ЗМІСТ

ПЕРЕЛІК ПОСИЛАНЬ

Гб – гігабайт.

ЕОМ – електронно-обчислювальна машина.

НМ – нейронна мережа.

СЛАР – система лінійних алгебраїчних рівнянь.

ФВЧ – фільтр високих частот.

ФНЧ – фільтр низьких частот.

AAM – Active Appearance Models, метод активних моделей зовнішнього виду.

CNN – convolutional neural networks, згорткова нейронна мережа.

CPU – central processing unit, центральний процесор.

DFD – data flow diagrams, діаграма потоків даних.

DoG – Difference of Gaussian, різниця гаусіанів.

FFNN – feed forward neural networks, нейронні мережі прямого поширення.

GHz – gigahertz, гігагерц.

GPU – graphics processing unit, графічний процесор.

HOG – Histogram of Oriented Gradients, гістограма направлених градієнтів.

IDLE – Integrated Development and Learning Environment, інтегроване середовище розробки.

ILSVRC – ImageNet Large Scale Visual Recognition Challenge – кампанія по широкомасштабному розпізнаванню образів в ImageNet.

SIFT – Scale-invariant feature transform, масштабонезалежне перетворення ознак.

SVM – Support Vector Machine, метод опорних векторів.

UML – Unified Modeling Language, уніфікована мова моделювання.

URL – Uniform Resource Locator, уніфікований вказівник ресурсу.

XML – eXtensible Markup Language, розширювана мова розмітки.

ВСТУП

Ідентифікація особистості людини стала цікавити людство достатньо давно. З появою ЕОМ і поступовим наростанням їх потужності стала можливою біометрична ідентифікація людини за допомогою разноманітних математичних методів і методів теорії обробки сигналів. Великим проривом стало застосування ідеї машинного навчання до обробки біометричних даних людини, в першу чергу не інвазивним шляхом, користуючись лише зовнішні ідентифікатори, такі як форма голови, зовнішній вигляд, патерни жестикуляції і зображення особи.

Завдання розпізнавання особи розділяється на два основних етапи: детектування особи на зображенні і безпосередньо його розпізнавання. Перший етап має на увазі стабільне виділення особи на зображенні з будь-яким фоном і в будь-якому положенні з низькою ймовірністю помилкового спрацьовування, а другий – звірення особи з базою й ідентифікація особи з низькою ймовірністю помилки. На кожному з цих етапів роботи алгоритмів перешкоджають різноманітні несприятливі фактори, такі як шум на зображенні, погане освітлення, низька роздільна здатність картинки, зміна положення голови відносно камери, мінливість особи людини і інші.

Машинне навчання дозволяє виконати завдання ідентифікації з допомогою методик, відмінних від звичних математичних. Побудовані за подобою мозкових тканин штучні нейронні мережі здатні виконувати завдання ідентифікації без розробки жорсткої математичної і алгоритмічної моделі, використовувані для класичних задач алгоритмізації. Важливу роль відіграє ідея навчання мереж, що дозволяє обробити інформацію за правилами, занадто абстрактним для математичних методів, зокрема завдання класифікації об'єктів.

Системи ідентифікації особи широко використовують в різних аспектах життя, наприклад, компанія Apple представила ідентифікатор обличчя на

iPhone X. Face ID має сенсор розпізнавання обличчя, який проектує інфрачервоні точки на обличчя і зчитує їх на шаблон, шаблон оброблюється для підтведження відповідності обличчя власника телефону. Подібні технології підтримуються Android, Xbox 360. Всі канадські міжнародні аеропорти використовують розпізнавання обличчя як частину програми первинної інспекції, яка порівнює обличчя мандрівника з фотографією, що зберігається в електронному паспорті. На мексиканських президентських виборах 2000 року мексиканський уряд використовував програмне забезпечення для розпізнавання осіб, щоб запобігти шахрайству виборців. Деякі особи реєструвалися для голосування за кількома різними іменами, намагаючись висловити кілька голосів. Порівнюючи нові зображення обличчя з тими, що вже є у базі даних виборців, органи влади змогли зменшити кількість дубльованих реєстрацій. Головною метою ідентифікації людини є забезпечення конфіденційності даних та безпеки суспільства.

Чому для ідентифікації особистості необхідно і доцільно використовувати саме нейронні мережі?

Саме прискорення швидкодії роботи систем ідентифікації обличчя засобами нейронної мережі та знаходження більш точних алгоритмів ідентифікації, порівняння особи є найбіль актуальною проблемою даної галузі.

Метою даної магістерської роботи є розробка системи ідентифікації обличчя засобами нейронної мережі, спераючись на певні ознаки її обличчя з використанням нейронних мереж. Для досягнення поставленої мети в роботі сформульовані такі завдання: проведення аналізу, дослідження та виявлення недоліків нейронних мереж та алгоритмів розпізнавання обличчя людини; здійснено аналіз значимих характеристик обличчя людини для її ідентифікації; проведено проектування нейронної мережі, здатної розпізнвати обличчя людини та ідентифікувати її за певними ознаками; виконана програмна реалізація системи.

Результати магістерської роботи можуть бути використані у системах безпеки, наприклад, проходження паспортного контролю, це дозволить міні-

мізувати помилку спроваковану людським фактором. Розроблювану систему ідентифікації обличчя засобами нейроної мережі можна використати у системах доступу до різноманітних об’єктів чи систем, наприклад: доступ працівника підприємства до певного приміщення чи офісу, документа або навіть доступ мешканця багатоквартирного будинку до підїзду. Розроблювана система дозволить виключити також несакційований доступ шляхом передачі пропуску.

Також дану систему можна реалізувати в системах оплати. Наприклад, створення кіоску для оплати послуг за допомогою обличчя. Побідна система була представлена на виставці в Барселоні. Для реєєстрації потрібно завантадити мобільний додаток, до якого є можливість прив’язати платіжну карту, електроний гаманець та сфотографуватися. При оплаті товару чи послуги шляхом застосування засобами системи ідентифікації обличчя за допомогою нейронної мережі можливо верифікувати операцію. Технологія довзволить здійснити покупку навіть якщо покупець за яких-небудь причин не може скористатися звичними методами розрахунку, а також знизить шанс використання картки третьою особою.

Загалом, безпека – базовий, але не єдиний сценарій використання системи розпізнання облич в інфраструктурі міст. Безпека виступає як базис і інфраструктура для багатьох сервісів, мова про деякі з них які йшла вище. Данні в форматі ідентифікації обличчя можна використати для розрахунків, оптимізації пасажиропотоку, пошуку зниклих людей, бар’єрному проході на заходи, офіси тощо.

1 ДОСЛІДЖЕННЯ І АНАЛІЗ ВИДІВ ТА АРХІТЕКТУР НЕЙРОННИХ МЕРЕЖ

В даному розділі здійснено досліження та аналіз основних видів архітектур штучних нейронних мереж, сфери їх застосування. Розглянуто основні архітектури штучних нейронних мереж для роботи із зображеннями, які найбіль придатні для ідентифікації обличчя засобами нейронної мережі.

1.1 Задача класифікації зображення

Одною з базових задач в машинному навчанні є задача класифікації зображення – визначення категорій об’єктів, котрий знаходиться на зображенні. В залежності від конкретної задачі, на зображенні може бути анотовано як один об’єкт так і декілька.

Для оцінки алгоритмів машинного навчання зазвичай використовують анотованні бази зображень, наприклад, CIFAR-10, ImageNet, PASCAL VOC.

Через те, що на зображеннях в базі ImageNet може бути присутнім декілька обєктів, і тільки один з них анотований, в ImageNet основною оцінкою похибки є top-5 похибка. При її використанні вважається, що алгоритм не помилився, якщо правильна категорія обєкта знаходиться серед пяти категорій, виданих алгоритмом як найбільш вірогідні. Внаслідок цього багато нейронних мереж для задачі класифікації оцінюються саме за допомогою top-5 помилки.

1.2 Аналіз нейронних мереж для роботи із зображеннями

З різноманіття типів нейронних мереж, спрямованих на рішення різних завдань, можна виділити кілька, основним завданням для яких є робота з образами і зображеннями. Далі будуть коротко розглянуті найбільш використовувані з них.

1.2.3 Нейронна мережа прямого поширення

Нейронні мережі прямого поширення (feed forward neural networks, FFNN) і перцептрони дуже прямолінійні, вони передають інформацію від входу до виходу (рисунок 1.1). Нейронні мережі часто описуються у вигляді листкового торта, де кожен шар складається з вхідних, прихованих або вихідних клітин.

Рисунок 1.1 – Нейронна мережа прямого поширення

Клітини одного шару не пов'язані між собою, а сусідні шари зазвичай повністю пов'язані. Найпростіша нейронна мережа має дві вхідних клітини і одну вихідну, і може використовуватися в якості моделі логічних вентилів. FFNN зазвичай навчається за методом зворотного поширення помилки, в якому мережа отримує безлічі вхідних і вихідних даних. Цей процес називається навчанням з учителем, і він відрізняється від навчання без учителя тим, що в другому випадку безліч вихідних даних мережу становить самостійно. Вищезазначена помилка є різницею між введенням і висновком. Якщо у мережі є достатня кількість прихованих нейронів, вона теоретично здатна змоделювати взаємодію між вхідним і вихідними даними. Практично такі мережі використовуються рідко, але їх часто комбінують з іншими типами для отримання нових [1][1)].

[1)][1] Нейронні мережі. URL: https://www.asimovinstitute.org/neural-network-zoo (Дата звернення 2.10.2019).

1.2.2 Нейронна мережа Хопфілда

Нейронна мережа Хопфілда (Hopfield network) – це полносвязная нейронна мережа із симетричною матрицею зв'язків. Під час отримання вхідних даних кожен вузол є входом, в процесі навчання він стає прихованим, а потім стає виходом (рисунок 1.2).

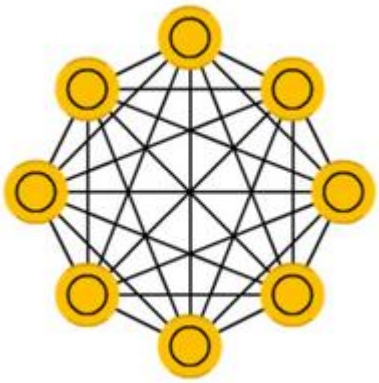


Рисунок 1.2 – Нейронна мережа Хопфілда

Мережа навчається так: значення нейронів встановлюються відповідно до бажаного шаблоном, після чого обчислюються ваги, які в подальшому не змінюються. Після того, як мережа навчилася на одному або декількох шаблонах, вона завжди буде зводитися до одного з них (але не завжди – до бажаного). Вона стабілізується в залежності від загальної "енергії" і "температури" мережі. У кожного нейрона є свій поріг активації, що залежить від температури, при проходженні якого нейрон приймає одне з двох значень (зазвичай -1 або 1, іноді 0 або 1). Така мережа часто називається мережею з асоціативної пам'яттю; як людина, бачачи половину таблиці, може представити другу половину таблиці, так і ця мережа, отримуючи таблицю, наполовину зашумленную, відновлює її до повної.

1.2.3 Згорткова нейронна мережа

Згорткова НМ (convolutional neural networks, CNN) – використовується для класифікації та пошуку образів предметів на зображенні (рис.1.3).

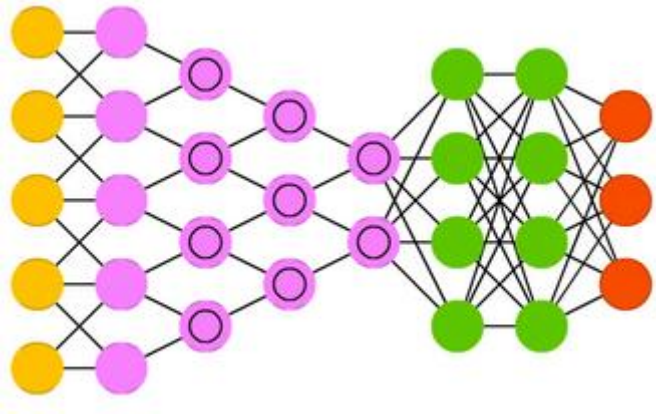


Рисунок 1.3 – Згорткова нейронна мережа

Глибинні згорткові зворотні графічні мережі – мережі, основним завданням яких є генерація зображень з декількома класифікаторами на основі тренувальної вибірки зображень з кожним з классіфікаторів – окремо. Може працювати і в зворотному напрямку.

Глибинна згортоква нейронна мережа – вдає із себе згорткову нейронну мережу на виході якої перебуває мережа прямого поширення для подальшої обробки даних. Використовується для пошуку способу і подальшої його класифікації [1][1)].

1.2.4 Згорткова нейронна мережа AlexNet

В 2012 році на конкурсі по класифікації зображень вперше перемогла нейронна мережа AlexNet, досягнувши top-5 похибки 15,31%. Для порівняння, метод, не використовуючий згорткові нейронні мережі, отримав похибку

[1)] [1] Нейронні мережі. URL: https://www.asimovinstitute.org/neural-network-zoo (Дата звернення 2.10.2019).

26,1%. В AlexNet були зібрані найновіші на той момент техніки для поліпшення роботи мережі, приклад архітектури даної мережі показано на рисунку 1.4.

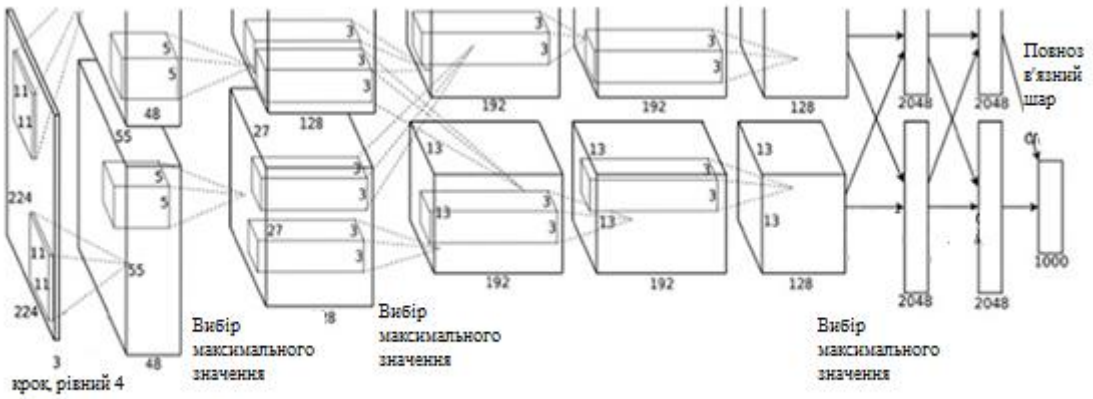


Рисунок 1.4 – Архітектура мережі AlexNet

Навчання AlexNet через кількість параметрів мережі було організовано на двух GPU, що дозволило скоротити нач навчання, порівняно з навчанням на CPU. Також було виявлено, що використання функції активації замість функції сигмоїди і гіперболічного тангенса дозволило знизити кількість епох навчання в шість раз. Функція активації дозволяє побороти проблему затухання градієнтів, яка є ознакою інших функцій активацій.

Також в AlexNet була отримана техніка відсіву, заключалася у випадковому відключенні на заданому с деякою вірогідністю на кожній епосі. Після навчання слоїв, на стадії розпізнавання, ваги слоїв, до яких була застосована техніка відсіву, мають бути помножені на одиницю поділену на кількість епох. Для пояснення ефективності даної техніки існує декілька інтерпретацій. Перша в тому, що техніка відсіву заставляє нейрони не розраховувати на сусідні нейрони, а навчатися розпізнавати більш стійкі ознаки. Зміст другої в тому, що навчання мережі за технікою відсіву представляє собою апроксимацію навчання ансамблю мереж, кожна з яких представляє собою мережу без деяких нейронів. Таким чином кінцевий результат приймає не

одна мережа, а ансамбль, кожна мережа якого навчена по-різному, що знижує вірогідність похибки [2][1)].

1.2.5 Згорткова мережа ZFNet

Основним досягненням даної архітектури є створення техніки візуалізації фільтрів – мережі розгортки, яка складена з операцій, зворотній операціям мережі. Архітектуру даної нейронної мережі продемонстровано на рисунку 1.5. Щоб розглянути поведінку фільтра на певному зображенні за допомогою навчання нейронної мережі, необхідно спочатку здійснити вивід мережею, після чого в слою розглядуваного фільтру обнулити всі ваги, крім вагів самого фільтру, а потім подати отриману активацію на слой мережі розгортки.

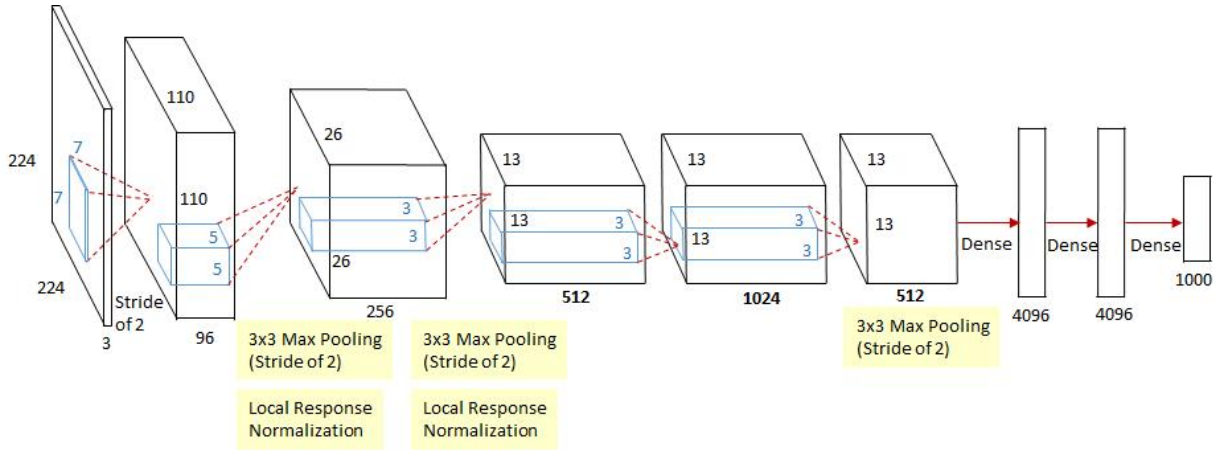


Рисунок 1.5 – Архітектура мережі ZFNet

В мережі розгортки поступово примінюються операції з даними та фільтрації. Дані операції частково поновлюють вхід відповідного слою субдискретизації. Слой фільтрації виконує операцію згортки с вагами відповідного слою згортки, але ваги кожного фільтру «перевернуті» вертикально та гори-

[1)] [2] Сикорский О.С. Обзор сверточных нейронных сетей для задачи классификации изображений. М: МГТУ им. Н.Э. Баумана, 2018. С. 3-8.

зонтально. Таким чином, виходячи з функції активація фільтра рухається в зворотньому напрямку, поки не буде відображена в оригінальному просторі зображення [3][1].

1.2.6 Згорткова мережа VGG Net

У 2014 році К. Симонян (K. Simonyan) і Е. Ціссерман (A. Zisserman) з Оксфордського університету запропонували архітектуру, яка називається VGG. Основнj. і відмінною ідеєю цієї структури є збереження фільтрів настільки простими, наскільки це можливо. Архітектура даної мережі приведена на рисунку 1.6.

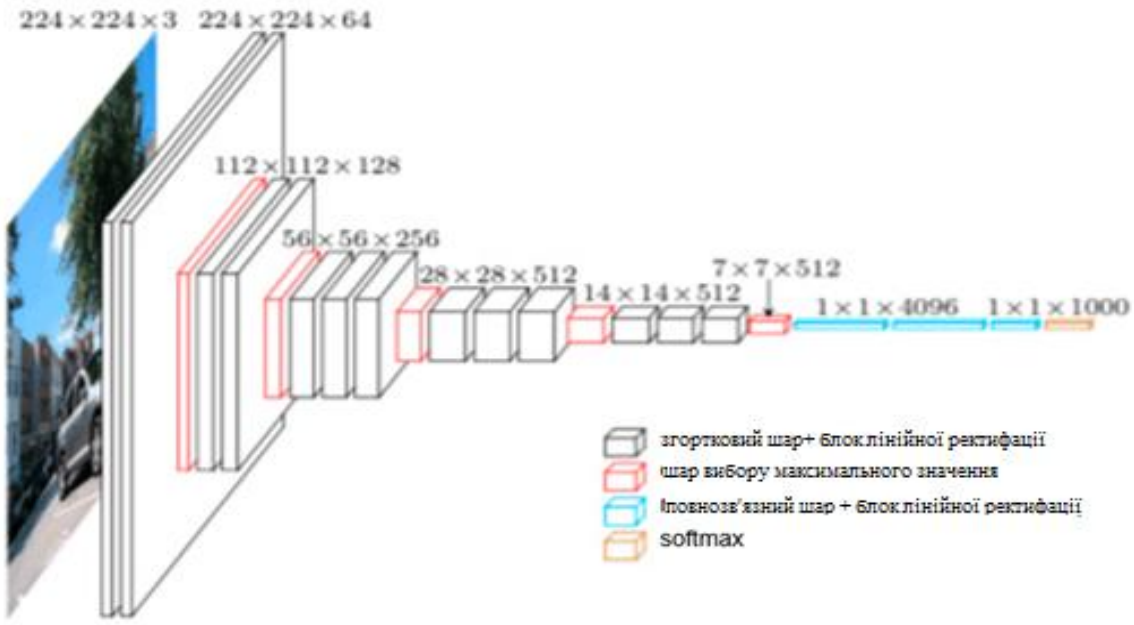


Рисунок 1.6 – Архітектура мережі VGG Net

Тому всі операції згортки виконуються за допомогою фільтра розміром 3 і кроку величиною 1, а всі операції субдискретизації – за допомогою фільтра розміром 2 і кроку величиною 2. Однак це не все. Одночасно з простотою

[1] [3] Структура нейронної мережі ZFNet. URL: https://www.oreilly.com/library/view/hands-on-transfer-learning/9781788831307/ff290260-c4fc-4924-9f30-bcddb36cc2 a.xhtml (Дата звернення: 9.10.2019).

згорткових модулів мережу значно зросла в глибину – тепер вона має 19 шарів. Найважливіша ідея, вперше запропонована в цій роботі, полягає в накладенні згорткових шарів без шарів субдискретизації. Суть даної ідеї полягає в тому, що таке накладення як і раніше забезпечує досить велике рецептивне поле (наприклад, три накладених один на одного згорткових шари розміром 3 × 3 з кроком 1 мають рецептивне поле аналогічне одному згортковому шару розміром 7 × 7), проте кількість параметрів при цьому значно менше, ніж в мережах з великими фільтрами (служить в якості регуляризатора). Приклад такої декомпозиції наведено на рисунку 1.7. Крім того, з'являється можливість внесення додаткових нелінійних перетворень.

По суті, автори продемонстрували, що навіть за допомогою дуже простих стандартних блоків можна домогтися чудової якості результатів в конкурсі ImageNet. Число помилок для п'яти провідних категорій скоротилося до 7,3 відсотка.

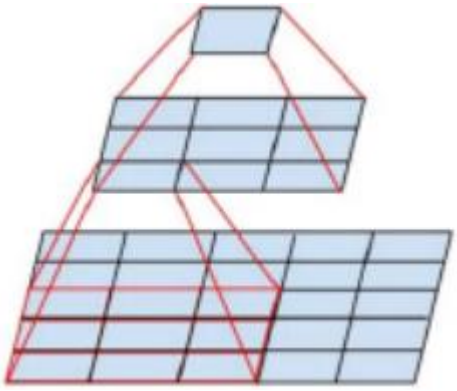


Рисунок 1.7 – Декомпозиція фільтру 5x5

На змаганні ILSVRC 2014 ансамбль з двох VGG Net отримав top-5 похибку 7,3%. Хоча дана модель не виграла в змаганні, через її простоту вона використовується в більш складних мережах, призначених для детектування предметів, сематичної сегментації чи маскування обєктів [4][1)].

[1)] [4] Matthew Zeiler, Rob Fergus. Visualizing and understanding convolutional networks. [s. l.] 2016. P. 818-834.

1.2.7 Згорткова мережа Inception

Inception-v1 – переможець ILSVRC 2014 з top-5 похибкою 6,7%, що на 1,4% краще ніж отримала мережа VGG Net в тому ж році. Дана мережа також відома як GoogLeNet. Творці даної мережі виходили з факту, що після кожного слою мережі необхідно зробити вибір – буде наступний слой сгорткою з фільтром 3x3, 5x5, 1x1 чи слоєм субдескретезації. На рисунку 1.8 приведена архітекура системи, зеленим на рисунку виділено модуль Inception, побудова мережі виконується на даних модулях [5][2)].

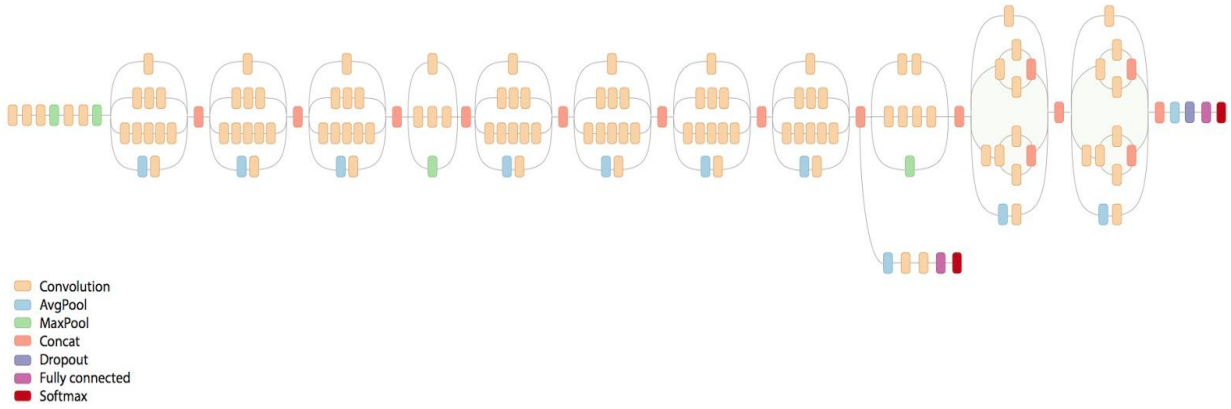


Рисунок 1.8 – Архітектура мережі GoogLeNet

Кожен за таких шарів корисний – фільтр 1x1 виявляє кореляцію між каналами, в той час як фільтри більшого розміру реагують на більш глобальні ознаки, а слой субдескретизації дозволяє зменшити розмірність без великих втрат інформації. Замість того, щоб виділити який саме слой буде наступний, пропонується використовувати всі слої одразу, паралельно один одному, а потім обєднати отримані результати в один. Щоб виключити ріст числа параметрів, перед кожним слоєм згортки використовується згортка 1x1, котра зменшує кількість карт ознак. Даний блок показаний на рисунку 1.9.

[2)] [5] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio. Learning to segment object candi-dates. [s. l]. 2015. P. 1990-1998.

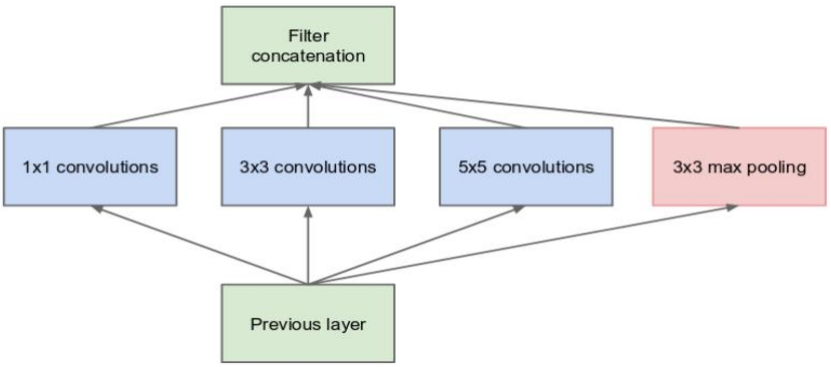


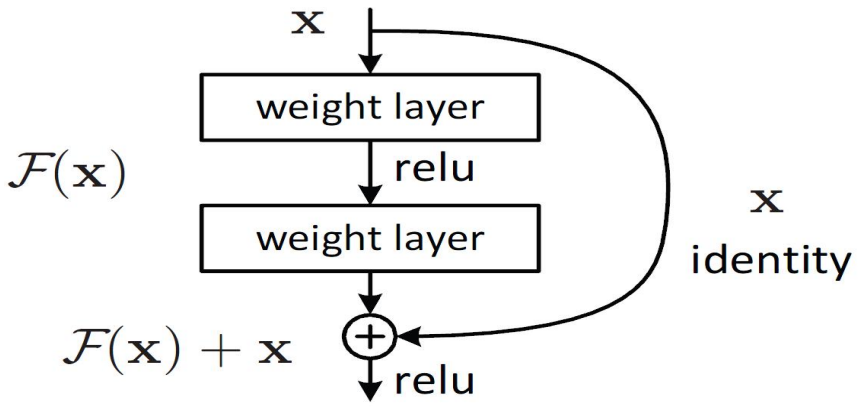
Рисунок 1.9 – Модуль Inception

Також в GoogLeNet відмовились від використання повнозвязного слою в кінці мережі, використовуючи замість нього слой середнього пулу, завдяки чому зменшилась кількість параметрів в мережі. Таким чином GoogLeNet, яка складається більш ніж зі ста базових слоїв має майже в 12 разів менше параметрів ніж AlexNet (близько 7 мільйонів параметрів проти 138 мільйонів).В наступній ітерації модуля, названим Inception-v2, автори декомпозували слой з фільтром 5x5 на два слої 3x3. Далі була використана техніка нормалізації, яка дозволяє багаторазово збільшити швидкість навчання за рахунок нормалізації розподілення виходів всередені мережі.Також автори даної мережі запропонували Inception-v3. В даній моделі вони розвили ідею декомпозиції фільтрів, запропонував декомпозувати фільтр NxN двума послідовними фільтрами 1xN та Nx1. Також в даній мережі використовують алгоритм RMSprop замість стандартного градієнтного спуску і використовують усічення градієнтів для підвищення стабільності навчання. Ансамбль з чотирьох Inception-v3 отримав top-5 похибку 3,58 на IlSVRC 2015, поступившись мережі ResNet [6][1)].

1.2.8 Згорткова мережа ResNet

[1)] [6] Нейронні системі типу Inception. URL: https://towardsdatascience.com/a-simple-g uide-to-the-versions-of-the-inception-network-7fc52b863202 (Дата звернення: 20.10.2019).

Переможцем ILSVRC 2015 з top-5 похибкою 3,57% став ансамбль з шести мереж типу ResNet (Residual Network), розроблений Microsoft Research. Автори ResNet помітили, що з зростанням числа слоїв згорткова нейронна мережа може почати деградувати – в неї знижується точність на валідаційній множині. Так як падає точність і на тренувальній множині, можна зробити висновок, що проблема складається не в навчанні мережі. Було зроблене припущення, що якщо згорткова мережа досягла свого максимуму точності на деякому слої, то всі наступні слої повинні будуть перетворитися в тотожні перетворення, але через складність навчання нейронних мереж це не відбувається [2][2)]. Для того, щоб вирішити дану проблему було прийнято ввести пропускні з'єднання, які приведені на рисунку 1.10.



Риснуок 1.10 – Пропускаюче з'єднання в мережі ResNet

Нехай оригінальна мережа повинна вираховувати функцію H(x). Її залишкова функція визначена як:

$$F(x) = H(x) - x, \tag{1.1}$$

[2)] [2] Сикорский О.С. Обзор сверточных нейронных сетей для задачи классификации изображений. М: МГТУ им. Н.Э. Баумана. 2018 С. 3-8.

Дана функція , в теорії, повинна бути простіше вивчена мережею. Додаючи пропускні з’єднання, як показано на рисунку 1.8, мережа навчається залишковій функції с тотожнім перетворенням.

Аналіз показав, що глубинні залишкові нейронні мережі можна разувати ансамблем, який складається з більш менших залишкових нейронних мереж, чия ефективна глибина збульшується в процесі навчання.

1.2.9 Inception-v4 та ResNet

В даному випадку модуль Inception розбитий на модулі A, B, та C, для входів розмірністю 35x35, 17x17 та 8x8 відповідно. Також були виділені блоки редукції, в котрих відбувається зниження розмірності і збільшення глибини даних всередині мережі. В Inception-v4 головним нововведенням є заміна максимального об’єднання на середнє в модулях. Для Inception-ResNet в модулі Inception були додані пропускні з’єднання. Були сконструйовані дві версії мережі Inception-ResNet-v1, для якої потрібно було менше обчислень та Inception-ResNet-v2 [6][1)]. В якості прикладу на рисунку 1.9 представлено модуль Inception-ResNet-C для Inception-ResNet-v1.

[1)] [6] Нейронні системі типу Inception. URL: https://towardsdatascience.com/a-simple-g uide-to-the-versions-of-the-inception-network-7fc52b863202 (Дата звернення: 20.10.2019).

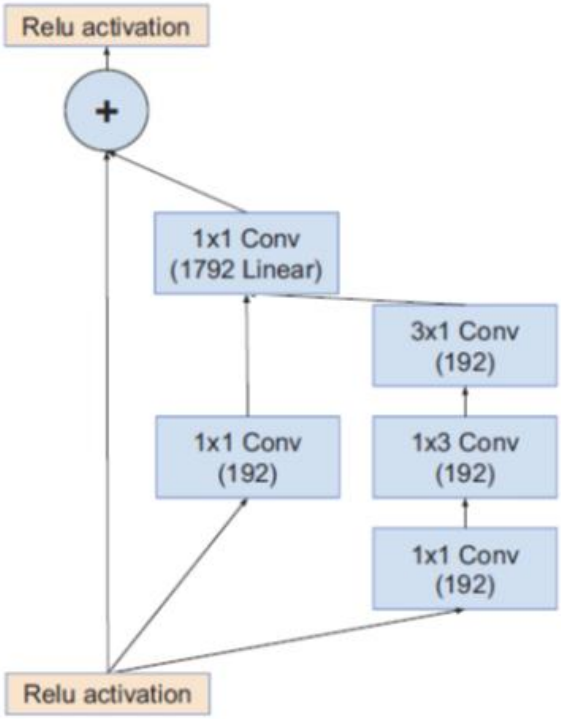


Рисунок 1.11 – Модуль Inception-ResNet-C для Inception-ResNet-v1

1.3 Порівняння моделей

Для оцінки моделей згорткових нейронних мереж окрім виду помилки зазвичай вказують кількість моделей в ансамблі і кількість фрагментів зображення, які подавались на вхід моделі. Наприклад 10 фрагментів означає, зо зроблено чотири фрагменти по кутам зображення, один в центрі і кожний окремий виріз горизонтально перевернутий [7][1)]. В табл. 1.1 представлені результати розглянутих нейронних мереж з однією моделлю та з одним вирізом на базі зображені ImageNet (окрім ResNet-151, для якої вказано результат для 10 фрагментів).

Таблиця 1.1 – Результати для однієї моделі з одним вирізом

[1)] [7] Порівняльна характеристика нейронних мереж. URL: https://cyberleninka.ru/article/n/obzor-svyortochnyh-neyronnyh-setey-dlya-zadachi-klassifikatsii-izobrazheniy (Дата звернення 27.10.2019).

Нейронна мережа	Top-1 %	Top-5 %
AlexNet	39.00	17
ZF Net	37,50	16
VGG NET	25,60	8,10
GoogLeNet	29,00	9,20
Inception-v3	21,20	5,60
Inception-v4	20,00	5
Inception-ResNet-v2	19,90	4,90
ResNet-151	19,38	4,49

В табл. 1.2 наведено результати використання асамблей моделей з багатьма фрагментами на базі зображень ImageNet.

Таблиця 1.2 – Результати для асамблей з багатьма фрагментами.

Нейронна мережа	Моделі	Фрагменти	Top-1 (%)	Top-5 (%)
AlexNet	7	1	36,70	15,31
ZF Net	6	10	36	14,70
VGG NET	2	150	23,70	6,80
GoogLeNet	7	44	-	6,67
Inception-v3	4	144	17,20	3,58
Resnet-151	6	144	-	3,57
Inception-v4+3x Inception-ResNet	4	144	16,50	3,10

Як видно за даних таблиць, за останні роки Top-5 похибка на ImageNet для одиночних моделей зменшилась в чотири рази (з 17% до 4,49%), а для ансамблеї – майже в п’ять разів (з 15,40% до 3,10%). Також з даних таблиць видно, що найменшу похибку має згорткова нейронна мережа ResNet.

Кожна з наведених мережевих архітектур має свої унікальні підходи до різних проблем. Наприклад у AlexNet є дві паралельні лінії CNN, навчені на двух графічних процесорах з крос-з'єднаннями, GoogLeNet має початкові модулі, ResNet – остаточні з'єднання. Особливістю мережі VGG Net є однорідна архітектура, виконує згортування 3x3 і пулінг 2x2 від початку до кінця. Мережа ZF Net є вдосконаленою архітектурою AlexNet, в ній збільшили розміри середній згорткових шарів та зменшили крок і розмір фільтрів в першому шарі.

Дані мережі можуть бути використані для різних цілей при роботі з зображеннями, та мають різні підходи до вирішення певних проблем при навчанні. Деякі з приведених згорткових нейронних мереж працюють з великими обсягами даних, деякі – з малими, також мають різні показники при роботі в ансамблях нейронних мереж.

В ході проведення аналізу та вивчення основних видів архітектур штучних нейронних мереж найбіль придатними для вирішення задачі роботи із зображеннями було визначено штучні нейронні мережі згорткового типу. Даний тип мереж використовується для роботи з зображеннями, пошуку класифікаторів на ньому, та відтворенню зображень по класифікаторам.

Було проаналізовано основні види архітектур та алгоритмів згортоквих нейронних мереж, для вирішення задачі ідентифікації обличчя засобами нейронної мережі доцільно використовувати нейронну мережу типу ResNet. Згідно дослідження дана мережа має найменший відсоток похибки за шкалою Top-5 та Top-1, має досить велику швидкодію.

Мережа ResNet є найбільш глибокою мережею навченою на ImageNet. Також дана система має більш низький параметр ніж у VGG Net, котрий у 8 разів менше за глибиною, що впливає на швидкість часу навчання на користь ResNet.

2 СПЕЦИФІКАЦІЯ ВМИМОГ ДО СИСТЕМИ

В даному розділі висуваються головні вимоги до системи ідентифікації обличчя засобами нейронної мережі. Будуть розглянуті основні види навчання нейронної мережі, більш детально розглянута архітектура згорткової нейронної мережі, розглянуті на проаналізовані алгоритми роботи із розпізнавання облич на зображенні.

Розроблювана система повинна відповідати наступним вимогам:

- Висока точність роботи алгоритму для виявлення облич;
- Висока швидскість пошуку обличчя, необхідних параметрів та розрахунків;
- Система повинна мати високу точність у виводі кінцевих результатів
- Кросплатформеність системи;
- Широкий спектр можливостей для подальшої модернізації.

2.1 Навчання нейронних мереж

Існує навчання мереж з учителем (supervised), без вчителя (unsupervised), а також навчання з підкріпленням (reinforcement learning). Перший тип використовується набагато частіше і має на увазі наявність тренувальної вибірки даних, достовірно правильні результати обробки яких вже відомі. Другий тип використовується для самонавчання, наприклад, алгоритми групування даних за певними параметрами. При цьому групи не задаються ззовні, а є результатами роботи самого алгоритму. Третій тип цікавий тим, що мережі надається право самій знайти спосіб рішення, а вчитель лише заохочує правильні відповіді.

Після ініціалізації нейронної мережі, її необхідно навчити. Для навчання використовуються набори даних, званих тренувальними сетами. Після проходження кожного тренувального сету на одиницю збільшується лічиль-

ник ітерацій навчання. Після проходження певної кількості ітерацій (число яких, як правило, пов'язано з повнотою тренувальної множини навчання, однак може визначатися і іншими параметрами) закінчується епоха. Епох може бути кілька, тобто багаторазове навчання мережі на одній і тій самій множині, збільшують точність роботи мережі [8][1)].

Так як значення кожного нейрона є лінійною комбінацією значення попередніх, то навчання такої мережі можна звести до математичної задачі багатопараметричної лінійної оптимізації. Параметром оцінки оптимізації є помилка нейронної мережі при свідомо вірному відповіді. Завдання оптимізації в даному випадку буде мати вигляд :

$$r(w) = \sum_i C(a(x_i), y_i) \to \min_w, \tag{2.1}$$

Де C – функція втрат, яка визначає величину помилки при свідомо відомому результаті.

Одним з методів поновлення ваг при навчанні є метод градієнтного спуску. На кожному кроці, величиною ɳ ваги будуть змінюватися як :

(2.2)

При дифференціюємих функціях ризику r(w) і активації f можна підставити значення формули ризику і отримати:

$$w^{(k+1)} = w^{(k)} - n \sum_i C_a'(a(x_i), y_i) f'(\langle w, x_i \rangle) x_i, \tag{2.3}$$

[1)] [8] LeCun Y. et al. Gradient-based learning applied to document recognition //Proceedings of the IEEE.1998.T. 86. P. 2278–2324.

Але в цьому випадку ваги змінюються лише після обробки всього пакету ітерацій (batch). Якщо необхідна зміна ваг після проходження кожного тренувального сету, можна використовувати так званий стохастичний градієнтний спуск :

$$w^{(k+1)} = w^{(k)} - n\, C'_a(a(x_i), y_i) f'((w, x_i))x_i, \quad (2.4)$$

У кожного з цих методів присутні свої переваги і недоліки. Простий пакетний спуск передбачає роботу одночасно з усім пакетом даних, що при великому розмірі пакета вимагає наявності більшого обсягу пам'яті для його зберігання, проте на малих обсягах даних працює помітно швидше завдяки можливості використання матричного обчислення результату. Стохастичний метод, більш простий в обчисленні, виграє на великих обсягах даних. Крім того, до його переваг можна віднести можливість додавання нових даних в набір прямо під час навчання, що дозволяє більш гнучко проводити навчання.

Обєднання переваг і нівелює недоліки обох методів комбінований міні – пакетний (mini–batch) метод, який використовується в більшості випадків. З навчального набору даних на кожній ітерації алгоритму випадковим чином формується невеликий міні – пакет, що розраховується методом пакетного градієнтного спуску. Виходячи з вимог до пакетного і стохастическому методу можна ви – рушити загальні вимоги до функції втрат – вона повинна бути подана в вигляді :

(2.5)

Де С – функція втрат для всього набору даних;

Cx – функція втрат для всього набору даних.

Функція втрат вибирається евристично, виходячи з вимог конкрентної нейронної мережі і завдань, що вирішуються нею, однак, найбільш часто використуються дві з них.

Квадратична функція з параметрами a(xi) – реальне значення виходу нейронної мережі, yi – свідомо вірне значення під час i-ої ітерації, відповідає формулі:

(2.6)

Квадратична функція через простоту обчислення на практиці зустрічається набагато частіше.

Функція перехресної ентропії трохи складніше :

(2.7)

2.2 Метод зворотнього розповсюдження помилки

Перед тим як використовувати методи градієнтного спуску в мережах з декількома шарами, потрібно реалізувати метод для розрахунку функції втрат для всієї мережі, а також подальшого коректування значень ваг мережі в умовах завдання мінімізації. Для виконання цих завдань використовують метод зворотного поширення помилки [9][1)].

[1)] [9] Алгоритми зворотнього розповсюдження помилки. URL: www.irbis-nbuv.gov.ua /irbis_nbuv/cgiirbis_64 (Дата звернення: 30.10.2019)

Нехай є кінцевий набір тренувальних даних (m прикладів). Функція активації xi елемента a(xi) залежить від параметрів W і b. Тоді для m прикладів функція втрат набуде вигляду:

$$\mathbf{c}(w,b) = \left(\frac{1}{\mathbf{m}}\sum_{\mathbf{i=1}}^{\mathbf{m}}(\mathbf{kb_1x \cdot 1y_i})\right) + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{S_l+1}\left(w_{ij}^{l}\right)^2 = = \left(\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\right.\right. \quad (2.8)$$

Де перша частина – це сумма квадратів похибок, тоді як друга – член регуляризації зменшення ваг, який запобігає надлишковому перенавчанню нейронної мережі. В ньому параметр λ відповідає за оцінку значності тих частин даного прикладу відносно один одного.

Виконується прямий підхід до мережі і обчислення активацій шарів до вихідного шару Ln.

Для кожного i–того вихідного нейрону на шарі Ln вираховується вихідна похибка:

$$\delta_{\mathbf{i}}'\mathbf{n} = \frac{\boldsymbol{\partial}\left(\frac{1}{2}\left\|a_i(\mathbf{x}) - \mathbf{y_i}\right\|^2\right)}{\boldsymbol{\partial}\mathbf{z_i^{Lm}}} = -\left(y_i - a_i^{Ln}\right)f'\left(z_i^{Ln}\right), \quad (2.9)$$

Пошарово для кожного k–ого шару, починаючи з Ln і закінчуючи L2 для кожного елементу вираховується похибка:

$$\delta_i^{Lk} == \left(\sum_{j=1}^{Sl+1} W_{ji}^{Lk}\delta_j^{Lk+1}\right)f'\left(z_i^{Lk}\right), \quad (2.10)$$

Для метода градієнтного спуску вираховуються похідні:

$$\frac{dC_{Wb}(x,y)}{dW_i^{Lk}} = a_j^{Lk}\delta_i^{Lk+1}, \quad (2.11)$$

$$\frac{dC_{Wb}(x,y)}{dW_i^{Lk}} = \delta_i^{Lk+1}, \tag{2.12}$$

Для використання переваг матричних алгоритмів в швидкості роботи необхідно представити послідовність кроків в матричному вигляді:

– Виконується прямий підхід по мережі і розрахування активацій шарів до вихідного шару Ln.
– Встановлюється матриця помилок вихідного шару Ln.
– Пошарово для кожного k–ого шару починаючи з Ln і закінчуючи L2 для кожного елементу шару вираховується похибка.

$$\delta^{Lk} == \left(W^{Lk}\delta^{Lk+1}\right) * f'\left(z^{Lk}\right), \tag{2.13}$$

– Для методу градієнтного спуску обчислюються приватні похідні:

$$\nabla w C_{wb}(x,y) = \left(a^{Lk}\right)\delta^{Lk+1}, \tag{2.14}$$

$$\nabla w C_{wb}(x,y) = \delta^{Lk+1}, \tag{2.15}$$

∇wCWb(x, y) – це градієнт, тобто вектор, який складається із приватних похідних $\frac{\partial \mathbf{C}}{\partial \mathbf{w}}$ або $\frac{\partial \mathbf{C}}{\partial \mathbf{b}}$ відповідно.

Таким чином, скомбінувавши отриманий матричний алгоритм зворотнього поширення помилки зі змішаним методом градієнтного спуску (міні – пакетним) можна побудувати алгоритм навчання нейронної мережі.

Для кожного з вхідних об'єктів мережі розрахувати активацію і виконати алгоритм зворотнього поширення помилки (пряме поширення – розрахунок помилки – зворотне поширення).

Виконати градієнтний спуск, для кожного шару 1 = L, L –1, …, 2, тобто перебалансувати ваги по правилу:

$$w^{(k+1)} = w^{(k)} - \frac{n}{m}\sum \delta^{(x,k)} \left(a^{(x,k-1)}\right)^T, \quad (2.16)$$

А також зміщення по формулі:

(2.17)

,

Основним параметром оцінки правильності налаштування і роботи нейронної мережі є збіжність – ітеративне спадання цільової функції під час процесу навчання. Приклади графіків: а) з наявністю збіжності, б) з коливаннями на графіці цільової функції, в) відсутністю східності представлені на рисунку 2.1.

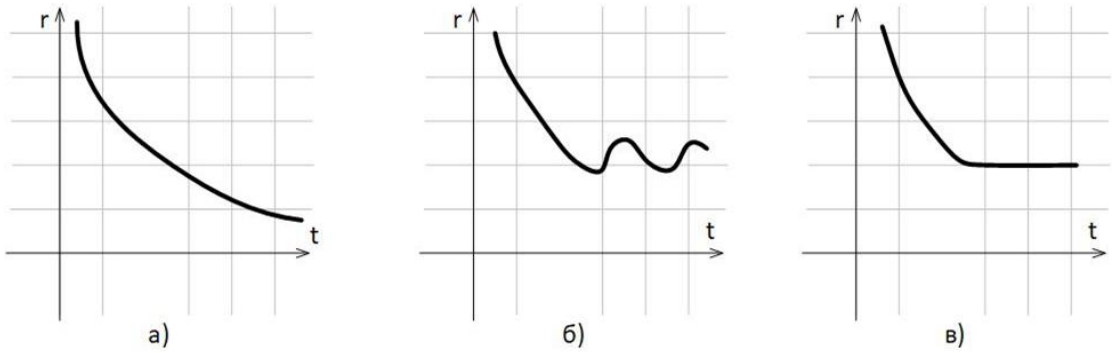


Рисунок 2.1 – Графіки з: а) з наявністю збіжності, б) з коливаннями на графіці цільової функції, в) відсутністю збіжності

Крім очевидних переваг, метод поширення зворотного помилки має і ряд недоліків: певні функції активації, у яких є горизонтальні асимптоти можуть ввести мережу в стан « паралічу ». Якщо на під час корекції ваги мережі стають великими величинами, то їх похідна стискаючої функції наближається до нуля приводячи до зменшення кроку градієнтного спуску і ваги нейрона. При настанні «паралічу» нейрон не здатний вийти з цього стану самостійно, проте, паралічу можна не допустити зменшуючи крок, але, відповідно, збільшуючи і час навчання.

Якщо в мережі існує велика кількість ваг, то мережа схильна до перенавчання.

Таким чином варто з обережністю підходити до завдання для зі – зберігання балансу між швидкістю навчання та недопущенням входження нейронів в стан паралічу.

Щоб уникнути перенавчання мережі використовують ранню зупинку (early stopping). Ідея ранньої зупинки полягає в перевірці точності класифікації після проходження кожної з епох. Як тільки поліпшення точності припиняється, необхідно зупинити навчання – це значить, що відбулося насичення, і продовження процесу призведе до перенавчання.

Для прискорення навчання використовують змінний темп навчання. У тому випадку темп навчання залишається незмінним до моменту погіршення точності даних перевірки. Потім темп навчання зменшується (як правило в 5 або 10 разів) і навчання триває, ця дія повторюється кілька разів.

2.3 Гіперпараметри нейронної мережі

Управління роботою нейронної мережі здійснюється зміною ряду параметрів, розглянутих раніше. Як правило налаштування проводиться евріс-

тично, керуючись знаннями про призначення параметрів і технічного завдання.

Розмір пакету. Занадто великий пакет приведе до рідкого оновлення ваг і не дозволить оперативно додавати до навчальної множини нові об’єкти. Занадто маленькй пакет не дозволить бібліотекам матричної обробки розкрити весь свій потенціал швидкості. Цей параметр підбирається як еврістично. Так і за допомогою алгоритмів оптимізації.

Кількість скритих шарів. Як правило параметр підбираєься еврістичто і варюється від 1 до 5.

Швидкість навчання регулює розмір кроку в градієнтному спуску, вибирається порогове значення, при якому швидкість максимальна, але не спостерігається коливань значень цільової функції. Як правило при налаштуванні спочатку встановлюють $\eta = 0{,}01$, а потім збільшують його до настання коливань, або зменшують, якщо крок занадто великий.

Параметр регуляції. Визначає внесок кожного тестового сету в навчання мережі. Чим параметр більше, тим сильніше обробляємий тестовий об'єкт змінює мережу. Занадто велике значення цього параметра призводить до занадто сильного навчання мережі з кожної итерації; занадто мале значення збільшує кількість епох, необхідних для навчання мережі. Найчастіше цей параметр приймають рівним 1 і збільшують в 10 разів на кожному кроці до тих пір, поки збіжність цільової функції не буде порушена

2.4 Згорткові нейронні мережі

В цьому розділі детально розглянуті згорткові нейронні мережі на їх гіперпараметри.

Вимоги до архітектури згорткової нейронної мережі продиктовані її спеціалізацією. Обробка великих зображень передбачає наявність великої кількості ваг. Наприклад, для невеликого зображення 100 х 100 точок з трьома каналами кольору потрібно 100 * 100 * 3 = 30 000 ваг на прихованому рівні.

Для обробки великих зображень потрібно в рази більша кількість ваг, відповідно і навчання таких мереж буде невигідним.

Спрощена архітектура згорткової мережі показана на рисунку 2.2. Зліва знаходиться шар вхідних нейронів, праворуч – вихідних. Так само в правій частині малюнка знаходиться мережа прямого поширення. Безпосередньо згортка відбувається на вузькому каскаді нейронів в правій частині графіка [1][1)].

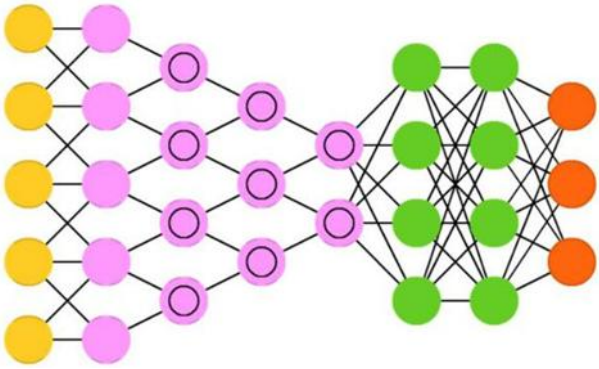


Рисунок 2.2 – Архітекткура згорткової нейронної мережі

Кількість нейронів у кожному наступному шарі зменшується, як правило для цього використовуються ступеня числа 2:..., 64, 32,..., 4, 2. При цьому зчитування зображення відбувається з використанням операції згортки. Чи невелика матриця, яку називають згортковою, проходить по зображенню, як правило, починаючи з верхнього лівого кута і зрушуючи на одну точку за крок. Так, матриця ваг залишається незмінною для різних нейронів, а результат застосування матриці до всього полю піддається лінійній ректифікації (rectified linear unit, ReLU), мета якої полягає в відсіканні непотрібних деталей в каналі

[1)] [1] Нейронні мережі. URL: https://www.asimovinstitute.org/neural-network-zoo (Дата звернення 2.10.2019).

Згорткові шари чергуються з шарами субдискретизації [8][1)], як це показано на рисунку 2.3. На шарі субдискретизації зменшується розмірність карт ознак, що подаються на вхід. Вибирається максимальне значення з кількох сусідніх значень нейронів карти ознак та цей нейрон стає новим нейроном створюваної карти.

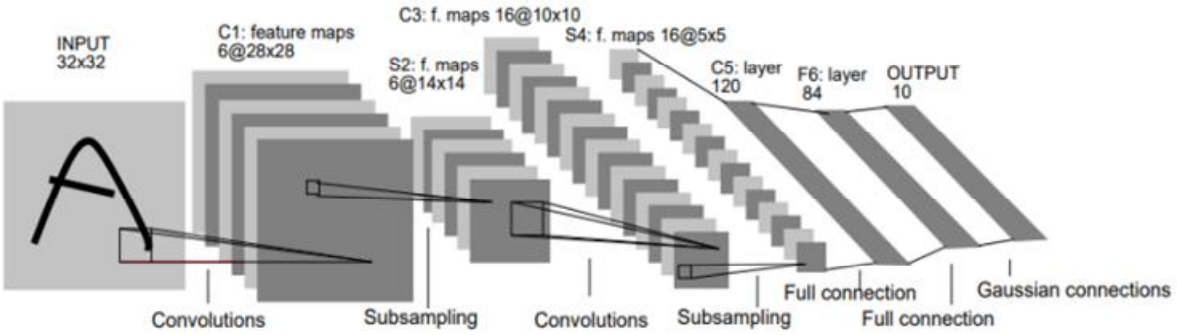


Рисунок 2.3 – Приклад архітектури НМ
для розпізнавання зображень

З проходження шарів згортки – субдискретизації розмірність зображення зменшується і в підсумку виході мережі вдає із себе мале число параметрів, які визначають самі абстрактні поняття мережі. Вони подаються на вхід наступної мережі прямого поширення, що виконує оброблення результату. Далі відбувається навчання згорткової мережі, наприклад, способом зворотнього поширення помилки [9].

Повторна перевірка на кількість прихованих шарів показує, що при тому ж розмірі вихідного зображення 100 х 100, 3 каналів кольору, а також розмірності ядра згортки 3 х 3 з виходом мережі в 6 каналів, потрібно знайти лише 9 * 3 * 6 = 162 параметра.

[1)] [8] LeCun Y. et al. Gradient-based learning applied to document recognition //Proceedings of the IEEE.1998.T. 86. P. 2278–2324.

[1)] [9] Алгоритми зворотнього розповсюдження помилки. URL: www.irbis-nbuv.gov.ua /irbis_nbuv/cgiirbis_64 (Дата звернення: 30.10.2019).

2.5 Алгоритми розпізнавання образів

В даному розділі приводиться формальна постановка задачі розпізнавання образів, приклади алгоритмів розпізнавання образів математичні та з використанням нейронних мереж, а також наводиться детальний опис алгоритму Віоли - Джонса.

Перед постановкою завдання розпізнавання образів необхідно ввести поняття образу. Образ – об'єкт, який характеризується кінцевим набором ознак. Тоді розпізнавання образів – це класифікація вихідного об'єкта за допомогою пошуку і виділення значимих ознак із загального числа інформації про даний об'єкт [10][1)].

До постановки задачі можна відноситися з двох позицій. Перша – позиція теорії штучних нейронних мереж, де використовується метод експериментального отримання результату. Друга – математична, в якій основою є логічні міркування і докази з використанням математичних методів.

Таким чином формальна постановка задачі розпізнавання образів виглядає так: дано вхідна множина об'єктів і його опис, опис кожного з об'єктів і інформація про класи. Кожен клас є підмножиною вхідної множини. Відносно вхідної множини об'єктів провести класифікацію об'єктів.

Як правило вхідною множиною служить бібліотека зображень, які представляються як функції. Для простоти обчислення функцій зображення, за умови відсутності втрат корисної інформації зображення можуть бути представлені в монохромному вигляді. Тоді зображення розкладується в безліч точок на площині S, де B (x, y) – це функція яскравості (світності, або іншої подібної характеристики) кожної точки, параметрами якої є координати точки [11][2)].

[1)] [10] Ту Дж., Гонсалес Р. Принципы распознавания образов. М.: «Мир».1978. С. 34-45.

[2)] [11] Чабан Л. Н. Теория и алгоритмы распознавания образов. Учебное пособие. М.: МИИГАиК. 2004. С. 70.

По суті, завдання розпізнавання образів зводиться до задачі оцінки відносної ймовірності приналежності вихідних даних до кожного з введених в системі класів. Її рішення здійснюється шляхом пошуку у початкового об'єкта інваріантних властивостей (ознак), що утворюють певну сукупність, притаманну якому–небудь класу.

2.6 Етапи розпізнавання образу

Розпізнавання образів може використовуватися безпосередньо на вхідному зображенні, однак в цьому випадку високої точності розпізнавання вдасться досягти лише в ідеальних умовах. При використанні реальних зображень набагато ефективніше перед розпізнаванням зробити початкову підготовку зображення – виконати попереднє виділення об'єктів обробки.

Алгоритми для виділення областей на зображеннях називаються фільтрами і в більшості своїй використовують єдине перетворення на всі точки вихідного зображення. Алгоритми фільтрації в порядку їх ускладнення:

Бінаризація зображення по порогу. Найшвидший і простий спосіб перетворення, використовує значення кольору зображення. На гістограмі виділяється якесь порогове значення, кольору нижче якого відсікаються і не відображаються. Використовується, в основному, для відділення контрастних предметів від однотонного фону, або пошуку на зображенні зон певного кольору (наприклад, детектор шкіри).

ФВЧ (фільтр високих частот) і ФНЧ (фільтр низьких частот) фільтри. Успішне застосування перетворення Фур'є і фільтрів високих і низьких частот при обробці сигналів може бути перенесено і перенесено і на обробку зображень, які представляються як складні сигнали. Потім відбувається згортка з готовим фільтром (наприклад, Гаусса або Габора). Результатом обробки зображення таким фільтром є видалення шуму або його винесення в окремий канал зображення.

Вейвлет – перетворення [12][1)]. Багато в чому схожий з попереднім способом, але для згортки з вихідним зображенням використовують моделі визначених патернів. Основне завдання даного методу – вибрати правильний патерн або набір патернів. Існують кілька класичних паттернів для вейвлет-аналізу: вейвлети Хаара, Добеши, Мейера, максиканський капелюх відповідно представлені на рисунку 2.4. У деяких випадках вейвлетами є самі зображення, які необхідно знайти

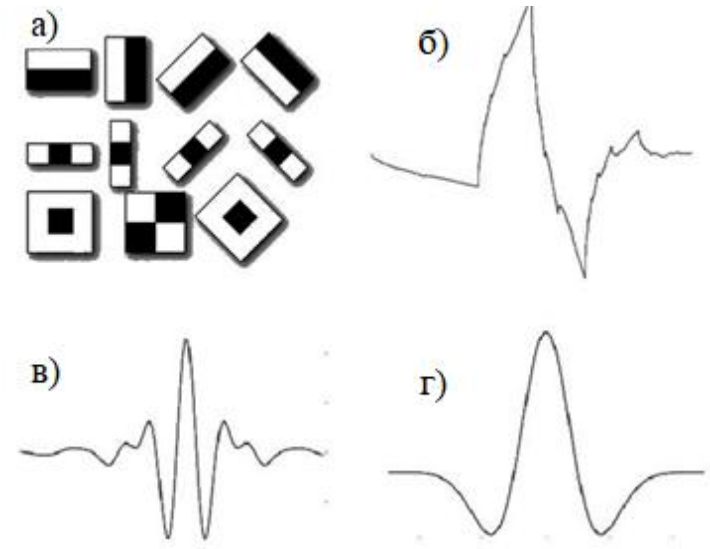


Рисунок 2.4 – а) Вейвлети Хаара, б) Добеши, в) Мейера, г) мексиканський капелюх

Фільтрації функцій. Повністю математичний метод, заснований на пошуку функцій, що породжують кожну з точок вихідного зображення. Для математичної функції відтворення використовують параболи, кола, або, в простому випадку, пряму. Класичним, в даному методі, можна назвати досить повільне перетворення Хафа, або перетворення Радону, що використовує швидке перетворення Фур'є для інтерполяції.

[1)] [12] Lee B. Tarng, Y. S. Application of the discrete wavelet transform to the monitoring of tool failure in end milling using the spindle motor current. International Journal of Advanced Manufacturing Technology. 2009. P. 238–243.

Фільтрація контурів. Набір фільтрів, що дозволяє шукати на зображеннях точки максимально відмінні за значенням кольору від сусідніх. Дозволяють виділяти контури будь-яких складних ділянок зображень за умови їх відмінності від фону. До контурних фільтрів відносять оператор Кенні, Собеля, Лапласа та ін. Використовуються в основному для деталізірованних зображень з великою роздільною здатністю.

Методи виділення образів можуть використовуватися як поодинці, так і в комбінаціях, обумовлених умовами технічного завдання. Основним недоліком алгоритмів є значне збільшення часу обробки з ростом розміру вихідного зображення

Після застосування до даних фільтрів вони стають придатними для логічної обробки. Загальне завдання обробки можна сформулювати так: перейти від набору пікселів зображення до об'єктів або властивостям цих об'єктів. Далі будуть описані способи здійснення таких переходів.

Математична морфологія є безпосереднім прикладом застосування теорії множин і логічних операцій над множинами до зображень, представленим як набір точок. На додаток до вихідного зображення задається набір спеціальних бінарних (пікселі можуть бути лише двох кольорів) патернів. Потім за допомогою логічних операцій над обома множинами пікселів: нарощування, ерозії, замикання і розмикання виконується обробка зображення. За допомогою морфологічної обробки можна знайти границю об'єкта, видалити малі об'єкти, прибрати шум і т.д., тому використовується він в основному для поліпшення зображень в графічних редакторах.

Контурний аналіз, наприклад, літак або пістолет, досить легко розпізнати по їх контуру, який має досить очевидну форму. Контурний аналіз має досить просту логіку, про яку було сказано раніше і швидкий математичний аппарат, однак для точного розпізнавання неочевидних контурів об'єктів вимагає ідеальних умов.

Відстеження особливих точок. Особливими точками називають досить широку безліч об'єктів, яке цілком залежить від завдання і об'єкта пошуку. З

одного боку, в якості особливих точок можуть виступати локальні екстремуми зображення, різкі кути, максимуми дисперсії та т.п. У цій ситуації методи відстеження виявляються більш універсальні і під область їх завдань потрапляють відстеження руху певних об'єктів в відеопотоці, стабілізація відеоізображеня, пошук аномалій форми, зведення зображень з різних камер і ін. Однак більш широка спеціалізація компенсується складністю і необхідністю більшої кількості точок для розпізнавання об'єкта. З іншого боку, якщо підходити до області роботи більш предметно, то евристично можна виділити якісь особливі точки властиві лише об'єкту розпізнавання. Це може бути наявність гострого кута у ножа або будь-яка інша комбінація ознак, за якими можна однозначно ідентифікувати об'єкт. При вдалому виборі таких точок відбувається незначний програш за складністю ознаки, але більший виграш за загальною кількістю ознак, необхідних для розпізнавання об'єкта.

За результатами розпізнавання по зображенню властивостей об'єкта необхідне подальше прийняття рішення про статус або класифікацію об'єкта. Для цього використовуються принцип навчання, описаний в загальному випадку в першому пункті.

Щоб виконати навчання спочатку необхідно зіставити дані логістичного аналізу вихідних даних навчання [13][1)]. Є набір ознак вихідного зображення. У кожної з ознак є вага, яка відображає внесок ознаки в остаточне прийняття рішення. На етапі тренування в просторі ознак проводиться розбиття на області, відповідно кожному з рішень. З кожної ітерації простір ознак стає все більш роздробленим, причому в основному це відбувається не за рахунок зміни коефіцієнта співвідношення між частками простору, що відносяться до кожного ознакою, і за рахунок все більш точного розподілу областей простору, що відповідають необхідним ознаками.

На простому прикладі бінарного вибору, наприклад, якщо необхідно визначити чи є на зображенні певний об'єкт, можна розглянути послідовне

[1)] [13] Ripley B. D. Pattern recognition and neural networks. Cambridge university press. 2007. P 3-30.

розбиття простору на зони і простежити поступове зменшення (в даному випадку ще й збіжність) помилки. На рисунку 2.4 показане послідновне навчання для кількості ітерацій t = 1,7,40 на прикладі бінарного класифікатора (бустера) AdaBoost, а на рисунку 2.5 збіжність помилки.

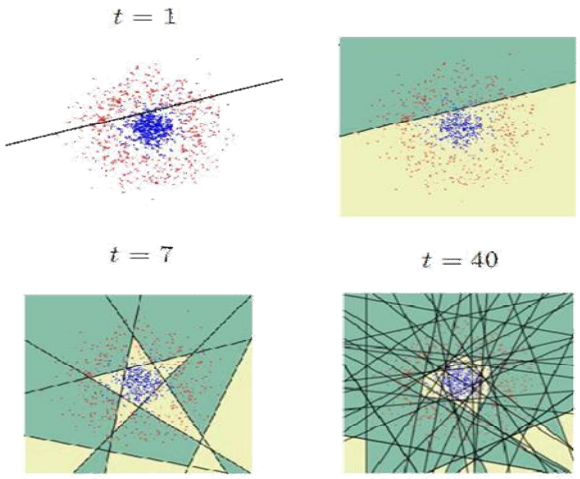


Рисунок 2.4 – Процесс навчання в просторі ознак

Завдяки даному набору технік система автоматично знайде представлення для виявлення ознак або класифікаціх початкових даних. Ці дії замінять ручне конструювання ознак і дозволить машині як і вивчати ознаки так і використовувати їх для вирішення специфічних задач, в даному випадку – пошук обличчя людини на вхідному зображенні.

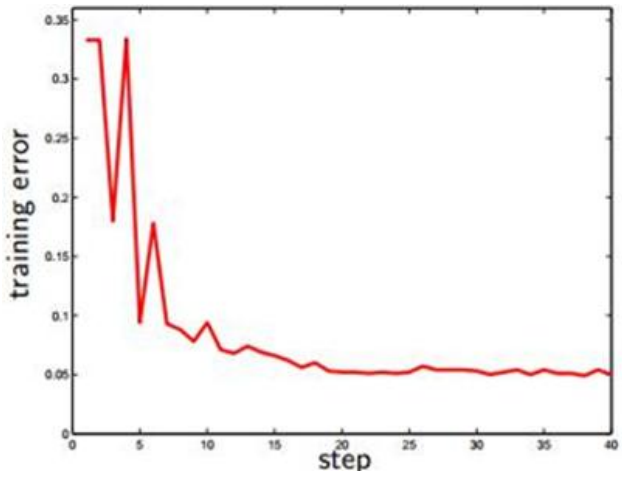


Рисунок 2.5 – Графік залежності похибки від ітерації ознак

Таким чином, правильний вибір алгоритмів обробки на кожному з етапів розпізнавання зображення в залежності від конкретної розв'язуваної задачі дозволяє виконати обробку з максимальною точністю і використовуючи мінімум апаратних ресурсів ЕОМ.

Виходячи із вимог до даної системи були розглянуті основні види і принципи навчання нейронних мереж, більш детально розглянуті згорткові нейронні мережі. Для вирішення задачі ідентифікації обличчя шляхом використання штучних нейронних мереж було розглянуто основні методи і алгоритми пошуку образів на зображеннях.

3 ПРОЕКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ

В даному розділі буде сформована архітектура системи, діаграми та алгоритми роботи системи. Будуть поетапно та детально розглянуті алгоримтми роботи проектуємої системи для ідентифікації обличчя людини за допомогою нейронних мереж. Буде обрана мова програмування системи.

3.1 Мова програмування та бібліотеки

Для взаємодії з нейронною мережею та програмуванням самої системи була обрана мова програмування Python.

Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою [14][1)]. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом з динамічною семантикою і динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python і стандартні бібліотеки доступні як в скомпільованій так і у вихідній формі на всіх основних платформах. Серед основних її переваг можна назвати наступні:

- чистий синтаксис (для виділення блоків слід використовувати відступи); переносимість програм (що властиво більшості інтерпретованих мов); • стандартний дистрибутив має велику кількість корисних модулів (включаючи модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисно для експериментування та рішення простих задач);

[1)] [14] Марк Саммерфилд. Python на практике. М.: ДМК Пресс. 2014. С. 338.

- стандартний дистрибутив має просте, але разом з тим досить потужне середовище розробки, яка називається IDLE (Integrated Development and Learning Environment) і яке написано на мові Python;
- зручний для вирішення математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, в діалоговому режимі може використовуватися як потужний калькулятор). Python має ефективні структури даних високого рівня і простий, але ефективний підхід до об'єктно-орієнтованого програмування.

Інтерпретатор мови Python може бути розширений функціями і типами даних, розробленими на C або C ++ (або іншою мовою, яку можна викликати за C). Python також зручна як мова розширення для додатків, що вимагають подальшого налагодження.

Після вибору середовища розробки для виконання поставленої задачі потрібно розглянути бібліотеки, які матимуть змогу працювати з обраною мовою програмування, та допоможуть в роботі з нейронною мережую. В розпізнаванні об’єктів найкращими є бібліотека OpenCV та Dlib. Обидві мови написані на мові C++, підтримуються мовою Python, мають набір схожих функцій. Тому основним критерієм при виборі бібліотеки буде їх точність.

Бібліотека OpenCV краще працює з малими зображеннями, також коли невідомо місце знаходження обличчя на картинці. Dlib швидше працює на процесорі, має проблеми зі знаходженням малих облич (70x70 точок). Розроблювана система не буде націлена на роботу з настільки малими зображенням. Крім того, для детектування можна використовувати графічний процесор.

3.2 Формування загального алгоритму для роботи системи

Мова Python є об’єкто-орієнтованою мовою програмування, передбачає використання інкапсуляції, поліморфізму і наслідування. Так як основною

структурою для обробки зображень і збереження інформації про набір даних є нейронна мережа, то система для роботи з нею повинна бути відведена в окремий клас. При алгоритмів і методів роботи із зображенням, я прийшов до розуміння роботи алгоритму системи, коротка блок-схема дій алгоритму наведена на рисунку 3.1.

Рисунок 3.1 – Блок-схема роботи алгоритму знаходження ознаки, за допомоги якої відбувається порівняння облич

3.3 Проектування системи за допомогою методології UML

Діаграма варіантів використання показує відношення між користувачем системи і прецедентами (частинами функціоналу системи). Головною

функцією діаграми використання є опис функціональності і поведінки системи, та результати взаємодії з користувачем даної системи [15][1)].

На рисунку 3.2 зображена діаграма прецедентів проектуємої системи ідентифікації обличчя.

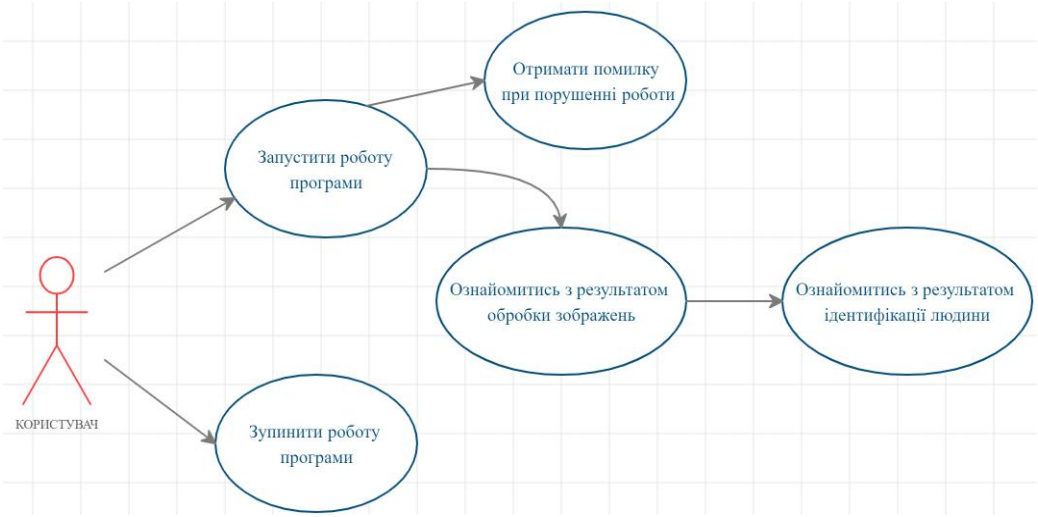


Рисунок 3.2 – Діаграма прецедентів проекту

Діаграма показує, що користувач може запускати систему на виконання або зупинити її дію. При невдалому процесі виконання система сповістить користувача про невдалий процес виконання. Якщо в процесі виконання помилок не сталося, користувач має змогу ознайомитися з результатами роботи системи на кожному зображенні. Після переконання в правильному детектуванні користувач має змогу оглянути висновок системи про ідентифікацію.

3.4 Проектування системи за допомогою методології DFD

Діаграма потоків даних це методологія графічного структурного аналізу, яка описує зовнішні по відношенню до системи джерела і адресати даних, логічні функції, потоки даних та сховища до яких виокнується доступ [16][1)].

[1)] [15] Діаграми UML. URL: https://prog-cpp.ru/uml-classes/ (Дата звернення 4.11.2019).

На рисуннку 3.3 зображена діаграма потоків даних проектуємої системи для ідентифікації обличчя.

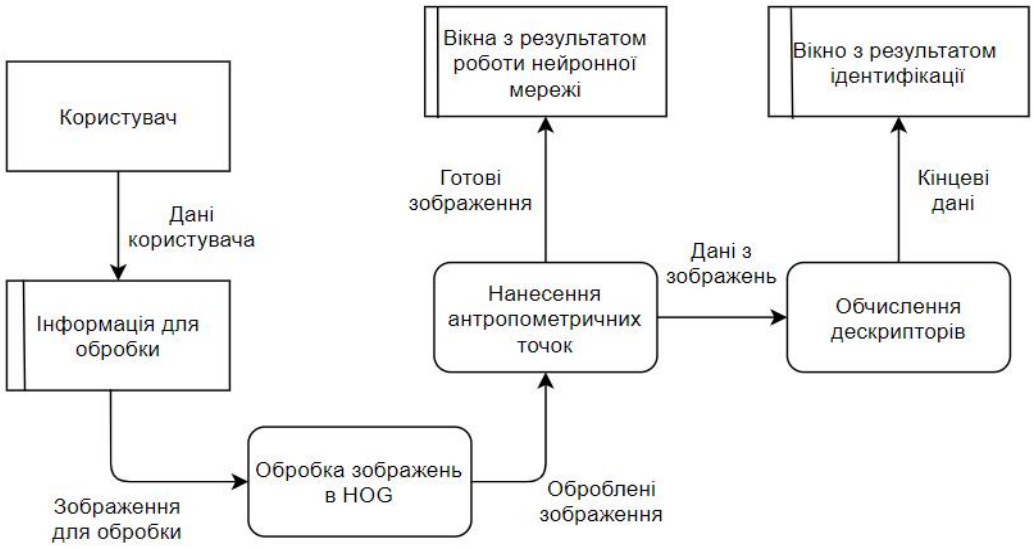


Рисунок 3.3 – Діаграма потоків даних

Завдяки даній діаграмі потоків даних стало можливим візуалізувати процеси і потоки даних в розроблюваній системі. Можна побачити взаємодію між модулями системи. В майбутньому будь-яка діаграма може бути уточнена шляхом деталізації процесів та потоків даних з метою показати розлого розроблювану систему, але в даному випадку такої потреби немає.

3.5 Штучна мережа даних ResNet

Проаналізувавши види нейронних мереж, я прийшов до висновку, що нйабільш оптимальним видом нейронних мереж для вирішення поставленої задачі будуть згорткові нейронні мережі, а саме згорткова мережа ResNet.

[1)] [16] Методологія DFD. URL: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6_3 (Дата звернення 4.11.2019).

Глибокі згорткові нейронні мережі перевершили людський рівень класифікації зображень в 2015 році [17][1)]. Глибокі мережі витягають низькорівневі, середньорівневі і високорівневі ознаки наскрізним багатошаровим способом, а збільшення кількості шарів може збагатити «рівні» ознак. Кількість стеків шарів має вирішальне значення, зображено на рисунку 3.5.

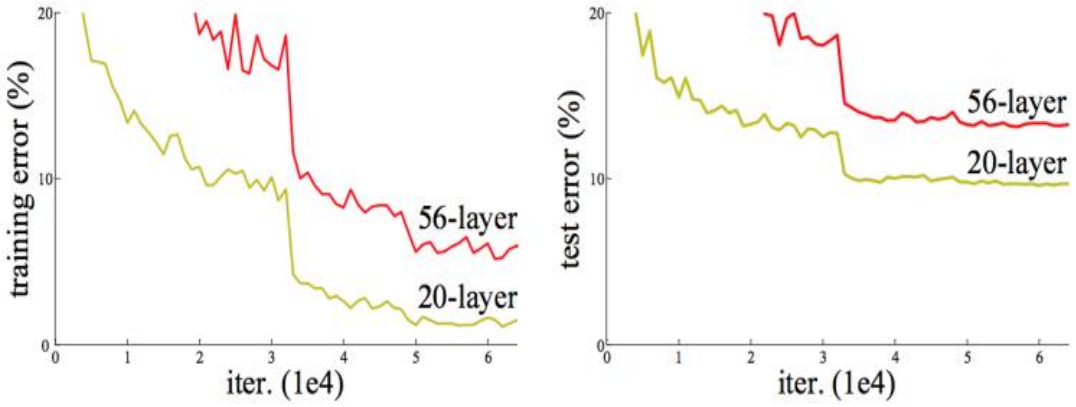


Рисунок 3.5 – Похибка навчання (зліва) і похибка тесту на CIFAR з 20-рівневими і 56-рівневими мережами.

З рисунку 3.5 видно, що більш глубока мережа має більшу похибку навчання і, слідуючи, похибку тестування. Коли більш глубока мережа починає згортатися, з'являжться проблема: зі збільшенням глибини мережі точність синалу збільшується, а потім різко погіршується. Зменшення точносіт навчання показує, що не всі мережі легко оптимізувати.

Щоб здолати цю проблему, було введено глубоку «залишкову» стркту-ру навчання. Замість того, щоб наіятись на те, що кожні кілька стеків шарів безпосередньо відповідають бажаному основному представленню, вони явно дозволяють цим шарам відповідати «залишковому» [8][1)].

[1)] [17] Fitzpatrick, David. The Functional Organization of Local Circuits in Visual Cortex: Insights from the Study of Tree Shrew Striate Cortex. London: Cerebral Cortex. 2016. P. 329–341.

[2)] [8] LeCun Y. et al. Gradient-based learning applied to document recognition //Proceedings of the IEEE.1998.T. 86. P. 2278–2324.

З'єднання швидкого доступу пропускають один або кілька шарів і виконують зіставлення ідентифікаторів. Їх виходи додаються до виходів шарів. Використовуючи ResNet, можна вирішити безліч проблем, таких як:

– ResNet відносно легко оптимізувати: «прості» мережі (які просто складають шари) показують велику помилку навчання, коли глибина збільшується;
– ResNet дозволяє відносно легко збільшити точність завдяки збільшенню глибини, чого з іншими мережами домогтися складніше.

На основі описаної вище простої мережі додано швидке з'єднання (рис. 3.5), яке перетворює мережу в її залишкову версію. Ідентифікаційні швидкі з'єднання F (x {W} + x) можуть використовуватися безпосередньо, коли вхід і вихід мають однакові розмірності Коли розмірності збільшуються , він розглядає два варіанти:

– Швидке з'єднання виконує зіставлення ідентифікаторів з додатковими нулями, доданими для збільшення розмірності. Ця опція не вводить ніяких додаткових параметрів;
– Проекція швидкого з'єднання в F (x {W} + x) використовується для зіставлення розмірності (виконано за допомогою 1 × 1 згорток);

3.6 Метод гістограм орієнтованих градієнтів

За допомогою даного методу є підготовка зображення для роботи алгоритму класифікації шляхом перетворення зображення в форму, більш придатну для алгоритму класифікації зображення.

Основною ідеєю алгоритму є припущення, що зовнішній вигляд і форма об'єкта на ділянці зображення можуть бути описані розподілом градієнтів інтенсивності чи напрямком країв. Реалізація цих дескрипторів може бути проведена шляхом поділу зображення на маленькі зв'язкові області, іменовані комірками, і розрахунком для кожної комірки гістограми напрямків градієнтів або напрямків країв для пікселів, що знаходяться всередині комірки.

Комбінація цих гістограм і є дескриптором. Щоб збільшити точність локальні гістограми піддаються нормалізації по контрасту. З цією метою обчислюється міра інтенсивності на більшому фрагменті зображення, який називається блоком, і отримане значення використовується для нормалізації. Нормалізовані дескриптори мають кращу інваріантність по відношенню до освітлення.

Дескриптор HOG має кілька переваг над іншими дескрипторами. Оскільки HOG працює локально, метод підтримує інваріантність геометричних і фотометричних перетворень, за винятком орієнтації об'єкта. Подібні зміни з'являться тільки в великих фрагментах зображення. Більш того, як виявили Далал і Тріггс, грубе розбиття простору, точне обчислення напрямків і сильна локальна фотометрична нормалізація дозволяють ігнорувати рух пішоходів, якщо вони підтримують вертикальне положення тіла. Дескриптор HOG таким чином, є хорошим засобом знаходження людей на зображеннях.

При описі фрагментів зображення воно розбивається на декілька невеликих ділянок, які найменуються «комірки». В комірках обчислюються гістограми hi напрвавлених градієнтів внутрішніх точок. Зазвичай вони об’єднуються в одну гістограму h = f (h1…,hk), після чого вона нормалізується по яскравості.

$$h_{L_2} = \frac{h}{\sqrt{|h|_2^2 + \varepsilon}}, h_{L_1} = \frac{h}{|h|_1 + \varepsilon}, \sqrt[h]{L_1} = \sqrt{h L_1} \quad (3.1)$$

,

В дескрипторі фунції HOG в якості функції використовують розподілення напрямку градієнтів. Для розрахунку використовують розрахунок градієнта. На цьому кроці вираховується величина gx та gy градієнта із оригінального зображення, це можна зробити за допомогою фільтрацій з наступними ядрами: -1;0;1 та -1;0;1. Використовуючи ці ядра обчислюється величина і напрям градієнту.

В результаті чого створюються дві матриці Dx та Dy похідних вздовж осей x та y відповідно.

Метою є виділення того, наскільки темним є поточний піксель в порівнянні з пікселями, які примикають до нього. Потім можна провести умовну стрілку, напрям якої показує, в якому напрямку зображення стає темнішим. Після повторення даного процесу до кожного пікселю на зображенні, то в кінцевому результаті кожний піксель буде замінено на стрілку. Ці стрілки називають градієнтом, вони показують оптіік від світла до темноти по всьому зображенню (рис. 3.6)

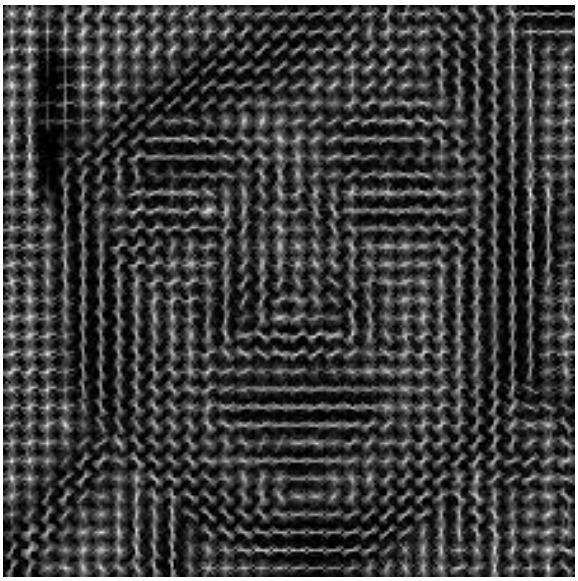


Рисунок 3.6 – Карта HOG

Для прийняття до уваги яскравості і контрастності градієнти слід локально нормувати, для чого комірці потрібно групувати в більші зв'язкові блоки. Дескриптор HOG, таким чином, є вектором компонент нормованих гістограм комірок з усіх областей блоку. Як правило, блоки перекриваються, тобто кожна клітинка входить більш ніж в один кінцевий дескриптор. Використовуються дві основні геометрії блоку: прямокутні R-HOG і круглі C-HOG. Блоки R-HOG зазвичай є квадратними сітками, що характеризуються трьома

параметрами: кількістю комірок на блок, кількістю пікселів на комірку і кількістю каналів на гістограму кумірки.

Більше того, поліпшення продуктивності можна досягти, застосувавши фільтр Гаусса у кожному блоці перед таблицею голосів гістограми, щоб менше важити пікселів по краю блоків. Блоки R-HOG виглядають досить схожими на масштабні інваріантні дескриптори перетворення функцій, однак, незважаючи на подібне утворення, блоки R-HOG обчислюються в щільних сітках на деякій шкалі без вирівнювання орієнтації, тоді як дескриптори SIFT (Scale-invariant feature transform) обчислюються в розріджених, інваріантних масштабах ключових точках зображення і повертаються для вирівнювання орієнтації. Крім того, блоки R-HOG використовуються спільно для кодування інформації про просторові форми, тоді як дескриптори SIFT використовуються поодиноко.

Блоки C-HOG можна знайти у двох варіантах: блоці з одинарною центральною коміркою та центральній комірці з кутом. Крім того, ці блоки C-HOG можна описати чотирма параметрами: кількістю кутових та радіальних бункерів, радіусом центральної бункери та коефіцієнтом розширення для радіусу додаткових радіальних бункерів.

Може здатися, що результатом є дещо випадкове, але є хороша причина для заміни пікселів градієнтами. Коли ми аналізуємо пікселі, то в темних і світлих зображень однієї і тієї ж людини будуть сильно відмінні значення інтенсивності пікселів. Але якщо розглядати тільки направлення яскравості, то як темне так і світле зображення будуть мати абсолютно однакове представлення. Це значно полегшує вирішення проблеми.

Але збереження градієнта для кожного окремого пікселя дає спосіб, що несе занадто багато подробиць, а також гістограма з минулого кроку не дуже стійка до змін освітлення. Було б краще, якби можна було просто бачити основний потік світлого чи темного на більш високому рівні, розглядаючи таким чином базову структуру зображення.

Для цього потрібно розбити зображення на невеликі квадрати 16x16 пікселів в кожному.У кожному квадраті слід підрахувати, скільки градієнтних стрілок показує в кожному головному напрямку (тобто скільки стрілок направлено вгору, вгору-вправо, вправо і т.д.). Потім розглядається квадрат на зображенні замінюють стрілкою з напрямком, що переважає в цьому квадраті.В кінцевому результаті ми перетворюємо вихідне зображення в дуже просте представлення, яке показує базову структуру особи в простій формі. Щоб знайти обличчя на даному HOG-зображенні потрібно знайти таку ділянку зображення, яка найбільше схожа на відому HOG-структуру, отриману з групи обличь, які були використані для навчання.

3.7 Алгорим класифікації

Після роботи HOG алгоритму вихідна інформація готова до подальшої обробки та потрапляє на алгоритм класифікації зображення.За допомогою реалізації даного алгоритму нейронна мережа зможе знайти обличчя на вхідному зображенні та відділити його від іншої частини зображення. Дана операція дозволить системі далі провести необхідні операція з нормалізації зображення та підготовки до знаходження та зчитування дескрипторів обличчя.

Перед стабільною роботою, класифікуючий алгоритм необхідно натренерувати, показавши зображення на яких є предмет який нас цікавить або його немає. В своїй роботі я використав алгоритм під назвою SVM (метод опорних векторів). SVM (Support Vector Machine) один за найбільш популярних алгоритмів двійкової класифікації, який відноситья до методів навчання з вчителем, який добре підходить для вирішення завдання класифікації. Для розуміння роботи розмірність характеристичного вектору зменшено до двійки. Робота даного алгоритму проілюстрована на рисунку 3.7.

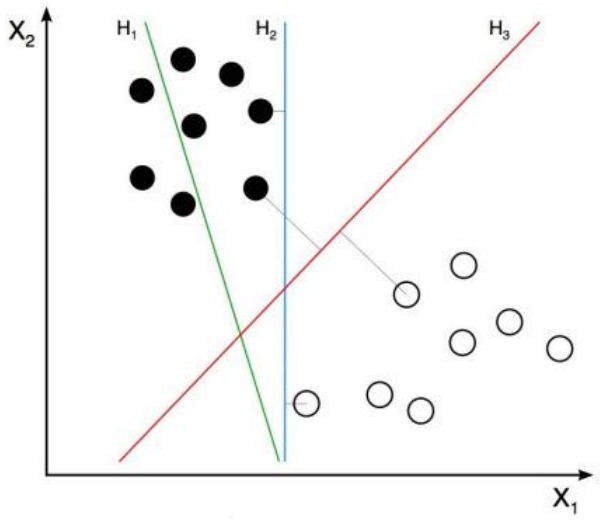


Рисунок 3.7 – Робота SVM алгоритму

Точки на зображенні представляють два класи (наявність об'єкта і його відсутність). На рисунку 3.7 два класи представлені двома типами точок. У процесі тренувань алгоритму пропонується багато прикладів зображених двох класів, алгоритм збирає інформацію про координати точок, а також, чи є точки чорними або білими. SVM намагається знайти найбільш точну лінійність, яка розділяє два класа. На рисунку 3.3 є три лінії H1, H2 і H3. H1 не розділяє два класса і не є хорошим класифікатором. H2 і H3 розділяють два класи, але H3 робить це найкраще. Таким чином, в процесі тренування SVM знайде лінію H3. Якщо вектор є трьохмірним, SVM побудує відповідну площину, яка максимально розділяє два класи. Для дескриптора HOG буде побудована гіперплощина (рисунок 3.8).

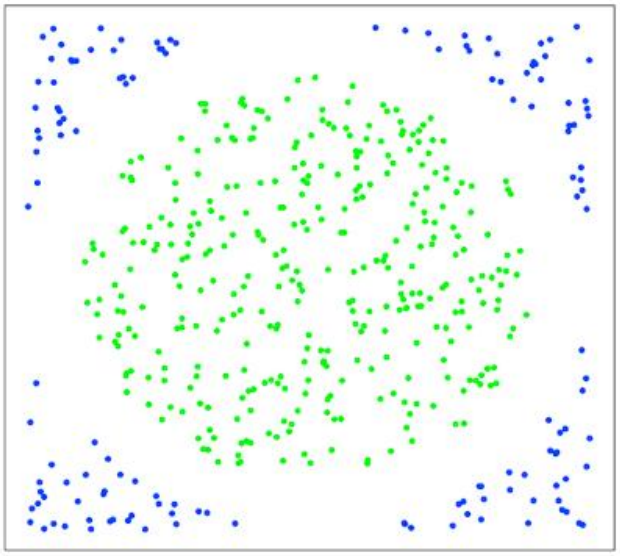


Рисунок 3.8 – Приклад гіперплощини

Оптимальна роздільна гіперплощина – це площина, відстань від якой до найблищого вектора з любої сторони максимально. А найблищі до площини вектори, «опираючись» на які вона побудованга і є опорними векторами.

3.8 Активні моделі зовнішнього вигляду

Для виокнання задачі розпізнавання на виділеному обличчі певних параметрів для подальої ідентифікації людини обрано метод активних моделей зовнішнього виду (Active Appearance Models, AAM) [18][1)]. Активна модель зовнішнього вигляду має два типи параметрів: параметри, пов'язані з формою (параметри форми) та параметри, пов’язані зі статистичною моделлю пікселів зображення або текстурою. Перед виконистанням даної моделі вона повина була бути навчена на великій кількості зазделагіть розмічених зображень, тому для реалізації поставленої мети була обрана заздалегідь навчена модель. Кожна мітка на даній моделі має свій номер і визначає характерну

[1)] [18] Cootes, T. F., Edwards, G. J., Taylor, C. J. Active appearance models Computer Vision – ECCV'98. Lecture Notes in Computer Science. 1998. P. 484.

точку, яку має знайти модель під час адаптації до нового зображення, це показано на рисунку 3.9.

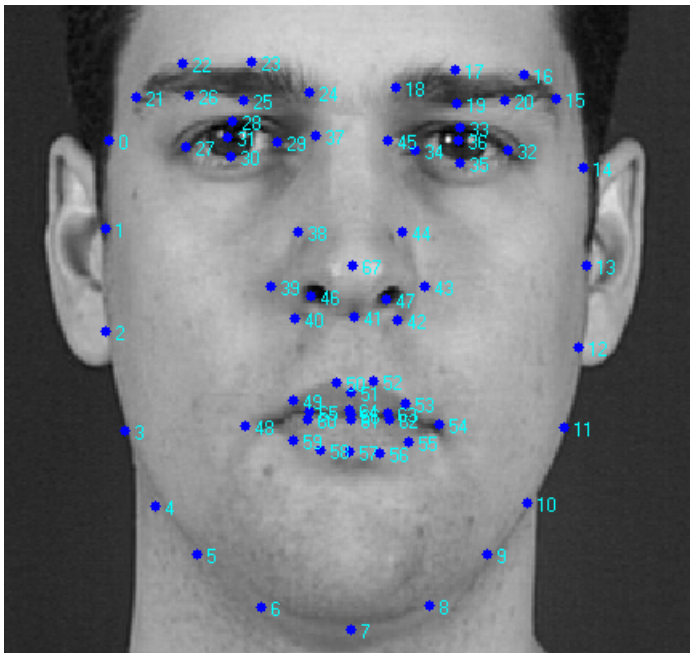


Рисунок 3.9 – Приклад розмітки зображення обличчя з 68 точок, які утворюють форму AAM

В якості даних точок можуть бути кути очей, рот, брови і т.п. Процедура навчання AAM починається з нормалізації форм на розмічених зображеннях з метою компенсацій різниці в масштабі, нахилі та зміщенні. Для цього використовується метод, який був визначений Вахідом Каземи та Джозефом Салліваном у 2014 році.

Даний алгоритм виконує пошук контурів на базі AAM за мілісекунди та досягає великої точності та швидкості, працює краще за аналогічні алгоритми. Збільшення швидкості порівняно з іншими алгоритмами пов’язано з наслідком ідентифікації основних компонентів попередній алгоритмів пошуку контурів обличчя і наступного їх включення в оптимізованій формі в каскад регресійних моделей з високою пропускною здатністю, налаштованих за допомогою градієнтного бустингу.

В даній роботі було вирішено, що саме регресійна модель в каскаді ефективно зможе спрогнозувати форму обличчя на основі попереднього прогнозу інтенсивності пікселів, проіндексованих порівняно з попереднім прогнозом. В даному алгоритмі один процес виконує роль сканування області індексації інтенсивності пікселів відносно прогнозу форми обличчя. Виділенні ознаки в векторному представленні можуть відрізнятися через деформацію форми, так і через фактор освітлення. Щоб уникнути даної проблеми в даній роботі зображення попередньо перетворюється в нормалізовану систему координат на основі поточного прогнозу форми обличчя, а потім знаходяться ознаки для прогнозування оновлюючого вектора для параметрів форми, даний процес може повторюватись декілька разів. Другий процес вирішує проблеми пояснення форми обличчя – вектор високої розмірності, котрий найкращим чином узгоджується з даними зображення і моделлю форми. В даному алгоритмі оптимізовується функція Даний алгоритм вирішує дану проблему, припускаючи, що прогнозована форма повинна бути в лінійному просторі, котре можна знайти шляхом знаходження основних компонентів навчальних форм. Нижче наведено основні поняття навчання каскадів регресії, які будуть використані в даній роботі.

Для початку введемо деякі позначення. Нехай Xi є R2 – x, y-координати і-го орієнтира обличчя на зображенні I. Тоді вектор $S = (x_1^T, x_2^T \dots x_n^T)^T \in R^{2p}$ позначає координати всіх p лицьових орієнтирів в I. Для оцінки вектору використана змінна S_1^t. Кожен регресорів r_t в каскаді пророкує вектор поновлення з зображення S_1^t і, який додається до поточної оцінки форми, щоб поліпшити оцінку:

$$S_1^{t+1} = S^t + r_t(I, S_1^t), \tag{3.2}$$

Ключовий момент каскаду полягає в тому, що регресор r_t робить свої прогнози на основі ознак, таких як значення інтенсивності пікселів, обчисле-

них по I і проіндексованих щодо поточної оцінки форми S_1^t. Це вводить деякий рід геометричної інваріантності в процес, і в міру проходження каскаду можна бути більш впевненим в тому, що індексується точне семантичне розташування на обличчі

Діапазон вихідних даних, розширений ансамблем, гарантовано лежить в лінійному підпросторі навчальних даних, якщо початкова оцінка належить цьому простору. Тому не потрібно вводити додаткові обмеження на передбачення, що значно спрощує даний метод. Початкова форма може бути просто обрана в якості середньої форми навчальних даних, зосереджених і масштабованих відповідно до вихідними даними обмежувальної рамки загального детектора особи.

Кожний регресор навчається з використанням алгоритму градієнтного бустингу дерев. Тобто наступна модель навчається на помилках іншої та оснований на лійній регресії. Перше правило лінійної регресії в тому, що сума відхилень = 0, тобто відхилення повинні бути випадково розподілені в районі нуля (рисунок 3.10).

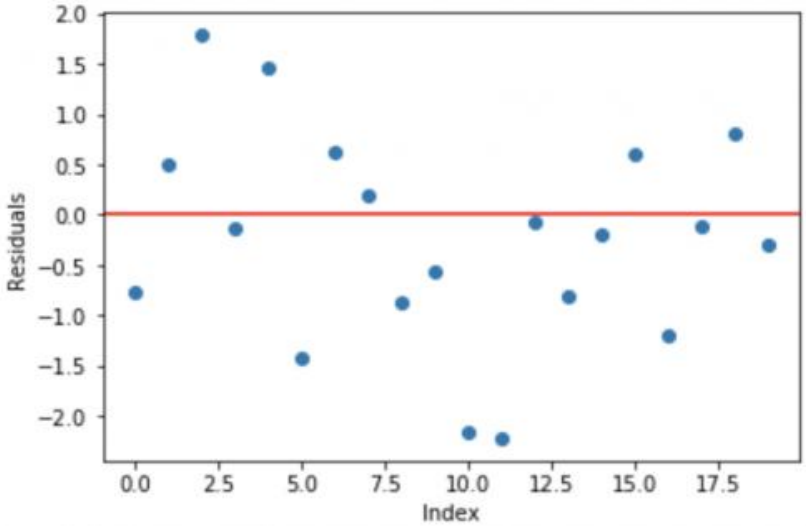


Рисунок 3.10 – Нормальне розподілення виборки відхилень з середнім 0

Задача градієнтного бустингу – ітеративно примінити петтерни відхилення і поліпшити їх передбачання. Як тільти в даному алгоритмі буде досягнуто момент, коли відхилення не мають ніякого паттерна, потрібно зупинити навчання моделі.

Перевагою системи, яка буде побудована в даній роботі є новий метод пошуку контурів обличчя, який оснований на ансамблеї дерев регресії, котрий виконує набір інваріативних ознак форми, мінімізуючи при цому функцію втрат під час навчання, котра повинна бути мінімізована під час тестування системи.

3.9 Афінні перетворення

Після знаходження основних точок на обличчі та накладення на нього знайденої маски потрібно привести два зображення обличчя до єдиного вигляду, адже кут нахилу, відхилення обличчя чи голови на зображеннях може бути різним, відповідно і підрахунки дескриптора обличчя будуть не вірними. Для вирішення даної проблеми було вирішено застосувати афінні перетворення. Афінні перетворення це перетворення системи координат за допомогою множення вектора на спеціальну матрицю. Такі перетворення як: поворот, переміщення, масштабування, відзеркалення та інші. Основною особливістю даних перетворень є те, що об’єкти залишаються в тому ж просторі і те, що якщо прямі пертинались, були паралельні, схрещувались до перетворення, то ця особливість після перетворення зберігається. Приклад афінних перетворень показано на рисунку 3.11.

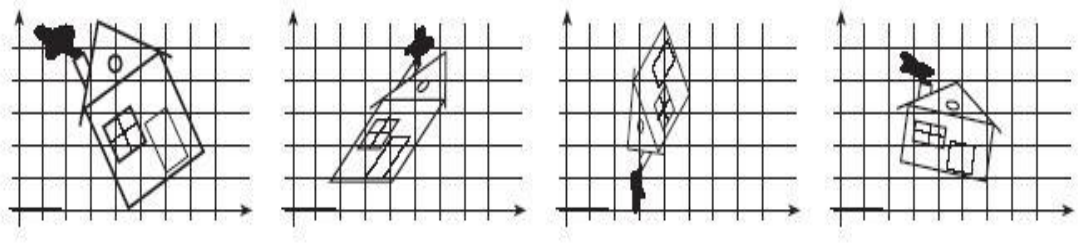


Рисунок 3.11 – Приклад афінних перетворень

Вирівнювання зображення обличчя на площині виконується наступним чином:

– На зображенні шукають шість опорних точок: центри очей, кінчик носу, края та центри роту;
– По даним точкам поновлюють три параметри, яки характеризують положення обличчя на площині: розмір, поворот та здвиг;
– Знайдене лінійне перетворення приміняють до зображення обличчя, в результаті чого буде отримано вирівнене зображення, строго певного розміру, при чому вертикальна вісь симетрії знаходиться в центр зображення.

Результат роботи афінних перетворень зображено на рисунку 3.12

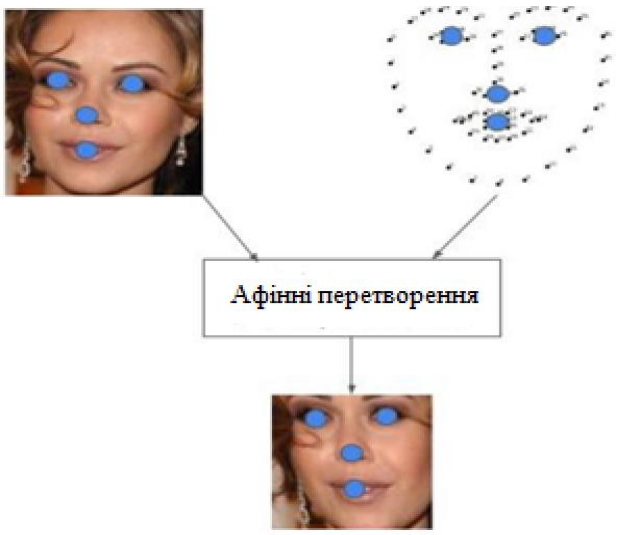


Рисунок 3.12 – Результат роботи афінних перетворень

Як видно з прикладу на рисунку 3.12, окрім вищезгаданих операцій з зображенням афінні перетворення також обрізають зображення по крайнім точкам обличчя для більш точного подальшого розрахунку схожості особи.

3.10 Робота нейронної мережі в пошуку дескриптора обличчя

Для того, щоб нейронна мережа знайшла різницю або схожість між обличчями людини потрібно обчислити дескриптори особливих точок. Для того щоб обчислити дескриптори осбливих точок в даній роботі вирішено використовувати алгоритм SIFT (Scale Invariant Feature Transform).

Основним моментом у детектуванні особливих точок є побудова піраміди гауссіанов (Gaussian) і різниць гауссіанов (Difference of Gaussian, DoG) [19][1)] це показано на рисунку 3.13. Гауссіаном (або зображенням, розмитим гаусовим фільтром) є зображення:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \quad (3.3)$$

Тут L - значення гауссіана в точці з координатами (x, y), а sigma - радіус розмиття. G - гауссово ядро, I - значення вихідного зображення, * - операція згортки.

Різницею гауссіанов називають зображення, отримане шляхом попиксельного віднімання одного гауссіна вихідного зображення з гауссіана з іншим радіусом розмитія.

$$L(x, y, \sigma) = (G(x, y, k\sigma)F(x, y, \sigma)) * I = \quad (3.4)$$

[1)] [19] Young, Richard . The Gaussian derivative model for spatial vision: I. Retinal mechanisms.1987. P. 273–293.

,

Масштабованим простором зображення є набір згладжених деяким фільтром, версій вихідного зображення. Доведено, що гауссово масштабований простір є лінійним, інваріантним щодо зрушень, обертань, масштабу, де не зміщується локальні екстремуми, і має властивість напівгруп. Для нас важливо, що різна ступінь розмиття зображення гаусовим фільтром може бути прийнята за вихідне зображення, взяте в деякому масштабі.

Загалом, інваріантність щодо масштабу досягається за рахунок знаходження ключових точок для вихідного зображення, взятого в різних масштабах. Для цього будується піраміда гауссіанов: весь масштабований простір розбивається на деякі ділянки - октави, причому частина масштабованого простору, займаного наступною октавою, в два рази більше частини, займаної попередньою. До того ж, при переході від однієї октави до іншої робиться ресемплінг зображення, його розміри зменшуються вдвічі. Природно, що кожна октава охоплює безліч гауссіанов зображення, тому будується тільки деяке їх кількість N, з певним кроком по радіусу розмиття. З тим же кроком добудовуються два додаткових гауссіана (всього виходить N + 2), що виходять за межі октави. Масштаб першого зображення наступної октави дорівнює масштабу зображення з попередньої октави з номером N.

Паралельно з побудовою піраміди гауссіанов, будується піраміда різниць гауссіанов (рисунок 3.13), що складається з різниць сусідніх зображень в піраміді гауссіанов. Відповідно, кількість зображень в цій піраміді буде N + 1.

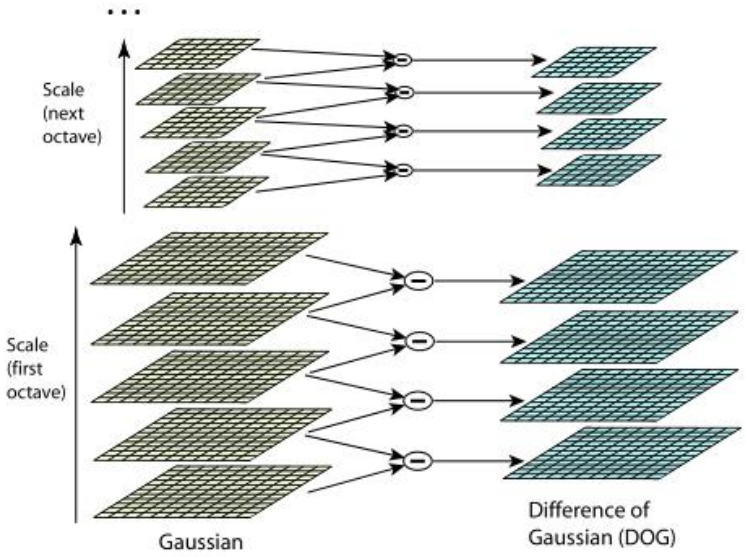


Рисунок 3.13 – Піраміда гаусіанів (зліва), різниця гаусіанів (справа)

На рисунку 3.13 показано, що кожна різниця отримується з двох сусідніх гаусіанів, кільскість різностей на одиницю менше кількості гаусіанів, при переході до наступної октави розмір зображення зменшується вдвоє.

Пілся побудови пірамід виявляються точки, які виявляються локальними точками екстремуму різниці гаусіанів.

В даній роботі доцільним є використання наступного методу пошуку екстремумів: в кожному зображенні із піраміди DoG шукаються точки локального екстремуму (рисунок 3.14), де кожна точка порівнюється з її сусідами та сусідами з DOG, які знаходяться вище чи нижче в піраміді. Якщо ця точка більше (менше) всіх сусідів, то вона приймається за точку екстремуму.

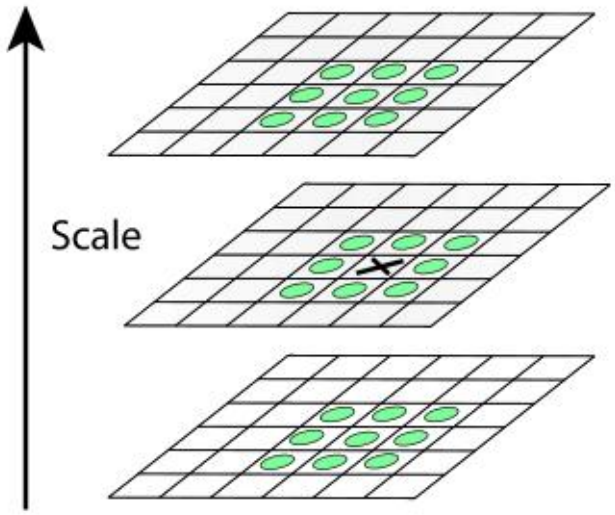


Рисунок 3.14 – Графічне зображення пошуку
локального екстремуму

Наступним кроком є перевірка придатності точки екстремума на роль ключової. Дана операція досягається за допомогою апроксимаціх функції DOG многочленом Тейлора другого порядку, взятого в точці обчислення екстремуму:

$$D(x) = D + \frac{\delta D^T}{\delta x} x + \frac{1}{2} x^T \frac{\delta^2 D}{\delta x^2} x \quad , \tag{3.5}$$

В даному прикладі:

D-функція DoG;

x = (x,y,sigma) – вектор зміщення відносно точки розложення, перша похідна DoG – градієнт, друга похідна DoG – матриця Гессе.

Екстремум многочлена Тейлора знаходиться шляхом обчислення позіднох і приірвнювання xx до нуля. В кінці даних обчислень буде отримано зміщення точки обчисленого екстремума, відносно точного:

$$x = -\frac{\delta^2 D^{-1}}{\delta x^2}\frac{\delta D}{\delta x}, \tag{3.6}$$

Всі похідні обчислюються по формулам кінцевих різниць. Отримаємо СЛАР розмірності 3x3 відносно компонент вектора x.

Коли положення точки екстремума обчислено, перевіряється значення DoG в точці по формулі:

$$(x) = D + \frac{1}{2}\frac{\delta D^T}{dx}x, \tag{3.7}$$

Якщо ця точка не проходить, вона виключається як точка з малим контрастом. Якщо особа точка лежить на границі будь-якого об'єкта, або погано освітлена, то таку точку алгоритм не прйимає за точку екстремуму.

Після знаходження екстремума, алгоритм повинен обчислити її орієнтацію (точка може мати декілька направлень). Всі обчислення градієнтів знаходяться на зображенні в піраміді гауссіанів, з масштабом найбільш близьким до ключової точки. Спочатку алгоритм знаходить вікно ключової точки, на якому будуть розглянуті градієнти. Направлення точки знаходять з гістограми направлення градієнтів, мова про яку шла в попередніх розділах.

Далі алгоритм обчислює сам дескриптор, в методі SIFT [20][1)] дескриптором є вектор. Як і напрям ключової точки. Дескриптор обчислюється на гауссіані, найблищій до масштабу по ключовій точці, роблячи висновок з градієнтів в деякому вікні ключової точки. На рисунку 3.15 показано отриманий дескриптор.

[1)] [20] Основні положення SIFT. URL: https://opencv-python-tutorials.readthedocs/.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html (Дата звернення 10.11.2019).

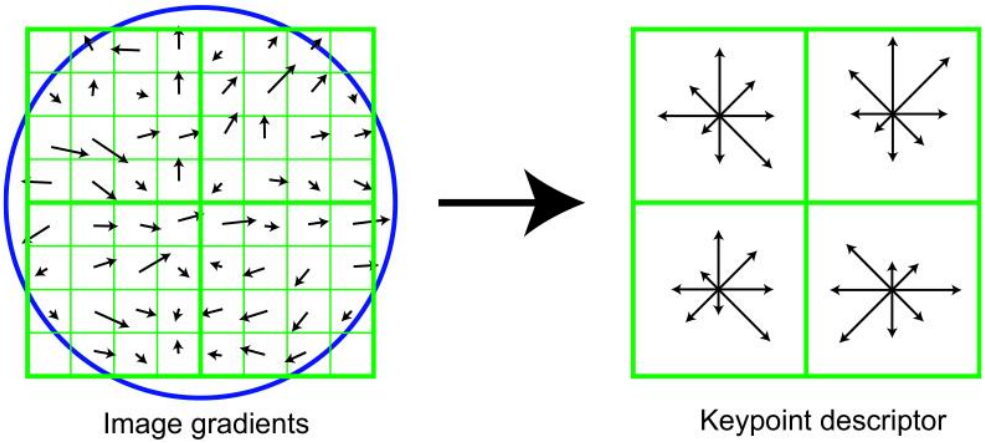


Рисунок 3.15 – Отриманий дескриптор

На лівій частині показано пікселі, які представляють собою квадрати. Ці пікселі отримані з квадратного вікна дескриптора, яке в свою чергу поділено на чотири рівні частини (регіони). Стрілка всередені кожного пікселя показує градієнт даного пікселя. Коло це визначення вікна згортки з гауссовим ядром. Для даного ядра обчислюжться значення, рівне половині ширини вікна дескриптора. В подальшому при роботі алгоритму значення кожної точки вікна дескриптора буде помножено на значення гауссового ядра в даній точці, як на ваговий коефіцієнт.

На рисунку 3.15 зправа показано схематично зображений дескриптор особливої точки розміром 2x2x8. Перші дві цифри показують кількість регіонів по горизонтіл і вертикалі. Третя цифра показує кількість компонент гістограми даного регіону. Кожному градієнту в вікні дескриптора можна приписати три речові координати (x, y, n), де x - відстань до градієнта по горизонталі, y - відстань по вертикалі, n - відстань до напрямку градієнта в гістограмі. За точку відліку береться лівий нижній кут вікна дескриптора і початкове значення гістограми. За поодинокі відрізки беруться розміри регіонів по горизонталі і вертикалі для x і y відповідно, і кількість градусів в компоненті гістограми для n. Коефіцієнт трилінійної інтерполяції визначається для кожної координати (x, y, n) градієнта як 1-d, де d дорівнює відстані від координати градієнта до середини того одиничного проміжку в який ця координата

потрапила. Кожне входження градієнта в гистограмму множиться на всі три вагові коефіцієнта трилинейной інтерполяції.

В алгоритмі даної роботи було вирішено збільшити розмірність дескриптора до 128 компонент (4x4x8). Збільшення кількості точок екстремуму дозволить системі обчислювати результат більш точно, ала водночас не позбавить її швидкодій, що є частиною вимог до проектуємої системи.

3.11 Знаходження евклідової відстані між значеннями дескриптора

Для вирішення задачі ідентифікації та кінцевої обробки даних з нейронної мережі потрібно обчислити Евклідову відстань між дескрипторами ключових на зображеннях. Формула евклідової відстані знаходить відстань між різними двума точками в евклідовому просторі. Евклідів простір описаний аксіомами евклідової геометрії, в даному випадку мається на увазі, що розмірність рівна 3. Приклад відстані евкліда показана на рисунку 3.16.

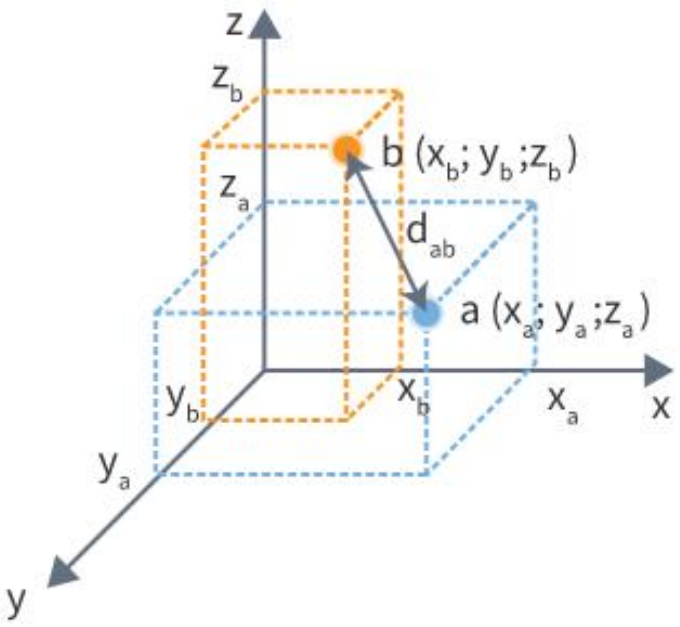


Рисунок 3.16 – Графічне зображення пошуку евклідової відстані та зображення евклідового простору

Евклідова відстань між дескрипторами може бути обчислена на основі виразу:

$$E = \sqrt{\sum_{i=1}^{m} (D_i^1 - D_i^2)^2}, \quad (3.8)$$

В даному виразі: E- евклідова відстань, D – члени двух порівнюваних дескрипторів, m- загальна кількість членів в дескрипторі. Величина евклідової відстані знаходитья в межах $[0; +\infty)$. Чим менша евклідова відстань, тим вища схожість двух порівнюваних дескрипторних масивів.

В процесі вирішення задачі були опрацьовані та описані алгоритми роботи із нейронною мережею. Зокрема в розроблюваній системі будуть використані алгоритми HOG, алгоритм класифікації SVM. Для пошуку ключових точок на обличчі опрацьовано алгоритм AAM. Для нормалізації зображення перед опрацюванням нейронною мережею використано алгиритми афінних перетворень. Для обчислення нейронною мережею ключових точок розглянутий алгоритм SIFT та для підрахунку кінцевого результату використовується розрахунок Евклідової відстані між дескрипторами обличчя.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Розбір ключових функцій

Теоретична база, яка була розгляута в попередніх частинах, дозволяє реалізувати алгоритм розпізнавання та ідентифікації обличчя людини за допомогогою нейронної мережі.

Приклад HOG перетворення:

```
    import os
    import sys
    import dlib
    image = cv2.imread(args.image)
    if image is None:
        print("Could not read input image")
        exit()
    hog_face_detector = dlib.get_frontal_face_detector()
    cnn_face_detector =
dlib.cnn_face_detection_model_v1(args.weights)
    start = time.time()
    faces_hog = hog_face_detector(image, 1)
    end = time.time()
    print("Execution Time (in seconds) :")
    print("HOG : ", format(end - start, '.2f'))
    for face in faces_hog:
        x = face.left()
        y = face.top()
        w = face.right() - x
        h = face.bottom() - y
        cv2.rectangle(image, (x,y), (x+w,y+h), (0,255,0), 2)
```

Для HOG перетворення зображення в даній роботі використовується функція get_frontal_face_detector (). Цей детектор обличчя побудований з використанням гістограми орієнтованих функцій градієнтів в сукупності з лінійним класифікатором, пірамідою зображення і схемою знаходження методом скануючого вікна. Піраміда це зображення, яке представлено в декількох масштабах. В даному випадку використання пірамідальної структури даних при порівнянні зображень дозволить: скоротити час обробки зображень, визначити більш точні початкові приближення для обробки низчих рівнів по результатам обробки вищих рівнів. Піраміда зображень представляє собою

послідовність N зображень, при цьому кожне наступне зображення отримане з попереднього шляхом фільтрації (подавлення високочастотних шумів) та прорідження зображення. Приклад виклику детектора показаний нижче:

```
    import sys
    import dlib
    detector = dlib.get_frontal_face_detector()
    win = dlib.image_window()
    dets = detector(img, 1)
    for f in sys.argv[1:]:
        print("Processing file: {}".format(f))
        img = dlib.load_rgb_image(f)
    for k, d in enumerate(dets):
    print("Detection {}: Left: {} Top: {} Right: {} Bottom:
{}".format(
            k, d.left(), d.top(), d.right(), d.bottom()))
        shape = sp(img, d)
        win1.clear_overlay()
        win1.add_overlay(d)
        win1.add_overlay(shape)
    if (len(sys.argv[1:]) > 0):
        img = dlib.load_rgb_image(sys.argv[1])
        dets, scores, idx = detector.run(img, 1, -1)
        for i, d in enumerate(dets):
            print("Detection {}, score: {},
face_type:{}".format(d, scores[i], idx[i]))
```

В даному коді зображення завантажується і зчитується як масив форми (H,W,C), де Нозначає висоту зображення в пікселях, W – ширину зображення, C – кількість каналів кольору. Перед цим аргумент функції встановлюється в скільки разів потрібно збільшити зображення для полегшення детектування, в данному випадку цей коефіцієнт дорівнює одиниці. Потім виконується розпізнавання обличчя, використовуючи фронтальний детектор обличчя на основі HOG. Другий аргумент в коді виклику детектора говорить про те, скільки раз зображення буде підвержене частковій дискретизації.

В приведеному вище коді функція Enumerate дає змогу провести ітерацію по значенням в самому списку. Функція Shape повертає розмір масиву, його форму. Це кортеж натуральних чисел, які показують довжину масиву по кожній осі. Для матриці з n строк і m стовбців, shape буде (m, n). Число елементів кортежа shape дорівнюватиме рангу масива.

В даній роботі використовується алгоритм класифікації SVM. Даний метод навчений на вибірках зображень та довзоляє знайтина зображенні об-

личчя людини. Вхідними даними до цього класифікатора є дані з гістограми орієнтованих градієнтів. Приклад коду для навчання SVM детектору приведений нижче.

```
     import os
     import sys
     import glob
     import dlib
     from skimage import io
     if len(sys.argv) != 4:
             print(
             "Give the path to the faces directory as the
argument to this "
             "program with training and test xml files in order.
For example: \n"
             "    ./train_object_detector_modified.py ../faces
../faces/training.xml ../faces/testing.xml")
         exit()
     faces_folder = sys.argv[1]
     training_xml_path = sys.argv[2]
     testing_xml_path = sys.argv[3]
     options = dlib.simple_object_detector_training_options()
     options.add_left_right_image_flips = True
     options.C = 5
     options.num_threads = 8
     options.be_verbose = True
     dlib.train_simple_object_detector(training_xml_path,
"detector.svm", options)
     print 'training end'
```

Спочатку відбувається підключення сторонніх бібліотек, потім в даному коді вказується шлях до директорії облич, за допомогою яких буде натренована модель для подальшої класифікації обличчя в даній роботі. Також вказується перевірка XML файлів по порядку. Наступним кроком задаються опції класифікатора, де рядок dlib.simple_object_detector_training_options() є контейнером для опцій функції. Також існує параметр для регуляції. Таким чином він передається в Structure_object_detection_trainer :: set_c (). Великі значення параметру регуляції пробудять тренера краще відповідати даним, але можуть призвести до перенавчання моделі. Щоб запобігти цьому потрібно правильно задати правильні налаштування детектора. Останнім кроком є запуск моделі на навчання. Пілся вдалого закінчення навчання модель класифікатору готова для роботи.

Загалом, після роботи HOG та SVM алгоритмів на виході отримано зображення, на якому знайдено обличчя людини та взято в рамки для обмеження.

В даному коді завантажується навчена модель класифікатору для подальшої роботи. Для прикладу, функціонально викликається функція показу зображення з HOG фільтром, на якому будуть зображені градієнти векторів направлення.

Для вирівнювання зображення та знаходження ключових точок в діній роботі використовуються алгоритми афінних перетворень. Для вирівнювання обличчя з ансамблем дерев регресії для досягнення поставленох мети використовується алгоритм Каземіа та Саллівана, який пройшов навчання на наборі даних наземних орієнтирів. Нижче приведений лістинг коду функції вирівнювання обличчя по параметрам.

```
    def CropFace(image, eye_left=(0,0), eye_right=(0,0),
offset_pct=(0.2,0.2), dest_sz = (70,70)):
        offset_h = math.floor(float(offset_pct[0])*dest_sz[0])
        offset_v = math.floor(float(offset_pct[1])*dest_sz[1])
        eye_direction = (eye_right[0] - eye_left[0], eye_right[1]
- eye_left[1])
        rotation                          =                       -
math.atan2(float(eye_direction[1]),float(eye_direction[0]))
        dist = Distance(eye_left, eye_right)
        reference = dest_sz[0] - 2.0*offset_h
        scale = float(dist)/float(reference)
        image   =   ScaleRotateTranslate(image,   center=eye_left,
angle=rotation)
        crop_xy = (eye_left[0] - scale*offset_h, eye_left[1] -
scale*offset_v)
        crop_size = (dest_sz[0]*scale, dest_sz[1]*scale)
        image   =   image.crop((int(crop_xy[0]),   int(crop_xy[1]),
int(crop_xy[0]+crop_size[0]), int(crop_xy[1]+crop_size[1])))
        image = image.resize(dest_sz, Image.ANTIALIAS)
        return image
```

В даному коді на обличчі знаходяться ключові точки, далі шляхом алгоритму афінних перетворень дані точки переносяться на раніше підготовлену модель, яка скадається з точкой, на яких повинні знаходитися ключові точки з обличчя людини. При цьому об’єкти залишаються в тому ж просторі і положенні в якому були до перетворення. Це означає те, що якщо прямі пертинались, були паралельні, схрещувались до перетворення, то ця особливість

після перетворення зберігається. Також примінюється коефіцієнт точності перетворень. Даний коефіцієнт вказує наскільки точно потрібно здійснити перетворення, при цьому потрібно мати на увазі параметри початкових зображень. Занаднто малий коефіцієнт буде погано працювати на зображеннях з високою роздільною здатністю та навпаки. Результати знаходження обличчя та реалізація алгоритмів нормалізації наведено на рисунку 3.1.

Рисунок 3.1- Результат роботи функцій
знаходження обличчя на зображенні

З рисунку 3.1 видно, що алгоритм досить чітко виявляє рамки обличчя на зображенні, навіть якщо обличчя знаходиться під кутом відносно камери зйомки. Алгоритм не виявляє обличчя, які на зображенні знаходяться в профіль, оскільки при такому положенні обличчя відсутні ознаки для коректної роботи нейронної мережі.

Для знаходження дескриптору обличчя потрібно вірахувати вектор розмірністю 128. Для досягнення мети даної роботи було прийнято рішення збільшити розмірність вектору з 32 до 128. Збільшення кількості точок екстремуму дозволить системі обчислювати результат більш точно, ала водночас не позбавить її швидкодій, що є частиною вимог до проектуємої системи. В загальному випадку дані дескриптори підпорядковані Евклідовій геометрії.

Дію алгоритму SIFT дескриптору детально було розглянуто в попередніх розділах даної роботи. Хоча розмірність дескриптора (тобто 128) здаєть-

ся високою, дескриптори з меншою розмірністю не працюють настільки добре, а обчислювальні витрати залишаються низькими, оскільки для пошуку найближчого сусіда застосовується метод наближеного. Довші дескриптори давали б кращі результати, але не на багато, і є небезпека збільшення чутливості до спотворень і накладенню (перешкодам). Також було показано, що точність зіставлення ознак вище 50% для зміни точки огляду до 50 градусів. Тому SIFT-дескриптори інваріантні малим аффінним змін. Приклад роботи алгоритму знаходження дескрипторів приведено на рисунку 3.2.

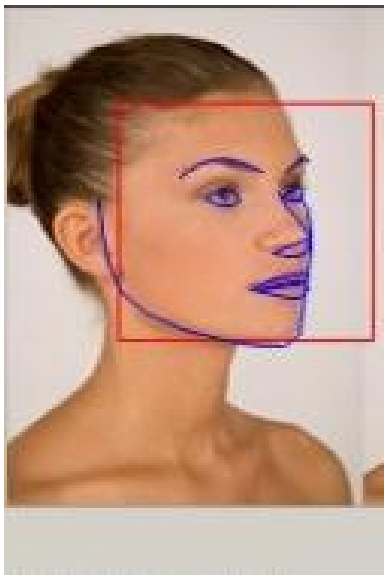


Рисунок 3.2 – Знаходження ключових точок на зображенні

На наступному етапі отримані дескриптори обчислюються, а саме обчислюється евклідова відстань між ними.

Отримавши результат обчислення потрібно правильно його інтерпретувати. В бібліотеці dlib рекомендується використовувати граничне значення евклдіової відстані між дескрипторами зображення менше 0,6. Якщо евклідова відстань менше ніж 0,6, значить фотографії належать одній людині. З ви-

користанням такої метрики dlib гарантує точність в 99,38% на тесті розпізнавання облич Labeled faces in the World [21][1)].

4.2 Тестування розробленої системи

Після завершення роботи по розробці системи для ідентифікації обличчя з використаннями нейронної мережі потрібно провести тесування системи. Тестування дозволить провести дослідження, метою якого є перевірка відповідності між реальною поведінкою програми на кінцевому наборі тестів та знаходження помилок при виконанні коду програми.

Тестування проведено на робочій станції під керуванням операційної системи Windows 10, процесор Ryzen 5 1600 3.2GHz та 8 Гб оперативної пам'яті. Запуск системи на виконання виконується з домопомогою інтерпретатору командної строки Windows, на якій попередньо встановлено Python 3 та необхідні бібліотеки мови python, які входять до стандарттного дистрибутиву та зовнішня бібліотека dlib. Всі рисунки для роботи системи були попередньо завантажені в директорію з початковим кодом для підвищення простоти користування з системою, однак за необхідності зображення можуть знаходитись на будь-якому носієві інформації (зовнішні накопичувачі, мережеві диски або папки). Після вдалого запуску системи потрібно поетапно вказати шлях до зображень відносно місця розташування виконуваного файлу. На першому етапі вводиться шлях до першого зображення та подальша його обробка системою, на другому – вказується шлях до другого зображення та його обробка. На третьому етапі порівнюються дескриптори зображень та робиться висновок про ідентифікацію особи. На рисунку 3.3 показано приклад команди для запуску системи, завантаження першого фото для детектування обличчя та результат виконання.

[1)] [21] Знаходження обличчя за допомогою глибиних нейронних мереж. URL: http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html (Дата звернення 25.11.2019).

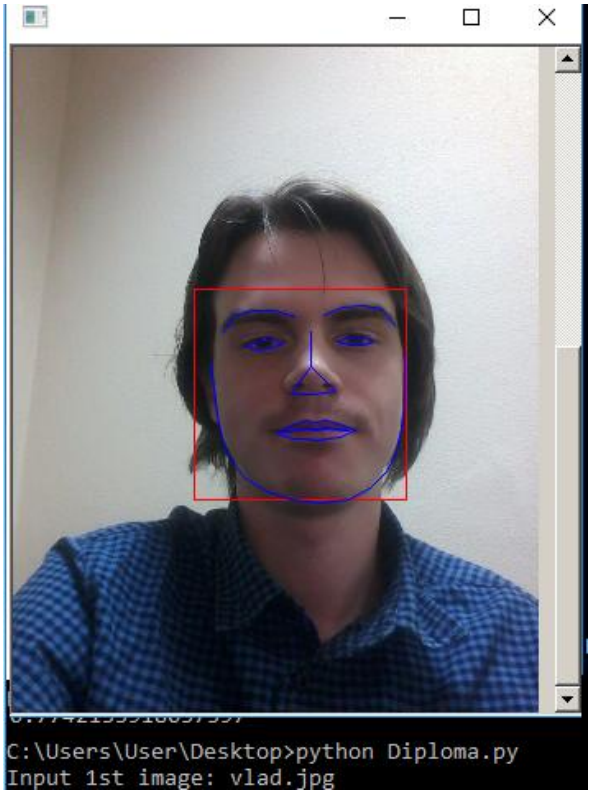


Рисунок 3.3 – Запуск системи та обробка першого фото

Як видно з рисунку 3.3 система успішно заватнажила фото з поточної директорії, чітко виявила на даному фото обличчя людини та встановила дескриптори на обличчі, які відповідають анатомічним рисам (очі, ніс,овал обличчя і т.д.). Значення дескрипторів обличчя наведено в додатку А.

Наступним кроком в роботі системи є завантаження наступного фото та подальша його обробка. На рисунку .3.4 приведено команду для завантаження фото, де вказується шлях до фото та його назва, оскільки дане зображення знаходиться в директорііі з програмою, то в такому випадку не є обов’язковим написання повного шляху до зображення.

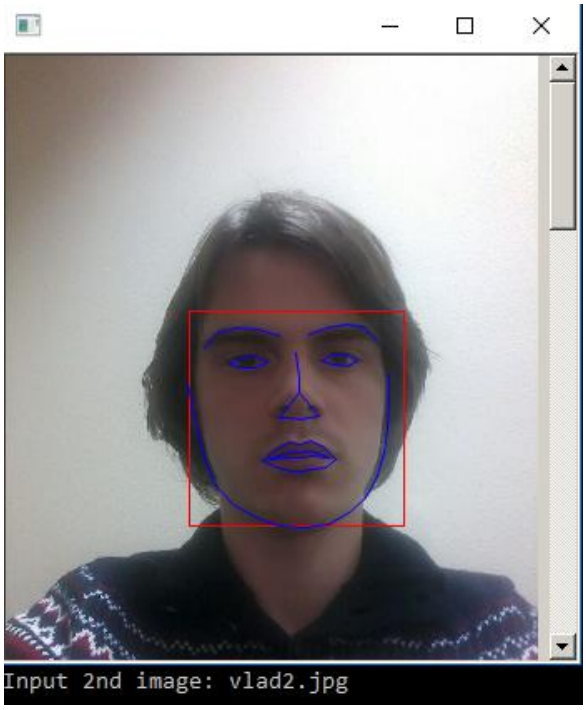


Рисунок 3.4 – Заватнаження другого зображення та його обробка

Як видно з рисунку 3.4 в даному випадку програма теж правильно виділила обличчя на зображенні та досить чітко описала головні дескриптори на ньому. Значення дескрипторів приведено в додатку А.

Наступним кроком в рооботі системи є розрахунок евклідової відстані між дескрипторами двох облич. Результат виконання функції розрахунку евклідової відстані та резульат порівняння двох облич зображено на рисунку 3.5.

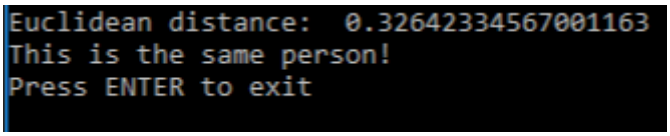


Рисунок 3.5 – Результат порівняння облич

Як видно з рисунку 3.5 евклідова відстань між дескрипторами з округленням до десятичних значень дорівнює 0,3. Виходячи з даного значення си-

стема зробила висновок, що обличчя належить одній і тій же особі,що є правильним результатом, оскільки значення евклідової відстані менше ніж 0,6.

Тепер проведемо порівняння двох різних людей, результати обробки зображень представлені на рисунку 3.6, а дескриптори приведено в додатку Б.

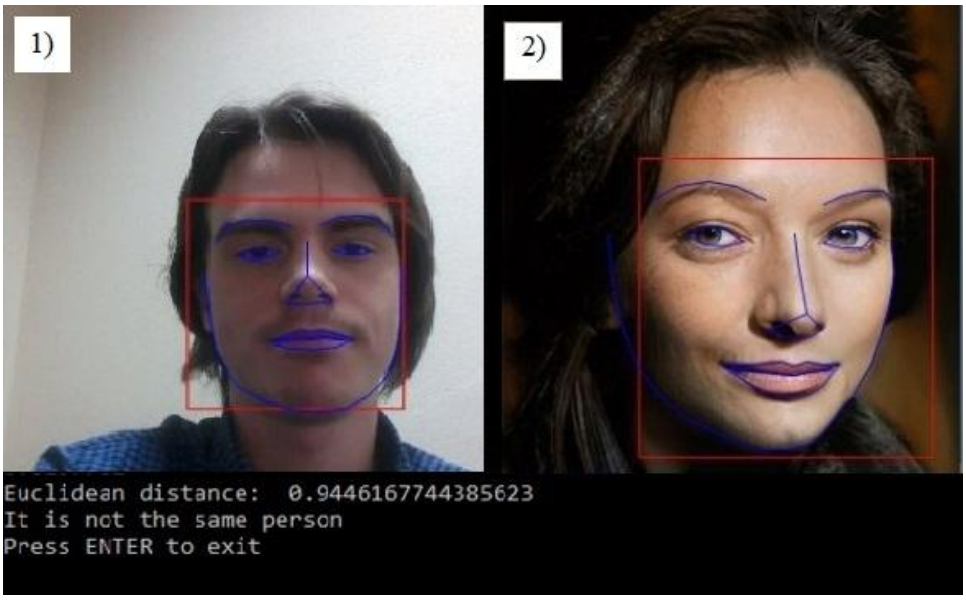


Рисунок 3.6 – Результат обробки фото з різними людьми

Система розразувала евклідову відстань між двома дескрипторами, яка дорівнює 0,9, та зробила висновок що дані зображення облич не належать одній особі. Даний висновок є правильним, оскільки евклідова вістань більше ніж 0,6. Також потрібно звернути увагу на те, що друге зображення обличчя знаходиться під кутом, але алгоритм розпізнавання досить чітко встановив рамки обличчя та виявив дескриптори. З даного результату можна зробити висновок про правильну роботу алгоритму афінних перетворень.

ВИСНОВОК

В результаті виконання магістерської роботи була розроблена система для ідентифікації обличчя засобами нейронної мережі, опираючись на ознаки обличчя. Для здійсненя проектування та розробки системи було проведено аналіз, порівняння та вибір архітектур нейронних мереж для роботи із зображеннями використовуючи дані про точність та швидкість роботи. В результаті було обрано найбільш придатну архітектуру нейронної мережі для вирішення завдання – глибинну мережу ResNet, яка належить до мереж згорткового типу. Застосування мережі ResNet дозволило вирішити поставлені завдання розбробки, отримати якісну швидкодію, зберегти точність результатів системи та мати змогу до подальшої оптимізації роботи системи. Застосування бібліотеки dlib дозволило оптимально використовувати ресурси центрального процесору, при цьому показувати добру швидкодію, а також дала змогу для розширення обчислень шляхом залучення графічного процесору. Поєднання нейронної мережі ResNet, мови Python та бібліотеки dlib дозволяє застосувати систему на широкому спектрі операційних систем та платформ.

Система для ідентифікації обличчя засобами нейронної мережі надає змогу завантажувати зображення облич з будь-якого накопичувача, використовуючи термінали операційних систем. Система дає можливість отримати результат високої точності в заходженні дескрипторів обличчя та надавши висновок про ступінь принадлежнсоті облич одній людині. Завдяки сучасним технологіям розробки система може бути легко розширена та доповнена новими функціями.

У результаті проведеного проектування системи для ідентифікації обличчя засобами нейронної мережі визначена архітектура системи, реалізовані алгоритми детектування, нормалізації, знаходження дескрипторів і їх обробки. Основним результатом проетування є реалізація вимог до системи, побудова її структури та програмна реалізація функціональних можливостей.

ПЕРЕЛІК ПОСИЛАНЬ

1. Нейронні мережі. URL: https://www.asimovinstitute.org/neural network-zoo (Дата звернення 2.10.2019).
2. Сикорский О.С. Обзор сверточных нейронных сетей для задачи классификации изображений. М: МГТУ им. Н.Э. Баумана, 2018. С. 3-8.
3. Структура нейронної мережі ZFNet. URL: https://www.oreilly.com /library/view/hands-on-transfer-learning/9781788831307/ff290260-c4fc-4924-9f30-bcddb36ccf2a.xhtml (Дата звернення: 9.10.2019).
4. Matthew Zeiler, Rob Fergus. Visualizing and understanding convolutional networks. [s.l.] 2016. P. 818-834.
5. Razvan Pascanu, Tomas Mikolov, Yoshua Bengio. Learning to segment object candidates. [s. l]. 2015. P. 1990-1998.
6. Нейронні системі типу Inception. URL: https://towardsdatascien ce.com /a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202 (Дата звернення: 20.10.2019).
7. Порівняльна характеристика нейронних мереж. URL: https://cyberle ninka.ru/article/n/obzor-svyortochnyh-neyronnyh-setey-dlya-zadachi-kl assifikatsii-izobrazheniy (Дата звернення 27.10.2019).
8. LeCun Y. et al. Gradient-based learning applied to document recognition //Proceedings of the IEEE.1998.T. 86. P. 2278–2324.
9. Алгоритми зворотнього розповсюдження помилки. URL: www.irbis nbuv.gov.ua/irbis_nbuv/cgiirbis_64 (Дата звернення: 30.10.2019).
10. Ту Дж., Гонсалес Р. Принципы распознавания образов. М.: «Мир». 1978. С. 34-45.
11. Чабан Л. Н. Теория и алгоритмы распознавания образов. Учебное пособие. М.: МИИГАиК. 2004. С. 70.

12. Lee B. Tarng, Y. S. Application of the discrete wavelet transform to the monitoring of tool failure in end milling using the spindle motor current. International Journal of Advanced Manufacturing Technology. 2009. P. 238–243.
13. Ripley B. D. Pattern recognition and neural networks. Cambridge university press. 2007. P 3-30.
14. Марк Саммерфилд. Python на практике. М.: ДМК Пресс. 2014. С. 338.
15. Діаграми UML. URL: https://prog-cpp.ru/uml-classes/ (Дата звернення 4.11.2019).
16. Методологія DFD. URL: https://sites.google.com/ site/ anisimovkhv/ learning/pris/lecture/tema6/tema6_3 (Дата звернення 4.11.2019).
17. Fitzpatrick, David. The Functional Organization of Local Circuits in Visual Cortex: Insights from the Study of Tree Shrew Striate Cortex. London: Cerebral Cortex. 2016. P. 329-341.
18. Cootes, T. F., Edwards, G. J., Taylor, C. J. Active appearance models Computer Vision – ECCV'98. Lecture Notes in Computer Science. 1998. P. 484.
19. Young, Richard. The Gaussian derivative model for spatial vision: I. Retinal mechanisms. 1987. P. 273–293.
20. Основні положення SIFT. URL: https://opencv-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html (Дата звернення 10.11.2019).
21. Знаходження обличчя за допомогою глибиних нейронних мереж. URL: http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html (Дата звернення 25.11.2019).

ДОДАТОК А

Дескриптори облич однійєї людини

№	Значення №	Значення	Значення №	Значення	Значення №	Значення	Значення №	Значення	Значення №	Значення
	Значення дескрипторів першого фото						Значення дескрипторів другого фото			
1	-0.0479	35	0.001878	69	0.113952	103	-0.07102	1	-0.02475	35	0.082518	69	0.090916	103	-0.04667
2	0.046426	36	0.017918	70	-0.11381	104	0.0424	2	0.07841	36	0.046389	70	-0.15457	104	0.017487
3	-0.04873	37	-0.12215	71	-0.07015	105	-0.16996	3	-0.02338	37	-0.14321	71	-0.0382	105	-0.14983
4	-0.06773	38	-0.08876	72	0.166732	106	0.133418	4	-0.06497	38	-0.06328	72	0.183872	106	0.177711
5	-0.01991	39	0.240389	73	-0.10238	107	0.035265	5	-0.03256	39	0.257766	73	-0.06571	107	-0.00171
6	-0.08523	40	0.025059	74	-0.29551	108	-0.07791	6	-0.07778	40	-0.01662	74	-0.27705	108	-0.08736
7	-0.0347	41	-0.09168	75	-0.26193	109	-0.0993	7	-0.0273	41	-0.06333	75	-0.21637	109	-0.09228
8	-0.05316	42	-0.09879	76	0.068901	110	-0.06984	8	-0.0305	42	-0.03481	76	0.097503	110	-0.01598
9	0.130916	43	0.011676	77	0.378534	111	-0.15789	9	0.202463	43	0.016849	77	0.358474	111	-0.16077
10	-0.03832	44	0.293329	78	0.157751	112	0.006849	10	-0.10165	44	0.294916	78	0.241713	112	0.005505
11	0.253732	45	0.193962	79	-0.14491	113	0.205744	11	0.273303	45	0.157976	79	-0.11982	113	0.196327
12	-0.05166	46	-0.04493	80	0.054439	114	-0.32415	12	-0.06145	46	0.013261	80	0.051733	114	-0.32838
13	-0.21736	47	0.025146	81	-0.08401	115	0.10812	13	-0.20754	47	0.008508	81	-0.03432	115	0.144348
14	-0.06544	48	-0.04668	82	-0.11146	116	0.153678	14	-0.07239	48	-0.08544	82	-0.10884	116	0.152169
15	-0.0351	49	0.091448	83	0.044276	117	0.060443	15	-0.02914	49	0.075399	83	0.050105	117	0.072709
16	0.042327	50	-0.24784	84	0.010206	118	0.193503	16	0.059933	50	-0.23283	84	0.002687	118	0.231264
17	-0.14716	51	0.068639	85	-0.11089	119	0.066482	17	-0.1192	51	0.109378	85	-0.09416	119	0.087115
18	-0.05673	52	0.158043	86	0.09441	120	0.01066	18	-0.05446	52	0.165093	86	0.043739	120	0.055561
19	-0.12891	53	0.080037	87	-0.06751	121	-0.01852	19	-0.09201	53	0.073546	87	-0.05925	121	0.013842
20	-0.13056	54	0.111962	88	0.085023	122	-0.04508	20	-0.15289	54	0.145968	88	0.097034	122	-0.05853
21	0.053288	55	0.112643	89	0.1755	123	-0.17995	21	0.068852	55	0.119838	89	0.14208	123	-0.1439
22	0.039756	56	-0.11286	90	-0.01675	124	-0.02265	22	0.050848	56	-0.11291	90	-0.03661	124	-0.05736
23	0.085769	57	0.050294	91	-0.06699	125	0.12235	23	0.090713	57	0.086905	91	-0.06525	125	0.09069
24	0.110739	58	0.25294	92	0.231185	126	0.037163	24	0.091098	58	0.219301	92	0.237417	126	0.024171
25	-0.21552	59	-0.10948	93	-0.02769	127	0.053984	25	-0.21997	59	-0.13379	93	-0.00239	127	0.111365
26	-0.20918	60	0.012912	94	-0.01468	128	-0.01777	26	-0.22035	60	0.040979	94	0.033446	128	-0.00563
27	-0.0978	61	-0.04503	95	0.12225			27	-0.11158	61	0.00505	95	0.079384		
28	-0.17054	62	0.01917	96	0.004282			28	-0.13041	62	-0.02681	96	-0.00999		
29	-0.04881	63	-0.00135	97	-0.06339			29	0.016808	63	-0.01111	97	-0.10457		
30	-0.14429	64	-0.05771	98	0.042356			30	-0.14116	64	-0.08145	98	0.048895		
31	0.028702	65	0.184079	99	-0.11842			31	-0.00304	65	0.158486	99	-0.11102		
32	0.040458	66	0.03494	100	0.051691			32	0.051026	66	0.02039	100	0.054544		
33	-0.19849	67	-0.11656	101	0.09268			33	-0.17442	67	-0.10706	101	0.087007		
34	-0.1051	68	-0.0738	102	-0.14844			34	-0.13132	68	-0.09857	102	-0.136		

А1 –

деск

ДОДАТОК Б

Дескриптори облич різних людей

Значення дескрипторів першого фото

№	Значення	№	Значення	№	Значення	№	Значення
1	-0.0479	35	0.001878	69	0.113952	103	-0.07102
2	0.046426	36	0.017918	70	-0.11381	104	0.0424
3	-0.04873	37	-0.12215	71	-0.07015	105	-0.1696
4	-0.06773	38	-0.08876	72	0.166732	106	0.133418
5	-0.01991	39	0.240389	73	-0.10238	107	0.035265
6	-0.08523	40	0.025059	74	-0.29551	108	-0.07791
7	-0.0347	41	-0.09168	75	-0.26193	109	-0.0993
8	-0.05316	42	-0.09879	76	0.068901	110	-0.06984
9	0.130916	43	0.011676	77	0.378534	111	-0.15789
10	-0.03832	44	0.293329	78	0.157751	112	0.006849
11	0.253732	45	0.193962	79	-0.14491	113	0.205744
12	-0.05166	46	-0.04493	80	0.054439	114	-0.32415
13	-0.21736	47	0.025146	81	-0.08401	115	0.10812
14	-0.06544	48	-0.04668	82	-0.11146	116	0.153678
15	-0.0351	49	0.091448	83	0.044276	117	0.060443
16	0.042327	50	-0.24784	84	0.010206	118	0.193503
17	-0.14716	51	0.068639	85	-0.11089	119	0.066482
18	-0.05673	52	0.158043	86	0.09441	120	0.01066
19	-0.12891	53	0.080037	87	-0.06751	121	-0.01852
20	-0.13056	54	0.111962	88	0.085023	122	-0.04508
21	0.053288	55	0.112643	89	0.1755	123	-0.17995
22	0.039756	56	-0.11286	90	-0.01675	124	-0.02265
23	0.085769	57	0.050294	91	-0.06699	125	0.12235
24	0.110739	58	0.25294	92	0.231185	126	0.037163
25	-0.21552	59	-0.10948	93	-0.02769	127	0.053984
26	-0.20918	60	0.012912	94	-0.01468	128	-0.01777
27	-0.0978	61	-0.04503	95	0.12225		
28	-0.17054	62	0.01917	96	0.004282		
29	-0.04881	63	-0.00135	97	-0.06339		
30	-0.14429	64	-0.05771	98	0.042356		
31	0.028702	65	0.184079	99	-0.11842		
32	0.040458	66	0.03494	100	0.051691		
33	-0.19849	67	-0.11656	101	0.09268		
34	-0.1051	68	-0.0738	102	-0.14844		

Значення дескрипторів другого фото

№	Значення	№	Значення	№	Значення	№	Значення
1	-0.05923	35	0.071257	69	0.221833	103	0.027557
2	0.037623	36	0.133273	70	-0.26139	104	0.097257
3	0.097108	37	-0.01191	71	-0.08403	105	-0.10941
4	-0.0579	38	-0.08692	72	0.159246	106	0.198178
5	-0.11944	39	0.158304	73	-0.09517	107	-0.02921
6	-0.04981	40	0.000913	74	-0.07981	108	0.04324
7	-0.01652	41	-0.33077	75	-0.33867	109	-0.04939
8	-0.04758	42	-0.05332	76	-0.05304	110	-0.0065
9	0.167318	43	0.077028	77	0.42305	111	-0.0016
10	-0.10078	44	0.246415	78	0.136588	112	-0.03913
11	0.148258	45	0.251488	79	-0.08415	113	0.093829
12	-0.00363	46	-0.04803	80	0.109647	114	-0.22152
13	-0.1171	47	0.000117	81	0.028199	115	0.151791
14	-0.12949	48	-0.04074	82	-0.06465	116	0.108869
15	0.001936	49	0.201909	83	0.013796	117	0.052473
16	0.178168	50	-0.22051	84	0.209552	118	0.136475
17	-0.18886	51	0.012971	85	-0.00184	119	0.021596
18	-0.23464	52	0.185246	86	0.065174	120	0.101363
19	-0.07387	53	-0.00806	87	-0.05065	121	-0.06544
20	-0.07556	54	0.083503	88	0.024222	122	-0.10806
21	0.042304	55	0.07942	89	0.163214	123	-0.18946
22	0.074295	56	-0.10745	90	0.047979	124	-0.04766
23	-0.01593	57	0.040097	91	0.014104	125	0.016385
24	0.099178	58	0.133638	92	0.192614	126	-0.10016
25	-0.17231	59	-0.26963	93	-0.02447	127	0.045104
26	-0.32804	60	-0.03032	94	-0.01806	128	0.01977
27	-0.02314	61	0.012361	95	0.046505		
28	-0.09867	62	-0.19252	96	0.035982		
29	0.079476	63	-0.03213	97	-0.10758		
30	-0.04984	64	-0.13408	98	0.025876		
31	0.043186	65	0.10105	99	-0.1168		
32	0.162224	66	0.076861	100	-0.03084		
33	-0.20487	67	-0.07423	101	-0.06815		
34	0.004873	68	-0.25315	102	-0.02559		

Рисунок Б1 – Значення дескрипторів.