КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

«До захисту допущено»
Завідувач кафедри
Терещенко В. М. _____
(підпис)
«___» _____ 20__ р.

Дипломна робота

на здобуття ступеня бакалавра
за спеціальністю 122 Комп'ютерні науки
на тему:

СТІЙКА ОЦІНКА ГЛИБИНИ ДВОВИМІРНОГО ЗОБРАЖЕННЯ

Виконав студент 4-го курсу
Тара Олександр Миколайович _____
(підпис)

Науковий керівник:
завідувач кафедри математичної інформатики,
професор,
доктор фізико-математичних наук
Терещенко В. М. _____
(підпис)

Засвідчую, що в цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ - 2021

РЕФЕРАТ

Обсяг роботи 44 сторінок, 22 ілюстрацій, 36 джерел посилань.

НЕЙРОННА МЕРЕЖА, КАРТА ГЛИБИНИ, СЕМАНТИЧНА СЕГМЕНТАЦІЯ, ПОВНІСТЮ ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, КЛАСИФІКАТОРИ, РЕГРЕСОРИ, БАГАТОЦІЛЬОВЕ НАВЧАННЯ, ДАТАСЕТ, АРХІТЕКТУРА, ФУНКЦІЯ ВТРАТ, PYTORCH.

Об’єктом розв’язання задачі знаходження глибини двовимірного зображення є карта глибини, а задачі сегментації зображення - класи сегментації та мітки на зображенні, які вказують на розташування класів.

Метою дипломної роботи є запропонувати нову модель, яка дозволить одночасно знаходити карту глибини зображення, а також семантично його сегментувати..

Підстави для виконання: задача пошуку глибини зображення є фундаментальною задачею комп’ютерного бачення, від якої значною мірою залежать алгоритми автомотів - надзвичайно важливої сфери комп’ютерного бачення. Сегментація зображення є першим кроком у задачі генерації видозміненого зображення у реальному часі. Також карта глибини та сегментація зображення можуть бути використані для розробки мобільного додатку.

Результати роботи: виконано загальний огляд відомих підходів до розв’язання поставленої задачі, проаналізовано переваги та недоліки використання різних архітектур нейронних мереж, запропоновано модель, що об’єднує відомі архітектури для пошуку карти глибини та сегментації зображення. Створено новий датасет на основі відомого датасету CelebA-HQ. Реалізовано запропоновану модель на PyTorch та натреновано її.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта дослідження або розробки.

Існує два типи задачі про оцінку глибини зображення: оцінка глибина стереозображень і відео та статичного зображення. У цій дипломній роботі ми розглядаємо задачу оцінки глибини статичного зображення.

Ранні роботи на тему оцінки глибини зображення були зосереджені на оцінці глибини стереозображень на основі обчислювальної геометрії [1, 2], які покладаються на точкові відповідності між зображеннями та триангуляцією для оцінки глибини. Визначною роботою вважають працю Саксена [3], в якій він застосував навчання з вчителем на основі монокулярних ознак двовимірного зображення для визначення глибини. З тих пір пропонується різноманітний підхід з використання монокулярних ознак [4, 5, 6, 7]. Також часто використовуються ймовірнісні графічні моделі, такі як випадкові поля Маркова (MRF), щоб врахувати далекі та глобальні сигнали [4]. Іншим успішним способом використання глобальних сигналів є метод DepthTransfer [11], який використовує глобальні ознаки GIST [8] для пошуку зображень-кандидатів, які "подібні" до зображення у базі даних, що містить RGBD-зображення. На відміну від статичних зображень, роботи на тему оцінки глибини стереозображення та відео мають значний прогрес. [9, 10, 11, 12] Недавні розробки успішно використовували DCNN для вирішення задачі на статичних зображень. [13, 14, 15, 16, 10] Ці методи вирішують задачу статичного зображення шляхом навчання DCNN для оцінки неперервної карти глибини. Оскільки це стандартна задача регресії, як функцію втрат зазвичай приймають середньоквадратичну помилку у логарифмічному просторі або її варіанти.

Актуальність роботи та підстави для її виконання. Оцінка глибини 2D-зображень є важливим етапом реконструкції сцени та вирішення таких завдань як розпізнавання та сегментація 3D-об'єктів. Хоча оцінка глибини на основі стереозображень та відео значно просунулась, оцінка глибини статичних зображень все ще має широке поле для розвитку.

Мета й завдання роботи. Метою дипломної роботи є запропонувати новий ефективний підхід розв’язання задачі знаходження глибини статичного зображення лиця і в комбінації з алгоритмами сегментації 3д зображень виділити частини лиця людини з двовимірного портрету. Для досягнення цієї мети поставлено наступні завдання:

- Дослідити існуючі наукові статті за темою роботи
- Проаналізувати відомі підходи до розв’язання поставленої задачі
- Описати ідею власного розв’язку задачі оцінки глибини двовимірного зображення та його сегментації.
- Запропонувати модель глибинного навчання, що вирішує поставлену задачу.
- Зібрати тренувальні дані для моделі.
- Виконати навчання запропонованої моделі.
- Використати отриману модель для отримання результатів.

Об’єкт, методи й засоби розроблення. Об’єктом розв’язання задачі оцінки глибини статичного зображення та його сегментації є карта глибини двовимірного зображення, а задачі сегментації - набір класів з відповідними мітками, які вказують на розташування класів на зображенні. Методом розв’язання поставленої задачі є використання навченої моделі, яка дозволить визначити глибину зображення, а також сегментувати його. Робота виконується за допомогою Python 3.8 і Pytorch. Серед допоміжних засобів можна згадати бібліотеки numpy, matplotlib, PIL, scipy.

Можливі сфери застосування.

Проблема отримання об’ємного зображення з двовимірного відома уже десятки років. Серед можливих сфер застосування методу знаходження глибини двовимірного зображення варто згадати задачу семантичної сегментації зображення, яку також розглянуто в цій роботі. Глибина зображення дозволяє краще зрозуміти особливості об’ємної сцени і зрозуміло використовується в проблемах комп’ютерного бачення: автомотонах та

доповненій реальності. Визначення глибини статичного зображення також допомагає для вирішення суміжної задачі - визначення глибини відео чи стереозображення. Разом з задачею сегментації, глибина зображення може допомогти у задачі видозміни лиця в реальному часі.

Взаємозв'язок з іншими роботами. Опорними дослідженнями, які зробили найбільший внесок в цю роботу є [32, 34]. Також ключовим було використання датасету [31]. Цікавим є можливе розширення даної роботи за допомогою [36], а також написання мобільного додатку, який дозволить користувачам спробувати запропоновану модель на собі.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Постановка задачі.

Оцінка глибини зображення є вирішальним кроком на шляху до отримання геометричних ознак сцени з 2D-зображень. Метою монокулярної оцінки глибини є прогнозування значення глибини кожного пікселя з урахуванням лише одного зображення RGB. Маючи метод знаходження глибини зображення використати його для вирішення задачі семантичної сегментації зображення.

1.2. Загальні означення.

В ході виконання поставленого завдання необхідно побудувати карту глибини зображення (depth map).

У 3D-комп’ютерній графіці та комп’ютерному зорі карта глибини - це зображення або канал зображення, що містить інформацію, що стосується відстані поверхонь об’єктів сцени від точки спостереження. Цей термін пов’язаний із буфером глибини, Z-буфером, Z-буферизацією та Z-глибиною та може бути аналогом. "Z" у цих термінах відноситься до домовленості про те, що центральна вісь огляду камери знаходиться в напрямку осі Z камери, а не до абсолютної осі Z сцени.

Методом отримання карти глибини зображення є нейронна мережа, яка повертає карту глибини.

Штучний алгоритм навчання нейронної мережі, або просто нейронна мережа - це обчислювальна система навчання, яка використовує мережу функцій для розуміння та переведення введених даних однієї форми у бажаний результат, як правило, в іншій формі. Концепція штучної нейронної мережі була натхненна людською біологією та тим, як нейрони людського мозку функціонують разом, щоб зрозуміти вхідні дані отримані від людських органів чуття.

Нейронні мережі - лише один із багатьох інструментів та підходів, що використовуються в алгоритмах машинного навчання. Сама нейронна мережа

може використовуватися як одна частина в багатьох різних алгоритмах машинного навчання для переведення складних вхідних даних у простір, який можуть зрозуміти комп'ютери.

Нейронні мережі сьогодні застосовуються до багатьох реальних проблем, включаючи розпізнавання мови та зображень, фільтрування спаму, електронну пошту, фінанси та медичну діагностику.

Алгоритми машинного навчання, які використовують нейронні мережі, як правило, не потребують програмування з певними правилами, які визначають, чого очікувати від вхідних даних. Натомість алгоритм навчання нейронних мереж навчається на обробці багатьох маркованих прикладів (тобто даних із «відповідями»), які подаються під час навчання, і використовуючи цей ключ відповіді, щоб дізнатись, які характеристики вхідних даних необхідні для побудови правильного результату, це так зване навчання з вчителем. Після обробки достатньої кількості прикладів нейронна мережа може почати обробляти нові, невідомі вхідні дані та успішно повертати точні результати. Чим більше прикладів та різноманітності вхідних даних бачить програма, тим точнішими стають результати, оскільки програма навчається з досвідом. [17]

1.3. Згорткові нейронні мережі

Архітектури нейронних мереж, що показали найкращі результати в оцінці глибини зображення базувались на згорткових нейронних мережах (CNN, ConvNet), тому важливо детально розуміти це поняття.

CNNs – це категорія нейронних мереж, що зарекомедувала себе як ефективна нейронна мережа в сфері розпізнавання та класифікації зображень. ConvNets успішно проявила себе в ідентифікації облич, об’єктів і дорожніх знаків не кажучи про забезпечення зору роботів та безпілотників.

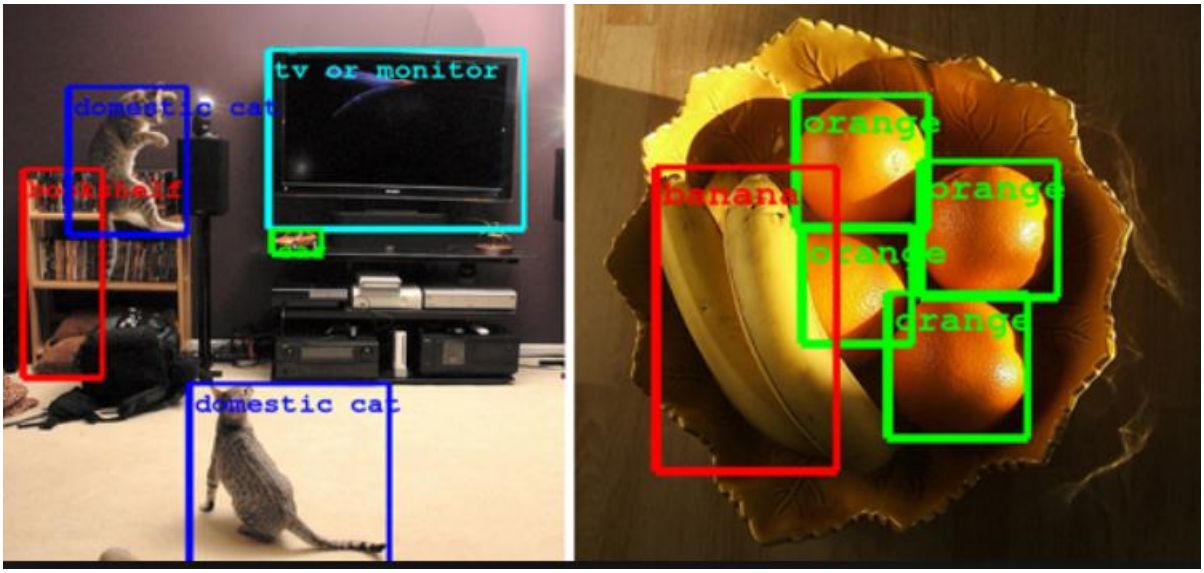

рис 1.

LeNet, розроблена Yann LeCun, була однією з перших згорткових нейронних мереж, яка допомогла розвинути сферу глибинного навчання [18]. Згорткова нейронна мережа на рис. 1 схожа за архітектурою до LeNet і класифікує вхідне зображення за певними сталими категоріями. Коли нейронна мережа отримує на вхід зображення, то на виході повертає ймовірність попадання зображення в певну категорію. Найбільшу ймовірність отримує та категорія, яка на думку нейронної мережі є представлена на вхідному зображенні.

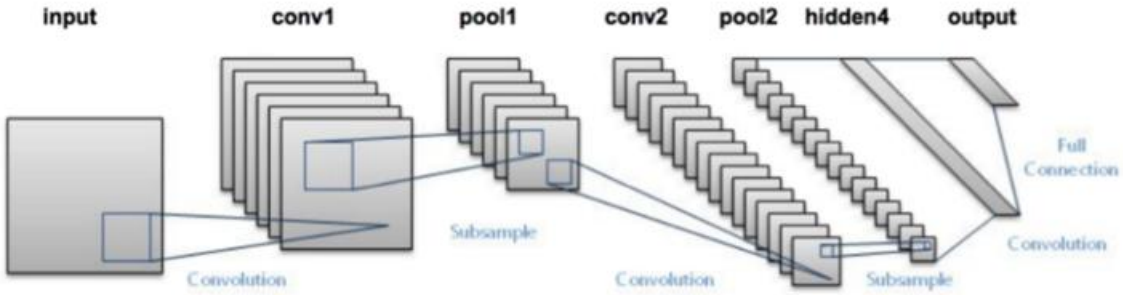

Рис 3.

Є чотири основні операції в ConvNet, які показані на рис. 3

- Convolution
- Non Linearity (ReLU)
- Pooling or Sub Sampling

- Classification (Fully Connected Layer)

Ці операції є фундаментом будь-якої згорткової нейронної мережі.

Канал [19] – це загальноприйнятий термін, який використовується щоб звернутись до певного елементу зображення. Зображення зі стандартної цифової камери матиме три канали – червоний, зелений і синій.

Крок згортки [20]

Основна мета згортки у випадку ConvNet – виділити особливі риси з вхідного зображення. Згортка зберігає просторові зв'язки між пікселями за допомогою вивчення особливих рис зображення, використовуючи невеликі квадрати вхідних даних.

Як згадано вище, кожне зображення можна вважати матрицею значень пікселів. Розглянемо зображення $5 * 5$, чиї пікселі можуть набувати значень 0 і 1 (хоча для чорно-білого зображення значення бувають від 0 до 255, зелена матриця внизу є особливим випадком, де пікселі набувають значень 0 та 1):

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1

Таблиця 1.

Також, розглянемо іншу матрицю $3 * 3$, як показано вищеmatrix. Тоді згортка матриць зображень $5 * 5$ та $3 * 3$ може бути обчислене як почергове знаходження суми по профілю (поелементне множення елементів матриць

$3 * 3$ і $5 * 5$ і подальше знаходження суми по-елементних добутків, яка формує значення 1 клітинки у вихідній матриці) матриці $3 * 3$ у матриці $5 * 5$. Кожного разу профіль матриці $3 * 3$ зсувається вправо на одну позицію. В даному випадку результатом згортки буде матриця $3 * 3$, яка показана у Таблиці 2.

4	3	4
2	4	3
2	3	4

Таблиця 2

В термінології CNN, матриця $3 * 3$ називається фільтром, ядром або детектор особливостей, а матриця, утворена ковзанням фільтру по зображенню і обчисленням точкового результату, називаєтсья згорнена особливість або картою активації. На Рис. 4 показано результат застосування згортки для реальної картинки із певним фільтром.

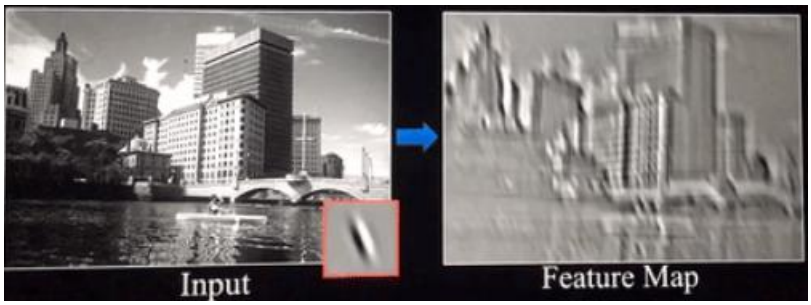

Рис. 4

Важливо відзначити, що етап згортки охоплює локальні залежності в оригінальному зображенні. На практиці CNN засвоює значення цих фільтрів самостійно в процесі навчання (хоча нам як і раніше необхідно вказати такі параметри, як кількість фільтрів, розмір фільтра, архітектура мережі і т.д. до початку процесу навчання). Чим більше фільтрів, тим більше особливостей зображень витягуються і тим краще наша мережа стає у розпізнаванні образів в довільних зображеннях. При роботі з багатовимірними вхідними даними, такими як зображення розмірності входів, таких як зображення, як видно вище, непрактично з'єднувати нейрони зі всіма нейронами попереднього рівня.

Замість цього ми будемо підключати кожен нейрон тільки в локальній області вхідного рівня. Просторова протяжність цього зв'язку є гіперпараметр, що називається сприйнятливе поле.

Розмір карти особливостей контролюється трьома параметрами, які ми визначаємо перед етапом згортки:

- Глибина: Глибина відповідає кількості фільтрів, які ми використоуємо для операції згортки. У мережі, показаній на рис 4., ми виконуємо згортку зображення човна з використанням трьох різних фільтрів, таким чином, створюючи три карти особливостей, як показано на малюнку. Можна вважати ці три карти особливостей stacked двовимірними матрицями, а тому, глибина карти особливостей буде три.

Рис. 4

- Крок: крок– це кількість пікселів, на яку ми ковзаємо фільтром по вхідній матриці. Якщо крок рівний 1, фільтр пересувається на один піксель. Коли крок рівний 2, фільтр пересувається на 2 пікселя за одне ковзання. Більший крок буде створювати менші карти особливостей.
- Доповнення нулями: вхідна матриця доповнюється нулями на межі зображення, щоб можна було використати фільтр для елементів на межі матриці. Хорошою властивістю zero padding є можливість контролю розміру матриці особливостей. Додавання zero-padding також називається широкою згорткою, а не додавання – вузька згортка.

Розширені згортки.

Останні розробки включають в себе введення ще одного гіперпараметру, що називається розширення [21]. Досі ми розглядали тільки суміжні фільтри

CONV. Проте можливо мати фільтри, в яких є вільне місце між кожною клітинкою. Вони називаються розширення. Як приклад, в одному вимірі фільтр w розміру 3 для входу x застосується так:

w[0]*x[0] + w[1]*x[1] + w[2]*x[2].

Це розширення для 0. Для розширення 1 фільтр працюватиме так:

w[0]*x[0] + w[1]*x[2] + w[2]*x[4];

Іншими словами, існує розрив розміру 1 між використаннями. Це може бути дуже корисно в деяких випадках для використання в поєднанні з 0-розширеними фільтрами, оскільки вона дозволяє об'єднати просторову інформацію вхідних даних набагато агресивніше з меншою кількістю шарів. Наприклад, якщо скласти два 3x3 CONV шари, то ви можете бути впевнені, що нейрони на 2-му шарі є функцією 5x5 (ефективне сприйнятливе поле цих нейронів - 5x5). Якщо ми будемо використовувати розширені згортки, то це ефективне сприйнятливе поле буде збільшуватись набагато швидше.

Нелінійність

Нелінійність – це поелементна інформація (застосована для кожного пікселя), що заміняє значення кожного пікселя на інше згідно з вибраною нелінійною функцією. Зараз найбільш часто вживана нелінійність для згорткових шарів є ReLU [6] та її модифікації. Метою нелінійності є введення нелінійності в нашому ConvNet, так як більша частина реальних даних, на яких ConvNet повинна навчатись буде нелінійним (згортка є лінійної операцією - поелементно матричне множення і додавання, тому ми враховуємо нелінійність, вводячи нелінійну функцію ReLU).

Пулинг (Pooling)

Пулинг зменшує розмірність кожної карти особливостей, але зберігає найважливішу інформацію. Пулинг може бути різних типів: Max, Average, Sum і т.д.

У випадку Max Pooling, ми визначаємо просторову околицю (наприклад область 2×2) і вибираємо найбільший елемент з матриці особливостей в цій області. Замість того, щоб вибирати найбільший елемент, можна також вибрати середній (Average Pooling) або суму всіх елементів в цій області. На практиці, Max Pooling зарекомендувало себе найкраще. На таб. 3 показано приклад роботи Max Pooling на Rectified Feature map (отриманої після операції згортки та операції ReLU), використовуючи область 2 × 2.

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

Таб. 3

Функція пулингу використовується для прогресивного зменшення просторового розміру представлення вхідних даних. Зокрема, pooling:

Робить представлення вхідних даних меншим і легшим в керуванні.

Зменшує кількість параметрів і обчислень в мережі, а тому контролює перенавчання [7].

Робить мережу стійкою до малих трансформацій, спотворень та переміщень у вхідному зображенні (невелике спотворення у вхідному зображенні не змінить вивід функції пулингу, оскільки ми вибираємо максимальне/середнє значення в околиці).

допомагає перейти до практично стійкого до зміни розширення представлення зображення. Це дуже важливо, оскільки ми можемо виявити об'єкти в зображенні незалежно від того, де вони знаходяться [8].

Відмова від пулингу.

Деякі дослідники вважають, що без операції пулингу можна обійтись. Наприклад Striving for Simplicity: The All Convolutional Net пропонує відкидати шар пулингуна користь архітектури, що складається лише з шарів згортки. Для того, щоб зменшити розмір представлення, вони пропонують використовувати більший крок в CONV шарі. Відкидання pooling шарів також є важливими для навчання генеративних моделей, такі як варіаційні автоассоціатори (VAEs) або generative adversarial networks (Gans).

1.4 Пакетне навчання

Для навчання нейронних мереж використовується стохастичний градієнтний спуск із можливими модифікаціями, що направлені на покращення сходження в напрямку глобального мінімуму. Серед таких методів найбільш дієвими є momentum, nesterov momentum, AdaGrad, AdaDelta, RMSPROB, ADAM. Основною спільною рисою цих методів є те, що вони не рахують градієнт по всіх навчальних даних, а беруть тільки випадково вибрану частину вибірки. Ця випадкова частина вибірки називається пакетом. Також є таке поняття як епоха навчання. Епохою навчання закінчується тоді, коли кожен елемент вибірки попав у якийсь із пакетів, і був прорахований в одній ітерації пакетного навчання. У порівнянні із класичним стохастичним градієнтним спуском, пакетний стохастичний градієнтний спуск менш схильний до випадкового блукання в напрямках часткових градієнтів. Напрямок пакетного градієнту ближчий до градієнту, порахованого по всій тренувальній виборці. Нагадаю, що градієнт по всій тренувальній вибірці порахувати поки неможливо через обмеженість обчислювальних ресурсів системи. Зокрема, не вистачає пам’яті для того, щоб розмістити зразу весь датасет і порахувати градієнт.

1.5. Поняття передачі навчання та сценарії його застосування

На практиці, не багато дослідників тренують Convolutional Network з чистого листа(з випадковою ініціалізацією ваг), тому що це потребує набору

даних достатнього розміру, що у більшості випадків - рідкісне явище. Натомість, загально прийнятою методикою є попереднє тренування ConvNet на дуже великих обсягах інформації(наприклад, ImageNet, який містить 1,2 мільйони зображень з 1000 категорій), і потім їх використання у якості ініціалізації або як fixed feature extractor для потрібного завдання. Існують наступні сценарії передачі навчання [22]:

- ConvNet як fixed feature extractor. У ConvNet тренованому на ImageNet видаляють останній шар, потім його використовують як фіксований feature extractor для нового набору даних. В AlexNet це використовують як вектор розмірності 4096 для кожного зображення яке містить активацію для прихованого шару перед використанням класифікатора (Linear SVM або класифікатора Softmax).

- Fine-tuning ConvNet. Другою стратегією використання є не тільки заміна класифікатора на вершині ConvNet, але також і для налаштування попередньо натренованої мережі за допомогою backpropagation. Також є можливим налаштування усіх шарів ConvNet, або можливо зберегти декілька з ранніх шарів зафіксованими (для того щоб запобігти перенавчанню) і тільки налаштувати вищі рівні мережі. Це вмотивовано спостереженнями, які показують, що нижні рівні ConvNet містять більш загальні ознаки, які можуть бути корисними для багатьох завдань, але вищі рівні ConvNet більше зосереджені на деталях конкретного набору даних. У випадку ImageNet, який містить багато порід собак, більша частина презентаційного рівня ConvNet націлена на ознаки, які специфічні під час розрізнення порід собак.

1.6. Використання тренованих моделей

Так як ConvNet потребує 2-3 тижня на тренування на декількох GPU на ImageNet, дослідники часто викладають у відкритий доступ свої фінальні ConvNet мітки для інших розробників, які можуть використати ці мережі для власних потреб. Наприклад, бібліотека від Caffe має Model Zoo, де люди діляться вагами для своїх мереж. Постає питання того, коли потрібно

використовувати зовнішні налаштування і який тип передачі навчання використати. Це залежить від багатьох факторів, але можна виділити два основних: розмір набору даних і його схожість на початкових датасет(наприклад, ImageNet в термінах змісту і класів). Беручи до уваги, що ознаки ConvNet більш загальні на нижніх рівнях і більш специфічні на пізніх, наведемо чотири загальні сценарії[22]:

1. Новий набір даних невеликий і схожий до початкового. Так як даних небагато, то використання налаштувань для ConvNet не є гарним вибороб через загрозу перенавчання. Так як дані схожі , то очікувано, що ознаки в вищих рівнях ConvNet будуть релевантними для цього датасету також. Тому найоптимальнішим виходом буде тренування лінійного класифікатора по CNN codes.

2. Великий новий набір даних, схожий до початкового. Так як наявно більше даних, є впевненість у тому, що перенавчання не відбудеться, якщо ми використаємо налаштування по всій мережі.

3. Невеликий новий набір даних, який сильно відрізняється від початкового. Так як даних небагато, найкращим виходом є тренування тільки лінійного класифікатора. Через своєрідність даних, краще тренувати класифікатор не з верхівки мережі, яка містить більш специфічні ознаки. Натомість, можливо використати SVM класифікатор, активувавши його десь на ранніх рівнях мережі.

4. Новий набір даних досить великий і зовсім не схожий на початковий. Унаслідок розміру даних, можливо припустити доречність тренування ConvNet з нуля. Хоча, на практиці ініціалізація ваг вже з тренованої моделі дає свої переваги. В цьому випадку, у нас є достатньо даних і впевненості для налаштувань усієї мережі.

1.7. Класифікатори

Метою цієї роботи є одночасне створення карти глибини зображення та його сегментації. В задачі знаходження глибини ми передбачаємо значення

глибини кожного окремого пікселя, а тому це проблема регресії. Проблема сегментації ж - це типовий приклад задачі класифікації з декількома класами.

Проблеми класифікації, що мають кілька класів із незбалансованим набором даних, представляють цілковито інший виклик, ніж проблема бінарної класифікації. Нерівномірний розподіл робить багато звичайних алгоритмів машинного навчання менш ефективними, особливо при прогнозуванні прикладів класів з невеликої кількістю представників. Тому варто спочатку зрозуміти проблему, а потім обговоримо шляхи її подолання

Багатокласова класифікація: Класифікаційне завдання з більш ніж двома класами; наприклад, класифікувати набір зображень фруктів, які можуть бути апельсинами, яблуками чи грушами. Багатокласова класифікація робить припущення, що кожен екземпляр належить одному і тільки одному класу: фруктом може бути як яблуко, так і груша, але не обидва одночасно.

Незбалансований набір даних: Незбалансовані дані є поширеною проблемою в задачах класифікації. Наприклад, у вас може бути проблема класифікації 3 класів набору фруктів, які класифікуються як апельсини, яблука або груші із загальним числом 100 екземплярів. Загалом 80 екземплярів позначено Класом-1 (апельсини), 10 екземплярів Класом-2 (Яблука), а решта 10 екземплярів позначено Класом-3 (Груші). Це незбалансований набір даних і співвідношення 8: 1: 1. Більшість наборів даних для класифікації мають не однакові кількості екземплярів у кожному класі, але невелика різниця часто не має значення. Існують задачі, для яких дисбаланс класів не просто поширений, а очікується. Наприклад, у наборах даних, подібних до тих, що характеризують шахрайські транзакції, є незбалансованість. Переважна більшість транзакцій відбуватиметься у класі "Не-шахрайство", і дуже невеликий відсоток - у класі "Шахрайство".

Підходи до вирішення задачі багатокласової класифікації.

Приведення до бінарної класифікації.

1. Стратегія один проти інших: (OvR або один проти всіх, OvA) передбачає підготовку одного класифікатора для кожного класу, причому екземпляри цього класу є позитивними зразками а всі інші екземпляри негативні. Ця стратегія вимагає, щоб базові класифікатори обчислювали коефіцієнт довіри для свого рішення, а не лише ярлик класу; окремі ярлики класів можуть призвести до неоднозначностей, коли для однієї вибірки передбачається декілька класів. [23]
2. Стратегія один проти одного (OvO) розробляє K (K - 1) / 2 бінарних класифікаторів для багатокласової задачі на K класів; кожен отримує зразки пари класів з оригінального навчального набору і повинен навчитися розрізняти ці два класи. Під час прогнозування застосовується схема голосування: усі класифікатори K (K - 1) / 2 застосовуються до вибірки, а клас, який отримав найбільшу кількість прогнозів "+1", прогнозується комбінованим класифікатором. [23] Як і OvR, OvO має недолік неясності, оскільки деякі класифікатори його вхідного простору можуть набрати однакову кількість голосів. [23]

Нейронні мережі.

Багатокласні персептрони є природнім продовженням проблеми багатокласової класифікації. Замість того, щоб просто мати один нейрон у вихідному шарі, що має двійковий вихід, можна було б мати N двійкових нейронів, що і веде до багатокласової класифікації. На практиці останнім шаром нейронної мережі, як правило, є функціональний шар softmax, що є алгебраїчним спрощенням N логістичних класифікаторів, нормованих на кожен клас за сумою N-1 інших логістичних класифікаторів.

Помилка класифікатора

Мета обчислення частоти помилок полягає в тому, щоб якомога краще передбачити правильну позначку нових екземплярів. Для цього нам потрібно визначити, в чому полягає помилка нашої прогностичної моделі. Ми також

припускаємо, що існує "ідеальна" модель f: X → Y така, що $y_i = f(x_i)$, ця функція невідома, і насправді це те, що ми намагаємось навчитися робити. Для функції прогнозування h ми визначаємо помилку класифікатора за ймовірністю обрання випадкового прикладу з розподілу D такого, що h (x) != f (x). Ця помилка називається помилкою узагальнення.

Ще однією корисною помилкою є помилка навчання, це помилка функції прогнозування на прикладах, які ми маємо у своєму розпорядженні. Оскільки вони є репрезентативними для розподілу D, природним способом знайти хорошу функцію прогнозування є мінімізація цієї помилки в навчанні. Цей теоретичний підхід називається емпіричною мінімізацією ризиків (ERM) і призводить до проблем оптимізації, які часто є опуклими проблемами оптимізації. [24]

Надмірне навчання

Спроба мінімізувати помилку в навчанні будь-якою ціною може призвести до того, що називається перенавчанням. Це означає, що наша функція прогнозування має дуже низьку похибку у навчанні, але не узагальнюється за межами вибірки. Ми занадто близькі до прикладів, і тому ми втрачаємо інформацію про загальний розподіл. Наприклад, давайте розглянемо таку проблему класифікації, де було знайдено досить добрий розподіл: Рисунок 5. Прикладом перенавчання цих даних може бути: Рисунок 6. На противагу перенавчанню є те, що називається недостатнє навчання, коли ви не отримуйте багато інформації з ваших прикладів: Рисунок 7

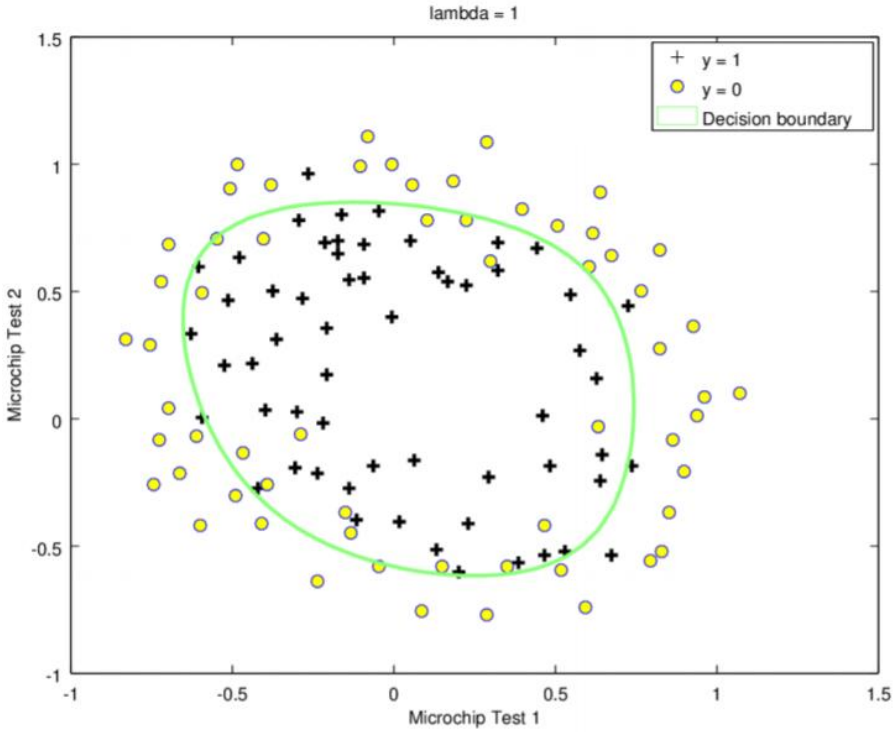

Рис. 5 Нелінійна класифікація

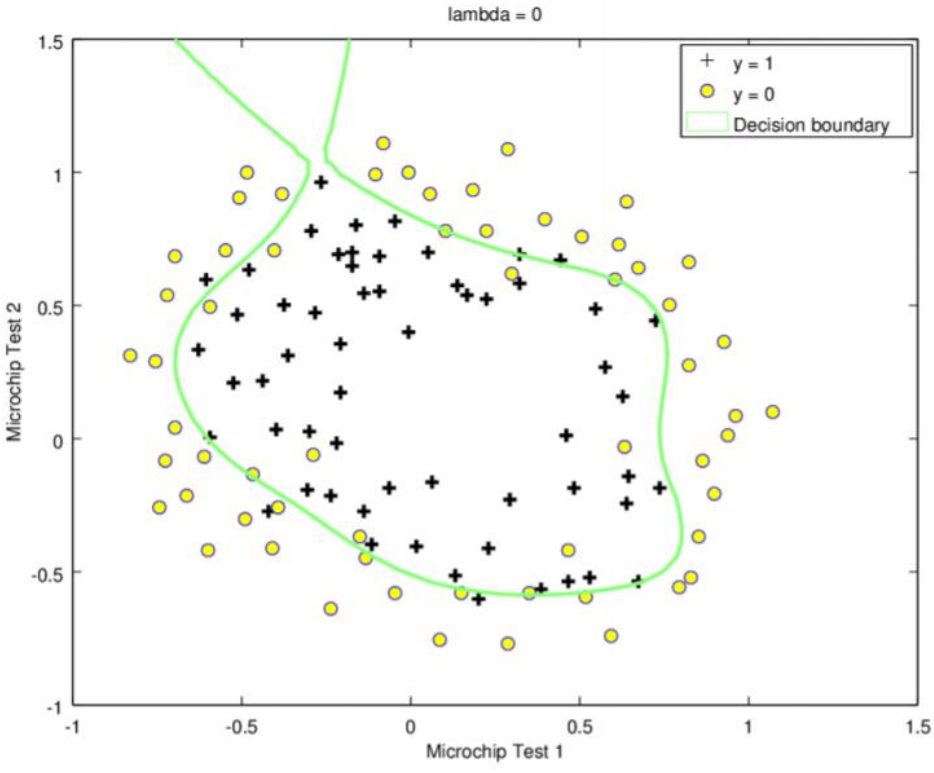

Рис. 6 Надмірне навчання

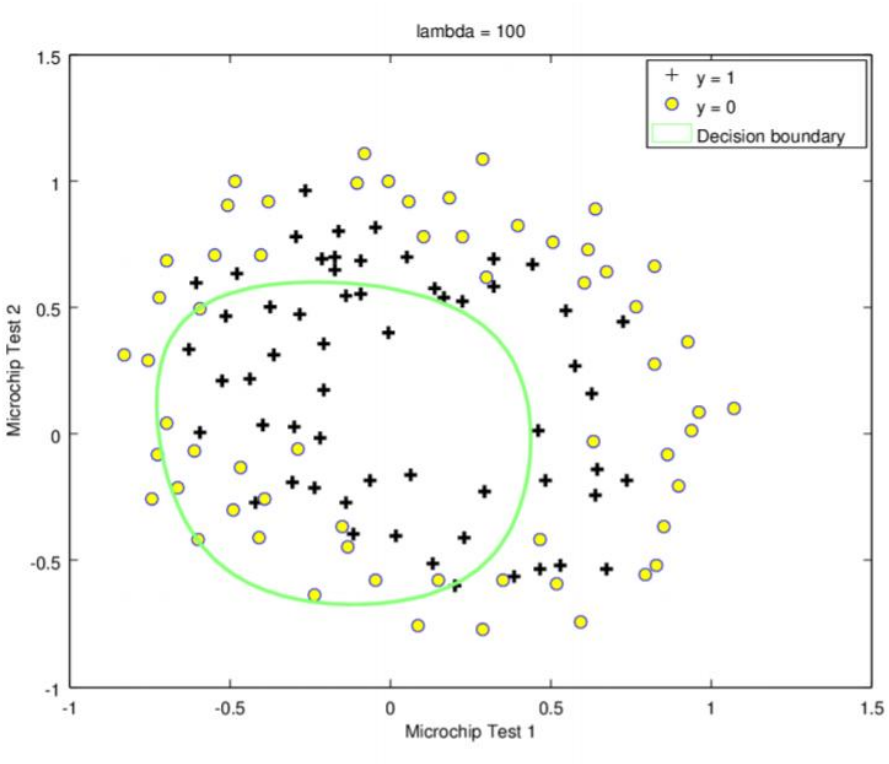

Рис. 7 Недостатнє навчання.

Для запобігання надмірному навчанню класичним підходом є додавання терму регуляризації до помилки з подальшою мінімізацією цільової функції. Наприклад, помилка, яку ми намагаємося мінімізувати у випадку регресії, часто є нормою L1 або L2, що означатиме таку помилку:

$$E = \sum_{i=1}^{n} |f(x_i) - h(x_i)|, \text{ або}$$

$$E = \sum_{i=1}^{n} (f(x_i) - h(x_i))^2$$

Наша функція h(x) буде залежати від m параметрів W_i, які ми хочемо визначити і які можна записати $h(x, W) = \sum_{i=1}^{m} h_i W_i$.

Після того як ми додамо терм регуляризації, який часто називають L1 регуляризацією або L2-регуляризацією наші дві функції помилок стають:

$$E = \sum_{i=1}^{n} |f(x_i) - h(x_i)| + \lambda \sum_{i=1}^{n} |W_i|,$$

$$E = \sum_{i=1}^{n} (f(x_i) - h(x_i))^2 + \lambda \sum_{i=1}^{n} W_i^2$$

Потрібно визначити параметр λ, який називається параметром регуляризації. При занадто низькому λ ми, швидше за все, матимемо перенавчання, а при занадто високому λ ми отримаємо недостатнє навчання.

1.8. Багатоцільове навчання

У багатоцільовому навчанні ми прагнемо вирішити кілька завдань машинного навчання одночасно. Це означає, що функція, яку ми намагаємось знайти h: X → Y_1 × ... × Y_t, насправді є вектором функцій: h = (h_1, ..., h_t), де t - кількість задач, які потрібно вирішити, і h_i : X → Y_i. Проблеми, які нам потрібно вирішити, не обов'язково однотипні, і тому ми можемо мати Y_i = R та Y_j = {0, 1, 2, 3}.

Наприклад, в Інтернет-рекламі, коли компанія хоче передбачити цікавість клієнта до своєї продукції, між продуктами виникатимуть залежності. Якщо його цікавлять ракетки для тенісу, він, ймовірно, також зацікавиться тенісними м’ячами. Ідея сучасного підходуполягає у використанні залежностей, що стоять за завданнями, для підвищення точності моделей. Уявімо, що у нас є пацієнт, який нещодавно переніс інфаркт. Ми зібрали дані про нього (артеріальний тиск, рівень інсуліну, ...) і, можливо, ми хочемо передбачити, що стало причиною недавнього серцевого нападу, а також коли відбудеться наступний. Одним із способів зробити це може бути вирішення проблеми класифікації для виявлення причини серцевого нападу, а потім проблеми регресії окремо для передбачення часу наступного випадку. Але такий підхід не дозволяє отримати спільну інформацію про дві проблеми, і ми можемо думати, що інформація одного завдання може допомогти іншому.

Тому ще одним способом зробити це було б вирішення першого завдання, а потім вирішити друге з інформацією з першого. Ми могли б мати такі функції: h1 (x) = y1, h2 (x, y1) = y2. Але знову ж таки, це не зовсім задовільно, оскільки ми не зможемо ділитися всією інформацією між завданнями.

Незважаючи на те, що деякі результати показали, що багатоцільове навчання може перевершити одноцільове навчання [25], вони стосуються двох завдань

регресії та ранжування, які мають багато спільного та не можуть бути застосовані, наприклад, до регресії та класифікації. В даний час не існує єдиної основи для багатоцільового навчання.

Багатозадачне навчання

У процесі багатозадачного навчання метою є реалізація кількох однотипних завдань (наприклад, множинні класифікації або множинні регресії) одночасно. Уявімо, що у нас є два завдання машинного навчання T1 і T2. Нам потрібно буде звести до мінімуму E1 + E2, де дві функціональні помилки, пов'язані із цими завданнями:

$E_1 = L_1(x, W_1) + \lambda_1 R_1(W_1)$ і $E_2 = L_2(x, W_2) + \lambda_2 R_2(W_2)$,

де R_i (W_i) - терм регуляризації для завдання i, а L_i (x, W_i) - помилка для завдання i. Спочатку ми могли вирішити завдання 1 і отримати параметр W_1, вибравши фіксований (випадковий) W_2, а потім за допомогою цього щойно знайденого W_1 ми змогли мінімізувати помилку і знайти W_2. Ці два завдання вирішуватимуться окремо.

Метою багатозадачного навчання є можливість врахувати залежність між результатами двох завдань. Тому воно дуже схоже на багатоцільове навчання, але багатоцільове навчання є більш загальним, що стосується завдань, які можуть бути абсолютно різними.

Головною метою було б знайти строгу математичну базу для багатоцільового навчання на основі того, що було зроблено для багатозадачного навчання, а отже, знайти математичні результати щодо помилки наступного виду:

$E = \alpha_1 L_1 (x, W_1) + \alpha_2 L_2 (x, W_2) + \lambda R (W_1, W_2)$

Було показано, що ми можемо отримати хороші результати для багатьох подібних завдань [27] за допомогою деяких заздалегідь визначених моделей (SVM, функції ядра) [28] [29]. Таким чином, для багатоцільового навчання з K завданнями.

РОЗДІЛ 2 РЕАЛІЗАЦІЯ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ЗНАХОДЖЕННЯ ГЛИБИНИ ДВОВИМІРНОГО ЗОБРАЖЕННЯ ТА ЙОГО СЕМАНТИЧНА СЕГМЕНТАЦІЯ

2.1. Підготовка даних

Машинне навчання допомагає нам знаходити закономірності в даних - шаблони, які ми потім використовуємо для прогнозування нових даних. Щоб ці прогнози були правильними, ми повинні побудувати набір даних і правильно перетворити дані. Якщо є багато недоречної та надлишкової інформації, або наявні дані ненадійні чи зашумлені, то виявлення закономірностей на етапі навчання є складнішим. Етапи підготовки та фільтрації даних можуть зайняти значну кількість часу на обробку. Попередня обробка даних включає очищення, вибір екземпляра, нормалізацію, трансформацію, вилучення та виділення особливостей тощо. Результат попередньої обробки даних є остаточним навчальним набором.

Попередня обробка даних може вплинути на те, як ми інтерпретуємо остаточні дані. [30] Цей аспект слід ретельно розглянути, коли інтерпретація результатів є ключовим моментом, наприклад, при багатовимірній обробці хімічних даних (хемометрія).

Набір даних можна розглядати як сукупність об'єктів даних, які часто також називають записами, точками, векторами, шаблонами, подіями, випадками, зразками, спостереженнями або сутностями.

Об'єкти даних описуються низкою ознак, що охоплюють основні характеристики об'єкта, такі як маса фізичного об'єкта або час, коли сталася подія тощо. Функції часто називають змінними, характеристиками, полями, атрибути або розміри.

Особливості можуть бути:

Категоричні: Функції, значення яких беруться з певного набору значень. Наприклад, дні в тижні: {понеділок, вівторок, середа, четвер, п’ятниця, субота, неділя} - це категорія, оскільки її значення завжди береться з цього набору. Іншим прикладом може бути логічний набір: {True, False}

Числові: Функції, значення яких є безперервними або цілочисельними. Вони представлені числами і володіють більшістю властивостей чисел. Наприклад, кількість кроків, які ви проходите за день, або швидкість руху вашої машини.

Оцінка якості даних

Оскільки дані часто беруться з безлічі джерел, які зазвичай не надто надійні, а також у різних форматах, більше половини нашого часу витрачається на вирішення питань якості даних під час роботи над проблемою машинного навчання. Очікувати, що дані будуть ідеальними, просто нереально. Можуть виникнути проблеми через людські помилки, обмеження вимірювальних приладів або недоліки в процесі збору даних. Давайте розглянемо декілька з них та методи боротьби з ними:

Відсутні значення:

Дуже звично мати відсутні дані у вашому наборі даних. Можливо, це сталося під час збору даних, або, можливо, через якесь правило перевірки даних, але незалежно від цього відсутність значень необхідно враховувати.

Виключити рядки з відсутніми даними:

Проста і часом ефективна стратегія. Якщо в об’єкті здебільшого відсутні значення, то сам цей об'єкт також може бути усунений.

Оцінка відсутніх значень:

Якщо бракує лише розумного відсотка значень, ми також можемо запустити прості методи інтерполяції для заповнення цих значень. Однак

найпоширенішим методом роботи з відсутніми значеннями є заповнення їх середнім, медіанним або значенням режиму відповідної ознаки.

Невідповідні значення:

Ми знаємо, що дані можуть містити суперечливі значення. Наприклад, поле «Адреса» містить «Номер телефону». Можливо, це пов’язано з людською помилкою або, можливо, інформація була прочитана неправильно під час сканування з рукописного документу. Тому завжди рекомендується проводити оцінку даних, наприклад, знаючи, яким має бути тип даних об’єктів і чи однаковий він для всіх об’єктів даних.

Повторювані значення:

Набір даних може включати об’єкти даних, які є дублікатами один одного. Це може статися, коли, скажімо, одна і та ж особа подає форму неодноразово. Термін дедуплікація часто використовується для позначення процесу роботи з дублікатами. У більшості випадків дублікати видаляються, щоб не дати конкретному об'єкту даних перевагу або упередження під час запуску алгоритмів машинного навчання.

Агрегація даних

Агрегації даних виконується таким чином, щоб взяти агреговані значення наявних даних для їх покращення. Прикладом цього є дані транзакцій, припустимо, у нас є повсякденні операції з продуктом, що реєструють щоденні продажі цього товару в різних магазинах протягом року. Агрегація транзакцій до єдиних щомісячних або річних транзакцій магазину допоможе нам скоротити сотні або потенційно тисячі транзакцій, які відбуваються щодня в певному магазині, тим самим зменшуючи кількість об'єктів даних.

Це призводить до зменшення споживання пам'яті та часу обробки. Агрегації забезпечують нам високий рівень даних, оскільки поведінка груп або агрегатів стабільніша, ніж окремі об'єкти даних.

Опис використаних даних.

Основою використаного датасету є CelebA-HQ [31], який містить 30000 портретних знімків знаменитостей з виконаною вручну семантичною сегментацією частин обличчя. Рис. 8-10 демонструє приклад об’єктів з датасеті CelebA-HQ. Цей набір даних гарно зарекомендував себе в низці робіт з виділення об’єктів на обличчі, для прикладу BiSeNet успішно сегментував обличчя на частини, визначені в цьому датасеті.

Рис. 8. Фотографія з датасету.

Рис. 9. Мітка волосся з датасету.

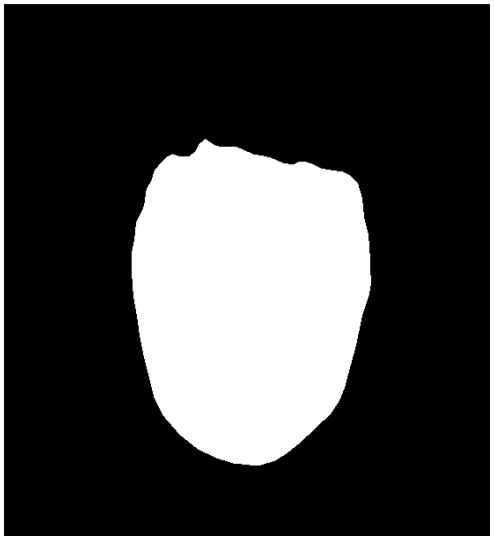

Рис. 10. Мітка шкіри обличчя.

CelebA-HQ - це готовий набір даних для семантичної сегментації, проте він не може бути використаний для задачі пошуку глибини зображення, яка також розглядається у цій роботі. Нам необхідно доповнити існуючий датасет

мітками, які є картою глибини. Існує декілька підходів отримання датасету карти глибини. Один з найбільш популярних - використання професійних вимірювачів глибини поверхні, які втім є дуже дорогими і також не можуть бути використані у нашому випадку готового датасету зображень. Цікавим вирішенням такої проблеми є використання наявних алгоритмів пошуку глибини зображення. Такий підхід був успішно застосований у попередніх роботах. [32] У цій роботі ми виконали попередню обробку датасету CelebA-HQ за допомогою алгоритму DF2Net запропонованого у роботі [32].

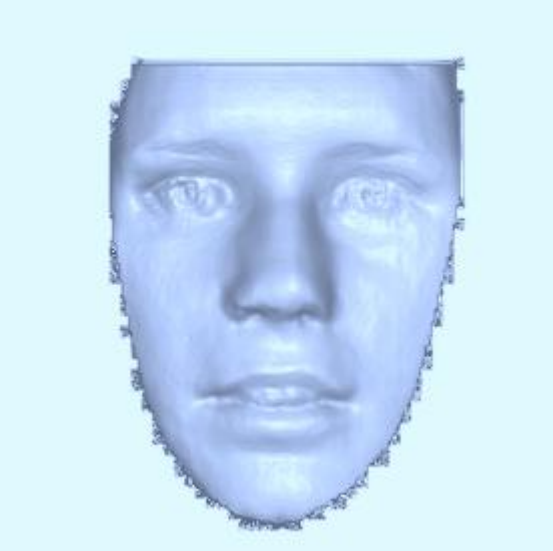

Рис. 11 Карта глибини отримана за допомогою DF2Net

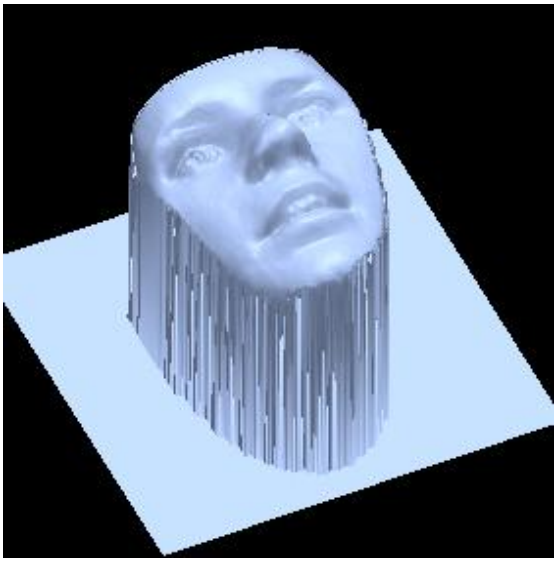

Рис. 12 Карта глибини отримана за допомогою DF2Net

При обробці CelebA-HQ за допомогою DF2Net виникли деякі труднощі. Перша проблема - DF2Net виконує попереднє виділення маски лиця відкидаючи при цьому волосся та шию, які є серед міток початкового датасету. Зрозуміло, що DF2Net не надійний у задачі обчислення глибини волосся, а тому було вирішено використовувати такі ж самі маски лиця, які використовує DF2Net. Іншою проблемою при обробці зображень став час виконання, оскільки початковий датасет містить 30000 зображень загальним обсягом пам’яті 3 ГБ. Враховуючи обчислювальну складність роботи із зображеннями, повна обробка

за допомогою DF2Net зайняла більше доби, а враховуючи те, що вихідна карта глибини - це матриця записана в Matlab файл (.mat), вона ніяк не закодована і займає 2МБ, отриманий датасет займає в сумі більше 60 ГБ. Наступною проблемою стала ненадійність DF2Net. Хоча загалом цей алгоритм працює відмінно, на деяких зображеннях він не зміг побудувати карту глибини. Зважаючи на це, було проведено ручну очистку отриманого датасету і вилучено непоказові дані. Враховуючи кількість даних, ми не можемо гарантувати повну чистоту отриманого датасету, проте відсоток помилок дуже низький.

2.2. Опис архітектури моделі

Нейронні мережі - це складні структури, які складаються зі штучних нейронів, які можуть приймати кілька входів для отримання єдиного виходу. Це основна робота нейронної мережі - перетворити вхід у значущий результат. Зазвичай нейронна мережа складається з вхідного та вихідного рівня з одним або декількома прихованими шарами всередині.

У нейронній мережі всі нейрони впливають один на одного, а отже, всі вони пов’язані. Мережа може визнавати та спостерігати кожен аспект набору даних, що знаходиться під рукою, і те, як різні частини даних можуть стосуватися чи не співвідноситися між собою. Таким чином нейронні мережі здатні знаходити надзвичайно складні шаблони у величезних обсягах даних.

У нейронній мережі потік інформації відбувається двома шляхами.

Мережі прямої передачі: у цій моделі сигнали рухаються лише в одному напрямку, до вихідного шару. Мережі Feedforward мають вхідний рівень і один вихідний рівень з нулем або декількома прихованими шарами (Рис. 13.). Вони широко використовуються при розпізнаванні зразків.

Мережі зворотного зв’язку: у цій моделі рекурентні або інтерактивні мережі використовують свій внутрішній стан (пам’ять) для обробки послідовності входів. У них сигнали можуть рухатися в обох напрямках через петлі (прихований шар / шари) в мережі. Вони зазвичай використовуються в часових рядах та послідовних завданнях.

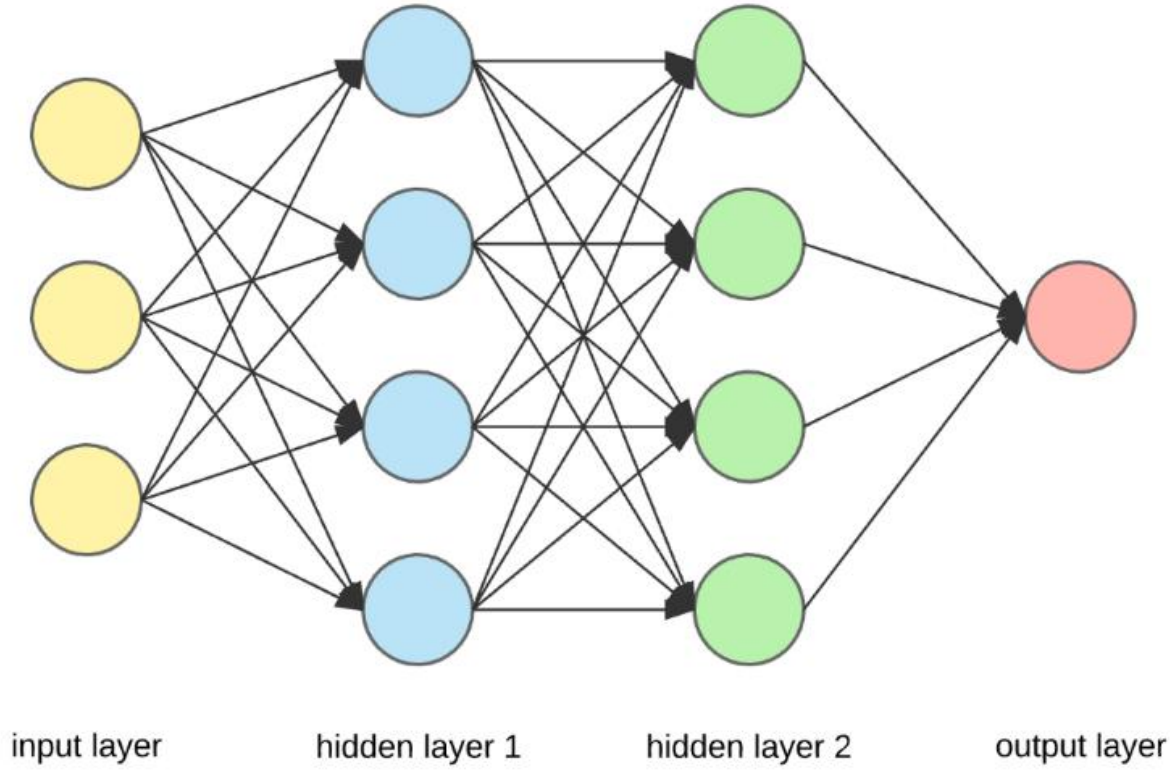

Рис. 13. Приклад мережі прямої передачі.

U-Net.

Для пошуку глибини зображення було обрано архітектуру під назвою U-Net, запропонованою у [33]. Ця архітектура добре показала себе у задачах, де обробка зображення відбувається попіксельно. Таким чином у нашому випадку кожному пікселю вхідного зображення ставиться у відповідність значення - глибина пікселя. Отже це задача регресії.

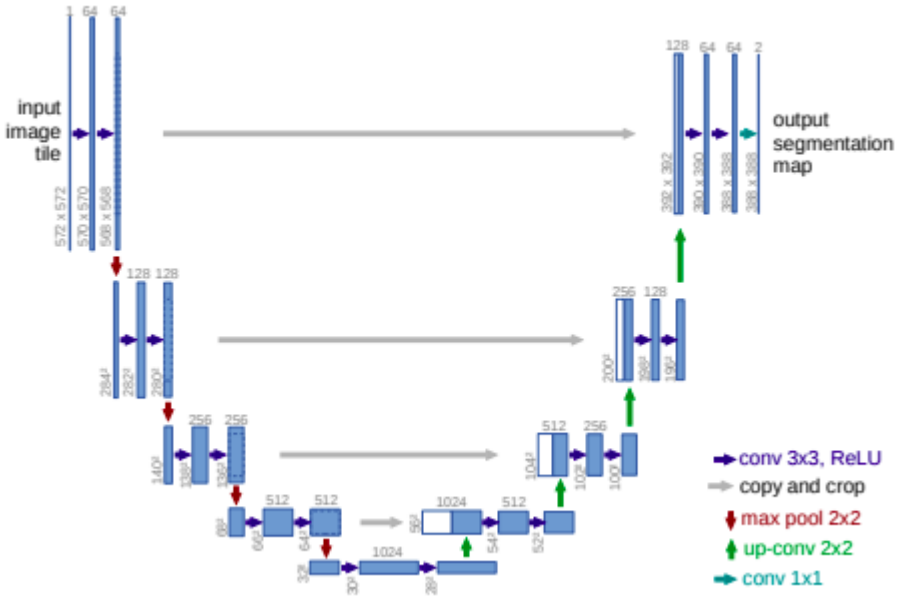

Рис. 14. Архітектура U-Net у випадку зображення 32x32.

U-Net є прикладом так званої повністю згорткової нейронної мережі (Fully convolutional neural network, FCNN). На рис. 14 зображена архітектура U-Net. Вона складається із звужуючого шляху (ліва сторона) та розширюючого шляху (права сторона). Шлях звуження відповідає типовій архітектурі згорткової мережі. Він складається з багаторазового застосування двох згорток 3x3, є ReLU та операція об'єднання 2x2 з кроком 2 для зменшення вибірки. На кожному кроці зменшення вибірки ми подвоюємо кількість функціональних каналів. Кожен крок у розширюючому шляху складається з розширення карти об’єктів, за яким слідує згортка 2x2, яка зменшує вдвічі кількість каналів об’єктів, об’єднання з відповідно обрізаною картою об’єктів із шляху звуження та дві 3x3 згортки, кожна з яких супроводжується ReLU. Обрізання необхідне через втрату граничних пікселів в кожній згортці. На кінцевому шарі згортка 1x1 використовується для відображення кожного 64-компонентного вектора ознак до бажаної кількості класів. Загалом мережа має 23 згорткові шари. [33]

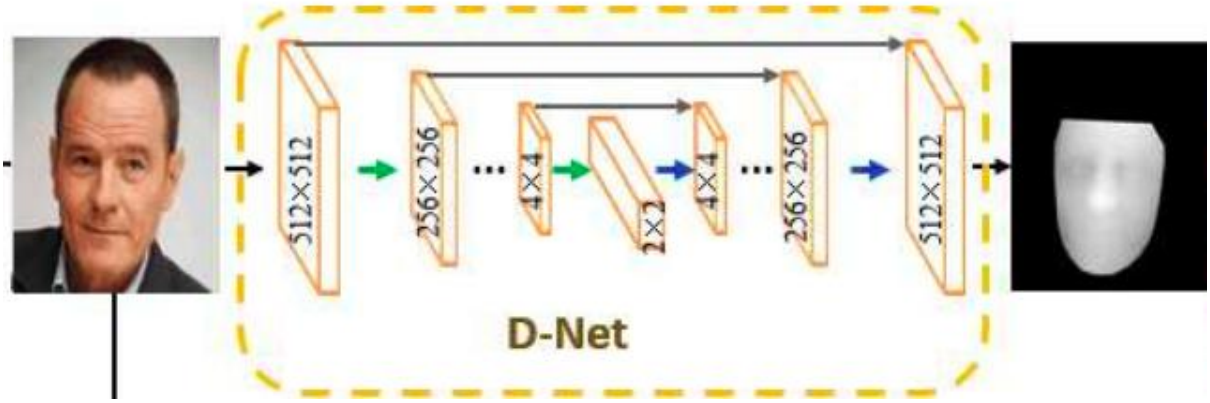

Рис. 15. Використання U-Net (D-Net) для отримання глибини зображення. [32]

BiSeNet

Задача сегментації зображення вимагає дещо іншого підходу, а тому для її вирішення було обрану другу архітектуру під назвою BiSeNet [34].

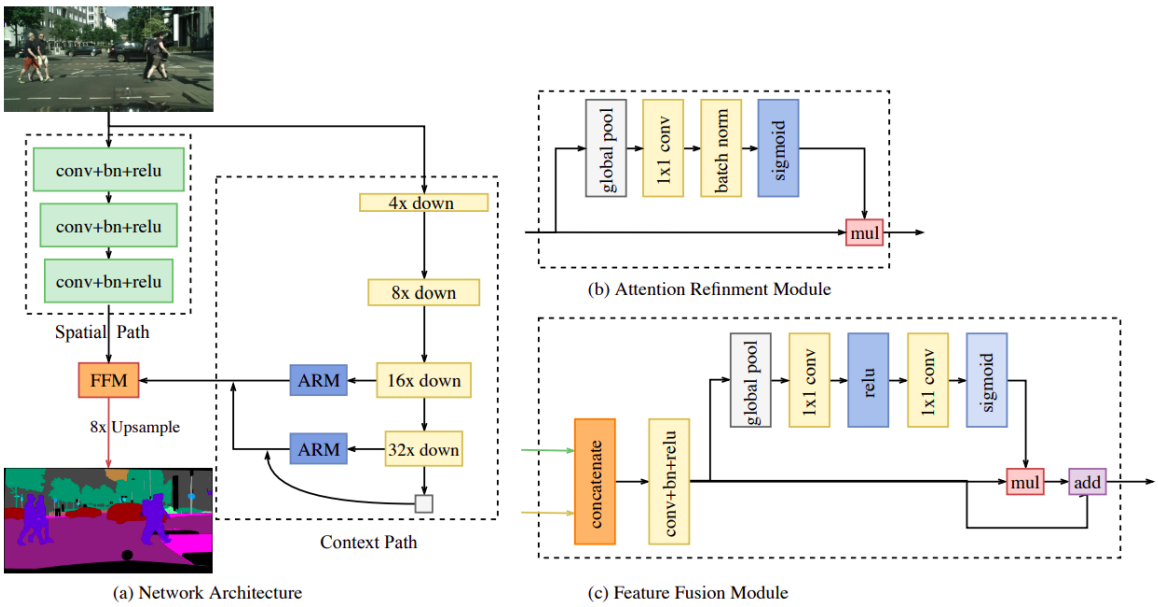

Рис. 16. Архітектура BiSeNet

Ця архітектура показала себе як одна з найкращих для завдання семантичної сегментації зображень.

Таким чином наша модель складається з двох архітектур: U-Net (D-Net) та BiSeNet. Перша виконує завдання пошуку глибини зображення, а друга сегментації зображення.

2.3. Функція втрат

Для оцінки роботи архітектури BiSeNet ми використовуємо функцію допоміжних втрат для нагляду за навчанням, яку показано у роботі [34]. Ми використовуємо функцію основних втрат для нагляду за результатами роботи цілого BiSeNet. Більше того, ми додаємо дві специфічні допоміжні функції втрат для нагляду за результатами контекстного шляху, такі як глибокий нагляд [35]. Як показує наступне рівняння, усі функції втрат є втратами Softmax.

Крім того, ми використовуємо параметр α, щоб збалансувати вагу основних втрат та допоміжних втрат, як представлено у другому рівнянні. Значення α в нашій роботі дорівнює 1. Спільні втрати роблять оптимізатор більш зручним для оптимізації моделі.

$$loss = \frac{1}{N}\sum_{i=1}^{n} L_i = \frac{1}{N}\sum_{i=1}^{n} -log(\frac{e^{p_i}}{\sum_{j=1}^{n} e^{p_j}}),$$

де p - передбачення моделі.

$$L(X;\ W) = l_p(X;\ W) + \alpha \sum_{i\ =\ 2}^{K} l_i(X_i; W)$$

де l_p - основна втрата об'єднаного виходу. X_i - вихідне значення з етапу i моделі Xception. l_i - допоміжна втрата для стадії i. K дорівнює 3 у нашій роботі. Тут ми використовуємо лише допоміжні втрати на етапі навчання. [34]

Оцінка архітектури DNet відбувається за допомогою функції втрат L_1, також відомою як MAE (Mean Absolute Error).

$$L_1 = || D_{gt} - D_{pred} ||,$$

де D_{gt} - тренувальне значення, D_{pred} - передбачене значення.

2.4. Програмна реалізація та підготовка даних

Описана модель була реалізована на Python 3.8 та PyTorch. Основою моделі є клас torch.nn.Module, від якого наслідують класи DNet, BiSeNet та DBiSeNet. Вся підготовка даних також виконана за допомогою Python 3.8. Спочатку для кожного зображення в датасеті було створено вирізане лице та його маску, після чого невдалі зразки видалено. На цьому етапі маємо 30000

зображень початкового датасету та ~56000 нових зображень. З початкового датасету обрану підмножину вдалих зображень і використовуючи для таких зображень їх вирізане лице та маску було створено карту глибини за допомогою DF2Net. Отримано ~23000 карт глибини. Таким чином ми отримали необхідний датасет з ~23000 зображень, для яких маємо відповідні мітки сегментації та карти глибини.

2.5. Навчання

Через значну кількість тренувальних даних, а також природу роботи із зображеннями, для тренування такої мережі необхідні значні обчислювальні ресурси. Перша спроба тренування була проведена на навчальному ноутбуку без відеокарти, а тому ми не змогли в достатній мірі натренувати мережу, але навіть з таким навчанням, можна побачити, що модель має хороший потенціал. Наступним кроком буде навчання на серверах Google за допомогою платформи GCloud.

Попередні результати

Далі наведемо деякі результати з процесу навчання, які демонструють прогрес, який було отримано за 1000 ітерацій.

Рис. 17 Сегментація зображення після 10 ітерацій.

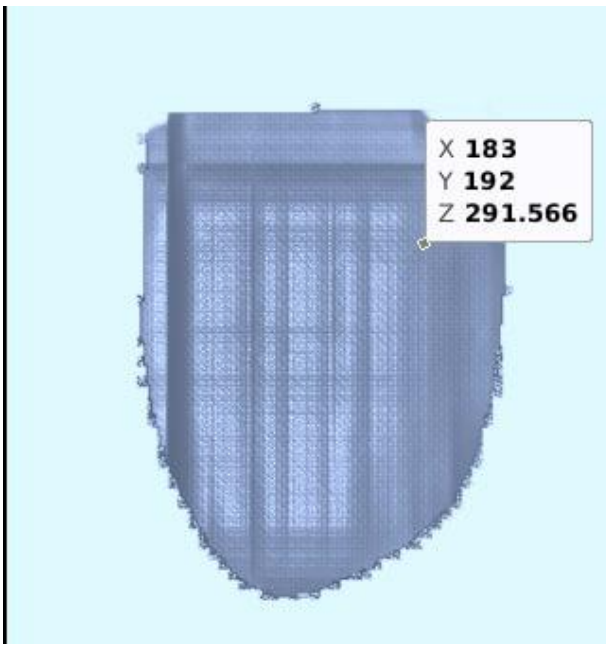

Рис. 18. Карта глибини після 10 ітерацій.

Як і варто було очікувати, після 10 ітерацій навчання, сегментація не відображає ніяких корисних даних, по суті маємо просто випадковий набір класів накладених на фотографію. Карта глибини також не показує ніяких результатів, маємо набір майже випадкових значень глибини пікселів.

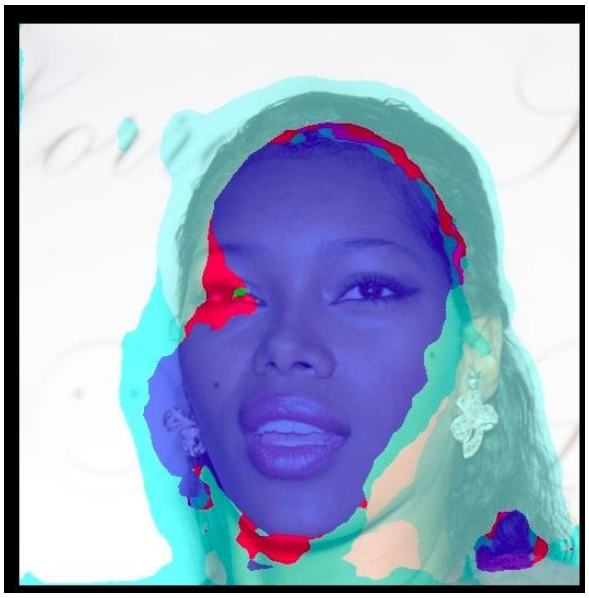

Рис. 19. Сегментація зображення після 500 ітерацій

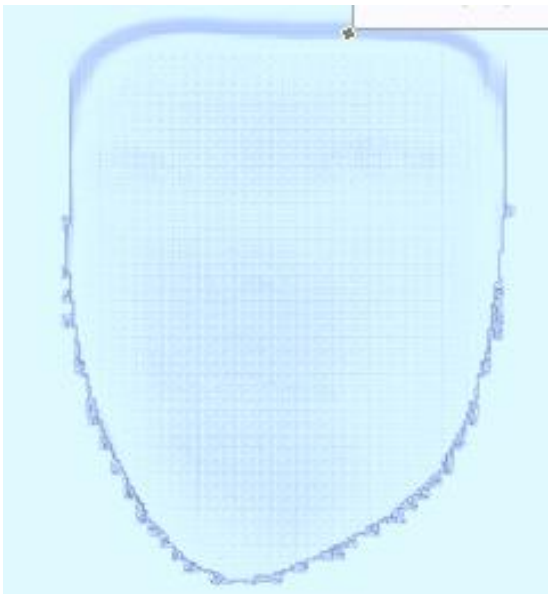

Рис. 20. Карта глибини після 500 ітерацій.

На рис. 18 можна помітити, що кількість класів на зображенні значно зменшилась, а їх точність збільшилась. Модель вже значно краще розрізняє великі об’єкти, як шкіру лиця, а також починає помічати менші, як око. Карта глибини також покращилась, тепер відтінені області справді відповідають об’єктам, які ближче до спостерігача.

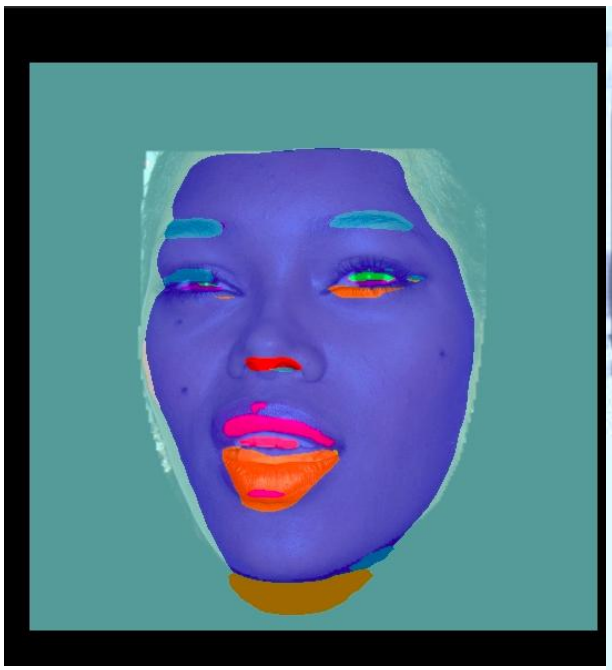

Рис. 21. Сегментація зображення після 1000 ітерацій

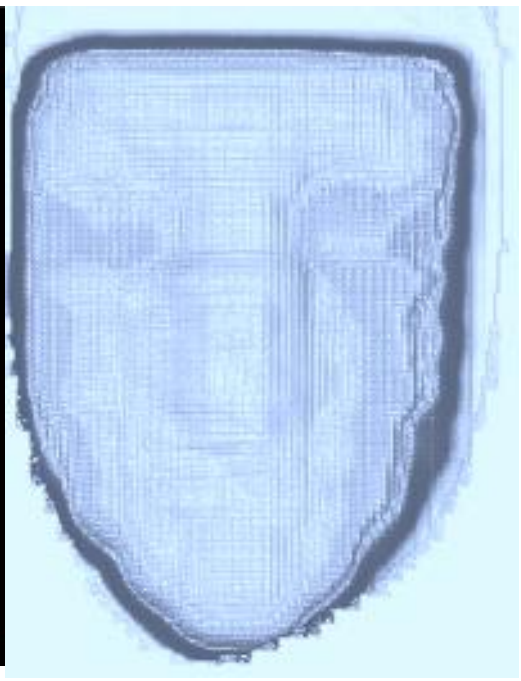

Рис. 22. Карта глибини після 1000 ітерацій.

Після 1000 ітерацій модель досить добре виділяє шкіру лиця, а також точно визначила положення очей. Також видно початок розрізнення інших класів - роту, носу, брів. Карта глибини навчилась добре виділяти глибину на контурах обличчя, а також можна побачити контури носу, роту і очних впадин. Очевидно, що з більшою кількістю ітерацій результати значно покращаться.

ВИСНОВКИ

Перед початком виконання роботи були поставлені цілі, серед яких

- Дослідити існуючі наукові статті за темою роботи
- Проаналізувати відомі підходи до розв’язання поставленої задачі
- Описати ідею власного розв’язку задачі оцінки глибини двовимірного зображення та його сегментації.
- Запропонувати модель глибинного навчання, що вирішує поставлену задачу.
- Зібрати тренувальні дані для моделі.
- Виконати навчання запропонованої моделі.
- Використати отриману модель для отримання результатів.

Ці завдання були успішно виконані: наведено низку робіт, які розглядали задачі знаходження глибини зображення та сегментації, систематизовано необхідні теоретичні відомості для виконання завдань. Було запропоновано модель нейронної мережі, яка знаходить карту глибини зображення, а також сегментує його. На основі існуючого датасету створено новий, який дозволяє тренувати мережу для пошуку глибини зображення та його сегментації. Проведено навчання моделі на особистому ноутбуці, що дало змогу довести перспективність наведеної моделі, хоча і не показало остаточні результати, які відповідають рівню існуючих моделей.

У цій роботі ми показали, що нейронні мережі мають великий потенціал у задачах знаходження глибини двовимірного зображення і їх семантичної сегментації. Серед можливих застосувань запропонованої моделі можна згадати мобільні додатки, які останнім часом стають все більш популярними. Серед успішних додатків зі схожою ідеєю можна назвати ReFace, який дозволяє міняти лице знаменитостей на своє, а також досить поширені в різних додатках маски. Наша модель має потенціал для створення додатку, який дозволяє

міняти частини лиця на інші, як запропоновано в [36]. Проте у нашому випадку ми маємо можливість демонструвати результати в тривимірному просторі.

Загалом очевидно, що глибинні нейронні мережі є надзвичайно потужним інструментом у задачах обробки зображення. Поле їх можливих способів застосування неосяжне, а робота в галузі семантичної сегментації і виділення карти глибини не закінчена. Ця робота дозволяє виконувати обидві задачі одночасно заощаджуючи час і пам’ять. Перспективи покращення, а також застосування чітко зрозумілі, а тому є сенс продовжувати розвивати тему розглянуту у дипломній роботі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. IJCV, 47(1-3):7–42, 2002
2. D. Forsyth and J. Ponce. Computer Vision: a Modern Approach. Prentice Hall, 2002.
3. A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In NIPS, 2006.
4. A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. IEEE TPAMI, 31(5):824–840, 2009.
5. D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. IJCV, 75(1):151–172, 2007.
6. L. Ladicky, J. Shi, and M. Pollefeys. Pulling things out of perspective. In CVPR, 2014.
7. X. Li, H. Qin, Y. Wang, Y. Zhang, and Q. Dai. Dept: depth estimation by parameter transfer for single still images. In ACCV, 2014.
8. A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. IJCV, 42(3):145–175, 2001.
9. H. Ha, S. Im, J. Park, H.-G. Jeon, and I. S. Kweon. High quality depth from uncalibrated small motion clip. In CVPR, 2016.
10. N. Kong and M. J. Black. Intrinsic depth: Improving depth transfer with intrinsic images. In ICCV, 2015.
11. K. Karsch, C. Liu, and S. B. Kang. Depth transfer: Depth extraction from video using non-parametric sampling. IEEE TPAMI, 36(11):2144–2158, 2014.
12. A. Rajagopalan, S. Chaudhuri, and U. Mudenagudi. Depth estimation and image restoration using defocused stereo pairs. IEEE TPAMI, 26(11):1521–1525, 2004

13. F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. IEEE TPAMI, 38(10):2024–2039, 2016.
14. P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. Yuille. Towards unified depth and semantic prediction from a single image. In CVPR, 2015
15. A. Roy and S. Todorovic. Monocular depth estimation using neural regression forest. In CVPR, 2016.
16. D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In ICCV, 2015.
17. https://deepai.org/machine-learning-glossary-and-terms/neural-network
18. http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
19. https://en.wikipedia.org/wiki/Channel_(digital_image)
20. https://en.wikipedia.org/wiki/Convolution
21. https://arxiv.org/abs/1511.07122
22. http://cs231n.github.io/transfer-learning/#tf
23. Bishop, Christopher M. (2006). Pattern Recognition and Machine Learning. Springer
24. Stephen Boyd and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.
25. David Sculley. Combined regression and ranking. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 979–988. ACM, 2010.
26. Pierre Lajous. Multi-Target Learning. Rapport - Introduction a la Recherche en Laboratoire. 2A MMIS, 2016
27. Abhishek Kumar and Hal Daume III. Learning task grouping and overlap in multi-task learning. arXiv preprint arXiv :1206.6417, 2012

28. Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 109–117. ACM, 2004
29. Theodoros Evgeniou, Charles A Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. In Journal of Machine Learning Research, pages 615–637, 2005.
30. Oliveri, Paolo; Malegori, Cristina; Simonetti, Remo; Casale, Monica (2019). "The impact of signal preprocessing on the final interpretation of analytical outcomes – A tutorial". Analytica Chimica Acta.
31. https://github.com/switchablenorms/CelebAMask-HQ
32. Xiaoxing Zeng, Xiaojiang Peng, Yu Qiao. DF2Net: A Dense-Fine-Finer Network for Detailed 3D Face Reconstruction. ICCV, 2019
33. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. Unet: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015.
34. Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, Nong Sang. BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation (2018)
35. Xie, S., Tu, Z.: Holistically-nested edge detection. In: IEEE International Conference on Computer Vision (2015)
36. Taesung Park, Ming-Yu Liu, Ting-Chun Wang, Jun-Yan Zhu, Semantic Image Synthesis with Spatially-Adaptive Normalization, CVPR 2019