежності від регіональних традицій і звичок місцевих громад. Наприклад, сьогодні відомо у всьому світі, що салат з помідорами, оливами і цибулею відноситься до середземноморської дієти. Таким чином, знаходження відмінності серед грецького або ізраїльського салату, може бути важким завданням, навіть для людини. Те, як інгредієнти нарізані і поміщені на тарілці, може бути провідним

для відмінності. Однак пропонована система дає прийнятні результати та дає змогу розпізнати основні забраження, які типові для визначеної області.

## Загальні висновки

У будь-якому сучасному завданні машинного навчання найбільш дискусійним є створення нової моделі глибокого навчання (архітектура), або використання існуючої моделі через навчання шарів, або використання доступної платформи прогнозування вмісту на основі зображень (зазвичай через API). Розробка нової архітектури вимагає великих знань та інтуїції, тоді як передача навчання, в основному, вимагає конкретних знань і розробки додатків на платформі - це лише технічне рішення. З іншого боку, з точки зору продуктивності та легкості розгортання (з компромісом в ефективності), підходи ранжують у зворотному напрямку, при цьому платформи легко ранжують в першу чергу.

Це дослідження було зосереджено на особливо цікавій і складній проблемі розпізнавання продуктів харчування на основі зображень і опиралося на рішення машинного навчання, щоб виявити сильні і слабкі сторони кожного підходу і виділити можливості. Крім того, оскільки підходи до машинного навчання є необробленими даними, були представлені популярні великі набори даних для розпізнавання харчових продуктів і виявлено необхідність ще кращих наборів даних з більш конкретними та всесвітньо відомими категоріями продуктів харчування. У дослідженні наголошувалось на необхідності маркування харчових інгредієнтів, що дозволило б розширити можливість системи для досягнення кращих показників і розширення можливостей більш корисних реальних додатків (наприклад, в харчуванні та туризмі). В цілому, дослідження дає хороші результати в сфері розпізнавання продуктів харчування з підходами глибокого навчання, і пропонує ряд майбутніх напрямків, таких як:

– створення великомаштабних наборів даних про харчові продукти, з певним вмістом всесвітньо відомих категорій харчових даних;

– маркування наборів даних не тільки з типами продуктів харчування (класів), але і харчовими інгредієнтами (багатозначні класи);

– розробка гібридних методів для кращого результату, таких як комбінації нових моделей з платформами прогнозування;

– розробка та інтеграція харчових продуктів та інгредієнтів для надання більш точних прогнозів в цілому;

– розробити правила, засновані на контекстних факторах, які можуть призвести до розвитку аналогічних правил поділу харчових продуктів, заснованих на походженні, звичках, культурі тощо, можуть допомогти в цільових прогнозах на основі географічних місць, особистих уподобань тощо.

Отже, в деякій мірі, розв’язано проблему прогнозування та розпізнання продуктів харчування та інгредієнтів за їхніми зображеннями за допомогою моделі глибокого навчання.

Основні наукові та практичні результати доповідалися на XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп’ютерних наук АПКН-2020» (Хмельницький, 9-10 листопада 2020 року, Хмельницький національний університет). Тема доповіді: «Застосування штучного інтелекту для класифікації продуктів харчування».

За темою дипломної роботи магістра автором здійснена одна наукова публікація.

## Перелік посилань

1. Нейронные сети: полный курс. С. Хайкин. 2-е издание, Издательский дом Вильямс, 2008. - 1104 с.

2. Neural Networks: A Systematic Introduction. R. Rojas. Springer Science & Business Media. - 502 с.

3. Нейронные сети для обработки информации. С. Осовский. Пер. с польского И.Д. Рудинского. — М.: Финансы и статистика, 2002. – 344 с.

4. LeCun Y., Bengio Y., Hinton G. Deep Learning // Nature. 2015. Vol. 521. P. 436–444.

5. Ravı D., Wong Ch., Deligianni F., та інші Deep Learning for Health Informatics // IEEE Journal of Biomedical and Health Informatics. 2017. Vol. 21, No. 1. P. 4–21.

6. Schmidhuber J. Deep Learning in Neural Networks: an Overview // Neural Networks. 2015. Vol. 1. P. 85–117.

7. McCulloch W.S., Pitts W. A Logical Calculus of the Ideas Immanent in Nervous Activity // The Bulletin of Mathematical Biophysics. 1943. Vol. 5, No. 4. P. 115–133.

8. Hinton G., Salakhutdinov R. Reducing the Dimensionality of Data with Neural Networks // Science. 2006. Vol. 313, No. 5786. P. 504–507.

9. Hinton G.E., Osindero S., Teh Y.-W. A Fast Learning Algorithm for Deep Belief Nets // Neural Computing. 2006. Vol. 18, No. 7. P. 1527–1554.

10. S’ıma J. Loading Deep Networks Is Hard // Neural Computation. 1994. Vol. 6, No. 5. P. 842–850.

11. Windisch D. Loading Deep Networks Is Hard: The Pyramidal Case // Neural Computation. 2005. Vol. 17, No. 2. P. 487–502.

12. Gomez F.J., Schmidhuber J. Co-Evolving Recurrent Neurons Learn Deep Memory POMDPs // Proc. of the 2005 Conference on Genetic and Evolutionary Computation (GECCO) (Washington, DC, USA, June 25–29, 2005), 2005. P. 491–498.

13. Ciresan D.C., Meier U., Gambardella L.M., Schmidhuber J. Deep, Big, Simple Neural Nets for Handwritten Digit Recognition // Neural Computation. 2010. Vol. 22, No. 12. P. 3207–3220.

14. He K., Zhang X., Ren S., та інші Deep Residual Learning for Image Recognition // 2016 IEEE Conference on Computer Vision and Pattern Recognition (Las Vegas, NV, USA, 27–30 June 2016), 2016. P. 770–778.

15. P'erez-Ortiz J.A., Gers F.A., Eck D., та інші Kalman Filters Improve LSTM Network Performance in Problems Unsolvable by Traditional Recurrent Nets // Neural Networks. 2003. Vol. 16, No. 2. P. 241–250.

16. Weng J., Ahuja N., Huang T.S. Cresceptron: a Self-Organizing Neural Network Which Grows Adaptively // International Joint Conference on Neural Networks (IJCNN) (Baltimore, MD, USA, 7–11 June 1992). 1992. Vol. 1. P. 576–581.

17. Weng J.J., Ahuja N., Huang T.S. Learning Recognition and Segmentation Using the Cresceptron // International Journal of Computer Vision. 1997. Vol. 25, No. 2. P. 109–143.

18. Ranzato M.A., Huang F.J., Boureau Y.L., та інші Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition // IEEE Conference on Computer Vision and Pattern Recognition (Minneapolis, MN, USA, 17–22 June 2007), 2007. P. 1–8.

19. Scherer D., Muller A., Behnke S. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition // Lecture Notes in Computer Science. 2010. Vol. 6354, P. 92–101.

20. Smolensky P. Information Processing in Dynamical Systems: Foundations of Harmony Theory // Parallel Distributed Processing: Explorations in the Microstructure of Cognition. 1986. Vol. 1. P. 194–281.

21. Hinton G.E., Sejnowski T.E. Learning and Relearning in Boltzmann Machines // Parallel Distributed Processing. 1986. Vol. 1. P. 282–317.

22. Фасеткова система класифікації [Електронний ресурс]. – Режим доступу: https://lifeprog.ru/2_79230_fasetnaya-sistema-klassifikatsii.html

23. Введение в искусственные нейронные сети [Електронний ресурс]. Режим доступу: https://www.osp.ru/os/1997/04/179189/

24. В чем суть искусстенного интелекта [Електронний ресурс]. Режим доступу: https://polygant.net/ru/blog/v-chem-sut-iskusstvennogo-intellekta/

25. Recipe1M+: A Datadet for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images [Електронний ресурс]. Режим доступу: im2recipe.cdail.mit.edu

26. Горелик А. Л., Скрипкин В. А. Методы распознавания. — 4-е изд. — М.: Высшая школа, 1984, 2004. — 262 с.

27. Горбань А.Н. Обучение нейронных сетей. — М.: СССР-США СП «Параграф», 1990. — 160 с.

28. Recommender Systems Handbook / Ricci F., Rokach L., Shapira B., Kantor P.B. – Springer, 2011. – 875 p.

29. Сергієнко В.П. Комп’ютерні технології в тестуванні: навч. посіб. / В. П. Сергієнко, М. П. Малежик, Т. В. Сіткар. – Луцьк: «Волиньполіграф», 2012. – 290 с.

30. Архіпов В. В. Організація ресторанного господарства: навч. посіб. / В. В. Архіпов. - К. : Центр учбової літератури,Фірма «Інкос», 2007. - 335 с.

31. Гройлов, А. С. Информационные технологии в сфере общественного питания / А. С. Гройлов, Е. М. Аверина, А. С. Бугаенко, И. А. Винокурова. // Молодой ученый. — 2011. — № 3 (26). — Т. 1. — С. 100-102.

32. Корнєєв В.В., Гарєєв А.Ф., Васютін С.В., Райх В.В. Бази даних. Інтелектуальна обробка інформації // Нолідж. – 2016. – С. 97–127.

33. Засоби штучного інтелекту: навч. посіб. / Р. О. Ткаченко, Н. О. Кустра, О. М. Павлюк, У. В. Поліщук ; М-во освіти і науки України, Нац. ун-т «Львів. політехніка». — Львів: Вид-во Львів. політехніки, 2014. — 204 с.

34. Системи штучного інтелекту: навч. посіб. / Ю. В. Нікольський, В. В. Пасічник, Ю. М. Щербина ; за наук. ред. В. В. Пасічника ; М-во освіти і науки,

молоді та спорту України. — 2-ге вид., виправл. та доповн. — Львів: Магнолія-2006, 2013. — 279 с.

35. Системи штучного інтелекту: нечітка логіка, нейронні мережі, нечіткі нейронні мережі, генетичний алгоритм : монографія / В. П. Лисенко, В. М. Решетюк, В. М. Штепа, Н. А. Заєць, В. О. Мірошник, А. О. Дудник. - Київ : НУБіП України, 2014. - 332 с.

36. Класифікація продуктів харчування [Електронний ресурс]. Режим доступу: https://awesomeopensource.com/project/stratospark/food-101-keras

37. Методи класифікації продуктів харчування [Електронний ресурс]. Режим доступу: https://www.cse.scu.edu/~mwang2/projects/ML_foodRecognition_18s.pdf

38. Інтелектуальні технології в управлінні [Електронний ресурс]. Режим доступу: https://uadoc.zavantag.com/text/9284/index-5.html

39. Комп'ютерні технології. Нейрокомп`ютери [Електронний ресурс]. Режим доступу: https://ua-referat.com/Нейрокомп`ютери

40. Штучні нейронні мережі [Електронний ресурс]. Режим доступу: http://ukrefs.com.ua/print:page,1,170581-Iskusstvennye-neiy-ronnye-seti.html

41. Навчання штучних нейронних мереж [Електронний ресурс]. Режим доступу: https://opticstoday.com/katalog-statej/stati-na-uk-rainskom/nejromerezhi/navchannya-shtuchnix-nejronnix-merezh

42. Нейронные сети: распознавание образов и изображений с помощью ИИ [Електронний ресурс]. Режим доступу: center2m.ru/ai-recognition

43. Застосування нейронних мереж [Електронний ресурс]. Режим доступу: https://academyfx.ru/article/blogi/2436-nejronnye-seti-v-trejdinge-na-foreks

44. Топ фреймворки для веб-разработки [Електронний ресурс] // JETRUBY BLOG. – 2018. – Режим доступу: https://jetruby.com/ru/blog/top-freimworki-2018/

# Додатки

# Додаток А

## Фрагмент програмного коду

```
pic_path = '/home/stratospark/Downloads/soup.jpg'
pic = img.imread(pic_path)
preds = predict_10_crop(np.array(pic), 0)[0]
best_pred = collections.Counter(preds).most_common(1)[0][0]
print(ix_to_class[best_pred])
plt.imshow(pic)
french_onion_soup
checkpointer = ModelCheckpoint(filepath='model4b.{epoch:02d}-
{val_loss:.2f}.hdf5', verbose=1, save_best_only=True)
csv_logger = CSVLogger('model4b.log')
corrects = collections.defaultdict(int)
incorrects = collections.defaultdict(int)
root_dir = 'food-101/images/'
rows = 17
cols = 6
fig, ax = plt.subplots(rows, cols, frameon=False, figsize=(15, 25))
fig.suptitle('Random Image from Each Food Class', fontsize=20)
sorted_food_dirs = sorted(os.listdir(root_dir))
for i in range(rows):
    for j in range(cols):
        try:
            food_dir = sorted_food_dirs[i*cols + j]
        except:
            break
        all_files = os.listdir(os.path.join(root_dir, food_dir))
        rand_img = np.random.choice(all_files)
        img = plt.imread(os.path.join(root_dir, food_dir, rand_img))
        ax[i][j].imshow(img)
        ec = (0, .6, .1)
        fc = (0, .7, .2)
        ax[i][j].text(0, -20, food_dir, size=10, rotation=0,
                ha="left", va="top",
                bbox=dict(boxstyle="round", ec=ec, fc=fc))
plt.setp(ax, xticks=[], yticks=[])
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```

```
preds_top_5 = {k: [y[0] for y in v] for k, v in top_5_per_ix.items()}
%%time
right_counter = 0
for i in range(len(y_test)):
    guess, actual = preds_top_1[i][0][0], y_test[i]
    if guess == actual:
        right_counter += 1

print('Top-1 Accuracy, 10-Crop: {0:.2f}%'.format(right_counter / len(y_test) *
100))
Top-1 Accuracy, 10-Crop: 86.97%
CPU times: user 28 ms, sys: 0 ns, total: 28 ms
Wall time: 27.3 ms
%%time
top_5_counter = 0
for i in range(len(y_test)):
    guesses, actual = preds_top_5[i], y_test[i]
    if actual in guesses:
        top_5_counter += 1
```

**Додаток Б**

**Ксерокопії наукових публікацій, виконаних при роботі над дипломною роботою магістра**

*(ксерокопії титульної сторінки, сторінки змісту та всіх сторінок із публікацією)*

Перелік наукових публікацій:

1. Шевченко А. О. Застосування штучного інтелекту для класифікації продуктів харчування / А.О. Шевченко, В.Ц. Міхалевський // Збірник наукових праць за матеріалами XII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук - 2020». - Хмельницький, 2020. – С.357-358.

Міністерство освіти і науки України
Хмельницький національний університет


# ЗБІРНИК НАУКОВИХ ПРАЦЬ

за матеріалами XII всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2020»

*9-10 листопада 2020*

Хмельницький 2020

***Хома Д. М., Цюрпіта Ю. С., Медзатий Д. М.***
Дослідження метрологічних характеристик технічного автоматизованого засобу інформаційно-вимірювальної системи вологості паперу ..... 323

***Хомяк Б. В., Драч І. В.***
Розрахунок параметрів рідинних автобалансувальних пристроїв ..... 328

***Цимбал О. В., Корнєв В. П.***
Електронний блок аналізу для металошука ..... 333

***Чугай О. М., Шпичко А. В., Мазурець О. В.***
Інформаційна модель кіберспортивної команди для автоматизованого формування складу команд ..... 339

***Шагін В. Ю., Ковальчук Д. В., Каштальян А. С.***
Централізована розподілена система виявлення атак в корпоративних комп'ютерних мережах на основі мультифрактального аналізу ..... 345

***Шаповалова А. С., Райко Г. О.***
Застосування інформаційних технологій у сфері страхування ..... 348

***Шевцов О. О., Савенко О. С.***
Розподілена система виявлення зловмисного програмного забезпечення в локальних мережах на основі Баєсовської мережі ..... 351

***Шевцова А. В., Кисіль Т. М.***
Баєсовська мережа і система виявлення зловмисного програмного забезпечення на основі дослідження аномалій ..... 354

***Шевченко А. О., Міхалевський В. Ц.***
Застосування штучного інтелекту для класифікації продуктів харчування ..... 357

***Шевчук О. О.***
Мобільний додаток для вибору кольору ниток для вишивання хрестиком ..... 359

***Шпак О. О., Богданов А. Р., Сова О. Я.***
Модель системи логування подій у мережевій інфраструктурі на основі стеку ELK+KAFKA ..... 362

УДК 004.4

Шевченко А. О., Міхалевський В. Ц.

*Хмельницький національний університет*

# ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ КЛАСИФІКАЦІЇ ПРОДУКТІВ ХАРЧУВАННЯ

*Проведено дослідження з пошуку методів та аналізу підходів класифікації зображень з використання нейронних мереж.*

*Applied aspects of information system development for classification and retrieval of semantic constituent text and visualization of the obtained results for further analysis are considered, which provides the most accurate transmission of the content of the input text. The offered information system provides accurate and fast classification of the text according to the given categories.*

Автоматичне розпізнавання продуктів харчування на основі зображень є особливо складним завданням. Традиційні підходи до аналізу зображень досягали низької точності класифікації в минулому, тоді як підходи глибокого навчання дозволили ідентифікувати типи продуктів харчування та їх інгредієнти. Вміст харчових страв, як правило, є деформованими предметами, як правило, включаючи складну семантику, що робить завдання визначення їх структури дуже складним. Методи глибокого навчання вже показали дуже перспективні результати в таких викликах, тому наше дослідження присвячено презентації деяких популярних підходів і методів, що застосовуються для розпізнавання продуктів харчування на основі зображень. Три основні напрямки рішень, а саме дизайн з нуля, навчання передачі та підходи на платформі, окреслені, особливо для поставленого завдання, і перевіряються та порівнюються, щоб виявити притаманні сильні та слабкі сторони.

Розпізнавання продуктів харчування на основі зображень є складним завданням, яке цікаве для наукових спільнот комп'ютерного зору та штучного інтелекту в останнє десятиліття. Звичайно, що блюдо з їжі може містити різноманітні інгредієнти, які можуть відрізнятися за формами і розмірами в залежності від регіональних традицій і звичок місцевих громад. Наприклад, сьогодні відомо по всьому світу, що салат з помідорами, оливами і цибулею відноситься до середземноморської дієти. Таким чином, знаходження відмінності серед грецького або ізраїльського салату може бути важким завданням навіть для людини. Те, як інгредієнти нарізані і поміщені на тарілку, може бути провідним для відмінності. Однак пропонована система дає прийнятні результати та дає змогу розпізнати основні зображення, які типові для визначеної області.

Розпізнавання образів і класифікація. Суть полягає в розпізнаванні вхідного образу (символів тексту, зображення, мовних сигналів і т.д.) і вказанні його приналежності до певного класу. Під час навчання нейронної мережі на вхід їй подають вектор значень ознак образу із вказанням його класу. Після навчання їй можна пред'являти невідомі раніше образи й одержувати відповідь про приналежність до певного класу. У такого роду задачах встановлюється відповідність між вихідним шаром штучної нейронної мережі і класом, який він представляє.

Здатність нейронних мереж до пророкування результатів обумовлена узагальненням і виділенням загальних залежностей між вхідними й вихідними даними. Після навчання мережа здатна передбачити майбутнє значення деякої послідовності, ґрунтуючись на її попередніх змінах. Для того, щоб штучна нейронна мережа могла дати максимально точну відповідь, необхідно навчати її на даних, у яких дійсно прослідковується якась залежність. А якщо ні, то прогнозування, найімовірніше, не дасть ніяких результатів.

Входи з'єднані із гніздом нейрона S за допомогою синаптичних зв'язків. Кожний синапс має свою вагу, при передачі в гніздо нейрона вхідного параметра він відповідно множиться на вагу, тобто $x_i * w_i$. Стан нейрона $S$ визначається формулою:

$$S = \sum_{i=1}^{N} x_i * w_i \qquad (1)$$

Вихід нейрона є функція його стану $y=f(S)$.

Початкові експерименти використовували більш сучасні оптимізатори, такі як Adam і AdaDelta, разом з більш високими показниками навчання. Певний час були нижче 80% точності, перш ніж вирішили слідувати літературі більш уважно і використовувати стохастичний градієнтний спуск (SGD) з швидким зниженням графіку навчання. Коли шукаємо через багатовизначну поверхню, іноді навчання відбувається повільніше та проходить довгий шлях.

Через деяку нестабільність з багатопроцесним кодом іноді потрібно перезапустити ноутбук, завантажити останню модель, потім продовжити навчання.

**Перелік посилань**

1. Ciresan D.C., Meier U., Gambardella L.M., Schmidhuber J. Deep, Big, Simple Neural Nets for Handwritten Digit Recognition // Neural Computation. 2010. Vol. 22, No. 12. P. 3207–3220.
2. He K., Zhang X., Ren S., та інші Deep Residual Learning for Image Recognition // 2016 IEEE Conference on Computer Vision and Pattern Recognition (Las Vegas, NV, USA, 27–30 June 2016), 2016. P. 770–778.
3. P'erez-Ortiz J.A., Gers F.A., Eck D., та інші Kalman Filters Improve LSTM Network Performance in Problems Unsolvable by Traditional Recurrent Nets // Neural Networks. 2003. Vol. 16, No. 2. P. 241–250.

Mon Nov 30 11:59:33 EET 2020, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

# Anti-Plagiarism v-15.257

**Максимальне співпадіння з одним документом 0.0%**

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

| ID: 81658<br>Назва: Застосування штучного інтелекту для класифікації продуктів харчування<br>Додано в БД: 2020-11-30<br>Автора: Шевченко Артем Олександрович<br>Керівники: Міхалевський В.Ц.<br>Консультанти:<br>Опоненти: | Документ | | Сумарний збіг по Базі Даних | |
|---|---|---|---|---|
| | Символи | Лексеми | Символи | Лексеми |
| | 80615 | 610 | 1010 (1%) | 13 (2%) |

Джерело плагіату

| ID | Опис | Наявність плагіату в документі | |
|---|---|---|---|
| | | Символи | Лексеми |

РІШЕННЯ КАФЕДРИ **КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: **Застосування штучного інтелекту для класифікації продуктів харчування**
Автор: **Шевченко А.О.**
Спеціальність: **122 Комп'ютерні науки**
Науковий керівник: **к.ф.-м.н., доц. Міхалевський В.Ц.**

Після аналізу звіту подібності зроблено такий висновок:

| № | Висновок | Позначка про відповідність |
|---|---|---|
| 1 | Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту. | **відповідає** |
| 2 | Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи | - |
| 3 | Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту. | - |
| 4 | Інше: | - |

Підтвердження: Показник збігів у межах допустимого – 21,5%. Виявленні запозичення не є плагіатом, т.я. розміщенні в розділах, які не описують безпосередньо авторське дослідження, та мають посилання на приведений список літературних джерел. (*Найбільша схожість 17.6% з інтернет-джерелом https://awesomeopensource.com/project/stratospark/food-101-keras виправдана тим, що джерело є відкритим ресурсом загальноприйнятої класифікації продуктів харчування. Схожість 9.41% з інтернет-джерелом https://www.cse.scu.edu/~mwang2/projects/ML_foodRecognition_18s.pdf виправдана тим, що студент скористався відомими методами, алгоритмами та їх кодами, так як не було необхідності розробляти їх самому. Схожість 1.2% з інтернет-джерелом https://uadoc.zavantag.com/text/9284/index-5.html виправдана тим, що студент скористався відомими теоретичними розробками з напрямку інтелектуальних технологій в управлінні. Схожість 1.12% з інтернет-джерелом https://ua-referat.com/Нейрокомп`ютери виправдана тим, що студент скористався відомими теоретичними матеріалами з комп'ютерних технологій. Схожість 1.07% з інтернет-джерелом http://ukrefs.com.ua/print:page,1,170581-Iskusstvennye-neiyronnye-seti.html ви-правдана тим, що студент скористався відомими теоретичними матеріалами про штучні нейрон-ні мережі. Схожість 1.05% з інтернет-джерелом https://opticstoday.com/katalog-statej/stati-na-uk-rainskom/nejromerezhi/navchannya-shtuchnix-nejronnix-merezh виправдана тим, що студент скорис-тався відомими теоретичними розробками з напрямку навчання штучних нейронних мереж.)*

Таким чином, робота виконана самостійно з використанням відкритих ресурсів та джерел, а виявлені співпадіння мають або випадковий характери або зумовлені необхідністю використати загальнодоступні ресурси технологічного та програмного змісту як такі, де представлені загальновідомі методи, алгоритми і коди, що можуть використовуватися для побудови самостійних технологій.

Робота приймається до захисту.

02.12.2020 _____ Підпис керівника
Дата

_____ Підпис завідувача кафедри

Хмельницький національний університет

Кафедра комп'ютерних наук та інформаційних технологій

**ВІДГУК ОПОНЕНТА**

**на дипломну роботу магістра**

Магістра *гр. КНм-19-1 Шевченка Артема Олександровича*

На тему: *Застосування штучного інтелекту для класифікації продуктів харчування.*

1. Актуальність і значення теми

*Сучасні інформаційні технології та системи штучного інтелекту ще рідко впроваджуються в практичних задачах класифікації та сортування продуктів харчування на складах великих підприємств, а також в середовищі користувачів магазинів для швидкої оплати товару. Штучні нейронні мережі, як напрямок наукового дослідження, в даний час активно розбудовуються, бо працюють за принципом біологічних нейронних мереж. Тому необхідно на практиці використовувати такі технологій для розпізнавання об'єктів, аналізу даних, оптимізації та прийняття рішень, зокрема, в предметній області зображень продуктів для послуг харчування.*

2.Оцінка якості та достовірності проведених досліджень.

*Отримані результати добре співвідносяться з результатами, наведеними в наукових роботах і довідниках.*

3. Оцінка запропонованих заходів та пропозицій, практичної цінності та ефективності.

*Проведені дослідження представляють науково-технічну цінність, є ефективним дослідженням у сфері послуг харчування, їх можна використати з метою підвищення ефективності класифікації продуктів за зображеннями.*

4. Загальний висновок та оцінка

*Робота виконана в повному обсязі. Досліджені та проаналізовані дані за допомогою комплексу входять в рамки допустимих відхилень. Пояснювальна записка оформлена у відповідності з нормами. Незначні недоліки не знижують цінності дипломної роботи. За своєю структурою, практичними результатами, поставленою метою та вирішеними задачами робота відповідає вимогам вищої школи і вимогам, що пред'являються до освітньо-кваліфікаційного рівня «магістр», а її автор Шевченко А.О. заслуговує присвоєння кваліфікації магістра з комп'ютерних наук та інформаційних технологій.*

Робота заслуговує на оцінку « задовільно ».

Опонент Дика О.В., д.т.н., проф., зав.каф. ТАМХНУ