НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 339.72.015

До захисту допущено
В. о. завідувача кафедри ММСА
О.Л.Тимощук
«___» _____ 2018 р.

Магістерська дисертація

на здобуття ступеня магістра за спеціальністю 124 Системний аналіз
на тему: «Порівняння якості методів розпізнавання емоцій з відеопотоку»

Виконав:
студентка ІІ курсу, групи КА-71 мп
Матвіїв Катерина Юріївна _____

Керівник: доцент кафедри ММСА,
к.т.н., доц., Тимошенко Ю.О. _____

Рецензент:
доцент кафедри прикладної математики
НТУУ КПІ ім. Сікорського,
к.т.н., доц., Маслянко П.П. _____

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів
без відповідних посилань
Студент _____

Київ 2018

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність (спеціалізація) — 124 «Системний аналіз» («Системний аналіз і управління»)

ЗАТВЕРДЖУЮ
В. о. завідувача кафедри ММСА
О. Л. Тимощук
«___» _____ 2018 р.

ЗАВДАННЯ

на магістерську дисертацію студентці Матвіїв Катерині Юріївній

1. Тема дисертації: «Порівняння якості методів розпізнавання емоцій з відеопотоку», науковий керівник дисертації Тимошенко Юрій Олександрович, к.т.н., доцент, затверджені наказом по університету від «07» листопада 2018 р. № 4121-с

2. Термін подання студентом дисертації: _____

3. Об’єкт дослідження: емоції людини із відеопотоку

4. Предмет дослідження: порівняння якості методів розпізнавання емоцій людини

5. Перелік завдань, які потрібно розробити:

1) проаналізувати актуальність, визначити предмет та об’єкт дослідження;
2) вибрати методи побудови нейронних мереж;
3) розробити моделі розпізнавання емоцій на основі нейронних мереж;
4) проаналізувати їх якість, вибрати найращі архітектури;
5) розробити стартап-проект.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:

1) рисунки для обгрунтування математичної бази нейронних мереж;
2) рисунки матриці помилок для моделей;
3) графіки значення помилки, точності та повноти моделей в залежності від кількості епох;
4) рисунки архітектур найкращих моделей.

7. Орієнтовний перелік публікацій:

1. Ніколаєв С.С. Залежність якості детектора облич на ознаках Хаара від варіативності навчальної вибірки / С.С. Ніколаєв, Ю.О. Тимошенко, К.Ю. Матвіїв // KPI Sci. NEWS. – 2017. – № 6. – С. 38–46.

8. Дата видачі завдання: _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
	Вибір та узгодження теми магістерської дисертації, формування вступу роботи	07.09.2018 – 14.09.2018	
	Визначення актуальності, предмета та об’єкта дослідження, формування першого розділу	14.09.2018 – 21.09.2018	
	Дослідження математичної бази нейронних мереж, формування теоретичного розділу	21.09.2018 – 30.09.2018	
	Розробка та навчання моделей розпізнавання емоцій на базі згорткових нейронних мереж	01.10.2018 – 11.10.2018	
	Розробка та навчання моделей розпізнавання емоцій на базі нейронних мереж, що використовують ключові точки	11.10.2018 – 21.10.2018	
	Оцінка якості побудованих моделей та формування практичного розділу роботи	21.10.2018 – 31.10.2018	
	Розробка стартап-проект роботи та оформлення переліку посилань	01.11.2018 – 14.11.2018	
	Остаточне оформлення роботи відповідно до ГОСТу	14.11.2018 – 26.11.2018	

Студент К.Ю.Матвіїв

Науковий керівник дисертації Ю.О.Тимошенко

РЕФЕРАТ

Магістерська дисертація: 100 с., 40 рис., 23 табл., 3 додатки, 67 джерел.

Тема: «Порівняння якості методів розпізнавання емоцій з відеопотоку»

Об’єктом дослідження даної роботи є емоції з відеопотоку.

Предмет досідження – якість методів розпізнавання емоцій.

Мета роботи – побудова моделей розпізнавання емоцій людини на основі відеопотоку та порівняння їх якості.

Теоретичною та методологічною основою дослідження є зарубіжні роботи у галузі машинного навчання.

В роботі розглянуто задачу розпізнавання емоцій, її актуальність. Здійснено побудову моделей розпізнавання емоцій обличчя людини на основі згорткових нейронних мереж та нейронних мереж на основі ключових точок обличчя. Також було визначено якість та швидкість побудованих моделей та здійснено їх порівняння. Таким чином було отримано моделі високої якості точність, яких становить 0.7 – 0.9.

Побудовані моделі можуть бути використані у маркетингових, рекламних та соціальних аспектах життя людини. У даній роботі пропонується використання даних моделей для побудови продукту для роботи з аудиторією та оцінено перспективи реалізації такого продукту.

ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, МАТРИЦЯ ПОМИЛОК, РОЗПІЗНАВАННЯ ЕМОЦІЙ З ВІДЕО, TENSORFLOW, KERAS.

ABSTRACT

The master thesis: 100 p., 40 p., 23 tabl., 3 appendixes, 67 references.

Work name: «Emotions Recognition from Video Stream Methods Quality Comparison»

Object of the work is human emotions.

Subject of work is emotion recognition methods quality.

The purpose of work is to implement models for human emotion recognition based on video flow and their quality comparison.

Theoretical and methodological basis of study are works of foreign scholars in the field of machine learning.

The work discovers the emotion recognition problem, its relevance. The work involves building models for this purpose using neural network with different architectures and comparing their quality. There are different models based on neural network with face landmarks or based on convolutional neural network. Moreover, quality and working time have beed defined in this work. Consequently, we have got high quality models with 0.7 – 0.9 precision value.

These models can be used in market, advertisement and social purposes. This work suggests using these model for building start-up project devoted to designing a product for emotion recognition of audience.

CONVOLUTIONAL NEURAL NETWORK, CONFUSION MATRIX, EMOTION RECOGNITION USING VIDEO STREAM, TENSORFLOW, KERAS.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

CNN – Convolutional Neural Network

PCA – Principal Component Analysis

NN – Neural Network

KDL – Kullback-Leibler Distance

LDC – Linear Discriminant Classifier

FACS – Facial ActionCoding System

HOC – Higher Order Crossing

AU – Action Units

ВСТУП

Одними з актуальних задач сьогодення є задачі комп’ютерного зору, які покликані створити машину, що була б схожа на людину, автоматизувувала її діяльність. Прикладом такої задачі є розпізнавання емоцій людини.

Дана задача є актуальною з огляду на те, що вона знаходить своє застосування у комерційних та соціальних аспектах людського життя. Інтерес до цього питання виникнув спершу у нейробіологів, яких цікавив зв’язок емоцій із нервовими процесами людинами. Найбільш значущою роботою для розпізнавання емоцій стала робота Екмана та Фрізана [1], які доводили зв’язок емоцій людини з певним набором скорочень м’язів, що проявляються на обличчі у вигляді міміки, та відповідно є візуальною складовою обличчя.

На сьогоднішній день існує чимало методів для розпізнавання емоцій. Одним з найпопулярніших методів є нейронні мережі та, зокрема, згорткові нейронні мережі. Завдяки тому, що варіантів, як саме побудувати ту чи іншу нейронну мережу, надзвичайно багато, дослідники продовжують працювати у даній сфері, отримуючи все нові і нові результати. З огляду на актуальність та широкий спектр можливостей, саме нейронні мережі було обрано як засіб для розв’язання поставлених задач.

Таким чином об’єктом даної роботи є визначення емоцій людини з відеопотоку, а предметом - порівняння якості методів розпізнавання емоцій. Метою роботи є побудови оптимальної моделі. Більш детально опис об’єкту та актуальність поставленої задачі розглянуто у першому розділі даної роботи.

У другому розділі проведено огляд та аналіз математичної основи побудови нейронних мереж, а також додаткових методів, необхідних для покращення її навчання та роботи в цілому.

Третій розділ присвячено процесу побудови моделей. Наведено основі етапи, а саме: збір даних, побудова моделей, їх тестування. Розробка моделей здійснювалась за допомогою бібліотеки TensorFlow та її розширення Keras із використанням мови програмування Python у середовищі розробки Visual Studio Code. Також було

розроблено програму з графічним інтерфейсом мовою програмування R у середовищі розробки RStudio та візуальний звіт, створений за допомогою Power BI Desktop.

У останньому розділі проводився аналіз стартап-проекту, який можна реалізувати на основі даного дослідження. Було визначено його економічну цінність, можливість реалізації та стратегії розвитку. Також проведено аналіз поточного стану ринку та можливість виходу на нього.

РОЗДІЛ 1 ЗАДАЧА РОЗПІЗНАВАННЯ ЕМОЦІЙ

1.1 Актуальність задачі розпізнавання емоцій

На сьогоднішній день доволі популярним є розробки у сфері штучного інтелекту. Людство хоче створити системи, які могли б замінити або покращити її роботу. Однією з таких задач є задача розпізнавання емоцій людини, яка знайшла своє широке застосування у комерційних та соціальних сферах.

Так розглядаючи комерційні аспекти, можемо навести в якості прикладу наступне: кожна компанія, яка постачає продукт прагне зробити його більш популярним, проводячи рекламні кампанії та вдосконалюючи продукт. Для цього компанії важливо отримати відповідь від користувачів, проте останні не завжди готові витрачати власний час, щоб надати цю відповідь. Крім того, сформульовані відповіді можуть бути неточними. Власне задля цього виникає необхідність визначати емоції людини, коли вона взаємодія з продуктом в магазині, чи коли пише оцінку про продукт, тощо.

Розглядаючи соціальні аспекти, можемо сказати, що доволі часто для нас є важливими емоції інших людей, адже вони відображають їх наміри та вподобання. Інтуїтивно кожен вміє в тій чи іншій мірі помічати емоції інших людей. Але інколи, якщо людина незнайома, або ми самі переповнені власними емоціями, або кількість людей занадто велика, ми можемо пропустити настрої інших людей. Саме через це буде корисним визначення емоцій людей. Для того, щоб краще розуміти інших, запобігати конфліктним ситуаціям, покращувати власну роботу, тощо.

Таким чином задачею розпізнавання емоцій почили працювати іще з 1960-тих, проте спершу дослідники штучного інтелекту не надавали їй особливого значення. Ця тематика більше цікавила лікарів, що працювали у сфері нейронних систем людини. Так однією з основоположних робіт була робота Пола Екмана [1]. Екман разом із Фрізаном дослідили загальні закономірності виникнення тих чи інших емоцій. Таким чином вони розробили систему кодів, в якій кожній емоції людини поставили у відповідність набір скорочень тих чи інших м’язів обличчя, а відповідно

і картинку того, що відбувається на обличчі людини. Автори доводили, що дані відображення емоцій не залежить від статі, регіональних особливостей, тощо. Похибки у картинці лиця можливі лише при фізіологічних відхиленнях.

Таким чином, отримані результати надихнули інженерів розробляти технічні системи, які б розпізнавали емоції на основі зображень та відео [2, 3]. На сьогоднішній день велика кількість дослідників розробляє нейронні мережі для розпізнавання емоцій [4–6], використовуючи різноманітні архітектури.

1.2 Об’єкт задачі розпізнавання емоцій

В даному підпункті ми більш детально опишемо об’єкт та предмет нашого дослідження. Як уже згадувалось об’єктом дослідження даної роботи є емоції людини з відеопотоку. По-перше, поняття емоції потребує формалізації, а також ми обґрунтуємо вибір відеопотоку, як джерела оцінки емоцій.

Перш за все визначимо, що ми будемо називати емоціями людини. Таке, здавалось просте поняття насправді важко формалізується, адже оцінити ті чи інші емоції можна по-різному, тому все залежить від експерта, який проводить оцінки. Саме ця умова зумовила основну складність формування загальної класифікації наборів даних. Однією з спроб вирішити цю проблему була розробка загальної Кох-Канаде бази даних [7]. Іншою проблемою є те, що ми маємо широкий спектр емоцій, дуже часто ми відчуваємо кілька емоцій одночасно. Все це додатково створює неточності та відхилення у початковій вибірці.

Проте, як уже згадувалось раніше, інтерес до розпізнавання емоцій започаткувала саме робота Екмана [1], який власне і визначив основні стандартні класи емоцій, а саме: радість, сум, злість, страх, здивування та відразу. Саме ці класи емоцій будуть розглядатись у даній роботі. І наша мета буде побудувати моделі, які б визначали приналежність вхідних даних до того чи іншого класу. Крім цього Екман та Фріз визначили так зване кодування емоцій, назвавши його FACS – Facial

ActionCoding System та доводили, що будь-яка емоція визначається певним набором мікровиразів обличчя.

1.3 Канали емоційної інформації

Коли людина відчуває ту чи іншу емоцію, то це проявляється у її мові, виразі обличчя, жестах, позі, а також внутрішніх процесах, таких як пульс, активність мозку, тощо. Розглянемо кожен із каналів більш детально, а також технології та методи, які існують, на їх основі.

Аналізуючи емоції на основі обличчя людини, ми можемо знайти різноманітні роботи, які створювали моделі на основі геометричної форми губ, роту, очей [8, 9]. Пізніше на основі ідей Екмана та Фріза створювались моделі, що брали більшу кількість ознак обличчя для визначення тих чи інших емоцій. Так на основі цієї ідеї була створена наступна робота [10], що використовувала методи PCA, AAM, SAPP. Основною складністю для даних моделей є можливість існування перепон, які ускладнюють роботу алгоритмів детекції обличчя, а відповідно і розпізнавання емоцій.

Серед відомих технологій, які працюють із розпізнаванням емоцій, використовуючи візуальне зображення обличчя можемо навести Emotion API (Microsoft Cognitive Service), Affectiva, Kairos, nViso.

Аналізуючи емоції з мовлення ми маємо 2 канали інформації, а саме: текст повідомлення та те, як людина це говорить, її тон, висота, темп, тощо. Так другий канал надзвичайно важливий, адже дає можливість визначити сарказм та брехню.

Розглядаючи другий канал більш детально, можна зазначити, що він включає дані про інтонацію, мел-частостні спектральні коефіцієнти [11], лінійні спектральні коефіцієнти [12], тощо.

У роботі [13] було представлено відносно новий підхід до визначення емоцій із голосу людини, а саме з використанням висоти звуку, яку спершу порівнювали за

допомогою симетричного KDL (Kullback-Leibler Distance), після чого використовували логістичну регресійну модель, Гаусіанову модель та LDC (Linear Discriminant Classifier). Загалом у роботі було отримано модель із точністю 77%.

Серед відомих технологій, можемо навести наступні: Beyond Verbal, EmoVoice, Vokaturi, Good Vibrations. Причому перша технологія визначає додаткові параметри, які відображають емоційний стан, а саме: рівень збудженості, рівень значущості, характер емоції.

Проблемою при визначенні емоцій по голосу може бути шум.

Інколи виникає ситуація, коли не можна отримати візуальне зображення співрозмовника, наприклад при спілкування в мережі, поштою, тощо. Таким чином ми маємо наступне джерело визначення емоцій, а саме – текст повідомлення. Даний аналіз включає в себе лінгвійстичний аналіз тексту, зокрема визначення частин мови, сенсу повідомлення, пошук ключових слів, тощо. Моделі такого типу можна знайти у наступних розробках, як: Tone Analyzer, BiText, Receptivity, тощо.

Наступні канали визначення емоцій є менш очевидними, проте як виявляється і за їх допомогою можна визначити емоційний стан особи. Зокрема, хоча люди не використовують жести та позу цілеспрямлено при спілкуванні, проте підсвідомо вони змінюються у залежності від емоцій. Так найпростішим прикладом може бути той факт, що зазвичай люди відхиляються від чогось, якщо відчувають страх чи відразу, та навпаки нахиляються при відчутті радості чи гніву. При визначенні емоцій по жестах та позі можна використовувати ключові точки суглобів людини та визначати кути між відповідними прямими, що з'єднують дані точки [10]. Даний канал, напевно найменш розвинений, проте і тут існують певні технології, такі як Kinect, SSI.

Серед дослідників даний канал використовувався в наступному ключі: так у роботі [14] емоції визначались за допомогою жестів рук людини, точніше їх положення відносно голови. Так автор визначав трикутники з 3 точок: центрів рук та голови, після чого виділяв вектор ознак, який подавав на вхід методу PCA, після якого і будував класифікатор.

Також варто згадати про наступні роботи, в яких також використовувались жести для визначення емоцій: [15, 16].

Також напевно, найбільш вірогідним каналом емоцій є фізіологічні процеси, що виникають у людині, коли та відчуває ту чи іншу емоцію. Тут ми говоримо про серцебиття, пульс, активності мозку, розширення зрачка, тощо. Проте в даного каналу є і суттєвий недолік, а саме на даний момент може бути доволі проблематичним отримання таких даних без відома особи, хоча і існують технології визначення пульсу по відео, проте питання щодо інших даних залишається відкритим.

Проте незважаючи на це існують роботи, які будують моделі для визначення емоцій на основі електроенцефалограми, зокрема, дослідників цікавить лобна частина мозку, адже саме вона відповідає за емоції. На основі ЕЕГ існують роботи, що використовують частотні та часові параметри, адаптивну авторегресію [17], в іншій роботі застосовуються гібридний адаптивний фільтр, HOC-аналіз та метод опорних векторів [18]. У роботі [19] проводився аналіз емоцій, які виникали у людей у момент прослуховування музики з фільмів, після чого також оброблялись за допомогою методу опорних векторів.

Таким чином ми бачимо, що каналів отримання інформації про емоційний стан надзвичайно велика кількість і існує чимало робіт, які представляють використання даних каналів. Кожен із каналів має свої слабкі та сильні сторони, тому також існують мультимоделі, що використовують декілька каналів одночасно, що дає змогу покращити точність моделі.

Тим не менш у наступній роботі, доводиться, що емоції людини при спілкування виражаються в основному у насупних аспектах, а саме через зміст повідомлень (7%), голос (38%) та вираз обличчя (55%) [20]. Крім цього вираз обличчя відображає стан особи, відображаючи його інтерес, стрес, сумніви чи інші аспекти сприйняття зовнішніх сигналів. Саме через те, що візуальна складова несе найбільшу частку інформації про емоційний стан людини, саме через це було обрано відеопотік у якості джерела даних та об’єкту даної роботи. І з відеопотоків виділялись зображення обличчя людини та відслідковувались зміни виразів за допомогою нейронних мереж.

1.4 Методи розпізнавання емоцій на основі зображення обличчя

Більш детально розглянемо основні моделі для розпізнавання емоцій на основі візуального зображення обличчя людини.

Одними з найпростіших моделей є моделі, що використовують форму губ, рота, очей, брів, форма яких отримується за допомогою методів пошуку ключових точок. Так зокрема, при відчутті радості – ми як правило маємо усмішку на обличчі, брови підняті, при відчутті злості – посмішка відсутня, брови зведені, рот може бути привідкритий, при відчутті здивування – ми маємо привідкритий рот, припідняті брови і т.д. Дану ідею використали дослідники у наступних роботах: [8, 9, 21]. Проте даний підхід не дає доволі високих результатів, в межах 60-70%, через те, що, по-перше, він не враховує велику кількість ознак, по-друге, він має не дуже велику кількість таких ознак.

Один із наступних кращих методів став метод, що базується на методі PCA (Eigenfaces). Даний метод на відмінну від попередніх не старався виділити певні ознаки, а прагнув використовувати інформацію з всього зображення. Ідея методу полягає у наступних кроках:

- кожне зображення нормалізується та масштабується до єдиного розміру;
- отримані зображення переводять у вектори попіксельно;
- отримується векторний простір із розмірністю рівною розміру зображення;
- формується матриця коваріації на основі даних векторів;
- за допомогою методу основних компонент зменшується розмірність вихідного простору шляхом знаходження певної кількості найбільших власних чисел матриці коваріації та відповідних їм власних векторів;
- отриманий простір будемо вважати простором облич.

Даний метод знайшов широке застосування при розв’язанні різних задач детекції та розпізнавання на основі зображень. Зокрема першими, хто вирішив

використати даний метод для розпізнавання облич людини були Сірович та Кірбі у 1987 році [22] та Турк та Пентланд у 1991-му [23].

У контексті розпізнаванні емоцій даний підхід може бути застосованим наступним чином:

- отриманий простір векторів розбивається на групи, що відповідають тій чи іншій емоції;
- в межах кожної групи знаходиться вектор ваг, який би певним чином представляв вибраний клас, наприклад у роботі [23] це робилось за допомогою мінімізації суми Евклідових відстаней до інших векторів, що належать до даної групи;
- отримувались вектори ваг для кожного класу емоцій;
- далі при розпізнаванні вхідного зображення, його передобробляли, перетворювали на вектор та проектували у простір облич, після чого обчислюються Евклідові відстані між отриманим вектором та векторами ваг;
- отримані значення порівнювались та найменше значення відповідає приналежності до даного класу емоції;
- також встановлюється деяке порогове значення, якщо усі вище обчислені значення перевищують його, то дане зображення не вважаються емоцією.

Недоліками даного підходу є уразливість до положення обличчя, освітленості, наявності перепон, тощо. Тому розроблялися модифікації для підвищення точності моделі, зокрема:

- підхід, що використовував простори до очей, роту, носу (Modular Eigenfaces);
- Fisherfaces [24];
- Kernel Eigenfaces [25].

Так метод Fisherfaces, що використовує лінійний дискримінант Фішера, відрізняється способом вибору основних векторів. Якщо вище наведений метод обирав основні компоненти за допомогою методу PCA для всього простору облич, то метод Fisherfaces оперує класами, які повинна класифікувати модель та має на меті знайти такий базис, в якому відстань між класами була максимальна, а відстань всередині

класів мінімальною. Для цього необхідно максимізувати відношення: $|V^T S_b V|/|V^T S_w V|$, де V – матриця базисних векторів, S_w – сума матриць коваріацій для класів (матриця розкиду), S_b – матриця відхилень середніх значень класів від загального середнього значення. Більш детальні формули можна знайти за посиланням [24]. Даний підхід значно підвищив точність моделей в умовах затіненості. Для порівняння, даний метод давав точність 95%, а попередній – 53% у роботі [26]. Також даний метод знайшов своє застосування в чималій кількості робіт, зокрема [27–29].

Крім даного методу ще існує чимало методів для розпізнавання емоцій, які також заслуговують на увагу, зокрема методи, що засновані на методі опорних векторів, що мають на меті розділити простір векторів гіперплощинами, що розділять його підпростори на класи. Даний метод у роботі [30] дає точність від 70 до 80 % при лінійному ядрі, та 75-80% при поліноміальному.

Активні моделі форми, які на основі ключових точок визначають контури обличчя визначають набори облич, що відповідають певному класу. Коли отримується нове зображення, то відбувається підгонка його під один із контурів. Зокрема використання даного підходу можна знайти у роботах: [31, 32]. Також дещо схожий метод – активні моделі зовнішності, які наведені у роботах: [33, 34].

Як можемо помітити перші прості моделі базувались на ознаках, які виділялись розробником із певних емпіричних міркувань чи фізіологічних досліджень. Кількість таких ознак не була достатньою, тому отримані моделі мали низьку точність. По мірі розвитку методів прослідковується тенденція до збільшення кількості параметрів та перенесення завдання вибору ознак на машину. При цьому використовувалось більша частина вхідного зображення, або і все зображення. Зрозуміло, що такі методи виявились значно точнішими.

І одним із методів традиційної школи ми можемо вважати нейронні мережі, на вхід яких ми подамо ключові точки обличчя. Іншою групою методів будуть згорткові нейронні мережі, на вхід, яких будуть подаватись кадри відео повністю. Як ми побачимо, друга група моделей дає значно більшу точність у порівнянні з першими.

1.5 Огляд існуючих баз для розпізнавання емоцій

В даному підпункті розглянемо основні набори даних, які можуть бути використані при розпізнаванні емоцій людини. Основними типами даних будуть зображення та відео. Отримані результати зведемо у наступну таблицю.

Таблиця 1.1 – Бази даних з візуальними даними для розпізнавання емоцій

Назва бази даних	Тип даних	Розмір вибірки	Кількість учасників	Наявні розмітки, окрім емоцій	Тип
Ryerson Audio-Visual Database of Emotional Speech and Song [35]	відео- та аудіозаписи	7356	24	інтенсивність	неспонтанний
MMI Database [36]	відеозаписи та зображення	1280 відео та 250 зображень	43	AU	спонтанний неспонтанний
Japanese Female Facial Expressions (JAFFE) [37]	зображення	213	10	-	неспонтанний

Продовження таблиці 1.1

Extended Cohn-Kanade Dataset (CK+) [38]	зображення	593	123	FACS	спонтанний неспонтанний
DISFA [39]	відеозаписи	4845	27	AU інтенсивність	спонтанний
The Yale Face Database B [40]	зображення	16128	20	-	неспонтанний
Belfast Database [41]	відеозаписи	1400	114	-	неспонтанний
KDEF [42]	зображення	4900	70	-	неспонтанний

Як бачимо різноманітних баз даних існує чимало. Основними складнощами при їх створенні є: складність розмітки даних, яка зазвичай будується на експертних судженнях; складність створення природних даних, адже якщо попередити дати можливість людині зображати емоцію, то виникає питання наскільки правдивою вона буде, якщо людину не попереджати, то виникають складнощі, як зробити так, щоб доброволець не дізнався про це, але при цьому отримати гарні дані.

Так у роботі [38] пропонувався наступний спосіб: було запрошено людей, яким давали дивитись відео з фільмів перед комп’ютером. При цьому їх не попереджали про справжню причину експерименту та непомітно записували на камеру. Таким чином було отримано непогано вибірку, проте її частину відкинули через те, що люди не завжди сиділи рівно, відволікались, тощо.

У нашій роботі ми обрали наступну базу даних: Ryerson Audio-Visual Database of Emotional Speech and Song, адже вона містить відеозаписи та має достатній розмір.

1.6 Оцінка якості моделей розпізнавання емоцій

Після побудови моделей необхідно оцінити їх метрики якості, а саме точність та швидкодію. Для цього окремо формується вибірка, або залишається частина початкової вибірки, яка не брала участі у навчанні та валідації і на ній проводиться тестування моделей.

Для оцінки якості моделей класифікації використовуються так звану матрицю помилок (confusion matrix). Нехай ми маємо задачу, в якій у нас є вибірка з елементів $\{x_i\}_{i=1}^{N}$ і K – кількість класів для розпізнавання, в нашому випадку $K = 8$, адже стільки емоцій вибрано у якості базових. Тоді матриця помилок визначається як матриця з елементами $M = \{m_{ij}\}_{i,j=1}^{K}$, де m_{ij} визначається за наступною формулою (1.1):

$$m_{ij} = \sum_{k=1}^{N} 1[f(x_k) = j] \cdot 1[x_k = i], \tag{1.1}$$

де $f(x_k)$ – модель, яка передбачає, до якого класу потрібно віднести спостереження;

$1[\cdot]$ - оператор, що рівний 1, якщо виконується умова всередині, інакше він рівний 0 [43].

Таким чином кожним елементом матриці є значення, яке відповідає кількості елементів, які була віднесені до класу j, при цьому належачи до класу i. Зрозуміло, що модель буде вважатись кращою, якщо значення на діагоналі такої матриці будуть великими, а недіагональні елементи маленькими.

Також часто дану матрицю нормалізують, ділячи кожен елемент на суму значень у рядку, у якому вони знаходяться, таким чином, кожен елемент в новій матриці буде відповідати частці спостережень, які належать до класу i, але розпізнані як клас j, при цьому значення будуть лежати в інтервалі $[0; 1]$. Отриману матрицю зображають наступним чином, як показано на рисунку 1.1. Дана матриця зразу дає

змогу зрозуміти наскільки точна модель, а також які класи викликають у моделі проблеми. Для задачі розпізнавання емоцій – це часто почуття суму та страху.

Також додатковими агрегуючими показниками якості моделі є показник точності (precision) та повноти (recall), які визначаються наступними формулами (1.2) – (1.3):

$$Precision(i) = \frac{m_{ii}}{\sum_{j=1}^{K} m_{ij}} \tag{1.2}$$

$$Recall(i) = \frac{m_{ii}}{\sum_{j=1}^{K} m_{ji}}, \tag{1.3}$$

де m_{ij} – елемент матриці помилок.

Тобто точність – це відношення діагонального елемента до суми рядка, а повнота – відношення діагонального елемента до суми стовпця.

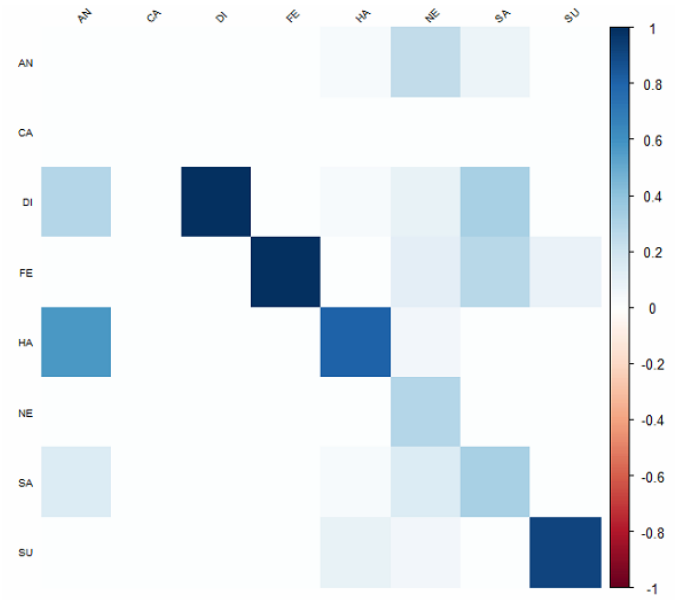

Рисунок 1.1 – Нормована матриця помилок

Також використовують агреговану метрику, що об’єднує показники точності та повноти для всіх класів, а саме F-міру, яка визначається за наступною формулою (1.4):

$$F = (1 + \beta^2)\frac{\sum_{i=1}^{K} Precision(i)Recall(i)}{\sum_{i=1}^{K}(\beta^2 Precision(i) + Recall(i))}, \quad (1.4)$$

де β – вага точності;

Таким чином, якщо повнота або точність будуть маленькими, то даний показник буде низьким. Дуже часто на практиці користуються збалансованою F-мірою, при $\beta = 1$ [44].

1.7 Постановка задачі розпізнавання емоції

Підсумовуючи все вище зазначене, остаточно сформулюємо наступне: перед нами стоїть задача класифікації облич людини за 8 класами емоцій на основі відеопотоку.

Актуальність даної тематики уже була описана у попередньому підпункті, єдине, що варто додати – це те, що у якості практичного застосування буде представлено стартап, продукт якого пропонує оцінювати емоції людей в аудиторії для підвищення ефективності проведення зустрічей у компаніях. Адже зустрічі це доволі великі витрати будь-якої сучасної компанії, тому виникає необхідність максимізувати переваги від їх проведення. Саме через це виникає потреба розробити точні та швидкі моделі для визначення емоцій.

Таким чином ми приходимо до предмета дослідження, а саме якості методів для розпізнавання емоцій. У даній роботі було вибрано нейронні мережі, а саме нейронну мережу на основі ключових точок обличчя та згорткові нейронні мережі.

Перед нами будуть стояти наступні задачі, а саме:

- підготувати вибірку даних;
- виділити області з обличчями;

- перевести отримані дані у чорно-білий канал та масштабувати їх до одного розміру;
- побудувати моделі на основі нейронних мереж, використовуючи різні підходи;
- оцінити якість отриманих моделей, а саме їх точність, повноту та швидкодію;
- вибрати найкращу модель.

Висновки до розділу 1

В даному розділі було розглянуто актуальність задачі розпізнавання емоцій, визначено предмет та об’єкт дослідження, сформульовано основні задачі роботи. А саме було вибрано 8 емоцій людини як об’єкт дослідження. В якості предмету було обрано якість методів розпізнавання емоцій людини, а саме було вибрано 2 групи методів: перша базується на нейронних мережах із використанням елючових точок обличчя, друга – згорткових нейронних мереж.

Таким чином було визначено наступні задачі роботи, такі як: підготовка даних у вигляді відео, виділення обличчя з даних, побудова нейронних мереж на основі ключових точок та згорткових мереж, а також порівняння їх якості.

Також в даному розділі було приділено значну увагу дослідженням, які проводились у області розпізнавання емоцій людини, які канали інформації використовуються для цього.

Як бачимо дана тематика дуже актуальна і в ній можна знайти чимало робіт.

Більш детально розглянуто деякі наявні методи розпізнавання емоцій на основі візуального зображення обличчя, а також відповідні бази даних. Було обговорено проблеми, які існують перед дослідником при побудові бази даних для розпізнавання емоцій.

РОЗДІЛ 2 НЕЙРОННІ МЕРЕЖІ У РОЗПІЗНАВАННІ ЕМОЦІЙ

2.1 Детекція облич

Першим етапом у розпізнаванні емоцій людини буде виділення облич із заданого зображення чи відео. Такий етап зумовлений необхідністю зменшити варіативність ознак, які занадто слабо відображають емоції людини, такі як елементи фону, волосся, елементи одягу, тощо. Оскільки перед нами стоїть задача розпізнавання емоцій саме за обличчям, відповідно ми будемо виділяти обличчя з зображень чи відеопотоку.

Під задачею детекції облич розуміється пошук усіх прямокутників, які містять усі обличчя з вхідного матеріалу, з мінімальною кількістю зайвих елементів. Успішність існуючих алгоритмів залежить від багатьох факторів, зокрема таких як: поза, кут повороту обличчя, кут камери, освітленість, наявність аксесуарів та ілюзій.

Так одним із найпопулярніших алгоритмів є алгоритм Віоли-Джонса та його модифікації, що засновані на обчисленні ознак Хаара. Даний алгоритм дає надзвичайно гарні результати для фронтальних облич, тобто таких, що кути повороту яких мають $[-15°; 15°]$ у горизонтальній площині та $[-35°; 10°)$ у вертикальній. Точність моделей заснованих на цьому методі погіршується по мірі збільшення кутів повороту облич. В даній роботі ми будемо розпізнавати емоції фронтальних облич за допомогою моделей заснованих саме на методі Віоли-Джонса. Таким чином ми розуміємо, що точність даних моделей залежить від кута повороту, кута камери.

Другим аспектом є наявність аксесуарів, таких як окуляри, борода, вуса, макіяж, тощо, що також вносять похибки у роботу алгоритмів, а тому вимагають використання кількох моделей, або більш складних моделей. Так існують алгоритми детекції облич, які шукають очі людини, адже будь-яке обличчя має очі. Але і такому алгоритму ми можемо протиставити контрприклад, на якому модель даватиме погані результати. Наприклад, взявши обличчя, яке матиме затемнені окуляри.

Останнім згаданим аспектом є можливість ілюзій. Це так звані об’єкти, що є схожими на обличчя людини, проте не є ним, зокрема манекени чи маски.

Усі згадані вище аспеки надзвичайно важливі при розв’язанні задачі детекції обличчя, проте це не є задачею даної роботи. Ми будемо працювати з фронтальними зображеннями, без аксесуарів та ілюзій.

У якості методу детекції візьмемо метод Віоли-Джонса, який був опублікований у 2001 році Полом Віолою та Майклом Джонсам [45]. Дана робота мала значний вплив на розвиток алгоритмів розпізнавання образів, адже даний алгоритм, не дивлячись на свою простоту, впершу досволив розв’язати задачу детекції облич у реальному часі.

Ідея даного методу брала за основу Хаар-подібні ознаки. Ознакою Хаара називається величина, що обчислюється за формулою (2.1):

$$T = \left(\sum_{j=1}^{N} w_j \sum_{(x;y)\in A(j)} I(x,y) \right) / s^2, \tag{2.1}$$

де T – значення ознаки;

w_j - це константи, що обернено пропорційні до розміру відповідної області (кількості пікселів);

N – кількість областей;

$I(x,y)$ – це значення пікселя зображення;

$A(j)$ – j-та область зображення;

s – коефіцієнт маштабування ($s \geq 1$).

Ознаки Хаара визначаються як прямокутники з чорно-білими областями, а значення ознаки, як сума значень пікселів у чорні області мінус сума пікселів у білих областях. Додатково значення суми з області множиться на ваговий коефіцієнт. Графічно це можна представити рисунками 2.1 – 2.2:

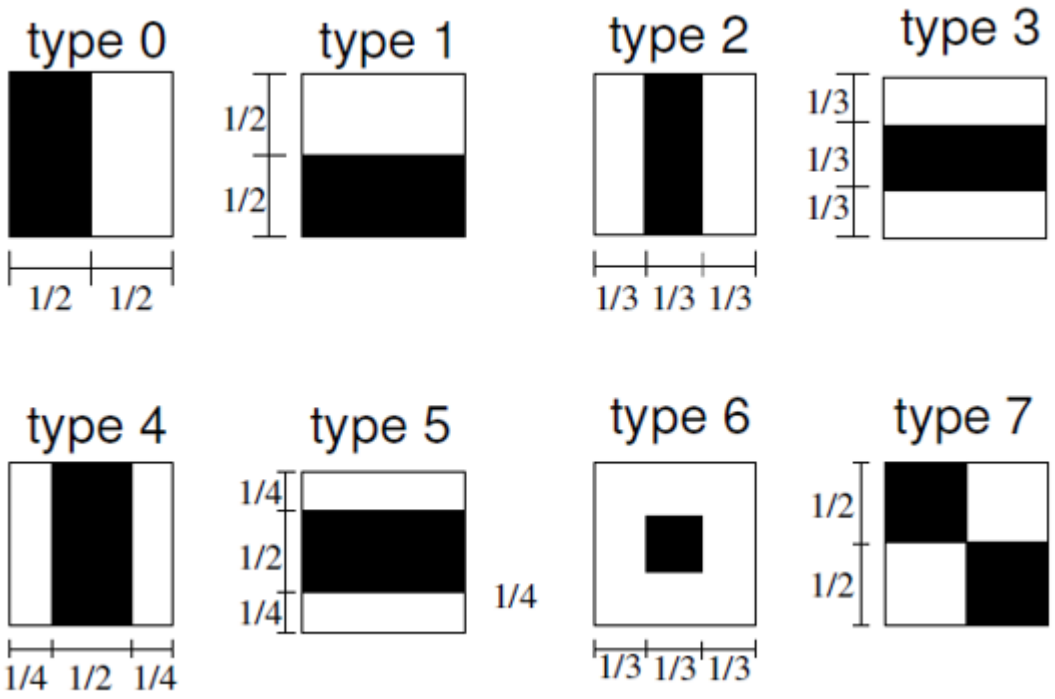

Рисунок 2.1 – Типи ознак Хаара

Пізніше були додані додаткові ознаки, повернуті на 45 градусів і виглядали, як представлено на рисунку 2.2:

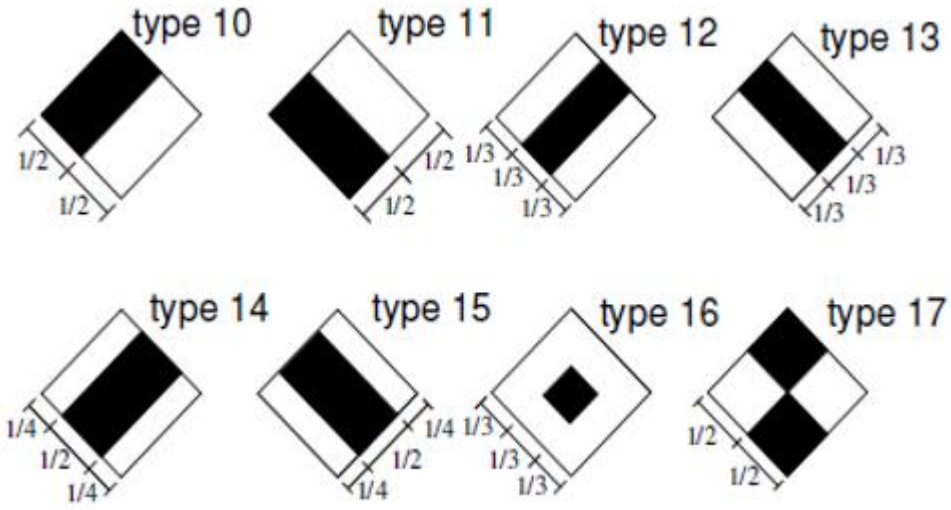

Рисунок 2.2 – Додаткові ознаки Хаара

Таким чином отримані ознаки обчислюються з використанням інтегрального представлення зображення, а тому не вимагають великої кількості обчислень і при цьому дані ознаки є інваріантні відносно маштабування.

Більш детально з даним алгоритмом можна ознайомитись за джерелом [46].

2.2 Нейронні мережі

Штучні нейронні мережі виникли як спроба змоделювати шляхи навчання людини за допомогою побудови складних шарів із персептонів та зв'язків між ними. По аналогії з людським розумом для побудови нейронної мережі достатньо визначити її архітектуру, побудувати необхідні ваги, а також визначити вхідні дані.

Хоча така постановка нейронних мереж не є новою, проте в даному пункті ми розглянемо математичну основу штучних нейронних мереж, щоб детальніше розуміти необхідність введення тих чи інших допоміжних елементів при побудові певної архітектури мережі.

Отже, ми вважатимемо нейронною мережею наступну математичну модель, що для зручності може бути визначена у вигляді графу, що складається з шарів персептонів. Будь-яка нейронна мережа містить вхідний шар та вихідний шар нейронів, а також може містити приховані шари.

Між нейронами шарів визначаються зв'язки, які задаються як ребра графу з певними вагами $\vec{w}$. На рисунку 2.3 представлена нейронна мережа з повними ребрами, двома прихованими шарами в просторах R^5 та R^4, та вхідним та вихідним шарами R^4 та R^3 відповідно.

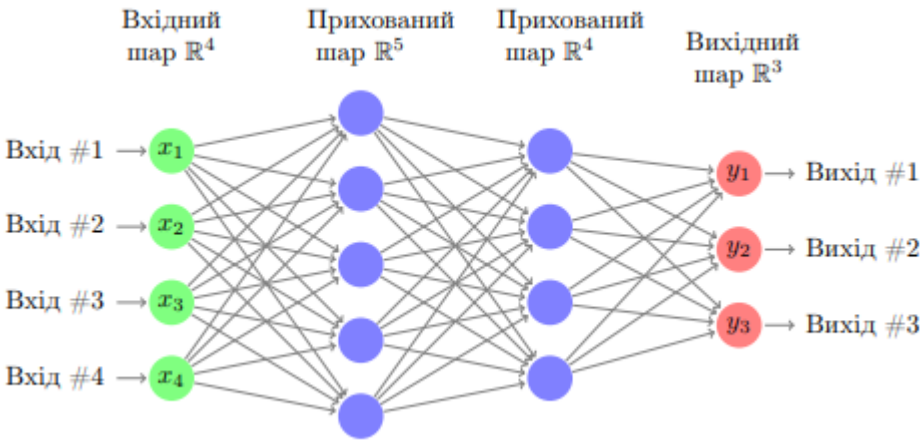

Рисунок 2.3 – Приклад структури нейронної мережі

Якщо ми отримаємо уже побудовану нейронну мережу, тобто таку, в якій ми знайдемо оптимальні ваги $\vec{w}$, то значення нейрону кожного шару визначається з значень нейронів попереднього шару за наступними формулами (2.2) – (2.3):

$$y_i^s = f(z_i^s), \tag{2.2}$$

$$z_i^s = \sum_{j=1}^{m_{s-1}} w_{i,j}^s \, y_j^{s-1} + w_{i,0}^s, \tag{2.3}$$

де y_i^s – значення вершин шару s;

f – активаційна функція;

m_{s-1} – розмір $(s-1)$-го шару;

$w_{i,j}^s$ – вага ребра i, j із шару s;

$w_{i,0}^s$ – зміщення шару s для i вершини [47].

Тобто для визначення значень вершин шару s потрібно знайти зважене значення вершин попереднього шару та додати зміщення. Інколи для спрощення розуміння та спрощення формул в нейронну мережу додатково вносять вершину, що має значення 1, а її зв’язки з вершинами наступного шару відображають ваги зміщення.

Таким чином ми отримуємо відображення, яке приймає на вхід значення вхідного шару та прогнозує значення вихідного.

У якості активаційної функції зазвичай беруть функцію Хевісайда, логістичну сигмоїду, сигмоїду, гіперболічний тангенс, softsign, тощо. Використання тиж чи інших активаційних функцій у нейронних мережах можна знайти у роботак різних дослідників [48–50]. Звичайно це лише частина тих функцій, які можуть бути використані у якості активаційних [51]. Доволі часто обираються функції, які є диференційованими у всіх точках. Проте досліджувались і недиференційовані функції єфективність яких доволдилась у вище згаданих роботах.

Крім цього у задачах класифікації, а саме така задача розглядається у даній роботі, часто використовують додатковий шар перед вихідним шаром нейронів відомий як softmax-шар, який задається рівнянням (2.4):

$$\sigma(z,i) = \frac{e^{z_i}}{\sum_{k=1}^{m} e^{z_k}}, \tag{2.4}$$

де z – вектор шару, що подається на вхід шару softmax;

$\sigma(z,i)$ – значення виходу i.

Таким чином вихідний шар відображає імовірнісну складову класифікації [47].

Отже, ми переходимо до того, як ж моделюється нейронна мережа. Як уже згадувалось для побудови нейронної мережі, нам потрібно визначити архітектуру мережі, тобто по суті задати загальний вигляд відображення вхідних даних у вихідні, а також, що найважливіше визначити оптимальні ваги мережі.

Якщо перед нами постає задача навчання з учителем, тобто ми знаємо вихідні дані моделі для певної вибірки, то у якості функцію, яку необхідно мінімізувати часто береться сума квадратів похибок, яка у нашому випадку набуде вигляду:

$$E(w) = \sum_{i=1}^{n} (\tilde{y}_i - y_i)^2, \tag{2.5}$$

де $E(w)$ – функція квадрату похибок;

n – розмірність вихідного шару;

$\widetilde{y_i}$ – оцінене значення, спрогнозоване моделлю;

y_i – значення задане у вибірці.

Інколи в якості функції похибок береться функція перехресної ентропії [52].

Для оптимізації використовуються метод градієнтного спуску та його модифікації, зокрема метод Adam.

У даних методах часто використовується перша похідна від функції відображення вхідного шару у вихідний, тому для пришвидшення обчислень користуються методом зворотнього обчислення похибок.

2.2.1 Метод зворотнього поширення помилок

Оскільки, як ми могли зрозуміти функція, яка оцінюється при побудові нейронних мереж є надзвичайно громіздкою, тому безпосереднє обчислення градієнту функції помилок, яка використовується у більшості методів оптимізації, призведе до повільного навчання, а тому існує спеціальний алгоритм, що носить назву методу зворотнього поширення помилок (error backpropagation). Даний метод передбачає використання наступних кроків.

Спершу визначається значення вхідного шару x_n.

Після цього для вихідного шару визначаємо його помилку, за формулою (2.6):

$$\delta_i^m = \frac{\partial E}{\partial y_i^m} F'(z_i^m), \tag{2.6}$$

де δ_i^m – значення звортоньої помилки на вихідному шару;

E – функція помилок;

y_i^m – значення вихідного шару;

F – загальна функція моделі;

z_i^m – згортка значень попереднього шару [53].

Далі на кожному прихованому шарі визначається так помилка за наступною формулою (2.7):

$$\delta_i^s = f'(z_i^s) \sum_{j=1}^{m_{s+1}} w_{i,j}^{s+1} \delta_j^{s+1}, \tag{2.7}$$

де m_{s-1} – розмір $(s-1)$-го шару;

$w_{i,j}^s$ – вага ребра i, j із шару s [53].

На останньому кроці визначається градієнт функції помилок настпною формулою (2.8):

$$\frac{\partial E}{\partial w_{i,j}^s} = \delta_i^s y_i^{s-1}, \tag{2.8}$$

де E – функція помилок;

$w_{i,j}^s$ – вага ребра i, j із шару s і параметр нашої моделі;

y_i^{s-1} – значення виходу попееднього $(s-1)$-го шару [53].

Таким чином даний метод сприяє більш швидшому та точнішому навчанні у порівнянні з стандартним способом знаходження градієнта функції.

Тому саме даний метод став основним при навчання моделей, що були побудовані у даній роботі.

2.2.2 Тренування нейронних мереж

У даній роботі розглядається тренування нейронних мереж на основі пакетного навчання, тобто оскільки кількість даних надзвичайно велика, то всі дані розбиваються на невеликі пакети і мінімізації функції помилок відбувається лише на них в межах 1 ітерації певної епохи навчання. Деякі пакети можуть використовуватись повторно, таке повторне використання пакетів будемо називати епохами.

Для мінімізації параметрів моделей нейронних мереж використовують різноманітні алгоритми, загального поширення набув метод Адама (Adaptive moment estimation), який відрізняється від звичайного методу градієнтного спуску тим, що крок, за яким оптимізовується вектора параметрів не є постійним, а змінюється у залежності від того, як відбувався рух у минулі кроки та наскільки важливим є той чи інший параметр. Формально на кожному кроці даний алгоритм передбачає використання наступних формул (2.9) – (2.16):

$$g_t = \frac{\partial E}{\partial w}(w_{t-1}), \tag{2.9}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{2.10}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{2.11}$$

$$\overline{m_t} = \frac{m_t}{1-{\beta_1}^t} \tag{2.12}$$

$$\overline{v_t} = \frac{v_t}{1-{\beta_2}^t} \tag{2.13}$$

$$w_t = w_{t-1} - \alpha \frac{\overline{m_t}}{\sqrt{\overline{v_t}} + \varepsilon}, \tag{2.14}$$

де g_t – градієнт функції, що мінімізується;

E – функція помилок;

w_{t-1} – значення вектора параметрів на $(t-1)$-му кроці;

$m_t, v_t, \overline{m_t}, \overline{v_t}$ – внутрішні параметри введені для зручності;

β_1, β_2 – параметри забування, які задаються наперед і знаходяться в інтервалі $[0; 1)$;

α – крок зміни вектора параметрів, що задається наперед;

ε – додаткове значення, що запобігає можливості виникнення 0 у знаменнику, $\varepsilon > 0$ [53].

Таким чином даний метод є менш уразливим до можливих плато у функції і швидшим у порівнянні з стандартним методом градієнтного спуску завдяки адаптивній настройці кроку, з яким відбувається зміна вектора параметрів.

2.2.3 Регуляризація

При навчанні нейронних мереж може виникнути їх перенавчання, що проявляється в тому, що модель, мінімізуючи функцію похибок на тестових даних, при цьому гірше відображає реальну модель, а тому її похибки на даних, що відсутні у тестовій вибірці починають зростати. Для уникнення такої ситуації вводяться додаткові умови на вектор ваг w або на умови навчання.

Зокрема існує метод L_p-регуляризації, який полягає в тому, що задачу знаходження похибок приводять до задачі умовної оптимізації, накладаючи обмеження на велечину вектора ваг, або по аналогії з методом штрафів визначається функція похибок за наступною формулою (2.15):

$$\tilde{E}(w) = E(w) + \lambda P(w), \tag{2.15}$$

де $\tilde{E}(w)$ – функція втрат із накладеним штрафом;

$E(w)$ – початкова функція втрат;

λ – вага штрафу;

$P(w)$ – штрафна функція [54].

У випадку L_p-регуляризації у якості штрафної функції береться норма порядку p, яка задається наступною формулою (2.16):

$$P(w) = \|w\|_p = \left(\sum_{i=1}^{n} w_i^p\right)^{1/p}, \tag{2.16}$$

де w_i – i-те значення вектора w;

p – порядок норми.

Популярним є використання L_2 та L_1-регуляризації [54] через їх ефективність у роботі. L_2-регуляризації відома як регуляризація Тихонова. У даному випадку ваги обмежуються гіперсферою, отримана функція є диференційованою, що є корисно при використанні методів мінімізації першого порядку та вище.

Окрім методу L_p-регуляризації, існує чимало інших способів запобігання перенавчанню, зокрема метод поширення ваг, рання зупинка навчання, при якій тестування моделі відбувається паралельно з навчанням і в момент, коли помилка на вибірці для валідації моделі починає зростати, навчання призупиняється.

Також варто згадати про метод відсіювання (dropout), при якому кожен нейрон залучається на епохи навчання з певною наперед заданою ймовірністю. Таким чином відбувається навчання кількох нейронних мереж одночасно, тому і отриманий результат є кращим [55].

В даній роботі використовувався метод ранньої зупинки для запобігання системи від перенавчання.

2.2.4 Нейронні мережі на основі ключових точок обличчя

Оскільки основну увагу в даній роботі планується надати згортковим мережам та їх порівнянням із мережами, побудованими на основі ключових тоок обличчя, тому в даному підпунктів ми не буде зусереджуватись на деталях, а лише в загальному опишемо, як відбувається визначення ключових точок обличчя, а також які саме значення подаються на вхід нейронної мережі.

Отож, алгоритм визначення ключових точок обличчя наступний: спершу ми визначаємо положення обличчя задля економії обчислювальних ресурсів, адже не потрібно аналізувати більш детально ті області, які навіть не є обличчями.

Після цього ми можемо готуємо вибірку наступним чином: створюємо розмітку з координат: $(x_1, y_1, x_2, y_2 \dots, x_m, y_m)$. Для пришвидшення навчання майбутньої моделі використовуємо метод PCA (principal component analysis), який виділяє основні компоненти множини точок.

В загальному це виражається тим, що необхідно мінімізувати наступну функцію:

$$\sum_{i=1}^{m} \left\| z_i - a_0 - \sum_{j=1}^{k} a_j (a_j, z_i - a_0) \right\|^2 \to \min, \tag{2.17}$$

де $\{z_i\}_{i=1}^{m}$ – вектори простору R^n, які ми будемо апроксимувати;

$\{a_i\}_{i=1}^{k}$ – вектори, якими апроксимуємо, $k \leq n$.

Тобто наша мета найти меншу кількість векторів, таких, щоб їх лінійна комбінація відновлювала обраний вектор z_i.

Наступним етапом підготовки даних для навчання нейронної мережі є визначення відстаней між парами ключових точок. Саме ці значення подаються на вхід нейронної мережі. Навчивши модель визначати ключові точки, ми можемо використовувати інші вибірки, для яких дана модель буде визначати ключові точки.

Отримані результати, а точніше знову ж таки відстані між цими точками ми подамо на вхід нової нейронної мережі для класифікації емоцій.

2.3 Згорткові нейронні мережі

Як уже згадувалось раніше для побудови нейронної мережі достатньо визначити її архітектуру, побудувати необхідні ваги, а також визначити вхідні дані.

Таким чином першою задачею буде саме виділення вхідних даних із вхідного матріалу. Загалом дослідники використовували відео, статичні зображення, аудіо, письмо, тощо у якості вхідного матеріалу, виділяючи певним чином ті чи інші ознаки, які потім подавали на вхід моделі для навчання.

Так із аудіо – це можуть бути певні коливання звуку чи пікові точки, темп і тональність мовлення, які будуть відображати емоційний стан особи. Для написаного тексту – це буде використання тих чи інших слів.

У випадку зображень та відео існує чимало робіт, що доводять можливість використання нейронних мереж на вхід, яких подаються виділені певним чином ознаки. Зокрема можемо згадати про нейронні мережі на основі ключових точок обличчя, ефективність яких доведена у роботах [5, 6].

Проте недоліком даного підходу є те, що ми власноруч обираємо ознаки, які нам здаються важливими. Власне через це може виникати похибка роботи моделей.

Таким чином у 1989 Лекун представив світові згорткові нейронні мережі [4] або convolutional neural network (CNN).

Основна ідея даних мереж полягає у тому, що зображення або відео, що є фактично набором зображень, подається на вхід нейронної мережі. І мережа автоматично навчається та визначає ієрархію необхідних ознак. Таким чином можна отримати мережу, яка буде точнішою у порівнянні з мережею побудованою на основі традиційних підходів [5].

Для підвищення точності моделі дослідник попередньо використовує додаткові шари, які доволі часто призводять до колосального збільшення об’єму даних та збільшення кількості обчислень.

Незважаючи на чудову ідею, вона не завжди працює однаково гарно. Адже для нас важливе не лише точність системи, але і її швидкодія, куди ми включаємо як швидкість розпізнавання, так і швидкість навчання. І тут потрібно розуміти, що побудувавши нейронну мережу з більшою кількістю шарів та зв’язків між ними, більшою кількістю шарів, ми, можливо отримаємо модель більш вищої точності. Проте і час навчання та розпізнавання виросте в чималу кількість раз. Дана проблема посилюється, коли перед нами дані великої розмірності, такі як зображення чи відео.

Саме тому виникає потреба якимось чином зменшити об’єми даних без суттєвих втрат у точності моделей. Тому перед безпосереднім навчанням додатково визначаються шари, що покликані зменшити розмір даних шляхом їх усереднення, зменшення, тощо.

Таким чином дослідник, що обрав метод згорткових нейронних мереж повинен не лише визначити кількість шарів при навчанні, кількість вузлів та зв’язків між шарами, але також обрати методи фільтрації даних.

Можливо через таку різноманітність архітектур згорткових нейронних мереж, а також підвищення точності моделей, дана тематика залишається цікавою для дослідників.

Початково дана ідея була запропонована Лекуном для розпізнавання ZIP-кодів [4]. Дані ідея була використана для розпізнавання символів та цифр у роботі [56]. Додаткові дослідження, що включали в себе обговорення активаційних функцій та навчання без вчителя наявні у роботах [57–59], обговорення архітектурних змін наявні у роботах [60, 61]. Одною з цікавих ідей є візуалізація активації ознак вищих рівнів [62].

2.3.1 Згортковий шар

Як уже було сказано вище згорткові нейронні мережі автоматично виділяють необхідні дані з зображень чи відео для оптимальної побудови нейронних мереж. Для цього перед подачею даних до нейронної мережі додаються додаткові шари згорток, які покликані виділити дані ознаки сильніше на фоні всього іншого.

Вхідні дані визначаються як тензор 3-го або 2-гого рангу, в залежності від кількості каналів, які присутні у даних. Під каналами розуміється кількість змінних, якими описується колір пікселя. Тобто у випадку RGB-кодування ми матимемо 3 канали. Проте для зменшення об'єму даних доволі часто вхідні дані переводять у 1 чорно-білий канал. Тому наступні формули ми будемо наводити у припущенні, що ми працюємо з чорно-білим зображення розміру $n \times m$. Таким чином визначимо функцію I, як відображення $I: \{\overline{1;n}\} \times \{\overline{1;m}\} \to \mathrm{R}$. Згорткою будемо називати перетворення заданого формулою (2.18:

$$(I * K)(r,s) = \sum_{i=-u}^{u} \sum_{j=-v}^{v} K(i,j) I(r+i, s+j), \tag{2.18}$$

де $(I * K)(r,s)$ – значення результуючої матриці у точці (t,s);

K – ядро згортки, задане у вигляді матриці розмірності $(2u+1;\ 2v+1)$;

$I(i,j)$ – значення пікселя вхідного зображення.

Тобто кожен піксель відфільтрованого зображення обчислюється на основі квадратної області навколо нього з використанням ядра згортки. Рисунок 2.4 демонструє дане перетворення:

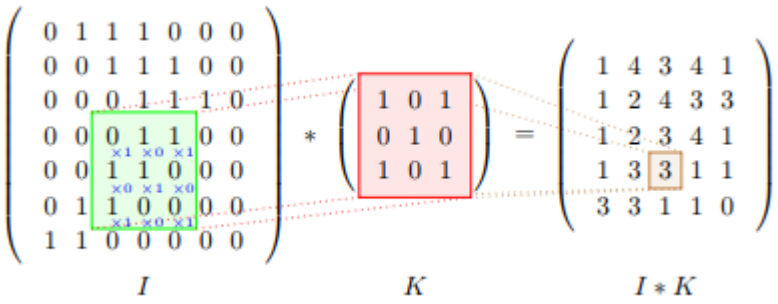

Рисунок 2.4 – Перетворення згорткою

Можемо також помітити, що для коректності вище наведеної формули, для конкретно вибраного пікселя необхідно мати достатню кількість сусідніх пікселів. Тому відфільтрвоне зображення буде дещо меншого розміру, а саме $(m - 2u - 1; n - 2v - 1)$. У випадку, зображеному вище, ми маємо: $m = n = 7; u = v = 1;$ і відповідно розміри вихідного зображення: (5×5).

У випадку, коли ми маємо кілька каналів, ми здійснюємо згортку по всім каналам незалежно.

У випадку даних більшої розмірності, наприклад коли ми працюємо з відео, ми маємо уже 3 координатні осі: 2 для зображення кадру та 1 для часової осі. У цьому випадку ми уже оперуємо тензором 3-го порядку, при цьому у загальному випадку ядро може задаватись як тензор вищих порядків, скажімо 4-го. Відповідно вище наведена формула згортки набуде наступного вигляду (2.19):

$$(I * K)(r, s, p) = \sum_{i=-u}^{u} \sum_{j=-v}^{v} \sum_{t=1}^{IC} K(i, j, t, p) I(r + i, s + j, t), \qquad (2.19)$$

де $(I * K)$ – вихідний тензор розмірності $(m - 2u - 1; n - 2v - 1; OC)$;

K – ядро згортки, задане у вигляді тензору розмірності $(2u + 1;\ 2v + 1; IC; OC)$;

I – вхідний тензор 3-го рангу розмірності $(m; n; IC)$;

IC – кількість вхідних каналів;

OC – кількість вихідних каналів.

В загальному такому випадку дане перетворення можна зобразити у вигляді, приведеному на рисунку 2.5:

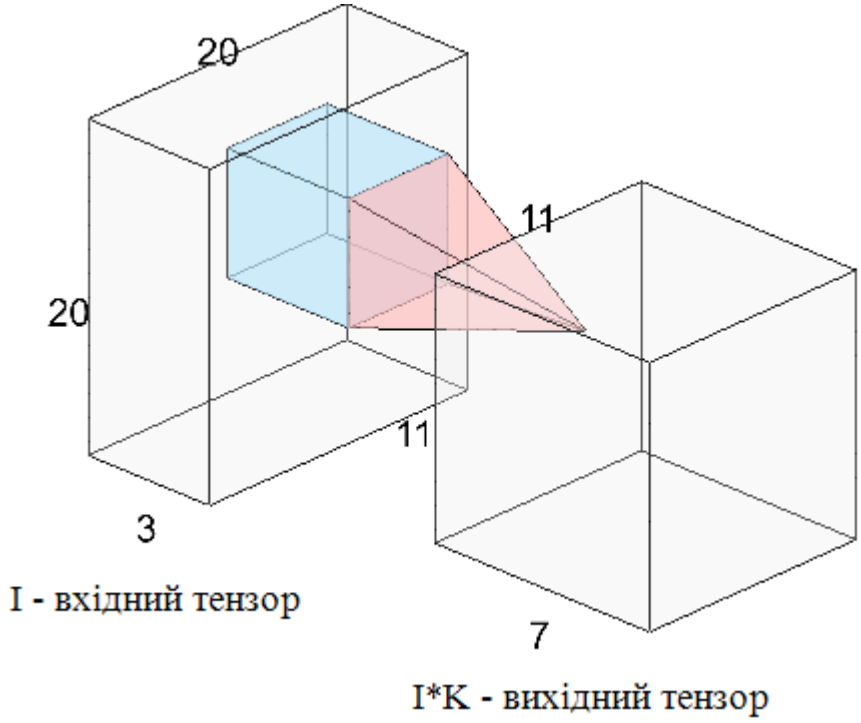

Рисунок 2.5 – Приклад перетворення вищої розмірності

На рисунку вибрані наступні розмірності $m = n = 20; u = v = 4; IC = 3; OC = 7$.

У випадку двовимірного ядра часто обирають Гаусове розмиття або фільтр Гауса, який має формулою (2.20):

$$K(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{2.20}$$

де (x, y) – координати нашої матриці [63].

Даний фільтр використовується у багатьох напрямках машинного навчання, а також у дизайні, адже як свідчить його назва він розмиває чіткі кути. Таким чином мінімізується залежність даних від відстані до об’єкта.

Варто також зауважити, що зазвичай при побудові згорткового шару одночасно використовується декілька фільтрів, в результаті яких на виході ми отримуємо нові зображення, які зазвичай називають картами ознак. Тобто в загальному кількість ознак може змінювати як зображено на рисунку 2.6:

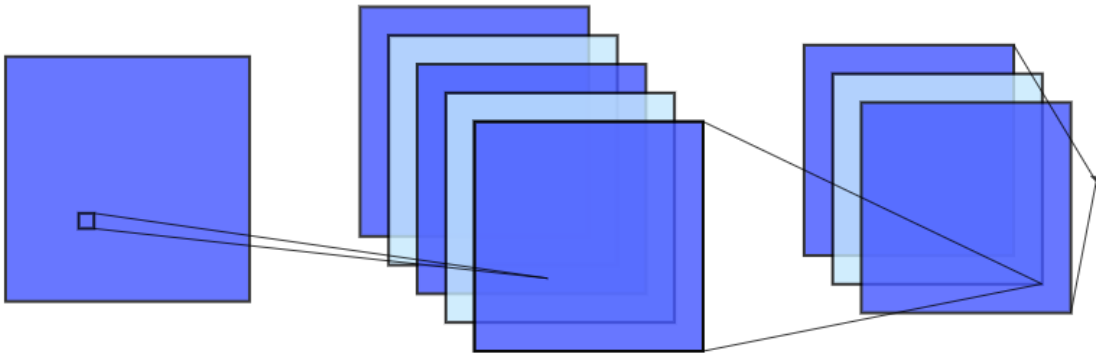

Рисунок 2.6 – Схема двох згорткових шарів

Формулою (2.21) це виражається наступним чином:

$$y_i^s = b_i^s + \sum_{j=1}^{m_{s-1}} K_{i,j}^s * y_j^{s-1}, \tag{2.21}$$

де y_i^s – значення вершин шару s;

b_i^s – зміщення шару s для i вершини;

m_{s-1} – розмір $(s-1)$-го шару

$K_{i,j}^s$ – значення ядра [47].

Отже, таким чином згортковий шар дозволяє виділити ознаки по окремості на кожну карту ознак. В результаті таких дій ми отримуємо зображення, з якими простіше працювати нейронній мережі, адже значна її частина заповнена нулями, що покращує швидкість обчислень.

2.3.2 Агрегуючий шар

Як ми побачили у попередньому підпункті при використанні згорткового шару ми збільшуємо дані у кілька разів.Тому очевидною проблемою в даному випадку стає швидкість навчання та роботи моделі. Тому доволі часто згортковий шар використовують разом із агрегуючими (pooling layer). Виглядає це як зображено на рисунку 2.7:

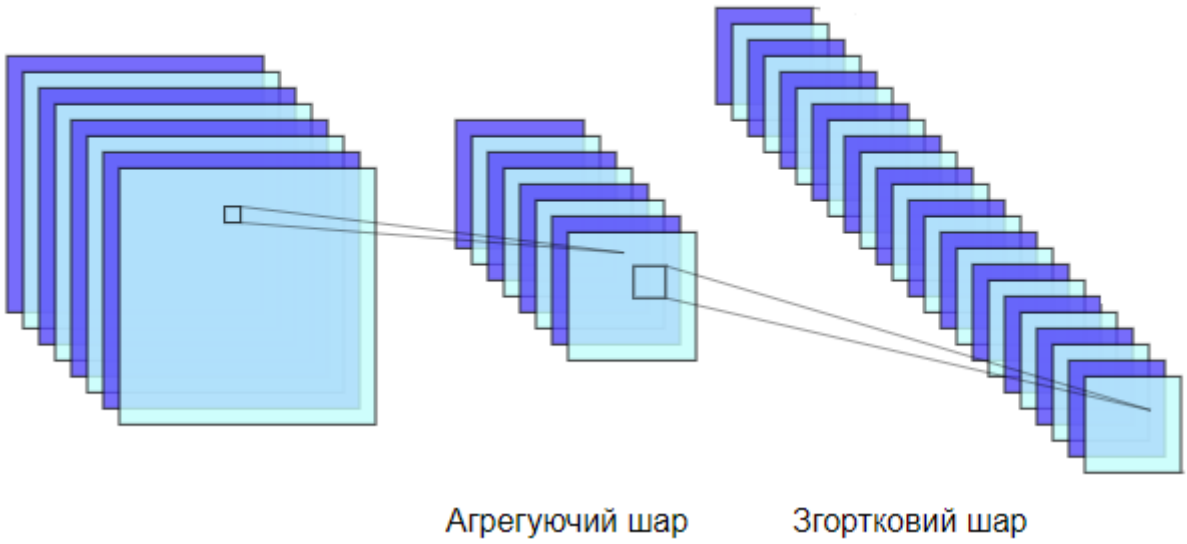

Рисунок 2.7 – Композиція агрегуючого та згорткового шарів

У результаті агрегуючого шару кількість даних зменшується в k^2 разів, де k – розмір вікна, по якому проводиться агрегування.

Існує два найбільш поширені способи агрегування: агрегування шляхом макмимізації вікна (max pooling) та шляхом усереднення вікна (average pooling).

Для обчислення даних шарів використовується вікно. Наше зображення розбивається на дані вікна, які при цьому не перетанють і перекривають максимальну площу зображення. В межах кожного вікна обирається максимальне або відповідно середнє значення. Результати роботи таких шарів продемонстровані на рисунках 2.8 – 2.9:

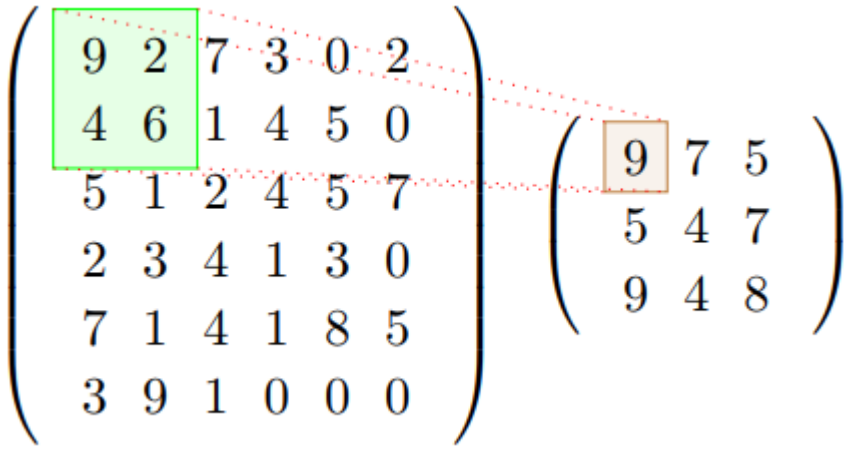

Рисунок 2.8 – Агрегування максимізацією

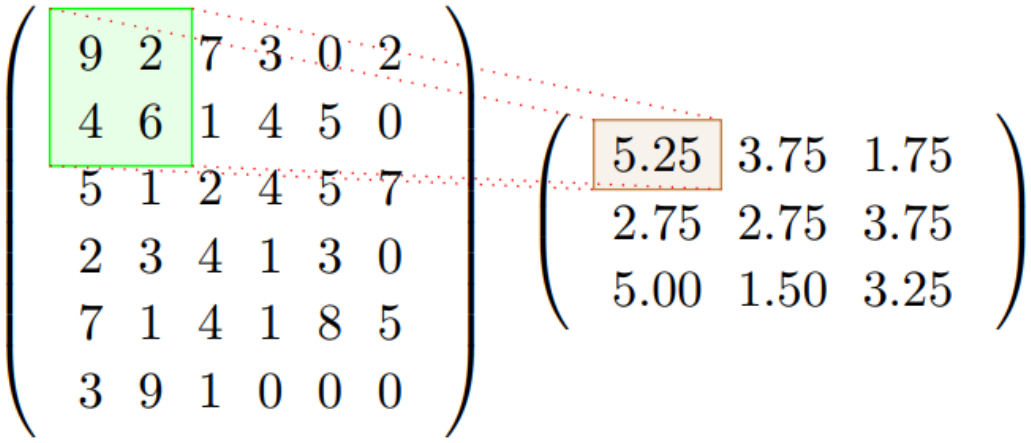

Рисунок 2.9 – Агрегування усередненням

2.3.3 Додаткові шари у згорткових нейронних мережах

У попередніх підпунктах ми розглянули основні шари, які використовуються у згорткових нейронних мережах. В даному підпункті ми додатково опишемо кілька відомих шарів, які використовуються додатково при побудові мереж, зокрема нелінійний, випрямлуючий, нормалізаційний та повнозв’язний шари.

Нелінійний шар виражається формулою (2.22):

$$y_i^s = g_i f\left(y_i^{s-1}\right), \tag{2.22}$$

де y_i^s – значення вершин шару s;

g_i – коефіцієнт пропорційності;

f – активаційна функція [47].

Як і раніше в якості активаційної функції може використовуватись нелінійні функції, такі як сигмоїда, гіперболічний тангенс, тощо. При цьому інколи 1 обирають у якості коефіцієнту пропорційності. Ефективність даного шару розглядалась в роботах деяких дослідників [62].

Випрямляючий шар задається даною формулою (2.23):

$$y_i^s = \left|y_i^{s-1}\right|, \tag{2.23}$$

де y_i^s – значення вершин шару s [47].

Згідно роботи [62] даний шар покращує швидкої мережі. Інколи даний шар можна включити всередину нелінійного шару, обравши в якості активаційної функції функція, яка залежить від модуля значень попереднього шару.

Нормалізаційний шар, або як його ще називають рішуча нормалізація задається наступною формулою (2.24):

$$y_i^s = y_i^{s-1} - \sum_{j=1}^{m_{s-1}} K_{i,j}^s * y_j^{s-1}, \tag{2.24}$$

де y_i^s – значення вершин шару s;

m_{s-1} – розмір $(s-1)$-го шару;

$K_{i,j}^s$ – значення ядра Гауса [47].

Даний шар також розглядався у роботі дослідників [62].

Повнозв’язний шар – це шар, що використовує усі карти разом, отримані на попередньому шарі, що задаються формулами (2.25) – (2.26):

$$y_i^s = f(z_i^s), \tag{2.25}$$

$$z_i^s = \sum_{j=1}^{m_{s-1}} \sum_{r=1}^{m} \sum_{t=1}^{n} w_{i,j,r,t}^s \left(y_j^{s-1}\right)_{r,t}, \tag{2.26}$$

де y_i^s – значення вершин шару s;

f – активаційна функція;

m_{s-1} – розмір $(s-1)$-го шару

$m \times n$ – розмір карт із $(s-1)$-го шару;

$w_{i,j,r,s}^s$ – вага ребра з шару s, що пов’язана з позицією $(r;t)$ j-тої карти ознак;

$\left(y_j^{s-1}\right)_{r,t}$ – значення позиції $(r;t)$ j-тої карти ознак $(s-1)$-го шару [47].

Таким чином поєнуючи різні шари, їх кількість та їх параметри можна отримати колосальну кількість архітектур нейронних мереж. Можливо, саме таке різноманіття привертає увагу дослідників до цієї сфери. Зокрема на рисунку 2.10 наведемо приклад однієї з можливих наборів шарів, яка використовує згортковий, агрегуючі, нормалізуючий та повнозв’язний шари:

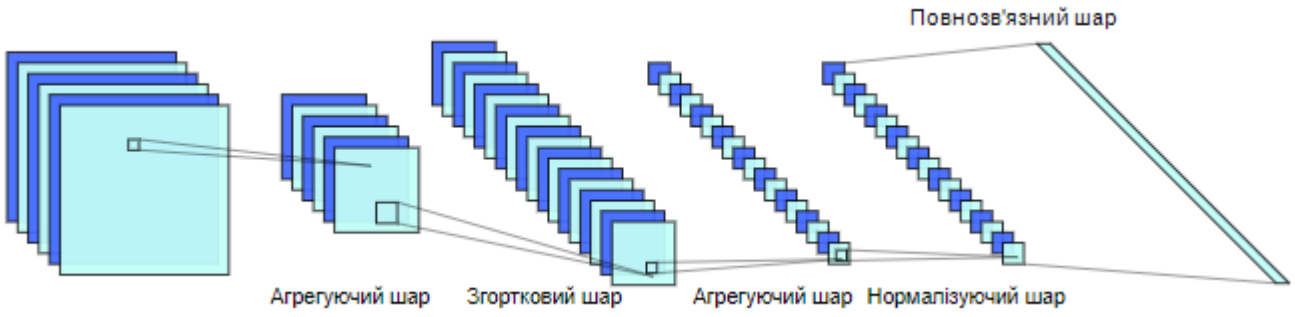

Рисунок 2.10 – Приклад архітерктури шарів згорткової нейронної мережі

2.3.4 Архітектури згоркових нейронних мереж

Традиційні архітектури згорткових нейронних мереж включають в себе використання нелінійних шарів, які включають у склад згорткових шарів та агрегуючих шарів шляхом усереднення. Основна властивість даних шарів – це використання нелінійності гіперболічного тангенса і поширення ваг [47].

Більш сучасні архітектури згорткових нейронних мереж містять більшу кількість згорткових шарів, зокрема працюючи з базою даних [62] було розроблено архітектуру з 5 згорткових та агрегуючих шарів, випрямляючого шару, шару нормалізації. Також дана модель використовує 2 повнозв’язних шари [64]. Таким чином сучасні архітектурні рішення пропонують використання більшої кількості згорткових шарів у порівнянні з стандартними.

Висновки до розділу 2

В даному розділі було більш детально розглянуто математичні основи методів, що будуть використовуватись у роботі та підкріплено їх ефективність посиланнями на роботи інших дослідників. Таким чином було обрано метод Віоли-Джонса для детекції облич, нейронні мережі на основі ключовий точок та згорткові нейронні мережі як основні методи, що будуть використовуватись для побудови та порівняння моделей.

Також було детально розглянуто методи боротьби із перенавчанням, такі як регуляризація, метод ранньої зупинки, метод відсіювання, тощо, та різноманітні шари, які можна використовувати у згорткових нейронних мережах, зокрема згортковий, агрегуючі, нормалізуючі шари, тощо.

Також було обговорено традиційну і більш сучасну архітектури нейронних мереж.

РОЗДІЛ 3 МОДЕЛІ РОЗПІЗНАВАННЯ ЕМОЦІЙ

3.1 Засоби моделювання

Для моделювання використовувалась відкрита бібліотека Google – TensorFlow, а також її додаток Keras [65], яка містить зручні засоби для моделювання та анлізу побудованих моделей. Дані бібліотеки працюють із даними як із тензорами, що робить їх інваріантними до розморності обраних даних.

Для попередньої обробки даних використовувались бібліотеки мови програмування Python такі, як: dlib, OpenCV, тощо, які дали змогу перш за все виділити область, які містять лише обличчя, визначити ключові точки та маштабувати та перевести дані у чорно-білий канал.

Для візуалізації результатів використовувались засоби Python, а також TensorBoard, які дали можливість отримати візуальне зображення побудованих моделей, оцінити значення параметрів у процесі навчання, а також параметрів якості моделі.

Додатково було створено додаток для зручної роботи з моделями та випробування моделей на довільних вибірках.

3.2 Збір та передобробка даних

Для тренування моделей використовувались відео-потоки з бази даних Рейсона, що містить 7356 файли з відео та аудіо-записами 24 акторів (12 жінок та 12 чоловіків), загальним об’ємом 24.8 Гб [35]. З них нас цікавили лише відео, на основі, яких ми і будували моделі. Дана база даних є відкритою для некомерційного використання.

Спершу для зменшення розмірності даних із відео-потоку покадрово виділимо зображення обличчя за допомогою каскаду Хаара, вбудованого у бібліотеку OpenCV.

Тобто з кожного кадру ми виріжемо обличчя і збережемо лише його, як представлено на рисунку 3.1:

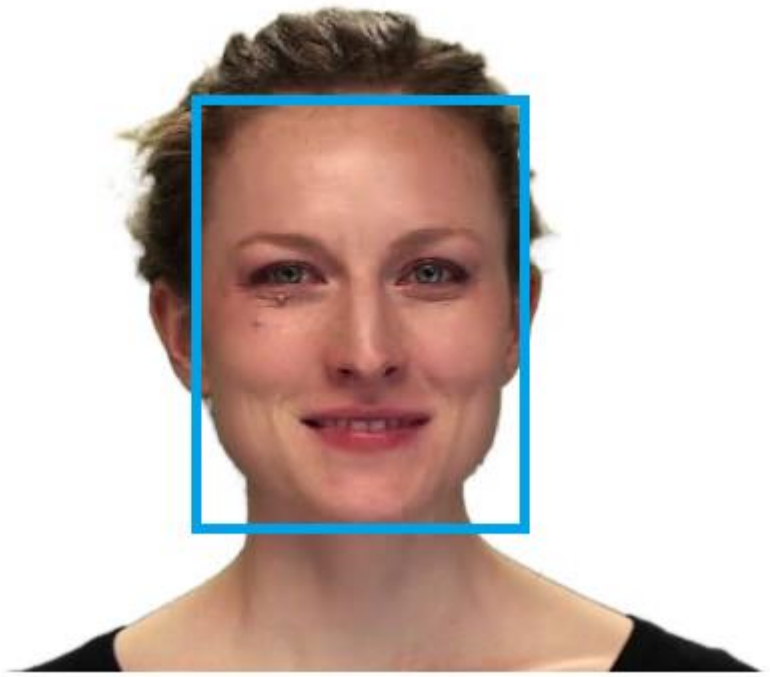

Рисунок 3.1 – Виділення зображення на кадрі каскадом Хаара

Отримані зображення приведимо до єдиного розміру 200 на 200 пікселів.

Наступним кроком було переведення зображення у чорно-білий канал, адже як зазначалось у теоретичній частині це покращить швидкість навчання моделі.

Наступні кроки залежали від виду нейронної мережі.

3.2.1 Передобробка даних для згорткових нейронних мереж

У випадку згорткової нейронної мережі на кадри додатково накладались фільтри, які по суті розмивали зображення, це зробило дані менш вразливими до чіткості отриманих даних. Серед фільтрів використовувалась вирівнююча гісторграма та Гаусове розмиття. Таким чином було отримано зображення, приклад якого представлено на рисунку 3.2:

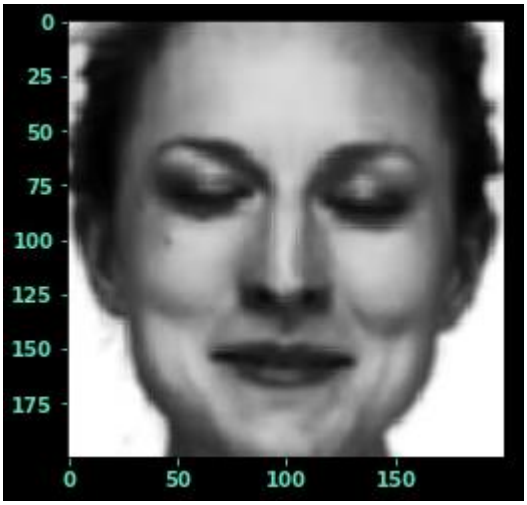

Рисунок 3.2 – Передоброблене зображення для згорткової нейронної мережі

Також для зменшення об’єму даних серед відео були вибрана частина кадрів. Початкове відео мало 30 кадрів у секунду, нами були вибрані по 15 кадрів на відео.

Було отримані кадри, частина яких представлена на рисунку 3.3.

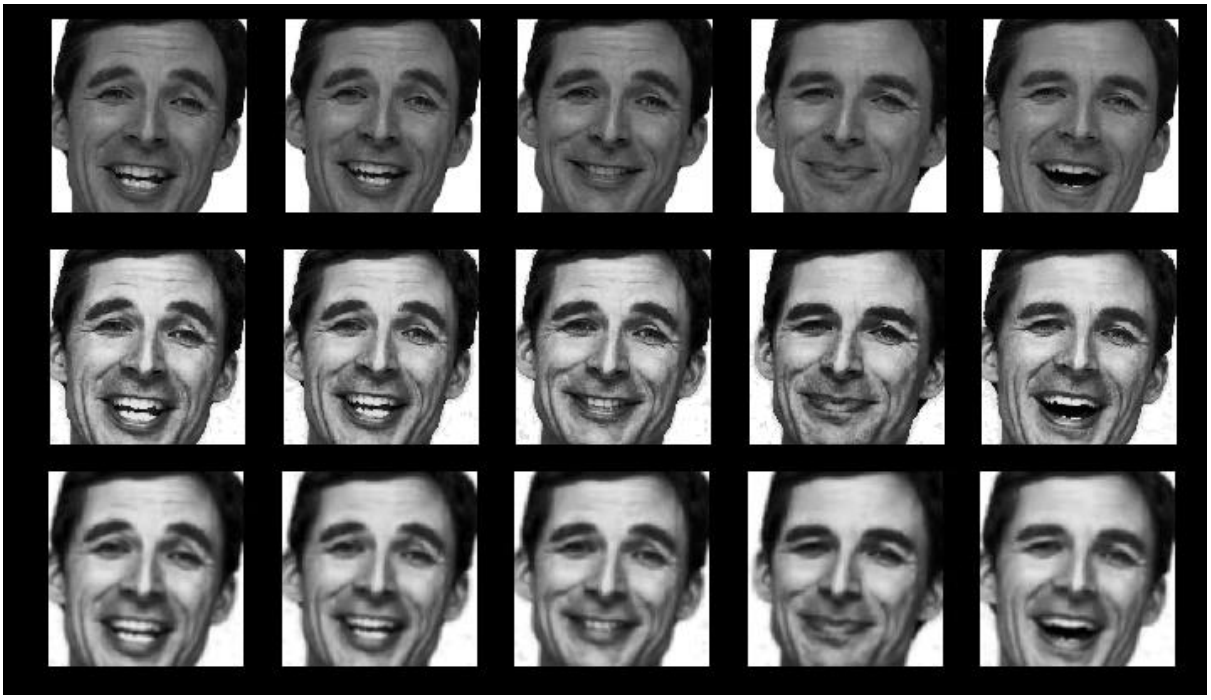

Рисунок 3.3 – Частина передоброблених кадрів

3.2.2 Передобробка даних для нейронної мережі на основі ключових точок

У випадку нейронної мережі, побудованої на основі ключових точок, власне визначались ключові точки для кадрів за допомогою бібліотеки dlib та OpenCV. Всього стандартна бібліотека знаходить 68 ключових точок.

Приклад зноходження ключових точок представлений на рисунку 3.4:

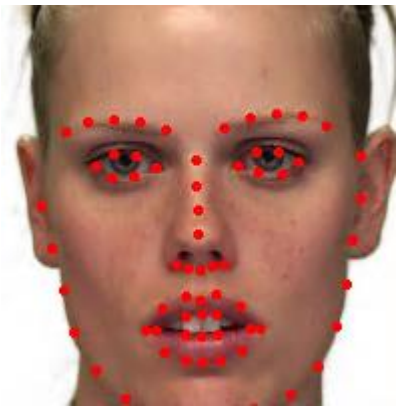

Рисунок 3.4 – Приклад знаходження ключових точок

Після цього відбувалась нормалізація даних для кожної з осей координат по окремості за наступною формулою (3.1):

$$\overline{x_i} = \frac{x_i - \min\limits_{0 \le j \le N} x_j}{\max\limits_{0 \le j \le N} x_j - \min\limits_{0 \le j \le N} x_j}, \tag{3.1}$$

де x_i – координати по осі OX;

N – кількість ключових точок, в нашому випадку $N = 68$.

Таким чином отримувались координати, що знаходились в інтервалі [0,1]. Приклад отриманих значень виведений у вигляді графіку на рисунку 3.5, що відповідає зображеному обличчі вище.

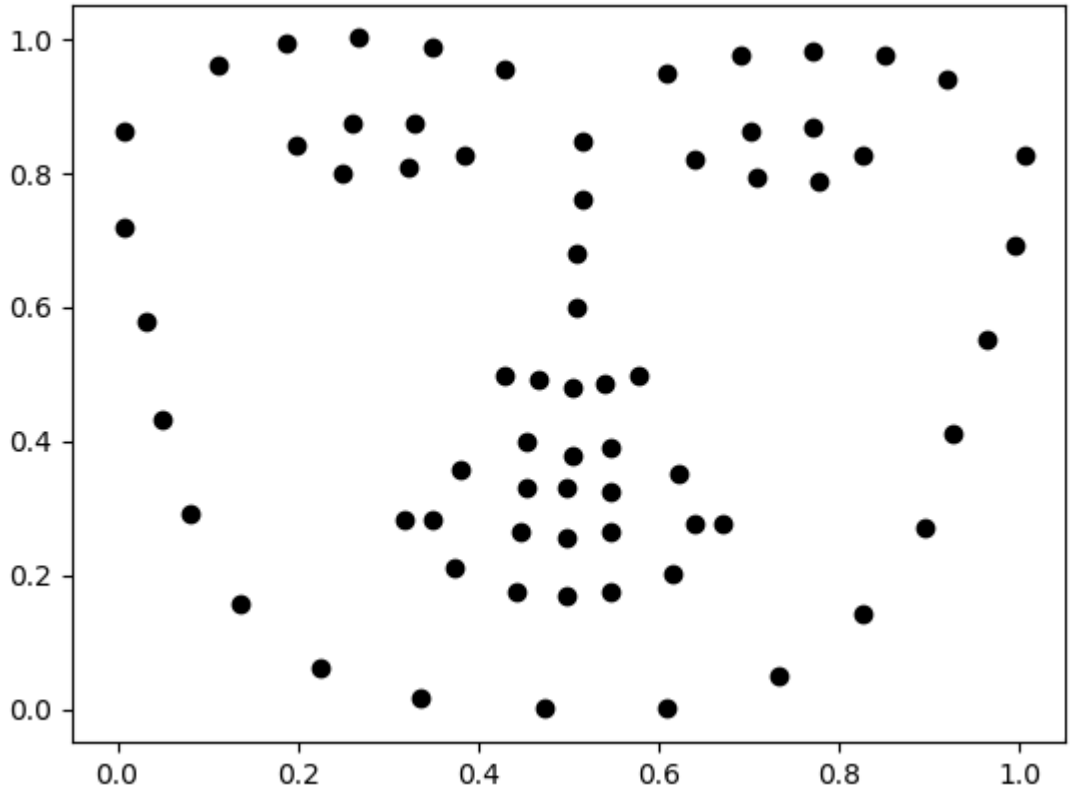

Рисунок 3.5 – Нормалізовані ключові точки

Після цього знаходилась Евклідова відстань між всіма парами, всього було отримано $68 \times 68 = 4624$ значення, які пізніше і подавались на вхід нейронної мережі.

Усі передоброблені дані розбивались на 3 набори: вибірки для тренування та валідації, що містили відповідно 0.75 та 0.25 частини від вибірки, що виділялась на навчання моделі, яка становила 0.8 частини від всієї вибірки. Інша частина вибірки 0.2 – залишалась на тестування моделей на даних, що не брали участі у навчанні моделей.

Відповідний код, написаний на мові програмування Python можна знайти у додатку А.

3.3 Побудова та оцінка якості моделей

Тренування усіх моделей відбувалось, за допомогою пакетного навчання, коли навчання відбувається не на усій вибірці, а на її пакетах. У нашому випадку брались пакети довжини 16. При цьому навчання відбувалось 20 епох. Для оптимізації параметрів моделі використовувався метод Адама. В якості функції для мінімізації бралась функція категоріальної перехресної ентропії, яка вважається ефективною для задач класифікації. Для уникнення перенавчання використовувався метод ранньої зупинки.

Код для побудови усіх моделей наведений у додатку А.

3.3.1 Побудова та оцінка якості моделей на основі ключових точок

Як було сказано у попередньому розділі для побудови даного виду моделей ми будемо виділяти ключові точки та використовувати попарні відстані між ними.

Зібрані дані подавались на вхід нейронним мережам різних архітектур.

Серед нейронних мереж, побудованих на ключових найкращими виявились наступні моделі. Перша модель (далі для зручності Модель 1) містить по 1 згортковому та агрегуючому шарах, шарі нормалізації, 5 повнозв’язних шарів та глобальний агрегуючий шар. Архітектура даної нейронної мережі може бути представлена на рисунку 3.6. Оцінимо якість даної мережі, а саме матрицю помилок, точність, повноту та значення F-міри для даної моделі (див. рис. 3.7).

Для даної моделі, середнє значення точності становить 0.735, а середнє значення повноти 0.772, значення F-міри при $\beta = 1$ становить 0.747. Бачимо, що найбільша похибка моделі при розпізнавані страху та відрази, найвища точність при розпізнванні радості.

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 15, 4)	36996
max_pooling1d_1 (MaxPooling1	(None, 15, 4)	0
batch_normalization_1 (Batch	(None, 15, 4)	16
activation_1 (Activation)	(None, 15, 4)	0
dense_1 (Dense)	(None, 15, 1024)	5120
dense_2 (Dense)	(None, 15, 512)	524800
dense_3 (Dense)	(None, 15, 512)	262656
dense_4 (Dense)	(None, 15, 512)	262656
global_max_pooling1d_1 (Glob	(None, 512)	0
dense_5 (Dense)	(None, 8)	4104

Total params: 1,096,348
Trainable params: 1,096,340
Non-trainable params: 8

Рисунок 3.6 – Архітектура Моделі 1

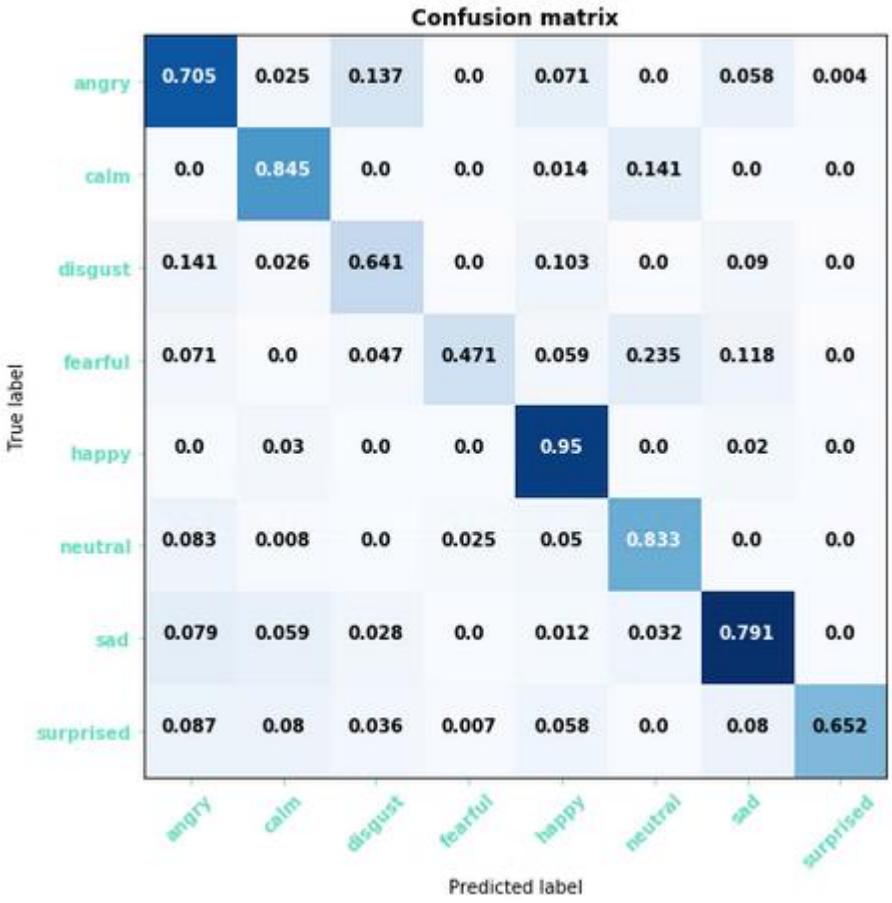

Рисунок 3.7 – Матриця помилок Моделі 1

Також зобразимо процес навчання даної моделі в залежності від епох на рисунках нижче:

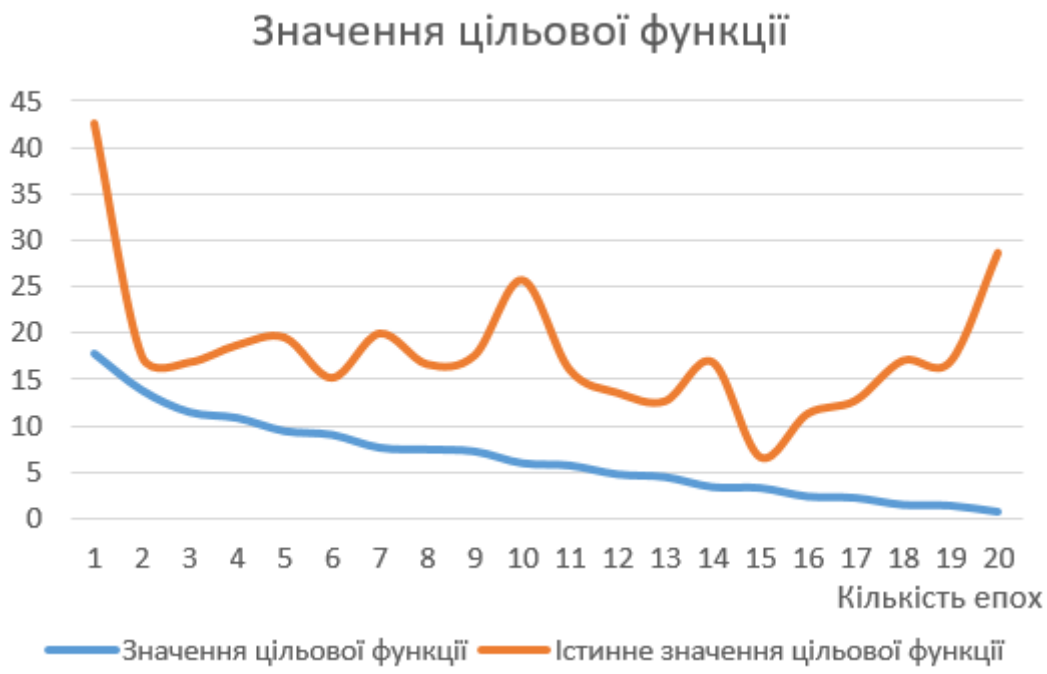

Рисунок 3.8 – Значення цільової функції в залежностві від епох для Моделі 1

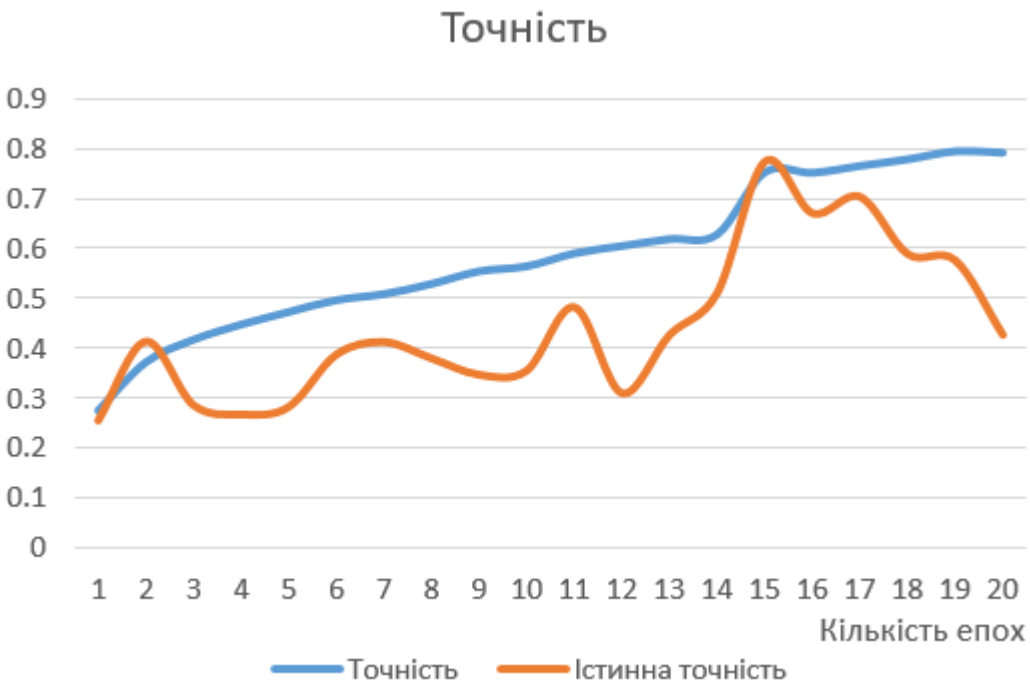

Рисунок 3.9 – Точність Моделі 1 в залежності від кількості епох

На даних рисунках показані значення точності та цільової функції для навчальної вибірки та вибірки для валідації.

Аналогічні дані наведемо для Моделі 2, яка має дещо простішу архітектуру, а саме вона складається з 6 повнозв'язних шарів та глобального агрегуючого шару (див. рис. 3.10):

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 15, 512)           2368000
_________________________________________________________________
dense_7 (Dense)              (None, 15, 512)           262656
_________________________________________________________________
dense_8 (Dense)              (None, 15, 512)           262656
_________________________________________________________________
dense_9 (Dense)              (None, 15, 512)           262656
_________________________________________________________________
dense_10 (Dense)             (None, 15, 512)           262656
_________________________________________________________________
global_max_pooling1d_2 (Glob (None, 512)               0
_________________________________________________________________
dense_11 (Dense)             (None, 8)                 4104
=================================================================
Total params: 3,422,728
Trainable params: 3,422,728
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.10 – Архітектура Моделі 2

Матриця помилок має наступний вигляд, як на рисунку 3.11.

Для даної моделі, середнє значення точності становить 0.739, а середнє значення повноти 0.751, значення F-міри при $\beta = 1$ становить 0.747. Бачимо, що найбільша похибка моделі при розпізнавані страху та відрази, найвища точність при розпізнаванні радості.

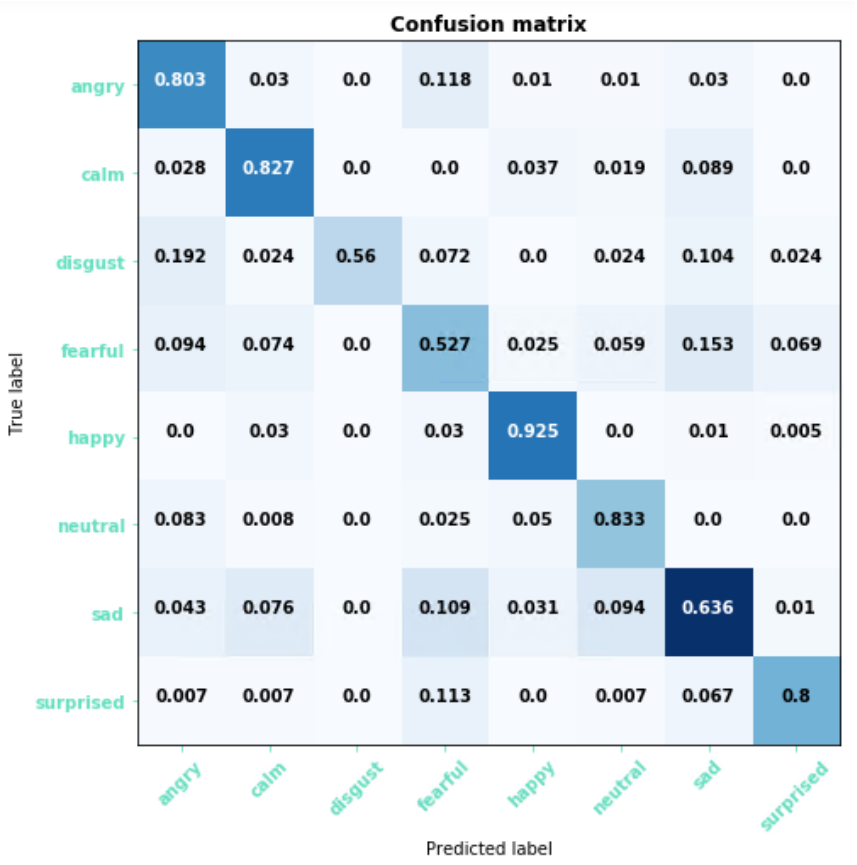

Рисунок 3.11 – Матриця помилок Моделі 2

Також зобразимо процес навчання даної моделі в залежності від епох на рисунках нижче:

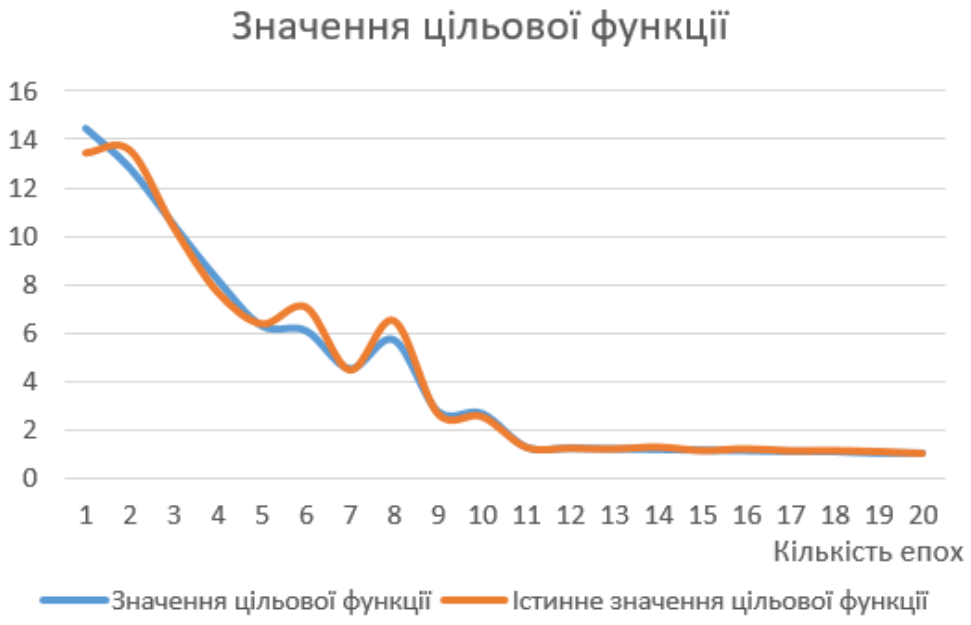

Рисунок 3.12 – Значення цільової функції в залежностві від епох для Моделі 2

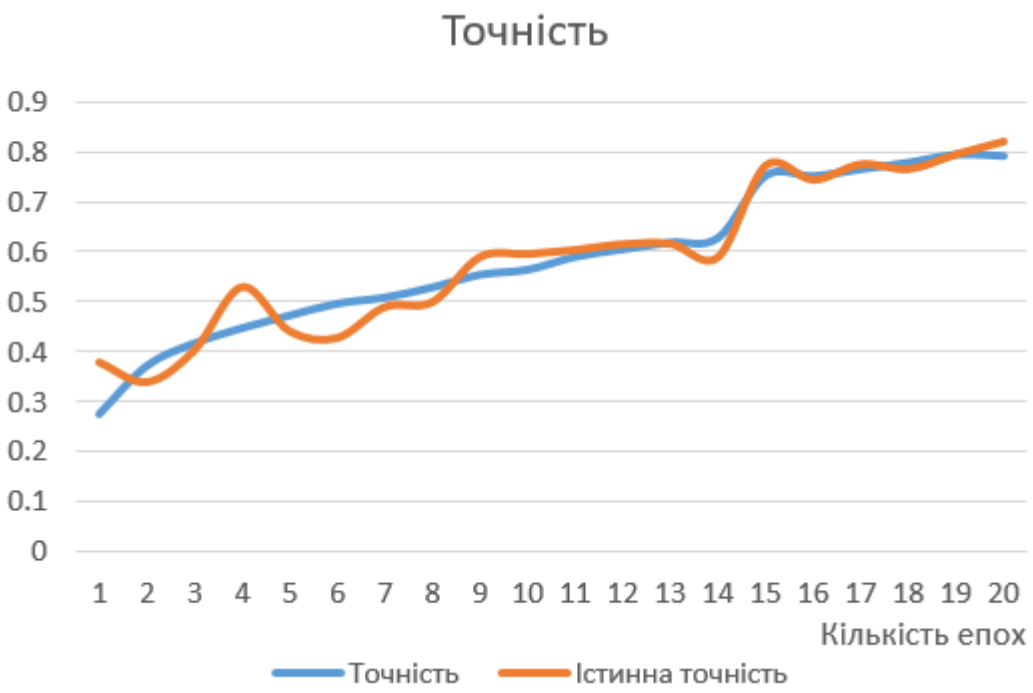

Рисунок 3.13 – Точність Моделі 2 в залежності від кількості епох

Бачимо, що побудована модель практино не перенавчалась протягом епох, на відміну від першої моделі, оптимальні значення, якої досяглись на 15-тій епосі.

Окрім наведених моделей було побудовано чимало інших моделей, проте їх точність була на вище 0.6, тому вони вважаються навдалими, а тому не наведені у роботі, проте код, для їх побудови можна знайти у додатку А.

3.3.2 Побудова та оцінка якостей згорткової нейронної мережі

На основі передоброблених даних було здійснено побудову згорткових нейронних мереж з різними архітектурами. Кожна архітектура містить згортковий шар, завдання якого є виділити ознаки. Виглядає це наступним чином, представленим на рисунку 3.14.

Було побудовано кілька моделей із різними архітектурами. Діаграми усіх моделей наведено у додатку Б. Тут наведемо лише найкращі з побудовиних моделей моделей та обґрунтуємо процес її побудови.

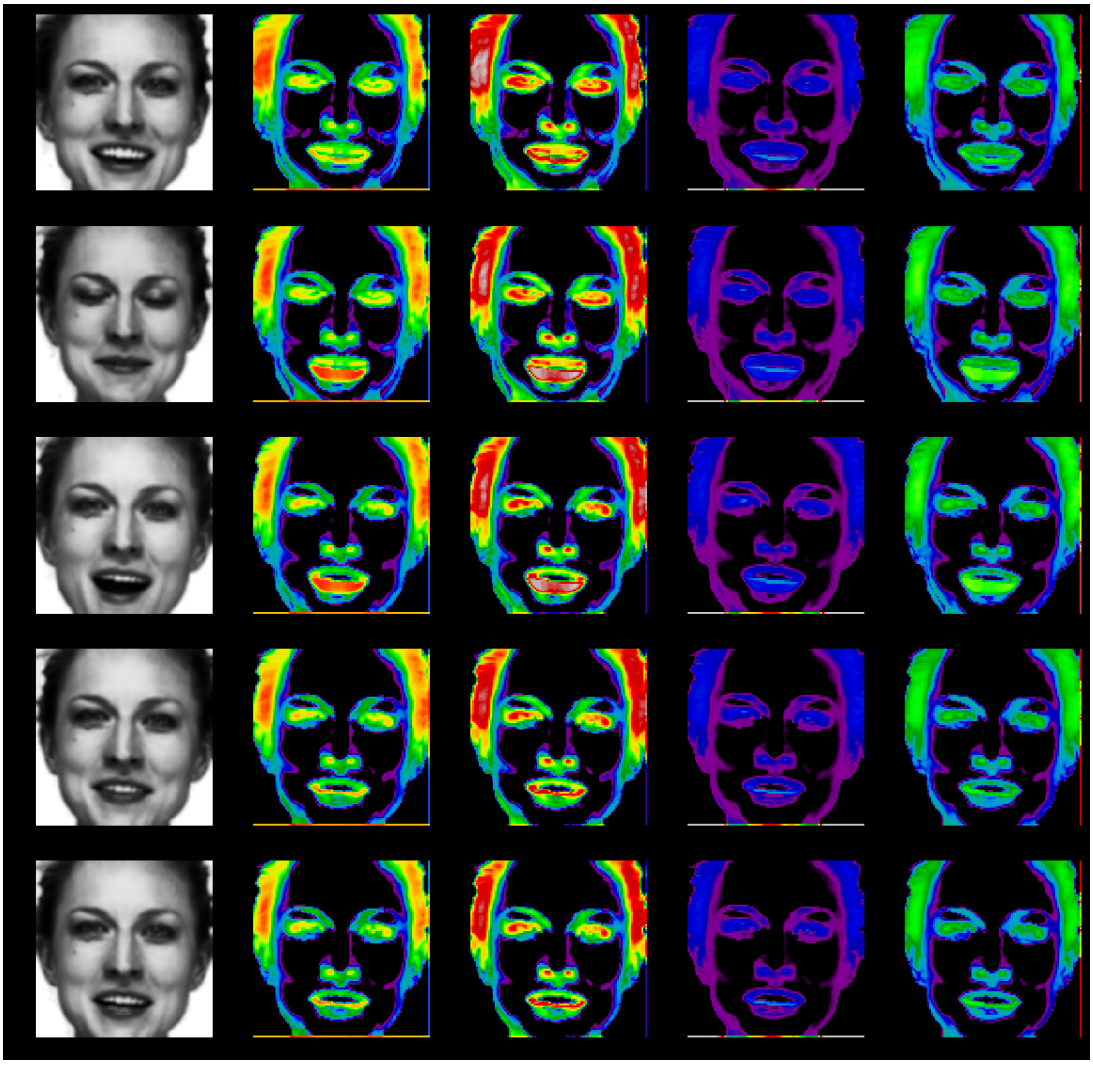

Рисунок 3.14 – Карти ознак

А саме було побудовано модель, зображену на рисунку 3.15, що містить 6 згорткових шарів та відповідно агрегуючі шари після них. Після 2-гого згорткового шару виділення карт ознак відбувається паралельно на 4, 5 та 3 шарах відповідно, після чого отримані ознаки сумуються. Після цього ми маємо ще один згортковий шар, та ряд агрегуючих шарів.

Дана архітектура використовує згорткові шари для виділення характерних ознак, та агрегуючі шари для зменшення розмірності даних. На виході ми маємо стандартних для даного виду нейронний мереж шар, а саме повнозв’язний шар, після якого ми уже і отримуємо 8 вихідних значень, які відповдають 8 класам емоцій, які повинна розпізнати модель.

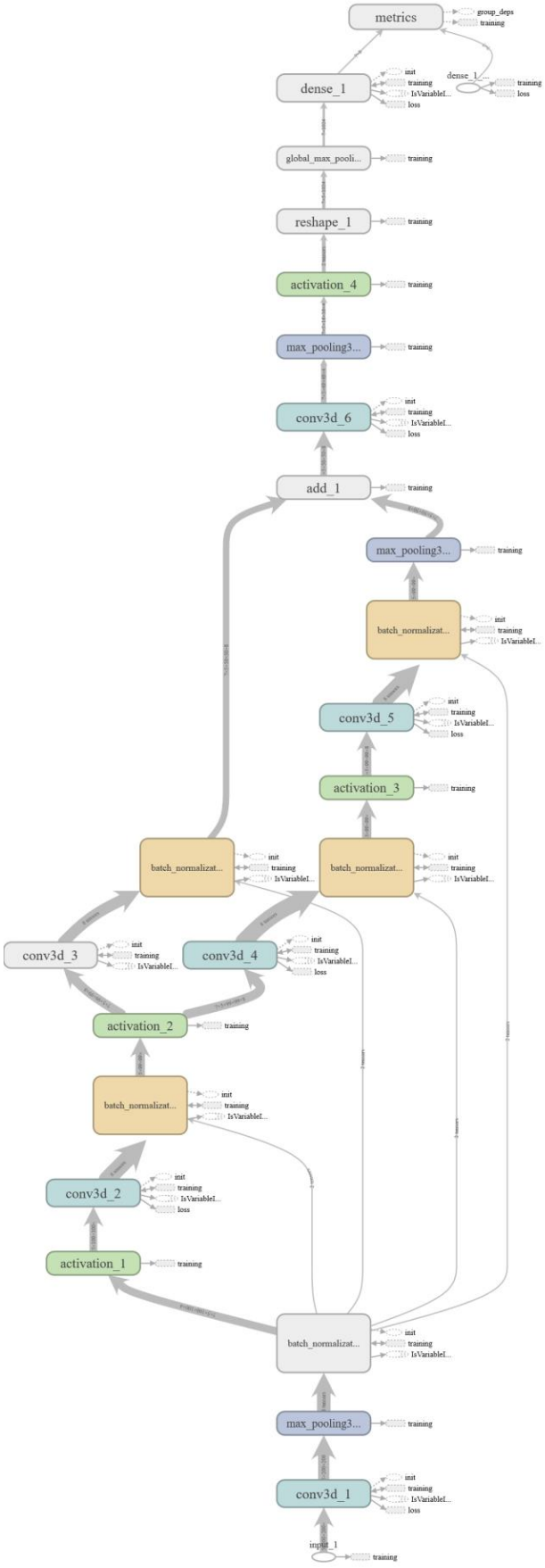

Рисунок 3.15 – Архітектура моделі згорткової нейронної мережі (Модель 3)

Більш детальну архітектуру моделі можна отримати з додатку Б.

Дана модель є найбільш вдалою з розроблених з точки зору точності та повноти моделі. Тому представимо матрицю помилок для даної моделі на рисунку нижче:

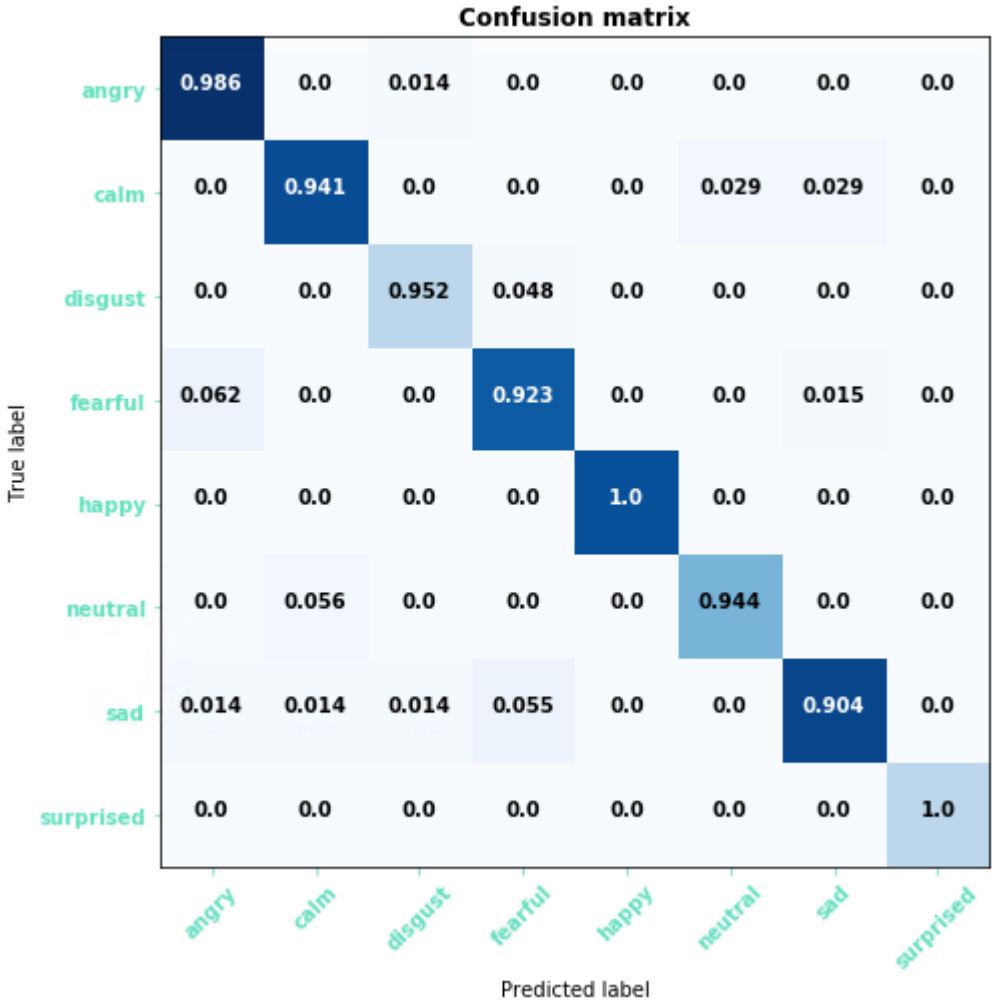

Рисунок 3.16 – Матриця помилок Моделі 3

Таким чином середня точність отриманої моделі становить 0.957, середня повнота – 0.953, значення F-міри при $\beta = 1$ – 0.955.

Для даної мережі представимо процес її навчання у залежності від епох на рисунках нижче:

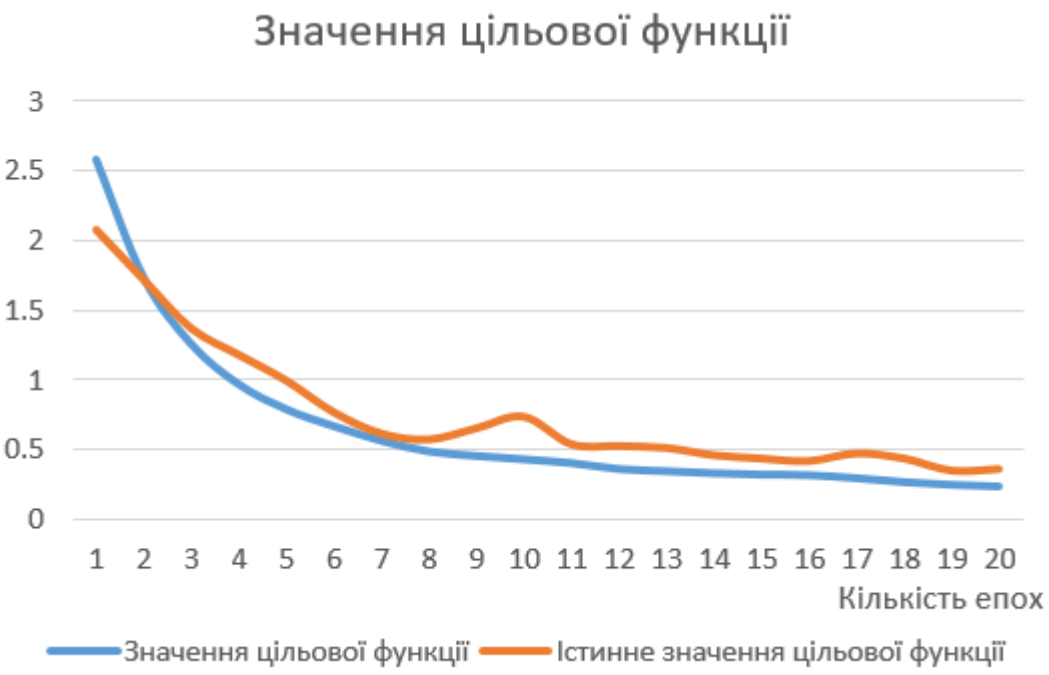

Рисунок 3.17 – Значення цільової функції в залежностві від епох для Моделі 3

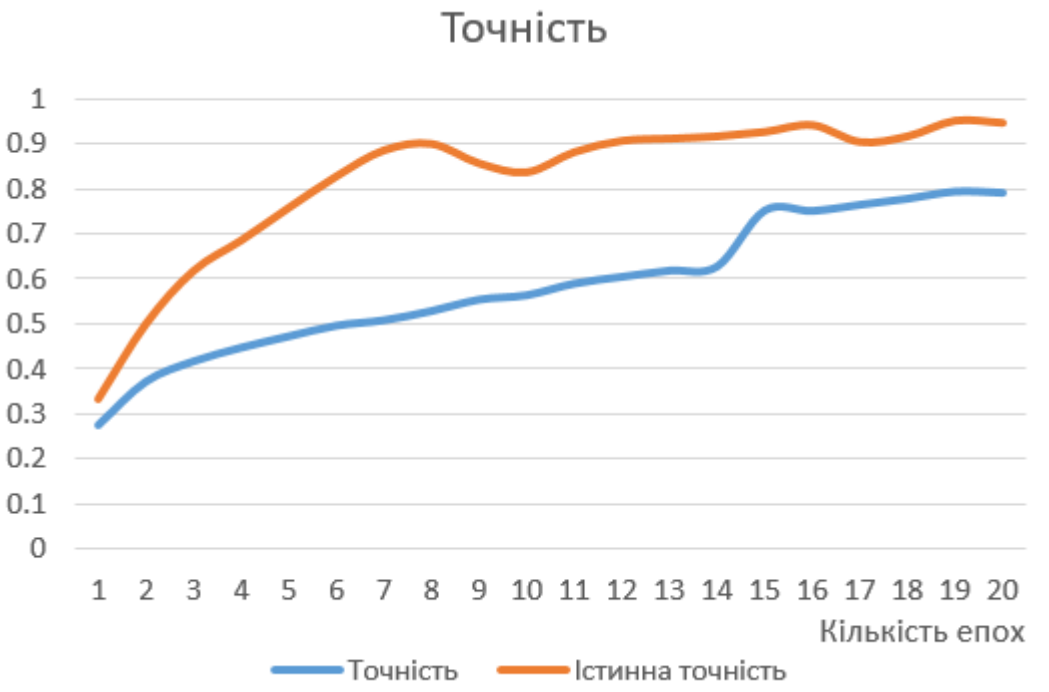

Рисунок 3.18 – Точність Моделі 3 в залежності від кількості епох

Як бачимо завдяки шарам нормалізації дана мережа не зазнала перенавчання і були отримані високі значення точності.

Аналогічно проаналізуємо наступні згорткову моделі. Їх архітектури розмістимо у додатку Б через їх громіздікість. Тут лише зазначимо, що наступна згорткова модель (далі Модель 4) має таку архітектуру: спершу йде 2 згорткових шари, після чого ми розділяємо дані на 2 гілки, перша містить теж 2 згорткових шари, інша – 1 згортковий шар, після чого додавшись дані проходять через ще 1 згорткових шар, а потім знову розділяютья прохоячи по 1 шару по окремості, після чого дані з'єднуються та проходять через повнозв'язний та глобальний агрегуючий шар. Загалом 8 згорткових шарів, після кожного з яких йде по агрегуючому шарі та шарі нормалізації.

Дана модель має дещо гіршу точність у порівнянні з попередньою, не зважаючи на більш складну архітектуру. Наведемо зматрицю помилок для даної моделі, як на рисунку нижче:

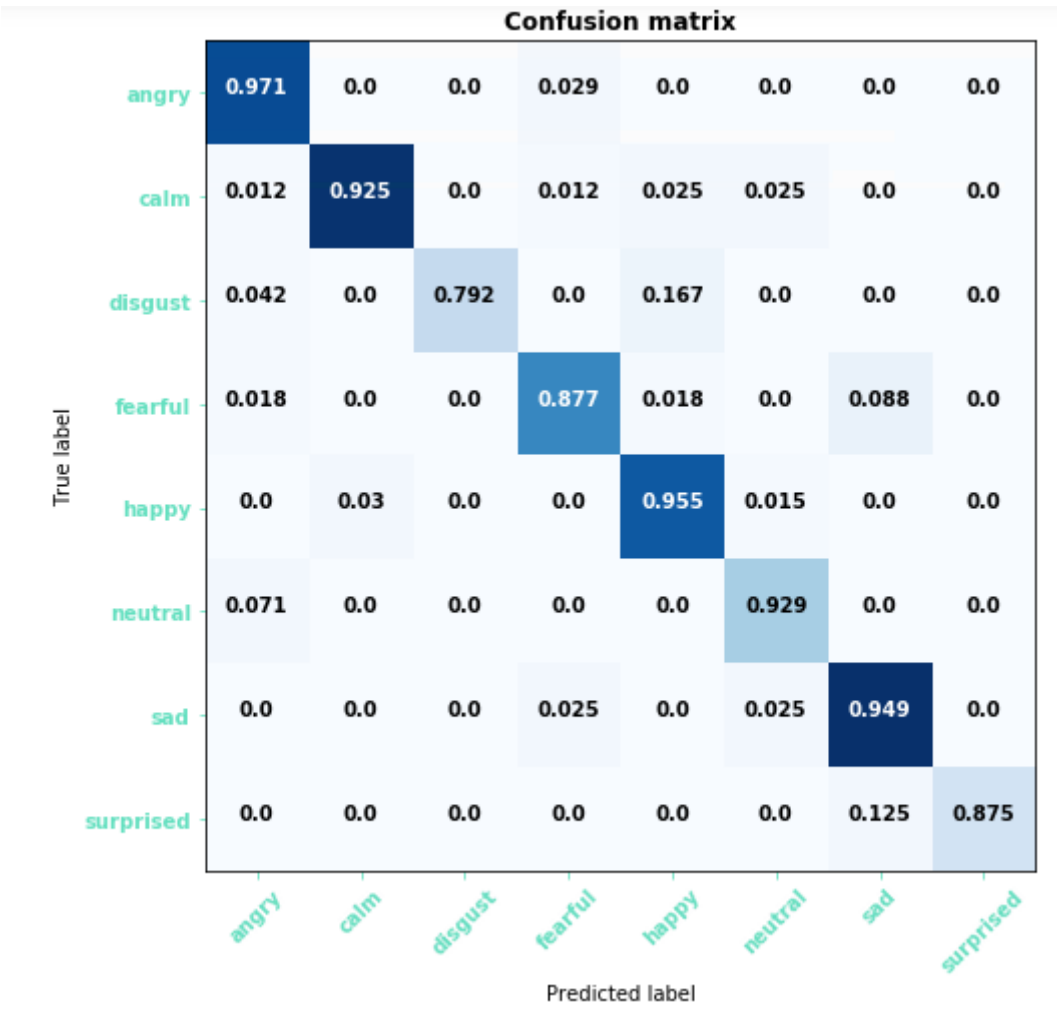

Рисунок 3.19 – Матриця помилок Моделі 4

Таким чином середня точність отриманої моделі становить 0.909, середня повнота – 0.933, значення F-міри при $\beta = 1$ – 0.919.

Для даної мережі представимо процес її навчання у залежності від епох на рисунках нижче:

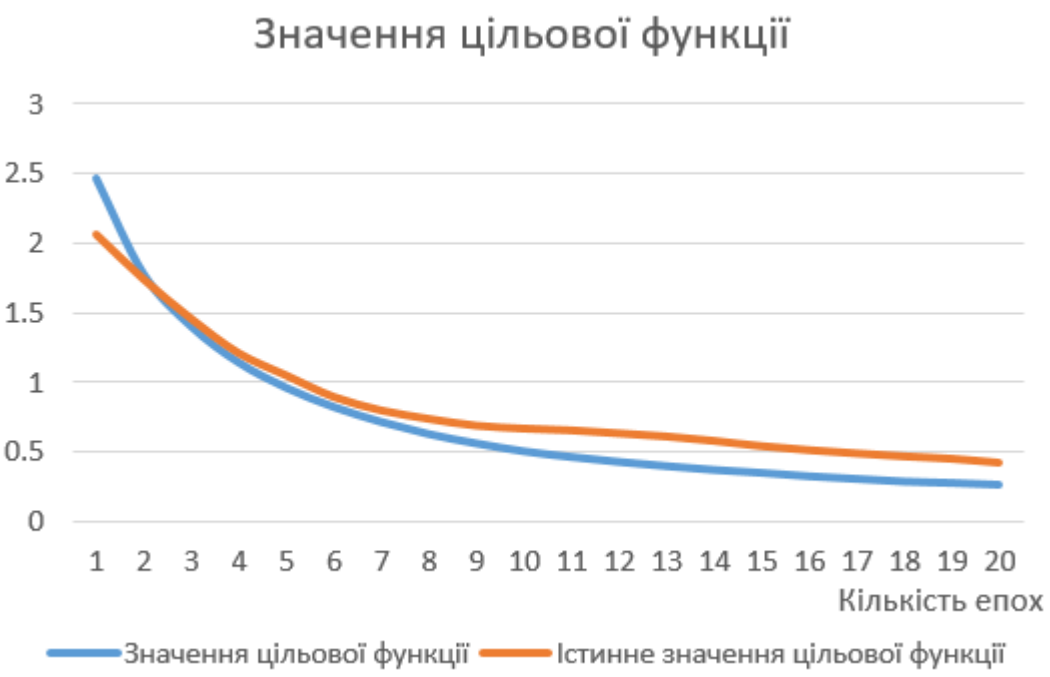

Рисунок 3.20 – Значення цільової функції в залежностві від епох для Моделі 4

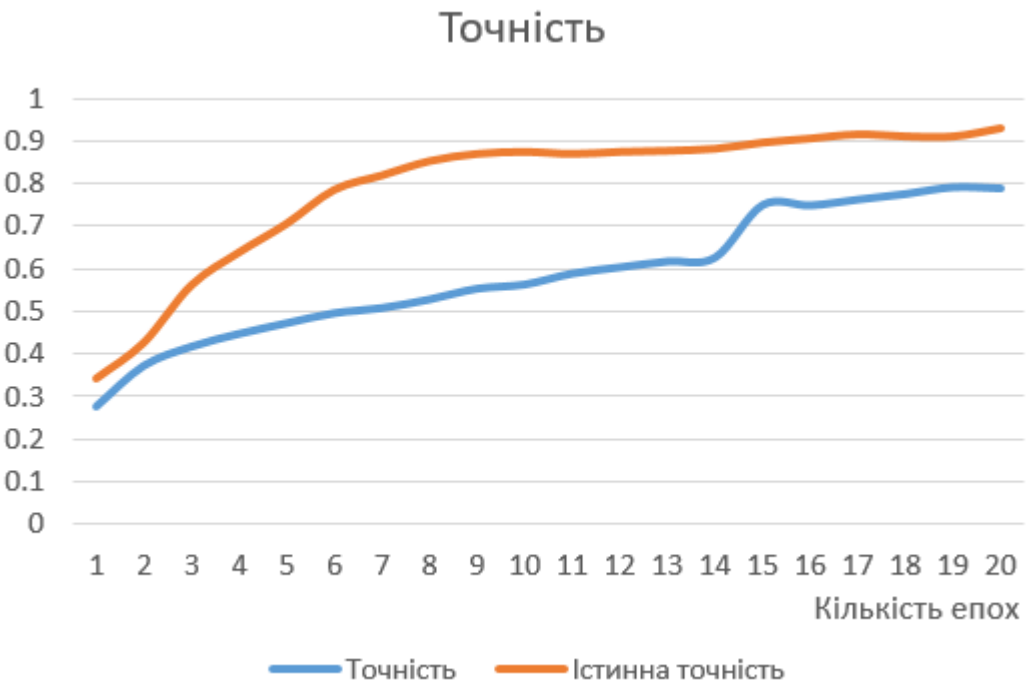

Рисунок 3.21 – Точність Моделі 4 в залежності від кількості епох

Остання 5-та модель складається з 1 згорткового шару, після якого ми маємо 2 гілки, перша має ще 2 згорткових шари, друга – 1. Після чого 2 повторюється наступна структура: дані з’єднуються, потім розділяються на 2 гілки, кожна з яких має по 1 згортковому шарі, а потім дані знову з’єднуються, після чого подаються на повнозв’язний шар. Як завжди після кожного згорткового шару йде агрегуючий та шар нормалізації. Таким чином загалом модель містить 8 згорткових шарів, 8 агрегуючих, 8 нормалізуючий, 1 повнозв’язний та 1 глобальної агрегації.

Дана модель має наступну матрицю помилок, як на рисунку нижче:

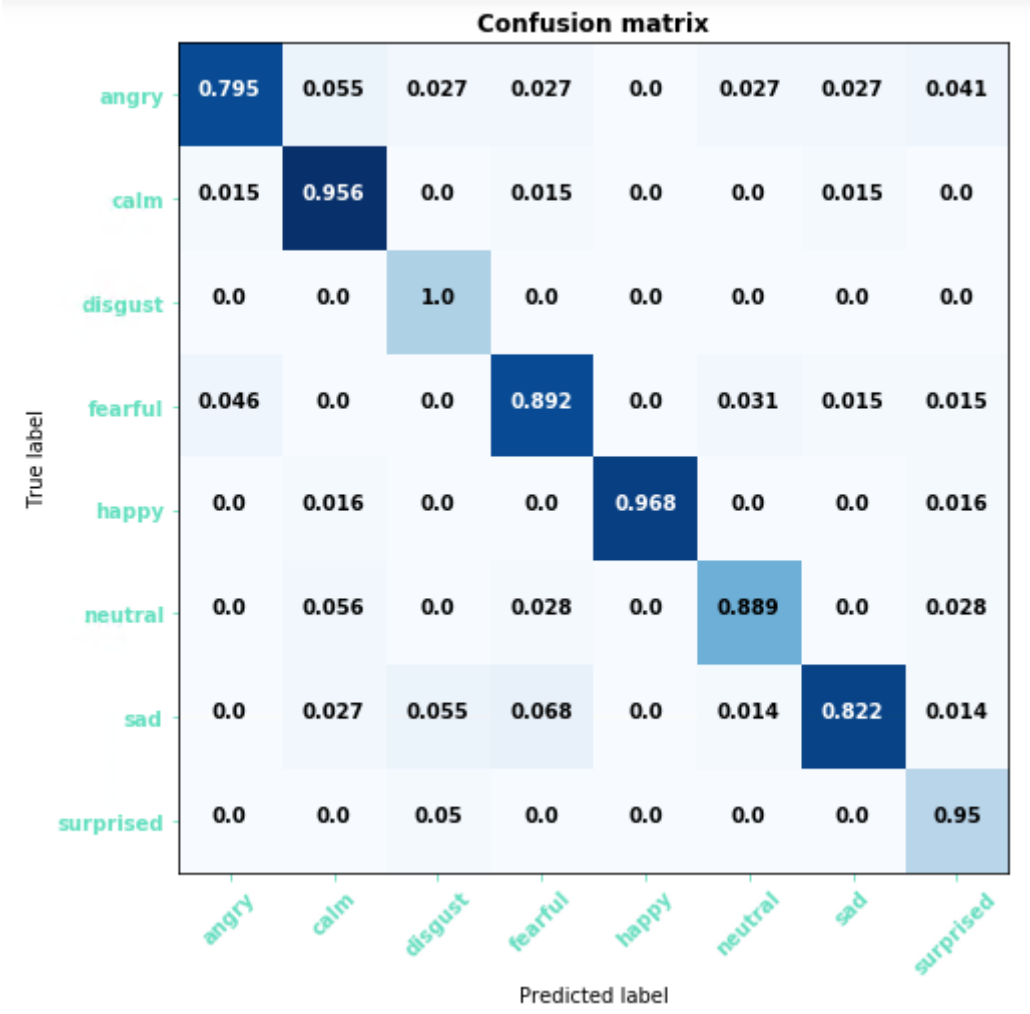

Рисунок 3.22 – Матриця помилок Моделі 5

Середня точність отриманої моделі становить 0.908, середня повнота – 0.870, значення F-міри при $\beta = 1$ – 0.886.

Для даної мережі представимо процес її навчання у залежності від епох на рисунках нижче:

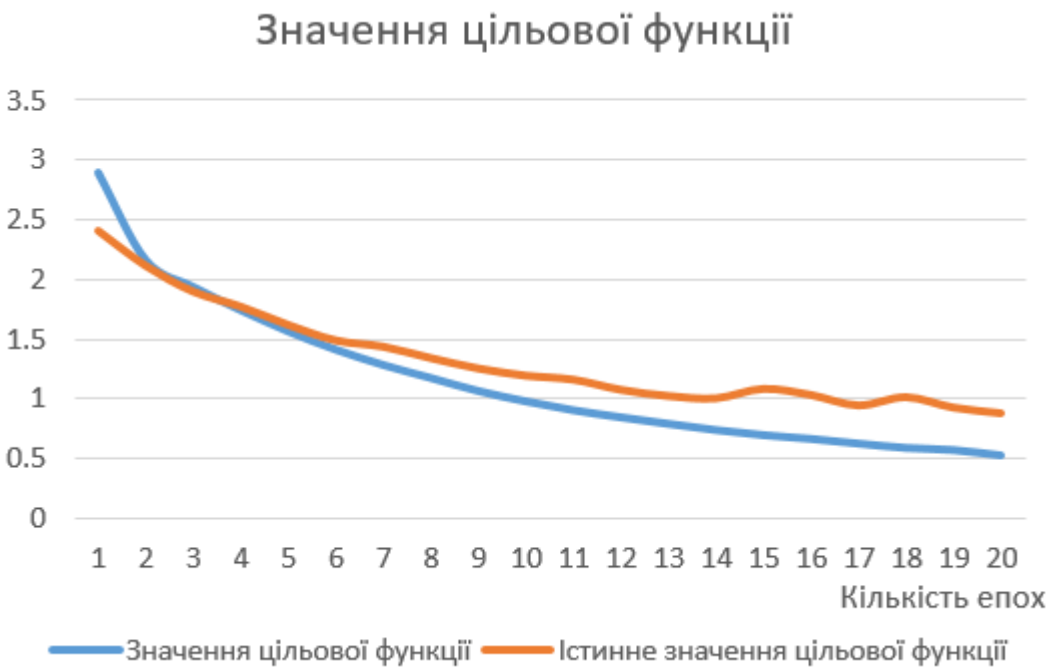

Рисунок 3.23 – Значення цільової функції в залежностві від епох для Моделі 5

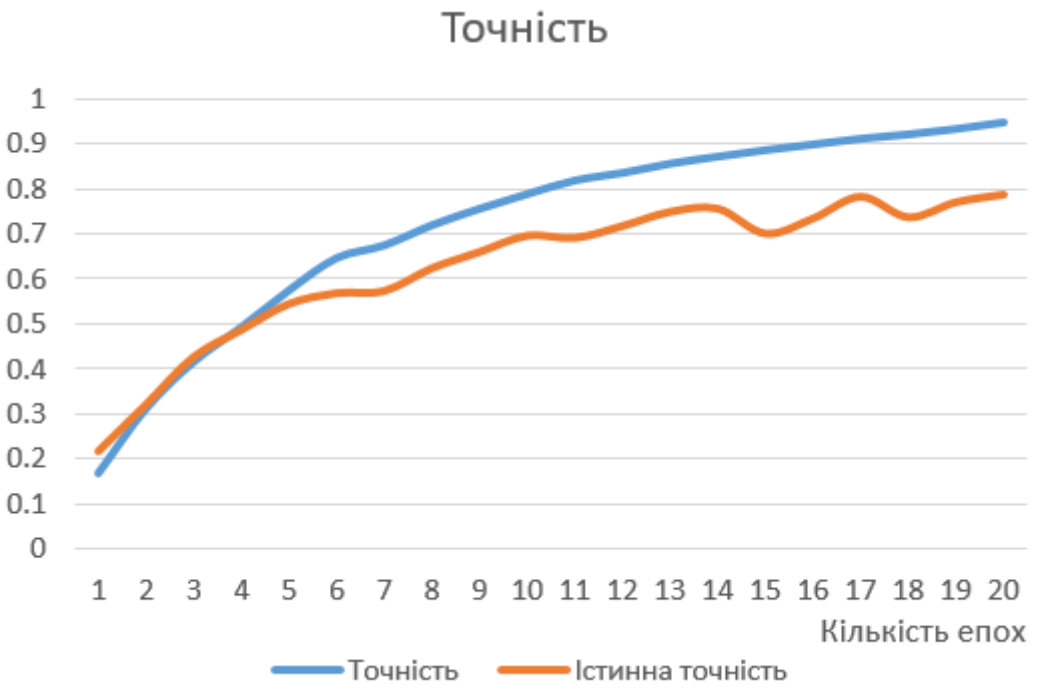

Рисунок 3.24 – Точність Моделі 5 в залежності від кількості епох

3.4 Порівняння якості побудованих моделей

В даному підпункті ми порінняємо основні показники моделей, а саме: їх точність, повноту, F-міри при $\beta = 1$, матриці похибок та швидкість розпізнавання. Для цього будемо використовувати дані отримані після навчання та дані, отримані в ході тестування на даних, що не брали участі у навчанні. Усі необхідні скрипти можна знайти у додатку А.

Для цього з початкової вибірки було виділено 1000 відео, які попередньо не формувались у hd5-файли, як у випадку навчання, а безпосередньо подавались на вхід коду, який повинен їх розпізнати. Таким чином отримувались файли, які зберігали шлях до файлу, початкове та оцінене значення емоції, а також час витрачений на передобробку та власне розпізнавання. Усі експерименти проводились на віртуальній машині з наступними параметрами: 8 vcpu, 32 Гб оперативної пам’яті та SSD диском.

Крім цього визначався середній час, необхідний на завантаження моделі, кількість параметрів моделі та її розмір.

Отримані значення зведемо у таблиці, представлені нижче:

Таблиця 3.1 – Оцінка точності моделей

	Точність (навч)	Повнота (навч)	F-міра (навч)	Точність	Повнота	F-міра
Модель 1	0.735	0.772	0.748	0.701	0.711	0.705
Модель 2	0.738	0.751	0.747	0.682	0.709	0.687
Модель 3	0.956	0.953	0.955	0.872	0.891	0.878
Модель 4	0.909	0.933	0.919	0.778	0.836	0.816
Модель 5	0.908	0.870	0.886	0.789	0.834	0.819

Таким чином з точки зору точності найкращими є згорткові моделі, а саме Модель 3. Варто також зазначити, що час витрачений на навчання згорткових мереж також значно вищий, ніж час навчання нейронних мереж.

У наступній таблиці представлено порівняння часу роботи моделей та їх розмір.

Таблиця 3.2 – Оцінка швидкодії та розміру моделей

	Час загрузки	Час обробки	Час моделі	Час розпізнавання	Кількість параметрів	Розмір моделі (Mb)
Модель 1	1.244	2.372	0.010	2.382	1096348	10.12
Модель 2	1.924	2.360	0.009	2.369	3422728	15.072
Модель 3	4.803	1.279	0.139	1.418	9896	0.204
Модель 4	4.481	1.288	0.144	1.432	6680	0.185
Модель 5	4.253	1.286	0.147	1.433	3592	0.169

Як бачимо нейронні мережі на основі ключових точок загружаються швидше, не зважаючи на їх більшу кількість параметрів, а відповідно і більший розмір. Це пов’зано з більш складною архітектурою згорткових нейронних мереж. При цьому час на передобробку відео у нейронних мереж на основі ключових точок майже у 2 рази більший, ніж для згорткових нейронних мереж, а час на роботу самої моделі навпаки у згорткових нейронних мереж значно вищий. Незважаючи на це, загальний час розпізнавання найкращим є для Моделі 3.

Таким чином, на повну обробку 1 відео, разом із загрузкою моделей становить 3.5 – 4 с для Моделей 1 та 2, та 5.6 – 5.8 для інших моделей, проте по мірі зростання кількості відео, які будуть оброблятись 1 запитом, ми розуміємо, що час загрузки моделі не є суттєвим.

Таким чином, якщо перед нами буде стояти завдання побудувати систему розпізнавання емоцій в реальному часі, то Модель 3 є найкращим варіантом, а саме це модель згорткової нейронної мережі, архітектуру якої можна знайти у попердньому підпункті.

3.5 Тестування моделей

Для тестування моделей було створено додаток, за допомогою мови програмування R, який містить 2 вкладки. Перша використовується для пакетного тестування. Ми можемо загрузити файл, що містить шлях до файлів та протестувати їх на вибраних моделях. Результати будуть збережені у окремих файлах. При тестуванні вибірок запускався код на Python, який виконував необхідну переобробку відео та власне розпізнавання, після чого формував файли з результатами. Дані результати подавались на вхід звіту, написаному з використанням Power BI Desktop.

Інтерфейс першої вкладки представлений на рисунку 3.25.

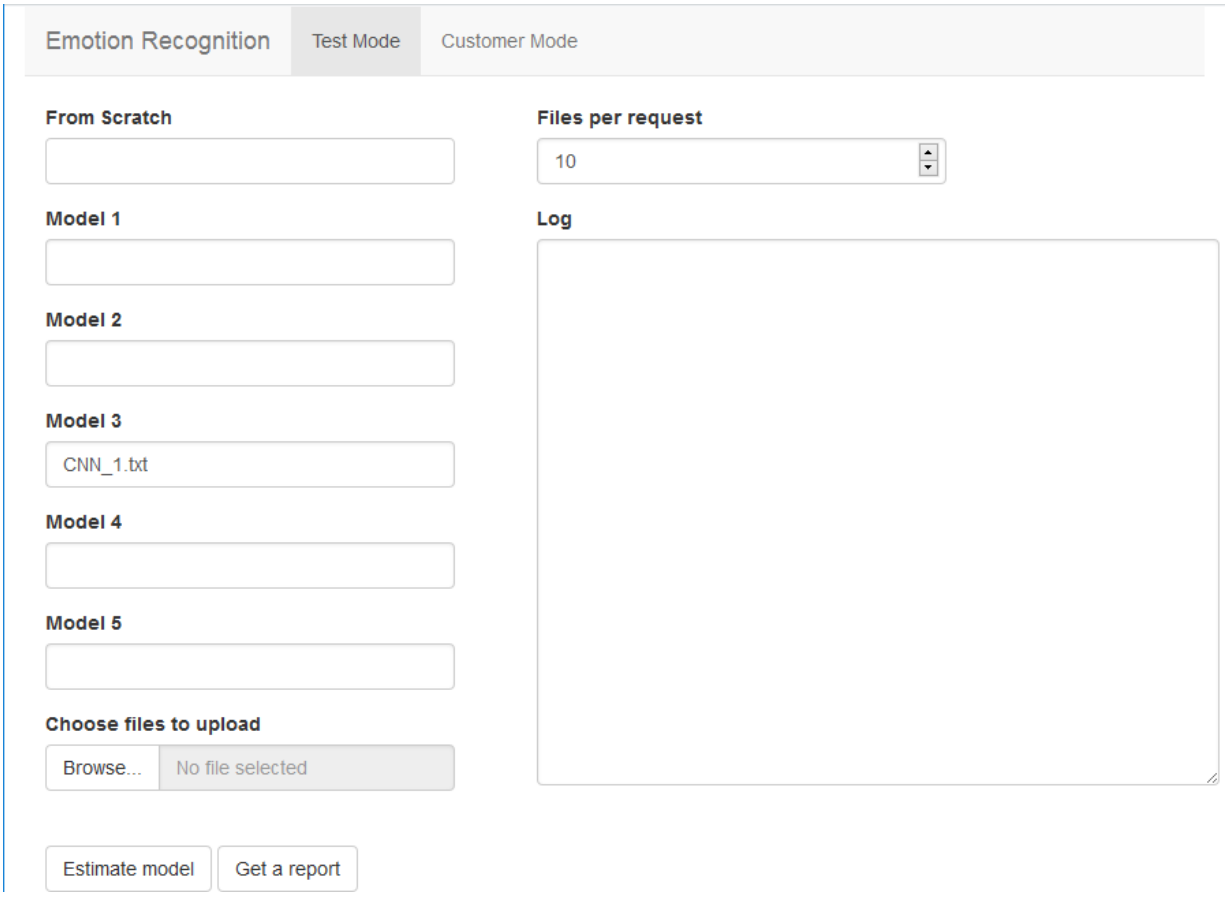

Рисунок 3.25 – Інтерфейс вкладки для пакетного тестування

Приклад звіту наведений на рисунку 3.26:

Рисунок 3.26 – Приклад звіту для аналізу моделей

Також користувач має змогу проаналізувати кілька вибраних відео файлів або використати відеокамеру для визначення емоції у реальному часі. У першому випадку користувач отримує вікно, в якому представлені вибраний файл, який можна переглянути, а також результати, отримані прогнозом обраних моделей. Приклад отриманих результатів можна побачити на рисунках 3.27 – 3.28:

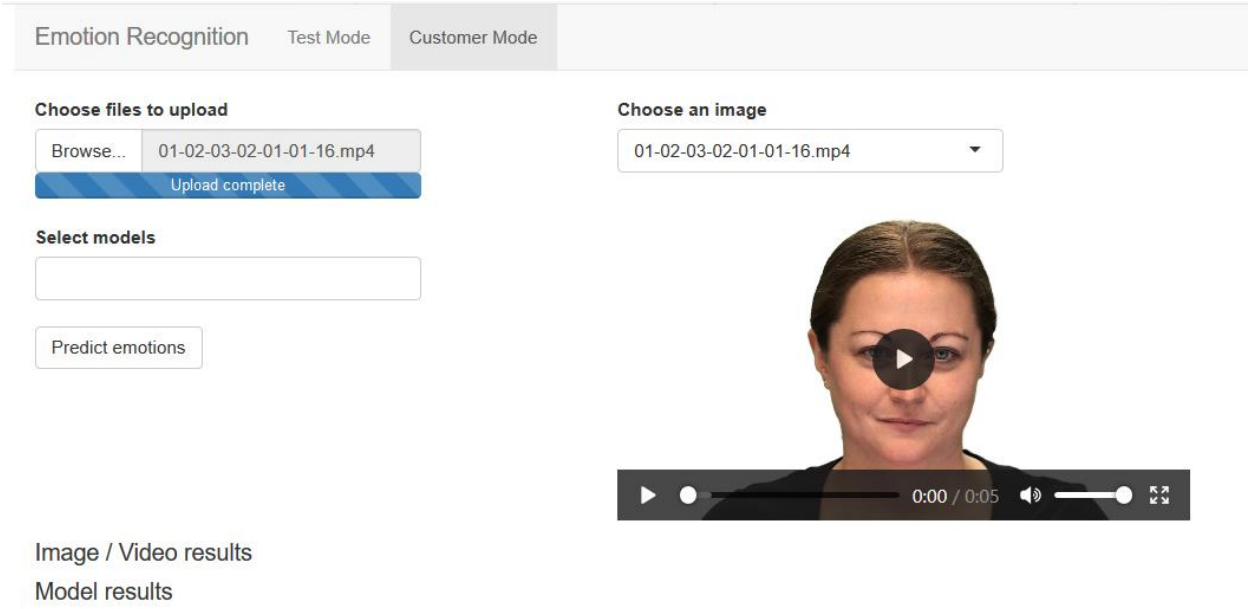

Рисунок 3.27 – Інтерфейс вкладки для детального тестування

Image / Video results

Model	angry	calm	disgust	fearful	happy	neutral	sad	surprised	max	Emotion
Model 3	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	1.00	happy
Model 4	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	1.00	happy

Model results

Model 3 | Model 4

File name	angry	calm	disgust	fearful	happy	neutral	sad	surprised	max	Emotion
01-02-03-02-01-01-16.mp4	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	1.00	happy
01-02-03-02-01-01-16.mp4	0.00	0.02	0.00	0.00	0.97	0.00	0.00	0.00	0.97	happy
01-02-03-02-01-01-16.mp4	0.00	0.31	0.00	0.00	0.60	0.00	0.09	0.00	0.60	happy
01-02-	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	1.00	happy

Рисунок 3.28 – Результати роботи моделей для вибраного файлу

Висновки до розділу 3

В даному розділі було здійснено побудову моделей розпізнавання емоцій. Для цього було визначено основні засоби для моделювання, здійснено збір та передобробку вибірки для покращення результатів навчання та пришвидшення процесу навчання.

Наступним кроком була власне побудова моделей на основі нейронних мереж. Побудовані моделі було проаналізовано шляхом визначення їх точності, повноти, часу роботи та розміру.

Для зручного тестування моделей була написана програма зі зручним інтерфейсом, що містить вкладку для пакетного тестування, з візуальним звітом та вкладка для детальної обробки невеликої кількості відеофайлів.

Згідно отриманих результатів згорткові нейронні мережі давали кращі результати стосовно точності, повноти та розміру. Єдине в чому кращими виявились нейронні мережі на основі ключових точок – це час завантаження моделі та час навчання моделі. Дані показники не є більш суттєвими, адже час завантаження моделі витрачається 1 раз в умовах класифікації емоцій по відеопотоку у реальному часі, а навчання моделі відбувається меншу кількість раз у порівнянні із її використанням.

Таким чином, ми робимо висновок, що згорткові нейронні мережі виявились кращими. А зокрема, найкращею виявилась Модель 3 – згорткова нейронна мережа, що містить 6 згорткових, 6 агрегуючих та нормалізуючих шарів та 2 повнозв’язних шарів. Точність даної моделі становить 0.956 при навчанні та 0.872 при тестуванні, повнота – 0.953 при навчанні та 0.891 при тестуванні, дана модель витрачає 1.418 с на обробку 1 відеофайлу та 4.803 с на завантаження. При цьому отримана модель має 9896 параметрів та займає 0.204 Mb.

РОЗДІЛ 4 РОЗРОБКА СТАРТАПУ

У даному розділі проводиться оцінка основних характеристик програмного.

Головним завданням даного розділу є встановлення економічної вигоди даної роботи, її можливість реалізувати у вигляді стартапу та визначення основних напрямків проекту. Для цього нам необхідно провести маркетинговий аналіз, що передбачає виконання нижченаведених підпунктів.

4.1 Опис ідеї проекту

Таким чином перш за все, ми опишемо основні ідеї проекту, що передбачають визначення:

- змісту ідеї;
- можливих способів використання;
- основної цінності для потенційного користувача;
- порівняння продукту з існуючими аналогами та замінниками.

Отже, визначемо три перші пункти за допомогою таблиці 4.1, яке сформує уявлення про те,у чому полягатиме наш продукт, яку користь він нестиме користувачам, а відповідно ми зрозуміємо, яким користувачам даний продукт був б цікавим.

За допомогою наступної таблиці 4.2 ми визначимо потенційних конкурентів нашого продукту, а також порівняємо їх основні характеристи, а відповідно і техніко-економічні переваги нашої ідеї.

У якості конкурентів було обрано продукти під назвами: MeetingPulse (Конкурент 1) та Voicera+Wrappup (Конкурент 2). У додатку В можна отримати схеми бізнес-моделей даний продуктів. Тут лише відзначимо, що обидва продукти націлені

півщити ефективність проведення зустрічей шляхом збільшення зацікаленості її учасників. Перший конкурент це здійнює за допомогою проведення опитувань, другий ж робить це на основі аналізу голосових повідомлень учасників зустрічі.

Таблиця 4.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка продукту для оцінки ставлення аудиторії до виступу за допомогою проведення емоційної аналітики	1. При зустрічах, де один виступаючий і велика або середня аудиторія	Зменшення витрати на зустрічі, шляхом покращення контенту зустрічей, підвищення їх якості
	2. При зустрічах, де ведуться дискусії	Покращення якості зустрічей, шляхом виявлення кращих рішень через реальне ставлення учасників зустрічей
	3. При зустрічах один на один	Допомагає при підборі кадрів, шляхом оцінки їх емоційного стану та ставлення до майбутньої роботи

Таблиця 4.2. – Визначення характеристик ідеї проекту

Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
	наш проект	конк 1	конк 2			
Простота використання	Середня простота – вимагає встановлення і додаткового обладнання	Простий у встановленні	Простий у встановленні	(W) Наш продукт програє по відношенню до товарів конкурентів, адже його встановлення вимагає додаткового обладнання		
Точність результатів	Висока	Висока	Низька	(S) Точність системи висока, тому і відповідно якість результатів висока		

Продовження таблиці 4.2

Спосіб оплати	Гнучкий	Негнучкий	Негнучкий	(S)Ми пропонуємо різні способи оплати і клієнти можуть підібрати найкращий для себе, тоді як розглянуті конкуренти пропонують лише 1 спосіб оплати
Велечина сектору клієнтів	Середній	Ширший	Середній	(N)Наш продукт і продукт конкурента 2 націлений на менеджерів, тоді як продуктом конкурента 2 можуть користуватися будь-які офісні працівники
Надійність	Середня	Середня	Висока	(N)При реалізації продукту лише з використанням серверів розробника дані користувача будуть передаватись по мережі, тому існує ризик втрати персональних даних, проте при правильній розробці можна мінімізувати даний ризик

Таким чином наш продукт пропонує більш лояльний спосіб оплати для користувачів. Точність продукт очікується вища. Основною слабкою стороною продукту є більш складне встановлення системи.

4.2 Технологічний аудит ідеї проекту

Метою даного підпункту є провдення аудиту технології, тобто нам необхідно визначити чи є ідеї здійсненною з технологічної точки зору. Тому ми оберемо технологію, за допомогою якої буде вироблятись продукт, визначемо чи існують необхідні моделі чи їх потрібно розробити, а також визначемо їх доступність.

Результати даного аналізу наведемо у вигляді таблиці 4.3:

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
Розробка продукту для оцінки ставлення аудиторії до виступу за допомогою проведення емоційної аналітики	Модель розпізнавання емоцій та модель детекції обличчя	Такі технології існують, проте точність розпізнавання емоцій на даний момент не достатньо висока, тому необхідно вдосконалювати дану модель. Моделі детекції обличчя розроблені на достатньому рівні і не потребують доробки	В готовому варіанті автору доступна лише beta-версія розроблена Microsoft та кілька тестових моделей, представлених у відкритому доступі у Інтернеті
	Моделі прогнозування	Такі моделі існують вже доволі давно і відрізняються гарною точністю, тому тут вдосконалення не потрібне	Чимало моделей реалізовано у різних мовах програмування та відкриті для використання
	Моделі розпізнавання голосу	Альтернативна модель – це модель розпізнавання голосу, по якій можна також визначити настрої аудиторії, такі моделі також існують, проте більшість із них призначена для англійськомовних записів, тому це автоматично звужує аудиторію. Відповідно моделі вимагають значного вдосконалення	Автору також відомі лише beta-версії, які надаються великими компаніями
Обрана технологія реалізації ідеї проекту: модель розпізнавання емоцій та модель детекції обличчя, моделі прогнозування. Наш вибір обґрунтований необхідністю створення точного продукту.			

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Надзвичайно важливим є проведення аналізу ринку, тобто для нас важливо чи готовий ринок сприйняти наш продукт, чи буде достатній попит на нього, тощо. Адже чимало компаній програли через те, що не до кінця розуміли потреби ринку.

Даний аналіз дасть нам можливість зрозуміти основні напрямки розвитку стартапу, маючи дані про ринок, основних конкурентів, можливі перешкоди при впровадженні продукту, тощо.

У наступній таблиці 4.4 наведемо аналіз попиту, його обсяг та заповненість ринку.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартапу

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	2 (Google і Microsoft, які працюють у напрямках штучного інтелекту)
2	Загальний обсяг продаж, грн/ум.од	1.4 млрд $
3	Динаміка ринку (якісна оцінка)	Зростає, адже зростає кількість технологій, а отже і зацікавленість клієнтів
4	Наявність обмежень для входу (вказати характер обмежень)	Якість та швидкість реалізації: виграє той продукт, який швидше реалізують та який буде давати найточніші результати
5	Специфічні вимоги до стандартизації та сертифікації	Таких вимог поки немає, адже галузь розвивається, а тому відповідно стандарту поки не існує
6	Середня норма рентабельності в галузі (або по ринку), %	180%, адже глобальні витрати оцінюються у 500 млн $

Основні дані про ринок було отримано на основі наступних джерел [66, 67].

Отже, оскільки рентабельність продукції доволі висока, то є сенс розпочинати розробку продукту

Наступним кроком остаточно визнаємо групи потенційних користувачів та функціонал, який буде їх цікавити (див. табл. 4.5):

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Збільшити якість зустрічей та зменшити їх кількість	Ділові люди, викладачі, політики	Ділові люди найкращий сегмент, адже для них є критичним ефективне використання часу, адже сааме він визначає їх прибуток. Для викладачів система носитиме скоріше допоміжний характер, тому використання для них повинно реалізовуватись по знижені ціні, або у тестовому режимі.	- точність системи - швидкість роботи - простота підтримки

Тепер визначимо основні значення показників ринкового середовища, а саме таких факторів як конкуренція, збут, попит, тощо.

Результати проведеного аналізу преставлено у таблиці 4.6, у якій показники подаються у порядку спадання їх важливості.

Таблиця 4.6 – Визначення ринкових можливостей і загроз

Параметри оцінки	Можливості	Загрози
Конкуренція	Система не є поширеною, тому існує можливість виграти за рахунок унікальності	Великі компанії розробляють власні сервіси штучного інтелекту, тому можливе створення схожих систем із готових моделей
Збут	Оскільки конкуренція невисока, то відповідно простіше буде утримати клієнта	Необхідно переконати користувача у корисності системи

Продовження таблиці 4.6

Попит	Ринок зростає в силу популярності IT-спеціальностей та способів ведення проектів	
Економічні	Більша частина українського IT-ринку співпрацює з іноземними компаніями, що робить її більш стабільним у порівнянні з економічною ситуацією в цілому	
Науково-технічні	Доступність Інтернету мінімізує ризик неможливості залишатись неконкурнетноспроможним через точність моделі	Система базується на моделях оцінки емоцій, тому існує можливість розробки точніших та швидших моделей за кордоном, що створить конкурента на ринку

Таким чином автор вбачає, що ринок готовий сприймати даний продукт.

У наступній таблиці 4.7 ми визначемо основні риси конкуренції на ринку:

Таблиця 4.7 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції - монополія/олігополія/ монополістична/чиста	Існують моделі оцінки емоцій, які пропонуються великими компаніями, тому існує можливість розробки схожих продуктів	Розробка унікального продукту з додатковими можливостями та аналітикаим

Продовження таблиці 4.7

2. За рівнем конкурентної боротьби - локальний/національний/міжнародний	Товар орієнтований на ІТ-бізнес	Компанії потрібно старатись захоплювати клієнтів спершу на національному, а пізніше і міжнародному рівні Необхідно розробити можливість самостійного встановлення продукту користувачем, задля розширення бази клієнтів
3. За галузевою ознакою - міжгалузева/ внутрішньогалузева	Товар достатньо актуальний для ділових осіб і є допоміжним засобом у їх роботі	Необхідно зробити продукт якомого зручнішим у використанні, таким , щоб він вимагав мінімального часу при підтримці
4. Конкуренція за видами товарів: - товарно-родова - товарно-видова - міжбажаннями	Можлива розробка схожих продуктів	Необхідно збільшувати точність та якість роботи продукту
5. За характером конкурентних переваг - цінова / нецінова	На ринку присутні обидва види конкурнеції, адже користувач хоче менше платити за продукт, але при цьому отримувати нові можливості	Необхідно покращувати продукт
6. За інтенсивністю - марочна/не марочна	Поняття марка на цьому ринку поки не існує	Немає особливих потреб витрачатись на розкрутку бренду

Тепер проведемо аналіз конкуренції згідно з моделлю сил М. Портера, результати занесемо до таблиці 4.8:

Таблиця 4.8 – Аналіз конкуренції в галузі за М. Портером

Скла-довіаналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	MeetingPulse Voicera+Wrappup	Бар'єри входження, готовність аудиторії, швидкість реалізації	Визначити фактори сили постачальників Висока	Визначити фактори сили споживачів Висока	Фактори загроз з боку замінників

Основними висновками даного аналізу є наступні твердження:

На даний момент прямі конкуренти існують, проте вони також розвиваються, адже технології все ще розвиваються.

Існує ризик того, як користувач сприйме товар, адже цей тип товарів все ще розвивається. Тому можливо варто скористатись бета-версією, яка б спершу була безкоштовною (і існувала б за рахунок реклами). Таким чином можна зацікавити ринок і виявити основні недоліки на ранніх стадіях. Також складність полягає в тому, що більший шанс на успіх буде мати та розробка, яка вийде на ринок першою та буде достатньо точною.

Постачальними є лише розробники камер, яких зараз існує достатньо багато, щоб нівелювати даний фактор.

Стан клієнтів визначає чи стануть вони купувати продукт, адже система є допоміжною, а тому користувач стане використовувати систему лише у тому випадку, коли основна діяльність компанія є прибутковою. Але оскільки сектор ІТ-технологій та бізнес розвиваються, то відповідно сила споживачів буде лише рости.

На даний момент готових прямих товарів-замінників не існує. Існує ризик створення схожих продуктів.

Таким чином у данного продукту є можливість працювати на такому ринку, адже ринок ще розвивається, а тому конкуренти хоча і існують, але їх розробки ще не

повністю готові. Товар повинен бути достатньоо точним та вийти на ринок якомога раніше для того, щоб зайняти конкурентну позицію.

На основі вище проведеного аналізу сформулюємо основні фактори маркетингового середовища та обґрунтуємо даний вибір у таблиці 4.9.

Таблиця 4.9 – Обґрунтування факторів конкурентоспроможності

Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
Унікальність	На даний момент існує мало продуктів аналогів, а тому існує можливість захопити ринок, базуючись на унікальності продукту
Простота використання	Після втановлення продукту ним буде просто користуватись, а тому він буде більш привабливим для користувача
Точність	Система повинна бути достатньо точною
Швидкість реалізації	Чим швидше вийде продукт, тим більша ймовірність захопити ринок

Отже, наш продукт обіцяє мати необхідні характеристики, які зроблять його конкурентноспроможним, тому існує можливість успішного виходу на ринок.

Наступним кроком за допомогою визначених факторів проведемо порівняльний аналіз сильних та слабких сторін стартапу у таблиці 4.11. Для зручності будемо використовувати наступні позначення: Конкуренти: MeetingPulse (MP), Voicera+Wrappup(VW), наш продукт “EmoAudience”(EA).

Таблиця 4.10 – Порівняльний аналіз сильних та слабких сторін «EmoAudience»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з “EmoAudience”						
			–3	–2	–1	0	+1	+2	+3
1	Унікальність	2		MP				EA/VW	
2	Простота використання	1					EA	VW	MP
3	Точність	2			MP			EA/VW	
4	Швидкість реалізації	-1			EA/VW				MP

На основі проведеного аналізу складемо SWOT-матрицю, тобто остаточно сформулюємо сильні та слабкі сторони продукту, його можливості та загрози. Результати представимо ц вигляді таблиці 4.11:

Таблиця 4.11 – SWOT- аналіз стартап-проекту

Сильні сторони: Унікальність Простота використання Точність	Слабкі сторони: Швидкість реалізації
Можливості: Задоволення потреб шляхом зменшення витрати на зустрічі	Загрози: Можливість несприйняття ринком даного виді товарів

Для мінімізації ризику виникає необхідність розробки кількох стратегій поведінки на ринку. Остаточний вибір відбудеться після отримання перших результатів відклику ринку. Таким чином у таблиці 4.12 визначемо основні альтернативи, оцінимо строки реалізації того чи іншого рішення, а також оцінимо ймовірність отримання ресурсів для обраного напрямку.

Таблиця 4.12 – Альтернативи ринкового впровадження стартап-проекту

Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Швидка розробка продукту, розробка продукту по моделі Code&Fix	Низька, адже з одного боку ми вийдемо одні з перших на ринок, проте моделі навряд чи будуть точними	6 міс
Повільна розробка продукту, розробка проекту по методології WaterFall	Середня, адже продукт вийде достатньо якісним, проте існує ймовірність програти конкурентам	3р
Середня швидкість розробки, введення beta-версій, проведення проекту згідно методології Kanban або Prototyping	Висока, адже і продукт буде достатньо якісним та користувачі побачать його достатньо швидка	1р

Таким чином була обрана третя альтернатива, адже вона дає найбільшу ймовірність захоплення ринку та прийнятні строки реалізації. При виборі стратегії потрібно розуміти, що ринок не може чекати занадто довго, адже у такому разі можуть з'явитись аналогічні продукти. І користувачі не готові платити за недороблений продукт. Тому випуск слабо працюючого продукту може відлякати потенційних користувачів у майбутньому.

4.4 Розроблення ринкової стратегії проекту

На основі проведеного аналізу здійснимо розробку ринкової стратегії. Для цього остатчно опишемо цільовий сегмент користувачів згідно з таблицею 4.13:

Таблиця 4.13 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
	Менеджери та офісні працівники	Готові сприйняти продукт, якщо зрозуміють його цінність	Середній	Середня	Висока
	Викладачі	Не готові, готовими можуть бути лише викладачі приватних шкіл	Низький	Низька	Середня
	Політики	Середня	Середній	Середній	Середня
Які цільові групи обрано: Менеджери та офісні працівники					

Наступним кроком визначимо основну стратегію розвитку у таблиці 4.14:

Таблиця 4.14 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Розробка системи з використанням локальних і віддалених серверів	Збільшуватимемо швидкодію системи та знижуватимемо ціну абонентської плати шляхом введення можливості користувачем зберігати дані у себе на серверах. Клієнти як правило будуть мати власні сервери, тому використання власних ресурсів зниже для них ціну	Система буде вимагати меншої підтримки з боку розробника та збільшуватиме свободу та приватність користувача. Також зменшиться ціна на продукт за рахунок зменшення витрати розробника і збільшиться швидкодія системи, адже не потрібно пересилати великі об’єми даних по мережі	Статегія спеціалізації: Обрано розробляти продукт для вибраного сегменту користувачів, яка б допомагала інтерактивно оцінювати настрої аудиторії при доповіді
2	Розробка системи з можливістю донавчання	Користувач буде мати можливість покращувати модель оцінки емоцій до навчаючи її на учасниках, які будуть приймати участь у зустрічах(наприклад маючи вибірку з працівників компанії можна покращити якість моделі)	Збільшення точності моделі збільшить її якість, а відповідно і задоволення користувачів продуктом	

Продовження таблиці 4.14

3	Розробка додаткової аналітики	Збільшення спектру можливостей, додаючи нові моделі для аналізу даних, наприклад аналіз зацікавленості аудиторії у залежності від тематики	Більша кількість можливостей завжди приваблює користувачів

Оберемо стратегії конкурентної поведінки (див. табл. 4.15):

Таблиця 4.15 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Проект не є «першопроходцем» на ринку	Компанія буде шукати нових споживачів, адже ринок все ще розвивається	Так буде, хоча товар є доволі унікальним, проте корисні властивості конкурентів теж можна по можливості включати у продукт	Стратегія наслідування лідера Оскільки продукт тільки розробляється очікувати лідерства у сфері не варто, адже існують великі корпорації, які хоч і не займаються таким типом продукції безпосередньо, проте вони мають змогу це зробити дуже швидко

Таким чином розробимо стратегію позицінування,тобто комплекс асоціацій, який повинен виникати у користувача по відношенню до нашого продукту. Отримані результати, що представлені у таблиці 4.16, грунтується на результатах попередніх аналізів, а саме виборі сегментів користувачів (див. табл. 4.5), стратегії розвитку (див. табл. 4.14) та стратегії конкурентної поведінки (див. табл. 4.15).

Таблиця 4.16 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій
1	Точність	Виготовлення системи інтерактивної оцінки настроїв аудиторії при доповіді	За рахунок гарно підготовленої моделі система буде здатна коректно розпізнати емоцію і робити відповідний аналіз	Доступність, ефективність зустрічей, точність моделей
2	Доступна ціна		Через простоту підтримки продукт матиме порівняно низьку ціну у вигляді абонентської плати	
3	Ефективне використання		Користувач зможе платити за кількість запитів до системи або за об’єм даних для аналізу	
4	Покращення якості зустрічей		Система покликана покращити якість зустрічей ділових осіб або зменшити їх кількість, що зменшить витрати компаній	

Таким чином ми сформули основні елементи ринкової поведінки. Приблизний план такої поведінки буде наступним:

- розробка продукт із середньою швидкістю та вихід на ринок із безкоштовною beta-версією продукту, можливо з рекламою;
- збільшення спектру користувачів та доробка продукту, збільшення його точності;
- введення абонентської плати для скороченої та розширеної версії продукту;
- постійна доробка продукту.

4.5 Розроблення маркетингової програми стартап-проекту

Наступним важливим кроком є розробка маркетингової програти стартап-проекту. Для цього у таблиці 4.17 підсумуємо результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.17 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Зменшення витрати на зустрічі	Система підказує чи є(була) зустріч ефективною, а тому дає можливість відміняти непотрібні зустрічі	Простота підтримки, зменшення витрати на зустрічі та підвищення їх ефективності
2	Розуміння запитів аудиторії	Система пропонує поради, які покращать розуміння того, які теми цікавлять аудиторію	

Надалі розробимо трирівневу маркетингова модель товару, а саме визначимо ідею продукту, його фізичні складові, особливості процесу його надання (див. табл. 4.18).

Важливим питання при розробці продутку є вибір цінових меж. Потрібно розуміти, що користувачі не стануть платити занадто високу ціну за продукт, проте і занижувати ціну теж не варто, адже по-перше, нам потрібен капітал для розвитку, по-друге, якщо ми плануємо підняти ціну у майбутньому це обов'язково призведе до втрат користувачів і вони можуть виявитись вищим, ніж очікується саме через те, що попередня ціна була значно нижчою. Таким чином на основі екпертного методу було визначено рівень доходів потенційних користувачів і на основі цього визначено ціну на продукт (див. табл. 4.19).

Таблиця 4.18 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="2">Сутність та складові</td></tr>
<tr><td>І. Товар за задумом</td><td colspan="2">Система зменшує витрати на малоефективні зустрічі шляхом підвищення розуміння потреб аудиторії та їх ставлення до виступів доповідачів</td></tr>
<tr><td rowspan="7">ІІ. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>Значення</td></tr>
<tr><td>Камери на стороні користувача</td><td>Користувач здатен самостійно обирати камер, але для коректної роботи вони повинні бути не менше 2mpx</td></tr>
<tr><td>Сервери з моделями та даними</td><td>Комп’ютери, на яких буде опубліковано сервіс повинні мати принаймні 16Gbоперативної пам’яті та 800Gbпостійної пам’яті для збереження даних</td></tr>
<tr><td>Сервери для до навчання(можуть бути суміщені з попередніми серверами)</td><td>Комп’ютери, які мають донавчати систему по запиту користувача. Для цього вони повинні мати принаймні 64Gbоперативної пам’яті та 800Gbпостійної. Такі сервери можна оредувати на деякий час у великих компаній</td></tr>
<tr><td colspan="2">Якість: точність моделей більше 90%</td></tr>
<tr><td colspan="2">Пакування: відсутнє, адже основа – це програмний продукт</td></tr>
<tr><td colspan="2">Марка: система “EmoAudience”</td></tr>
<tr><td rowspan="2">ІІІ. Товар із підкріпленням</td><td colspan="2">До встановлення консультація, демонстрація ефективності системи</td></tr>
<tr><td colspan="2">До продажу: можливість скористаться пробною версією</td></tr>
<tr><td colspan="3">За рахунок чого потенційний товар буде захищено від копіювання: моделі оцінки емоцій осіб та моделі прогнозування та їх параметри не будуть поширюватися клієнту та залишатимуться у розробника</td></tr>
</table>

Таблиця 4.19 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	1-20$ на місяць	1-20$ на місяць	25000 - 50000 грн в місяць	75 – 250 грн на місяць

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (див. табл. 4.20):

Таблиця 4.20 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Основними групами є ділові особи, тому для того, щоб такою продукцією зацікавились необхідно поширювати її по Інтернету та зробити максимально простою у встановленні	Рекомендація пристрою	Для сайту нашої компанії – в залежності від попиту	Продаж через сайт компанії, адже саме там можна знайти потенційних клієнтів
2		Демонстрація ефективності системи		

На основі обраної стратегії позиціонування, розробимо концепцію маркетингових комунікацій у таблиці 4.21:

Таблиця 4.21 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Ділові люди стануть купувати продукт, якщо будуть впевнені в його корисності та при простому встановленні системи	Інші продукти	Простота використання, ефективність проведення зустрічей, зменшення безкорисних зустрічей	Допомогти користувачу зрозуміти ефективність такої покупки	Ніколи не було так просто і ефективно проводити зустрічі
2		Інтернет			
3		Телебачення			

Висновки до розділу 4

Отже, у даному розділі було проведено розробку стартап-проекту, а саме проведено аналіз ринку, виділено сегмент користувачів, проведено аналіз конкурентів, визначено ринкову стратегію, тощо. Таким чином ми зрозуміли наступне:

- обраний ринок розвивається, попит буде лише зростати, його рентабельність висока;
- існують перспективи впровадження, адже ринок зацікавлений в потребах, які буде реалізовувати продукт, конкуренція на цьому ринку існуватиме, проте обрана стратегія дасть можливість бути конкурнетноспроможним;
- для розробки продукту доцільно обрати стратегію описану у висновках до підпункту 4, яка полягає у виході на ринок з мінімально-доцільним продуктом, пропонуючи користувач використовувати beta-версію продукт, по мірі покращення продукту необхідно ввести абонентську плату;
- подальша імплементація доцільна та сприятлива.

На основі проведеного аналізу ми отримали бізнес-модель даного продукту, з якою можна ознайомитись у додатку В.

ВИСНОВКИ

В даній роботі здійснено глибокий аналіз проблеми розпізнавання емоцій. Визначено актуальність даної роботи, адже її результати можна використати у маркетингових та соціальних сферах.

На основі теоретичного матеріалу було здійснено побудову нейронних мереж для розпізнавання емоцій, а саме використовувались 2 різних підходи: на основі ключових точок та згорткові нейронні мережі з різними архітектурами. Отримані моделі мають високу точність, яка коливається від 0.7 до 0.95.

В середньому було отримано, що згорткові нейронні мережі є кращими з точки зору точності, повноти та швидкості обробки відео. Час витрачений на їх навчання також вищий через більш складну архітектуру.

Дана поведінка пов’язана з тим, що згорткові нейронні мережі на відміну від нейронних мереж, побудованих на основі ключових точок, самі визначають більш важливі ознаки, які й використовуються для навчання.

Деякі моделі зазнавали перенавчання, тому використовувався метод ранньої зупинки для уникнення цього.

Серед побудованих моделей, найкращою виявилась Модель 3 – згорткова нейронна мережа, що містить 6 згорткових, 6 агрегуючих та нормалізуючих шарів та 2 повнозв’язних шарів. Точність даної моделі становить 0.956 при навчанні та 0.872 при тестуванні, повнота – 0.953 при навчанні та 0.891 при тестуванні, дана модель витрачає 1.418 с на обробку 1 відеофайлу та 4.803 с на завантаження. При цьому отримана модель має 9896 параметрів та займає 0.204 Mb.

Проте відкритим залишається питання оптимальної архітектури нейронних мереж, адже, незважаючи на високі результати, отриманих моделей стає зрозумілим, що можлива побудова інших моделей також.

Також в межах даної роботи було розроблено продукт із користувацьким інтерфейсом для пакетного тестування побудованих моделей та детальному аналізу емоцій на вибраних відео.

ПЕРЕЛІК ПОСИЛАНЬ

1. Ekman P. Facial action coding system: Investigator’s guide / P. Ekman, W.V. Friesen, J.C. Hager. – Palo Alto: Consulting Psychologists Press, 1978. – 42 p.
2. Littlewort G. Fully automatic codeing of basic expressions from video [Electronic resource] / G. Littlewort, I. Fasel, M. Stewart Bartlett. – San Diego, 2002. – [Cited: 2018,25 Oct.]. – Available from: ftp://shelton.snl.salk.edu/pub/marni/Littlewort_JSNC02.pdf.
3. Pantic M. Automatic Analysis of Facial Expressions / M. Pantic, M. Rothkrantz // IEEE Trans. Pattern Recognit. Mach. Intell. – 2000. – P. 1424 – 1444.
4. LeCun Y. Backpropagation applied to handwritten zip code recognition / Y. LeCun, B. Boser, J.S. Denker // Neural Comput. – 1989. – vol. 1, no. 4. – P. 541–551.
5. Day M. Exploiting Facial Landmarks for Emotion Recognition in the Wild [Electronic resource]. – 2014. – P. 390. – Available from: https://arxiv.org/ftp/arxiv/papers/1603/1603.09129.pdf.
6. Ko B.C. A Brief Review of Facial Emotion Recognition Based on Visual Information [Electronic resource]. – Daegu, 2018. – Available from: www.mdpi.com/1424-8220/18/2/401/pdf.
7. Sebe N. Emotion Recognition Using a Cauchy Naive Bayes Classifier / N. Sebe, S. Michael, I. Cohen // Proceedings of the International Conference on Pattern Recognition. – 2002. – P. 17–20.
8. Yuille A. Extraction from faces using deformable templates / A. Yuille, D. Cohen, P. Halliman // Int. J. Comput. Vis. – 1992. – vol. 8, no. 4. – P. 104–109.
9. Kass M. Snakes: active contour models / M. Kass, A. Witkin, D. Terzopoulus // Int. J. Comput. Vis. – 1988. – vol. 1, no. 4. – P. 321–331.
10. Lucey P. Automatically detecting pain in video through facial action units / P. Lucey, J.F. Cohn, I. Matthews // IEEE Trans. Syst. Man Cybern. Part B. – 2011. – vol. 41, no. 3. – P. 664–674.
11. Chen S.H. Speaker verification using MFCC and support vector machine / S.H. Chen,

Y.R. Luo // IMECS. – 2009. – vol. 1, no. 2. – P. 234–241.

12. Aida-Zade K.R. InvestigationofcombineduseofMFCC andLPCfeaturesinspeechrecognitionsystems / K.R. Aida-Zade, C. Ardil, S.S. Rustamov // Int.J.Comput.Inf.Syst.ControlEng. – 2008. – vol. 2, no. 7. – P. 6–12.
13. Busso C. Analysis of emotionally salient aspects of fundamental frequency for emotion detection / C. Busso, S. Lee, S. Narayanan // IEEE Trans. Speech Audio Process. – 2009. – vol. 17, no. 4. – P. 582–596.
14. Glowinski D. Toward a minimal representation of affective gestures / D. Glowinski et al. // IEEE Trans. Affect. Comput. – 2011. – vol. 2, no. 2. – P. 106–118.
15. Castellano G. Recognising human emotions from body movement and gesture dynamics / G. Castellano, S.D. Villalba, A. Camurri // Affect. Comput. Intell. Interact. – 2007. – vol. 4738, no. 1. – P. 71–82.
16. Camurri A. Recognizing emotion from dance movement: comparison of spectator recognition and automated techniques / A. Camurri, I. Lagerlog, G. Volpe // Int. J. Hum. Comput. Stud. – 2003. – vol. 59, no. 1. – P. 213–225.
17. Singh G. Negative emotion recognition from stimulated EEG signals / G. Singh, A. Jati, A. Khasnobish // Third International Conference on Computing Communication & Networking Technologies. – 2012.
18. Petrantonakis P.C. Emotion recognition from brain signals using hybrid adaptive filtering and higher order crossings analysis / P.C. Petrantonakis, L.J. Hadjileontiadis // IEEE Trans. Affect. Comput. – 2010. – vol. 1, no. 2. – P. 81–97.
19. Lin Y.P. EEG-based emotion recognition in music listening / Y.P. Lin, C.H. Wang, T.P. Jung // IEEE Trans. Biomed. Eng. – 2010. – vol. 57, no. 7. – P. 1798–1806.
20. Mehrabian A. Communication without words / A. Mehrabian // Psychol. Today. – 1968. – vol. 2, no. 4. – P. 53–56.
21. Ko B.C. A Brief Review of Facial Emotion Recognition Based on Visual Information / B.C. Ko // Sensors. – 2017. – vol. 2, no. 3. – P. 1–20.
22. Sirovich L. Low-dimensional Procedure for the Characterization of Human Faces / L. Sirovich, M. Kirby // J. Opt. Soc. Am. – 1987. – vol. 4, no. 4. – P. 519--524.
23. Turk M. Eigenfaces for Recognition / M. Turk, A. Pentland // J. Cogn. Neurosci. –

1991. – vol. 3, no. 1. – P. 195–200.

24. Belhumeur P. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection / P. Belhumeur, J. Hespanha, D. Kriegman // IEEE Trans. Pattern Anal. Mach. Intell. – 1997. – vol. 19, no. 7. – P. 711--720.
25. Yang M.H. Face Recognition Using Kernel Methods / M.H. Yang // Adv. Neural Inf. Process. Syst. – 2002. – vol. 3, no. 4. – P. 215–220.
26. Simonyan K. Fisher Vector Faces in the Wild / K. Simonyan, O.M. Parkhi, A. Vedaldi // British Machine Vision Conference. – 2013. – P. 8.1-8.11.
27. Martinez A.M. PCA versus LDA / A.M. Martinez, A.C. Kak // IEEE Trans. Pattern Anal. Mach. Intell. – 2001. – vol. 23, no. 2. – P. 228–233.
28. Swets D.L. Using Discriminant Eigenfeatures for Image Retrieval / D.L. Swets, J.J. Weng // IEEE Trans. Pattern Anal. Mach. Intell. – 1996. – vol. 18, no. 8. – P. 831–836.
29. Etemad K. Discriminant Analysis for Recognition of Human Face Images / K. Etemad, R. Chellapa // J. Opt. Am. A. – 1997. – vol. 14, no. 8. – P. 1724–1733.
30. Dumas M. Emotional Expression Recognition using Support Vector Machines [Electronic resource] / M. Dumas. – 2001. – P. 1–7. – Available from: http://cseweb.ucsd.edu/~elkan/254spring01/mdumasrep.pdf.
31. Shbib R. Facial Expression Analysis using Active Shape Model / R. Shbib, S. Zhou // Int. J. Signal Process. Image Process. Pattern Recognit. – 2015. – vol. 8, no. 1. – P. 9–22.
32. Jang G.-J. Facial Emotion Recognition Using Active Shape Models and Statistical Pattern Recognizers / G.-J. Jang, J.-S. Park, A. Jo // Ninth International Conference on Broadband and Wireless Computing, Communication and Applications. – 2014. – P. 1–10.
33. Liliana D.Y. Human emotion recognition based on active appearance model and semi-supervised fuzzy C-means / D.Y. Liliana, M.R. Widyanto, T. Basaruddin // International Conference on Advanced Computer Science and Information Systems (ICACSIS). – 2016. – P. 439–445.
34. Ratliff S. Emotion recognition using facial expressions with active appearance models

/ S. Ratliff, M. Patterson // Proc. ASTED International Conference on Human Computer Interaction. – 2008. – P. 138–143.

35. Livingstone S. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English [Electronic resource] / S. Livingstone, F. Russo. – 2018. – Available from: https://doi.org/10.1371/journal.pone.0196391.
36. MMI Facial Expression Database [Electronic resource]. – 2010. – Available from: https://mmifacedb.eu/.
37. University P.D. in K. The Japanese Female Facial Expression (JAFFE) Database [Electronic resource]. – Available from: http://www.kasrl.org/jaffe.html.
38. Kanade T. Comprehensive database for facial expression analysis [Electronic resource] / T. Kanade, J.F. Cohn, Y. Tian // Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition. – 2000. – Available from: http://www.consortium.ri.cmu.edu/ckagree/.
39. Mavadati S.M. DISFA: A spontaneous facial action intensity database [Electronic resource] / S.M. Mavadati, M.H. Mahoor, K. Bartlett // IEEE Trans. Affect. Comput. – 2013. – Available from: http://www.engr.du.edu/mmahoor/DISFA.htm.
40. The Yale Face Database B [Electronic resource]. – 2017. – Available from: http://vision.ucsd.edu/~leekc/ExtYaleDatabase/Yale Face Database.htm.
41. Sneddon I. The Belfast induced natural emotion database / I. Sneddon et al. // IEEE Trans. Affect. Comput. – 2012. – vol. 3, no. 1. – P. 32–41.
42. Lundqvist D. The Karolinska Directed Emotional Faces – KDEF [Electronic resource] / A. Flykt, A. & Öhman // Department of Clinical Neuroscience, Psychology section, Karolinska Institutet. – 1998. – Available from: http://www.emotionlab.se/kdef/.
43. Powers D. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation / D. Powers // J. Mach. Learn. Technol. – 2011. – vol. 2, no. 1. – P. 37–63.
44. Sasaki Y. The truth of the F-measure [Electronic resource] / Y. Sasaki // School of Computer Science, University of Manchester. – 2007. – P. 1–5. – Available from: http://www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-

26Oct07.pdf.

45. Viola P. Rapid Object Detection using a Boosted Cascade of Simple Features / P. Viola, M.J. Jones // Comput. Vis. Pattern Recognit. – 2001. – vol. 2, no. 3. – P. 234–237.
46. Sander S. Object detection using Haar-cascade Classifier / S. Sander // Mach. Learn. Investig. – 2014. – vol. 2, no. 3. – P. 245–247.
47. Stutz D. Understanding Convolutional Neural Networks [Electronic resource]. – 2014. – P. 23. – Available from: https://davidstutz.de/wordpress/wp-content/uploads/2014/07/seminar.pdf.
48. Agostinelli F. Learning Activation Functions to Improve Deep Neural Networks / F. Agostinelli, M. Hoffman, P. Sadowski // Neural Networks: Tricks of the Trade. – 2014. – vol. 1524, no. 2. – P. 124–127.
49. Huaiqin W. Global stability analysis of a general class of discontinuous neural networks with linear growth activation function / W. Huaiqin // Inf. Sci. (Ny). – 2009. – vol. 179, no. 19. – P. 3432–3441.
50. Duda R. Pattern Classification / R. Duda, P. Hart, D. Stork. – New York: 2nd Edition Wiley-Interscience, 2001. – 656 p.
51. Snyman J. Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms. – Springer Science & Business Media, 2005. – 257 p.
52. Goodfellow I. Deep Learning / I. Goodfellow, Y. Bengio, A. Courville. – MIT Press, 2016. – 781 p.
53. Rumelhart D. Learning representations by back-propagating errors / D. Rumelhart, G. Hinton, R. Williams // Nature. – 1986. – vol. 323, no. 6088. – P. 533–536.
54. Tetko I.V. Neural network studies. 1. Comparison of overfitting and overtraining / I.V. Tetko, D.J. Livingstone, A.I. Luik // J. Chem. Inf. Comput. Sci. – 1995. – vol. 35, no. 5. – P. 826–833.
55. Srivastava N. Dropout: A Simple Way to Prevent Neural Networks from Overfitting / N. Srivastava, G. Hinton, A. Krizhevsky // J. Mach. Learn. Res. – Toronto, 2014. – vol. 15, no. 5. – P. 1929–1958.

56. LeCun Y. Gradient-based learning applied to document recognition / Y. LeCun, L. Buttou, Y. Bengi. – Proceedings of the IEEE, 1998. – vol. 2, no. 3. – P. 125–128.
57. Erhan D. The difficulty of training deep architectures and the effect of unsupervised pre-training / D. Erhan, P.-A. Manzagol, Y. Bengio // In Artificial Intelligence and Statistics, International Conference. – 2009. – P. 153–160.
58. Erhan D. Why does unsupervised pre-training help deep learning? / D. Erhan, Y. Bengio, A. Courville // J. Mach. Learn. Res. – 2010. – vol. 11, no. 3. – P. 625– 660.
59. Larochelle H. Exploring strategies for training deep neural networks / H. Larochelle, Y. Bengio, J. Louradour // J. Mach. Learn. Res. – 2009. – vol. 1, no. 140. – P. 120–124.
60. Ciresan D.C. Multi-column deep neural networks for image classification [Electronic resource] / D.C. Ciresan, U. Meier, J. Schmidhuber // IEEE Computer Society Conference on Computer Vision and Pattern Recognition. – 2018. – Available from: https://arxiv.org/abs/1202.2745.
61. LeCun Y. Convolutional networks and applications in vision / Y. LeCun, K. Kavukvuoglu, C. Farabet // Int. Symp. – 2010. – vol. 2, no. 4. – P. 253–256.
62. Zeiler M.D. Visualizing and understanding convolutional networks / M.D. Zeiler, R. Fergus // Comput. Res. Repos. – 2013. – P. 818–833.
63. Zarei A. The Effect of Applying Gaussian Blur Filter on Captcha’s Security / A. Zarei // First International Conference on Computer Science & Information Technology. – Computer Science & Information Technology, 2017. – P. 161–165.
64. Krizhevsky A. ImageNet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G. Hinton // Adv. Neural Inf. Process. Syst. – 2012. – vol. 4, no. 3. – P. 1097–1105.
65. Google Keras: The Python Deep Learning library [Electronic resource]. – 2018. – Available from: https://keras.io/.
66. Business Ukraine IT industry: Ukrainian IT exports increase by 20% in 2017 to reach USD 3.6 billion [Electronic resource]. – 2018. – Available from: http://bunews.com.ua/economy/item/it-industry-ukrainian-it-exports-increase-by-20-in-2017-to-reach-usd-36-billion.

67. Henrik Pedersen Ukranian IT industry overview: The result of 2017 & trends for 2018 [Electronic resource]. – 2017. – Available from: https://outsourcingreview.org/ukrainian-it-industry-overview-the-results-of-2017-trends-for-2018/.

ДОДАТОК А СКРИПТИ ДЛЯ МОДЕЛЮВАННЯ

Скрипт для створення файлу з метаданими:

```
# This code get videos from the folder defined in the __init__ method of Processing class
# Ut creates a bin\\metadata.csv file where we save metadata including:
#file path
#emotion type
#actor number

import cv2
import os
import shutil
import pandas as pd

cnt = 0
def parse_file_name(file_name):
  global cnt
  if cnt%20 == 0:
      print(cnt)
      print(file_name)
  cnt+=1
  row = {}

  meta, extension = file_name[:-4], file_name[-4:]
  meta = meta.split('-')
  #print(meta)
  _, vocal_channel, emotion, emotional_intensity, _, _, actor = meta

  row['file_name'] = file_name

  if extension == '.mp4':
    row['modality'] = 'video'
  if extension == '.wav':
    row['modality'] = 'audio'

  if vocal_channel == '01':
    row['vocal_channel'] = 'speech'
  if vocal_channel == '02':
    row['vocal_channel'] = 'song'

  if emotion == '01':
    row['emotion'] = 'neutral'
  if emotion == '02':
    row['emotion'] = 'calm'
  if emotion == '03':
    row['emotion'] = 'happy'
  if emotion == '04':
    row['emotion'] = 'sad'
  if emotion == '05':
    row['emotion'] = 'angry'
  if emotion == '06':
    row['emotion'] = 'fearful'
  if emotion == '07':
    row['emotion'] = 'disgust'
  if emotion == '08':
    row['emotion'] = 'surprised'

  if emotional_intensity == '01':
    row['emotional_intensity'] = 'normal'
  if emotional_intensity == '02':
    row['emotional_intensity'] = 'strong'

  row['actor'] = int(actor)
  #print(row)
  return row

class Preprocessing:
  #Defines folder where we get data
  def __init__(self, folder='..\DataSet'):
    self.folder = folder

  def unpack_all(self):
    zips = [x for _, _, files in os.walk(os.getcwd()) for x in files if x.endswith('.zip')]
    os.mkdir('data')
    for archive in zips:
      os.system('unzip {x} -d data/'.format(x=archive))

  def parse_audio_video(self):
    generator = os.walk(self.folder)
    video = [x for _, _, files in os.walk(self.folder) for x in files if x.endswith('.mp4')]
    for record in video:
      os.system('ffmpeg -i ' + os.path.join(self.folder, record) + ' ' + os.path.join(self.folder, record[:-4] + '.wav'))

  def generate_metadata(self):
    data = [os.path.join(paths, x) for paths, dirs, files in os.walk(self.folder) for x in files]
    data = [parse_file_name(file_name) for file_name in data]
    data = pd.DataFrame(data)
    data = pd.concat([data, pd.get_dummies(data['emotion'])], axis=1)
    data.to_csv('bin\\metadata.csv', index=False)

obj = Preprocessing()
obj.generate_metadata()
print("Done")
```

Скрипт для формування вибірки для згорткових нейронних мереж

```
#This code using metadata.csv file finds each video, extracts faces and save data to zipped format

#PREREQUISITES
#Import packages using cmd:
#python -m pip install h5py
#Import cv2:
#python -m pip install opencv-python
#python -m pip install opencv-contrib-python
#Import dlib
#python -m pip install --user dlib
#python -m pip install dlib
import numpy as np
import pandas as pd
import h5py
import cv2
import dlib
import time
import matplotlib.pyplot as plt
import datetime

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
base_path = 'D:\\'
metadata = pd.read_csv(base_path + 'metadata.csv')
metadata = metadata.loc[metadata.modality == 'video']

predictor = dlib.shape_predictor(

'C:\\Users\\kam\\Documents\\DiplomaExperiments\\Project_Video\\models\\shape_predictor_68_face_landmarks.dat'
)

def get_video_tensor(file_name, rescale=0.5, face_size=(200, 200)):
  cap = cv2.VideoCapture(file_name)
  frames = []
  isTooBig = False
  while True:
    retval, image = cap.read()
    if not retval:
      break
    #image = cv2.resize(image, None, fx=rescale, fy=rescale)
    #Check if image has been uploaded
    #show_image(image)
    try:
      (x, y, w, h) = face_cascade.detectMultiScale(image)[0]
      image = image[y:y+h, x:x+w, :]
      image = cv2.resize(image, dsize=face_size)
      frames.append(image)
      #show_image(image)
    except IndexError:
      print('omg!')
      print(file_name)
      isTooBig = True
  cap.release()
  frames = np.array(frames)
  #if isTooBig == False:
  # cap.release()
  # frames = np.array(frames)
  #else:
  # frames = np.array([])
  return frames

def show_image(image):
  cv2.imshow('image',image)
  cv2.waitKey(0)
  cv2.destroyAllWindows()

def save_hdf5():
  hf_data = h5py.File(base_path + 'data.h5', 'w')
  hf_labels = h5py.File(base_path + 'labels.h5', 'w')
  cnt=0
  #print(metadata.iterrows())
  for index, row in metadata.iterrows():
    if cnt%20 == 0:
      print(cnt)
    if cnt%100 == 0:
      print(datetime.datetime.now())
    cnt+=1
    tensor = get_video_tensor(row.file_name)
    if len(tensor) != 0:
      labels = np.repeat(row[-8:].values.reshape(1, -1), tensor.shape[0], axis=0)
      labels=labels.astype(int)
      hf_data.create_dataset(row.file_name, data=tensor)
      hf_labels.create_dataset(row.file_name, data=labels)
    #if cnt == 10:
    # break

def generator(batch_size=32):
  hf_data = h5py.File(base_path + 'data.h5', 'r')
  hf_labels = h5py.File(base_path + 'labels.h5', 'r')
  while True:
    for index, row in metadata.iterrows():
      tensor = hf_data.get(row.file_name)
      labels = hf_labels.get(row.file_name)
      for i in range(tensor.shape[0]//batch_size):
        batch = tensor[i*batch_size:(i+1)*batch_size]
        batch_labels = labels[i*batch_size:(i+1)*batch_size]
        yield batch, batch_labels

def save_size():
  hf_data = h5py.File(base_path + 'data.h5', 'r')
  metadata['size'] = 0
  cnt=0
  for index, row in metadata.iterrows():
    if cnt%20 == 0:
      print(cnt)
    if cnt%100 == 0:
      print('save_size:')
      print(datetime.datetime.now())
    cnt+=1
    tensor = hf_data.get(row.file_name)
    if not tensor:
      break
    metadata.loc[metadata.index == index, 'size'] = tensor.shape[0]
```

```
  #if cnt == 10:
  #  break
metadata.to_csv(base_path + 'metadata.csv', index=False)
```

```
print(datetime.datetime.now())
save_hdf5()
save_size()
print("Done")
```

Скрипт для формування вибірки для нейронних мереж на основі ключових точок.

```
#This code using metadata.csv file finds each video, extracts faces and save data to zipped format

#PREREQUISITES
#Import packages using cmd:
#python -m pip install h5py
#Import cv2:
#python -m pip install opencv-python
#python -m pip install opencv-contrib-python
#Import dlib
#python -m pip install --user dlib
#python -m pip install dlib
import numpy as np
import pandas as pd
import h5py
import cv2
import dlib
import time
import matplotlib.pyplot as plt
import datetime
import dlib
import math

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
base_path = 'bin\\'
metadata = pd.read_csv(base_path + 'metadata.csv')
metadata = metadata.loc[metadata.modality == 'video']

predictor = dlib.shape_predictor(

'C:\\Users\\kam\\Documents\\DiplomaExperiments\\Project_Video\\models\\shape_predictor_68_face_landmarks.dat'
)
detector = dlib.get_frontal_face_detector()

def get_landmarks(image):
  d = dlib.rectangle(0,0,200,200)
  shape = predictor(image, d) #Draw Facial Landmarks with the predictor class
  xlist = []
  ylist = []
  for i in range(0,68): #Store X and Y coordinates in two lists
    xlist.append(float(shape.part(i).x))
    ylist.append(float(shape.part(i).y))

  landmarks_vectorised = []

  xmin = np.min(xlist)
  xmax = np.max(xlist)
  xnorm = [(x-xmin)/(xmax - xmin) for x in xlist]

  ymin = np.min(ylist)
  ymax = np.max(ylist)
  ynorm = [(y-ymin)/(ymax - ymin) for y in ylist]

  #plt.plot(xnorm, ynorm, 'o', color='black')
  #plt.show()

  for x1, y1 in zip(xnorm, ynorm):
    for x2, y2 in zip(xnorm, ynorm):
      dist = (y2-y1)*(y2-y1)+(x2-x1)*(x2-x1)
      landmarks_vectorised.append(dist)
  if len(landmarks_vectorised) == 0:
    print('stop here')
  #landmarks_vectorised = landmarks_vectorised.reshape(1, len(landmarks_vectorised))
  return landmarks_vectorised

def get_video_tensor(file_name, rescale=0.5, face_size=(200, 200)):
  cap = cv2.VideoCapture(file_name)
  frames = []
  isTooBig = False
  while True:
    retval, image = cap.read()
    if not retval:
      break
    #show_image(image)
    try:
      (x, y, w, h) = face_cascade.detectMultiScale(image)[0]
      image = image[y:y+h, x:x+w, :]
      image = cv2.resize(image, dsize=face_size)
      image_vectors = get_landmarks(image)
      #if image_vectors.shape[1] != 0:
      frames.append(image_vectors)
      #show_image(image)
    except IndexError:
      print('omg!')
      print(file_name)
      isTooBig = True
  cap.release()
  frames = np.array(frames)
  return frames

def show_image(image):
  cv2.imshow('image',image)
  cv2.waitKey(0)
  cv2.destroyAllWindows()

def save_hdf5():
  hf_data = h5py.File(base_path + 'data_dlib.h5', 'w')
  hf_labels = h5py.File(base_path + 'labels_dlib.h5', 'w')
  cnt=0
  #print(metadata.iterrows())
  for index, row in metadata.iterrows():
    if cnt%20 == 0:
      print(cnt)
```

```
    if cnt%100 == 0:
      print(datetime.datetime.now())
    cnt+=1
    tensor = get_video_tensor(row.file_name)
    if len(tensor) != 0:
      labels = np.repeat(row[-8:].values.reshape(1, -1),
tensor.shape[0], axis=0)
      labels=labels.astype(int)
      hf_data.create_dataset(row.file_name, data=tensor)
      hf_labels.create_dataset(row.file_name, data=labels)
    #if cnt == 10:
    #  break

def generator(batch_size=32):
  hf_data = h5py.File(base_path + 'data_dlib.h5', 'r')
  hf_labels = h5py.File(base_path + 'labels_dlib.h5', 'r')
  while True:
    for index, row in metadata.iterrows():
      tensor = hf_data.get(row.file_name)
      labels = hf_labels.get(row.file_name)
      for i in range(tensor.shape[0]//batch_size):
        batch = tensor[i*batch_size:(i+1)*batch_size]
        batch_labels = labels[i*batch_size:(i+1)*batch_size]
        yield batch, batch_labels

def save_size():
  hf_data = h5py.File(base_path + 'data_dlib.h5', 'r')
  metadata['size'] = 0
  cnt=0
  for index, row in metadata.iterrows():
    if cnt%20 == 0:
      print(cnt)
    if cnt%100 == 0:
      print('save_size:')
      print(datetime.datetime.now())
    cnt+=1
    tensor = hf_data.get(row.file_name)
    if not tensor:
      break
    metadata.loc[metadata.index == index, 'size'] =
tensor.shape[0]
    #if cnt == 10:
    #  break
  metadata.to_csv(base_path + 'metadata.csv', index=False)

print(datetime.datetime.now())


save_hdf5()
save_size()
print("Done")
```

Скрипт для навчання та оцінки якості згорткових нейронних мереж

```
# coding: utf-8# In[1]:
import tensorflow as tf
# In[2]:
from matplotlib.pyplot import imshow
import numpy as np
import pandas as pd
import h5py
import matplotlib.pyplot as plt
import cv2
import time
# In[3]:
#changing style of plotsfrom matplotlib import rcParams
rcParams['axes.facecolor'] = 'black'
rcParams['figure.facecolor'] = 'black'
rcParams['font.weight'] = 'bold'
rcParams['axes.titleweight'] = 'bold'
rcParams['xtick.color'] = '#64E0C0'
rcParams['ytick.color'] = '#64E0C0'
rcParams['grid.color'] = '#64E0C0'
# In[4]:
from keras.layers import Activation, Conv3D
from keras.layers import BatchNormalization
from keras.layers import Reshape, Dense
from keras.models import Model, load_model
from keras.layers import Input, add
from keras.layers import MaxPooling3D,
GlobalMaxPooling1D
from keras.regularizers import l2
from keras.callbacks import EarlyStopping,
ModelCheckpoint, TensorBoard
# In[5]:

def grayscale(tensor):
  """Convert RGB tensor (n_frames, size_x, size_y,
channel) to
  grayscale tensor (n_frames, size_x, size_y, 1)"""
  r = tensor[:, :, :, 0]
  g = tensor[:, :, :, 1]
  b = tensor[:, :, :, 2]
  gray = 0.2989 * r + 0.5870 * g + 0.1140 * b
  gray = gray.reshape((tensor.shape[0], tensor.shape[1],
tensor.shape[2], 1))
  return gray
def sliding_window(data, size, stepsize=1, axis=0,
copy=True):
  """Get slices of the video tensor"""
  if axis >= data.ndim:
    raise ValueError(
      "Axis value out of range"
    )
  if stepsize < 1:
    raise ValueError(
      "Stepsize may not be zero or negative"
    )
  if size > data.shape[axis]:
    raise ValueError(
      "Sliding window size may not exceed size of selected
axis"
    )
  shape = list(data.shape)
  shape[axis] = np.floor(data.shape[axis] / stepsize - size /
stepsize + 1).astype(int)
  shape.insert(1, size)
```

```
  strides = list(data.strides)
  strides[axis] *= stepsize
  strides.insert(1, data.strides[axis])
  strided = np.lib.stride_tricks.as_strided(
    data, shape=shape, strides=strides
  )  if copy:
    return strided.copy()
  else:
    return strided

def get_few_frames(tensor, center, n_points, step, eq_hist=True, blur=True):
  """From video tensor with shape (n_frames, ...) get n_points
  frames around center with some step"""
  indexes = center + np.arange(-n_points//2, n_points//2+1)*step
  indexes = indexes[:tensor.shape[0]]
  tensor = tensor[indexes]

  if eq_hist:
    shape_3d = (tensor.shape[1], tensor.shape[2], tensor.shape[3])
    for i in range(tensor.shape[0]):
      tensor[i] = cv2.equalizeHist(tensor[i].astype('uint8')).reshape(shape_3d)

  if blur:
    shape_3d = (tensor.shape[1], tensor.shape[2], tensor.shape[3])
    for i in range(tensor.shape[0]):
      tensor[i] = cv2.GaussianBlur(tensor[i].astype('uint8'), (5, 5), 3).reshape(shape_3d)
  return tensor
# In[6]:
base_path = 'D:\\'
# In[7]:
metadata = pd.read_csv(base_path + 'metadata.csv')
# In[8]:
len(metadata)
# In[9]:
metadata = metadata.loc[metadata.modality == 'video']
# In[10]:
emotions = {
  0: 'angry',
  1: 'calm',
  2: 'disgust',
  3: 'fearful',
  4: 'happy',
  5: 'neutral',
  6: 'sad',
  7: 'surprised'
}
# In[11]:
def generate_slices(n_video=10, metadata=metadata, size=50, stepsize=10):
  hf_data = h5py.File('data.h5', 'r')
  hf_labels = h5py.File('labels.h5', 'r')
  while True:
    n_video_groups = np.ceil(metadata.shape[0]/n_video).astype(int)
    for index, row in metadata.iterrows():
      tensor = grayscale(hf_data.get(row.file_name)).astype('uint8')
      labels = hf_labels.get(row.file_name)
      tensor = sliding_window(tensor, size, stepsize, copy=False)
      yield tensor, labels[0]
# In[12]:
def generator_few_frames(batch_size=16, metadata=metadata):
  """Generate uniformly few frames around center of the video (center included)"""
  hf_data = h5py.File(base_path + 'data.h5', 'r')
  hf_labels = h5py.File(base_path + 'labels.h5', 'r')
  while True:
    n_batches = np.ceil(metadata.shape[0]/batch_size).astype(int)
    for i in range(n_batches):
      file_names = metadata.file_name[i*batch_size:(i+1)*batch_size]
      batch, labels = [], []
      for name in file_names:
        tensor = hf_data.get(name)
        if not(tensor is None):
          tensor = grayscale(tensor)
          label = hf_labels.get(name)[0]
          tensor = get_few_frames(tensor, center=tensor.shape[0]//2, n_points=4, step=7)
          batch.append(tensor)
          labels.append(label)

      batch = np.array(batch).astype('uint8')
      labels = np.array(labels)
      yield batch, labels
# In[13]:
g = generator_few_frames()
# In[14]:
x, label = next(g)
# In[15]:
x.shape
# In[16]:
print(emotions[label[1].argmax()])
imshow(x[1][1].reshape(200, 200), cmap='gray')
# In[17]:
def model_first(input_shape=(5, 200, 200, 1), num_classes=8, l2_regularization=0.01):
  regularization = l2(l2_regularization)  # base
  img_input = Input(input_shape)
  x = Conv3D(4, (1, 2, 2),
       strides=(1, 1, 1),
       kernel_regularizer=regularization,
       use_bias=False,
       padding='same')(img_input)
  x = MaxPooling3D((2, 2, 2),
         strides=(1, 2, 2),
         padding='same')(x)
  x = BatchNormalization()(x)
  x = Activation('relu')(x)

  x = Conv3D(8, (1, 2, 2),
```

```
        strides=(1, 1, 1),
        kernel_regularizer=regularization,
        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # module 1

    residual = Conv3D(8,
            (1, 2, 2),
            strides=(1, 2, 2),
            padding='same',
            use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = Conv3D(8, (2, 2, 2),
        padding='same',
        kernel_regularizer=regularization,
        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv3D(8, (2, 2, 2),
        padding='same',
        kernel_regularizer=regularization,
        use_bias=False)(x)
    x = BatchNormalization()(x)  x = MaxPooling3D((3, 3, 3),
            strides=(1, 2, 2),
            padding='same')(x)
    x = add([x, residual])
    x = Conv3D(4, (1, 2, 2),
        padding='valid',
        kernel_regularizer=regularization,
        use_bias=False)(x)
    x = MaxPooling3D((1, 3, 3),
            strides=(1, 3, 3),
            padding='valid')(x)
    x = Activation('relu')(x)

    shape = (int(x.shape[1]),
        int(x.shape[2]*x.shape[3]*x.shape[4]))
    x = Reshape(shape)(x)
    x = GlobalMaxPooling1D()(x)
    output = Dense(num_classes, activation='softmax')(x)
    model = Model(img_input, output)
    model.compile(loss='categorical_crossentropy',
            optimizer='adam',
            metrics=['accuracy'])
    return model
# In[18]:
def model_second(input_shape=(5, 200, 200, 1),
num_classes=8, l2_regularization=0.01):
    regularization = l2(l2_regularization)  img_input =
Input(input_shape)
    x = Conv3D(4, (1, 2, 2),
        strides=(1, 1, 1),
        kernel_regularizer=regularization,
        use_bias=False,
        padding='same')(img_input)
    x = MaxPooling3D((2, 2, 2),
            strides=(1, 2, 2),
            padding='same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv3D(8, (1, 2, 2),
        strides=(1, 1, 1),
        kernel_regularizer=regularization,
        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    residual = Conv3D(8, (1, 2, 2),
            strides=(1, 2, 2),
            padding='same',
            use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = Conv3D(8, (2, 2, 2),
        padding='same',
        kernel_regularizer=regularization,
        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv3D(8, (2, 2, 2),
        padding='same',
        kernel_regularizer=regularization,
        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)  x = MaxPooling3D((3, 3, 3),
            strides=(1, 2, 2),
            padding='same')(x)
    x = add([x, residual])

    x = Conv3D(4, (1, 2, 2),
        padding='valid',
        kernel_regularizer=regularization,
        use_bias=False)(x)
    x = Activation('relu')(x)
    x = MaxPooling3D((1, 3, 3),
            strides=(1, 2, 2),
            padding='valid')(x)

    residual = Conv3D(4, (1, 2, 2),
            strides=(1, 2, 2),
            padding='same',
            use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = Conv3D(4, (2, 3, 3),
        padding='same',
        kernel_regularizer=regularization,
        use_bias=False)(x)
    x = Activation('relu')(x)
    x = MaxPooling3D((1, 2, 2),
            strides=(1, 2, 2),
            padding='same')(x)

    x = add([x, residual])

    shape = (int(x.shape[1]),
        int(x.shape[2]*x.shape[3]*x.shape[4]))
    x = Reshape(shape)(x)
    x = GlobalMaxPooling1D()(x)
    output = Dense(num_classes, activation='softmax')(x)
```

```
  model = Model(img_input, output)
  model.compile(loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])
  return model
# In[19]:
def model_third(input_shape=(5, 200, 200, 1),
num_classes=8, l2_regularization=0.01):
  regularization = l2(l2_regularization) img_input =
Input(input_shape)
  x = Conv3D(4, (1, 2, 2),
       strides=(1, 1, 1),
       kernel_regularizer=regularization,
       use_bias=False,
       padding='same')(img_input)
  x = MaxPooling3D((2, 2, 2),
          strides=(1, 2, 2),
          padding='same')(x)
  x = BatchNormalization()(x)
  x = Activation('relu')(x)

  x = Conv3D(8, (1, 2, 2),
       strides=(1, 1, 1),
       kernel_regularizer=regularization,
       use_bias=False)(x)
  x = BatchNormalization()(x)
  x = Activation('relu')(x)  residual = Conv3D(8, (1, 2, 2),
          strides=(1, 2, 2),
          padding='same',
          use_bias=False)(x)
  residual = BatchNormalization()(residual)

  x = Conv3D(8, (2, 2, 2),
       padding='same',
       kernel_regularizer=regularization,
       use_bias=False)(x)
  x = BatchNormalization()(x)
  x = Activation('relu')(x)
  x = Conv3D(8, (2, 2, 2),
       padding='same',
       kernel_regularizer=regularization,
       use_bias=False)(x)
  x = BatchNormalization()(x)
  x = Activation('relu')(x)  x = MaxPooling3D((3, 3, 3),
          strides=(1, 2, 2),
          padding='same')(x)
  x = add([x, residual])

  x = Conv3D(4, (1, 2, 2),
       padding='valid',
       kernel_regularizer=regularization,
       use_bias=False)(x)
  x = Activation('relu')(x)
  x = MaxPooling3D((1, 3, 3),
          strides=(1, 2, 2),
          padding='valid')(x)

  residual = Conv3D(4, (1, 2, 2),
          strides=(1, 2, 2),
          padding='same',
          use_bias=False)(x)
  residual = BatchNormalization()(residual)

  x = Conv3D(4, (2, 3, 3),
       padding='same',
       kernel_regularizer=regularization,
       use_bias=False)(x)
  x = Activation('relu')(x)
  x = MaxPooling3D((1, 2, 2),
          strides=(1, 2, 2),
          padding='same')(x)

  x = add([x, residual])

  residual = Conv3D(4, (1, 2, 2),
          strides=(1, 2, 2),
          padding='same',
          use_bias=False)(x)
  residual = BatchNormalization()(residual)

  x = Conv3D(4, (2, 3, 3),
       padding='same',
       kernel_regularizer=regularization,
       use_bias=False)(x)
  x = Activation('relu')(x)
  x = MaxPooling3D((1, 2, 2),
          strides=(1, 2, 2),
          padding='same')(x)

  x = add([x, residual])

  shape = (int(x.shape[1]),
      int(x.shape[2]*x.shape[3]*x.shape[4]))
  x = Reshape(shape)(x)
  x = GlobalMaxPooling1D()(x)
  output = Dense(8, activation='softmax')(x)
  model = Model(img_input, output)
  model.compile(loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])
  return model
# In[20]:
model = model_first()
# In[21]:
model.summary()
# In[22]:
file_path = 'model_11_20_2093_1.hdf5'
checkpoint = ModelCheckpoint(file_path,
monitor='val_loss', verbose=1, save_best_only=True,
mode='min')
early = EarlyStopping(monitor='val_loss', mode='min',
patience=20)
tensor_board = TensorBoard(log_dir='./logs/1',
histogram_freq=0, batch_size=32, write_graph=True,
write_grads=False, write_images=False,
embeddings_freq=0, embeddings_layer_names=None,
embeddings_metadata=None)
callbacks_list = [checkpoint, early, tensor_board]
# In[23]:
#batch_size = 16
batch_size = 16
epochs = 20
# In[24]:
metadata = metadata.sample(frac=1, random_state=42)
```

```
# In[25]:
train = metadata.iloc[:int(0.8*metadata.shape[0])]
test = metadata.iloc[int(0.8*metadata.shape[0]):]
# In[26]:
gen_train = generator_few_frames(batch_size, train)
gen_test = generator_few_frames(batch_size, test)
# In[27]:
validation_steps=test.shape[0]//batch_size
print(test.shape[0])
print(batch_size)
# In[28]:
print(test)
# In[ ]:
model.fit_generator(gen_train,
        epochs=epochs,
        steps_per_epoch=train.shape[0]//batch_size,
        validation_data=gen_test,
        callbacks=callbacks_list,
        validation_steps=test.shape[0]//batch_size
        )
# In[65]:
model = load_model('model_11_20_2093_1.hdf5')
# In[31]:
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from itertools import product
#beatiful confusion matrix
def plot_confusion_matrix(cm,
            classes,
            normalize=True,
            title='Confusion matrix',
            cmap=plt.cm.spring):
  plt.imshow(cm, interpolation='nearest', cmap=cmap)
  plt.title(title)
  tick_marks = np.arange(len(classes))
  plt.xticks(tick_marks, classes, rotation=45)
  plt.yticks(tick_marks, classes)
  thresh = cm.max() / 2.

  for i, j in product(range(cm.shape[0]),
range(cm.shape[1])):
    if normalize:
      element = cm[i, j] / cm[i].sum()
      element = np.around(element, 3)
    else:
      element = cm[i, j]
    plt.text(j, i, element,
        horizontalalignment='center',
        color='white' if cm[i, j] < thresh else 'black')
plt.tight_layout()
  plt.ylabel('True label', color='#ED7DA0')
  plt.xlabel('Predicted label', color='#ED7DA0')
# In[18]:
y_test_true = []
y_test_pred = []hf_data = h5py.File(base_path + 'data.h5',
'r')
hf_labels = h5py.File(base_path + 'labels.h5', 'r')
for i in range(test.shape[0]):

  tensor = grayscale(hf_data.get(test.iloc[i].file_name))
  label = hf_labels.get(test.iloc[i].file_name)[0]

  tensor = get_few_frames(tensor,
center=tensor.shape[0]//2, n_points=4, step=7).reshape(1,
5, 200, 200, 1)

  y_test_true.append(label)
  y_test_pred.append(model.predict(tensor))
# In[70]:
# for valid set if exists
y_val_true = []
y_val_pred = []hf_data = h5py.File(base_path + 'data.h5',
'r')
hf_labels = h5py.File(base_path + 'labels.h5', 'r')
for i in range(valid.shape[0]):

  tensor = grayscale(hf_data.get(valid.iloc[i].file_name))
  label = hf_labels.get(valid.iloc[i].file_name)[0]

  tensor = get_few_frames(tensor,
center=tensor.shape[0]//2, n_points=4, step=7).reshape(1,
5, 200, 200, 1)

  y_val_true.append(label)
  y_val_pred.append(model.predict(tensor))
# In[33]:
emotions = {
  0: 'angry',
  1: 'calm',
  2: 'disgust',
  3: 'fearful',
  4: 'happy',
  5: 'neutral',
  6: 'sad',
  7: 'surprised'
}
# In[34]:
y_test_pred = np.array(y_test_pred)
y_test_true = np.array(y_test_true)
print(y_test_pred)
print(y_test_true)y_test_pred = y_test_pred.argmax(axis=-
1).reshape(-1,)
y_test_true = y_test_true.argmax(axis=-1)
print(y_test_pred)
print(y_test_true)
# In[73]:
y_val_pred = np.array(y_val_pred)
y_val_true = np.array(y_val_true)y_val_pred =
y_val_pred.argmax(axis=-1).reshape(-1,)
y_val_true = y_val_true.argmax(axis=-1)
# In[35]:
plt.figure(figsize=(7, 7))
plot_confusion_matrix(confusion_matrix(y_test_true,
y_test_pred), list(emotions.values()), cmap=plt.cm.hot)
plt.show()
# In[49]:
plt.figure(figsize=(7, 7))
plot_confusion_matrix(confusion_matrix(y_val_true,
y_val_pred), list(emotions.values()))
plt.show()
# In[50]:
#template
plt.figure(figsize=(7, 7))
```

```
plot_confusion_matrix(confusion_matrix(y_true,
y_predicted), list(emotions.values()))
plt.show()
# In[ ]:
df = pd.read_csv('/home/vaden4d/Documents/logs.csv')
# In[ ]:
df.columns
# In[ ]:
plt.figure(figsize=(8, 6))
plt.plot(df.train_loss, 'o-', label='train', linewidth=3)
plt.plot(df.test_loss, 'o-', label='test', linewidth=3)
plt.legend()
plt.xticks(np.arange(0, 20))
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Cross-entropy train/test')
# In[ ]:
plt.figure(figsize=(8, 6))
plt.plot(df.train_acc, 'o-', label='train', linewidth=3)
plt.plot(df.test_acc, 'o-', label='test', linewidth=3)
plt.legend()
plt.xticks(np.arange(0, 14))
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Accuracy train/test')
# ## visualization# In[ ]:
model = load_model('best13.06.hdf5')
# In[ ]:
model.summary()
# In[ ]:
from keras import backend as K# get 17 intermidiate layer
in test phase
get_3rd_layer_output = K.function([model.layers[0].input,
K.learning_phase()],
                  [model.layers[17].output])
layer_output = get_3rd_layer_output([x, 0])[0]
# In[ ]:
# get 4 intermidiate layer in test phaseget_3rd_layer_output
= K.function([model.layers[0].input, K.learning_phase()],
                  [model.layers[4].output])
layer_output = get_3rd_layer_output([x, 0])[0]
# In[ ]:
layer_output.shape
# In[ ]:
image_num = 1
# In[ ]:
test = layer_output[image_num]
# In[ ]:
test.shape
# In[ ]:
imshow(test[0, :, :, 2], cmap='nipy_spectral')
plt.colorbar()
# In[ ]:
imshow(x[image_num, 0, :, :, 0])
# In[ ]:
def     plot_feature_maps(original_tensor,     output_tensor,
save=True):
  """For tensor of shape (n_frames, size_1, size_2, channels)
plot feature map of
  each time frame for each channel. channels*n_frames
plots."""
  n_frames = output_tensor.shape[0]
  n_channels = output_tensor.shape[-1]
  f,    axarr    =    plt.subplots(n_frames,    n_channels+1,
figsize=(20, 20))
  for i in range(n_frames):
    axarr[i, 0].axis('off')
    img  =  axarr[i,  0].imshow(original_tensor[i,  :,  :,  0],
cmap='gray')
  for i in range(n_frames):
    for j in range(n_channels):
      axarr[i, j+1].axis('off')
      img  =  axarr[i,  j+1].imshow(output_tensor[i,  :,  :,  j],
cmap='nipy_spectral')
  if save:
    f.savefig('feature_maps.png', dpi=300)

def  get_few_frames_vis(tensor,  center,  n_points,  step,
eq_hist=True, blur=True):
  """From video tensor with shape (n_frames, ...) get
n_points
  frames around center with some step"""
  indexes      =      center      +      np.arange(-n_points//2,
n_points//2+1)*step
  indexes = indexes[:tensor.shape[0]]
  tensor = tensor[indexes]
  tensor_one = tensor.copy()

  if eq_hist:
    shape_3d      =      (tensor.shape[1],      tensor.shape[2],
tensor.shape[3])
    for i in range(tensor.shape[0]):
      tensor[i]                                                =
cv2.equalizeHist(tensor[i].astype('uint8')).reshape(shape_3
d)
  tensor_two = tensor.copy()
  if blur:
    shape_3d      =      (tensor.shape[1],      tensor.shape[2],
tensor.shape[3])
    for i in range(tensor.shape[0]):
      tensor[i]  =  cv2.GaussianBlur(tensor[i].astype('uint8'),
(5, 5), 3).reshape(shape_3d)
  tensor_three = tensor.copy()
  return      tensor_one,      tensor_two,      tensor_threedef
generator_vis(batch_size=1, metadata=metadata):
  """Generate uniformly few frames around center of the
video (center included)"""
  hf_data = h5py.File(base_path + 'data.h5', 'r')
  hf_labels = h5py.File(base_path + 'labels.h5', 'r')
  while True:
    n_batches                                                  =
np.ceil(metadata.shape[0]/batch_size).astype(int)
    for i in range(n_batches):
      file_names                                               =
metadata.file_name[i*batch_size:(i+1)*batch_size]
      batch, labels = [], []
      for name in file_names:
        tensor = grayscale(hf_data.get(name))
        label = hf_labels.get(name)[0]

        batch.append(tensor)
        labels.append(label)

      batch = np.array(batch).astype('uint8')
```

```
    labels = np.array(labels)
    yield batch, labels

def plot_preprocessing(tensor, save=True):
 tensors = get_few_frames_vis(tensor, center=tensor.shape[0]//2, n_points=4, step=7)
 f, axarr = plt.subplots(3, 5, figsize=(12, 12))
 for i in range(3):
  for j in range(5):
   axarr[i, j].axis('off')
   img = axarr[i, j].imshow(tensors[i][j][:, :, 0], cmap='gray')
 if save:
  f.savefig('preprocessing.png', dpi=300)
# In[ ]:
y, _ = next(generator_vis())
y, _ = next(generator_vis())
# In[ ]:
plot_preprocessing(y[0])
# In[ ]:
plot_feature_maps(x[image_num], test)
# In[ ]:
del model
```

Скрипт для навчання та оцінки якості нейронних мереж на основі ключових точок

```
# coding: utf-8# In[1]:
import tensorflow as tf
# In[2]:
from matplotlib.pyplot import imshow
import numpy as np
import pandas as pd
import h5py
import matplotlib.pyplot as plt
import cv2
import time
# In[3]:
#changing style of plotsfrom matplotlib import rcParams
rcParams['axes.facecolor'] = 'white'
rcParams['figure.facecolor'] = 'white'
rcParams['font.weight'] = 'bold'
rcParams['axes.titleweight'] = 'bold'
rcParams['xtick.color'] = '#64E0C0'
rcParams['ytick.color'] = '#64E0C0'
rcParams['grid.color'] = '#64E0C0'
# In[4]:
from keras.layers import Activation, Conv3D, Conv2D, Conv1D
from keras.layers import BatchNormalization
from keras.layers import Reshape, Dense, LSTM, GRU, RNN
from keras.models import Model, load_model, Sequential
from keras.layers import Input, add
from keras.layers import MaxPooling3D, GlobalMaxPooling1D, Flatten, MaxPooling1D
from keras.regularizers import l2
from keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard
from keras.layers import Dense, Activation, Dropout
# In[5]:
def sliding_window(data, size, stepsize=1, axis=0, copy=True):
 """Get slices of the video tensor"""
 if axis >= data.ndim:
  raise ValueError(
   "Axis value out of range"
  )
 if stepsize < 1:
  raise ValueError(
   "Stepsize may not be zero or negative"
  )
 if size > data.shape[axis]:
  raise ValueError(
   "Sliding window size may not exceed size of selected axis"
  )
 shape = list(data.shape)
 shape[axis] = np.floor(data.shape[axis] / stepsize - size / stepsize + 1).astype(int)
 shape.insert(1, size)

 strides = list(data.strides)
 strides[axis] *= stepsize
 strides.insert(1, data.strides[axis])
 strided = np.lib.stride_tricks.as_strided(
  data, shape=shape, strides=strides
 ) if copy:
  return strided.copy()
 else:
  return strided

def get_few_frames(tensor, center, n_points, step):
 """From video tensor with shape (n_frames, ...) get n_points
 frames around center with some step"""
 indexes = center + np.arange(-n_points//2, n_points//2+1)*step
 indexes = indexes[:tensor.shape[0]]
 tensor = tensor[indexes]
 return tensordef get_diff(tensor):
 res = []
 for i in range(1, len(tensor)):
  res.append(100*(tensor[i]-tensor[i-1]))
 return np.array(res)
# In[6]:
base_path = 'bin\\'
# In[7]:
metadata = pd.read_csv(base_path + 'metadata.csv')
# In[8]:
# for my model only
metadata = metadata.loc[metadata.modality == 'video']
print(len(metadata))
```

```
# In[9]:
emotions = {
  0: 'angry',
  1: 'calm',
  2: 'disgust',
  3: 'fearful',
  4: 'happy',
  5: 'neutral',
  6: 'sad',
  7: 'surprised'
}
# In[10]:
def generator_few_frames(batch_size=16, metadata=metadata):
  """Generate uniformly few frames around center of the video (center included)"""
  hf_data = h5py.File(base_path + 'data_dlib.h5', 'r')
  hf_labels = h5py.File(base_path + 'labels_dlib.h5', 'r')
  while True:
    n_batches = np.ceil(metadata.shape[0]/batch_size).astype(int)
    for i in range(n_batches):
      file_names = metadata.file_name[i*batch_size:(i+1)*batch_size]
      batch, labels = [], []
      for name in file_names:
        tensor = hf_data.get(name)
        if not(tensor is None):
          tensor = tensor[:,:]
          tensor = get_few_frames(tensor, center=tensor.shape[0]//2, n_points=14, step=5)
          batch.append(tensor)
          label = hf_labels.get(name)[0]
          labels.append(label)

      batch = np.array(batch)
      #.astype('uint8')
      labels = np.array(labels)
      yield batch, labels
# In[11]:
g = generator_few_frames(batch_size=64)
# In[12]:
x, label = next(g)
# In[13]:
print(x[4])
print(label[4])
print(x[5])
print(label[5])
# In[14]:
x.shape
# In[15]:
def model_1(input_shape=(15, 4624), num_classes=8, l2_regularization=0.0001):
  regularization = l2(l2_regularization)
#Result - 0.5

  # base
  img_input = Input(input_shape)
  x = img_input

  # create model
  model = Sequential()
  model.add(Dense(512, input_shape=input_shape, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(GlobalMaxPooling1D())
  model.add(Dense(output_dim=8, activation='softmax'))

  model.compile(loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])
  return model
# In[42]:
def model_2(input_shape=(15, 4624), num_classes=8, l2_regularization=0.0001):
  regularization = l2(l2_regularization)

  # base
  img_input = Input(input_shape)
  x = img_input

  # create model
  model = Sequential()
  model.add(Dense(512, input_shape=input_shape, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(GlobalMaxPooling1D())
  model.add(Dense(output_dim=8, activation='softmax'))

  model.compile(loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])
  return model
# In[17]:
def model_3(input_shape=(8, 4624), num_classes=8, l2_regularization=0.0001):
  regularization = l2(l2_regularization)
#Result - 0.6

  # base
  img_input = Input(input_shape)
  x = img_input

  # create model
  model = Sequential()
  model.add(Dense(1024, input_shape=input_shape, activation='relu'))
  model.add(Dense(1024, activation='relu'))
  model.add(Dense(1024, activation='relu'))
  model.add(Dense(1024, activation='relu'))
  model.add(GlobalMaxPooling1D())
  model.add(Dense(output_dim=8, activation='softmax'))

  model.compile(loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])
  return model
# In[18]:
def model_4(input_shape=(5, 4624), num_classes=8, l2_regularization=0.01):
```

```
  regularization = l2(l2_regularization)
#Result - 0.5

  # base
  img_input = Input(input_shape)
  x = img_input

  # create model
  model = Sequential()
  model.add(Dense(1024,           input_shape=input_shape,
activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(GRU(256, activation='relu'))
  #model.add(GlobalMaxPooling1D())
  model.add(Dense(output_dim=8, activation='softmax'))

  model.compile(loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])
  return model
# In[19]:
def    model_5(input_shape=(8,    4624),    num_classes=8,
l2_regularization=0.01):
  regularization = l2(l2_regularization)
#Result - 0.65

  # base
  img_input = Input(input_shape)
  x = img_input

  # create model
  model = Sequential()
  model.add(Conv1D(4, (2), input_shape=input_shape,
      strides=(1),
      kernel_regularizer=regularization,
      use_bias=False,
      padding='same'))
  model.add(MaxPooling1D((2),
        strides=(1),
        padding='same'))
  model.add(BatchNormalization())
  model.add(Activation('relu'))
  model.add(Dense(1024, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(GRU(256, activation='relu'))
  #model.add(GlobalMaxPooling1D())
  model.add(Dense(output_dim=8, activation='softmax'))

  model.compile(loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])
  return model
# In[20]:
def    model_6(input_shape=(15,    4624),    num_classes=8,
l2_regularization=0.0001):
  regularization = l2(l2_regularization)

  # base
  img_input = Input(input_shape)
  x = img_input

  # create model
  model = Sequential()

  model.add(Conv1D(4, (2), input_shape=input_shape,
      strides=(1),
      kernel_regularizer=regularization,
      use_bias=True,
      padding='same'))
  model.add(MaxPooling1D((2),
        strides=(1),
        padding='same'))
  model.add(BatchNormalization())
  model.add(Activation('relu'))
  model.add(Dense(1024, activation='relu'))

  model.add(Dense(512, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(Dense(512, activation='relu'))
  model.add(GlobalMaxPooling1D())
  model.add(Dense(output_dim=8, activation='softmax'))

  model.compile(loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])
  return model
# In[43]:
model = model_2()
# In[44]:
model.summary()
# In[23]:
file_path = 'model_11_25_2093_dlib_2.hdf5'
checkpoint          =          ModelCheckpoint(file_path,
monitor='val_loss',    verbose=1,    save_best_only=True,
mode='min')
early  =  EarlyStopping(monitor='val_loss',  mode='min',
patience=20)
tensor_board     =     TensorBoard(log_dir='./logs/dlib_2',
histogram_freq=0,    batch_size=32,    write_graph=True,
write_grads=False,                write_images=False,
embeddings_freq=0,       embeddings_layer_names=None,
embeddings_metadata=None)
callbacks_list = [checkpoint, early, tensor_board]
# In[24]:
#batch_size = 16
batch_size = 64
epochs = 20
# In[25]:
metadata = metadata.sample(frac=1, random_state=42)
# In[26]:
len(metadata)
# In[27]:
train = metadata.iloc[:int(0.9*metadata.shape[0])]
test = metadata.iloc[int(0.4*metadata.shape[0]):]
len(train)
# In[28]:
gen_train = generator_few_frames(batch_size, train)
gen_test = generator_few_frames(batch_size, test)
# In[29]:
validation_steps=test.shape[0]//batch_size
```

```
print(validation_steps)
print(gen_train)
# In[30]:
print(train)
# In[31]:
model.fit_generator(gen_train,
        epochs=epochs,
        steps_per_epoch=train.shape[0]//batch_size,
        validation_data=gen_test,
        callbacks=callbacks_list,
        validation_steps=test.shape[0]//batch_size
        )
# In[33]:
model = load_model('model_11_25_2093_dlib_2.hdf5')
# In[34]:
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from itertools import product
#beatiful confusion matrix
def plot_confusion_matrix(cm,
          classes,
          normalize=True,
          title='Confusion matrix',
          cmap=plt.cm.hot):
  plt.imshow(cm, interpolation='nearest', cmap=cmap)
  plt.title(title)
  tick_marks = np.arange(len(classes))
  plt.xticks(tick_marks, classes, rotation=45)
  plt.yticks(tick_marks, classes)
  thresh = cm.max() / 2.

  norm = []

  for i, j in product(range(cm.shape[0]),
range(cm.shape[1])):
    if normalize:
      element = cm[i, j] / cm[i].sum()
      element = np.around(element, 3)
    else:
      element = cm[i, j]
    norm.append(element)
    plt.text(j, i, element,
        horizontalalignment='center',
        color='black' if cm[i, j] < thresh else 'white')
  print(norm)
  plt.tight_layout()
  plt.ylabel('True label', color='black')
  plt.xlabel('Predicted label', color='black')
# In[35]:
y_test_true = []
y_test_pred = []hf_data = h5py.File(base_path +
'data_dlib.h5', 'r')
hf_labels = h5py.File(base_path + 'labels_dlib.h5', 'r')
for i in range(test.shape[0]):
#for i in range(0,1):

  tensor = hf_data.get(test.iloc[i].file_name)
  tensor = tensor[:,:]
  tensor = get_few_frames(tensor,
center=tensor.shape[0]//2, n_points=14, step=5)
  tensor = tensor.reshape(1, 15, 4624)
  label = hf_labels.get(test.iloc[i].file_name)[0]
  y_test_true.append(label)
  y_test_pred.append(model.predict(tensor))
# In[36]:
emotions = {
  0: 'angry',
  1: 'calm',
  2: 'disgust',
  3: 'fearful',
  4: 'happy',
  5: 'neutral',
  6: 'sad',
  7: 'surprised'
}
# In[37]:
y_test_pred = np.array(y_test_pred)
y_test_true = np.array(y_test_true)
print(y_test_pred)
print(y_test_true)
y_test_pred = y_test_pred.argmax(axis=-1).reshape(-1,)
y_test_true = y_test_true.argmax(axis=-1)
print(y_test_pred)
print(y_test_true)
# In[38]:
cm = confusion_matrix(y_test_true, y_test_pred)
print(cm)
# In[40]:
cm = np.array(cm)
print(cm)
# In[41]:
plt.figure(figsize=(7, 7))
plot_confusion_matrix(cm, list(emotions.values()),
cmap=plt.cm.Blues)
plt.show()
# In[49]:
plt.figure(figsize=(7, 7))
plot_confusion_matrix(confusion_matrix(y_val_true,
y_val_pred), list(emotions.values()))
plt.show()
# In[50]:
#template
plt.figure(figsize=(7, 7))
plot_confusion_matrix(confusion_matrix(y_true,
y_predicted), list(emotions.values()))
plt.show()
# In[ ]:
df = pd.read_csv('/home/vaden4d/Documents/logs.csv')
# In[ ]:
df.columns
# In[ ]:
plt.figure(figsize=(8, 6))
plt.plot(df.train_loss, 'o-', label='train', linewidth=3)
plt.plot(df.test_loss, 'o-', label='test', linewidth=3)
plt.legend()
plt.xticks(np.arange(0, 20))
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Cross-entropy train/test')
# In[ ]:
plt.figure(figsize=(8, 6))
plt.plot(df.train_acc, 'o-', label='train', linewidth=3)
plt.plot(df.test_acc, 'o-', label='test', linewidth=3)
plt.legend()
```

```
plt.xticks(np.arange(0, 14))
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Accuracy train/test')
# ## visualization# In[ ]:
model = load_model('best13.06.hdf5')
# In[ ]:
model.summary()
# In[ ]:
from keras import backend as K# get 17 intermidiate layer in test phase
get_3rd_layer_output = K.function([model.layers[0].input, K.learning_phase()],
                  [model.layers[17].output])
layer_output = get_3rd_layer_output([x, 0])[0]
# In[ ]:
# get 4 intermidiate layer in test phaseget_3rd_layer_output = K.function([model.layers[0].input, K.learning_phase()],
                  [model.layers[4].output])
layer_output = get_3rd_layer_output([x, 0])[0]
# In[ ]:
layer_output.shape
# In[ ]:
image_num = 1
# In[ ]:
test = layer_output[image_num]
# In[ ]:
test.shape
# In[ ]:
imshow(test[0, :, :, 2], cmap='nipy_spectral')
plt.colorbar()
# In[ ]:
imshow(x[image_num, 0, :, :, 0])
# In[ ]:
def plot_feature_maps(original_tensor, output_tensor, save=True):
  """For tensor of shape (n_frames, size_1, size_2, channels) plot feature map of
  each time frame for each channel. channels*n_frames plots."""
  n_frames = output_tensor.shape[0]
  n_channels = output_tensor.shape[-1]
  f, axarr = plt.subplots(n_frames, n_channels+1, figsize=(20, 20))
  for i in range(n_frames):
    axarr[i, 0].axis('off')
    img = axarr[i, 0].imshow(original_tensor[i, :, :, 0], cmap='gray')
  for i in range(n_frames):
    for j in range(n_channels):
      axarr[i, j+1].axis('off')
      img = axarr[i, j+1].imshow(output_tensor[i, :, :, j], cmap='nipy_spectral')
  if save:
    f.savefig('feature_maps.png', dpi=300)

def get_few_frames_vis(tensor, center, n_points, step, eq_hist=True, blur=True):
  """From video tensor with shape (n_frames, ...) get n_points
  frames around center with some step"""
  indexes = center + np.arange(-n_points//2, n_points//2+1)*step
  indexes = indexes[:tensor.shape[0]]
  tensor = tensor[indexes]
  tensor_one = tensor.copy()

  if eq_hist:
    shape_3d = (tensor.shape[1], tensor.shape[2], tensor.shape[3])
    for i in range(tensor.shape[0]):
      tensor[i] = cv2.equalizeHist(tensor[i].astype('uint8')).reshape(shape_3d)
  tensor_two = tensor.copy()
  if blur:
    shape_3d = (tensor.shape[1], tensor.shape[2], tensor.shape[3])
    for i in range(tensor.shape[0]):
      tensor[i] = cv2.GaussianBlur(tensor[i].astype('uint8'), (5, 5), 3).reshape(shape_3d)
  tensor_three = tensor.copy()
  return tensor_one, tensor_two, tensor_threedef generator_vis(batch_size=1, metadata=metadata):
  """Generate uniformly few frames around center of the video (center included)"""
  hf_data = h5py.File(base_path + 'data.h5', 'r')
  hf_labels = h5py.File(base_path + 'labels.h5', 'r')
  while True:
    n_batches = np.ceil(metadata.shape[0]/batch_size).astype(int)
    for i in range(n_batches):
      file_names = metadata.file_name[i*batch_size:(i+1)*batch_size]
      batch, labels = [], []
      for name in file_names:
        tensor = grayscale(hf_data.get(name))
        label = hf_labels.get(name)[0]

        batch.append(tensor)
        labels.append(label)

      batch = np.array(batch).astype('uint8')
      labels = np.array(labels)
      yield batch, labels

def plot_preprocessing(tensor, save=True):
  tensors = get_few_frames_vis(tensor, center=tensor.shape[0]//2, n_points=4, step=7)
  f, axarr = plt.subplots(3, 5, figsize=(12, 12))
  for i in range(3):
    for j in range(5):
      axarr[i, j].axis('off')
      img = axarr[i, j].imshow(tensors[i][j][:, :, 0], cmap='gray')
  if save:
    f.savefig('preprocessing.png', dpi=300)
# In[ ]:
y, _ = next(generator_vis())
y, _ = next(generator_vis())
# In[ ]:
plot_preprocessing(y[0])
# In[ ]:
```

```
plot_feature_maps(x[image_num], test)
# In[ ]:
```

```
del model
```

ДОДАТОК Б ДІАГРАМИ АРХІТЕКТУР МОДЕЛЕЙ

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 5, 200, 200,	0	
conv3d_1 (Conv3D)	(None, 5, 200, 200,	16	input_1[0][0]
max_pooling3d_1 (MaxPooling3D)	(None, 5, 100, 100,	0	conv3d_1[0][0]
batch_normalization_1 (BatchNor	(None, 5, 100, 100,	16	max_pooling3d_1[0][0]
activation_1 (Activation)	(None, 5, 100, 100,	0	batch_normalization_1[0][0]
conv3d_2 (Conv3D)	(None, 5, 99, 99, 8)	128	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, 5, 99, 99, 8)	32	conv3d_2[0][0]
activation_2 (Activation)	(None, 5, 99, 99, 8)	0	batch_normalization_2[0][0]
conv3d_4 (Conv3D)	(None, 5, 99, 99, 8)	512	activation_2[0][0]
batch_normalization_4 (BatchNor	(None, 5, 99, 99, 8)	32	conv3d_4[0][0]
activation_3 (Activation)	(None, 5, 99, 99, 8)	0	batch_normalization_4[0][0]
conv3d_5 (Conv3D)	(None, 5, 99, 99, 8)	512	activation_3[0][0]
batch_normalization_5 (BatchNor	(None, 5, 99, 99, 8)	32	conv3d_5[0][0]
conv3d_3 (Conv3D)	(None, 5, 50, 50, 8)	256	activation_2[0][0]
max_pooling3d_2 (MaxPooling3D)	(None, 5, 50, 50, 8)	0	batch_normalization_5[0][0]
batch_normalization_3 (BatchNor	(None, 5, 50, 50, 8)	32	conv3d_3[0][0]
add_1 (Add)	(None, 5, 50, 50, 8)	0	max_pooling3d_2[0][0] batch_normalization_3[0][0]
conv3d_6 (Conv3D)	(None, 5, 49, 49, 4)	128	add_1[0][0]
max_pooling3d_3 (MaxPooling3D)	(None, 5, 16, 16, 4)	0	conv3d_6[0][0]
activation_4 (Activation)	(None, 5, 16, 16, 4)	0	max_pooling3d_3[0][0]
reshape_1 (Reshape)	(None, 5, 1024)	0	activation_4[0][0]
global_max_pooling1d_1 (GlobalM	(None, 1024)	0	reshape_1[0][0]
dense_1 (Dense)	(None, 8)	8200	global_max_pooling1d_1[0][0]

Total params: 9,896
Trainable params: 9,824
Non-trainable params: 72

Рисунок Б.1 – Детальна діаграма згорткової нейронної мережі (Модель 3)

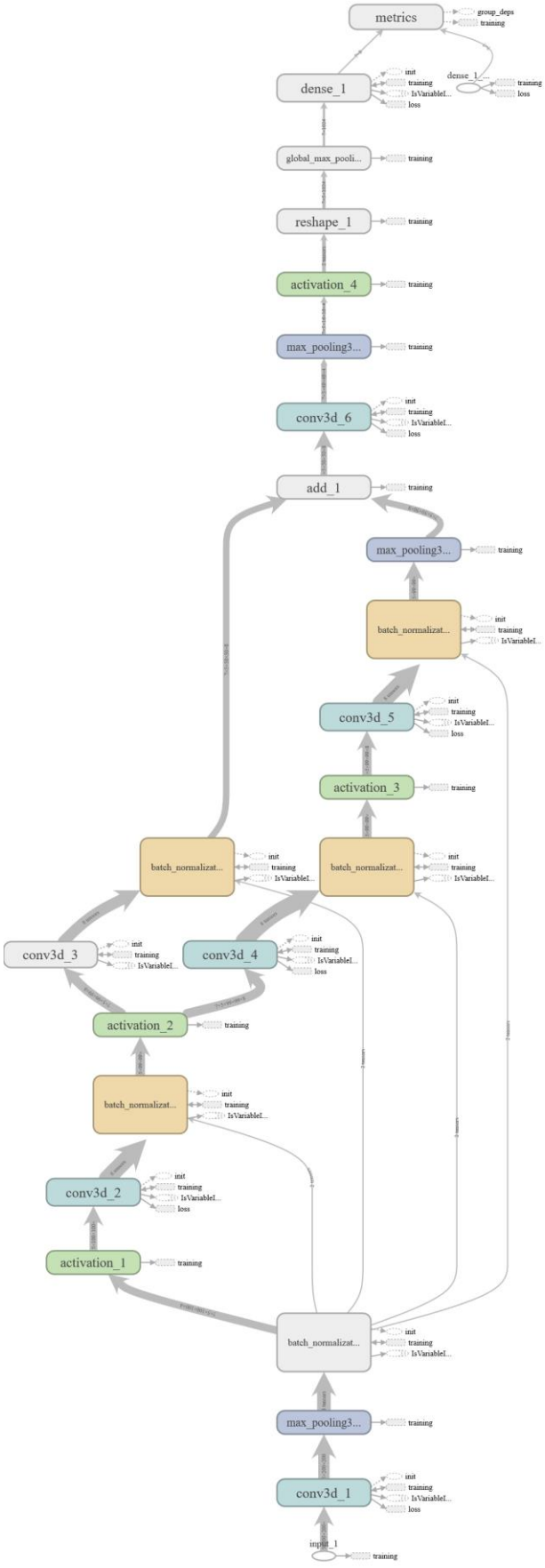

Рисунок Б.2 – Діаграма згорткової нейронної мережі (Модель 3)

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	(None, 5, 200, 200,	0	
conv3d_15 (Conv3D)	(None, 5, 200, 200,	16	input_3[0][0]
max_pooling3d_8 (MaxPooling3D)	(None, 5, 100, 100,	0	conv3d_15[0][0]
batch_normalization_12 (BatchNo	(None, 5, 100, 100,	16	max_pooling3d_8[0][0]
activation_11 (Activation)	(None, 5, 100, 100,	0	batch_normalization_12[0][0]
conv3d_16 (Conv3D)	(None, 5, 99, 99, 8)	128	activation_11[0][0]
batch_normalization_13 (BatchNo	(None, 5, 99, 99, 8)	32	conv3d_16[0][0]
activation_12 (Activation)	(None, 5, 99, 99, 8)	0	batch_normalization_13[0][0]
conv3d_18 (Conv3D)	(None, 5, 99, 99, 8)	512	activation_12[0][0]
batch_normalization_15 (BatchNo	(None, 5, 99, 99, 8)	32	conv3d_18[0][0]
activation_13 (Activation)	(None, 5, 99, 99, 8)	0	batch_normalization_15[0][0]
conv3d_19 (Conv3D)	(None, 5, 99, 99, 8)	512	activation_13[0][0]
batch_normalization_16 (BatchNo	(None, 5, 99, 99, 8)	32	conv3d_19[0][0]
activation_14 (Activation)	(None, 5, 99, 99, 8)	0	batch_normalization_16[0][0]
conv3d_17 (Conv3D)	(None, 5, 50, 50, 8)	256	activation_12[0][0]
max_pooling3d_9 (MaxPooling3D)	(None, 5, 50, 50, 8)	0	activation_14[0][0]
batch_normalization_14 (BatchNo	(None, 5, 50, 50, 8)	32	conv3d_17[0][0]
add_4 (Add)	(None, 5, 50, 50, 8)	0	max_pooling3d_9[0][0] batch_normalization_14[0][0]
conv3d_20 (Conv3D)	(None, 5, 49, 49, 4)	128	add_4[0][0]
activation_15 (Activation)	(None, 5, 49, 49, 4)	0	conv3d_20[0][0]
max_pooling3d_10 (MaxPooling3D)	(None, 5, 24, 24, 4)	0	activation_15[0][0]
conv3d_22 (Conv3D)	(None, 5, 24, 24, 4)	288	max_pooling3d_10[0][0]
activation_16 (Activation)	(None, 5, 24, 24, 4)	0	conv3d_22[0][0]
conv3d_21 (Conv3D)	(None, 5, 12, 12, 4)	64	max_pooling3d_10[0][0]
max_pooling3d_11 (MaxPooling3D)	(None, 5, 12, 12, 4)	0	activation_16[0][0]
batch_normalization_17 (BatchNo	(None, 5, 12, 12, 4)	16	conv3d_21[0][0]
add_5 (Add)	(None, 5, 12, 12, 4)	0	max_pooling3d_11[0][0] batch_normalization_17[0][0]
reshape_3 (Reshape)	(None, 5, 576)	0	add_5[0][0]
global_max_pooling1d_3 (GlobalM	(None, 576)	0	reshape_3[0][0]
dense_3 (Dense)	(None, 8)	4616	global_max_pooling1d_3[0][0]

Total params: 6,680
Trainable params: 6,600
Non-trainable params: 80

Рисунок Б.3 – Детальна діаграма згорткової нейронної мережі (Модель 4)

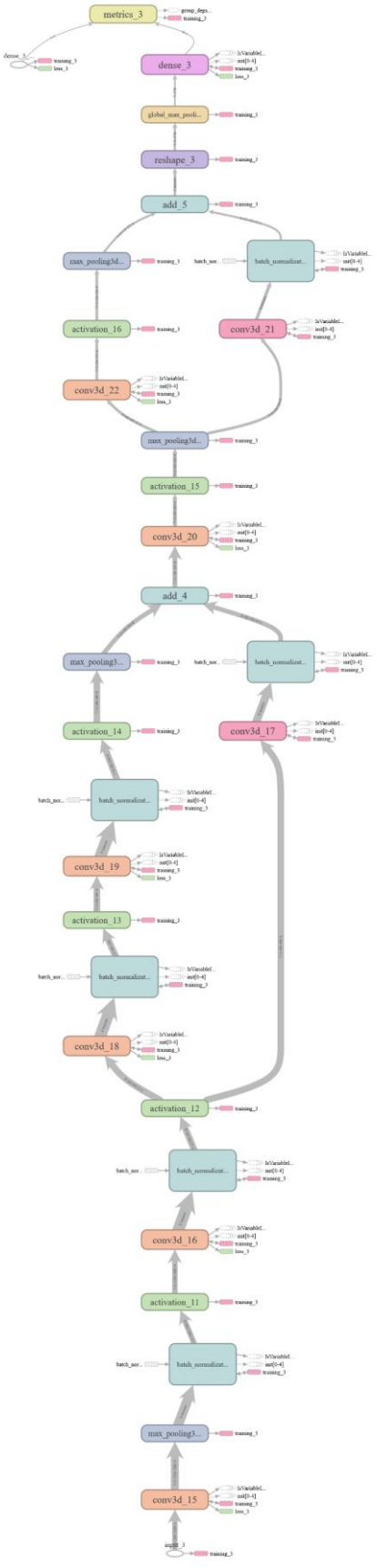

Рисунок Б.4 – Діаграма згорткової нейронної мережі (Модель 4)

Layer (type)	Output Shape	Param #	Connected to
input_4 (InputLayer)	(None, 5, 200, 200,	0	
conv3d_23 (Conv3D)	(None, 5, 200, 200,	16	input_4[0][0]
max_pooling3d_12 (MaxPooling3D)	(None, 5, 100, 100,	0	conv3d_23[0][0]
batch_normalization_18 (BatchNo	(None, 5, 100, 100,	16	max_pooling3d_12[0][0]
activation_17 (Activation)	(None, 5, 100, 100,	0	batch_normalization_18[0][0]
conv3d_24 (Conv3D)	(None, 5, 99, 99, 8)	128	activation_17[0][0]
batch_normalization_19 (BatchNo	(None, 5, 99, 99, 8)	32	conv3d_24[0][0]
activation_18 (Activation)	(None, 5, 99, 99, 8)	0	batch_normalization_19[0][0]
conv3d_26 (Conv3D)	(None, 5, 99, 99, 8)	512	activation_18[0][0]
batch_normalization_21 (BatchNo	(None, 5, 99, 99, 8)	32	conv3d_26[0][0]
activation_19 (Activation)	(None, 5, 99, 99, 8)	0	batch_normalization_21[0][0]
conv3d_27 (Conv3D)	(None, 5, 99, 99, 8)	512	activation_19[0][0]
batch_normalization_22 (BatchNo	(None, 5, 99, 99, 8)	32	conv3d_27[0][0]
activation_20 (Activation)	(None, 5, 99, 99, 8)	0	batch_normalization_22[0][0]
conv3d_25 (Conv3D)	(None, 5, 50, 50, 8)	256	activation_18[0][0]
max_pooling3d_13 (MaxPooling3D)	(None, 5, 50, 50, 8)	0	activation_20[0][0]
batch_normalization_20 (BatchNo	(None, 5, 50, 50, 8)	32	conv3d_25[0][0]
add_6 (Add)	(None, 5, 50, 50, 8)	0	max_pooling3d_13[0][0] batch_normalization_20[0][0]
conv3d_28 (Conv3D)	(None, 5, 49, 49, 4)	128	add_6[0][0]
activation_21 (Activation)	(None, 5, 49, 49, 4)	0	conv3d_28[0][0]
max_pooling3d_14 (MaxPooling3D)	(None, 5, 24, 24, 4)	0	activation_21[0][0]
conv3d_30 (Conv3D)	(None, 5, 24, 24, 4)	288	max_pooling3d_14[0][0]
activation_22 (Activation)	(None, 5, 24, 24, 4)	0	conv3d_30[0][0]
conv3d_29 (Conv3D)	(None, 5, 12, 12, 4)	64	max_pooling3d_14[0][0]
max_pooling3d_15 (MaxPooling3D)	(None, 5, 12, 12, 4)	0	activation_22[0][0]
batch_normalization_23 (BatchNo	(None, 5, 12, 12, 4)	16	conv3d_29[0][0]
add_7 (Add)	(None, 5, 12, 12, 4)	0	max_pooling3d_15[0][0] batch_normalization_23[0][0]
conv3d_32 (Conv3D)	(None, 5, 12, 12, 4)	288	add_7[0][0]
activation_23 (Activation)	(None, 5, 12, 12, 4)	0	conv3d_32[0][0]
conv3d_31 (Conv3D)	(None, 5, 6, 6, 4)	64	add_7[0][0]
max_pooling3d_16 (MaxPooling3D)	(None, 5, 6, 6, 4)	0	activation_23[0][0]
batch_normalization_24 (BatchNo	(None, 5, 6, 6, 4)	16	conv3d_31[0][0]
add_8 (Add)	(None, 5, 6, 6, 4)	0	max_pooling3d_16[0][0] batch_normalization_24[0][0]
reshape_4 (Reshape)	(None, 5, 144)	0	add_8[0][0]
global_max_pooling1d_4 (GlobalM	(None, 144)	0	reshape_4[0][0]
dense_4 (Dense)	(None, 8)	1160	global_max_pooling1d_4[0][0]

Total params: 3,592
Trainable params: 3,504
Non-trainable params: 88

Рисунок Б.5 – Діаграма згорткової нейронної мережі (Модель 5)

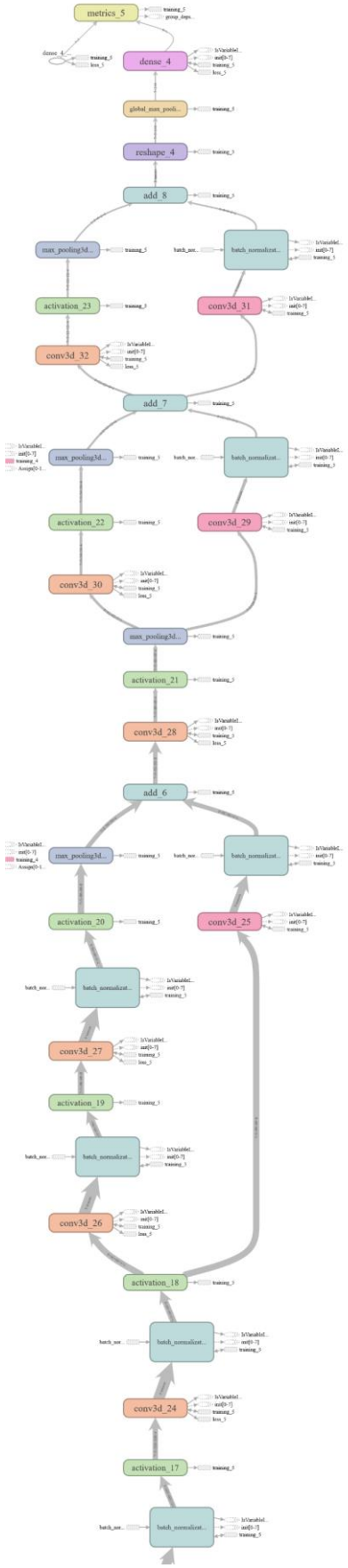

Рисунок Б.6 – Детальна діаграма згорткової нейронної мережі (Модель 5)

ДОДАТОК В БІЗНЕС МОДЕЛІ СТАРТАП ПРОЕКТІВ

Таблиця В.1 – Бізнес модель стартап проекту

Ключові партнери - Компанія CameraDigit», яка буде постачати камери - Компанія Microsoft, в якої будуть орендуватись хмарні технології Azure, які включають сервери, бази даних у хмарі, на яких буде публікуватись продукт для загального доступу по ключу -Сайти ІТ-розробки для розміщення реклами	Ключові види діяльності - Підтримка та вдосконалення продукту по аналізу виступів перед аудиторією за допомогою емоційної аналітики	Ціннісна пропозиція - Зменшення витрат на безкорисні зустрічі працівників - Зменшення витрат на малоефективні виступи, адже доповідачі зможуть корегувати свої дії при наступних виступах	Взаємовідносини з клієнтами - Реклама в інтернеті -Пропозиція автоматизованого або персонального обслуговування: користувач може самостійно налаштувати камери та отримати ключі до системи, або може звернутись до підтримки, яка це здійснить за них	Споживчі сегменти - Менеджери, CEO - Викладачі -Будь-які особи, що виступають перед аудиторією
	Ключові ресурси камери сервери та БД моделі для розпізнавання емоцій та для аналізу даних працівники	- Збільшення прибутків споживачів, які, наприклад, проводять презентації власного продукту і їм важливо підтримувати увагу аудиторії	Канали збуту - Продаж системи через наш сайт	
Структура витрат Закупівля камер Оренда серверів та баз даних Оплата електронернігії для розробки, підтримки та вдосконалення продукту Найбільш затратними буде час розробників Витрати на рекламу		Потоки доходів Продаж сервісу через абонентську плату користувачів, яка буде 2 видів: Оплата за період користування Оплата за кількість звернень до сервісу		

Таблиця В.2 – Бізнес модель продукту конкурента MeetingPulse

Ключові партнери -AngelList – для пошуку працівників HYPERLINK "https://angel.co/macha c c e l e r a t o r	Ключові види діяльності - Надання можливості користувачам спростити аналіз зацікавленості аудиторії та підвищити її при зустрічах шляхом проведення простих опитувань -Можливість спрощення організації зустрічей	Ключові ресурси Працівники Сервери для розміщення акаунтів користувачів та збереження даних про них Сервери для розміщення сервісу	Цінніснa пропозиція - Зменшення часу на рутинну роботу -Збільшення корисностей зустрічей, адже увага аудиторії аналізується та підвищується оповідачем	Взаємовідносини з клієнтами - Реклама в інтернеті та в продуктах партнерів -Веб-сайт -Можливість користуватись скороченою версією безкоштовно	Споживчі сегменти - Менеджери,
				Канали збуту - Інтернет, компанії-партери, що забезпечують можливість проведення зустрічей	
Структура витрат Оренда серверів та баз дани Оплата електронернігії для розробки, підтримки та вдосконалення продукту Найбільш затратними буде час розробників Витрати на рекламу			Потоки доходів Продаж сервісу через підписку, яка становить 150-850$ за місяць у залежності від додаткових можливостей продукту		

Таблиця В.3 – Бізнес модель продукту конкурента Voicera+Wrappup

<table>
<tr><td rowspan="2">Ключові партнери
- Bluejeans, Skype, Webex, UberConference, Zoom, GoogleHangouts, Highfive, GoToMeeting – компанії через, які постачається продукт до кінцевого клієнта
ffiliated with Google), Microsoft Ventures, Salesforce Ventures and Workday Ventures - інвестори</td><td>Ключові види діяльності
- Надання основної зжатої інформації після та під час зустрічей на основі голосового штучного інтелекту
-Виявлення відношення осіб до зустрічей</td><td rowspan="2">Ціннісна пропозиція
- Зменшення часу на рутинну роботу

-Збільшення корисностей зустрічей, адже найважливіше фіксується та може бути переглянуто</td><td>Взаємовідносини з клієнтами
- Реклама в інтернеті та в продуктах партнерів
-Веб-сайт
-Поки продукт безкоштовний, поки він знаходиться в beta-версії, пізніше він стане платним</td><td rowspan="2">Споживчі сегменти
- Менеджери,

-офісні працівники, зокрема працівники IT-компаній</td></tr>
<tr><td>Ключові ресурси
Моделі розпізнавання голосу
Працівники
Сервери для розміщення акаунтів та збереження даних
Сервери для розміщення сервісу</td><td>Канали збуту
- Інтернет, компанії-партери, що забезпечують можливість проведення зустрічей</td></tr>
<tr><td colspan="3">Структура витрат
Розробка та вдосконалення моделей по розпізнаванню голосу
Оренда серверів та баз даних
Оплата електронернігії для розробки, підтримки та вдосконалення продукту
Найбільш затратними буде час розробників
Витрати на рекламу</td><td colspan="2">Потоки доходів
Продаж сервісу через підписку, яка в майбутньому очкується становити 10$ за місяць</td></tr>
</table>