НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

«До захисту допущено»
В.О. завідувача кафедри
_____ О.Л. Тимощук

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 “Комп’ютерні науки”

на тему: «Система ідентифікації малярійних клітин використовуючи нейронні мережі»

Виконав:
студент IV курсу, групи КА-55
Срібний Андрій Євгенович _____

Керівник:
проф. кафедри ММСА,
професор, д.т.н. Зайченко О. Ю. _____

Консультант з економічного розділу:
доцент, к.е.н. Шевчук О. А. _____

Консультант з нормоконтролю:
доцент, к.т.н. Коваленко А. Є. _____

Рецензент:
заст. декана ТЕФ з педагогічної роботи,
к.т.н. Кондратюк В. А. _____

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

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки – 6.050101 "Комп'ютерні науки"

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.Л. Тимощук
(підпис)
«___»_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту
Срібному Андрію Євгеновичу

1. Тема роботи «Система ідентифікації малярійних клітин, використовуючи нейронні мережі», керівник роботи Зайченко Олена Юріївна, д.т.н. професор кафедри ММСА, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи _____
_____
_____
_____

4. Зміст роботи _____
_____
_____
_____

5. Перелік завдань, які потрібно розробити:
_____
_____
_____
_____
_____
_____
_____
_____
_____

6. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____
_____
_____
_____

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А, доцент, к.е.н.		

8. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка

Студент _____ А.Є. Срібний
(підпис)

Керівник роботи _____ О.Ю. Зайченко
(підпис)

РЕФЕРАТ

Дипломна робота: 110 с., 27 рис., 16 табл., 2 додатки, 37 джерел.

ДІАГНОСТИКА МАЛЯРІЇ, НЕЙРОННІ МЕРЕЖІ, МАЛЯРІЯ, ГЛИБИННЕ НАВЧАННЯ

В XXI сторіччі захворювання дуже поширені, але серед усіх малярія займає провідне місце, оскільки щорічно малярія є причиною близько трьох мільйонів смертей, третина з них - діти. Раніше було запропоновано та впроваджено підхід, завдяки якому малярію можна виявити шляхом взяття аналізу крові пацієнтів у лабораторії для дослідження, але цей метод викликає затримку на початку лікування, тому що витрачається багато часу на дослідження. За рахунок чого, коефіцієнт смертності значно зростає на захворювання малярією в світі.

Автоматизація процесу діагностики дозволить точно та швидко виявити та діагностувати захворювання і, отже, обіцяє забезпечити надійну медичну допомогу в районах з обмеженими ресурсами.

Об’єкт дослідження – вибірка зображень інфікованих та неінфікованих клітин.

Мета даного дослідження полягає в прискоренні процесу точного виявлення та діагностування малярії. Технології машинного навчання були використані для автоматичної діагностики малярії.

У запропонованому підході фотографії зараженої крові пацієнтів розглядаються як вихідні дані, система аналізує ці дані і прогнозує їх результат як позитивний або негативний для малярії.

Цей додаток буде корисний для тих районів, де немає лабораторії зі спеціальним обладнанням або там, де немає лікаря, але існує особа, яка здатна керувати цим додатком лише додаючи у нього фотографії аналізу крові пацієнта, тим самим швидко надати кваліфіковану допомогу.

ABSTRACT

The work consists of 110 pages, 27 images, 16 tables, 2 appendices and 37 sources.

The theme: «Identification system of Malaria cells using neural networks».

MALARIA DIAGNOSTICS, NEURAL NETWORKS, MALARIA, DEEP LEARNING

In the twenty-first century, the disease is very common, with malaria on the leading place, since malaria causes about three million deaths annually, one third of them are death of children. Previously, an approach through which malaria could be detected by taking blood samples from patients in the laboratory for research was proposed and implemented. But that method cause a delay at the start of treatment, as it takes a lot of time to research. Due to this, the death rate from malaria is significantly increasing in the world.

The automation of the diagnostic process will allow accurately and rapidly detect and diagnose the disease and, therefore, promises to provide reliable medical care in areas with limited resources. The object of the study is a sample of images of infected and uninfected cells.

The purpose of this study is to accelerate the process of accurate detection and diagnosis of malaria. Machine learning technologies have been used to automatically diagnose malaria.

In the proposed approach, photos of infected blood of patients are considered as the source data, the system analyzes this data and predicts their outcome as positive or negative for malaria.

This application will be useful for those areas where there is no laboratory with special equipment or where there is no doctor, but there is a person who is able to manage this application only by adding pictures of patient`s blood analysis, thereby quickly providing qualified assistance.

ЗМІСТ

ВСТУП

Малярію викликають найпростіші паразити роду Plasmodium, які передаються через укуси інфікованих самок Anopheles комарів, які інфікують червоні кров'яні тільця. Більшість смертей відбувається серед дітей у Африці, де дитина вмирає майже кожну хвилину від малярії, і де малярія є провідною причиною дитячої нейро-інвалідності. За даними Всесвітнього звіту Малярії 2018, приблизно 3,2 мільярда людей у 95 країнах та територіях ризикують заразитися малярією і хворобами, що розвиваються, і 1,2 млрд. з високим ризиком (> 1 на 1000 випадків отримання малярії на рік). У світі було близько 214 мільйонів випадків малярії у 2018 році та близько 438 000 смертей від малярії. Тягар був найважчим у Африканському регіоні, де, за оцінками, 92% всіх смертей були від малярії, і у дітей у віці до 5 років, на яких припадало понад дві третини від усіх. Типові симптоми Малярії включають лихоманку, стомлюваність, головний біль, а, у тяжких випадках, судоми і кома, що призводять до смерті. Сотні мільйонів зразків крові розглядаються кожен рік для знаходження малярії, що включає ручний підрахунок паразитів і інфікованих червоних кров'яних тілець підготовленим мікроскопістом. Точна кількість паразитів важлива не тільки для діагнозу малярії, вони також важливі для тестування на лікарську стійкість, вимірювання ефективності лікарських засобів та класифікації тяжкості захворювання. Однак мікроскопічна діагностика не стандартизована і значною мірою залежить від досвіду і майстерності мікроскопіста, які в умовах з обмеженим рівнем ресурсів працюють в ізоляції, і у них немає жорсткої системи, яка могла б забезпечити їх підтримку і, таким чином, це призводить до неправильних діагностичних рішень. Для помилково-негативних випадків, це призводить до непотрібного застосування антибіотиків, іншої консультації, втрати днів роботи, і в деяких випадках прогресування до важкої малярії. Для помилково-позитивних випадків помилковий діагноз спричиняє зайве використання ліків від малярії і

страждають від їх потенційних побічних ефектів, такі як нудота, біль у животі, діарея, і іноді важкі ускладнення.

Цей аналіз діагностики малярії підштовхнув мене зробити автоматичну діагностику малярії. Автоматичний підрахунок паразитів має ряд переваг у порівнянні з ручним підрахунком: він забезпечує більш надійну і стандартизовану інтерпретація зразків крові, також він дозволяє обслужити більше пацієнтів, за рахунок зменшення навантаження працівників по боротьбі з малярією, і це може зменшити витрати часу на діагностику.

Кілька етапів обробки зазвичай необхідно для автоматичного кількісного визначення хвороби. По-перше, необхідно отримати цифрові зображення слайдів крові, які часто вимагають попередньої обробки для нормалізації освітлення або варіацій фарбування. На другому етапі необхідно виявити клітини крові або паразити. Для клітин крові це, як правило, передбачає сегментацію клітин для ідентифікації окремих клітин у згустках клітин для отримання точного кількості клітин. На третьому етапі, після виявлення та сегментації клітин, обчислюються ознаки для опису типового зовнішнього вигляду інфікованої та неінфікованої клітини крові. На етапі остаточної класифікації класифікатор, який пройшов навчання на незалежному та зазвичай вручну вибраному навчальному наборі, потім розрізняє інфіковані та неінфіковані клітини.

Отже, дана атестаційна бакалаврська робота присвячена автоматизації діагностики малярії, як основний або додатковий метод знаходження клітин малярії.

ПОСТАНОВКА ЗАДАЧІ

Метою цієї роботи є побудова системи для автоматизації діагностики малярії, тобто для розпізнавання інфікованих клітин на зображенні. Для розв’язання поставленої задачі було використано згорткову нейронну мережу, оскільки вони мають можливість автоматично витягувати функції та навчати фільтри і активно використовується для розпізнавання зображень.

Отже, задача сформована таким чином:

1. Вивчення існуючих літературних джерел, присвячених малярії та її діагностиці, розробці та підвищенню ефективності роботи комп'ютерних систем медичної діагностики на зображеннях.
2. Проектування і розробка ефективної адаптивної моделі класифікатора малярії на медичних зображеннях.
3. Розробка програмної системи для оцінки ефективності запропонованої моделі класифікатора.
4. Проведення експериментів по оцінці ефективності запропонованої моделі, аналіз і інтерпретація результатів.

РОЗДІЛ 1 АНАЛІЗ ЗАДАЧІ. ОСНОВНІ ПОНЯТТЯ МАЛЯРІЇ, ДІАГНОСТИКА

1.1 Малярія. Основні поняття

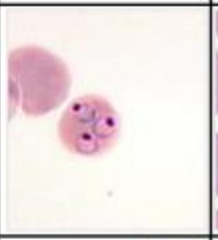
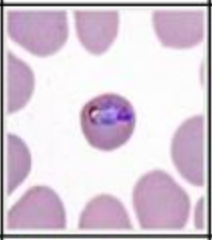
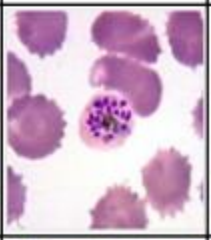
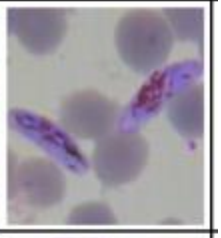
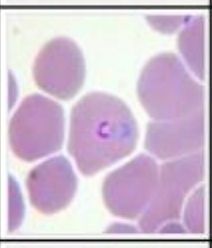
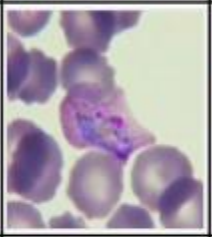
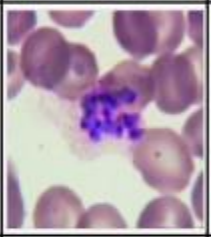
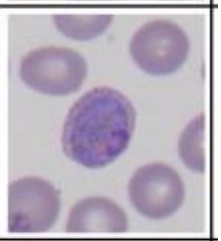
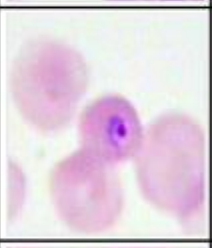
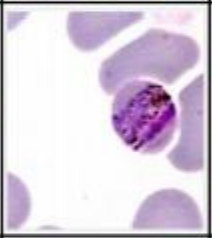
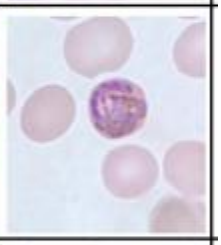
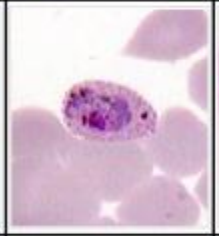
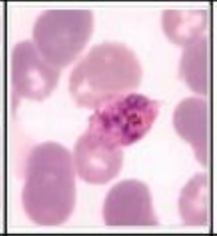
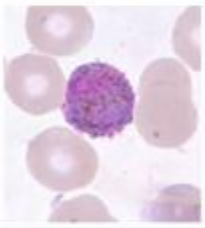
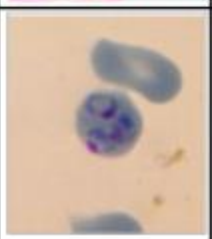
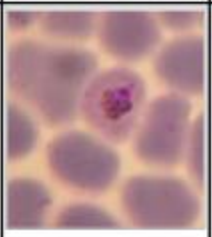
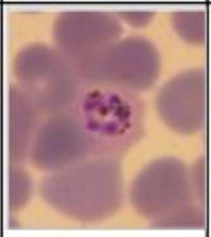
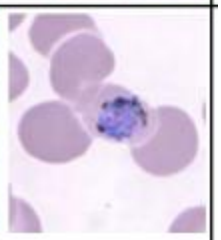
Існує 5 видів Plasmodium, які викликають малярію у людини: Plasmodium falciparum, Plasmodium vivax, Plasmodium malariae, Plasmodium ovale, Plasmodium knowlesi. Визначають 2 найбільш поширених виду - P. falciparum і P. vivax. P. falciparum є найбільш важкою формою і відповідає за більшість смертей, пов'язаних з малярією в усьому світі. P. falciparum є найбільш поширеним паразитом малярії в Африці, що знаходиться на південь від Сахари, що становить 99% випадків захворювання малярією в 2018 році. P. vivax є переважаючим паразитом у регіоні Америки, що становить 64% випадків малярії, і становить понад 30% у Південно-Східній Азії та 40% у східно-середземноморських регіонах. Кожен з цих видів паразитів проходить стадії протягом циклу розвитку (48 годин), що дає паразитам інший зовнішній вигляд, який можна спостерігати під мікроскопом. У хронологічному порядку ці стадії - кільцева стадія, трофозоїтна стадія, стадія шизонта і стадія гаметоцитів. На таблиці 1 показані типові приклади всіх стадій для кожного виду.

У несерйозній малярії, у більшості випадків на початковій стадії (<24 години) P. falciparum присутні в периферичній крові, тоді як для важкої малярії в периферичній крові можуть бути всі стадії. Для P. falciparum червоні кров'яні клітини, інфіковані трофозоїтами, зникають з периферичного кровообігу шляхом прикріплення до стінок капілярів у життєво важливих органах, що є процесом, що називається секвестрацією. Якщо капіляри заблоковані для ново інфікованих клітин вже прикріпленими клітинами, то в периферичній крові буде видно більш зрілі стадії паразита (трофозоїти і шизонти), що свідчить про важку інфекцію і поганий прогноз.

Для P. falciparum кільцеві стадії мають видиму цитоплазму і 1 або 2 невеликі точки хроматину. Інфіковані клітини крові не збільшені, але можуть

мати декілька інфекцій. Трофозоїти P. falciparum рідко зустрічаються в мазках периферичної крові. Цитоплазма зрілих трофозоїтів має тенденцію бути більш щільною, ніж молодші кільця, трофозоїти можуть з'являтися округлої форми з бурими малярними пігментами всередині. Шизонти P. falciparum також рідко

Таблиця 1 - П'ять різних видів малярії людини Plasmodium

Stages / Species	Ring	Trophozoite	Schizont	Gametocyte
P. falciparum				
P. vivax				
P. malariae				
P. ovale				
P. knowlesi				

зустрічаються в периферичній крові. Вони відображають більше 2 і до 32 ядер (мерозоїти) з темно-коричневим пігментом, зчепленим посередині. Гаметоцити P. falciparum мають форму півмісяця або ковбаси і можуть бути помічені в мазці крові через 1 тиждень після інфікування паразитами.

Хроматин видно як єдина маса або дифузна. Подібні спостереження можна зробити для стадій інших паразитних видів. Наприклад, для P. vivax клітини господаря часто збільшені і мають неправильну форму. Трофозоїти по формі є амебоїди з малярійним пігментом, і при важких інфекціях численні інфекції окремих клітин крові не є рідкістю. Для P. malariae клітини-хазяїни не збільшені. Трофозоїти мають сильну тенденцію до утворення смуги з малярійним пігментом, розкиданим по всьому діаметру інфікованих еритроцитів. Множинні інфекції є надзвичайно рідкісними для P. malariae. З іншого боку, для P. ovale клітини-господарі злегка збільшені і мають овальну форму з гофрованими кінцями, часто облямовані. Паразити злегка збільшені, а трофозоїди є амебоїдами у формі малярійного пігменту. Декілька інфекцій однієї клітини зустрічаються частіше, ніж P. vivax. Для P. knowlesi, інфіковані еритроцити не з'являються збільшеними. Паразитний цикл еритоцитів становить всього 24 години, що коротше, ніж цикл P. falciparum (48 годин) і набагато коротший за цикл P. malariae (72 години), що призведе до тієї ж стадії, що спостерігається в периферичній крові щодня в певний час. Морфологія паразитів P. knowlesi схожа на P. malariae. Трофозоїти можуть містити розсип малярійного пігменту всередині, форма смуги подібно до P. malariae, але їх цитоплазма є більш нерегулярною, і можна побачити кілька паразитів, які заражають 1 еритроцит, як у P. falciparum. На рисунку 1 зображено 2 приклади різних стадій паразита в одному і тому ж тонкому зображенні слайда крові. На першому зображенні слайди P. falciparum trophozoites і gametocytes можна побачити разом з білими клітинами крові. Останні більші і мають яскраво виражене ядро порівняно з багатьма еритроцитами на зображенні. На другому зображенні кільцеві стадії P. falciparum знаходяться разом з шизонтами. Крім того, на обох зображеннях видно інші об'єкти, такі як зовнішні клітини паразита і шум від фарбування. Зокрема, шум від фарбування може бути сплутаний з паразитами недосвідченим мікроскопістом.

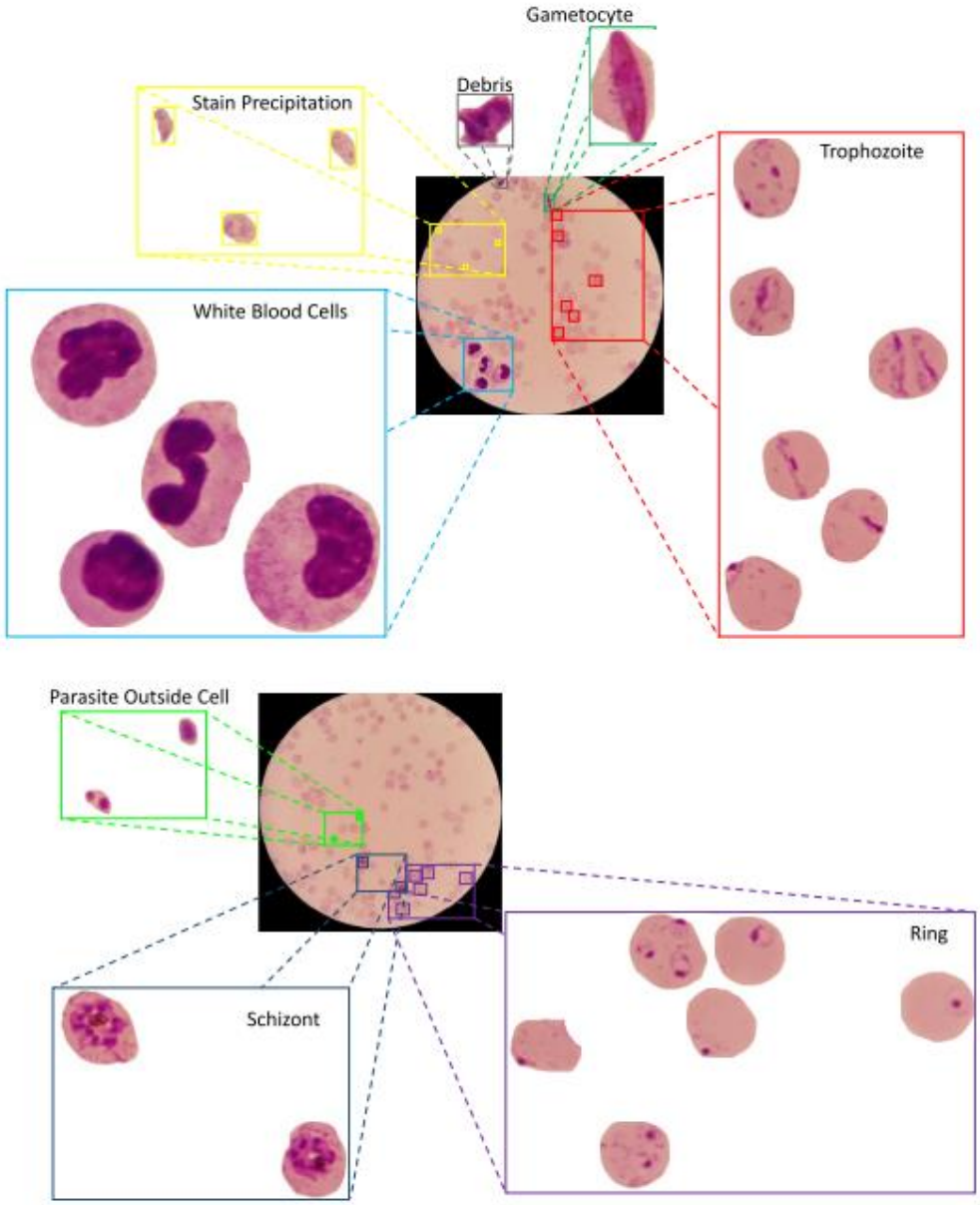


Рисунок 1 – Стадії паразиту на одному мазку крові

1.2 Актуальність задачі автоматизації розпізнавання мазків крові

Проблеми, пов'язані з діагностикою вручну, є причиною автоматизації процесу діагностики малярії. Автоматизація процесу діагностики забезпечить точну та швидку діагностику захворювання і, таким чином, обіцяє забезпечити надійне медичне обслуговування місцевості з обмеженими ресурсами. Таким чином, сільські райони, які страждають від відсутності спеціалізованої інфраструктури та кваліфікованої робочої сили, можуть отримати велику

користь від автоматизованої діагностики. Автоматизація діагностики малярії передбачає адаптацію методів, практик і знань звичайної мікроскопії до комп'ютеризованої структури системи . Раннє виявлення малярії є необхідним для забезпечення належної діагностики та збільшення шансів на лікування. З огляду на ступінь тяжкості та кількість загиблих від цієї хвороби, доцільно прийняти потенційні невеликі помилки впровадження, при введені автоматизованої системи. Автоматизована система складається з оптимізованих методів обробки зображень для початкової фільтрації та сегментації і набору алгоритмів розпізнавання і / або машинного навчання, спрямованих на надійне розпізнавання інфікованих клітин у світловому або оптовому мікроскопічному зображенні.

Отже, комп'ютерна система як система підтримки прийняття рішень може бути першорядною для швидшої і надійної діагностики. Це може допомогти забезпечити еталонний і стандартизований спосіб вимірювання ступеня інфекції захворювання.

1.3 Аналіз методів діагностики малярії

Малярія - це виліковна хвороба, з лікарськими засобами, доступними для лікування, включаючи препарати, які допомагають запобігти малярійним інфекціям у малярійних регіонах. Проте ще не існує ефективної вакцини проти малярії, хоча це і є область активних і польових досліджень. Після інфікування малярія є швидко прогресуючою хворобою з серйозним ризиком розвитку тяжкої та церебральної малярії з неврологічними симптомами для інфекцій P. falciparum. Тому своєчасна діагностика малярії дуже важлива. Незважаючи на те, що малярію можна діагностувати багатьма різними способами, існують можливості для поліпшення поточних діагностичних тестів малярії, включаючи зниження вартості, підвищення швидкості та специфічності, покращення простоти використання. Виділяють два найбільш важливих

діагностичних засоби- світлова мікроскопія та ШДТ(швидкі діагностичні тести).

Виявлення наявності паразитів є запорукою діагностики малярії. Крім того, важливим є виявлення видів паразитів і наявність потенційно змішаних інфекцій, а також спостереження за стадією розвитку паразитів P. falciparum щодо тяжкості захворювання. Підраховуючи паразитів для визначення рівня паразитемії, це не тільки важливо для виявлення інфекції і вимірювання її тяжкості, це також дозволяє контролювати пацієнтів шляхом вимірювання ефективності препарату і потенційної стійкості до лікарських засобів.

1.3.1 Світлова мікроскопія

Сучасним методом золотого стандарту для діагностики малярії є світлова мікроскопія плівок крові. Хоча інші форми діагностування існують і стали популярними в останні роки, зокрема ШДТ, мікроскопія залишається найпопулярнішим діагностичним засобом, особливо в умовах обмежених ресурсів. При мікроскопії можна виявити всі види паразитів. Це дозволяє обчислити рівень паразитії, перевірити пацієнта після успішного лікування, а також контролювати стійкість до лікарських засобів. Крім того, вона дешевша, ніж інші методи і широко доступна. Проте його найбільшими недоліками є велика підготовка, необхідна мікроскопістам для того, щоб стати досвідченим «читачем» слайдів малярії, високою вартістю навчання і працевлаштування, підтримкою навичок і великою складовою ручної роботи. Щоб діагностувати малярію під мікроскопом, на скляну склянку наносять краплю крові пацієнта, яка потім занурюється в розчин для фарбування, щоб зробити паразити більш помітними під звичайним світловим мікроскопом, як правило, з об'єктивом 100 × . Для діагностики малярії, як правило, готують два види мазків крові: товсті і тонкі мазки. Товстий мазок використовується для виявлення присутності паразитів у краплі крові. Товсті мазки дозволяють більш ефективно виявляти паразитів, ніж тонкі мазки, з чутливістю більшою у 11

разів. З іншого боку, тонкі мазки, які є результатом поширення краплі крові по склі, мають інші переваги. Вони дозволяють дослідникам виявляти види малярії і легше розпізнавати стадії паразита.

Фактичне мікроскопічне дослідження одного екземпляру крові, в тому числі кількісного виявлення паразитів та ідентифікації видів, займає у підготовленого мікроскопіста 15–30 хвилин. Враховуючи, що сотні тисяч зразків крові перевіряються вручну на малярію щороку, це становить величезні економічні зусилля, необхідні для діагностики малярії.

1.3.2 Швидкі діагностичні тести

Основна перевага мікроскопічної діагностики малярії полягає в її низькій вартості, що дає їм виразну перевагу в умовах, що обмежені в ресурсах. Інші існуючі діагностичні методи і будь-який новий метод повинні довести, що вони можуть забезпечити таку ж легкість використання та ціну як мікроскопія з урахуванням обмежених фінансових ресурсів, зазвичай доступних у схильних до малярії регіонах. Можливо, єдиним і головним конкурентом у цьому сенсі є ШДТ. Вони виявляють докази малярійних паразитів (антигенів) і займають близько 10-15 хвилин. Їх чутливість виявлення нижче, але порівнянна з ручною мікроскопією, і вони не вимагають спеціального обладнання і вимагають лише мінімальної підготовки. Хоча ШДТ в даний час є більш дорогим, ніж мікроскопія в високонавантажених областях, актуальним є питання, чи можуть ці тести замінити мікроскопію в найближчому майбутньому. За даними Всесвітньої організації охорони здоров'я, країни використовують мікроскопію більше, ніж ШДТ. ШДТ використовуються більше у сільській місцевості, де мікроскопія недоступна. Близько 47% тестів малярії в ендемічних країнах малярії в усьому світі були зроблені за допомогою ШДТ. Однак використання ШДТ не виключає необхідності мікроскопічної перевірки на малярію. Основним недоліком є те, що ШДТ не забезпечують кількісного визначення результатів. Тому в цей

момент мікроскопія і ШДТ більш доповнюють один одного, ніж один, замінює інший.

1.3.3 Інші тести

Є кілька додаткових методів діагностики малярії. Важливими критеріями є вартість за тест, чутливість і специфічність методу, час на тест і необхідний рівень кваліфікації користувача. Крім того, кількісне визначення кількості інфікованих еритроцитів є важливим як прогностичний показник.

- Полімеразна ланцюгова реакція (ПЛР). Молекулярний метод, званий ПЛР, показав більш високу чутливість і специфічність, ніж звичайне мікроскопічне дослідження мазку периферичної крові. Фактично, це вважається найбільш точним серед усіх тестів. Він може виявляти дуже низькі концентрації паразитів у крові і може диференціювати види. Проте ПЛР є складною технологією високої вартості, яка потребує багато годин для обробки кваліфікованим персоналом. Згідно з дослідженнями, ПЛР не впроваджується в країнах, що розвиваються, через складність тестування та відсутність ресурсів для адекватного та регулярного виконання цих тестів. Контроль якості та технічне обслуговування також мають важливе значення для техніки ПЛР, так що він може не підходити для діагностики малярії у віддалених сільських районах або навіть у звичайних клінічних діагностичних умовах.
- Флуоресцентна мікроскопія. Кількісне буферне покриття є лабораторним тестом для виявлення інфекції малярії або інших паразитів крові, використовуючи флуоресцентну мікроскопію. Флуоресцентний барвник робить паразити видимими під ультрафіолетовим світлом. За даними дослідження Adeoye і Nga, цей тест є більш чутливим, ніж звичайний товстий мазок. Сьогодні комерційно доступні портативні флуоресцентні мікроскопи з флуоресцентним реагентом для маркування паразитів. Хоча методика

кількісного покриття є простою, надійною та зручною для користування, вона вимагає спеціалізованої апаратури, є більш дорогою, ніж звичайна світлова мікроскопія, і не може визначити види і кількість паразитів.

- Проточна цитометрія. Це лазерний метод підрахунку та виявлення клітин, що дозволяє профілізувати тисячі клітин в секунду. Незважаючи на те, що проточна цитометрія пропонує автоматизований підрахунок кількісті паразитів, це компенсується досить низькою чутливістю. Цей метод менш підходить в якості діагностичної методики в польових умовах, коли для прийняття рішень щодо лікування потрібна пряма відповідь. Проте в розвинених країнах вона може застосовуватися в клінічних умовах для точного підрахунку чисельності паразитів, наприклад, у подальшому лікуванні.

1.4 Методи фарбування мазків крові

Більше 100 років тому вперше для діагностики малярії було застосовано пляму Giemsas (1902). З тих пір вона отримувала підвищену увагу. Через низьку вартість, високу чутливість і специфічність, в наш час широко застосовується в мікроскопічних дослідженнях малярії. Однак фарбування Giemsa вимагає численних реагентів, досвідченого персоналу, і є трудомістким і забирає багато часу (зазвичай потрібно щонайменше 45 хвилин для забарвлення одного слайду). Використовувалися також інші плями, такі як Полева(Field) пляма, що значно скорочує час фарбування, хоча вона вимагає сушіння зразків до і під час фарбування. Однак є і недоліки з плямами Поля, особливо у медичних центрах з обмеженими ресурсами, в яких пляма може бути використана. Погана підготовка крові часто призводить до генерації артефактів, які зазвичай приймають за малярійних паразитів, таких як бактерії, гриби, плями, забруднення та залишки клітин. Вони часто можуть викликати помилкові позитивні показники.

Іншим плямою є пляма Лейшмана (1901), яка має високу чутливість, є дешевою і порівняно легко виконувати. Серед інших використовуваних плям є, наприклад, пляма Райт-Гімса, яка являє собою комбінацію плями Райт і Гімса, і коли перша полегшує диференціацію типів клітин крові.

У 1970-х рр. Sodeman досліджували вплив флуорохромного фарбування у виявлення малярійних паразитів при інфекції низького рівня. Було показано, що флуорохромне фарбування є більш чутливим і витрачає менше часу, ніж за методами фарбування Романовського і Гімзе, але потребує значної практики і професійної підготовки, а також страждає від артефактів, в тому числі і від фотознебарвлення та фототоксичности. Крім того, флуоресцентні мікроскопи є більш дорогими, ніж стандартні світлові мікроскопи, що є важливим фактором у тропічних, бідних на ресурси регіонів, де малярія є ендемічною. Таблиця 2 показує типи мазків крові та методи фарбування, використані для підходів, опублікованих у літературі. Зрозуміло, що переважна більшість публікацій була для тонких мазків. Звичайно, одна з причин цього полягає в тому, що тонкі мазки дозволяють легше визначати види паразита і стадії , крім паразитії.

Таблиця 2 – Типи мазків крові та методи фарбування для діагностики малярії

Тип мазка	Тонкий	Товстий
Фарбування	Giemsa; Leishman; Leishman-Methylene blue; Комбінація ДНК і РНК; Флуоресцентна; Wright; Флюрохромне; Romanowsky; Acridine orange (AO) ; DAPI/Mitotracker; Toluidine blue; Unstained	Giemsa; Leishman

Отже, в певному сенсі, тонкі мазки є більш універсальними і містять більше інформації. Іншою важливою причиною, ймовірно, є те, що наявність еритроцитів дає проблему виявлення паразитів більшої структури і робить проблему легшою до певної міри, тому що паразитів потрібно виявляти тільки

всередині клітин. Для товстих плівок виявлення паразитів може бути складніше через шум і фарбування артефактів, що може призвести до помилкових спрацьовувань. Тим не менш, через важливість товстих мазків для практичної діагностики малярії, дуже ймовірно, що в майбутньому буде впроваджено більше підходів для товстих плівок. Однак, якщо з’являються переконливі оптичні апаратурні рішення, що сканують декілька полів в тонких мазках і досягають чутливості, порівнянної з густими мазками, то це може бути спірним питанням. Таблиця 2 також показує, що більшість підходів, як для тонких, так і для товстих мазків , прийняли найбільш популярні плями на практиці, Giemsa. Хоча плями, такі як Leishman, дають дуже хороші результати для малярійних паразитів, пляма Giemsa виявилася найкращою всебічною плямою для рутинної діагностики малярії. Недолік цього полягає в тому, що він є відносно дорогим, але його переважує його стабільність у часі і його незмінна якість фарбування у широкому діапазоні температур.

1.5 Автоматизована діагностика малярії

Першим кроком, як правило, є придбання цифрових зображень мазків крові, що багато в чому залежить від використовуваного обладнання та матеріалів. Розділ збору зображень розбиває різні підходи для різних типів мікроскопії, слайдів крові (тонкі або товсті) і фарбування. Після отримання зображення більшість систем виконують один або кілька методів попередньої обробки для видалення шуму та нормалізації освітлення та колірних змін, притаманних процесу отримання та фарбування зображення. Секція попередньої обробки сортує публікації відповідно до реалізованих методів попередньої обробки.

Наступний крок зазвичай передбачає виявлення і сегментацію (виокремлення) окремих клітин крові і, можливо, інших об'єктів, які можуть бути видимі в зображенні слайдів крові, таких як паразити або тромбоцити.

Придбання зображення. У таблиці 3 перераховані системи відповідно до використовуваного типу мікроскопії. Оскільки світлова мікроскопія є найпоширенішою формою діагностики малярії в умовах недостатнього використання ресурсів, де автоматизація також матиме найбільший вплив на охорону здоров'я та економіку, не дивно, що більшість авторів впроваджували системи для стандартної мікроскопії.

Таблиця 3 – Методи отримання зображень малярії

Методи візуалізації
Світлова мікроскопія
Мікроскопія Binocolor
Флуоресцентна мікроскопія
Поляризована мікроскопія
Мультиспектральна і мультимодальна мікроскопія
Цитометр на основі зображення
Субпіксельна розв'язуюча оптиофлюїдна мікроскопія (SROFM)
Кількісна фазова візуалізація (QPI)
Кількісна картридж-сканерна система
Скануюча електронна мікроскопія (SEM)
Масштабування на основі волоконної раманів
Послідовна скануюча електронна мікроскопія (SBFSEM)
Сканування цифрових зображень SightDx

Попередня обробка. У таблиці 4 перераховані всі підходи попередньої обробки, які були застосовані для автоматичного аналізу цифрових зображень слайдів крові. Попередня обробка в основному застосовується для поліпшення якості зображення і для зменшення коливань зображень, що невиправдано ускладнює наступні етапи обробки. Можна виділити три основні цілі: видалення шуму, покращення контрастності, освітлення та корекція фарбування.

Для видалення шуму найпопулярнішими підходами були добре встановлені фільтри, такі як середній і медіанний фільтри, або гауссові фільтри низьких частот. Крім того, застосування морфологічних операцій є дуже популярним. Для поліпшення контрасту, найбільш популярними підходами були контрастні методи розтягування та вирівнювання гістограми. Для варіацій освітлення та фарбування застосовані методи нормалізації кольору, включаючи популярне використання кольорів у градаціях сірого.

Таблиця 4 – Методи попередньої обробки зображень, що застосовуються для поліпшення зображення мазків крові малярії

Мазок крові	Проблема	Методи попередньої обробки	Зауваження
Тонкий	Зменшення шуму	Середнє фільтрування Медіанна фільтрація Геометричне середнє фільтрування Вінерська фільтрація Гамма-вирівнювання SUSAN нелінійна фільтрація Гауссова фільтрація низьких частот	Зняти імпульсний шум і зберегти грані
		Нелінійна дифузна фільтрація Гамма-перетворення Міжшаровий ортогональний вейвлет-поріг Модель шумозаглушення Перона-Маліка Морфологічні операції	Видалить небажані невеликі об'єкти, заповнення, закривання та відкривання отворів

Продовження таблиці 4

	Низький контраст зображення	Фільтрація за Лапласіаном Адаптивне / локальне вирівнювання гістограми Пряме дискретне перетворення curvlet Методи контрастного розтягування Фільтрація низьких частот	Виявлення країв Підвищення роздільної здатності зображення Підсилення контрастності Видалення компонентів високої частоти
	Нерівномірне освітлення	Морфологічна операція «верхньої шапки»	Видалить ефект неоднорідного освітлення
	Різниця забарвлення клітин	Лінійна модель Нормалізація кольору Нормалізація кольорів сірого Медіанна фільтрація	Корекція освітленості Нормалізація колірного профілю зображення
Товстий	Зменшення шуму	Покращення контрасту Гаусовий фільтр низьких частот Вирівнювання гістограми Просторовий фільтр Лапласа	

1.6 Огляд методів виявлення і сегментації еритроцитів

Таблиця 5 показує різні методи сегментації, застосовані до тонких мазків. Переважна більшість цих методів є пороговими методами, такими як поріг Otsu у поєднанні з морфологічними операціями. Однак ці методики

можуть не домінувати через їх чудову продуктивність у порівнянні з іншими методами, а скоріше через їх відносну простоту. Інші способи включають перетворення Хофа, яке робить припущення про форму клітин крові, і безконтрольну кластеризацію пікселів k-means. Сегментація клітин повинна бути точною, щоб обчислити правильну паразитію. Однак торкання клітин, зокрема, ускладнює ідентифікацію і сегментацію окремих клітин. Для цієї проблеми застосовані такі методи, як вододіл і активні контури.

Таблиця 5 – Методи сегментації тонких мазків крові

Тип мазку крові	Метод сегментації	Пояснення
Тонкий	Поріг Otsu	Обчислює оптимальний поріг, вважаючи, що зображення містить бімодальну гістограму
	(Адаптивна) гістограма Порогу	Важко визначити порогове значення
	Поріг Zack	Метод на основі трикутника особливо ефективний зі слабким піком у гістограмі зображення
	Порогове значення розподілу Пуассона	Знаходження порога, що розділяє передній план і фон з використанням мінімальної помилки
	Морфологічна операція	Математичні морфологічні операції в т.ч. гранулометрія, відкриття, закриття тощо
	Алгоритм виявлення країв	Добре працює для високо контрастних зображень з гострими краями, фальшиві виявлення країв повинні бути відфільтровані

Продовження таблиці 5

	Hough перетворення	Потрібні червоні кров'яні клітини, розміри, включаючи радіус, форму
	К-середні	Метод неконтрольованого навчання, який ітеративно призначає пікселі К кластерам, використовуючи їх ознаки
	Алгоритм вододілу	Витягнути неперервні граничні області, але переоцінка є типовою проблемою
	Вододіл, що контролюється маркером	В основному застосовується для роздільного знаходження клітин
	Моделі активного контуру	Підходи, що базуються на рівнях, забезпечують топологічну гнучкість, дорогий обрахунок
	Сегментація, заснована на правилах	Потрібно знати форму, розмір, колір тощо
	Сегментація на основі нечітких правил	Правила побудови нелегкі, коли невизначеність висока
	Сегментація нечіткої дивергенції	
	Шаблон відповідностей	
	Адаптивна модель перетворення відстані гауссівської суміші	
	Відстань трансформації	
	Ada-boost	
	Нормалізовані алгоритми розпилу	Дороге обчислення

Таблиця 6 показує різні методи сегментації в літературі для товстих мазків. Ситуація сегментації для товстих мазків відрізняється тим, що білі

клітини крові та паразити повинні бути сегментовані. Тим не менш, білі кров'яні клітини більше, ніж червоні кров'яні клітини і мають більше текстури, що робить їх сегментацію набагато простіше. Крім того, білі кров'яні клітини просто повинні бути ідентифіковані і не підлягати подальшій обробці або класифікації. В додаток, паразити дуже малі і їхня достовірна ідентифікація є найбільш важливою. Тому виявлення цих об'єктів важливіше, ніж їх сегментація, що може знову пояснити домінування порогових методів і морфологічних операцій.

Таблиця 6 – Методи сегментації для товстих мазків крові

Тип мазку крові	Метод сегментації	Пояснення
Товстий	Поріг Otsu	Обчислює оптимальний поріг, вважаючи, що зображення містить 2 класи після бімодальної гістограми
	Поріг гістограми	Важко визначити порогове значення, зазвичай використовується з іншими методами для підвищення продуктивності
	Морфологічні операції	Математичні морфологічні операції, включаючи гранулометрію, розкриття, закриття тощо, корисні для характеристики та представлення клітин крові круглої форми, розміру, кордонів, скелетів, текстури, градієнта тощо
	Нечіткі C-середні	

Видобування і виділення ознак. У таблиці 7 наведено різні ознаки, які використовуються для опису появи червоних кров'яних клітин, інфікованих і неінфікованих, в тонких мазках. Очевидно, тому, що паразити були забарвлені, кольорові особливості є найбільш природними і дійсно

використовуються багатьма статтями. Крім того, кілька текстурних і морфологічних ознак використовувалися для опису внутрішньої частини еритроцитів. Ідея полягає в тому, що у випадку інфікованих клітин ці ознаки можуть підібрати типовий вигляд кільцевих структур з видимою цитоплазмою та іншими унікальними характеристиками паразитів. Взагалі кажучи, більшість функцій, що використовуються, є перевіреними та довіреними функціями, які вже були застосовані в інших, часто немедичних областях застосування. Наприклад, особливості текстури Хараліка, локальні бінарні структури, матриці спільної зустрічі, гістограма градієнтів і багато інших успішно використовуються в широкому діапазоні застосувань. Це також включає морфологічні особливості форми та моменти.

У першу чергу, це використання різних колірних просторів, що призводить до набору більш специфічних для малярії функцій, залежно від використовуваного колірного простору. Є цілком вагома причина використовувати інший колірний простір, більш підходящий для вилучення типових фарбувальних кольорів, які часто варіюються в діапазоні від синього або пурпурового до коричневого відтінку Кольоровий простір HSV підтримується багатьма статтями, а деякі інші статті використовують зелений канал RGB, щоб витягувати кольорову інформацію, пов'язану з фарбуванням, у сірій шкалі.

Таблиця 7 – Розрахунок особливостей класифікації малярійних паразитів у тонких мазках крові

Мазок крові	Тип ознаки	Ознака	Опис
Тонкий	Колір	RGB HSV YCbCr LAB Інтенсивність	Надає інформацію про колір

Продовження таблиці 7

	Текстура	Харалік Матриці довжини сірого Місцевий бінарний шаблон Фрактал Вейвлет перетворення Текстура градієнта Матриця збігів сірого Ентропія SIFT Багаторівневий лапласіан Гаусса і Габора	Характеризують загальну форму і розмір еритроцита без врахування щільності
	Морфологічна	Форма (площа, периметр, компактність, ексцентричність тощо) Моменти(нульові, центральні) Площа гранулометрии	Кодує просторовий розподіл інтенсивності в конкретній області

Таблиця 8 показує особливості, що використовуються для товстих мазків. Взагалі, для товстих мазків використовуються подібні, якщо не ідентичні, особливості в порівнянні з тими, що використовуються для тонких мазків, експериментуючи з встановленими ознаками, а також з різними колірними просторами. Обчислюється великий набір багатьох різних функцій, а потім з практичних причин скорочують ці особливості, вибираючи підмножину ознак з самою відмінною рисою, використовуючи стратегії вибору функцій. Зокрема, методи вибору ознак, що використовуються для зменшення розмірності ознак, включають аналіз основних компонентів, F-статистику, 1-положення дисперсійного аналізу, інформаційний приріст, і підтримку векторного рекурсивного усунення рекурсивних функцій. Проте

такі класичні підходи до обчислення функцій та відбору піддаються серйозній небезпеці замінити їх за допомогою методів, які не покладаються на ручні функції, такі як глибоке навчання.

Таблиця 8 – Розрахунок особливостей класифікації малярійних паразитів у товстих мазках крові

Мазок крові	Тип ознаки	Ознака	Опис
Товстий	Колір	RGB HSV LAB Інтенсивність	Надає інформацію про колір
	Текстура	Харалік	Характеризують загальну форму і розмір еритроцита без врахування щільності
	Морфологічна	Форма (площа, периметр, коефіцієнт компактності) Момент (нульовий, центральний)	Кодує просторовий розподіл інтенсивності в конкретній області

1.7 Класифікаційні методи для діагностики малярії

У таблиці 9 перелічені всі класифікаційні методи, які використовувалися для розрізнення між інфікованими і неінфікованими червоними кров'яними клітинами в тонких мазках або виявлення паразитів у товстих мазках. Практично всі класифікаційні методи, популярні за останнє десятиліття, були застосовані до діагностики малярії, починаючи з дерев рішень і базових штучних нейронних мереж для машин підтримки векторів до класифікаторів випадкових дерев. Більшість знань, що стосуються малярії, полягають у

взаємодії сегментації, особливостей та класифікації. Порівняння продуктивності опублікованих систем дуже важко. Системи оцінювалися на предметних слайдах з зовсім іншого походження з значною мірою різними параметрами для отримання зображення та підготовки слайдів. Дуже часто набір оцінок є занадто малим або занадто обмеженим, щоб дозволити зробити заяву про загальну продуктивність системи. В даний час не існує жодного загальнодоступного набору показників зображень, малих чи великих, які могли б бути використані для справедливого порівняння систем. Тому, хоча багато статей повідомляють про досить високі показники продуктивності з

Таблиця 9 – Методи класифікації

Мазок крові	Методологія класифікації	
Тонкий	Безконтрольний	Кластеризація К-середні Кватернієве перетворення Фур'є
	Контрольований	Порогове значення Байєсівський класифікатор Метод співвідношення кільцевого кільця Просте байєсове дерево Логістична регресія дерева Лінійне програмування Класифікатор евклідової відстані Класифікатор К-найближчі сусіди Дерево рішень Шаблон відповідності Ada-boost Найближчий середній класифікатор Система нечіткого інтерфейсу Нормалізована взаємна кореляція

Продовження таблиці 9

		Машина опорного вектора Лінійний дискримінант Натовп джерел ігор Нейронна мережа Глибоке навчання
Товстий	Безконтрольний	Кластеризація К-середні
	Контрольований	Просте байєсове дерево Класифікатор випадкового дерева Найближчий середній класифікатор Порогове значення Машина опроного вектора Нейронна мережа Генетичний алгоритм

точки зору точності, чутливості, специфічності я вважаю, що краще не порівнювати ці числа у даній роботі. Спостерігається компроміс між робочими характеристиками конвеєра та його точністю. Як правило, при збільшенні точності техніки її обчислювальна складність все ж зростає. Наприклад, складні методи набору рівнів для сегментації комірок виконують краще, ніж Otsu, але також вимагають довшого часу виконання. Крім того, обчислення функцій може вплинути на ефективність системи. Тому в деяких статтях застосовуються способи вибору ознак для зменшення розмірності ознак і вилучення недискримінаційних ознак, які можуть покращити точність і ефективність. Нарешті, час виконання класифікації клітин залежить від використовуваної архітектури класифікації. Більшість систем виконуватиме свої завдання набагато разів швидше, ніж мікроскопіст або, принаймні, виконуватиметься швидше, ніж людина після невеликої оптимізації їх

реалізації. У поєднанні з програмним забезпеченням це дозволить повністю автоматизувати процес сканування слайдів таким чином, щоб мікроскопісту не потрібно було переміщати посуд для мікроскопа, щоб взяти зображення наступного зразку. Це також призведе до вищої пропускної здатності, що може підвищити чутливість системи дозволяючи перевіряти більше зразків одночасно. Для поліпшення точності системи, здається, існує тенденція дотримуватися основного методу класифікації під час публікації, щоб скористатися новітньою архітектурою класифікації та поліпшенням продуктивності.

1.8 Висновки до розділу 1

У даному розділі приведено всі необхідні відомості про малярію та методи її виявлення на кожному етапі. Перераховані всі підходи попередньої обробки, які були застосовані для автоматичного аналізу цифрових зображень слайдів крові. Були розглянуті різні методи сегментації, застосовані до тонких та товстих мазків крові. Наведено різні ознаки, які використовуються для опису появи червоних кров'яних клітин та класифікаційні методи для інфікованих і неінфікованих клітин і проаналізовано наскільки автоматизація прискорить отримання результатів.

Отже, автоматизація процесу діагностування забезпечить точну діагностику захворювання, що буде корисним для охорони здоров'я у регіонах з обмеженими ресурсами.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

2.1 Вступ

Згорткові нейронні мережі призначені для роботи з сітково-структурованими входами, які мають сильні просторові залежності в локальних регіонах сітки. Найбільш очевидний приклад сітко-структуровані дані - це двовимірне зображення. Цей тип даних також демонструє просторові залежності, оскільки сусідні просторові розташування у зображенні часто мають подібні значення кольору окремих пікселів. Додатковий вимір захоплює різні кольори, що створює тривимірний вхідний обсяг . Тому особливості у згортковій нейронній мережі мають залежності між собою на основі просторових відстаней. Інші форми послідовних даних - текст, часові ряди і послідовності також можуть розглядатися як особливі випадки структурованої сітки даних з різними типами відносин між суміжними елементами. Переважна більшість застосувань згорткових нейронних мереж фокусуються на даних зображень, хоча також можна використовувати ці мережі для всіх типів тимчасових, просторових і просторово-часових даних.

Важливою властивістю даних зображення є те, що вони мають певний рівень перекладу інваріантності , чого не має в багатьох інших типах даних, структурованих сітками. Згорткові нейронні мережі мають тенденцію створювати подібні значення характеристик з локальних регіонів з подібних шаблонів. Однією з переваг даних зображення є те, що ефекти конкретних входів на представлення ознак часто можна описати інтуїтивно.

Важливою визначальною характеристикою згорткових нейронних мереж є операція, яка називається згорткою . Операція згортки - це операція з дот-продуктом між сітковим набором ваг та подібними вхідними структурами, побудованими на основі сітки . Цей тип операції корисний для даних з високим рівень просторової або іншої місцевості, наприклад, дані зображення. Тому згорткова нервова мережі визначаються як мережі, які використовують

операцію згортки принаймні в одному шарі, хоча більшість згорткових нейронних мереж використовують цю операцію в декількох шарах.

2.2 Історична перспектива та біологічне натхнення

Згорткові нейронні мережі були однією з перших історій успіху глибокого навчання, задовго до недавніх досягнень у навчальних методах, що призвели до підвищення продуктивності в інших типах архітектур. Насправді, привабливі успіхи деяких згорткових нейронних мереж архітектури в конкурсах класифікації зображень після 2011 року привели до ширшої уваги до галузі глибокого навчання. Давні тести, такі як ImageNet з класифікацією топ-5 по показнику помилок показали, що понад 25% знизилися до менш ніж 4% в період між 2011 роком і 2015 р. Згорткові нейронні мережі добре підходять для процесу ієрархічної інженерної техніки з глибиною; це виражається в тому, що найглибші нейронні мережі у всіх областях витягуються з області згорткових мереж. Крім того, ці мережі також являють собою відмінні приклади того, як біологічно натхненні нейронні мережі можуть давати нагальні результати. Найкращі згорткові нейронні мережі сьогодні досягають або можуть перевищити продуктивність на рівні людей, що було неможливим для більшості експертів у сфері комп'ютерного зору лише кілька десятиліть тому.

Рання мотивація використання згорткових нейронних мереж була отримана з експериментів Hubel та Wiesel на зоровій корі кішки. Зорова кора має невеликі області клітин, чутливих до конкретних областей поля зору. Іншими словами, якщо конкретно ділянки поля зору збуджуються, тоді ті клітини в зоровій корі будуть активовані. Крім того, збуджені клітини також залежать від форми та орієнтації об'єктів у полі зору. Наприклад, вертикальні краї викликають збудження деяких нейрональних клітин, тоді як горизонтальні ребра викликають збудження інших нейрональних клітин. Клітини з'єднуються, використовуючи шарувату архітектуру, і це відкриття

призвело до припущення, що ссавці використовують ці різні шари створюючи частини зображень на різних рівнях абстракції. З точки зору машинного навчання, цей принцип подібний до принципу ієрархічної функції екстракції. На основі цих біологічних натхнень, найраніша нейронна модель була neocognitron. Проте між цією моделлю і сучасною було кілька відмінностей. Найбільш видатним з цих відмінностей було те, що поняття розподілу ваги не використовувався. Виходячи з цієї архітектури, однією з перших повністю згорткових архітектур була розроблена LeNet. Ця мережа використовувалася банками для ідентифікації рукописних цифр на чеках. З тих пір згорткова нейронна мережа значно не розвинулася; головна відмінність полягає у використанні більшої кількості шарів та функції активації, такі як ReLU. Крім того, численні трюки навчання і потужні апаратні опції доступні для досягнення кращого успіху у навчанні при роботі з глибокими мережами і великими наборами даних. Фактор, який відігравав важливу роль у підвищенні популярності згорткових нейронних мереж - щорічний конкурс ImageNet. Один з найбільш ранніх методів досягнення успіху в конкурсі ImageNet 2012 року був AlexNet. Крім того, було досягнуто надзвичайно велике поліпшення точності. Незважаючи на те, що переважна більшість привабливих показників продуктивності відбулися з 2012 по 2015 рік, архітектурні відмінності між останніми переможцями і деякі з найбільш ранніх згорткових нейронних мереж досить малі, принаймні, на концептуальному рівні.

2.2.1 Більш широкі спостереження про згорткові нейронні мережі

Секрет успіху будь-якої нейронної архітектури полягає в тому, щоб пристосувати структуру мережі з семантичним розумінням даного домену. Згорткові нейронні мережі в значній мірі базуються на цьому принципі, оскільки вони використовують розріджені з'єднання з високим рівнем обміну параметрами в домен-чутливому способі. Іншими словами, не всі стани в

певному шарі пов'язані з тими, що були в попередньому шарі, безрозбірним чином. Скоріше, значення ознаки в певному шарі з'єднане тільки з локальною просторовою областю в попередньому шарі з узгодженим набором спільних параметрів по всьому просторовому відбитку зображення. Такий тип архітектури можна розглядати як регуляризацію з урахуванням доменів, що випливає з біологічних уявлень про ранні роботи Хубеля та Візеля. Загалом, успіх згорткової нейронної мережі має важливі уроки для інших областей даних. Ретельно розроблена архітектура, в якій взаємозв'язки і залежності між елементами даних використовуються для того, щоб зменшити параметр footprint, забезпечує ключ до результатів високої точності. Значний рівень дозвільної регуляризації також доступний в рекурентних нейронних мережах, які поділяють параметри з різних часових періодів. Цей обмін базується на припущенні, що тимчасові залежності залишаються інваріантними з часом. Рекурентні нейронні мережі засновані на інтуїтивному розумінні тимчасових відносин, тоді як згорткові нейронні мережі засновані на інтуїтивному розумінні просторових відносин. Остання «інтуїція» була безпосередньо витягнута з організації біологічних нейронів у зоровій корі кішки. Цей видатний успіх є мотивацією для вивчення того, як неврологія може бути використана для розробки нейронних мереж у розумних способах. Хоча штучні нейронні мережі є лише карикатурами на справжню складність біологічного мозку, але не слід недооцінювати інтуїцію, яку можна отримати, вивчаючи основні принципи неврології.

2.3 Основна структура згорткової нейронної мережі

У згорткових нейронних мережах стани в кожному шарі розташовані за структурою просторової сітки. Ці просторові відносини успадковуються від одного шару до іншого, оскільки кожне значення ознаки базується на невеликій локальній просторовій області в попередньому шарі. Важливо зберегти ці просторові відносини між клітинами сітки, оскільки операція

згортки і перетворення наступного шару критично залежать від цих відносин. Кожен шар у згортковій мережі являє собою 3-мірну структуру сітки, яка має висоту, ширину і глибину. Глибину шару у згортковій нейронній мережі не слід плутати з глибиною самої мережі. Слово "глибина" (коли використовується в контексті одного шару) відноситься до числа каналів у кожному шарі, таких як кількість основних каналів кольору (наприклад, синій, зелений і червоний) у вхідному зображенні або кількість карт особливостей у прихованих шарах. Використання слова «глибина» для позначення як кількості карт у кожному шарі, так і кількості шарів - це невдале перевантаження термінології, що використовується у згорткових мережах. Згорткова нейронна мережа працює так само, як традиційна нейронна мережа, за винятком того, що операції в її шарах просторово організовані з розрідженими (і ретельно розробленими) зв'язками між шарами. Три типи шарів, які зазвичай присутні у згортковій нейронній мережі, - це згортка, об'єднання, і ReLU. Активація ReLU нічим не відрізняється від традиційної нейронної мережі. Крім того, кінцевий набір шарів часто повністю з'єднаний і відображається в конкретному додатку способом до набору вихідних вузлів. Вхідні дані для згорткової нейронної мережі організовані в 2-мірну структуру сітки, і значення окремих точок сітки називаються пікселями. Таким чином, піксель відповідає просторовому розташуванню всередині зображення. Однак для того, щоб кодувати точний колір пікселя, нам потрібен багатовимірний масив значень на кожному місці розташування сітки. У колірній схемі RGB ми маємо інтенсивність трьох основних кольорів, відповідаючих червоному, зеленому і синьому, відповідно. Отже, якщо просторові розміри зображення становлять 32 × 32 пікселя, а глибина - 3 (що відповідає колірним каналам RGB), то загальне число пікселів у зображенні становить 32 × 32 × 3. Цей розмір зображення цілком достатній та загальноприйнятий. Приклад цієї організації показаний на рисунку 2 (а). Природно представляти вхідний шар у цій 3-мірній структурі, оскільки для просторових відносин присвячено два виміри, а третій вимір – незалежний від властивостей уздовж цих каналів.

Наприклад, інтенсивності основних кольорів є незалежними властивостями в першому шарі. У прихованих шарах ці незалежні властивості відповідають різним типам фігур, витягнутих з локальних областей зображення. Для цілей обговорення припустимо, що вхід у q-й шар має розмір $L_q \times B_q \times d_q$. Тут L_q позначає висоту (або довжину), B_q - ширину, а d_q - глибину. Майже у всіх додатках, орієнтованих на зображення, значення L_q і B_q однакові. Для першого (вхідного) шару ці значення визначаються характером вхідних даних та їх попередньою обробкою. У наведеному вище прикладі значеннями є L_1 = 32, B_1= 32 і d_1 = 3. Наступні шари мають точно таку ж тривимірну організацію, за винятком того, що кожна з d_q 2-мірної сітки значень для конкретного входу більше не може вважатися сіткою необроблених пікселів. Крім того, величина d_q значно перевищує три для прихованих шарів, оскільки кількість незалежних властивостей даної локальної області, що мають відношення до класифікації, може бути досить значним. Для q> 1 ці сітки значень називаються картами ознак або картами активації. Ці значення аналогічні значенням у прихованих шарах у мережі перенаправлення.

У згортковій нейронній мережі параметри організовані в набори 3-мірних структурних одиниць, відомих як фільтри або ядра. Фільтр зазвичай квадратний з точки зору його просторових розмірів, які зазвичай набагато менше, ніж у шару, до якого застосовується фільтр. З іншого боку, глибина фільтра завжди є такою ж, як і в шарі, до якого він застосовується. Припустимо, що розміри фільтра в q-му шарі $F_q \times F_q \times d_q$. Приклад фільтра з F1 = 5 і d1 = 3 зображений на рисунку 2 (а). Загальне значення Fq має бути малим і непарним. Прикладами часто використовуваних значень Fq є 3 і 5, хоча є деякі цікаві випадки, в яких можна використовувати Fq = 1. Операція згортки поміщає фільтр на кожне можливе положення у зображенні (або в прихованому шарі), так що фільтр повністю перекривається з зображенням і виконує точковий продукт між параметрами $F_q \times F_q \times d_q$ у фільтрі і відповідною сіткою у вхідний обсяг (з однаковим розміром $F_q \times F_q \times d_q$). Точковий виріб виконується шляхом обробки записів у відповідній 3-мірній

області вхідного обсягу і фільтра як векторів розміру $F_q \times F_q \times d_q$, так що елементи в обох векторах упорядковані на основі їх відповідних позицій у сітці. Скільки можливих позицій для розміщення фільтра? Іншими словами, кількість вирівнювань між фільтром і зображенням визначає просторову висоту і ширину наступного прихованого шару. Відносні просторові положення ознак у наступному шарі визначаються на основі відносних положень верхнього лівого кута відповідних просторових сіток у попередньому шарі. При виконанні згортки у q-му шарі можна вирівняти фільтр на $L_{q+1} = (L_q - L_{q+1})$ позиціях вздовж висоти і $B_{q+1} = (B_q - F_{q+1})$ по ширині зображення (без частини фільтра «виступає» з меж зображення). Це призводить до $L_{q+1} \times B_{q+1}$ можливих точкових продуктів, що визначає розмір наступного прихованого шару. Таким чином: L2 = 32−5 + 1 = 28, B2=32−5+1=28 Наступний прихований шар розміром 28 × 28 вибирається на рисунку 2 (а). Цей прихований шар також має глибину розміру d2 = 5. Глибина досягається за допомогою 5 різних фільтрів з власними незалежними наборами параметрів. Кожен з цих 5 наборів просторово розташованих функцій, отриманих з виходу одного фільтра, називається картою ознак. Зрозуміло, що збільшене число карт властивостей є результатом більшої кількості фільтрів (тобто відбитків параметрів), що є $F_q^2 d_q d_{q+1}$ для q-го шару. Кількість фільтрів, що використовуються в кожному шарі, контролює ємність моделі, оскільки вона безпосередньо контролює кількість параметрів. Крім того, збільшення кількості фільтрів у певному шарі збільшує кількість карт об'єктів (тобто глибини) наступного шару. Можливо, щоб різні шари мали дуже різні числа карт властивостей, залежно від кількості фільтрів, які ми використовуємо для операції згортки в попередньому шарі. Наприклад, вхідний шар, як правило, має тільки три колірних канали, але для кожного з пізніх прихованих шарів можна мати глибину (тобто кількість карт об'єктів) більше 500. Кожен фільтр намагається ідентифікувати конкретний тип просторового малюнка в невеликій прямокутній області зображення, і тому велика кількість фільтрів необхідна для захоплення широкого спектру

можливих форм, які об'єднуються для створення кінцевого зображення (на відміну від випадку вхідного шару, в якому достатньо трьох каналів RGB). Як правило, пізніші шари, як правило, мають менший просторовий відбиток, але більшу глибину з точки зору кількості характеристичних карт. Наприклад, фільтр, показаний на рисунку 2 (b), являє собою горизонтальний детектор країв на зображенні у градаціях сірого з одним каналом. Як показано на рисунку 2 (b), результуюча функція буде мати високу активацію в кожному положенні, де видно горизонтальний край. Повністю вертикальний край дає нульову активацію, тоді як похилий край може давати проміжну активацію.

Отже, ковзання фільтра скрізь на зображенні вже виявить кілька ключових контурів зображення в одній карті об'єкта вихідної ємності. Кілька фільтрів використовуються для створення вихідної гучності з більш ніж однією картою об'єктів.

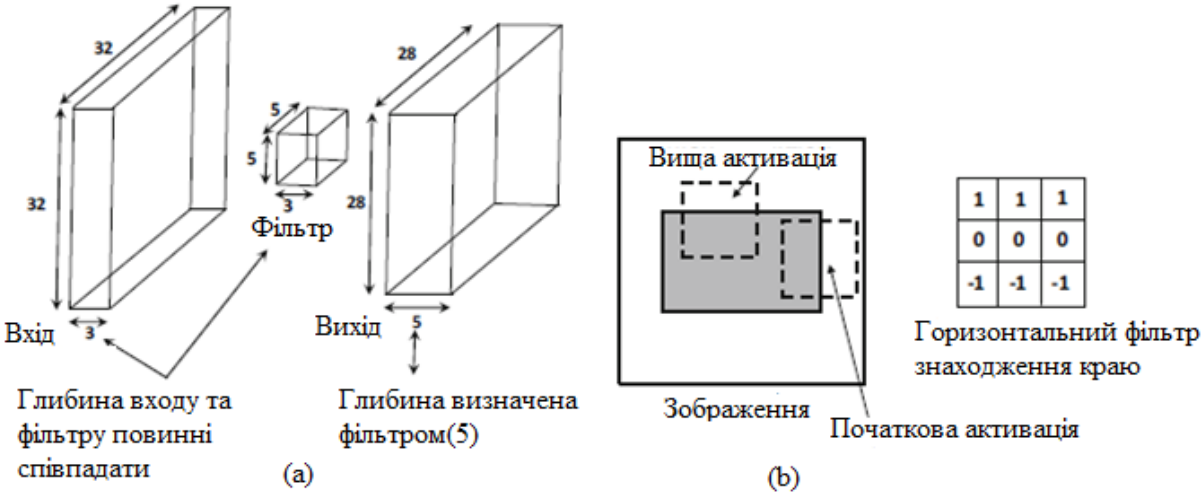


Рисунок 2 (a) Згортка між вхідним шаром розміром 32 × 32 × 3 і фільтром розміром 5 × 5 × 3 виробляє вихідний шар з просторовими розмірами 28 × 28. Отриманий результат залежить від кількості окремих фільтрів, а не від розмірів вхідного шару або фільтра. (b) Ковзний фільтр навколо зображення намагається знайти конкретний ознаку в різних вікнах зображення.

Визначаємо операцію згортки зображену на рисунку 3. P-фільтр у q-му шарі має параметри, позначені 3-мірним тензором $W^{(p,q)} = [w^{(p,q)}]$. Індекси i, j, k позначають положення вздовж висоти, ширини і глибини фільтра. Карти

характеристик у q-му шарі представлені 3-мірним тензором $H^{(q)} = [h^{(q)}]$. Коли значення q дорівнює 1, спеціальний випадок, що відповідає позначенню $H^{(1)}$, просто являє собою вхідний шар (який не є прихованим). Тоді згорткові операції від q-го шару до (q +1) -го шару визначаються наступним чином:

$$h_{ijp}^{(q+1)} = \sum_{r=1}^{F_q}\sum_{s=1}^{F_q}\sum_{k=1}^{F_q} w_{rsk}^{(p,q)} h_{i+r-1,j+s-1,k}^{(q)}, \forall i \in \{1 \dots, L_q - F_q + 1\}$$

$$\forall j \in \{1 \dots, B_q - F_q + 1\}$$

$$\forall p \in \{1 \dots, d_{q+1}\}$$

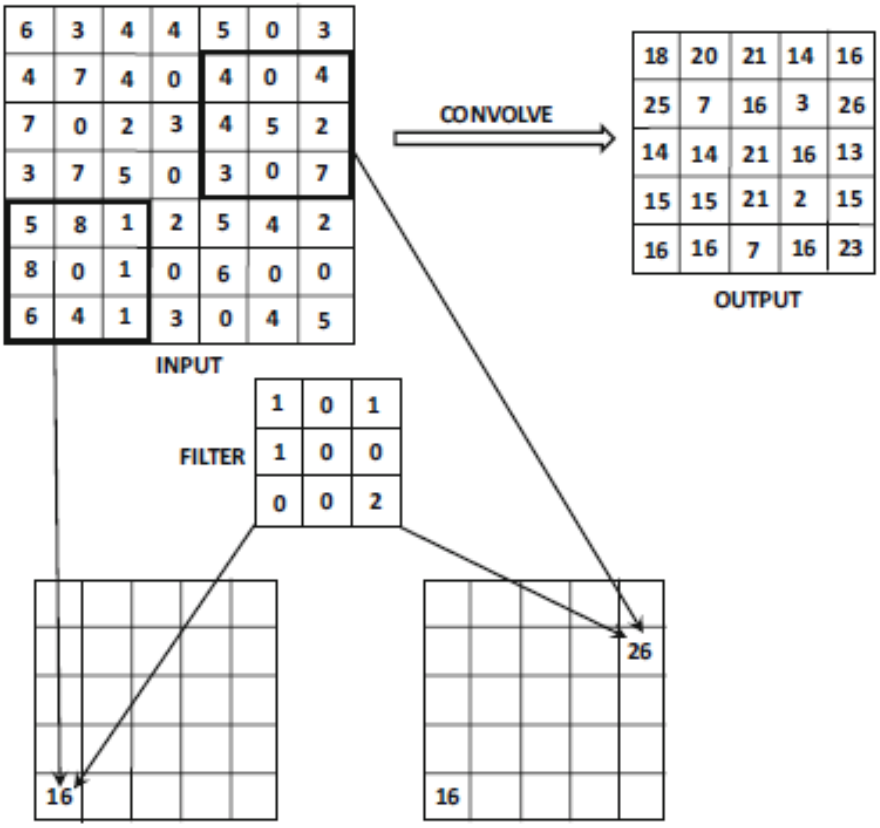


Рисунок 3 – Приклад згортки між входом 7 × 7 × 1 і фільтром 3 × 3 × 1 з кроком 1

Операція згортки нагадує експерименти Хубеля та Візеля, які використовують активації в малих областях поля зору для активації окремих нейронів. У випадку згорткових нейронних мереж це візуальне поле

визначається фільтром, який застосовується до всіх розташувань зображення для виявлення присутності форми в кожному просторовому розташуванні. Крім того, фільтри в більш ранніх шарах виявляють більш примітивні форми, тоді як фільтри в наступних шарах створюють більш складні композиції цих примітивних форм. Одним з властивостей згортки є те, що вона показує еквівалентність перекладу. Іншими словами, якщо ми змістили значення пікселів у вхідних даних у будь-якому напрямку на одну одиницю, а потім застосували згортку, відповідні значення ознак зміщуються з вхідними значеннями. Це пояснюється спільними параметрами фільтра по всій згортці. Причина обміну параметрами всієї згортки полягає у тому, що наявність певної форми в будь-якій частині зображення повинно бути оброблено таким же чином незалежно від його конкретної просторової локалізації

2.3.1 Додавання(Padding)

Одне спостереження полягає в тому, що операція згортки зменшує розмір (q + 1) -го шару порівняно з розміром q-го шару. Цей тип зменшення розміру не є бажаним взагалі, оскільки втрачається деяка інформація вздовж меж зображення (або карти об'єкта, у випадку прихованих шарів). Цю проблему можна вирішити за допомогою додавання, як приклад зображений на рисунку 4. Для доповнення можна додати$(F_q - 1) / 2$ “пікселів” навколо меж карти об'єкта, щоб зберегти просторовий слід. Встановлено значення кожного з цих значень до 0, незалежно від того, чи вводяться вхідні або приховані шари. Як результат, просторова висота і ширина вхідного обсягу збільшуватимуться за рахунок (Fq - 1), тобто, вони зменшуються (у вихідному обсязі) після згортки. Додані частини не сприяють кінцевому продукту, оскільки їх значення встановлено 0. У певному сенсі, доповнення виконує операцію згортки з частиною фільтра, що "виступає" з меж шару, а потім використовує точковий продукт тільки над частиною шару, де визначені значення.

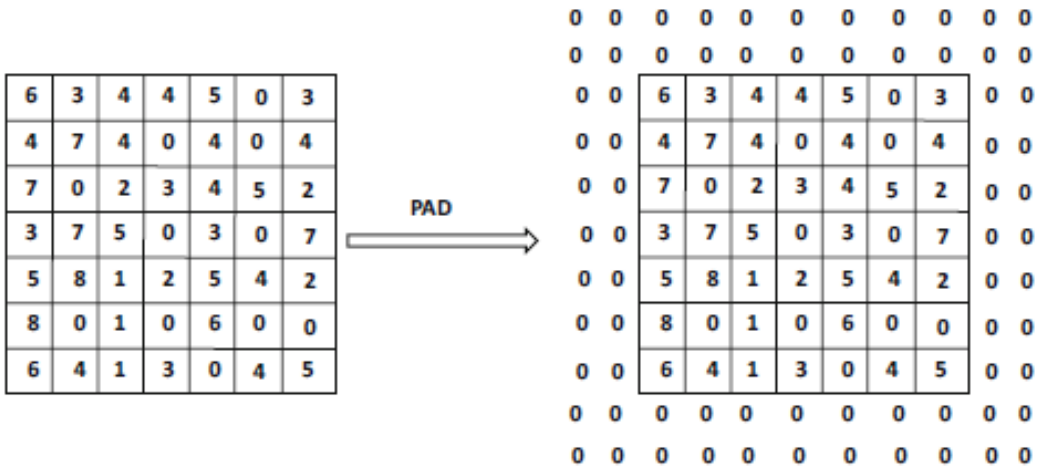


Рисунок 4 – Приклад додавання

Цей тип відступу позначається як напів-додавання, оскільки (майже) половина фільтра стирчить з усіх сторін просторового входу у випадку, коли фільтр поміщений в крайнє просторове положення вздовж країв. Напів- додавання призначений для підтримки точного просторового сліду. Коли додавання не використовується, результуюче "додавання" також називається дійсним додаванням . Дійсна обробка зазвичай не працює добре з експериментальної точки зору. Використання напів-додавання гарантує, що деяка критична інформація на кордонах шару буде представлена. У разі дійсного заповнення, розподіл пікселів на кордонах шару буде недостатньо представлений порівняно з центральними пікселями в наступному прихованому шарі, що небажано. Крім того, це недостатнє представництво поширюються на кілька шарів. Таким чином, додавання, як правило, виконується у всіх шарах, і не тільки в першому шарі, де просторові розташування відповідають вхідним значенням.

Іншою корисною формою заповнення є повне додавання. У повному додавання ми дозволяємо (майже) повному фільтру стирчати з різних сторін входу. Іншими словами, частина фільтра розміром Fq-1 дозволяється «стирчати» з будь-якої сторони входу з перекриттям тільки однієї просторової функції. Наприклад, ядро і вхідне зображення можуть перекриватися в одному пікселі в крайньому куті. Таким чином, вхідний сигнал заповнюється нулями (F_q - 1) на кожній стороні. Іншими словами, кожен просторовий розмір

вхідного сигналу збільшується на 2*(F_q - 1). Тому, якщо вхідні розміри у вихідному зображенні - Lq і Bq, об'ємні просторові розміри у вхідному обсязі стають Lq +2 (Fq −1) і Bq +2 (Fq −1). Після виконання згортки розміри карти особливостей у шарі (q + 1) стають Lq + Fq − 1 і Bq + Fq − 1 відповідно. Хоча згортка зазвичай знижує просторовий слід, повне додавання збільшує просторовий слід. Повне додавання збільшує кожен вимір просторового сліду тим самим значенням (Fq −1), що відсутність заповнення зменшує його. Цей зв'язок не є випадковим, оскільки «зворотню» операцію згортки можна здійснити, застосувавши іншу згортку на повністю просунутому виході (початкової згортки) з відповідним ядром того ж розміру. Цей тип «зворотної» згортки відбувається часто в алгоритмах зворотного поширення і автокодерів для згорткових нейронних мереж. Повністю додані входи є корисними, оскільки вони збільшують просторовий слід, який необхідний для декількох типів згорткових автокодерів.

2.3.2 Кроки

Є інші способи, в яких згортка може зменшити просторовий слід зображення (або прихованого шару). Вищенаведений підхід виконує згортку на кожній позиції в просторовому розташуванні карти об'єкта. Однак не потрібно виконувати згортку в кожному просторовому положенні в шарі. Можна знизити рівень зернистості згортки, використовуючи поняття кроків. При використанні кроку Sq у q-му шарі згортка виконується в місцях 1, Sq + 1,2Sq +1 і так далі вздовж обох просторових розмірів шару. Просторовий розмір виходу при виконанні цієї згортки має висоту (Lq - Fq) / Sq +1 і ширину (Bq −Fq) / Sq +1. В результаті використання кроків призведе до зменшення кожного просторового розміру шару на коефіцієнт приблизно Sq, а площа на S_q^2, хоча дійсний фактор може змінюватися внаслідок крайових ефектів. Більш великі кроки можуть бути корисними в налаштуваннях, обмежених пам'яттю, або для зменшення перенасичення, якщо просторова роздільна здатність зайво

висока. Штрихи мають ефект швидкого збільшення рецептивного поля кожної ознаки в прихованому шарі, одночасно зменшуючи просторовий слід всього шару. Збільшене сприйнятливе поле корисне для того, щоб захопити складну функцію у більшій просторовій області зображення. Інженерний процес ієрархічної особливості згорткової нейронної мережі захоплює більш складні форми у пізніших шарах.

2.3.3 Типові налаштування

У більшості параметрів звичайно використовуються розміри кроків = 1. Навіть коли використовуються кроки, використовуються малі розміри величини = 2. Більш того, вони є компромісним Lq = Bq. У випадках, коли вхідні зображення не є квадратними, попередня обробка використовується для забезпечення цієї властивості. Наприклад, можна витягти квадратні форми зображення для створення навчальних даних. Кількість фільтрів у кожному шарі часто є потужністю 2, оскільки це часто призводить до більш ефективної обробки. Такий підхід також призводить до глибин прихованих шарів, які мають потужність 2. Типові значення просторової протяжності розміру фільтра (позначаються Fq) дорівнюють 3 або 5. Загалом, малі розміри фільтрів часто забезпечують найкращі результати, хоча деякі практичні результати. існують проблеми з використанням малих розмірів фільтрів. Малі розміри фільтрів зазвичай призводять до більш глибоких мереж (для одного і того ж параметра параметрів) і тому мають тенденцію бути більш потужними.

Як і у всіх нейронних мережах, також можна додати упередження до прямих операцій. Кожен унікальний фільтр в шарі пов'язаний з власним зміщенням. Тому p-ий фільтр у q-му шарі має зміщення b(p, q). Коли будь-яка згортка виконується з фільтром p у q-му шарі, значення b (p, q) додається до точкового продукту. Використання зміщення просто збільшує кількість параметрів у кожному фільтрі на 1, і тому не є суттєвими великими витратами. Як і всі інші параметри, зміщення використовується під час алгоритму

зворотного поширення помилки. Зміщення можна розглядати як вагу з'єднання, вхід якого завжди встановлюється на +1. Цей спеціальний вхід використовується у всіх згортках, незалежно від просторового розташування згортки. Отже, можна припустити, що у вхідних даних з'являється спеціальний піксель, значення якого завжди дорівнює 1. Таким чином, кількість вхідних ознак у q-му шарі становить $1 + L_q \times B_q \times d_q$.

2.3.4 Шар ReLU

Операція згортки чергується з операціями об'єднання і ReLU. Активація ReLU не сильно відрізняється від того, як вона застосовується в традиційній нейронній мережі. Для кожного значення $L_q \times B_q \times d_q$ в шарі до нього застосовується функція активації ReLU для створення порогових значень $L_q \times B_q \times d_q$. Ці значення потім передаються на наступний шар. Таким чином, застосування ReLU не змінює розміри шару, тому що це просте однотонне відображення значень активації. У традиційних нейронних мережах функція активації поєднується з лінійним перетворенням з матрицею ваг для створення наступного шару активацій. Аналогічним чином, ReLU зазвичай слідує операції згортки (яка є грубим еквівалентом лінійного перетворення в традиційних нейронних мережах). У минулих роках використовувалися насичуючі активаційні функції, такі як сигмоподібний і tanh. Проте використання ReLU має величезні переваги перед цими активаційними функціями як з точки зору швидкості, так і точності.

2.3.5 Пулінг (Pooling)

Операція об'єднання(пулінг) об'єктів працює на малих областях сітки розміром $P_q \times P_q$ в кожному шарі і робить інший шар з однаковою глибиною (на відміну від фільтрів). Для кожної квадратної області розміру $P_q \times P_q$ в кожній з карт активації d_q повертається максимум цих значень. Цей підхід

називається max-pooling. Якщо крок 1 використовується, то це дасть новий шар розміру $(L_q - P_q + 1) \times (B_q - P_q + 1) \times d_q$. Тим не менш, більш поширеним є використання кроку $S_q > 1$ у пулінгу. У таких випадках довжина нового шару буде $(L_q - P_q) / S_q + 1$, а ширина буде $(B_q - P_q) / S_q + 1$. Тому об'єднання об'єктів різко зменшує просторові розміри кожної карти активації. На відміну від операцій згортки, об'єднання здійснюється на рівні кожної карти активації. У той час як операція згортки одночасно використовує всі карти властивостей d_q у поєднанні з фільтром, щоб створити єдине значення функції, об'єднання об'єктів незалежно працює на кожній карті властивостей для створення іншої карти об'єкта. Отже, операція пулінгу не змінює кількість карт об'єктів. Іншими словами, глибина шару, створеного за допомогою пулу, є такою ж, як і шару, на якому була виконана операція об'єднання(пулінгу). Приклади об'єднання з кроками 1 і 2 показані на рисунку 5. Тут ми використовуємо об'єднання над 3 × 3 регіонами.

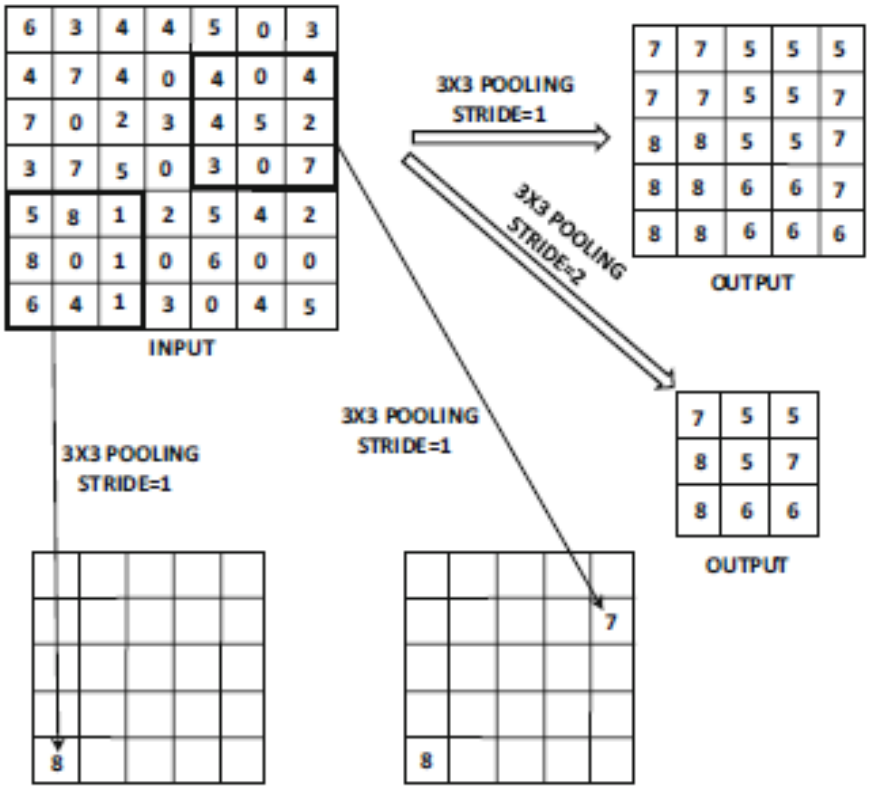


Рисунок 5 – Приклад максимального об'єднання однієї карти активації розміром 7 × 7 з кроками 1 і 2.

Крок 1 створює карту активації 5 × 5 з сильно повторюваними елементами через максимізацію у перекритих областях. Крок 2 створює карту активації 3 × 3 з меншим перекриттям. На відміну від згортки, кожна карта активації обробляється незалежно і тому кількість вихідних карт активації рівно дорівнює кількості вхідних карт активації.

Інші типи пулінгу (наприклад, середній пулінг) можливі, але рідко використовуються. У ранній згортковій мережі, що називалася LeNet-5, був використаний варіант середнього пулу і був згаданий як subsampling. Взагалі, макс-пулінг більш популярний, ніж середній пул. Макс-пулінг шари об'єднуються з згортковими / ReLU шарами, хоча перше зазвичай відбувається набагато рідше в глибоких архітектурах. Це пояснюється тим, що об'єднання суттєво зменшує просторовий розмір карти об'єктів, і лише кілька операцій об'єднання необхідні для зменшення просторової карти до невеликого постійного розміру.

Загальноприйнятим є використання пулу з 2 × 2 фільтрами і кроком 2, коли бажано зменшити просторовий слід карт активації. Об'єднання призводить до деякої інваріантності перекладу, оскільки зміщення зображення трохи не змінює карту активації. Ця властивість називається перекладною інваріантністю. Ідея полягає в тому, що подібні зображення часто мають дуже різні відносні розташування відмінних форм в них, і інваріантність перекладу допомагає класифікувати такі зображення подібним чином.

Іншою важливою метою пулінгу є те, що він збільшує розмір сприйнятливого поля при одночасному зменшенні просторового сліду шару через використання великих розмірів, що перевищують 1. Збільшення розмірів рецептивних полів необхідно для того, щоб мати можливість захопити більші області зображення у складі комплексу в більш пізніх шарах. Більшість швидких скорочень просторових слідів шарів (і відповідне збільшення рецептивних полів ознак) викликані операціями об'єднання. Згортки збільшують сприйнятливе поле лише обережно, якщо крок не перевищує 1. Макс-пулінг вводить нелінійність і більшу кількість інваріантності перекладу,

у порівнянні з крокуючими звивинами. Хоча нелінійність може бути досягнута за допомогою функції активації ReLU, ключовим моментом є те, що ефекти max-pooling не можуть бути точно відтворені за допомогою кроків згорток. Принаймні, ці дві операції не є повністю взаємозамінними.

2.3.6 Повнозв’язні шари

Кожна функція у кінцевому просторовому шарі з'єднана з кожним прихованим станом у першому повністю з'єднаному шарі. Цей шар функціонує точно так само, як традиційна мережа передачі даних. У більшості випадків для збільшення потужності обчислень до кінця можна використовувати більш ніж один повністю з'єднаний шар. Зв'язки між цими шарами точно структуровані, як традиційна мережа передачі даних. Оскільки повністю пов'язані шари щільно з'єднані, переважна більшість параметрів лежать у повністю пов'язаних шарах. Аналогічно, з'єднання від останнього просторового шару до першого повністю з'єднаного шару будуть мати велику кількість параметрів. Незважаючи на те, що згорткові шари мають більшу кількість активацій (і більший розмір пам'яті), повністю з'єднані шари часто мають більшу кількість з'єднань (і параметрів). Причиною того, що активації сприяють збільшенню кількості пам'яті, є те, що кількість активацій множиться на розмір міні-партії відстеження змінних в прямому і зворотному проходах алгоритму зворотного поширення помилки. Ці компроміси корисно мати на увазі, вибираючи склад нейронної мережі на основі конкретних типів обмежень ресурсів (наприклад, дані щодо доступності пам'яті). Вихідний шар згорткової нейронної мережі сконструйований за специфічним додатком. У такому випадку вихідний шар повністю з'єднаний з кожним нейроном в передостанньому шарі і має пов'язану з ним вагу. Можна використовувати логістичну, softmax або лінійну активацію в залежності від природи програми (наприклад, класифікації або регресії). Однією з альтернатив використання повністю пов'язаних шарів є використання середнього пулінга по всій

просторовій області кінцевого набору карт активації для створення єдиного значення. Тому кількість функцій, створених в кінцевому просторовому шарі, буде точно рівним кількості фільтрів.

2.3.7 Чергування шарів

Шари згортки, об'єднання і ReLU, як правило, перемежовуються в нейронній мережі, щоб збільшити виразну потужність мережі. Шари ReLU часто слідують за згортковими шарами, так само як нелінійна активаційна функція зазвичай відповідає лінійному точковому продукту в традиційних нейронних мережах. Отже, згорткові та ReLU шари зазвичай склеюються один за одним. Після двох або трьох наборів згорткових комбінацій ReLU можна мати шар з максимальним об'єднанням. Прикладами цієї основної моделі є наступні: CRCRP та CRCRCRP .Тут згортковий шар позначається C, шар ReLU позначається як R, а шар макс-об'єднання позначається P. Це цілий шаблон (включаючи шар макс-пулінг) може повторитися кілька разів, щоб створити глибоку нейронну мережу. Наприклад, якщо вищезгаданий шаблон повторюється три рази і слідує повністю пов'язаний шар (позначається F), то ми маємо наступну нейронну мережу: CRCRPCRCRPCRCRPF.

Шар об'єднання є ключовим кроком, який прагне зменшити просторовий відбиток карт активації, оскільки він використовує кроки, що перевищують 1. Крім того, можна зменшити просторові сліди з кроками згорток замість max-pooling. Ці мережі часто досить глибокі, і не рідко бувають згорткові мережі з більш ніж 15 шарами. Останні архітектури також використовують проміжні з'єднання між шарами, які стають все більш важливими, оскільки глибина мережі збільшується.

2.3.8 Нормалізація локальної відповіді

Нормалізація відповіді завжди використовується безпосередньо після шару ReLU. Використання цього трюку допомагає узагальненню. Основна ідея цього підходу нормалізації натхненна біологічними принципами, і вона покликана створити конкуренцію між різними фільтрами. Розглянемо ситуацію, в якій шар містить N фільтрів, і значення активації цих N фільтрів в конкретному просторовому положенні (x, y) задаються a1. . . aN. Тоді кожна a_i перетворюється в нормоване значення b_i, використовуючи наступну формулу:

$$b_i = \frac{a_i}{(k + \alpha \sum_j a_i^2)^\beta}$$

Значення базових параметрів, що використовуються, є k = 2, α = 10−4, β =0,75. Проте на практиці не нормалізуються всі N фільтри. Скоріше фільтри впорядковано довільно, щоб визначити "суміжність" серед фільтрів. Тоді нормалізацію виконують над кожним набором з n «суміжних» фільтрів для деякого параметра n. Значення n використовується - 5. Отже, ми маємо наступну формулу:

$$b_i = \frac{a_i}{(k + \alpha \sum_{j=i-[\frac{n}{2}]}^{i+[n/2]} a_i^2)^\beta}$$

У наведеній вище формулі будь-яке значення i - n / 2, яке менше 0, встановлюється рівним 0, і будь-яке значення для i + n / 2, що більше N, встановлюється в N.

2.4 Алгоритм навчання згорткової нейронної мережі

Навчанням штучної нейронної мережі називається процес підстроювання значень ваг зв'язків між нейронами мережі. У згортковій

нейронній мережі(ЗНМ) використовується навчання з учителем, що припускає використання навчальної вибірки для порівняння вихідного сигналу мережі з еталонним значенням навчальної вибірки, розрахунок помилки і підстроювання ваг зв'язків мережі з метою зменшення значення цієї помилки. У ЗНМ як алгоритм навчання використовується алгоритм градієнтного спуску або зворотного поширення помилки і його модифікації. Для повнозв'язних і згортвоих шарів розраховується значення помилки вихідного сигналу мережі за загальною формулою:

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

де E (w) - функція помилки мережі;

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

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

Навчання мережі направлено на мінімізацію функції помилки. Значення функції помилки для шару мережі визначається за загальною формулою:

$$\delta_i^{(q)} = \left(f_{ik}^{(q)}(S)\right)' \sum_j w_{ij}\, \delta_j^{(q+1)},$$

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

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

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

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

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

На основі розрахованої помилки нейрона визначається зміна ваги зв'язку цього нейрона за формулою:

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

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

η - значення, що визначає швидкість навчання;

δ_j - значення помилки j-го нейрона в шарі q;

x_i - значення і-го вхідного сигналу.

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

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

Описані в даному підрозділі типи шарів є основними при формуванні архітектури ЗНМ. Однак вони можуть розташовуватися яким завгодно чином і мати величезний розкид значень параметрів. Для того, щоб сформувати підхід до формування оптимальної архітектури згорткової мережі, необхідно проаналізувати існуючі ефективні архітектури та підходи до їх побудови.

2.5 Аналіз існуючих архітектур згорткових нейронних мереж

Першою запропонованої згортковою нейронною мережею була модель Яна Лекуна - LeNet, схема якої представлена на рисунку 6. Це була перша модель, яка містила шари, що чергуються - два рази згорткові шари і шари субдіскретізації, а також три повнозв'язних шари. Дана архітектура з параметрами, наведеними на рисунку 6, вважається класичною.

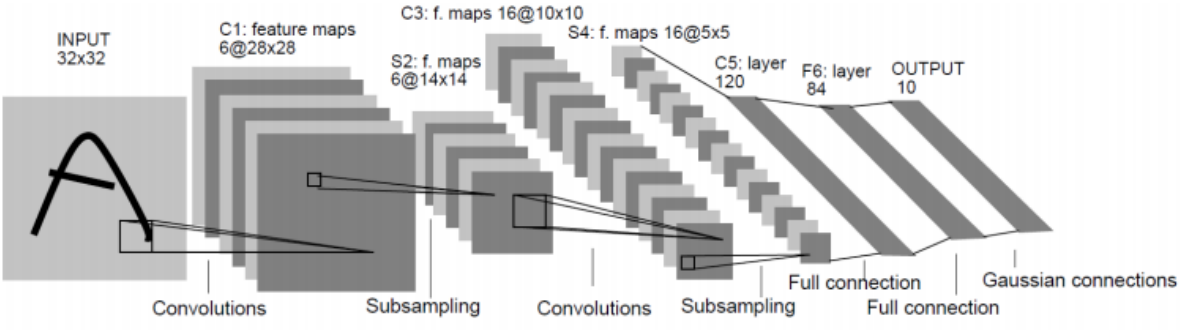


Рисунок 6 – Схема архітектури згорткової нейронної мережі LeNet

Дана архітектура без значних змін до сих пір в найчастіше використовується при вирішенні прикладних задач медичної діагностики використовуючи зображення і відеодані.

Наступною значною згортковою нейронною мережею була мережа AlexNet, запропонована Алексом Крижевський (Alex Krizhevsky). Дана мережа була дуже схожа на мережу LeNet, однак відрізнялася від неї більш масивною і складною архітектурою. Також вона мала всього лише один згортковий шар і кілька шарів субдіскретізація за принципом вибору максимального значення. Схема архітектури даної мережі представлена на рисунку 7.

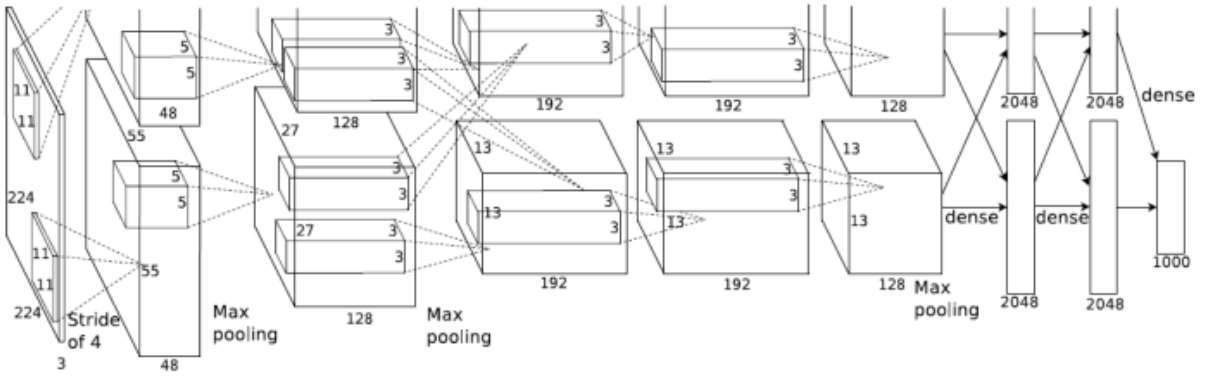


Рисунок 7 – Схема архітектури згорткової нейронної мережі AlexNet

В роботі, присвяченій розробці даної мережі також були запропоновані 96 фільтрів для згорткової нейронної мережі, які використовуються при проектуванні згорткових мереж і в даний час – рисунок 8.

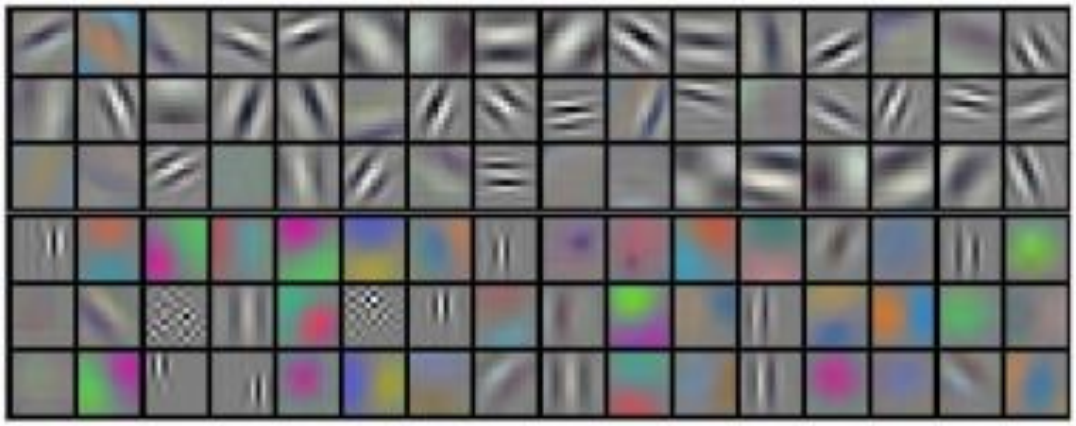


Рисунок 8 – Фільтри мережі AlexNet

Згорткова нейронна мережа AlexNet призначена для розпізнавання об'єктів будь-якої складності на великих зображеннях розміром 224 х 224. В 2012 року вона стала переможцем в конкурсі ImageNet, значно обігнавши своїх конкурентів.

Наступним значним внеском у розвиток побудови високоефективних архітектур згорткових нейронних мереж стала мережа ZFNet, створена Метью Цейлером (Matthew Zeiler) і Робом Фергюсом (Rob Fergus) у 2013 році. Дана мережа є модифікацією мережі AlexNet, основна особливість якої полягає в більш вигідному наборі параметрів мережі: збільшення розмірів внутрішніх згорткових шарів мережі, а також зменшення розмірів зміщення і розмірів

фільтрів у першому згортковому шарі. Архітектура мережі ZFNet представлена на рисунку 9.

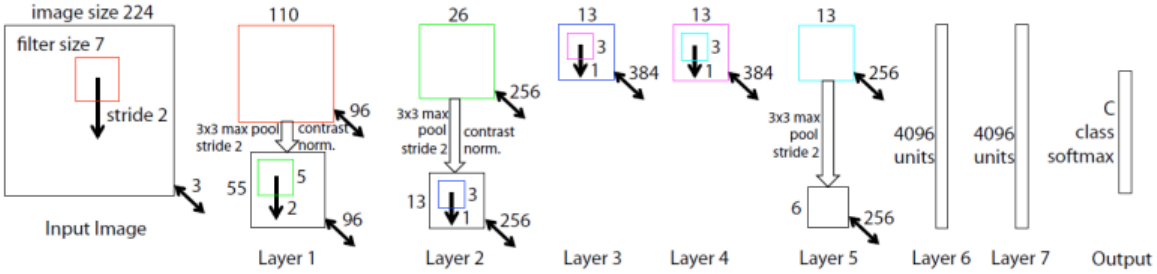


Рисунок 9 – Архітектура мережі ZFNet

Як видно на наведеному рисунку, дана мережа також як і AlexNet працює з зображеннями розміром 224 x 224. Також однією з заслуг роботи над ZFNet є створена авторами платформа для візуалізації роботи ЗНМ, за допомогою якої була проведена демонстрація виділення ознак, активації нейронів мережі тощо, що дуже вплинуло на популяризацію ЗНМ.

У 2014 році компанія Google почала в розробку своєї архітектури згорткових нейронних мереж і представила GoogLeNet. Дана згорткова мережа є дуже глибокою - до 22 шарів. Але, незважаючи на це, має в 10 разів менше параметрів, ніж мережа AlexNet, що позитивно позначається на продуктивності і використанні пам'яті. Також в ній використовуються малі розміри фільтрів, а шар субдіскретізація реалізований за принципом вибору середнього значення. Архітектура даної мережі представлена на рисунку 10.

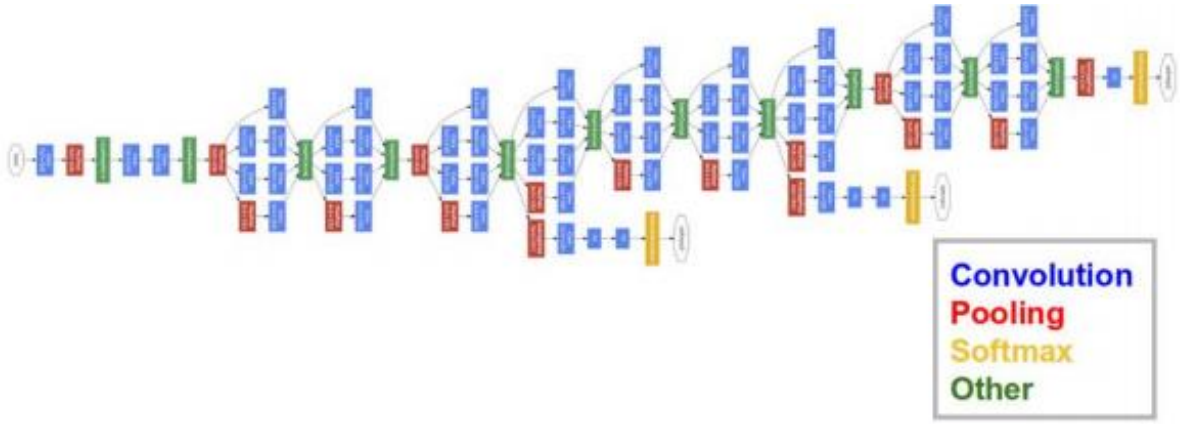


Рисунок 10 – Архітектура згорткової нейронної мережі GoogLeNet

Дана згорткова мережа була переможцем конкурсу Large Scale Visual Recognition Challenge 2014 (ILSVRC).

Наступний значний внесок в розвиток архітектур згорткових нейронних мереж внесла мережа VGGNet, створена Кареном Симоняном (Karen Simonyan) і Ендрю Зіссерманом (Andrew Zisserman) і, також як і GoogLeNet, брала участь в конкурсі ILSVRC 2014. Дана мережа, як і GoogLeNet, є дуже глибокою (до 16 шарів) і складається з великої кількості шарів згортки і субдіскретізації, що чергуються і, що мають малі розміри (3 х 3 - розмір фільтрів згорткового шару, 2 х 2 – розмір вікон шару субдіскретізації). Негативною стороною даної мережі є то, що вона зберігає до 140 мільйонів параметрів, що робить її громіздкою і низько продуктивною.

Однією з найбільш сучасних згорткових нейронних мереж на сьогоднішній день є мережа ResNet, що стала переможцем на конкурсі ILSVRC 2015. Архітектура даної мережі передбачає велику кількість згорткових шарів, що містять велику кількість (до 512) фільтрів малого розміру (3 х 3). Глибина мережі може досягати 152 шарів. На прикладі цієї мережі було встановлено, що ЗНМ може використовувати тільки згорткові шари і якість розпізнавання значно збільшується при збільшенні глибини мережі. Дана мережа є однією з найбільш ефективних згорткових нейронних мереж на сьогоднішній день. Архітектура даної мережі представлена на рисунку 11.

Проведений аналіз архітектур найбільш ефективних і високоточних ЗНМ, існуючих на сьогоднішній день, дозволив виділити особливості побудови ЗНМ, які забезпечують найкращу якість їх роботи.

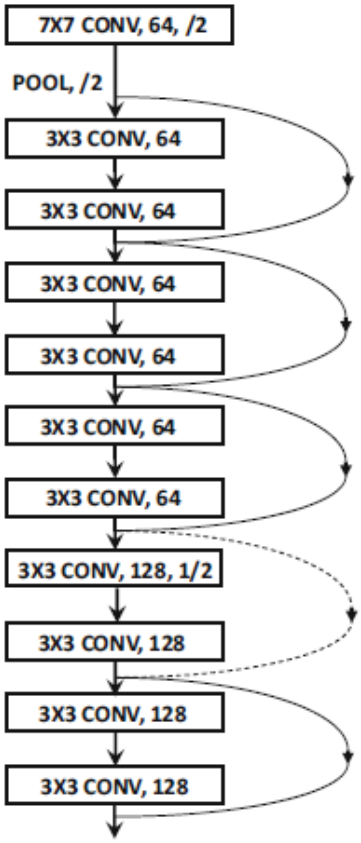


Рисунок 11 – Архітектура згорткової нейронної мережі ResNet

2.6 Висновки до розділу 2

В даному розділі приведено всі необхідні теоретичні дані для побудови згорткової нейронної мережі. Наведені основні принципи побудови та навчання нейронних мереж. Був розглянутий метод зворотного розповсюдження помилки та наведений алгоритм цього методу, який використовувався для розробки системи аналізу зображень. Також був запропонований алгоритм для побудови нейронної мережі та розглянуті критерії для оцінювання якості роботи системи.

РОЗДІЛ 3 АРХІТЕКТУРА ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ

3.1 Обґрунтування вибору платформи та мови реалізації програмного продукту

У цій роботі запропоновану модель розпізнавання плівок крові було реалізовано за допомогою мови програмування Pyhon. Python став лідером серед добре встановлених і оптимізованих мов, включаючи C, C ++ і Java, з дуже простих причин. Python був створений мати надзвичайно швидку і просту криву навчання та процес розвитку для інженерів програмного забезпечення Як результат, він вважається найбільш загальноприйнятою мовою програмування, оскільки користувачі можуть працювати практично в будь-якому досліджуваному домені і все ще мати змогу знайти корисний фрагмент коду для себе. Python використовував «силу» відкритого коду, що допомагало йому накопичити величезну базу користувачів практично з усіх верств суспільства. Будучи з відкритим вихідним кодом, Python дозволив людям робити невеликі програми та легко ділитися ними один з одним. У Python група програм для виконання різних завдань складається з модуля (пакета). Велика кількість модулів і розробників дозволили Python швидко перейти в комп'ютерну наукову спільноту і, нарешті, зайняти перше місце як найбільш сприятлива мова програмування.

Філософію Python можна визначити в одному реченні: Python - це багатофункціональна, портативна, об'єктно-орієнтована мова програмування високого рівня, яка дозволяє кодувати мінімалістичним способом.

Структура синтаксису була значною мірою отримана з оболонкових середовищ C і UNIX. Вони слугували натхненням для інтерпретаційного характеру робочого середовища. Інтерпретативна природа означає, що Python представляє розробнику інтерактивне середовище на основі REPL. Інтерактивна оболонка мови програмування Python широко відома як REPL (читання-оцінювання-друк-цикл), оскільки вона

• читає те, що вводить користувач,

• оцінює те, що він читає,

• виводить повернене значення після оцінки і

• повертається назад і робить це знову.

Особливо корисним виявляється таке інтерактивне робоче середовище для налагодження. Це також допомагає в прототипуванні проблеми, коли кожен крок може бути візуалізований для його виведення в живий спосіб. Користувачі можуть перевіряти результати певного коду, як тільки вони закінчать його написання. Спосіб роботи з REPL Python полягає в тому, щоб написати код, проаналізувати результати і продовжити цей процес, поки не буде обчислено кінцевий результат.

Також Python є об’єктно орієнтованою мовою. Більшість примітивних мов програмування носили процесуальний характер. Іншими словами, було визначено набір процедур для обчислення проблеми, і потік інформації контролювався в рамках цих процедур для отримання бажаного виходу. Отже, програма була просто розділена на блоки кодів, які взаємоділи один з одним, де один блок коду визначав підзадачу обчислення, що належить до розрахункової задачі. Навпаки, мова об'єктно-орієнтованого програмування (ООП) стосується даних як об'єкта, на якому діють різні методи для отримання бажаного результату. Все обчислюване розглядається як об'єкт. Його природа визначається як її властивості. Ці властивості можуть бути досліджені функціями, які називаються методами. Абстрактний характер об'єктів дає можливість вигадувати об'єкти вибору користувача і застосовувати концепції програмування для різноманітних додатків. Замість того, щоб забезпечити всі функціональні можливості своєї основної програми, творці Python спроектували його для розширення. Таким чином, користувачі можуть вибрати функціональність відповідно до їх вимог.

3.2 Алгоритм побудови нейронної мережі

Завдання побудови архітектури згорткової нейронної мережі для класифікації вхідного кольорового зображення зводиться до «згортання» вхідного шару мережі до шарів з найменшими розмірами. Запропонований підхід являє собою алгоритм, який регламентує вибір параметрів архітектури згоркової нейронної мережі виходячи з основних характеристик вхідних даних на кожному етапі послідовного формування шарів мережі.

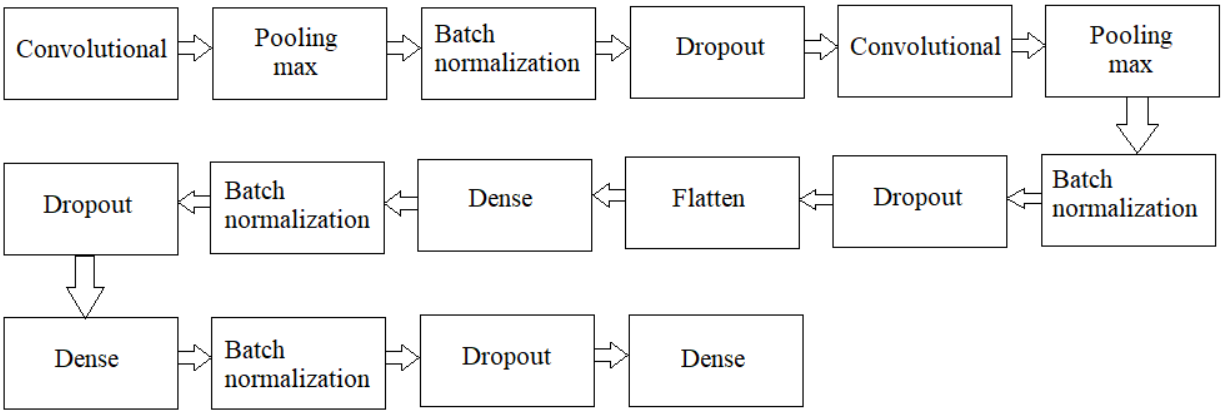


Рисунок 12 – Архітектура нейронної мережі

Модель має 2 шари згортки(convolutional). Цей шар застосовує операцію згортки до входу рецептивного поля і передає результат на наступний шар. Зазвичай складається з декількох фільтрів, які виявляють різні функції. Перший згортковий шар зазвичай виявляє прості особливості. Наступний згортковий шар виявить більш складні елементи і так далі. Також 2 шари пулінгу. Шар пулінгу збирає вихід кластера нейронів і об'єднує його в єдиний нейрон наступного шару. Об'єднання може використовувати різні функції агрегації (макс, ReLU тощо) для вибору значення, яке представляє кластер. Об'єднувальний шар зменшує розмір просторового представлення (отже, число параметрів), але зберігає трансляційну інваріантність. Це допомагає контролювати перенавчання. Далі йде BatchNormalization, щоб нормалізувати вихід з попередніх шарів і застосувати регуляризацію Dropout. Dropout(виключений) допомагає подолати перенасичення через випадіння випадкового набору активацій і встановлення їх до нуля. Мережа, підготовлена з відсічним шаром, повинна розвивати більш надмірність, тобто класифікувати, навіть якщо деякі нейрони деактивовані. Розрядний шар

використовується тільки в фазі навчання. Шар Flatten(згладжувальний) перетворює тривимірний вхід в один розмірний вектор. Вирівнювання шару видаляє просторову інформацію (не потрібну на цьому кроці) і передає вихід на наступний щільний шар, де він може бути оброблений для класифікації. Згладжені виходи потім передаються до штучної нейронної мережі, яка включає три щільні шари(Dense). Цей шар має властивості традиційної нейронної мережі персептрона, в якій кожен нейрон одного шару з'єднаний з кожним нейроном наступного шару. Щільні шари є останніми шарами, які виконують остаточну класифікацію.

3.3 Опис програмного продукту

Після запуску програмного продукту на екрані з’явиться опитування, що даний користувач хоче від програми.

Рисунок 13 – Головне меню програми

Натиснувши 1 користувач може перевірити зображення у даній директорії на малярію, як показано на рисунку 14.

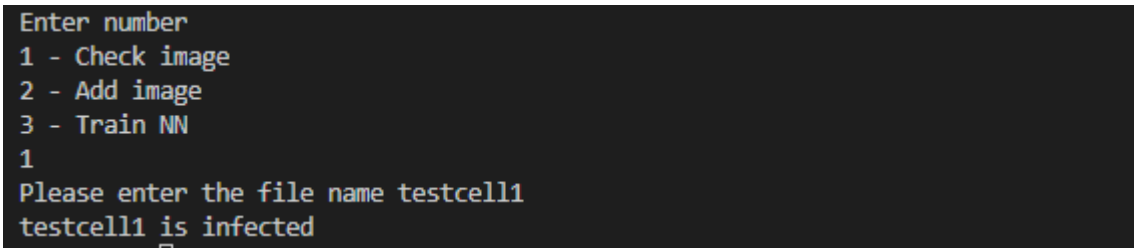


Рисунок 14 – Перевірка зображення

Якщо натиснути 2, то програма додасть фото до вибірки зображень, де потрібно буде вказати, на даному фото інфікована або неінфікована клітина як показано на рисунку 15.

```
Enter number
1 - Check image
2 - Add image
3 - Train NN
2
Please enter the filename for the file that should be added newcell1.png
Enter type of the image
1 - Infected
2 - Uninfected
1
newcell1.png successfully added to the infected images
```

Рисунок 15 – Додавання зображення

Натискаючи на 3, користувач зможе запустити навчання нейронної мережі з початку на нових даних.

3.4 Аналіз якості роботи системи

Після вибору даних, слід визначитися з метрикою, яка буде використовуватися для оцінки результатів. У загальному випадку метрика - це функція, яка приймає на вхід результати роботи алгоритму, а на виході повертає число, яке відповідає якості роботи алгоритму на конкретних даних. Для того, щоб протестувати якість розпізнавання малярійних клітин, потрібно розділи вихідні дані, наприклад, 80% - тренувальні та 20% тестові. Наступним кроком є перевірка коректності роботи нейронної мережі, тобто потрібно протестувати нейронну мережу на тестовій вибірці і підрахувати кількість правильно класифікованих зображень, у даному випадку – заражених клітин. Після обрахунку, виявилося що точність розпізнавання – 95,75%.

Для підвищення точності розпізнавання потрібно навчити модель на більш різноманітних даних. Наприклад, можна змінити масштаб зображення, повертати зображення на певний кут та перевертати їх. Після цього точність розпізнавання стала рівною 96,41%. Тобто при збільшенні тестової вибірки та варіацій розміщення зображень зростає точність.

У даній роботі є 2 типи зображень – інфіковані та неінфіковані зображення клітин.

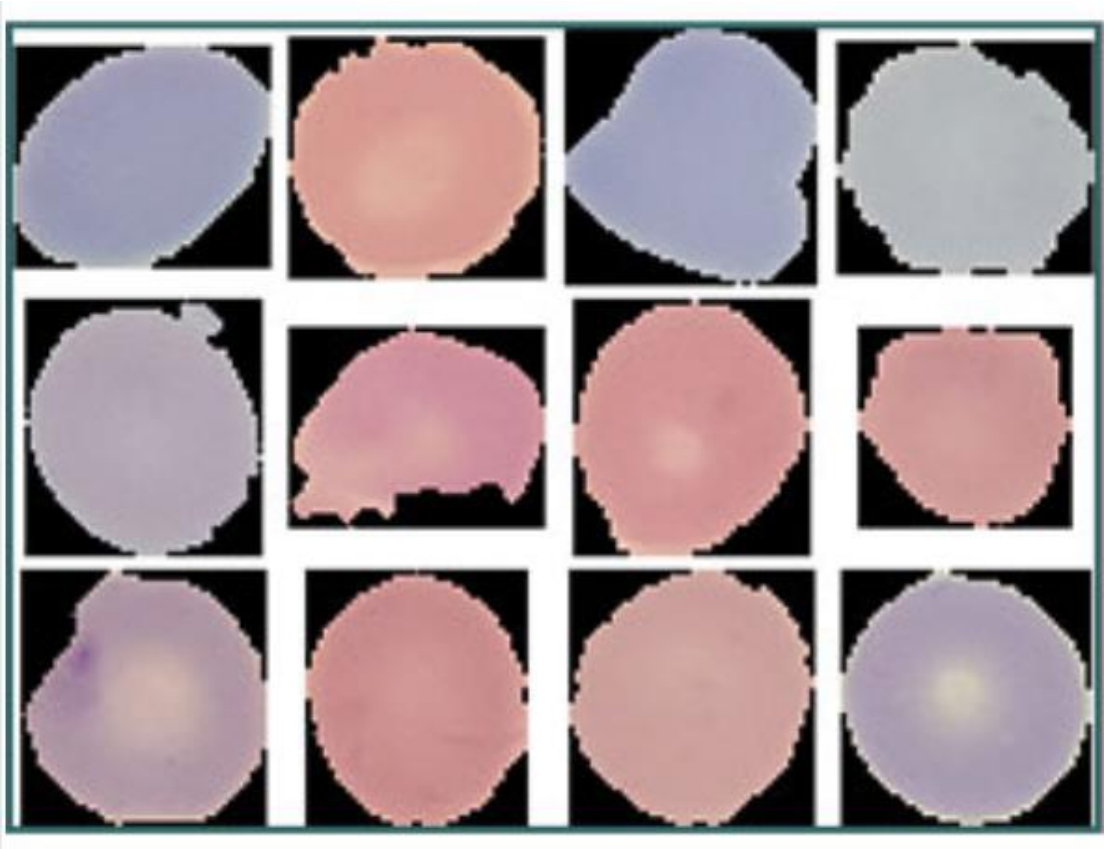


Рисунок 16 – Вибірка неінфікованих клітин

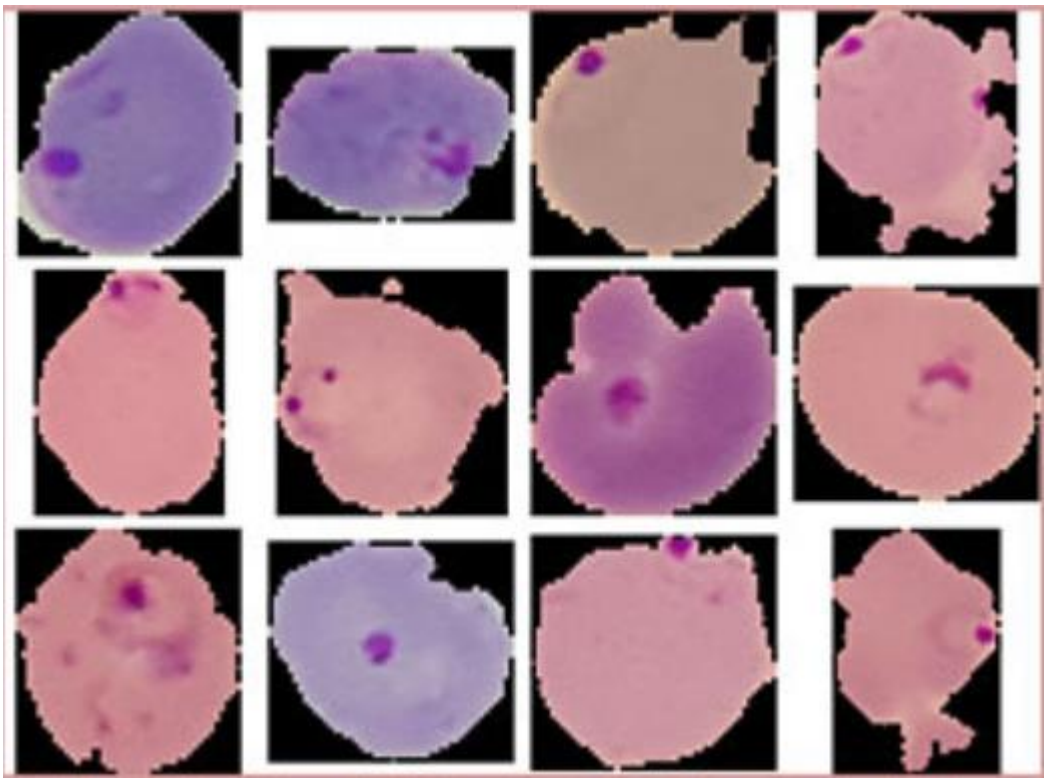


Рисунок 17 – Інфіковані клітини

Початкову вибірку зображень було розділено на 20% та 80%, для тесту роботи системи та для навчання відповідно.

Перевірка точності виробленої моделі з аугментацією даних, графік процесу навчання зображено на рисунку 18.

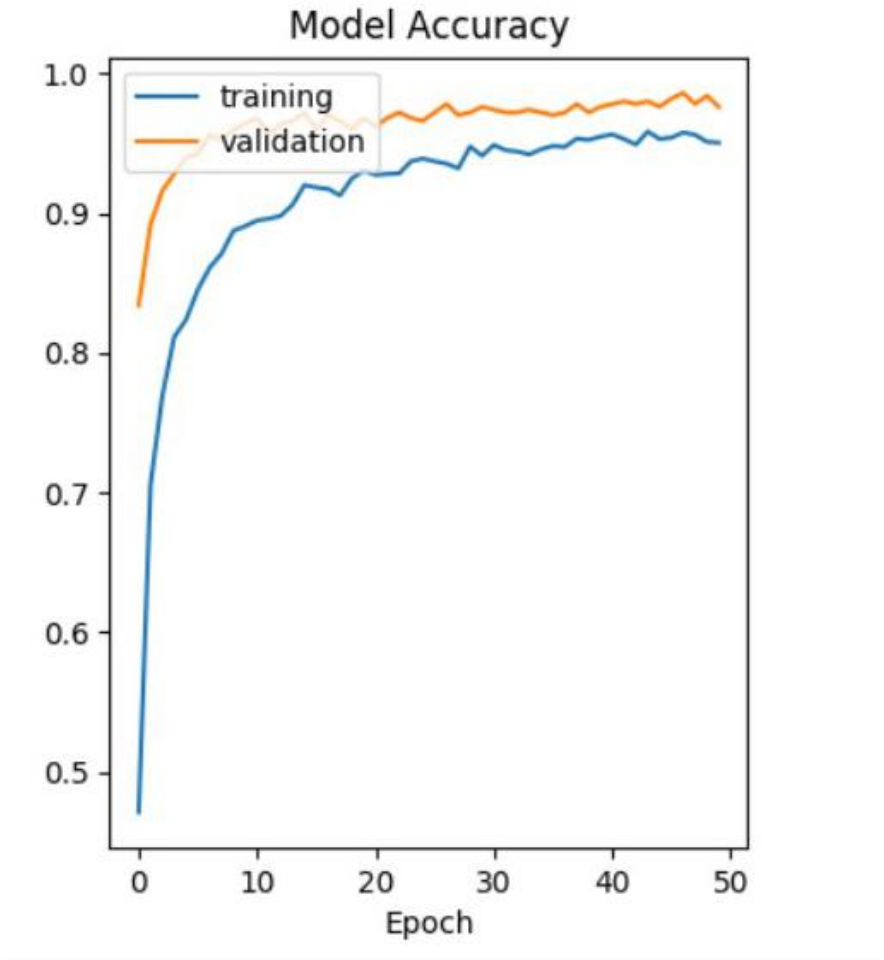


Рисунок 18 – Графік процесу навчання виробленої моделі

3.4.1. Порівняльний аналіз

Для порівняння будемо використовувати попередньо підготовлену модель глибокого навчання VGG-19, розроблену групою візуальної геометрії (VGG) в Оксфордському університеті. Модель VGG-19 являє собою 19-шарову (згорткові та повнозв’язні шари) мережу глибокого навчання, побудовану на базі даних ImageNet, яка побудована з метою розпізнавання і класифікації зображень. Вона складається з 16 шарів згортки та шарами макс-пулінга й двома повнозв’яними шарами. Її модель зображена на рисунку 19.

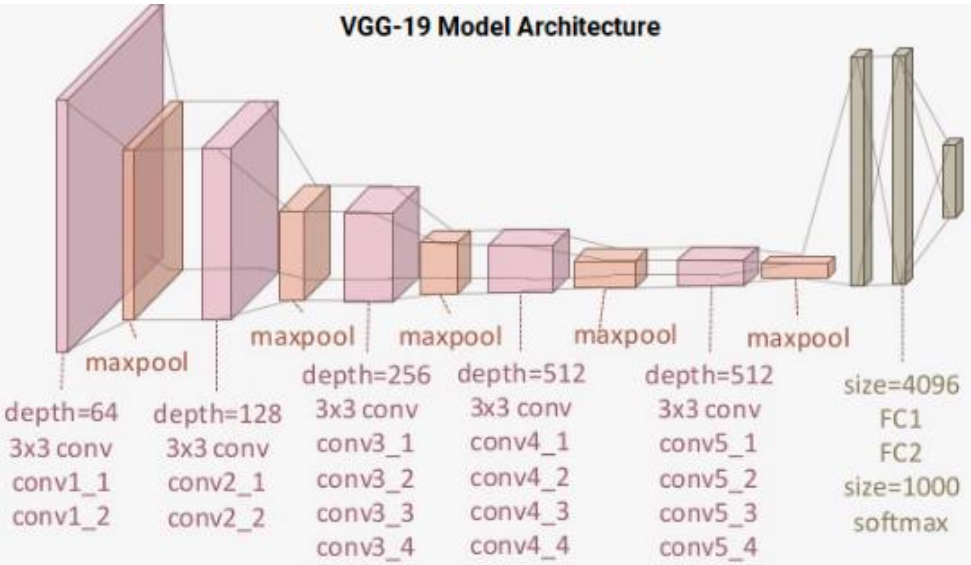


Рисунок 19 – Модель VGG

Після перевірки точності розпізнавання тих самих даних, але вже на моделі VGG отримали наступні результати зображені на рисунку 20.

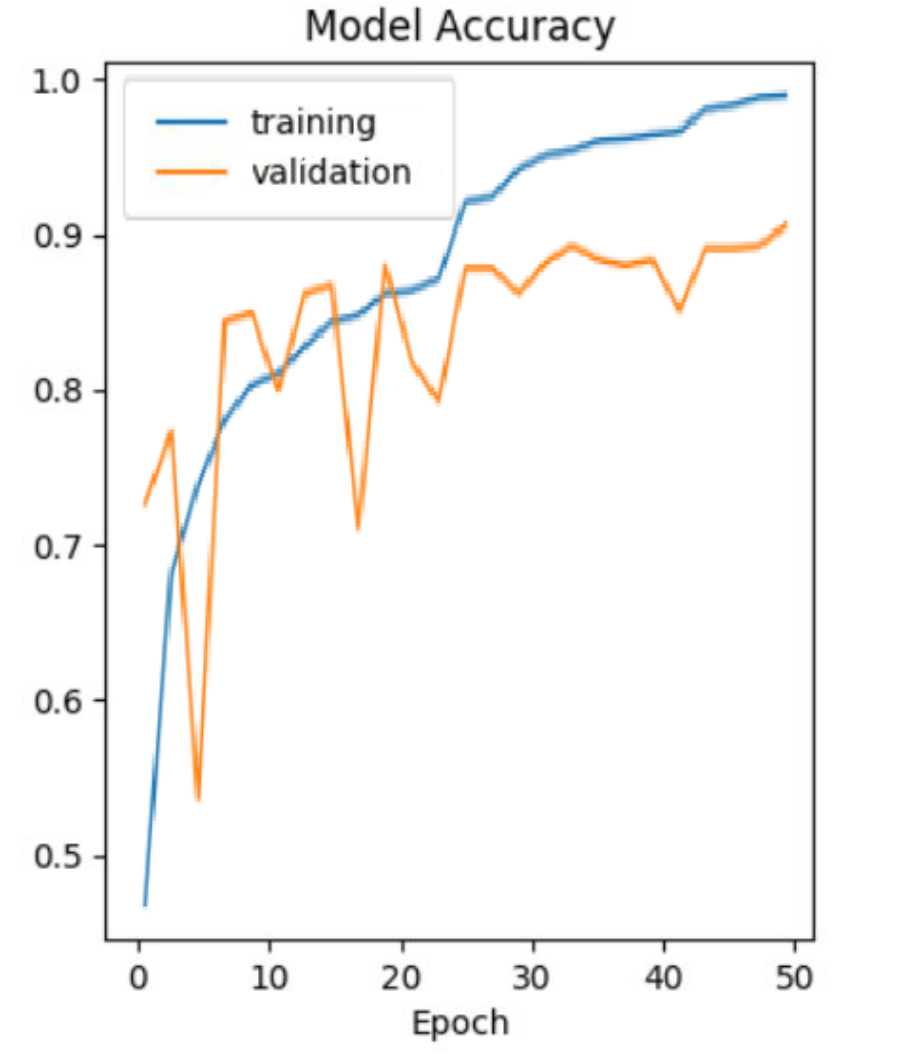


Рисунок 20 – Графік процесу навчання моделі

Цей графік показує що точність не дуже велика. Отже, точність власно зробленої моделі значно краща, досягається 96.41% точності розпізнавання, коли на іншій моделі лише приблизно 85%.

3.5 Висновки до розділу 3

У розділі показані основні переваги використаної мови програмування Python, було побудовано нейронну мережу для розпізнавання зображень інфікованих та неінфікованих клітин малярії та розроблено програмній продукт.

В роботі наведено результати розрахунку точності розпізнавань вихідної вибірки для 2 моделей – створеної власноруч та попередньо зробленої моделі-аналогу. Найкращий результат якого вдалося досягти – 96.14%, це точність розпізнавання для розробленої нейронної мережі.

РОЗДІЛ 4 ЕКОНОМІЧНА ЧАСТИНА. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ

4.1 Вступ

Даний розділ присвячений функціонально-вартісному аналізу програмного продукту — системи ідентифікації малярійних клітин.

Функціонально-вартісний аналіз — це метод комплексного техніко-економічного дослідження об’єкта з метою розвитку його корисних функцій при оптимальному співвідношенні між їхньою значимістю для споживача і витратами на їхнє здійснення. Є одним з основних методів оцінки вартості науково-дослідної роботи, оскільки ФВА враховує як технічну оцінку продукту, що розробляється, так і економічну частину розробки.

Крім того, даний метод дозволяє вибрати оптимальний, як з погляду розробника, так і з точки зору покупця варіант розв’язання будь-якої задачі, а також дозволяє оптимізувати витрати й час виконання робіт. Зниження витрат виробництва треба починати з аналізу властивостей виробу, що використовуються, а також технічних функцій його складових частин.

У даному розділі проводиться економічний, техніко-економічний аналіз програмного продукту, основним завданням якого є аналіз даних, а також виключення зайвих функцій, що дозволяє пришвидшити процес прийняття рішень, забезпечивши при цьому якісні та достовірні результати.

Фактично цей метод працює за таким алгоритмом:

а) визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

б) для кожної функції визначаються повні річні витрати й кількість робочих часів.

в) для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

г) після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.2 Постановка завдання техніко-економічного дослідження

Розроблюваний програмний продукт (ПП) є системою підтримки прийняття рішень для ідентифікації малярійних клітин. Система дозволяє організувати класифікацію зображень. Щоб швидко діагностувати малярію, необхідно мати зручний і надійний інструмент, який зможе автоматизувати роботу у медичній сфері. Задачею проекту було створити програму з оптимальним набором необхідних функцій і простим інтерфейсом, що не потребував би великої кількості часу на освоєння.

Технічні вимоги до СППР:

- ПП повинен функціонувати на комп'ютерах, що мають стандартний набір компонент;
- зручність і простота діалогової взаємодії з користувачем;
- мінімальні затрати на впровадження ПП (навчання користувача);
- захист інформації від помилок, що можуть виникнути в процесі експлуатації системи.

Для розробки ПП використовувалась мова програмування Python 3.7 і середовище Jupyter Notebook.

4.3 Аналіз виявлених варіантів програмного продукту

4.3.1 Виділення основних функцій

Головна функція F_0 – розробка програмного продукту, який дає найменшу похибку при порівняння реальних значень цільової змінної та

результуючого значення побудованого багаточлену, у відповідній точці. Виходячи з конкретних цілей, реалізованих програмним засобом, виділимо її основні функції:

F1 – генерація даних;

F2 –моделювання;

F3 –прогнозування на обрану кількість кроків;

F4 – збереження даних;

F5 – організація дружнього інтерфейсу.

4.3.2 Розробка варіантів реалізації ПП

Деякі з основних функцій можуть мати декілька варіантів рішень, інші – один:

для F1:

а) завантаження даних з носія інформації і їх сортування.

б) введення назв з клавіатури.

для F2:

а) моделювання і побудова моделі.

для F3:

а) прогнозування кожного наступного значення незалежно від попереднього.

для F4:

а) в оперативній пам’яті;

б) на диску.

для F5:

а) організація тільки користувальницького меню;

б) організація системи оперативної допомоги;

в) організація навчальної системи.

По розглянутих варіантах будуємо морфологічну карту зображено на рисунку 21.

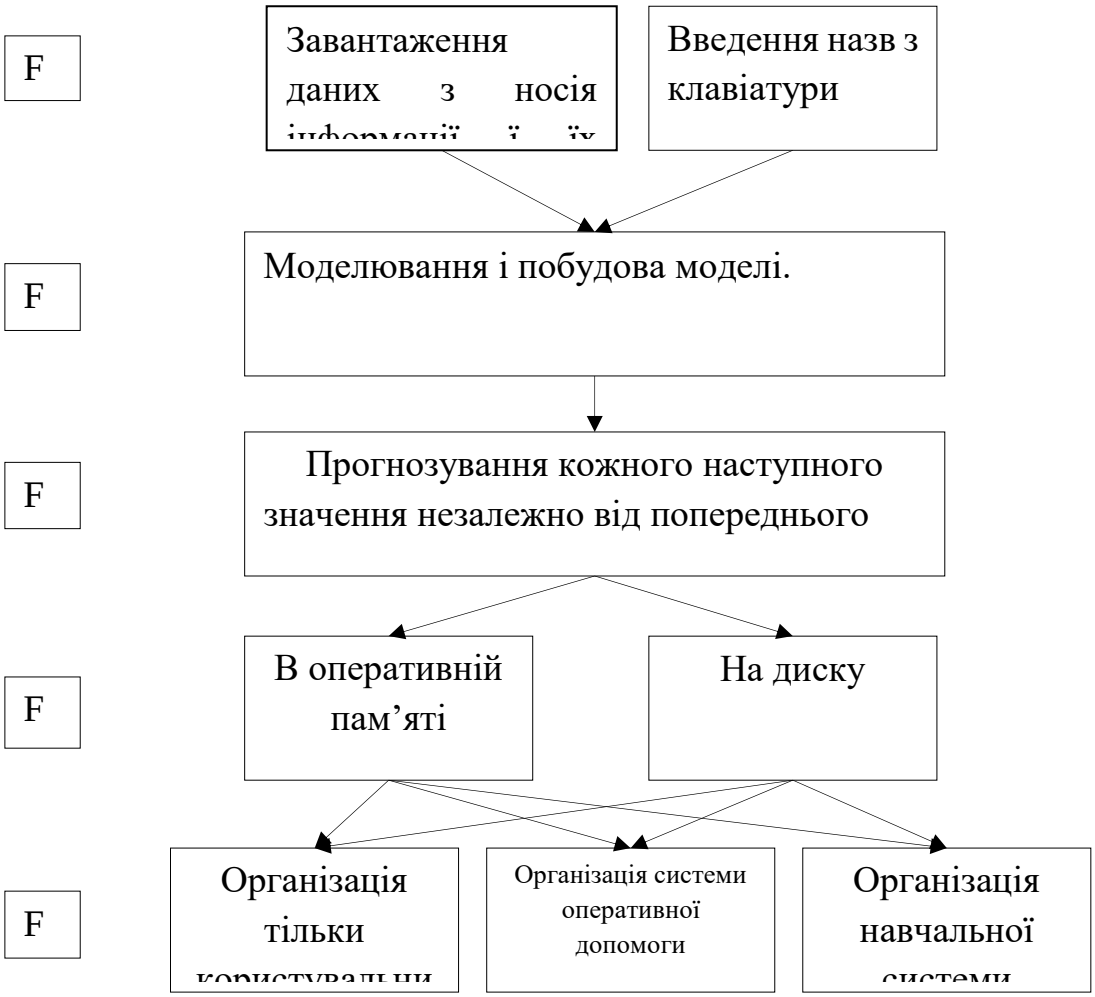


Рисунок 21 –Морфологічна карта

Побудуємо позитивно-негативну матрицю варіантів реалізації основних функцій зображену на таблиці 10.

Таблиця 10 – Позитивно-негативна матриця варіантів реалізації функцій

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	а	Можливість вибору потрібної кількості інформації і простота програмної реалізації	Вірогідність недостатньої кількості інформації
	б	Можливість введення нових даних	Великі затрати часу і незручність

Продовження таблиці 10

F2	а	Вибір потрібної інформації для обробки та побудова моделі залежності даних	Вірогідність втрати інформації за рахунок статистичної залежності даних
F3	а	Прогнозування значень без нарощування помилки з кожним кроком	Вірогідність некоректного значення враховуючи особливості моделі
F4	а	Збільшення швидкості обробки даних	Можлива втрата інформації при збоях
	б	Підвищення надійності за рахунок зменшення ймовірності втрати даних при збоях	Зменшення швидкості обробки даних
F5	а	Мінімальний обсяг програми	Ускладнення роботи користувача із програмою
	б	Простота й зручність роботи із програмою	Збільшення обсягу програми
	в	Простота й швидкість навчання роботи із програмою	Збільшення обсягу програми

Проаналізувавши матрицю, ми можемо виключити такі варіанти реалізації основних функцій:

F1: б) F5: а), в).

Варіанти F1: оскільки в пріоритеті є час виконання програми, то варіант б) відкидається.

Варіанти F5: оскільки користувач програми не є фахівцем з обчислювальної техніки, то для спрощення й зручності роботи із програмою необхідна організація системи оперативної допомоги. Тому варіант а) відкидається. Варіант в) був відкинутий внаслідок збільшення потреби в оперативній пам'яті, збільшення обсягу програми й ускладнення розроблювального продукту для реалізації цього варіанта. По варіантах, що залишилися, будуємо спрощену морфологічну карту зображену на рисунку 22.

Рисунок 22 – Спрощена морфологічна карта

Варіанти, що залишилися:

1) F1а + F2а + F3а + F4а + F5б
2) F1а + F2а + F3а + F4б + F5б

4.4 Аналіз системи параметрів. Обґрунтування параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для характеристики розроблювального ПП можна застосувати наступні параметри:

X1 – обсяг пам'яті, зайнятий даними;

X2 –час обробки даних;

X3 –точність розв’язку;

X4 – потреба в обсязі оперативної пам'яті;

X5 – потенційний обсяг ПП.

Значення параметра X1 визначається обсягом пам'яті, зайнятим даними будь-якому пристрої зберігання цифрової інформації.

Значення параметра X2 визначається часом, який потрібний для побудови адекватної моделі досліджуваним даним.

Параметр X3 указує на точність спрогнозованого значення, використовуючи при цьому значення нев’язки.

Параметр X4 визначається обсягом оперативної пам'яті, необхідним для збереження або подальшого використання даних.

Значення параметра X5 визначається обсягом пам'яті, зайнятим програмою на жорсткому диску або іншому носії.

З даних технічної літератури й досвіду попередніх розробок визначаємо гранично припустимі, середні одержувані й досяжні значення параметрів. Результати зводимо в таблиці 11.

Таблиця 11 – Основні параметри ПП

Найменування параметра	Од. вим.	Позначення параметра	Гранично припустиме значення	Середнє одержуване значення	Досяжне значення
Обсяг пам'яті, зайнятий даними	Мб	X1	5	2	1
Час обробки даних	с	X2	9	4	2
Точність розв’язку	Доля одиниці	X3	0,5	0,1	0,01
Потреба в оперативній пам'яті	Мб	X4	32	16	8
Потенційний обсяг програми	Кб	X5	2000	1400	800

Гранично припустимому значенню відповідає бальна оцінка - 0, середньому одержуваному значенню - 5, досяжному значенню параметра - 10. За даними таблиці 11 будуємо графічні характеристики параметрів зображені на рисунках 23- 27.

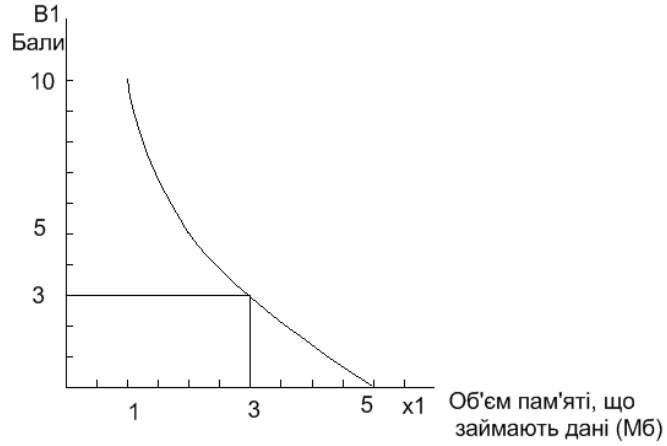


Рисунок 23 – Бальна оцінка обсягу пам'яті, який займають дані

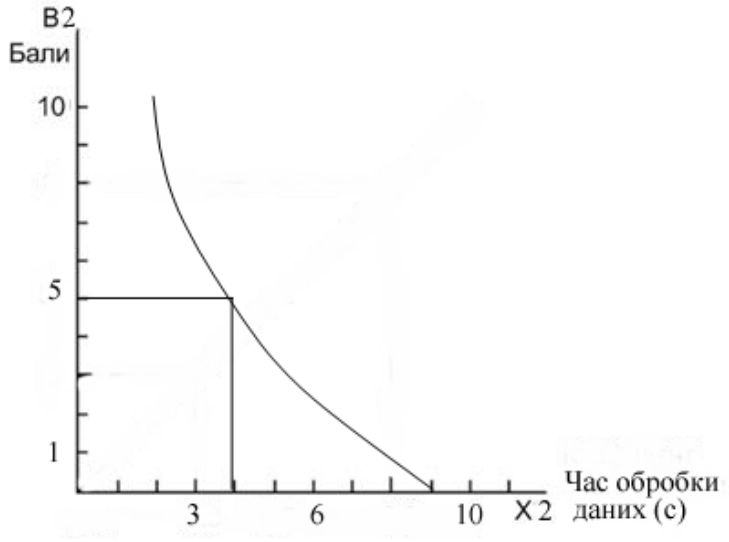


Рисунок 24 – Бальна оцінка часу обробки даних

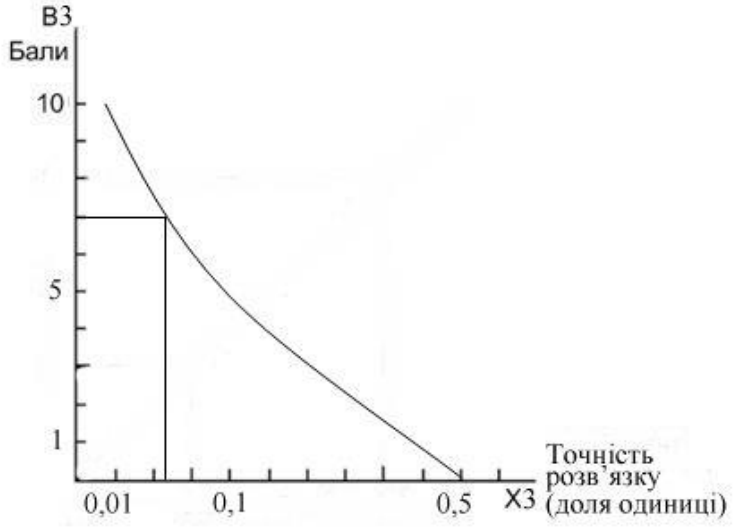


Рисунок 25 – Бальна оцінка точності розв’язку

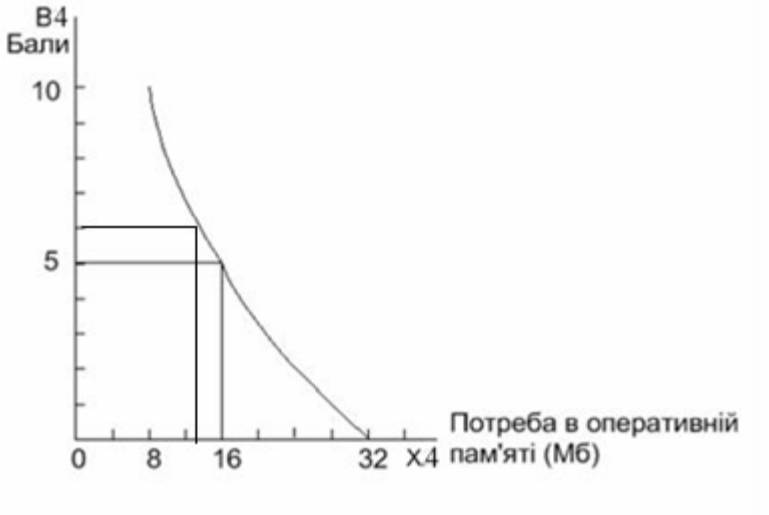


Рисунок 26 – Бальна оцінка потреби в оперативній пам'яті

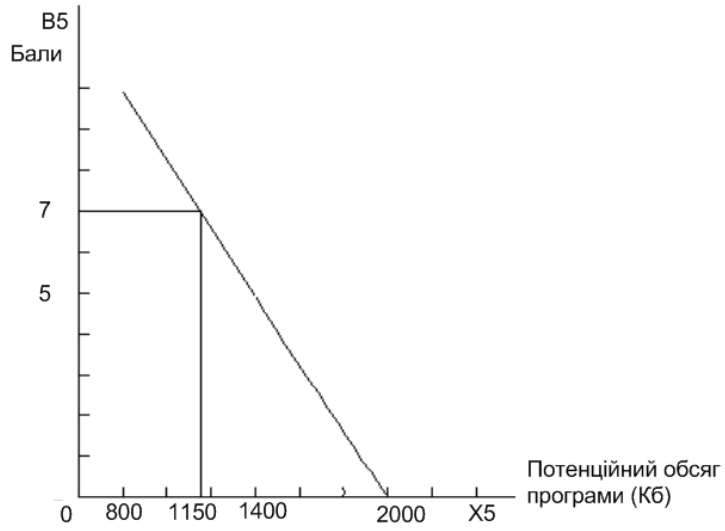


Рисунок 27 – Бальна оцінка потенційного обсягу програми

Вагомість параметрів визначаємо методом розміщення пріоритетів, відповідно до якого пріоритет параметрів визначає експертна комісія в складі п'яти спеціально запрошених фахівців.

Результати порівняння параметрів наведені в таблиці 12.

Таблиця 12 – Експертне порівняння параметрів

Параметри	Експерти							Підсумкова оцінка	Числове значення
	1	2	3	4	5	6	7		
X1,X2	>	>	>	<	>	<	>	>	1.5
X1,X3	>	>	>	>	>	<	>	>	1.5
X1,X4	<	<	<	<	<	<	<	<	0.5
X1,X5	<	<	<	<	<	<	>	<	0.5
X2,X3	<	<	>	<	<	<	<	<	0.5
X2,X4	>	>	<	>	>	<	>	>	1.5
X2,X5	>	>	<	>	<	>	>	>	1.5
X3,X4	>	>	>	<	>	<	>	>	1.5
X3,X5	>	>	>	>	>	>	>	>	1.5
X4,X5	<	<	<	<	>	<	<	<	0.5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij}визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$. Розрахуємо коефіцієнт конкордації (погодженості) експертних оцінок. Для цього скористаємося методом розміщення рангів. Суть методу полягає в тому, що кожному одиничному показникові якості продукту ставиться у відповідність ранг: найбільш важливому (на думку даного експерта) привласнюється ранг - 1, менш важливому, - 2, далі - 3 і т.д. Якщо, на думку експертів, два показники мають однакову важливість (рівноцінні), їм привласнюється середній ранг, що зображено на таблиці 13.

Таблиця 13 – Результати ранжирування показників і перевірка ступеня застосовності експертних оцінок

Параметр	Найменування параметра	Од. вим.	Ранг показника по оцінці експерта							Сума рангів R_i	Відхилення D_i	D^2_i
			1	2	3	4	5	6	7			
X_1	Обсяг пам'яті, зайнятий даними	Мб	5	3	4	4	5	4	5	30	6	36
X_2	Час обробки даних	с	2	2	1	3	2	3	2	15	-5	25
X_3	Точність розв’язку	Доля одиниці	1	1	2	1	1	1	1	8	-9	81

Продовження таблиці 13

X_4	Потреба в оперативній пам'яті	Мб	4	4	5	2	3	3	4	25	3	9
X_5	Потенційний обсяг програми	Кб	3	5	3	5	4	4	3	27	5	25
			15	15	15	15	15	15	15	105	0	176

Сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij},$$

де N – кількість експертів.

$$R_j = \sum_{i=1}^{n} r_{ij},$$

де n – кількість параметрів;

$$R_{ij} = \frac{N \cdot n \cdot (n+1)}{2} = 105$$

Середня сума рангів:

$$T = \frac{R_{ij}}{n} = 15;$$

відхилення $\Delta_i = R_i - T$, причому $\sum_{i=1}^{n} \Delta_i = 0$;

квадрат відхилень Δ_i^2 і сума $S = \sum_{i=1}^{n} \Delta_i^2 = 176$;

Коефіцієнт погодженості (конкордації) - W знайдемо по формулі:

$$W = \frac{12S}{N^2\left(n^3 - n\right)}, W = \frac{12*176}{\left(7^2 * \left(5^3 - 5\right)\right)} = 0{,}69 > W_k = 0{,}67$$

Оскільки отримане значення W більше нормативного ($W_н$), то це дає нам підставу скористатися результатами експертного опитування для подальших розрахунків.

Розрахунок вагомості кожного параметра робимо по формулах:

$$\varphi_i = \frac{b_i}{\sum_{i=1}^{n} b_i}$$

$$b_i = \sum_{j=1}^{n} a_{ij}$$

де φ_i - відносна оцінка i-го параметра;

b_i - вага i-го параметра за результатами експертних оцінок;

n - число параметрів;

a_{ij} - числове значення оцінки, що визначає ступінь переваги i-го параметра над j-тим.

Відносні оцінки для другого й наступного кроків:

$$\varphi_i^{/} = \frac{b_i^{/}}{\sum_{i=1}^{n} b_i^{/}}$$

$$b_i^{/} = \sum_{j=1}^{n} a_{ij} * b_j$$

Отримані дані зводимо в таблиці 14.

Таблиця 14 – Розрахунок пріоритету параметрів

Параметри X_i	Параметри X_j					Перший крок		Другий крок	
	X1	X2	X3	X4	X5	b_i	φ_i	$b_i^{/}$	$\varphi_i^{/}$
X1	1,0	1,5	1,5	0,5	0,5	5	0,2	26	0,211
X2	0,5	1,0	0,5	1,5	1,5	5	0,2	24	0,195
X3	0,5	1,5	1,0	1,5	1,5	6	0,24	29,5	0,239
X4	1,5	0,5	0,5	1,0	0,5	4	0,16	19,5	0,158
X5	1,5	0,5	0,5	1,5	1,0	5	0,2	24	0,195
Сума						25	1	123	1

Оскільки відносна оцінка i-го параметра, отримана на другому кроці, відрізняється від оцінки, отриманої на першому кроці, менш чим на 5 %, то вона приймається за коефіцієнт вагомості i-го параметра.

4.5 Оцінка рівня якості варіантів реалізації програмного продукту

Визначимо узагальнений показник рівня якості обраних варіантів реалізації основних функцій

$$K_{TP} = \sum_{i=1}^{n} \phi_i \cdot B_i$$

і варіантів ПП

$К_{ТРk} = К_{ТР}[F1] + К_{ТР}[F2] + К_{ТР}[F3] + К_{ТР}[F4] + К_{ТР}[F5]$,

де φ_i - коефіцієнт вагомості i-го параметра (з табл. 4.5);

B_i - бальна оцінка i-го параметра, обумовлена із графіків, зображених на рисунках 22-26;

n - число параметрів (n = 5).

Визначаємо показники рівня якості кожного варіанта реалізації основних функцій окремо. Результати розрахунку зводимо в таблиці 15.

Таблиця 15 – Розрахунок показників рівня якості варіантів реалізації основних функцій

Основна функція	Варіант реалізації	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	б	3	3	0,211	0,633
F2(X2)	б	4	5	0,195	0,975
F3(X3)	б	0,05	7	0,239	1,673
F4(X4)	б	16	5	0,158	0,79
	а	13	6	0,158	0,948
F5(X5)	а	1150	7	0,195	1,365

Абсолютні значення параметрів X1, X4, X5 відповідають технічним вимогам умов функціонування даного ПП. Абсолютне значення параметра X2 (час обробки) обрано з наступних міркувань: це є середнє отримуване значення. Абсолютне значення параметра X3 (точність розв’язку) обрано з наступних міркувань: середнє значення між досяжним значенням і середнім отримуваним, такий варіант є часто отримуваним на статистично незалежних даних.

За даними таблиці 4.6 визначимо показники рівня якості кожного з варіантів ПП.

$К_{ТР}$ 1 = 0,633 + 0,975+ 1,673+ 0,948 + 1,365 = 5,594;

$К_{ТР}$ 2 = 0,633 + 0,975+ 1,673+ 0,79+ 1,365 = 5,436

Отже, найбільш якісним є перший варіант, тому що він має максимальне значення коефіцієнту технічного рівня.

4.6 Економічний аналіз варіантів програмного продукту

4.6.1 Визначення трудомісткості

При визначенні трудомісткості ПП враховують наступні фактори:

- об’єм ПП в умовних машинних командах (УМК);
- складність розроблюваного ПП;
- ступінь новизни розроблюваного ПП;
- ступінь використання у розробці стандартних модулів, типових програм.

Даний програмний продукт призначений для вже існуючих комп’ютерів й операційних систем, використання типових програмних продуктів не планується.

Обидвава варіанти включають в себе 5 окремих завдань:

- створення інтерфейсу користувачу;
- обробка даних;
- сортування;
- довідка і навчання;
- математична статистика і прогнозування.

Варіант II містить ще одне завдання:

- написання алгоритму збереження інформації.

Завдання 1-5 за ступенем новизни належать до групи Б; завдання 6 – до групи В.

За складністю алгоритми, використовувані в завдані 5 належать до групи 1; завдання 2, 3 – до групи 2; завдання 1, 4, 6 – до групи 3.

Усі шість завдань використовують контроль вхідної інформації, яка характеризується групою 5,2, та контроль вихідної інформації, яка характеризується групою 5,4.

Для реалізації завдань 2, 4 і 6 використовується змінна інформація, для інших – нормативно-довідкова.

Розрахуємо норму часу на розробку та програмування для кожного шести завдань.

Загальна трудомісткість обчислюється за формулою:

$$To = Tp \bullet Kп \bullet Kск \bullet Km \bullet Kст \bullet Kст.м.$$

Для першого завдання, виходячи із норм часу для завдань ступеня новизни Б та групи складності алгоритму 3, трудомісткість Тр= 19 людино-днів. Поправковий коефіцієнт, який враховує вид використаної інформації для першого завдання (нормативно-довідкова інформація): Кпк = 0,9. Поправковий коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх шести завдань, Кск = 1. У зв'язку з тим, що використання типових програмних продуктів не планується, коефіцієнт Кст приймаємо рівним 1. Коефіцієнти Км і Кст.п, які враховують відповідно програмування на мові високого рівня та розробку стандартного програмного забезпечення, для всіх завдань становлять: Км = Кст.п = 1. Отже:

T_o = 19*0,9*1*1*1*1= 17,1 людино-днів.

Проведемо аналогічні розрахунки для інших завдань.

Для другого завдання (використовується алгоритм другої групи складності, ступінь новизни Б):

T_o = 27*1,8*1*1*1*1= 32,3 людино-днів.

Для третього завдання (використовується алгоритм другої групи складності, ступінь новизни Б):

T_o = 27*1,08*1*1*1*1= 29,16 людино-днів.

Для четвертого завдання (використовується алгоритм третьої групи складності, ступінь новизни Б):

T_o = 19*1,5*1*1*1*1= 28,5 людино-днів.

Для п’ятого завдання (використовується алгоритм першої групи складності, ступінь новизни Б):

T_o = 64*1,021*1*1*1*1= 65,344 людино-днів.

Для шостого завдання (використовується алгоритм третьої групи складності, ступінь новизни В):

T_o = 12*1*1*1*1*1= 12 людино-днів.

Складемо трудомісткість відповідних завдань для кожного з двох обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

T_I = (17,1 + 32,3 + 29,16 + 28,5 + 65,344) • 8 = 1379,232людино-годин;

T_{II} = (17,1 + 32,3 + 29,16 + 28,5 + 65,344 + 12) • 8 = 1475,232людино-годин;

Очевидно, що найвищу трудомісткість має варіант II.

У розробці бере участь один програміст з окладом *5000* грн. Визначимо зарплату програміста за годину:

$$C_ч = \frac{5000}{21 \cdot 8} = 29{,}76 \text{ грн.}$$

Тоді зарплата розробників за варіантами :

I. $C_з$ = 29,76 * 1379,232• 1,2 = 49255,2грн.

II. $C_з$ = 29,76 * 1475,232• 1,2 =52 683,5грн.

Відрахування на всі види соціального страхування за варіантами:

I. $C_{от}$ = 49255,2 *0,22= 10 836,14грн.

II. $C_{от}$ = 52 683,5*0,22= 11 590,37грн.

4.6.2 Визначення витрат на розробку ПП

У вартісному аналізі розраховуються витрати, необхідні для розробки й створення програмного продукту, тобто визначаються функціонально-

необхідна вартість виробу для усіх варіантів реалізації функцій, що досліджуються.

Функціонально-необхідні витрати на створення програмного продукту визначаються так:

$$C_{пп} = C_з + C_{от} + C_м + C_н$$

де $C_з$ – оплата праці розробників;

$C_{от}$ – єдиний соц. внесок в залежності від класу проф. ризику – становить 36,77% від фонду оплати праці;

$C_м$ – вартість машинного часу, необхідного для розробки й налагодження програмного продукту;

$C_н$ – накладні витрати в розмірі 50 – 150% від витрат на оплату праці, до складу яких входять витрати на оплату праці управлінського персоналу з нарахуваннями, оплату службових відряджень, консультаційно-інформаційні витрати, ремонт та техобслуговування інших основних фондів, крім ПК, оренду приміщення та ін.

Вартість машинного часу визначається:

$$C_м = C_{м-г} \cdot t_M$$

де t_M – тривалість машинного часу (сума часу машинних і машинно-ручних операцій), необхідного для розробки ПП;

$C_{м-г}$ – собівартість 1 машино-години роботи ПК:

$$C_{м-г} = \frac{C_{екс}}{T_{еф}}$$

де $C_{екс}$ – річні експлуатаційні поточні витрати на обслуговування ПК, що охоплюють:

- основну і додаткову плату спеціаліста, що обслуговує машину з урахуванням його зайнятості на обслуговуванні ПК;
- відрахування на соціальні заходи;
- амортизаційні відрахування (50% від залишкової вартості);
- витрати на електроенергію, які розраховуються як добуток тарифу на обсяг потужності, що споживається ПК і на ефективний годинний фонд часу ПК за рік;
- накладні витрати – 50-150% від витрат на оплату праці;
- $T_{еф}$ – ефективний годинний фонд часу роботи ПК за рік; визначається виходячи з календарного річного фонду часу, зменшеного з урахуванням вихідних, святкових днів і добового режиму роботи ПК, годин.

Вартісний аналіз варіантів реалізації функції завершується визначенням коефіцієнта техніко-економічного рівня кожного варіанта ($K_{ТЕПj}$), який розраховується за формулою:

$$K_{ТЕПj} = \frac{K_{TPj}}{C_{фj}},$$

де K_{TPj} – коефіцієнт технічного рівня j-го варіанту;

$C_{фj}$ – величина функціонально-необхідних витрат j-го варіанту.

Для розробки програмного продукту був задіяний один ПК, який працював 5 днів на тиждень по 8 годин.

Його вартість – 4500 грн.

Потужність – 120 Вт.

Тоді:

$T_{еф}$ =(365-52·2) ·8·0,9=1879,2.

З урахуванням часу на ремонт (15%):

$T_{еф}$ =1879,2·(1-0,15)=1597,32.

Оскільки ЕОМ обслуговує один спеціаліст з окладом 5000 грн з коефіцієнтом зайнятості 0,2, то для одного комп'ютера отримаємо:

$C_о$ = 12 * 5000 * 0,2 = 12000 грн.

З розрахунком додаткової заробітної плати:

$C_о$ = 12000 * (1 + 0,2) = 14400 грн.

Відрахування на всі види соціального страхування:

$C_о$ = 14400 * 0,22 = 3 168грн.

Амортизаційні відрахування розраховують за формулою (якщо амортизація становить 50 % за рік і вартість ЕОМ –4500 грн)

$C_а$= 1,15 * 0,5 * 4500 = 2587,5 грн.

Витрати на ремонт і профілактику, розраховують за формулою:

$C_Р$ = 1,15 * 5000 * 0,05 = 287,5грн.

Ефективний годинний фонд часу ПК за рік знаходять за формулою

$T_{ЕФ}$ = (365 – 104) * 8 * 0,9*(1–0,15) = 1597,32 год.

Витрати на оплату електроенергії обчислюють за формулою:

$C_{ЕЛ}$ = 1597,32 • 2.7515 • 0,12 = 527,4грн.

Накладні витрати:

$C_н$ = 12000 • 0,67 =8040грн.

Тоді, річні експлуатаційні витрат розраховують за формулою

$C_{екс}$ = 14400+ 3 168+ 2587,5 + 287,5+ 527,4+ 8040= 29010,4грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$C_{м-г}$=29010,4/1597,32=18,17грн/год.

$C_м$=18,17· $t_М$.

Оскільки у цьому разі всі роботи, пов'язані з розробкою програмного продукту, ведуться на ЕОМ, витрати на оплату машинного часу залежно від обраного варіанта реалізації пакета становить:

I. $С_м$= 18,17* 1379,232 = 25 060,64грн;

II. $С_м$ = 18,17* 1475,232 =26 804,96грн;

Накладні витрати становлять 67 % від заробітної плати:

I. $С_н$ = 49255,2* 0,67 = 33 000,98грн;

II. $С_н$ = 52 683,5* 0,67 = 35 297,94грн;

Знайдемо вартість розробки програмного продукту за варіантами:

I. $С_{пп}$= 49255,2+ 10 836,14+ 25 060,64+ 33 000,98= 118 152,96грн;

II. $С_{пп}$= 52 683,5+ 11 590,37+ 26 804,96+ 35 297,94= 126 376,77грн.

4.6.3 Оцінка техніко-економічного рівня варіантів ПП

Вибір кращого із двох варіантів ПП виконуємо на основі показників техніко-економічного рівня кожного варіанта.

$$K_{TEPj} = \frac{K_{TPj}}{Сф_j}$$

Визначимо показники техніко-економічного рівня для кожного варіанта:

$К_{ТЕР}$ 1 = 5,594/ 118 152,96= $4,7*10^{-5}$;

$К_{ТЕР}$ 2 = 5,436/ 126 376,77= $4,3*10^{-5}$;

Аналізуючи показники техніко-економічного рівня визначаємо раціональний варіант реалізації ПП. Це буде 1-й варіант, тому що він має максимальне значення $К_{ТЕР}$, що буде забезпечувати зберігання проміжних результатів розрахунку в оперативній пам'яті.

4.7 Висновки до розділу 4

Проведений аналіз розроблювального ПП дозволяє зробити наступні висновки. З можливих варіантів виконання ПП, що відрізняються за рівнем якості, переважає 1-й варіант. У даному варіанті узагальнений показник рівня

якості ПП дорівнює 5,594, що майже не відрізняється від другого показника, який на 0,158 нижчий. Також найкращий показник техніко-економічного рівня якості $К_{ТЕР}$ = $4,7*10^{-5}$. Для даного варіанта розраховані загальна трудомісткість і загальні витрати на розробку ПП. Основні характеристики зведені в таблицю 16.

Таблиця 16 – Основні показники економічної ефективності варіанта, що рекомендує ПП

Найменування показника	Позначення	Одиниця	Числове значення
Узагальнений показник рівня якості ПП	$К_{ТР}$		5,594
Показник техніко-економічного рівня якості ПП	$К_{ТЕР}$		$4,7*10^{-5}$
Загальна трудомісткість розробки ПП	Те	люд.-днів	1379,232
Загальні витрати на розробку ПП	$С_{пп}$	грн.	118 152,96

ВИСНОВКИ

У запропонованому підході, було досліджено існуючі джерела присвячені малярії, спроектовано та розроблено згорткову нейронну мережу, яка здатна досягати дуже високої класифікаційної точності для автоматизованої діагностики малярії, шляхом автоматичного вивчення особливостей з вхідних даних зображення. Серед правил навчання, алгоритм зворотного поширення помилки дає більш ефективні результати приблизно 96%. Також була порівняна точність запропонованої моделі й аналогів, розроблена модель показує значно кращі показники точності.

У результаті цього дослідження малярія може бути виявлена у сільських районах, невеликих лікарнях, медичних таборах, а також у будь-якому місці, де це необхідно, без необхідності завчасно підготовлених експертів. Ця програма робить їх діаностику без затримки в лікуванні, оскільки малярія прогнозується на початковому рівні, тому лікування буде розпочато незабаром, що дозволяє уникнути ризику тяжкого випадка у пацієнта.

У майбутньому це дослідження може бути розширене та удосконалене:

- дослідження можуть бути розширені на більшій кількості даних;
- малярія може бути передбачена за типом (P. falciparum, P. vivax, P. ovale, P. malariae, P.knowlesi);
- вдосконалити і переробити, як мобільний додаток, щоб мікроскоп підключався на пряму до смартфону.

Іншим напрямком застосування результатів проведеного дослідження може бути застосування їх для розробки та модифікації різних комп'ютерних систем медичної діагностики, що обробляють зображення. Отримані в даній роботі результати дозволять збільшити точність детектування малярії і забезпечити адаптивність систем діагностики до зміни вихідних даних, що в свою чергу підвищить ймовірність успішного лікування пацієнта і зробить внесок у зменшенні смертності від однієї з самих небезпечних на сьогоднішній день захворювань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.WHO. Malaria microscopy quality assurance manual-version

2.World Health Organization; 2018.2.WHO. World malaria report 2018. World Health Organization;2018.

3.Tek FB, Dempster AG, Kale I. Computer vision for microscopy diagnosis of malaria. Malar J 2009;8:153.

4.Das D, Mukherjee R, Chakraborty C. Computational microscopy imaging for malaria parasite detection: a systematic review. JMicrosc 2015;260:1–19.

5.Jan Z, Khan A, Sajjad M, Muhammad K, Rho S, Mehmood I.A review on automated diagnosis of malaria parasite inmicroscopic blood smears images. Multimedia Tools Appl2017;1–26.

6.Devi SS, Sheikh SA, Laskar RH. Erythrocyte features for malariaparasite detection in microscopic images of thin blood smear: areview. Int J Interact Multimedia Artificial Intell 2018;4:34–9.

7.Tangpukdee N, Duangdee C, Wilairatana P, Krudsood S. Malariadiagnosis: a brief review. Korean J Parasitol 2009;47:93

8. Quist, M., Bouma, H., van Kuijk, C., van Delden, O., Gerritsen, F.: Computer aided detection of pulmonary embolism on multi-detector CT. In: RSNA, Chicago, USA, November 2004.

9. Wang, X., Song, X., Chapman, B.E., Zheng, B.: Improving performance of computer-aided detection of pulmonary embolisms by incorporating a new pulmonary vascular-tree segmentation algorithm. In: SPIE Medical Imaging, International Society for Optics and Photonics, 2012. pp. 83152U–83152U.

10.Ozkan, H., Osman, O., S¸ahin, S., Boz, A.F.: A novel method for pulmonary embolism detection in cta images. Computer Methods and Programs in Biomedicine, 2014, 113(3), 757–766.

11. N. Tajbakhsh, M.B. Gotway, J. Liang. Computer-Aided Pulmonary Embolism Detection Using a Novel Vessel-Aligned Multi-planar Image Representation and Convolutional Neural Networks // Proceedings of 18th International Conference on

Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Munich, Germany, October 5-9, 2015, Volume 2, Pp. 62-69.

12. A. Krizhevsky, I. Sutskever, G.E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. Proceedings of Advances in Neural Information Processing Systems 25 (NIPS 2012), 2012, Pp. 1097-1105.

13. Ghosh M, Das D, Chakraborty C, Ray AK. *Plasmodium vivax* segmentation using modified fuzzy divergence. In: International conference on image information processing (ICIIP). IEEE; 2011:1–5.

14. Makkapati VV, Rao RM. Ontology-based malaria parasite stage and species identification from peripheral blood smear images. In: International conference on engineering in medicine and biology society. IEEE; 2011:6138–41.

15. Das D, Maiti A, Chakraborty C. Automated system for characterization and classification of malaria-infected stages using light microscopic images of thin blood smears. J Microsc 2015;257:238–52.

16. Das DK, Ghosh M, Pal M, Maiti AK, Chakraborty C. Machine learning approach for automated screening of malaria parasite using light microscopic images. Micron 2013;45:97–106.

17. Diaspro A, Chirico G, Usai C, Ramoino P, Dobrucki J. Photobleaching. In: Handbook of biological confocal microscopy. Springer; 2006:690–702.

18. Waters JC. Accuracy and precision in quantitative fluorescence microscopy. Rockefeller University Press; 2009.

19. Guy R, Liu P, Pennefather P, Crandall I. The use of fluorescence enhancement to improve the microscopic diagnosis of falciparum malaria. Malar J 2007;6:89–96.

20. Yang D, Subramanian G, Duan J, et al. A portable image-based cytometer for rapid malaria detection and quantification. PLoS ONE 2017;12:e0179161.

21. Mavandadi S, Dimitrov S, Feng S, et al. Distributed medical image analysis and diagnosis through crowd-sourced games: a malaria case study. PLoS ONE 2012;7:e37245.

22. Linder N, Turkki R, Walliander M, et al. A malaria diagnostic tool based on computer vision screening and visualization of Plasmodium falciparum candidate areas in digitized blood smears. PLoS ONE 2014;9:e104855.

23. Mohammed HA, Abdelrahman IAM. Detection and classification of malaria in thin blood slide images. In: International conference on communication, control, computing and electronics engineering (ICCCCEE). IEEE; 2017:1–5.

24. Tek FB, Dempster AG, Kale I. A colour normalization method for giemsa-stained blood cell images. In: 14th signal processing and communications applications. IEEE; 2006:1–4.

25. Di Rubeto C, Dempster A, Khan S, Jarra B. Segmentation of blood images using morphological operators. In: 15th international conference on pattern recognition, Vol. 3. IEEE; 2000:397–400.

26. Halim S, Bretschneider TR, Li Y, Preiser PR, Kuss C. Estimating malaria parasitaemia from blood smear images. In: 9th international conference on control, automation, robotics and vision. IEEE; 2006:1–6.

27. Nasir AA, Mashor M, Mohamed Z. Segmentation based approach for detection of malaria parasites using moving k-means clustering. In: EMBS conference on biomedical engineering and sciences (IECBES). IEEE; 2012:653–8.

28. Nanoti A, Jain S, Gupta C, Vyas G. Detection of malaria parasite species and life cycle stages using microscopic images of thin blood smear. In: International conference on inventive computation technologies, Vol. 1. IEEE; 2016:1–6.

29. Maiseli B, Mei J, Gao H, Yin S. An automatic and cost-effective parasitemia identification framework for low-end microscopy imaging devices. In: International conference on mechatronics and control (ICMC). IEEE; 2014:2048–53.

30. Liang Z, Powell A, Ersoy I, et al. CNN-based image analysis for malaria diagnosis. In: International conference on bioinformatics and biomedicine (BIBM). IEEE; 2016:493–6.

31. Bibin D, Nair MS, Punitha P. Malaria parasite detection from peripheral blood smear images using deep belief networks. Int J Appl Eng Res 2017;5:9099–108.

32. Adi K, Pujiyanto S, Gernowo R, Pamungkas A, Putranto AB. Identifying the developmental phase of Plasmodium falciparum in malaria-infected red blood cells using adaptive color segmentation and back propagation neural network. Int J Appl Eng Res 2016;11:8754–9.

33. Liang Z, Powell A, Ersoy I, et al. CNN-based image analysis for malaria diagnosis. In: International conference on bioinformatics and biomedicine (BIBM). IEEE; 2016:493–6.

34. Le MT, Bretschneider TR, Kuss C, Preiser PR. A novel semiautomatic image processing approach to determine Plasmodium falciparum parasitemia in Giemsa-stained thin blood smears. BMC Cell Biol 2008;9:15–27.

35. K. Simonyan, A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs.CV], 2015.

36. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1-9, 2015.

37. J.R.R. Uijlings, K.E.A. Van De Sande, T. Gevers, A.W.M. Smeulders. Selective search for object recognition. International Journal of Computer Vision, 104(2):154-171, 2013.

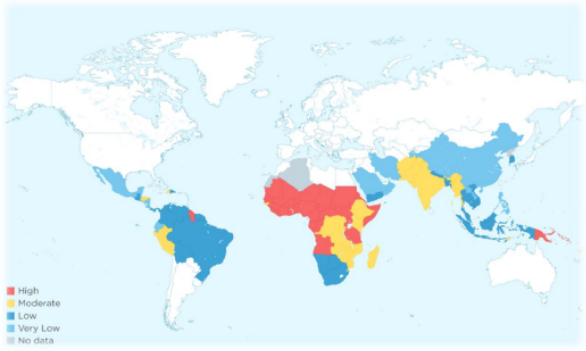
ДОДАТОК А ІЛЮСТРАТИВНИЙ МАТЕРІАЛ

Система ідентифікації малярійних клітин, використовуючи нейронні мережі

Виконав: студент групи КА-55
Срібний Андрій Євгенович
Науковий керівник:
д.т.н. проф. Зайченко Олена Юріївна

Актуальність роботи

- Автоматизація процесу діагностики забезпечить точну та швидку діагностику захворювання
- Раннє виявлення малярії є необхідним для забезпечення належної діагностики та збільшення шансів на лікування
- Сучасні методи діагностики цього захворювання є виснажливими і трудомісткими
- У 2018 р. малярією захворіло 219 мільйонів людей у 87 країнах
- Кількість випадків смерті від малярії у 2018 р досягло 435 000



Мета, об’єкт та предмет роботи

Мета даного дослідження – проаналізувати існуючі методи виявлення і діагностики малярії та розробити програмне забезпечення для прискорення й підвищення точності цього процесу.

Предмет дослідження – модель згорткової нейронної мережі, для вирішення задачі класифікації зображень клітин.

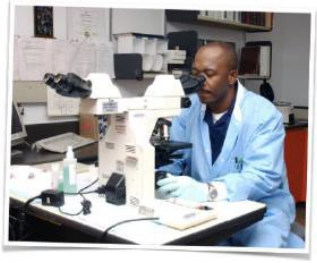
Об’єкт дослідження – вибірка зображень інфікованих та неінфікованих клітин.

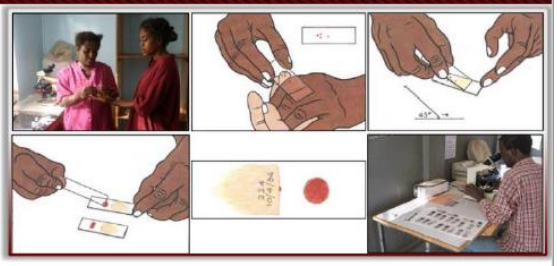
Постановка задачі

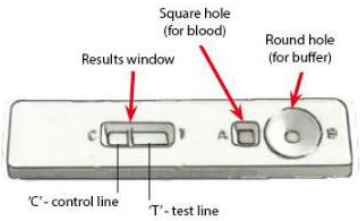
- Вивчення існуючих літературних джерел, присвячених малярії та її діагностиці, розробці та підвищенню ефективності роботи комп'ютерних систем медичної діагностики на зображеннях
- Проектування і розробка ефективної адаптивної моделі класифікатора малярії на медичних зображеннях
- Розробка програмної системи для оцінки ефективності запропонованої моделі класифікатора
- Проведення експериментів по оцінці ефективності запропонованої моделі, аналіз і інтерпретація результатів

Аналіз методів діагностики малярії

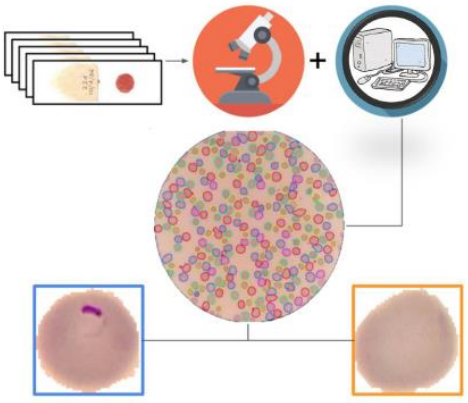
- Світлова мікроскопія
- Швидкі діагностичні тести.
- Інші тести(Лазерний, Флуоресцентний)



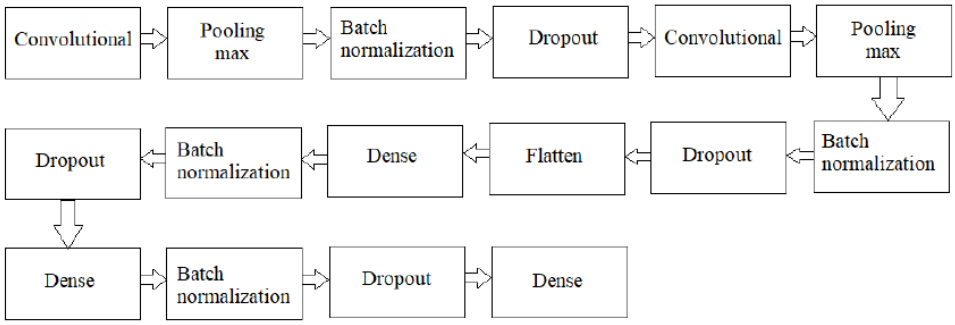




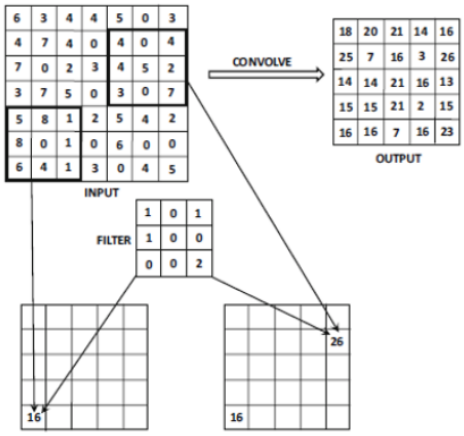
Вихідні дані



Згорткова нейронна мережа



Операція згортки



Програмна реалізація

```
Enter number
1 - Check image
2 - Add image
3 - Train NN
```

Головне меню програми

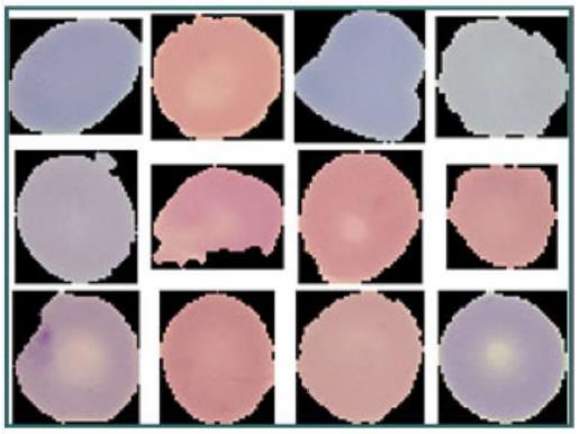
```
Enter number
1 - Check image
2 - Add image
3 - Train NN
1
Please enter the file name testcell1
testcell1 is infected
```

Перевірка зображення

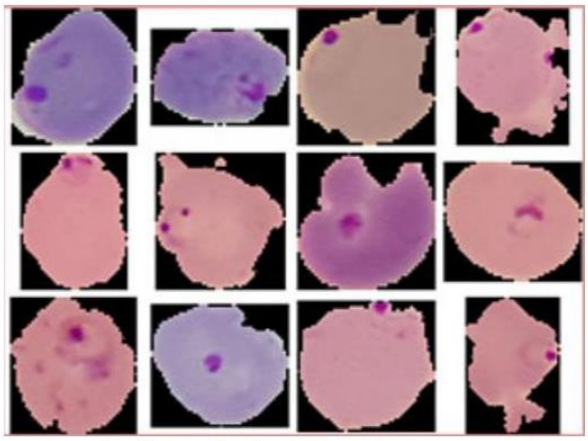
```
Enter number
1 - Check image
2 - Add image
3 - Train NN
2
Please enter the filename for the file that should be added newcell1.png
Enter type of the image
1 - Infected
2 - Uninfected
1
newcell1.png successfully added to the infected images
```

Додавання зображення

Вибірка даних

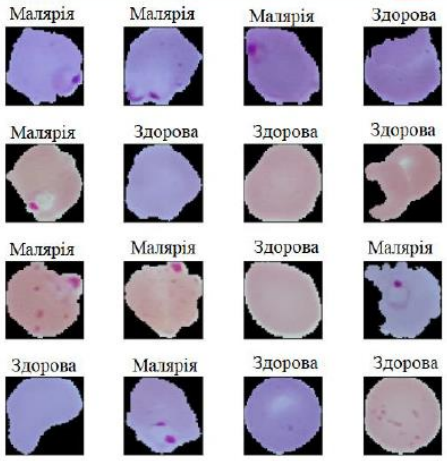


Неінфіковані клітини

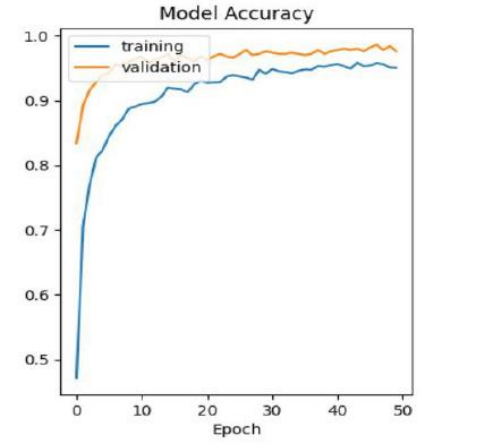


Інфіковані клітини

Аналіз результатів

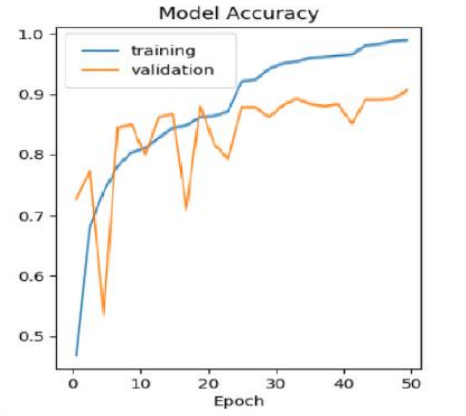


Приклади класифікації клітин



Процес навчання моделі

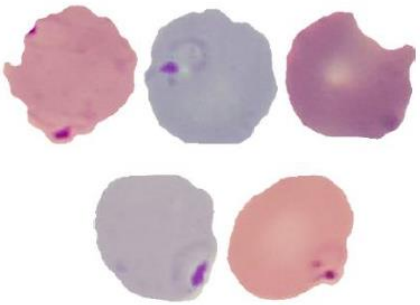
Точність класифікації ~96%



Графік процесу навчання моделі

Точність класифікації~85%

Результати моделі-аналогу



Приклади неправильно класифікованих клітин

Практична значущість результатів роботи

- Використання розробленого програмного продукту дозволить робити діагностику без затримки в лікуванні, оскільки малярія прогнозується на початковому рівні, тому лікування буде розпочато незабаром, що дозволяє уникнути ризику тяжкого випадка у пацієнта.
- Малярія може бути виявлена у сільських районах, невеликих лікарнях, медичних таборах, а також у будь-якому місці, де це необхідно, без необхідності завчасно підготовлених експертів.

Висновки

- Досліджено актуальні існуючі джерела присвячені малярії
- Надано загальну характеристику діагностиці малярії
- Спроектовано згорткову нейронну мережу, яка здатна досягати дуже високої класифікаційної точності для автоматизованої діагностики малярії
- Розроблено програмний продукт для автоматичної діагностики

Напрямки та перспективи подальших досліджень

Подальші дослідження за даною темою полягатимуть у розширенні можливостей розробленої моделі та програмного продукту:

- Дослідження можуть бути розширені на більшій кількості даних;
- Малярія може бути класифікована за типом (P. falciparum, P. vivax, P. ovale, P. malariae, P.knowlesi);
- Вдосконалити і переробити програмний продукт, як мобільний додаток, щоб мікроскоп підключався на пряму до смартфону.

Дякую за увагу!

ДОДАТОК Б ЛІСТИНГ ПРОГРАМИ

```
def curriculum_rules(epoch):
return { 'sentence_length': -1, 'flip_probability': 0.5, 'jitter_probability': 0.05 }


def train(run_name, speaker, start_epoch, stop_epoch, img_c, img_w, img_h, frames_n, absolute_max_string_len, minibatch_size):
DATASET_DIR = os.path.join(CURRENT_PATH, speaker, 'datasets')
OUTPUT_DIR = os.path.join(CURRENT_PATH, speaker, 'results')
LOG_DIR = os.path.join(CURRENT_PATH, speaker, 'logs')

curriculum = Curriculum(curriculum_rules)
lip_gen = BasicGenerator(dataset_path=DATASET_DIR,
minibatch_size=minibatch_size,
img_c=img_c, img_w=img_w, img_h=img_h, frames_n=frames_n,
absolute_max_string_len=absolute_max_string_len,
curriculum=curriculum, start_epoch=start_epoch).build()

lipnet = LipNet(img_c=img_c, img_w=img_w, img_h=img_h, frames_n=frames_n,
absolute_max_string_len=absolute_max_string_len, output_size=lip_gen.get_output_size())
lipnet.summary()

adam = Adam(lr=0.0001, beta_1=0.9, beta_2=0.999, epsilon=1e-08)

# the loss calc occurs elsewhere, so use a dummy lambda func for the loss
lipnet.model.compile(loss={'ctc': lambda y_true, y_pred: y_pred}, optimizer=adam)

# load weight if necessary
if start_epoch > 0:
weight_file = os.path.join(OUTPUT_DIR, os.path.join(run_name, 'weights%02d.h5' % (start_epoch - 1)))
lipnet.model.load_weights(weight_file)

spell = Spell(path=PREDICT_DICTIONARY)
decoder = Decoder(greedy=PREDICT_GREEDY, beam_width=PREDICT_BEAM_WIDTH,
postprocessors=[labels_to_text, spell.sentence])

# define callbacks
statistics = Statistics(lipnet, lip_gen.next_val(), decoder, 256, output_dir=os.path.join(OUTPUT_DIR, run_name))
visualize = Visualize(os.path.join(OUTPUT_DIR, run_name), lipnet, lip_gen.next_val(), decoder, num_display_sentences=minibatch_size)
tensorboard = TensorBoard(log_dir=os.path.join(LOG_DIR, run_name))
csv_logger = CSVLogger(os.path.join(LOG_DIR, "{}-{}.csv".format('training',run_name)), separator=',', append=True)
checkpoint = ModelCheckpoint(os.path.join(OUTPUT_DIR, run_name, "weights{epoch:02d}.h5"), monitor='val_loss', save_weights_only=True,
mode='auto', period=1)

lipnet.model.fit_generator(generator=lip_gen.next_train(),
steps_per_epoch=lip_gen.default_training_steps, epochs=stop_epoch,
validation_data=lip_gen.next_val(), validation_steps=lip_gen.default_validation_steps,
callbacks=[checkpoint, statistics, visualize, lip_gen, tensorboard, csv_logger],
initial_epoch=start_epoch,
verbose=1,
max_q_size=5,
workers=2,
pickle_safe=True)

if __name__ == '__main__':
run_name = datetime.datetime.now().strftime('%Y:%m:%d:%H:%M:%S')
speaker = sys.argv[1]
train(run_name, speaker, 0, 5000, 3, 100, 50, 75, 32, 50)
def text_to_labels(text):
ret = []
for char in text:
if char >= 'a' and char <= 'z':
ret.append(ord(char) - ord('a'))
elif char == ' ':
ret.append(26)
return ret

def labels_to_text(labels):
# 26 is space, 27 is CTC blank char
text = ''
for c in labels:
if c >= 0 and c < 26:
text += chr(c + ord('a'))
elif c == 26:
text += ' '
return text
import matplotlib.pyplot as plt
import numpy as np
```

```
import wave
import sys


spf = wave.open('test.wav','r')

#Extract Raw Audio from Wav File
signal = spf.readframes(-1)
print(spf.getframerate())
signal = np.fromstring(signal, 'Int16')



#If Stereo
#if spf.getnchannels() == 2:
# print 'Just mono files'
# sys.exit(0)

#plt.figure(1)
plt.title('Signal Wave...')
plt.plot(signal)
plt.show()
class Statistics(keras.callbacks.Callback):

def __init__(self, model_container, generator, decoder, num_samples_stats=256, output_dir=None):
self.model_container = model_container
self.output_dir = output_dir
self.generator = generator
self.num_samples_stats = num_samples_stats
self.decoder = decoder
if output_dir is not None and not os.path.exists(self.output_dir):
os.makedirs(self.output_dir)

def get_statistics(self, num):
num_left = num
data = []

while num_left > 0:
output_batch = next(self.generator)[0]
num_proc = min(output_batch['the_input'].shape[0], num_left)
y_pred = self.model_container.predict(output_batch['the_input'][0:num_proc])
input_length = output_batch['input_length'][0:num_proc]
decoded_res = self.decoder.decode(y_pred, input_length)

for j in range(0, num_proc):
data.append((decoded_res[j], output_batch['source_str'][j]))

num_left -= num_proc

mean_cer, mean_cer_norm = self.get_mean_character_error_rate(data)
mean_wer, mean_wer_norm = self.get_mean_word_error_rate(data)
mean_bleu, mean_bleu_norm = self.get_mean_bleu_score(data)

return {
'samples': num,
'cer': (mean_cer, mean_cer_norm),
'wer': (mean_wer, mean_wer_norm),
'bleu': (mean_bleu, mean_bleu_norm)
}

def get_mean_tuples(self, data, individual_length, func):
total = 0.0
total_norm = 0.0
length = len(data)
for i in range(0, length):
val = float(func(data[i][0], data[i][1]))
total += val
total_norm += val / individual_length
return (total/length, total_norm/length)

def get_mean_character_error_rate(self, data):
mean_individual_length = np.mean([len(pair[1]) for pair in data])
return self.get_mean_tuples(data, mean_individual_length, editdistance.eval)

def get_mean_word_error_rate(self, data):
mean_individual_length = np.mean([len(pair[1].split()) for pair in data])
return self.get_mean_tuples(data, mean_individual_length, wer_sentence)

def get_mean_bleu_score(self, data):
```

```
wrapped_data = [([reference],hypothesis) for reference,hypothesis in data]
return self.get_mean_tuples(wrapped_data, 1.0, bleu_score.sentence_bleu)

def on_train_begin(self, logs={}):
with open(os.path.join(self.output_dir, 'stats.csv'), 'wb') as csvfile:
csvw = csv.writer(csvfile)
csvw.writerow(["Epoch", "Samples", "Mean CER", "Mean CER (Norm)", "Mean WER", "Mean WER (Norm)", "Mean BLEU", "Mean BLEU (Norm)"])

def on_epoch_end(self, epoch, logs={}):
stats = self.get_statistics(self.num_samples_stats)

print('\n\n[Epoch %d] Out of %d samples: [CER: %.3f - %.3f] [WER: %.3f - %.3f] [BLEU: %.3f - %.3f]\n'
% (epoch, stats['samples'], stats['cer'][0], stats['cer'][1], stats['wer'][0], stats['wer'][1], stats['bleu'][0], stats['bleu'][1]))

if self.output_dir is not None:
with open(os.path.join(self.output_dir, 'stats.csv'), 'ab') as csvfile:
csvw = csv.writer(csvfile)
csvw.writerow([epoch, stats['samples'],
"{0:.5f}".format(stats['cer'][0]), "{0:.5f}".format(stats['cer'][1]),
"{0:.5f}".format(stats['wer'][0]), "{0:.5f}".format(stats['wer'][1]),
"{0:.5f}".format(stats['bleu'][0]), "{0:.5f}".format(stats['bleu'][1])])



class Visualize(keras.callbacks.Callback):

def __init__(self, output_dir, model_container, generator, decoder, num_display_sentences=10):
self.model_container = model_container
self.output_dir = output_dir
self.generator = generator
self.num_display_sentences = num_display_sentences
self.decoder = decoder
if not os.path.exists(self.output_dir):
os.makedirs(self.output_dir)

def on_epoch_end(self, epoch, logs={}):
output_batch = next(self.generator)[0]

y_pred = self.model_container.predict(output_batch['the_input'][0:self.num_display_sentences])
input_length = output_batch['input_length'][0:self.num_display_sentences]
res = self.decoder.decode(y_pred, input_length)

with open(os.path.join(self.output_dir, 'e%02d.csv' % (epoch)), 'wb') as csvfile:
csvw = csv.writer(csvfile)
csvw.writerow(["Truth", "Decoded"])
for i in range(self.num_display_sentences):
csvw.writerow([output_batch['source_str'][i], res[i]])
import numpy as np
import keras
import pickle
import os
import glob
import multiprocessing



# datasets/[train|val]/<sid>/<id>/<image>.png
# or datasets/[train|val]/<sid>/<id>.mpg
# datasets/align/<id>.align
class BasicGenerator(keras.callbacks.Callback):
def __init__(self, dataset_path, minibatch_size, img_c, img_w, img_h, frames_n, absolute_max_string_len=30, **kwargs):
self.dataset_path = dataset_path
self.minibatch_size = minibatch_size
self.blank_label = self.get_output_size() - 1
self.img_c = img_c
self.img_w = img_w
self.img_h = img_h
self.frames_n = frames_n
self.absolute_max_string_len = absolute_max_string_len
self.cur_train_index = multiprocessing.Value('i', 0)
self.cur_val_index = multiprocessing.Value('i', 0)
self.curriculum = kwargs.get('curriculum', None)
self.random_seed = kwargs.get('random_seed', 13)
self.vtype = kwargs.get('vtype', 'mouth')
self.face_predictor_path = kwargs.get('face_predictor_path', None)
self.steps_per_epoch = kwargs.get('steps_per_epoch', None)
self.validation_steps = kwargs.get('validation_steps', None)
# Process epoch is used by non-training generator (e.g: validation)
# because each epoch uses different validation data enqueuer
```

```
# Will be updated on epoch_begin
self.process_epoch = -1
# Maintain separate process train epoch because fit_generator only use
# one enqueuer for the entire training, training enqueuer can contain
# max_q_size batch data ahead than the current batch data which might be
# in the epoch different with current actual epoch
# Will be updated on next_train()
self.shared_train_epoch = multiprocessing.Value('i', -1)
self.process_train_epoch = -1
self.process_train_index = -1
self.process_val_index = -1

def build(self, **kwargs):
self.train_path = os.path.join(self.dataset_path, 'train')
self.val_path = os.path.join(self.dataset_path, 'val')
self.align_path = os.path.join(self.dataset_path, 'align')
self.build_dataset()
# Set steps to dataset size if not set
self.steps_per_epoch = self.default_training_steps if self.steps_per_epoch is None else self.steps_per_epoch
self.validation_steps = self.default_validation_steps if self.validation_steps is None else self.validation_steps
return self

@property
def training_size(self):
return len(self.train_list)

@property
def default_training_steps(self):
return self.training_size / self.minibatch_size

@property
def validation_size(self):
return len(self.val_list)

@property
def default_validation_steps(self):
return self.validation_size / self.minibatch_size

def get_output_size(self):
return 28

def get_cache_path(self):
return self.dataset_path.rstrip('/') + '.cache'

def enumerate_videos(self, path):
video_list = []
for video_path in glob.glob(path):
try:
if os.path.isfile(video_path):
video = Video(self.vtype, self.face_predictor_path).from_video(video_path)
else:
video = Video(self.vtype, self.face_predictor_path).from_frames(video_path)
except AttributeError as err:
raise err
except:
print "Error loading video: "+video_path
continue
if K.image_data_format() == 'channels_first' and video.data.shape != (self.img_c,self.frames_n,self.img_w,self.img_h):
print "Video "+video_path+" has incorrect shape "+str(video.data.shape)+", must be "+str((self.img_c,self.frames_n,self.img_w,self.img_h))+""
continue
if K.image_data_format() != 'channels_first' and video.data.shape != (self.frames_n,self.img_w,self.img_h,self.img_c):
print "Video "+video_path+" has incorrect shape "+str(video.data.shape)+", must be "+str((self.frames_n,self.img_w,self.img_h,self.img_c))+""
continue
video_list.append(video_path)
return video_list

def enumerate_align_hash(self, video_list):
align_hash = {}
for video_path in video_list:
video_id = os.path.splitext(video_path)[0].split('/')[-1]
align_path = os.path.join(self.align_path, video_id)+".align"
align_hash[video_id] = Align(self.absolute_max_string_len, text_to_labels).from_file(align_path)
return align_hash

def build_dataset(self):
if os.path.isfile(self.get_cache_path()):
print "\nLoading dataset list from cache..."
with open (self.get_cache_path(), 'rb') as fp:
self.train_list, self.val_list, self.align_hash = pickle.load(fp)
```

```
else:
print "\nEnumerating dataset list from disk..."
self.train_list = self.enumerate_videos(os.path.join(self.train_path, '*', '*'))
self.val_list = self.enumerate_videos(os.path.join(self.val_path, '*', '*'))
self.align_hash = self.enumerate_align_hash(self.train_list + self.val_list)
with open(self.get_cache_path(), 'wb') as fp:
pickle.dump((self.train_list, self.val_list, self.align_hash), fp)

print "Found {} videos for training.".format(self.training_size)
print "Found {} videos for validation.".format(self.validation_size)
print ""

np.random.shuffle(self.train_list)

def get_align(self, _id):
return self.align_hash[_id]

def get_batch(self, index, size, train):
if train:
video_list = self.train_list
else:
video_list = self.val_list

X_data_path = get_list_safe(video_list, index, size)
X_data = []
Y_data = []
label_length = []
input_length = []
source_str = []
for path in X_data_path:
video = Video().from_frames(path)
align = self.get_align(path.split('/')[-1])
video_unpadded_length = video.length
if self.curriculum is not None:
video, align, video_unpadded_length = self.curriculum.apply(video, align)
X_data.append(video.data)
Y_data.append(align.padded_label)
label_length.append(align.label_length) # CHANGED [A] -> A, CHECK!
# input_length.append([video_unpadded_length - 2]) # 2 first frame discarded
input_length.append(video.length) # Just use the video padded length to avoid CTC No path found error (v_len < a_len)
source_str.append(align.sentence) # CHANGED [A] -> A, CHECK!

source_str = np.array(source_str)
label_length = np.array(label_length)
input_length = np.array(input_length)
Y_data = np.array(Y_data)
X_data = np.array(X_data).astype(np.float32) / 255 # Normalize image data to [0,1], TODO: mean normalization over training data

inputs = {'the_input': X_data,
'the_labels': Y_data,
'input_length': input_length,
'label_length': label_length,
'source_str': source_str # used for visualization only
}
outputs = {'ctc': np.zeros([size])} # dummy data for dummy loss function

return (inputs, outputs)

@threadsafe_generator
def next_train(self):
r = np.random.RandomState(self.random_seed)
while 1:
# print "SI: {}, SE: {}".format(self.cur_train_index.value, self.shared_train_epoch.value)
with self.cur_train_index.get_lock(), self.shared_train_epoch.get_lock():
cur_train_index = self.cur_train_index.value
self.cur_train_index.value += self.minibatch_size
# Shared epoch increment on start or index >= training in epoch
if cur_train_index >= self.steps_per_epoch * self.minibatch_size:
cur_train_index = 0
self.shared_train_epoch.value += 1
self.cur_train_index.value = self.minibatch_size
if self.shared_train_epoch.value < 0:
self.shared_train_epoch.value += 1
# Shared index overflow
if self.cur_train_index.value >= self.training_size:
self.cur_train_index.value = self.cur_train_index.value % self.minibatch_size
# Calculate differences between process and shared epoch
epoch_differences = self.shared_train_epoch.value - self.process_train_epoch
if epoch_differences > 0:
```

```
self.process_train_epoch += epoch_differences
for i in range(epoch_differences):
r.shuffle(self.train_list) # Catch up
# print "GENERATOR EPOCH {}".format(self.process_train_epoch)
# print self.train_list[0]
# print "PI: {}, SI: {}, SE: {}".format(cur_train_index, self.cur_train_index.value, self.shared_train_epoch.value)
if self.curriculum is not None and self.curriculum.epoch != self.process_train_epoch:
self.update_curriculum(self.process_train_epoch, train=True)
# print "Train [{},{}] {}:{}".format(self.process_train_epoch, epoch_differences, cur_train_index,cur_train_index+self.minibatch_size)
ret = self.get_batch(cur_train_index, self.minibatch_size, train=True)
# if epoch_differences > 0:
# print "GENERATOR EPOCH {} - {}:{}".format(self.process_train_epoch, cur_train_index, cur_train_index + self.minibatch_size)
# print ret[0]['source_str']
# print "----------"
yield ret

@threadsafe_generator
def next_val(self):
while 1:
with self.cur_val_index.get_lock():
cur_val_index = self.cur_val_index.value
self.cur_val_index.value += self.minibatch_size
if self.cur_val_index.value >= self.validation_size:
self.cur_val_index.value = self.cur_val_index.value % self.minibatch_size
if self.curriculum is not None and self.curriculum.epoch != self.process_epoch:
self.update_curriculum(self.process_epoch, train=False)
# print "Val [{}] {}:{}".format(self.process_epoch, cur_val_index,cur_val_index+self.minibatch_size)
ret = self.get_batch(cur_val_index, self.minibatch_size, train=False)
yield ret
def on_train_begin(self, logs={}):
with self.cur_train_index.get_lock():
self.cur_train_index.value = 0
with self.cur_val_index.get_lock():
self.cur_val_index.value = 0

def on_epoch_begin(self, epoch, logs={}):
self.process_epoch = epoch
def update_curriculum(self, epoch, train=True):
self.curriculum.update(epoch, train=train)
print "Epoch {}: {}".format(epoch, self.curriculum)
datasets/video/<sid>/<id>/<image>.png
# or datasets/[train|val]/<sid>/<id>.mpg
# datasets/align/<id>.align

class RandomSplitGenerator(BasicGenerator):
def build(self, **kwargs):
self.video_path = os.path.join(self.dataset_path, 'video')
self.align_path = os.path.join(self.dataset_path, 'align')
self.val_split = kwargs.get('val_split', 0.2)
self.build_dataset()
# Set steps to dataset size if not set
self.steps_per_epoch = self.default_training_steps if self.steps_per_epoch is None else self.steps_per_epoch
self.validation_steps = self.default_validation_steps if self.validation_steps is None else self.validation_steps
return self

def build_dataset(self):
if os.path.isfile(self.get_cache_path()):
print "\nLoading dataset list from cache..."
with open (self.get_cache_path(), 'rb') as fp:
self.train_list, self.val_list, self.align_hash = pickle.load(fp)
else:
print "\nEnumerating dataset list from disk..."
video_list = self.enumerate_videos(os.path.join(self.video_path, '*', '*'))
np.random.shuffle(video_list) # Random the video list before splitting
if(self.val_split > 1): # If val_split is not a probability
training_size = len(video_list) - self.val_split
else: # If val_split is a probability
training_size = len(video_list) - int(self.val_split * len(video_list))
self.train_list = video_list[0:training_size]
self.val_list = video_list[training_size:]
self.align_hash = self.enumerate_align_hash(self.train_list + self.val_list)
with open(self.get_cache_path(), 'wb') as fp:
pickle.dump((self.train_list, self.val_list, self.align_hash), fp)
```