Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ВПРАВ ЙОГИ, РЕАЛІЗОВАНИХ ЗАСОБАМИ МАШИННОГО НАВЧАННЯ ТА НЕЙРОННИХ МЕРЕЖ
(тема)

Виконав:
здобувач 2 року навчання,
групи ІНФм-24-2
Подшивалова О. Є.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Науковий керівник доц. Творошенко І. С.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Подшиваловій Ользі Євгеніївні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів класифікації вправ йоги, реалізованих засобами машинного навчання та нейронних мереж

затверджена наказом університету від 14 листопада 2025 року № 1045Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 6 листопада 2025 р.
3. Вихідні дані до роботи методи класифікації зображень, літературні джерела щодо застосування методів класифікації, програмні засоби для реалізації вибраних методів класифікації та десктоп-застосунку, зображення виконання поз йоги для тренування та тестування моделей.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз сучасних методів класифікації об'єктів на зображеннях методами машинного навчання та нейронних мереж.
2. Аналіз літературних джерел щодо апробації методів класифікації об'єктів на зображеннях.
3. Формування покрокового алгоритму для кожного із вибраних методів класифікації.
4. Візуалізація сформованих покрокових алгоритмів.
5. Розроблення програмного застосунку, що надасть змогу класифікувати зображення з виконанням поз йоги кожним з досліджених методів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми класифікації об'єктів на зображеннях, об'єкт та мета дослідження, постановка задачі, аналіз структури обраних методів, приклад еталонних зображень для кожного класу, ілюстрація головного екрану розробленого застосунку, ілюстрація результатів класифікації кожним із вибраних методів, висновки, перспективи та апробація роботи.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	29.09.2025	
2	Аналіз завдання, підбір літератури	30.09.25-07.10.25	
3	Аналіз літератури з досліджуваної проблеми	08.10.25-14.10.25	
4	Вивчення особливостей поширених методів вирішення завдання	15.10.25-20.10.25	
5	Дослідження принципів роботи методів, пов'язаних із машинним навчанням і нейронними мережами	21.10.25-27.10.25	
6	Програмна реалізація	28.10.25-05.11.25	
7	Обґрунтування отриманих результатів	06.11.25-11.11.25	
8	Оформлення пояснювальної записки	12.11.25-14.11.25	
9	Перевірка на нормоконтроль	19.11.25-10.12.25	
10	Перевірка на плагіат	20.11.25-10.12.25	
11	Рецензування	21.11.25-10.12.25	
12	Підготовка презентації та доповіді	21.11.25-22.12.25	
13	Занесення роботи в електронний архів	21.11.25-22.12.25	
14	Попередній захист кваліфікаційної роботи	01.12.25-22.12.25	

Дата видачі завдання 29 вересня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Творошенко І. С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 88 с., 3 табл., 46 рис., 45 джерел.

ЕФЕКТИВНІСТЬ КЛАСИФІКАЦІЇ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, КОМП'ЮТЕРНИЙ ЗІР, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, ОЦІНЮВАННЯ ПОЗИ, ПРАКТИКА ЙОГИ, ФУНКЦІЯ АКТИВАЦІЇ, EFFICIENTNET, PYTORCH, RESNET, XCEPTRON.

Об'єктом дослідження є методи класифікації, реалізовані засобами машинного навчання та нейронних мереж.

Метою дослідження є порівняння методів класифікації зображень, основаних на машинному навчанні та нейронних мережах, шляхом розробки застосунку, що класифікує зображення основних поз йоги.

Використано методи класифікації HOG-SVM, ResNet-50, Xception та EfficientNet-B4 для класифікації вправ йоги на зображеннях. Проведено аналіз сучасних методів класифікації об'єктів на зображеннях та відповідних літературних джерел. Сформовано та візуалізовано структуру методів та їх окремих блоків за допомогою схем.

Наукова новизна роботи полягає у результатах порівняння методів класифікації зображень, основаних на машинному навчанні та нейронних мережах.

Робота пов'язана з дослідженнями, у яких порівнюються архітектури ResNet, Xception, EfficientNet, з метою оцінювання їх продуктивності.

Отримані результати можуть бути використані у низці практичних застосувань, пов'язаних із розпізнаванням людських поз та автоматизацією.

У результаті дослідження розроблено десктоп-застосунок класифікації виконання поз йоги на зображеннях із точністю у десятковому значенні та часом класифікування у секундах.

ABSTRACT

Explanatory note to the qualification work: 88 pages, 3 tables, 46 figures, 45 sources.

CLASSIFICATION PERFORMANCE, CONVOLUTIONAL NEURAL NETWORK, COMPUTER VISION, IMAGE CLASSIFICATION, POSE ESTIMATION, YOGA PRACTICE, ACTIVATION FUNCTION, EFFICIENTNET, PYTORCH, RESNET, XCEPTRON.

The object of the research is classification methods implemented using machine learning and neural networks.

The aim of the research is to compare image classification methods based on machine learning and neural networks by developing an application that classifies images of basic yoga poses.

Classification methods using HOG-SVM, ResNet-50, Xception and EfficientNet-B4 were used to classify yoga exercises in images. An analysis of modern methods for classifying objects in images, along with relevant literature sources, was conducted. The structure of the methods and their separate blocks are formed and visualized using diagrams.

Scientific novelty of the work lies in the results of comparing image classification methods based on machine learning and neural networks.

The work is related to studies that compare the architectures of ResNet, Xception, EfficientNet, in order to evaluate their performance in classification tasks. The results obtained can be applied to various practical applications related to human pose recognition and automation.

As a result of the research, a desktop application for classifying yoga poses in images with decimal precision and classification time in seconds was developed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CNN – Convolutional Neural Network (згорткова нейронна мережа)

DNN – Deep Neural Network (глибока нейронна мережа)

GDA – Generalized Discriminant Analysis (узагальнений дискримінантний аналіз)

GRU – Gated Recurrent Units (вентильний рекурентний вузол)

HOG – Histogram of Oriented Gradients (гістограма орієнтованих градієнтів)

k-NN – *k*-Nearest Neighbor Method (метод *k*-найближчих сусідів)

LDA – Linear Discriminant Analysis (лінійний дискримінантний аналіз)

LSTM – Long Short-Term Memory (довга короткочасна пам’ять)

SURF – Speeded Up Robust Features (прискорені стійкі ознаки)

SVM – Support Vector Machine (метод опорних векторів)

VGG – Visual Geometry Group (візуальна геометрична група)

ViT – Visual Transformer (зоровий трансформер)

ВСТУП

З кожним роком темп сучасного світу невпинно прискорюється. З технологічним розвитком та пропагандою успішного життя людині постійно необхідно кудись поспішати, оскільки їй потрібно встигнути все: робота, спортзал, духовний розвиток, догляд за собою, а також догляд за рідними та побутом.

Вимоги суспільства, які стосуються самореалізації особистості в усіх аспектах її життя, не мають межі, а навіть з плином часу тільки ростуть. Це призводить до зростання тривожності, виникнення відчуття невизначеності, загубленості та хаотичності. Брак часу на повноцінний відпочинок веде до вигоряння та емоційного виснаження.

У такому ритмі життя відбувається втрата здатності до усвідомленого сприйняття власних потреб і бажань. Не залишається часу осмислити, чого людина справді хоче, чи прагне вона й надалі рухатися тим шляхом, яким прямує зараз. У результаті людина рухається за інерцією. Це породжує внутрішній конфлікт, відчуття втрати гармонії із собою та потребу у практиках, які дозволять відновити баланс між зовнішніми викликами та внутрішнім світом.

Одним із популярних зараз методів для сповільнення та віднаходження внутрішньої рівноваги виступає йога. Йога є сукупністю психофізичних вправ, спрямованих на гармонію тіла та духовності, відновлення балансу та підвищення стійкості до стресових факторів. Вона включає у собі асани (так називають фізичні вправи), пранаями (тобто дихальні практики) та медитативні вправи, спрямовані на філософське осмислення буття.

Заняття йогою на регулярній основі сприяють зниженню рівня тривожності та позитивно впливають на психоемоційний стан. Асани розвивають гнучкість і витривалість та в цілому покращують фізичний стан організму.

Дихальні практики та медитації дають змогу заспокоїтися та сповільнитися, вони сприяють покращенню сну та підвищенню концентрації уваги. До того ж, йога підходить для людей різного віку та рівня фізичної підготовки, оскільки вправи є різноманітними за складністю та стилем виконання.

У роботі розглядатиметься саме напрям йоги, зосереджений переважно на виконанні асан, тобто фізичних вправ. У більшості людей слово «йога» асоціюється виключно з виконанням асан. Саме вони є найбільш популярними у сучасному світі. Виконання фізичних вправ є зрозумілим та доступним для широкого кола людей та не нав’язує заглиблення у філософський аспект, на відміну від дихальних практик та медитацій.

Таким чином, заняття йогою набувають широкої популярності у сучасному світі, а попит на застосунки, які є адаптованим тренером, зростає. Саме тому кваліфікаційна робота, пов’язана з методами розпізнавання поз йоги, є актуальною, оскільки в подальшому може мати практичне застосування у реальних сервісах.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ КЛАСИФІКАЦІЇ ФІЗИЧНИХ ВПРАВ, РЕАЛІЗОВАНИХ ЗАСОБАМИ МАШИННОГО НАВЧАННЯ ТА НЕЙРОННИХ МЕРЕЖ, ЗОКРЕМА ВПРАВ ЙОГИ

1.1 Аналіз та класифікація вправ йоги

Фізичний напрям йоги включає в собі різноманітну кількість вправ, які можна по-різному поділити на категорії. Класифікацію можна здійснювати за складністю, стилем виконання, вихідною позою тощо. У контексті даної роботи було доцільно провести аналіз поділу вправ за функціональним призначенням, положенням тіла та рівнем складності.

Класифікація вправ йоги за функціональним призначенням відбувається за напрямом розвитку тіла людини (рис. 1.1 [1–4]), вона включає такі категорії:

– вправи для розвитку сили («Планка», «Стілець»);

– вправи для гнучкості («Собака мордою вниз», «Трикутник», «Поза голуба»);

– вправи для балансу та координації («Дерево», «Поза орла»);

– вправи для відпочинку та відновлення («Трупна поза», «Поза дитини»).

Рисунок 1.1 – Види асан за функціональним призначенням

Класифікація вправ за положенням тіла відображає вихідне положення, яке приймається у певній асані (рис. 1.2 [2, 5–8]), можна зазначити категорії:

– стоячі пози («Воїн I», «Трикутник»);

– сидячі пози («Лотос», «Палка»);

– пози лежачи на спині («Міст», «Щаслива дитина»);

– пози лежачи на животі («Кобра», «Лук»);

– інверсійні пози («Стійка на плечах», «Стійка на голові»).

Рисунок 1.2 – Види асан за положенням тіла

Класифікація за рівнем складності передбачає поділ асан на різні рівні підготовки (рис. 1.3 [3, 9, 10]):

– базові (для початківців), просте виконання («Гора», «Поза дитини»);

– середнього рівня, вимагають сили і рівноваги («Воїн III», «Поза орла»);

– просунуті («Стійка на голові», «Колесо»).

Такий поділ вправ є найбільш практичним при класифікації вправ йоги в контексті комп’ютерного зору та машинного навчання, оскільки дає змогу підібрати найбільш влучний алгоритм для розпізнавання.

Рисунок 1.3 – Види асан за рівнем складності

Зосереджуючись суто на методи класифікації, найбільш практичним є поділ асан за положенням тіла, оскільки він передбачає відсіювання вправ за попереднім положенням тіла.

1.2 Аналіз сучасних методів класифікації фізичних вправ, реалізованих засобами машинного навчання та нейронних мереж, зокрема вправ йоги

Існує велика кількість методів комп’ютерного зору та машинного навчання для розпізнавання положення тіла людини, зокрема при виконанні вправ йоги. Їх використання залежить від багатьох аспектів поставленої задачі, таких як, наприклад, час виконання, відсоток правильної класифікації, гнучкість. Розглянемо найбільш поширені методи, які застосовуються в задачах, подібних до зазначених у цій роботі (рис. 1.4).

Рисунок 1.4 – Методи класифікації фізичних вправ

1.2.1 Методи класифікації фізичних вправ, зокрема вправ йоги, на основі машинного навчання

Методи класифікації на основі машинного навчання є поширеними у завданнях класифікації поз людини, оскільки мають низку переваг, серед яких є ефективна робота з великими обсягами даних різних типів (зокрема, фото та відеоданих) та на основі цього побудова моделей з високою ефективністю класифікації.

Метод опорних векторів (Support Vector Machine, SVM) є ефективним при застосуванні в задачах, які мають обмежений набір ознак. Принципом його роботи є пошук таких гіперплощин, щоб між точками з різних класів було утворено якнайширшу чисту область.

У випадку класифікації рухів людини SVM спрацьовує ефективно, оскільки у таких задачах набором ознак є координати суглобів, кутові значення у суглобах та траєкторії руху. Також цей метод може працювати як з лінійно розділеними даними, так і з нелінійними, що дає змогу обробити складні рухи.

Головним недоліком методу опорних векторів є те, що він має високу складність обчислення, що може призвести до сповільнення застосунку при великій кількості даних.

Метод *k*-найближчих сусідів (*k*-Nearest Neighbor Method, *k*-NN) також широко використовується у задачах розпізнавання та класифікації поз, його суть полягає у пошуку *k*-найближчих сусідів для об’єкта класифікації задля визначення, до якого класу його слід віднести. *k*-NN не потребує попереднього навчання, що робить його методом без явного навчання. У контексті розпізнавання рухів, *k*-NN є широко використовуваним через простоту реалізації та інтуїтивну зрозумілість. Основною проблемою цього методу є вибір кількості сусідів *k*, що може бути досить проблематичним.

Перевагою застосування дерев рішень є визначення залежностей між представленими рухами та видами асан, які можуть не бути лінійними.

Особливість полягає у тому, що такий метод є інтерпретованим, оскільки структура дерева дає можливість відстежити логіку класифікації еталонів. В той же час, дерева рішень схильні до перенавчання та підвищене використання пам’яті при збільшенні даних.

Випадкові ліси, що є ансамблевим поданням дерев рішень, забезпечують стійкість і точність за рахунок усереднення результатів великої кількості дерев. Хоча такий підхід зменшує ймовірність перенавчання, в той же час сильно збільшується проблема використання пам’яті.

Представлені вище методи є ефективними на етапі класифікації невеликої кількості статичних поз.

Однак, вони матимуть проблеми з класифікацією асан, що передбачають динамічні рухи, оскільки розглянуті вище підходи машинного навчання обмежені при роботі з даними, що включають у собі динамічні рухи.

1.2.2 Методи класифікації фізичних вправ, зокрема вправ йоги, на основі нейронних мереж

Найбільш поширеним у вирішенні прикладних задач є використання нейронних мереж у задачах класифікації візуальної інформації. Особливістю методів цього напряму, яка відрізняє їх від методів машинного навчання, є здатність самостійно визначати ключові ознаки із вхідного потоку даних. Такий підхід робить процес аналізу більш гнучким.

Вибір конкретного виду нейронної мережі зумовлено її специфікацією. Розглянемо такі методи, як згорткові нейронні мережі, рекурентні нейронні мережі, гібридні моделі, які поєднують декілька підходів в один метод, та візуальні трансформери.

1.2.2.1 Згорткові та рекурентні нейронні мережі

Згорткові нейронні мережі (Convolutional Neural Network, CNN) один найбільш поширений інструмент у задачах класифікації щодо використанням візуальних матеріалів, в основному растрових зображень. Він надає можливість витягнути багаторівневі ознаки, що збільшує ефективність класифікації. У контексті розпізнавання поз йоги CNN допомагає ефективно визначити конкретну асану за одним кадром.

Ще одною перевагою згорткових нейронних мереж є формування ієрархії ознак. Такий підхід дозволяє правильно розпізнати об’єкт навіть при зміні, наприклад, освітлення, ракурсу, наближення чи віддалення до камери.

Недоліком цього підходу є те, що CNN аналізують лише просторову структуру і не враховують часову послідовність рухів, що обмежує їх застосування у випадках класифікації динамічних об’єктів.

Рекурентні архітектури дозволяють аналізувати послідовності даних (наприклад, у вигляді відеоданих), враховуючи часову залежність.

Рекурентні моделі Long Short-Term Memory (LSTM) та Gated Recurrent Unit (GRU) застосовуються для класифікації вправ, що мають у собі динамічні елементи. Такі моделі є ефективними при класифікації динамічних асан, а також при відстежуванні переходів між різними асанами, якщо розглядати використання моделей, що розглядаються в цьому підрозділі, у задачах класифікації поз йоги.

Використання рекурентних нейронних мереж дозволяє виявляти помилки у техніці виконання ще під час руху, що відкриває перспективи для створення інтерактивних систем зворотного зв’язку. Недоліком їх вважається велика обчислювальна складність та чутливість до довгих послідовностей. Хоча сучасні модифікації LSTM та GRU вирішують значною мірою цю проблему.

Поєднання згорткових та рекурентних нейронних мереж дозволяє одночасно враховувати просторові характеристики, тобто вид статичної асани, та часові залежності, такі як переходи між асанами чи виконання динамічних вправ йоги.

CNN використовується для обробки кожного кадру відео і виділення ключових ознак об’єкта, як в той же час LSTM аналізує динаміку зміни цих ознак у часі. Такий підхід робить гібридну модель здатною одночасно поєднувати у собі сильні сторони обох підходів.

Подібні моделі показують особливо високі результати при розпізнаванні йога-комплексів або динамічних стилів йоги. Суттєвим недоліком поєднання двох видів моделей є підвищені вимоги до обчислювальних ресурсів та обсягів навчальних даних, що ускладнює їх навчання та подальше використання.

1.2.2.2 Зорові трансформери

Останнім часом трансформери, спочатку створені для обробки природної мови, успішно застосовуються і в комп’ютерному зорі.

Зорові трансформери (Visual Transformer, ViT) розбивають зображення на невеликі фрагменти, які обробляються як послідовності, що дозволяє вловлювати глобальні просторові залежності та закономірності. Підхід демонструє високу точність навіть при класифікації вправ, які включають в собі нестандартні позиції рук та ніг. Також слід відзначити, що трансформери добре працюють з великими наборами даних, швидко адаптуючись до різноманіття стилів виконання.

Ще однією ключовою перевагою методу, що розглядається, є його здатність інтегрувати мультимодальні дані, тобто, наприклад, поєднання відеоданих та даних із сенсорних показників. Це може забезпечити збільшення точності оцінки правильності виконання вправ у схожих із поставленою в цій роботі задачах.

Основним недоліком трансформерів, як і у гібридних моделей, є потреба у великій кількості обчислювальних потужностей та об’ємних навчальних вибірках. Це ускладнює використання цього методу.

Таким чином, існує велика кількість різних підходів класифікації фізичних вправ на зображенні, більшість з яких відносяться до машинного навчання або до методів, що ґрунтуються на нейронних мережах.

До цих пір у деяких задачах використовуються методи машинного навчання як додатковий метод за рахунок простого виконання, однак у більшості випадків використовуються нейронні мережі.

Найпопулярнішим методом класифікації виступають згорткові нейронні мережі, рідше рекурентні нейронні мережі та візуальні трансформери. Останні частіше використовуються для класифікації природньої мови, рідше – об’єкти чи пози людини.

1.3 Аналіз сучасних застосунків класифікації вправ йоги, реалізованих засобами машинного навчання та нейронних мереж

У відкритому доступі не знайдено застосунків або сервісів, які б мали вбудовані методи класифікації вправ йоги, однак, було знайдено декілька застосунків, що покликані розпізнавати класичні фізичні вправи. Обрано три застосунки та сервіси для детального розгляду.

KinesteX AI є мобільним застосунком, доступним на платформах Google Play та AppStore (рис. 1.5 [11]). Положення тіла попередньо калібрується, після чого застосунок підраховує кількість зроблених вправ. Наразі наявно шість комплексів, але застосунок знаходиться у ранньому доступі, тож прогнозується збільшення кількості вправ. Також розробники розробили модуль, який класифікує вправи і який можна вбудувати у власний застосунок.

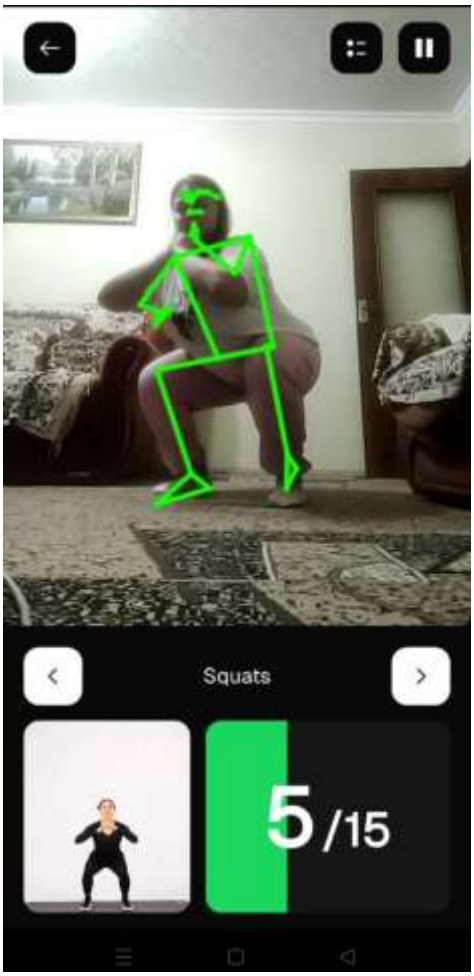


Рисунок 1.5 – Приклад використання застосунку KinesteX AI

До переваг застосунку KinesteX AI можна віднести стабільний захват суглобів людини. Таким чином розпізнавання рухів відбувається правильно навіть при недостатньому освітленні у приміщенні. Крім того, інтерфейс застосунку є зрозумілим та лаконічним.

Недоліками цього застосунку є обмежена кількість вправ та відсутність змоги створювати власні набори вправ. Також KinesteX AI потребує певного положення камери, а при виконанні вправи з іншого ракурсу не може її ідентифікувати.

Kemtai ще не вийшов у широке застосування, однак вже має закрите бета-тестування (рис. 1.6 [12]). Цей застосунок покликаний на класифікацію рухів людини під час реабілітаційного процесу. В основі розроблення сервісу ґрунтується власний закритий фреймворк із закритим кодом.

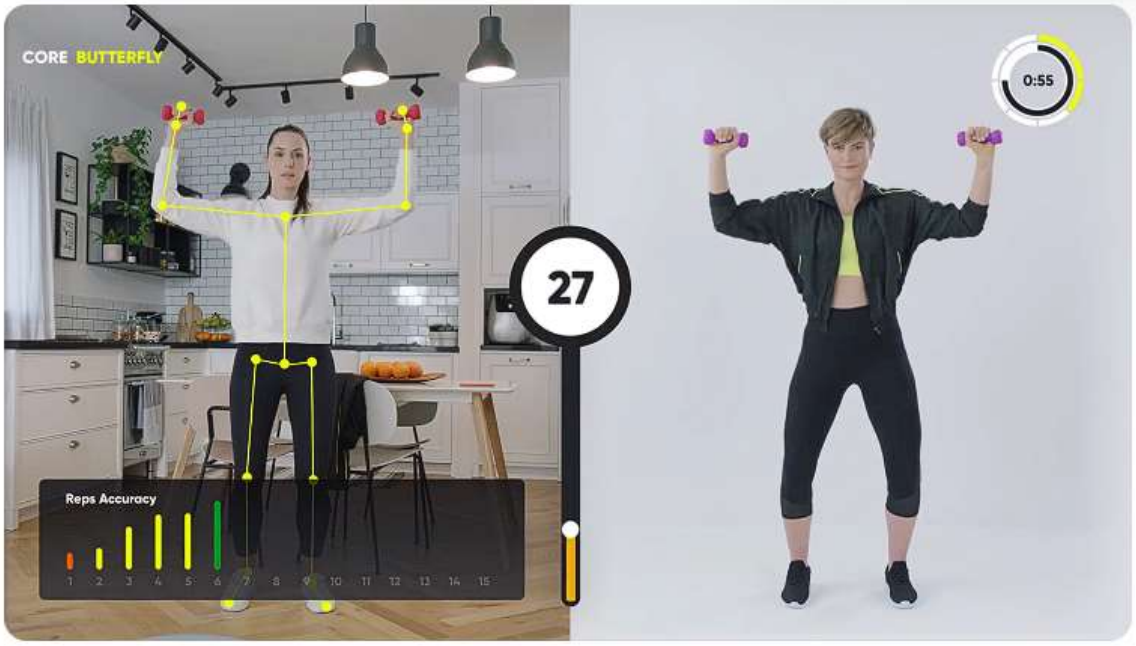


Рисунок 1.6 – Промо-матеріали застосунку Kemtai

Спираючись на промо-матеріали, перевагами застосунку Kemtai є чітке розпізнавання точок суглобів. Також у правій частині інтерфейсу наявне відео правильного виконання фізичної вправи, що знижує ризик травматизму. У лівій нижній частині інтерфейсу можна побачити оцінку виконання, що дає змогу відслідкувати свою правильність виконання вправ.

Головним недоліком Kemtai є відсутність відкритої демонстраційної версії застосунку, що не дає змоги побачити реальної картини. Також цей застосунок використовує закритий вихідний код, що не дозволяє провести досконалий аналіз.

GymAI є ранньою версією вебзастосунку, що класифікує фізичні вправи (рис. 1.7 [13]).

Рисунок 1.7 – Промо-матеріали застосунку GymAI

На відміну від розглянутих вище сервісів, GymAI має відкритий код. Принципом роботи є витягування ознак з відеопотоку за допомогою бібліотеки MediaPipe, а далі класифікується завчасно навченою моделлю штучного інтелекту, що ґрунтується на моделях рекурентних нейронних мереж та нейронних мереж із довгою короткочасною пам’яттю.

До переваг GymAI слід віднести не тільки можливість перегляду вихідного коду застосунку, а й ознайомлення з основними засобами та інструментами, що були використані при розробленні. Сервіс є вебзастосунком, а отже доступний.

Недоліки GymAI полягають у недостатньому проведенні досліджень щодо точності використаної моделі. Застосунок виглядає не як комерційний продукт, а дослідницький проєкт. Графічний інтерфейс GymAI є не дуже доступним для нового користувача.

Отже, застосунків, що класифікують фізичні вправи, наразі не так багато, але те, що багато з тих, що були знайдені, перебувають зараз у ранньому доступі чи у стані тестування, свідчить про те, що напрям є досить новим та в той же час перспективним.

1.4 Аналіз літературних джерел щодо апробації результатів застосування існуючих методів класифікації фізичних вправ, реалізованих засобами машинного навчання та нейронних мереж, зокрема вправ йоги

У дослідженні [14] виявлення пози йоги досягається за допомогою YOLOv3, тоді як оцінка пози йоги та вилучення ключових точок виконуються за допомогою моделі Movenet. Алгоритми корекції позування аналізують ключові моменти, щоб забезпечити реальне положення суглобів. Система була протестована з використанням спеціального набору даних поз йоги, досягаючи значної точності розпізнавання.

У дослідницькій статті [15] пропонується підхід до самонавчання виду йоги Сур’я Намаскар з автоматизованою класифікацією поз за допомогою класифікаторів машинного та глибокого навчання. На першому етапі зображення пози попередньо обробляються для вилучення частини пози за допомогою техніки обмежувальної коробки. На другому етапі отримують різні характеристики на основі форми, такі як прискорені стійкі ознаки (Speeded Up Robust Features, SURF) і гістограми орієнтованих градієнтів (Histogram of Oriented Gradients, HOG). Третій етап включає класифікацію поз Сур’я Намаскар за допомогою класифікаторів штучних нейронних мереж і *k*-найближчих сусідів. Результати роботи підтверджуються використанням архітектури глибокого навчання, такої як згорткова нейронна мережа.

Дослідження [16] встановлює класифікацію поз йоги в наборі даних йоги-16, використовуючи поєднання підходів глибокого навчання та машинного навчання.

Попередньо навчені архітектури CNN VGG16 і VGG19 збирають глибинні ознаки із зображень поз йоги. Потім зібрані ознаки об’єднуються та вводяться в класифікатори для тренування та оцінки результату класифікації. Щоб класифікувати позу із зібраного набору даних йоги-16, запропонована модель включає логістичну регресію, опорні векторні машини, випадковий ліс і додаткові класифікатори дерев. Запропонований підхід використовував набір даних йога-16, що містить 16 класів і 6561 зображення. Комбінований підхід із використанням лінійного SVM дає кращі результати в порівнянні з традиційними методами, які показують, що зазначений підхід ефективний для досягнення чудових показників у класифікації пози йоги.

Метою дослідження [17] є вивчення кількох методів класифікації поз йоги. Представлено модель LGDeep, яка поєднує нову залишкову згорткову нейронну мережу з трьома підходами глибокого навчання: Xception, VGGNet і SqueezeNet. Модель LGDeep включає саме такі методи вилучення функцій, як лінійний дискримінантний аналіз (Linear Discriminant Analysis, LDA) та узагальнений дискримінантний аналіз (Generalized Discriminant Analysis, GDA). Експериментальні результати демонструють, що класифікатор LGDeep перевершує усі підходи і досягає дуже високого коефіцієнта точності класифікації.

Стаття [18] поєднує основи машинного навчання зі структурами даних. Разом із використанням алгоритмів глибокого навчання пропонується підхід для ефективного виявлення та розпізнавання різних поз йоги. Обраний набір даних складається із 85 відео з 6 позами йоги, які виконують 15 учасників, де ключові моменти витягуються за допомогою бібліотеки MediaPipe. Комбінація згорткової нейронної мережі і довгої короткочасної пам’яті була використана для розпізнавання пози йоги за допомогою відео, що відстежуються в реальному часі, як модель глибокого навчання. Рівень CNN використовується для вилучення функцій із ключових точок, а наступний рівень LSTM розуміє появу послідовності кадрів для прогнозів, які мають бути реалізовані. Далі пози класифікуються як правильні або неправильні.

Якщо правильна поза визначена, то система надасть користувачеві відповідний зворотний зв’язок через текст/мовлення.

Дослідження [19] використовує підхід до трансферного навчання на згорткових нейронних мережах, щоб передбачити поставу йоги, зроблену людиною в реальному часі. Набір даних із 85 різних типів поз йоги створюється шляхом вебскрапінгу та захоплення зображень, що є більшим у порівнянні з існуючими роботами. Навчальний, валідаційний та тестовий набори отримані шляхом поділу отриманого набору даних у відношенні 70:10:20 відповідно. Попередньо навчені моделі, такі як EfficientNet-B0, Xception, ResNet-50 і MobileNet були обрані на основі їхніх минулих результатів і пройшли навчання на створеному наборі даних йоги. Експериментальні результати показують, що модель Xception з використанням трансферного навчання дала найкращі результати, а також посіла друге місце за часом виконання.

У роботі [20] запропоновано модель, засновану на глибокому навчанні, щоб визначити п’ять різних поз йоги на основі порівняно меншої кількості даних. Запропонована модель CNN містить особливості архітектури Xception. Також звичайна згортка була замінена на глибинно розділювальну згортку, що суттєво оптимізувало обчислювальні потужності. Запропонована архітектура витягує просторові та глибинні характеристики із зображення окремо та розглядає їх для подальшого розрахунку в класифікації. Здійснено порівняння продуктивності розробленої моделі із деякими найсучаснішими моделями класифікації зображень – ResNet, InceptionNet, InceptionResNet, Xception.

У статті [21] за допомогою OpenPose створено модель, яка оцінює певну позу людини. Порівняно результати, отримані моделлю для двовимірних і тривимірних точок зображення, і визначено, чи дійсно додавання більшої кількості функцій до набору даних підвищує точність моделі чи ні. Крім того, запропоновано просту модель нейронної мережі, яка ефективно аналізує вхідне зображення та передає, чи правильна поза, виконана на зображенні.

Ключовий етап у роботі [22] включає вилучення ключових точок, виконане за допомогою MoveNet, вдосконаленої моделі оцінки постави.

Щоб досягти цього, алгоритми глибокого навчання, такі як згорткові нейронні мережі, щільні нейронні мережі (DNN) і багатошарові перцептрони (MLP), розгортаються для ефективної класифікації позицій йоги та фізичних вправ. Кульмінація цього починання проявляється у вигляді інтерактивного вебзастосунку.

У дослідженні [23] пропонується новий підхід до класифікації пози вправ у реальному часі на основі згорткової нейронної мережі (CNN) у системі ансамблевого навчання. Використовуючи MediaPipe, запропонована система витягує спільні координати та кути людського тіла, які CNN використовує для вивчення складних моделей різних вправ. Крім того, цей новий підхід покращує продуктивність класифікації, поєднуючи прогнози з кількох кадрів зображення за допомогою методу ансамблевого навчання.

Для перевірки використовується базовий набір даних Fitness від Infinity AI. Експерименти демонструють високу точність у класифікації таких вправ, як підняття рук, присідання та вивага над головою. Запропонована модель продемонструвала свою здатність ефективно класифікувати пози вправ у режимі реального часу, досягнувши високих показників точності.

Проаналізувавши джерела [14–23] можна прийти до висновку, що найбільш популярними є моделі, що поєднують у собі згорткову нейронну мережу разом із класифікаторами, представленими як методами машинного навчання [15, 16], так і іншими видами нейронних мереж. Аналіз дав розуміння, які з моделей є поширеними для використання і модифікації в цій предметній області, а також виявив, які з моделей доцільно взяти для дослідження.

Слід зазначити, що у проаналізованих дослідженнях популярністю при вирішенні задач користується модель Xception. Це згорткова нейронна мережа, яка, на відміну від інших CNN, використовує глибинно розділювальну згортку (що було розкрито у дослідженні [20]), через що стає менш вибагливою до обчислювальних потужностей. Також у проаналізованих дослідженнях непогано показують себе такі згорткові нейронні мережі, як ResNet-50 та EffiicientNet-B4.

1.5 Постановка задачі дослідження

Таким чином, класифікація фізичних вправ, зокрема, поз йоги, є актуальним завданням. Прийнято рішення щодо розроблення програмного застосунку класифікації вправ йоги на зображенні чотирма методами (машинного навчання та згорткових нейронних мереж), а саме із застосуванням методу SVM, а також таких моделей, як ResNet-50, Xception та EffiicientNet-B4.

Об’єктом дослідження є методи класифікації, реалізовані засобами машинного навчання та нейронних мереж.

Метою дослідження є порівняння методів класифікації зображень, основаних на машинному навчанні та нейронних мережах, шляхом розробки застосунку, що класифікує зображення основних поз йоги.

Для досягнення мети необхідно вирішити такі завдання:

– проаналізувати літературні джерела щодо апробації методів класифікації об’єктів на зображеннях;

– проаналізувати сучасні методи класифікації об’єктів на зображеннях;

– проаналізувати обрані методи розпізнавання зображень, основані на згорткових нейронних мережах;

– деталізувати структуру кожної з вибраних моделей для класифікації об’єктів на зображеннях;

– сформувати покроковий план реалізації застосунку класифікування зображень асан йоги;

– дослідити модифіковану згорткову нейронну мережу щодо виконання поставленого завдання;

– розробити програмний застосунок, що надасть змогу класифікувати зображення виконання людиною пози йоги кожним із вибраних методів;

– проаналізувати отримані результати класифікації для кожного класу;

– виявити перспективи для подальшого розвитку.

2 ОСОБЛИВОСТІ ВИБРАНИХ МЕТОДІВ КЛАСИФІКАЦІЇ ВПРАВ ЙОГИ, РЕАЛІЗОВАНИХ ЗАСОБАМИ МАШИННОГО НАВЧАННЯ ТА НЕЙРОННИХ МЕРЕЖ

2.1 Метод HOG-SVM

Метод опорних векторів є класичним представником методів машинного навчання. Він часто використовується у задачах регресії та класифікації цифрових даних, а також підтримує імплементацію у подальшому нових даних, таких як класи та підкласи. Спираючись на це, його обрано як метод, що потребує дослідження.

Основний принцип його роботи полягає у пошуку таких гіперплощин у просторі даних, які будуть подані у вигляді рівняння (2.1), щоб між точками з різних класів було утворено якнайширшу область їх розділення. У результаті більшість даних одного класу повинна знаходитися в межах поділеної області.

$$w^T x + b = 0, \tag{2.1}$$

де w^T – ваги, отримані після SVM;

b – зсув площини.

Однак, ключова проблема метода опорних векторів у контексті класифікації зображень виконання поз йоги полягає у тому, що цей метод працює з числовими даними, поданими векторами, і не передбачає роботу з графічними вхідними даними напряму. У такому випадку є доцільним додатково використати екстрактори ознак. Прийнято рішення використати такі методи, як гістограма орієнтованих градієнтів (Histogram of Oriented Gradients, HOG), а також метод головних компонент (Principal Component Analysis, PCA).

Гістограми орієнтованих градієнтів покликані для екстракції ознак зображення. Перш за все, до вхідних даних застосовується чорно-білий фільтр.

Далі обчислюється градієнт шляхом обчислення похідних інтенсивності по вертикальному та горизонтальному напрямках. Таким чином вхідне двовимірне зображення трансформується у вектор, що представляє напрямки градієнтів інтенсивності.

Додаткове використання методу головних компонент зумовлене тим, що вихідні дані гістограм орієнтованих градієнтів подають шляхом центрування та побудови коваріаційної матриці, які застосовуються до вхідного вектора.

Новий вектор сортується за дисперсією та наполовину скорочується. Таким чином скорочується кількість витягнутих ознак зі збереженням інформативності.

Задля видалення шумів та збільшення відсотка точності є сенс додатково оброблювати зображення шляхом видалення фону та накладання шуму.

Отже, фінальна структура методу (рис. 2.1) є така:

– попереднє оброблення зображень;
– гістограма орієнтованих градієнтів;
– метод головних компонент;
– метод опорних векторів.

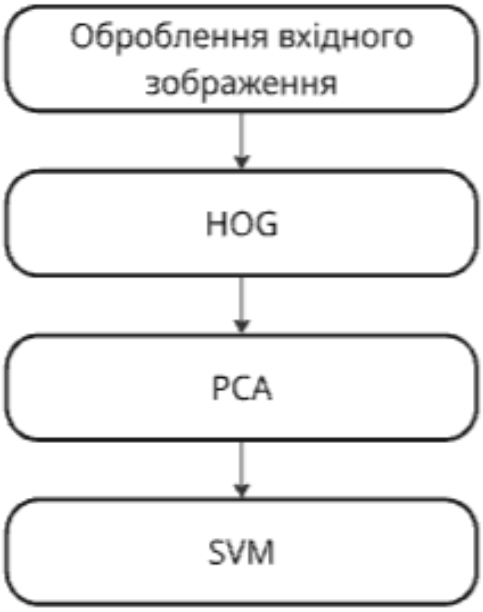


Рисунок 2.1 – Алгоритм методу HOG-SVM

2.2 Архітектура ResNet

ResNet – це архітектура глибокої згорткової нейронної мережі, яка була розроблена командою розробників Microsoft Research спеціально для роботи із зображеннями. Особливістю даної архітектури та її головною перевагою є використання залишкового навчання (residual learning) замість навчання на кожному кроці.

Такий підхід дозволяє здійснювати навчання мереж з великою кількістю шарів, при цьому уникати появи затухання градієнтів – ситуації, коли на великій кількості епох швидкість навчання нейронної мережі сповільнюється або зупиняється. Ще однією перевагою архітектури ResNet є висока точність навчання та гнучкість у масштабуванні, яка досягається за допомогою звужених залишкових блоків (bottleneck residual blocks).

Таким чином, модель можна розширювати та модифікувати методом додавання нових шарів. На основі цього також було побудовано такі моделі, наприклад, DenseNet, Wide ResNet, ResNeXt.

Серед найбільш фундаментальних недоліків цієї архітектури є висока обчислювальна складність. Оскільки моделі даної архітектури зазвичай досить глибокі, виникає проблема в обчисленні великої кількості параметрів. Через це вони є досить повільними, що ускладнює їх використання у системах реального часу на кшталт мобільних застосунків.

Крім того, процес навчання таких моделей з цих причин також потребує значної кількості часу для отримання високої точності. Також під час навчання може виникнути проблема, коли передавання залишкової інформації може сповільняти процес навчання, оскільки нові шари будуть вносити лише незначні зміни, спираючись на дані з попередніх епох.

Це також може дуже сповільнити процес навчання, а також призвести до обмеження ефективності нейронної мережі.

2.2.1 Модель ResNet-50

Архітектура ResNet-50, яка розглядатиметься далі, має 50 шарів, розподілених на декілька рівнів, включаючи останній, що представлений повнозв’язним шаром.

Усього ResNet-50 має 6 рівнів, з яких:

– Conv1 є початковим рівнем (рис. 2.2);

– Conv2_x має 3 звужені залишкові блоки;

– Conv3_x має 4 звужені залишкові блоки;

– Conv4_x має 6 звужені залишкові блоків;

– Conv5_x має 3 звужені залишкові блоки;

– шостий рівень є поєднанням функцій глобального усереднення (avgpool, Global Average Pooling), повнозв’язаного шару (fc, Fully Connected) та Softmax-функції.

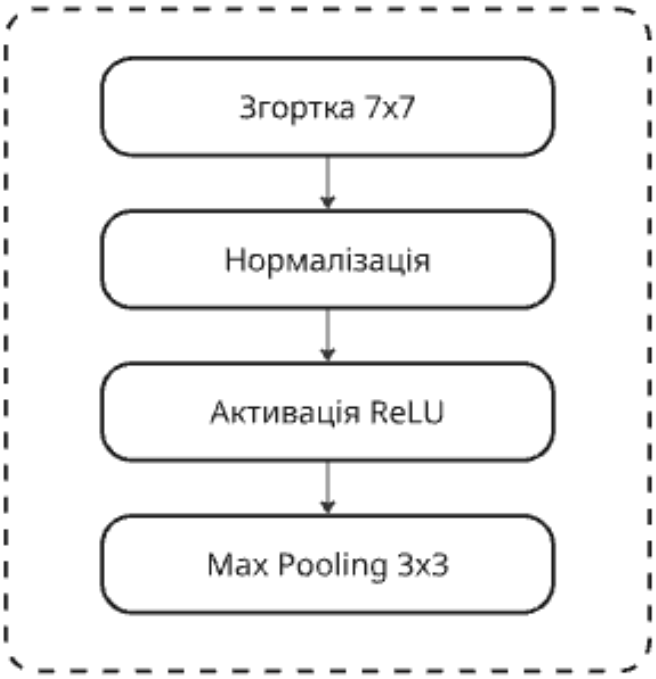


Рисунок 2.2 – Структура рівня Conv1

Перший рівень (рис. 2.2) моделі є початковим. Він виступає етапом підготовки вхідних даних для подальшого його оброблення.

Спочатку із зображення за допомогою згортки 7×7 відбувається витягування базових ознак, за рахунок чого зменшується розмір вдвічі. Таким чином одразу виявляються краї об’єктів та переходи кольору. Після цього до отриманого результату застосовуються функції нормалізації задля стабілізації виходу застосованої згортки та функції активації ReLU для додавання нелінійності, у результаті досягається гнучкість моделі класифікації.

Заключним етапом є застосування Max Pooling-функції. Вона виконує згортку 3×3, при цьому витягуючи максимальне число з області. Таким чином зменшується кількість даних, яку потрібно буде обробити на наступних рівнях та підсвічуються найбільш контрастні зони.

Структура рівней Conv2_x, Conv3_x , Conv4_x та Conv5_x подана на рисунку 2.3.

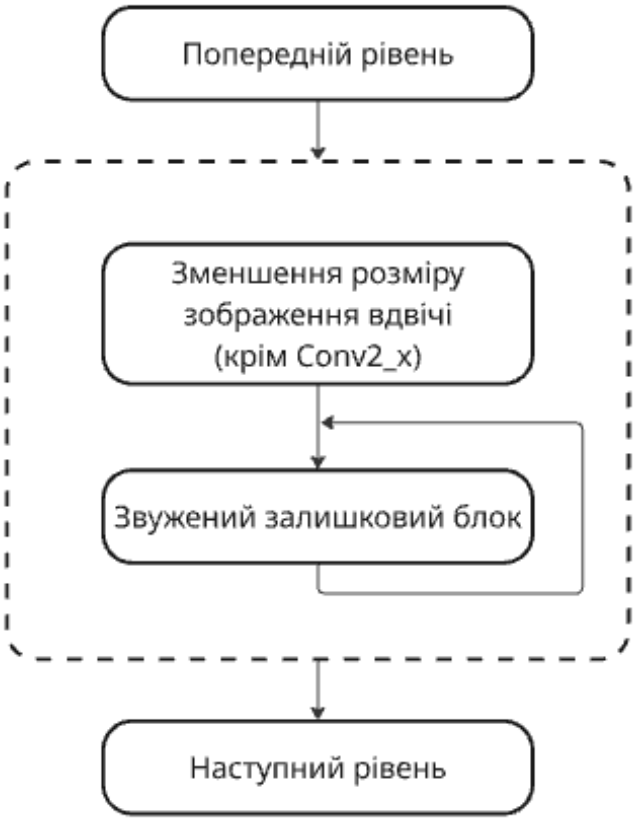


Рисунок 2.3 – Структура рівней Conv2_x, Conv3_x , Conv4_x та Conv5_x

Структура подальших рівнів (рис. 2.3) крім останнього, відрізняється лише кількістю звужених залишкових блоків, що застосовуються.

Також на рівні Conv2_x не відбувається попереднє зменшення розміру зображення, оскільки до вхідного зображення вже було застосовано згортку на заключному етапі рівня Conv1. Таким чином, рівні складаються лише із попередньої згортки зображення вдвічі та подальшого застосування до нього звужених залишкових блоків. Після блоків немає додаткових функцій нормалізації та активації, оскільки вони вже входять у структуру блоку.

Останній рівень ResNet-50 подано на рисунку 2.4.

Рисунок 2.4 – Останній рівень ResNet-50

Останній рівень (рис. 2.4) є вихідним. До карти ознак, що отримано з попередніх рівнів, застосовується глобальне усереднення, яке перетворює її у вектор шляхом взяття середнього по кожному каналу. Після цього отримані вектори передаються у повнозв'язний шар, де об'єднуються у єдиний вектор. Заключним етапом є застосування Softmax-функції.

2.2.2 Звужені залишкові блоки

Ключовою особливістю моделі ResNet-50 та архітектури ResNet в цілому є використання звужених залишкових блоків як основного засобу класифікації.

Шляхом додавання вхідного тензору до результату, отриманого після згортки, результати стають більш поступовими та сприяють стабілізації результатів при навчанні. Схему звуженого залишкового блоку подано на рисунку 2.5.

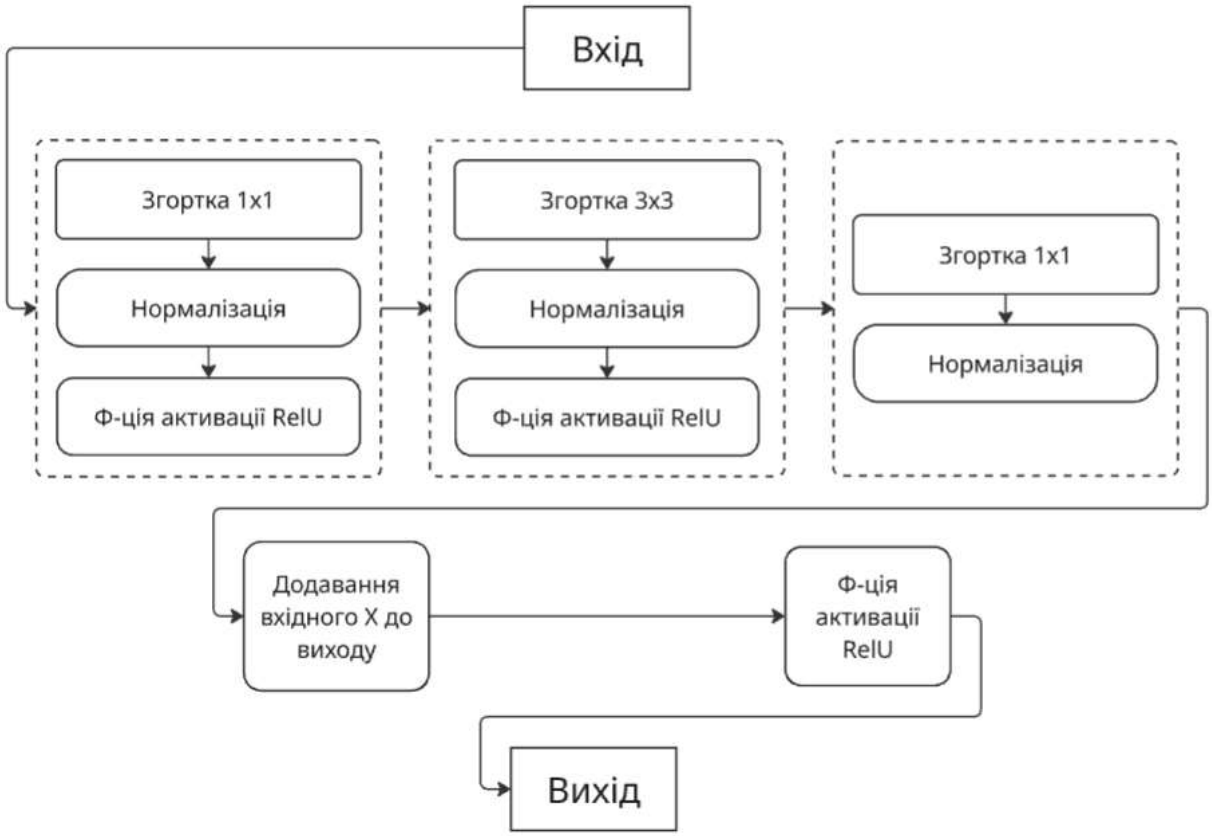


Рисунок 2.5 – Схема звуженого залишкового блоку

Звужений залишковий блок (рис. 2.5) можна подати у такому вигляді:

$$y = F(x, \{W_1, W_2, W_3\}) + x, \tag{2.2}$$

де x – вхід блоку;

$F(x, \{W_1, W_2, W_3\})$ – функція залишкового блоку (складається з 3 згорткових шарів: 1×1, 3×3, 1×1);

y – вихід блоку.

У свою чергу функцію залишкового блоку можна подати формулою:

$$F(x) = W_3 * BN(\sigma(W_2 * BN(\sigma(W_1 \cdot BN(x))))), \quad (2.3)$$

де W_1 – ядро першого шару 1×1;

W_2 – ядро другого шару 3×3;

W_3 – ядро третього шару 1×1;

$\cdot$ – операція згортки;

BN – функція нормалізації;

σ – функція активації ReLU.

Слід зазначити, що вихідні дані функції залишкового блоку є співставними з вхідними, оскільки в подальшому відбувається додавання залишкового тензору до отриманого.

2.3 Модель Xception

Згорткова нейронна мережа Xception (eXtreme Inception) була розроблена у 2017 році на основі вдосконаленої архітектури Inception v3.

Ключова особливість Xception полягає у використанні глибоко сепарабельних блоків. Ідея полягає у виконанні просторової згортки для кожного з каналів, після чого виконується згортка 1×1, яка поєднує результати сепарабельної згортки.

Основною перевагою є зменшення кількості обчислювальних параметрів внаслідок використання сепарабельної згортки, але в той же час зі збереженням точності моделі та, у деяких випадках, навіть її збільшення.

Недоліками Xception є висока потреба в потужних обчислювальних ресурсах при навчанні на великих об’ємах графічних даних, а на малих може бути схильна до перенавчання.

Також дана модель має високу чутливість до гіперпараметрів, що вимагає часу на більш ретельне налаштування перед проведенням навчання.

З рештою, Xception – це швидка модель, яка зберігає високу точність, однак має труднощі з налаштуванням.

2.3.1 Структура моделі Xception

Архітектуру згорткової нейронної мережі Xception можна поділити так:

- вхідний рівень;
- середній рівень;
- вихідний рівень.

Вхідний рівень (рис. 2.6) складається з двох згорток 3×3, а також залишкового блоку, схожого за принципом роботи на звужені залишкові блоки, які представлені в архітектурі ResNet.

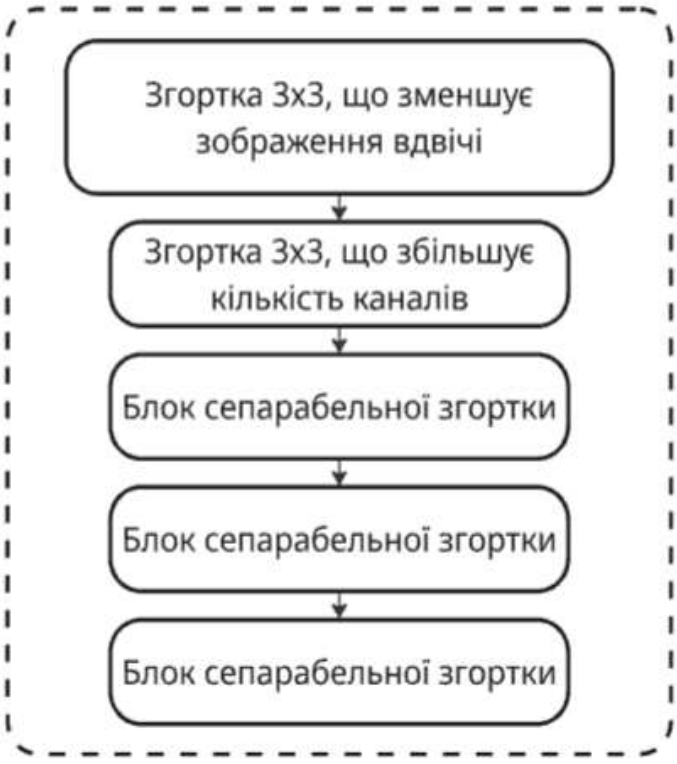


Рисунок 2.6 – Структура вхідного рівня згорткової нейронної мережі Xception

За допомогою першої згортки 3×3 вхідне зображення зменшується вдвічі, що призводить до збільшення швидкості навчання моделі внаслідок зменшення кількості даних для оброблення, а також це дозволяє ще на початку навчання виділити зони інтересу на лінії переходу. Друга згортка 3×3 не змінює розміри зображення, але збільшує кількість каналів. Слідом за цим йде залишковий блок, який містить у собі три блоки сепарабельної згортки. У таких блоках по кожному з каналів окремо відбувається згортка, яка після оброблення кожного об’єднується в єдиний вихідний вектор.

Середній рівень є основною частиною моделі Xception. Він складається з восьми залишкових блоків, подібних до того, що знаходиться в кінці вхідного блоку. На цьому рівні вхідне зображення не змінює розмір, не змінюється кількість каналів, але суттєво збільшується глибина витягання ознак.

Вихідний рівень (рис. 2.7) складається із залишкового блоку, двох блоків сепарабельної згортки, а також функцій глобального усереднення avgpool задля перетворення вихідного тензору на вектор, повнозв’язного шару та Softmax-функції.

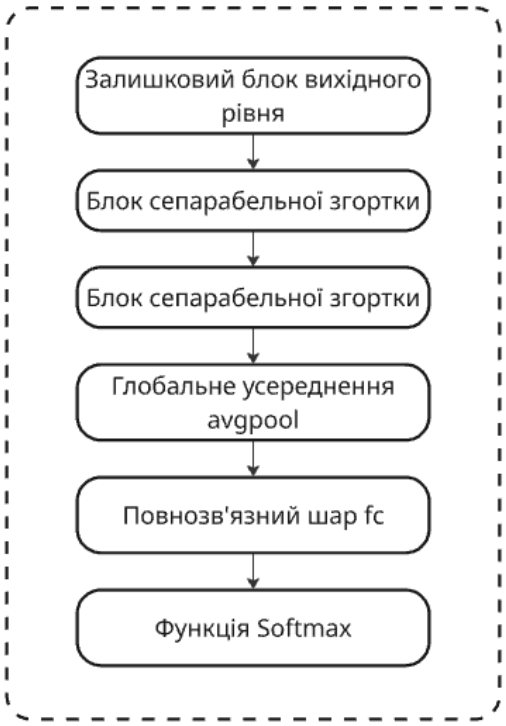


Рисунок 2.7 – Структура вихідного рівня

На відміну від залишкових блоків вхідного та середнього рівнів, залишковий блок вихідного рівня (рис. 2.8) містить лише два блоки сепарабельної згортки, другий із них збільшує кількість вихідних каналів.

Рисунок 2.8 – Структура залишкового блоку вихідного рівня

Завершальним етапом є використання Max Pooling-функції. Таким чином, вхідне зображення знову змінює свій розмір вдвічі.

Після залишкового блоку вихідного рівня виконуються два блоки сепарабельної згортки. У результаті згортання кількість каналів збільшується і виходить 2048 каналів. Далі виконуються завершальні функції, що обробляють вихідні результати.

2.3.2 Блок сепарабельної згортки

Особливістю моделі Xception є використання блоки сепарабельної згортки, які використовують depthwise-pointwise блоки. Основна їх ідея полягає у виконанні згортки із застосуванням фільтрів для кожного каналу окремо та подальшого об’єднання отриманих результатів у вихідний тензор.

Причому всередині такого блоку кількість каналів може як збільшуватися, так і зберігатися. Слід також зазначити, що цей фільтр не єдиний для усіх каналів, їх кількість відповідає кількості вихідних каналів.

Структура depthwise-pointwise блоку (рис. 2.9) має чотири шари, такі як функція активації ReLU, depthwise-згортка, pointwise-згортка та функція нормалізації. Оскільки функції активації та нормалізації є класичними для усіх моделей, що розглядаються, було зосереджено увагу на специфічних блоках, а саме на depthwise- та pointwise-згортках.

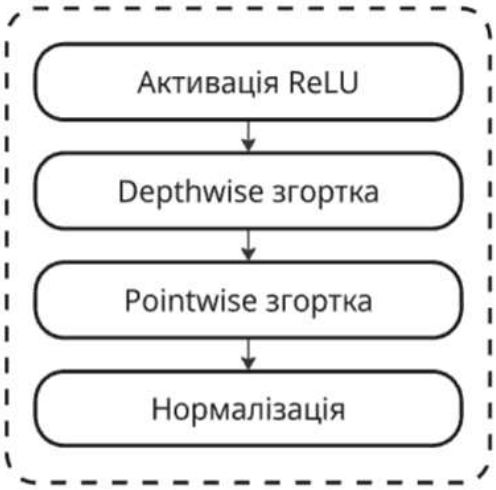


Рисунок 2.9 – Структура depthwise-pointwise блоку

Шар depthwise-згортки застосовує відповідний фільтр до вхідного тензору каналу та виконує згортку, при цьому даний шар не змінює розміри вхідних даних та не збільшує кількість каналів.

Підсумовуючи, структуру шару depthwise-згортки можна подати так:

$$y_{i,j,k} = \sum_{m,n} x_{i+m,j+n,k} K_{m,n,k}, \tag{2.4}$$

де $y_{i,j,k}$ – вихід тензора depthwise-згортки для *k*-го вихідного каналу;

$x_{i+m,j+n,k}$ – вхідний тензор;

$K_{m,n,k}$ – вага фільтра на позиції *m*, *n* для *k*-го каналу;

i, j – координати позиції на тензорі;

m, n – індекси координат фільтра;

k – номер каналу.

Головна ідея pointwise-згортки полягає в об’єднанні інформації, отриманої з попереднього шару, шляхом її перемішування. Далі застосовуються ваги, які регулюють ступінь значущості певного каналу. Саме на цьому етапі кількість каналів може змінюватися. Структуру етапу pointwise-згортки можна подати формулою:

$$z_{i,j,c} = \sum_k y_{i,j,k} K_{1,1,k,c}, \tag{2.5}$$

де $z_{i,j,c}$ – вихід тензора pointwise-згортки для *c*-го вихідного каналу;

$y_{i,j,k}$ – вихід тензора depthwise-згортки для *k*-го вхідного каналу;

$K_{1,1,k,c}$ – вага фільтра на позиції 1,1 для *k*-го каналу;

i, j – координати позиції на тензорі;

k – номер вхідного каналу;

c – номер вихідного каналу.

Блоки сепарабельної згортки (рис. 2.10) складаються з трьох послідовних depthwise-pointwise блоків, додавання залишкового тензору.

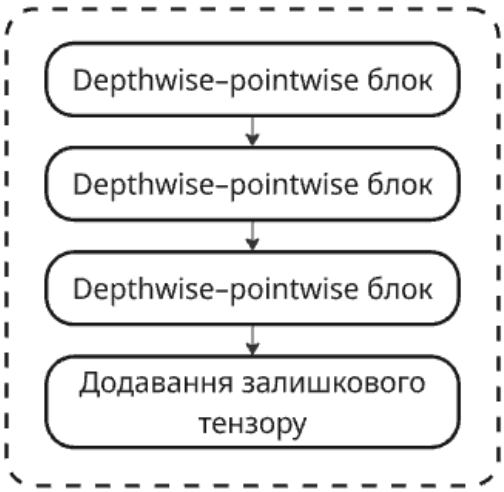


Рисунок 2.10 – Структура блоку сепарабельної згортки

2.4 Архітектура EfficientNet

EfficientNet є видом згорткових нейронних мереж, розроблених у 2019 році шляхом автоматичного пошуку архітектури.

Основною особливістю такої архітектури є використання мобільних інвертованих звужених блоків (MBConv, Mobile Inverted Bottleneck Convolution), що були запозичені з архітектури MobileNetV2. Такі блоки збільшують кількість каналів не в кінці, а на початку, перед згорткою. Причому принцип роботи згортки є подібним до блоків сепарабельної згортки, що використовуються в моделі Xception.

EfficientNet представлена лінійкою моделей B0–B7.

B0 була початковою моделлю, від якої відбувався подальший розвиток архітектури. Полегшеними вважаються EfficientNet B0–B3, оскільки вважаються швидкими порівняно з B4–B7, однак значно поступаються точністю.

Моделі B4 та B5 є такими, що зберігають баланс між швидкістю та точністю. Версії B6 та B7 використовують для високоточної класифікації на потужних GPU.

До переваг можна віднести високу ефективність та збалансоване масштабування моделей, що відносяться до цієї архітектури, за рахунок системного підходу, при якому одночасно змінюється кількість шарів, каналів, а також роздільна здатність зображення. Також до переваг EfficientNet можна віднести широкий спектр моделей, що дозволяє використовувати таку архітектуру при створенні застосунків, націлених на різні платформи.

Серед недоліків є проблема громіздкості такої архітектури, що робить її складною в реалізації, а також ускладнює модифікування та калібрування її параметрів. Глибокі версії EfficientNet B5–B7 дуже повільні, що обмежує їх використання.

2.4.1 Модель EfficientNet-B4

Як модель, що буде представляти лінійку EfficientNet, обрано версію B4. Ця модель досить точна, на відміну від B0–B3, і, у той же час, зберігає високу швидкість порівняно з моделями B5–B7.

Структура моделей B1–B7 розраховується на основі структури B0 та гармонічних коефіцієнтів масштабування, початкові значення яких:

– коефіцієнт глибини (α) дорівнює 1,2;

– коефіцієнт ширини (β) дорівнює 1,1;

– коефіцієнт роздільної здатності (γ) дорівнює 1,15.

Коефіцієнт глибини відповідає за кількість інвертованих згорткових блоків для 2–7 рівнів, коефіцієнт ширини позначає кількість вихідних каналів для кожного рівня, а коефіцієнт роздільної здатності за початкові розміри вхідного зображення.

Обчислення коефіцієнтів для конкретної моделі відбувається шляхом піднесення до ступеня, якому відповідає номер моделі (наприклад, для моделі B5 треба підносити коефіцієнти до ступеня 5). Далі значення моделі B0 множиться на отримані коефіцієнти, після чого округлюються до цілих значень. Для кількості каналів значення додатково округлюється до найближчого значення кратного 8.

Початкові значення для моделі B0:

– кількість блоків для рівнів відповідає [1, 2, 2, 3, 3, 4, 1];

– кількість каналів [16, 24, 40, 80, 112, 192, 320];

– роздільна здатність дорівнює 224.

Для моделі EfficientNet-B4 α = 2,0736, β = 1,4641 та γ = 1,7490.

Тоді кількість інвертованих згорткових блоків по рівнях для моделі дорівнює [2, 4, 4, 6, 6, 8, 2], кількість вихідних каналів за рівнями становить [24, 32, 56, 120, 160, 280, 472]. Роздільна здатність вхідного зображення за обчисленнями відповідає 392. Для заключного рівня кількість каналів дорівнює 1872.

Виходячи з обчислень, проведених вище, усього EfficientNet-B4 має дев’ять рівнів, таких як:

– підготовчий рівень;
– другий рівень включає 2 інвертованих згорткових блоки;
– третій рівень включає 4 інвертованих згорткових блоків;
– четвертий рівень включає 4 інвертованих згорткових блоків;
– п’ятий рівень включає 6 інвертованих згорткових блоків;
– шостий рівень включає 6 інвертованих згорткових блоків;
– сьомий рівень включає 8 інвертованих згорткових блоків;
– восьмий рівень включає 2 інвертованих згорткових блоки;
– заключний рівень.

Підготовчий рівень (рис. 2.11) не має блоків інвертованої згортки. Натомість на цьому рівні відбувається зменшення вхідного зображення з 392×392 до 196×196, а також застосовуються функції нормалізації та активації.

Рисунок 2.11 – Структура підготовчого рівня

На відміну від розглянутих вище моделей, у цій, замість ReLU, використовується її модифікована версія, SiLU.

Другий рівень складається з двох інвертованих згорткових блоків. На цьому рівні не відбувається зменшення розмірів зображення, однак зменшується кількість вихідних каналів.

З третього по восьмий рівні мають подібну структуру (рис. 2.12). Вони складаються з відповідної для кожного з них кількості інвертованих блоків, причому перший з них зменшує розміри зображення вдвічі, а всі інші мають наприкінці додавання попереднього тензору, як це було в архітектурі ResNet, та не змінюють розміри вхідних даних.

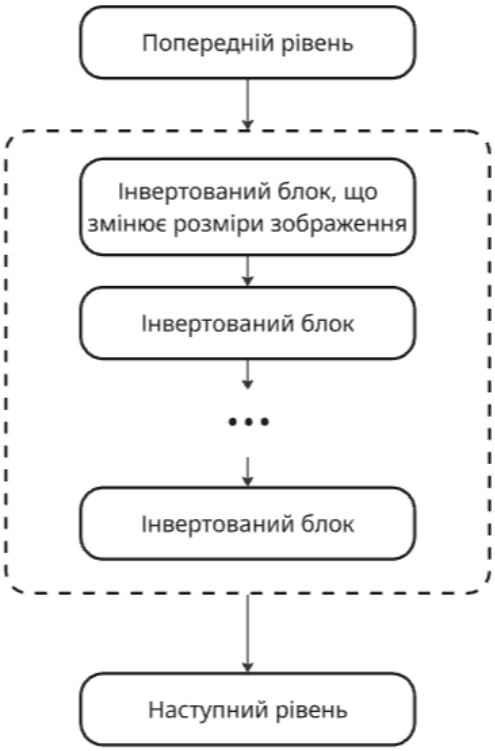


Рисунок 2.12 – Структура 3–8 рівнів

Заключний рівень (рис. 2.13) моделі EfficientNet-B4, як і в моделей, що розглядалися вище, складається із згортки 11, що збільшує кількість каналів до 1872, функції глобального усереднення, функції відсіювання Dropout, повнозв’язного шару, а також SoftMax-функції.

На відміну від розглянутих вище моделей, EfficientNet-B4 має функцію Dropout, яка відповідає за відсіювання отриманого результуючого вектора. Принципом її роботи є випадкове вимикання елементів вихідного вектора, отриманого з функції глобального усереднення, причому для кожного з елементів є своя ймовірність вимикання. Застосування Dropout обумовлене тим, що ця функція зменшує ризик перенавчання.

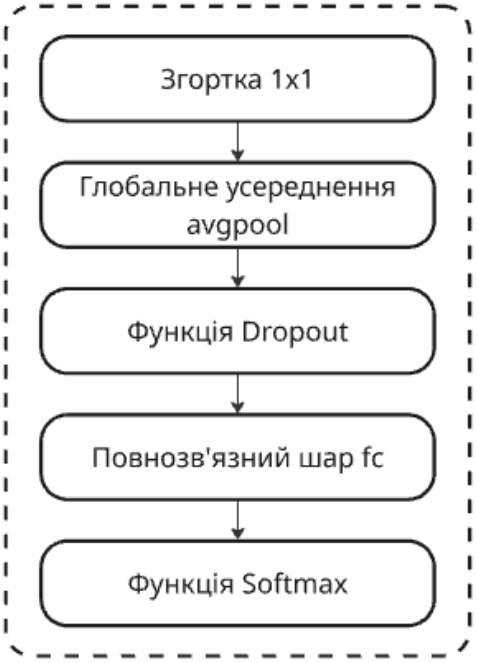


Рисунок 2.13 – Структура заключного рівня

2.4.2 Інвертований згортковий блок

Особливістю архітектури EfficientNet є використання інвертованих згорткових блоків (рис. 2.14). Вони містять в собі деякі з концепцій, розглянених вище моделей, такі як додавання залишкового вектору ознак та згортки для кожного каналу окремо, однак, новим є інвертованість порядку зміни кількості вихідних каналів: у традиційних моделях кількість каналів збільшується в кінці блоку, а тут це відбувається на початку. У кінці блоку міститься згортка, що відповідає за зменшення кількості каналів до числа, зазначеного для кожного з рівнів.

Першим шаром блоку виступає згортка 1×1. На цьому етапі відбувається збільшення кількості каналів у певну кількість разів. Наприклад, для моделі B4 кількість каналів становить 6.

Другим шаром виступає depthwise-згортка. Вона подібна до тієї, що знаходиться в блоках моделі Xception, однак, у цьому випадку немає наступної за нею pointwise-згортки, яка перемішувала отримані результати.

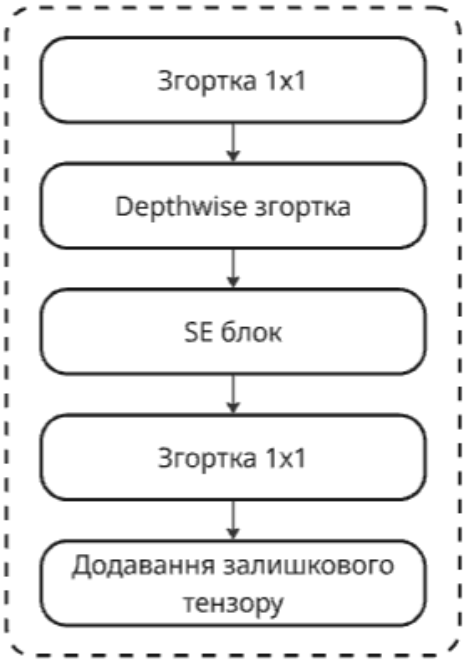


Рисунок 2.14 – Структура інвертованого згорткового блоку

Далі йде Squeeze-and-Excitation (SE) блок (рис. 2.15), у якому відбувається послідовне стиснення та розширення тензору ознак.

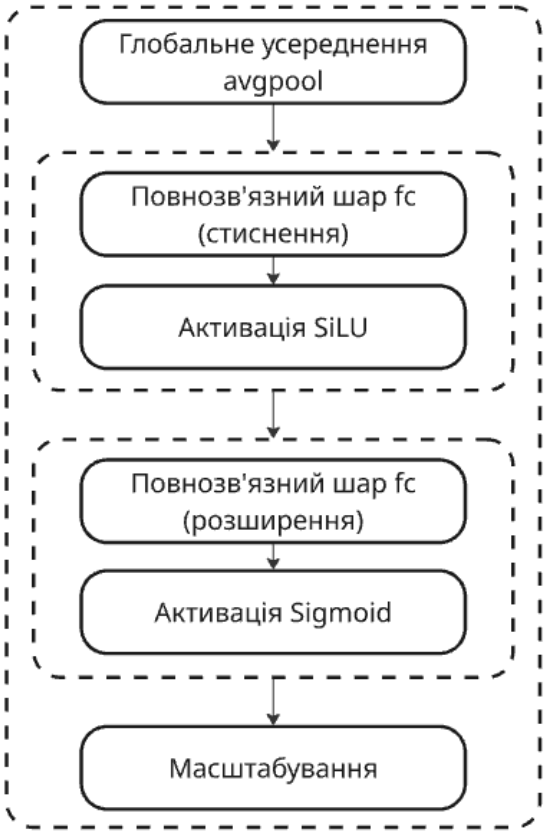


Рисунок 2.15 – Структура SE блоку

Принципом його роботи є проходження тензору через два послідовних повнозв’язних шари, які спочатку перетворюють його на вектор, а після цього розширюють до розмірів вхідних даних. Отриманий вектор є матрицею ступеня значущості кожного каналу. Саме цей блок є основною особливістю лінійки архітектур EfficientNet.

Після проходження блоку depthwise-згортки тензору останнім шаром виступає масштабування отриманих відносно вектору ваг, отриманих з SE блоку.

2.5 Структура модифікованої моделі, основаної на EfficientNeet-B4

Спираючись на моделі, що були розглянуті вище, встановлено, що кожна з моделей має свою власну особливість.

Метод HOG-SVM містить попереднє оброблення вхідних зображень, що може збільшити відсоток точності.

Модель ResNet-50 мітить звужені залишкові згорткові блоки, основною властивістю яких є додавання попереднього тензору до результату, отриманого з послідовних згорток. Цей принцип корисний у використанні та необхідний для пришвидшення навчання і подолання проблеми перенавчання, а, отже, може бути використаним при формуванні структури власної модифікованої моделі.

Модель Xception наслідує особливість архітектури ResNet щодо додавання залишкового тензору. Цей прийом використовується у особливості моделі Xception – сепарабельних згорткових блоках, перевагами використання яких виступають збільшення гнучкості та пришвидшення навчання. Такі блоки швидші порівняно з інвертованими блоками EfficientNet, тож можуть бути використані як полегшення модифікованої моделі.

Лінійка моделей EfficientNet використовує особливості архітектур ResNet та Xception у будові спеціальних інвертованих згорткових блоках. У них використовуються блоки depthwise-згортки, запозичені з блоків моделі Xception, однак pointwise-згортка не використовується.

Після огляду перелічених вище моделей виявлено, що архітектура EfficientNet є найбільш розвиненою порівняно з першими двома моделями, а, отже, прийнято рішення побудувати модифіковану модель на основі структури EfficientNet-B4.

Поставлено завдання полегшити обрану класичну модель та при цьому зберегти точність. Досягти такого результату планується шляхом введення блоків з моделей ResNet та Xception, які швидші порівняно з блоками, що застосовуються у EfficientNet.

Оскільки за основу модифікованої моделі береться EfficientNet-B4, то зберігається її структура, аналогічні початковий та заключний рівні. Основною ідеєю модифікації виступає заміщення інвертованих блоків, з яких складається класична архітектура, на блоки з моделей Xception та ResNet-50 з метою збільшення швидкості порівняно з класичною EfficientNet-B4 без втрати точності при класифікації.

Сепарабельні блоки моделі Xception швидші за інвертовані згорткові блоки, однак, вони більш ефективно показують себе на пізніх рівнях, тому їх слід додавати на вищих рівнях.

Звужені згорткові блоки з моделі ResNet-50 знизять ризик перенавчання за допомогою додавання до виходу із згорткових шарів залишкового тензору, тож є сенс додати їх до структури вищих рівнів з метою збільшення стабільності модифікованої моделі.

Блоки моделей Xception та ResNet-50 слід використовувати тільки на вищих рівнях через те, що початкові розміри вхідних даних дорівнюють 224, але для ефективної роботи моделі EfficientNet-B4 та, відповідно, її модифікованих версій, це значення дорівнює 380.

Структура модифікованої моделі показана на рисунку 2.16.

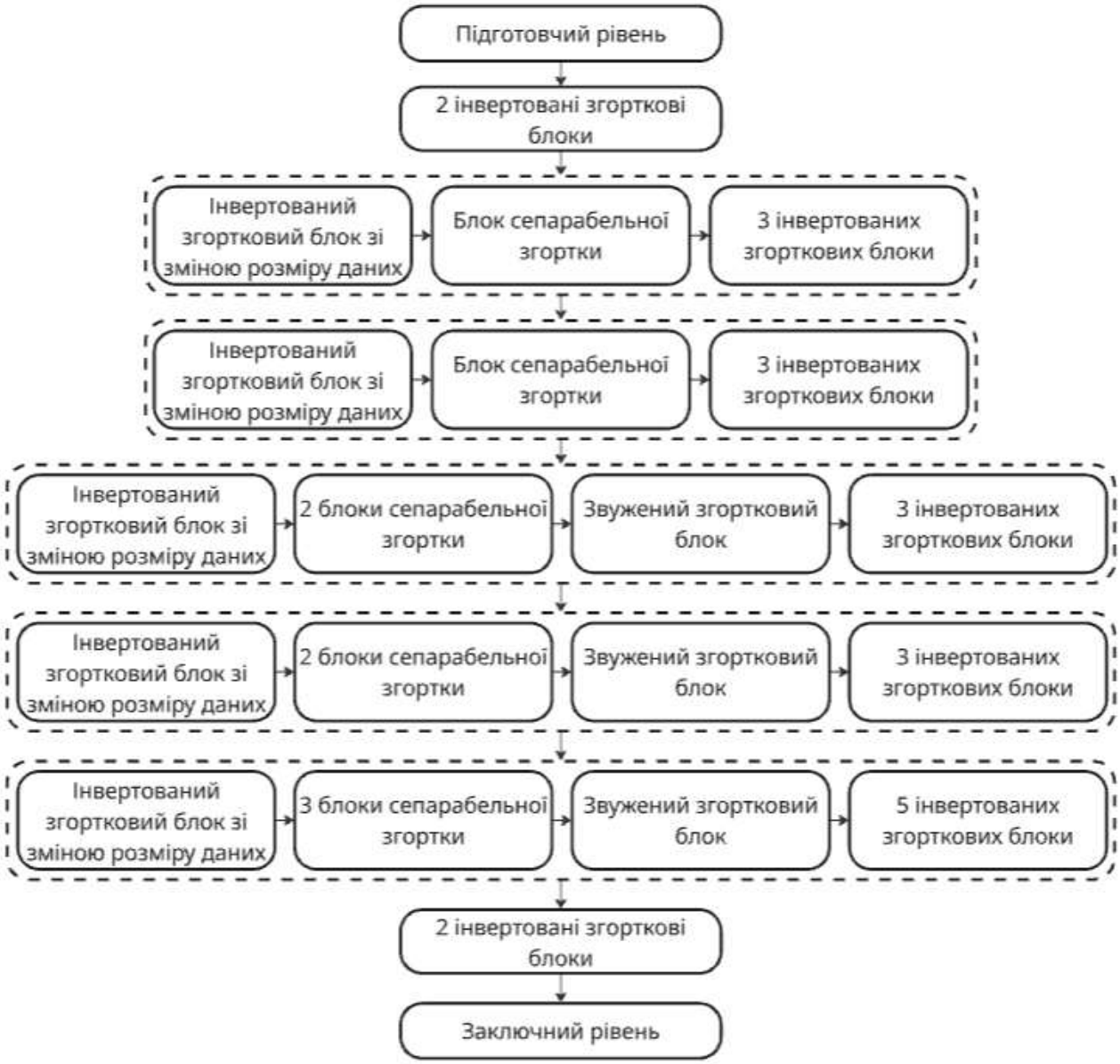


Рисунок 2.16 – Структура модифікованої моделі

Модифікована версія моделі EfficientNet-B4 має таку структуру:

а) підготовчий рівень;

б) другий рівень включає інвертованих згорткових 2 блоки;

в) третій та четвертий рівні включають 5 блоків:

1) інвертований згортковий блок;

2) сепарабельний блок;

3) інвертованих згорткових 3 блоки;

г) п’ятий та шостий рівні включають 7 блоків:

1) інвертований згортковий блок;

2) сепарабельних 2 блоки;

3) звужений блок;

4) інвертованих згорткових 3 блоки;

д) сьомий рівень включає 10 блоків:

1) інвертований згортковий блок;

2) сепарабельних 3 блоки;

3) звужений блок;

4) інвертованих згорткових 5 блоків;

е) восьмий рівень включає інвертованих згорткових 2 блоки;

є) заключний рівень.

3 ДОСЛІДЖЕННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ВПРАВ ЙОГИ, РЕАЛІЗОВАНИХ ЗАСОБАМИ МАШИННОГО НАВЧАННЯ ТА НЕЙРОННИХ МЕРЕЖ

3.1 Вибір інструментальних засобів для реалізації вибраних методів

Задля програмної побудови архітектур, розглянутих вище згорткових нейронних мереж, необхідно обрати найбільш ефективні та зручні інструменти.

Зручною для побудови та роботи з нейронними мережами та великими масивами даних вважається високорівнева мова програмування Python, оскільки вона має низку суттєвих переваг порівняно з іншими популярними об’єктно-орієнтованими мовами програмування, такими як, наприклад, C++ або C#.

Python має інтуїтивно зрозумілий та простий синтаксис, що дає змогу простіше читати вихідний код та звільнитися від надмірної, у випадку роботи з нейронними мережами, громіздкості класової структури, притаманної іншим мовам програмування.

Також Python має велику кількість простих у використанні засобів, націлених на роботу зі складними обчисленнями, моделями машинного навчання та нейронними мережами, а також зі складними багатовимірними масивами різних типів даних, що дуже важливо у роботі з нейронними мережами.

Обрана мова програмування має декілька бібліотек та велику кількість вбудованих у них модулів для візуалізації отриманих в ході виконання поставленої задачі результатів класифікації у вигляді зображень, таблиць та графіків, а також інструменти для розроблення кросплатформних застосунків. Це дає змогу більш наглядно подати дані, що були отримані в ході проведення дослідження.

Як основну бібліотеку для побудови та роботи зі згортковими нейронними мережами обрано PyTorch.

Фреймворк PyTorch має відкритий вихідний код, розроблений за сприяння організації Meta AI, та вважається одним із найбільш популярних інструментів у цій предметній області. Він має низку вбудованих методів для створення різних типів мереж, їх гнучкого налаштування та модифікування, а також швидкого завантаження вже побудованих класичних нейронних мереж.

Для візуалізації отриманих результатів обрано інструментарій PyQt5. Цей фреймворк використовується для побудови простих графічних інтерфейсів та кросплатформних застосунків. Він має велику кількість методів для налаштування віджетів, таких як таблиці, зображення, графіки, та інтерактивних елементів керування, таких як кнопки, панелі або списки, а, отже, ідеально підходить для нескладної візуалізації завдання, чого потребує завдання, що вирішується у цій роботі.

Основними критеріями вибору середовища розробки для виконання поставленого завдання, а саме побудови згорткових нейронних мереж, можна виділити такі:

– можливість швидкого підключення та оновлення в разі потреби пакетів, пов’язаних з машинним навчанням та нейронними мережами, а саме PyTorch та пов’язані з цією бібліотекою модулі на кшталт torchvision або torch.nn;

– можливість встановлення методу cuda задля перемикання в разі потреби під час навчання обчислювальних ресурсів з центрального процесора на графічний;

– можливість налагодження інструментів середовища, оскільки це дозволить ефективно аналізувати можливі виникнення помилок, відслідковувати значення параметрів, що налаштовуються при проведенні тренувань, контролювати навчання моделі та оптимізувати окремі частини програмного коду;

– можливість зручного використання засобів та пакетів, призначених для графічної візуалізації отриманих в ході роботи даних, а саме бібліотеки PyQt5;

– доступність графічного інтерфейсу.

Як основне середовище розроблення обрано середовище PyCharm, оскільки воно задовольняє зазначені вище пункти. PyCharm надає зручний інструментарій для роботи з Python, має гнучку систему автодоповнення, розширені можливості відлагодження вихідного коду, вбудовану підтримку роботи з віртуальними середовищами, а також добре інтегрується з бібліотеками та модулями машинного навчання та засобами візуалізації.

Оскільки навчання нейронних мереж потребує великої кількості обчислювальних ресурсів, як доступне середовище для навчання, що задовольняє зазначені вище вимоги, обрано безкоштовний онлайн сервіс Google Colab. Цей сервіс надає доступ до віддаленої обчислювальної машини, ноутбука, з графічною пам’яттю, що зменшує кількість витраченого на навчання часу та робить більш доступним роботу з глибокими нейронними мережами. У ньому можна обрати різні моделі апаратного прискорення для підбору для конкретного завдання.

3.2 Етапи програмної реалізації обраних методу і моделей класифікації вправ йоги

Під час першого етапу програмної реалізації визначено шляхи пошуку та створено вибірку із зображеннями виконання різних вправ йоги.

Як вибірку, за якою відбуватиметься навчання, обрано відкритий набір даних Yoga-16 (рис. 3.1), який складається з 16 класів та 1500 зображень формату .jpg. Вибірку розділено на три підвибірки: тренувальну, тестову та валідаційну у співвідношенні 7:1:2.

Вибірку даних було збільшено до 3000 зображень виконання поз йоги (до 200 зображень для кожного класу), взятих із відкритих джерел, для отримання кращих результатів навчання згорткових нейронних мереж та зменшення ризиків перенавчання моделі на недостатньо великому наборі даних.

Рисунок 3.1 – Зображення класів, за якими відбуватиметься навчання

Другий етап передбачає побудову обраного методу HOG-SVM та моделей ResNet-50, Xception та EfficientNet-B4, а також модифіковану згорткову нейронну мережу.

Третім етапом виступає навчання досліджених методів машинного навчання та згорткових нейронних мереж за однакових умов для подальшого порівняння. Таким чином можна буде об’єктивно їх оцінити.

Четвертий етап передбачає побудову графічного інтерфейсу у вигляді кросплатформного застосунку за допомогою інструментарію фреймворка PyQt5 для візуалізації результатів класифікування зображень побудованими моделями.

П’ятим етапом є аналіз отриманих в ході класифікації результатів та визначення перспектив для подальшої роботи, можливості до покращення результатів у майбутньому.

3.2.1 Побудова методу HOG-SVM

Модуль SVM має такі 6 методів:

– remove_background;
– preprocess_image;
– augment_image;
– extract_hog_features;
– load_dataset_with_augmentation;
– train_model.

Застосування методу remove_background видаляє фон зображення задля збільшення відсотка точності класифікації, оскільки без фону гістограмі орієнтованих градієнтів буде легше визначити границі об’єкта, що зробить вектор ознак більш інформативним.

У методі preprocess_image застосовується чорно-білий фільтр до вхідних зображень, змінюються його розміри.

Лістинг 3.1 Метод preprocess_image:

```
def preprocess_image(img: np.ndarray) -> np.ndarray:
    if img.ndim == 2:
        img_gray = img
    elif img.shape[-1] == 4:
        img_gray = color.rgb2gray(img[:, :, :3])
    else:
        img_gray = color.rgb2gray(img)
    h, w = img_gray.shape
    crop_h, crop_w = int(0.15 * h), int(0.15 * w)
    img_gray = img_gray[crop_h:h - crop_h, crop_w:w - crop_w]
    img_gray = exposure.equalize_hist(img_gray)
    img_gray = transform.resize(img_gray, IMAGE_SIZE, anti_aliasing=True)
    return img_gray
```

У методі augment_image відбувається накладання шумів, застосовується зсув та поворот вхідного зображення з метою розбавлення та урізноманітнення збірки даних.

Методи load_dataset_with_augmentation та extract_hog_features застосовують гістограми орієнтованих градієнтів до аугментованих даних, трансформуючи їх у вектор ознак.

У методі train_model до отриманого з попередніх методів вектора додається метод головних компонент, потім до оптимізованого вектора ознак застосовується метод опорних векторів та відбувається навчання метода на зображеннях.

3.2.2 Побудова моделі ResNet-50

За допомогою інструментарію бібліотеки PyTorch побудовано архітектуру ResNet-50. Архітектура представлена у вигляді класу ResNet50, який містить у собі методи, що реалізовують побудову структури, з якої складається модель, а також класу Bottleneck, який описує структуру звуженого згорткового блоку.

Клас Bottleneck забезпечує побудову звуженого згорткового блоку. Він складається з конструктора, де ініціалізуються основні методи, а також метод forward, який пропускає через структуру вхідні дані та повертає вже оброблені. Метод також присутній у класах інших моделей, а також у класах інших блоків.

Лістинг 3.2 Метод forward класу Bottleneck:

```
def forward(self, x: torch.Tensor) -> torch.Tensor:
    identity = x
    out = self.conv1(x)
    out = self.bn1(out)
    out = self.relu(out)
```

```
    out = self.conv2(out)
    out = self.bn2(out)
    out = self.relu(out)
    out = self.conv3(out)
    out = self.bn3(out)
    if self.downsample is not None:
        identity = self.downsample(x)
    out += identity
    out = self.relu(out)
    return out
```

Клас ResNet50 також містить конструктор та метод forward. Крім того, даний клас має додатковий метод для його побудови make_layer, за допомогою якого будується структура для кожного рівня на основі об’єктів класу Bottleneck.

Лістинг 3.3 Метод forward класу ResNet50:

```
def forward(self, x: torch.Tensor) -> torch.Tensor:
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.maxpool(x)
    x = self.layer1(x)
    x = self.layer2(x)
    x = self.layer3(x)
    x = self.layer4(x)
    x = self.avgpool(x)
    x = torch.flatten(x, 1)
    x = self.fc(x)
    return x
```

3.2.3 Побудова моделі Xception

Архітектура моделі Xception представлена трьома класами:

– Xception, що описує структуру моделі;

– XceptionBlock, що має структуру згорткового сепарабельного блоку;

– SeparableConv2d, що містить шари depthwise та pointwise згорток.

Клас Xception містить такі методи, як конструктор (де ініціалізуються три рівні – початковий, середній, представлений стеком сепарабельних блоків, а також заключний) та ініціалізації ваг, що використовуються у виконанні depthwise та pointwise згортках, а також forward.

Лістинг 3.4 Метод forward класу Xception:

```
def forward(self, x: torch.Tensor) -> torch.Tensor:
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.conv2(x)
    x = self.bn2(x)
    x = self.relu(x)
    x = self.block1(x)
    x = self.block2(x)
    x = self.block3(x)
    x = self.middle_flow(x)
    x = self.block_exit(x)
    x = self.global_pool(x)
    x = torch.flatten(x, 1)
    x = self.fc(x)
    return x
```

Клас XceptionBlock ініціює структуру блоку сепарабельної згортки. Він складається з таких методів, як конструктор та функція forward. У конструкторі ініціалізуються основні шари, проводяться перевірки на тип блоку (блоки, що використовуються на середньому рівні, і блоки останнього рівня відрізняються за структурою).

Лістинг 3.5 Метод forward класу XceptionBlock:

```
def forward(self, x: torch.Tensor) -> torch.Tensor:
    out = self.block(x)
    skip = x if self.skip is None else self.skip(x)
    out += skip
    return out
```

Лістинг 3.6 Метод forward класу SeparableConv2d:

```
def forward(self, x: torch.Tensor) -> torch.Tensor:
    x = self.depthwise(x)
    x = self.pointwise(x)
    return x
```

3.2.4 Побудова моделі EfficientNet-B4

Побудова моделі EfficientNet-B4 складається з чотирьох класів: Swish, SqueezeExcite, MBConv та EfficientNet.

Клас Swish містить обчислення функції активації SiLU, вдосконаленої версії ReLU, яка використовується у нейронних мережах нового покоління.

Лістинг 3.7 Метод forward класу Swish:

```
def forward(self, x):
    return x * torch.sigmoid(x)
```

Клас SqueezeExcite представляє побудову SE блоку. Він містить два повнозв’язні шари – один для звуження, а другий для розширення тензору, для першого використовується функція активації, а для другого – сигмоїдна функція.

Лістинг 3.8 Метод forward класу SqueezeExcite:

```
def forward(self, x):
    s = x.mean((2, 3), keepdim=True)
    s = F.silu(self.fc1(s))
    s = torch.sigmoid(self.fc2(s))
    return x * s
```

Клас MBConv призначений для побудови інвертованих згорткових блоків, з яких і складається архітектура EfficientNet. До блоку, побудованого з класу SqueezeExcite, додатково додаються функції нормалізації та активації, а також залишковий тензор (у випадку, коли блок не змінює розміри вхідних даних та кількість каналів).

Лістинг 3.9 Метод forward класу MBConv:

```
def forward(self, x):
    out = self.block(x)
    if self.use_residual:
        out = out + x
    return out
```

Клас EfficientNet розраховує в конструкторі параметри під специфічну модель EfficientNet-B4, відштовхуючись від структури моделі початкової версії лінійки EfficientNet, а саме B0, а також наповнює список з рівнями певної кількості інвертованих згорткових блоків відповідно до того рівня, що описується.

Метод forward збирає вхідний рівень, список з 2 по 8 рівень, а також останній рівень. Крім того, тут додатково використовується dropout-функція, на відміну від інших побудованих моделей.

Лістинг 3.10 Метод forward класу EfficientNet:

```
def forward(self, x):
    x = self.stem(x)
    x = self.blocks(x)
    x = self.head(x)
    x = self.pool(x)
    x = torch.flatten(x, 1)
    x = self.dropout(x)
    x = self.fc(x)
    return x
```

3.2.5 Побудова модифікованої моделі HybridEfficientNet

Програмна архітектура модифікованої згорткової нейронної мережі HybridEfficientNet складається з таких класів, як:

- Bottleneck з архітектури ResNet-50;
- SeparableConv2d та XceptionBlock з моделі Xception;
- SqueezeExcite та MBConv з архітектури EfficientNet-B4;
- HybridEfficientNet.

Класи, що представляють блоки архітектур, побудованих вище, не відрізняються від тих, що використовувалися у класичних архітектурах. Єдиною відмінністю є зміна функції активації з ReLU на SiLU, оскільки вона вважається більш гнучкою, а також, на відміну від ReLU, може приймати від'ємні значення.

Клас HybridEfficientNet складається з конструктора та методу forward. Конструктор займається ініціалізацією основних рівнів модифікованої моделі, а саме вхідного та вихідного рівнів, а також основної частини моделі (2–8 рівні), яка, на відміну від специфічного подання структури середніх рівнів у класі EfficientNet, задається як список із кортежами, де перше значення містить рівень, а друге – вид блоку, що застосовується, і його вхідні параметри.

Лістинг 3.11 Ініціалізація 2–8 рівнів моделі HybridEfficientNet:

```
levels = [
    [("mb", 2, 32, 24, 1)],
    [("mb", 1, 24, 32, 2),
("sep", 1, 32, 32, 1),
("mb", 2, 32, 32, 1)],
    [("mb", 1, 32, 48, 2),
("sep", 1, 48, 48, 1),
("mb", 2, 48, 48, 1)],
    [("mb", 1, 48, 64, 2),
("sep", 2, 64, 64, 1),
("bottleneck", 1, 64, 64, 1),
("mb", 3, 64, 64, 1)],
    [("mb", 1, 64, 96, 2), ("sep", 2, 96, 96, 1),
("bottleneck", 1, 96, 96, 1),
 ("mb", 3, 96, 96, 1)],
    [("mb", 1, 96, 160, 2),
("sep", 3, 160, 160, 1),
```

```
("bottleneck", 1, 160, 160, 1),
("mb", 6, 160, 160, 1)],
   [("mb", 2, 160, 256, 2)]
]
```

Таким чином, метод forward складається із вхідного (stem) та вихідного (head) рівнів моделі EfficientNet-B4, між якими знаходиться основна частина, яка конструюється методом комбінування основних блоків із досліджених моделей ResNet-50, Xception та EfficientNet-B4, а саме звужених, сепарабельних та інвертованих згорткових блоків.

Лістинг 3.12 Метод forward класу HybridEfficientNet:

```
def forward(self, x):
   x = self.stem(x)
   x = self.features(x)
   x = self.head(x)
   x = torch.flatten(x, 1)
   x = torch.dropout(x)
   x = self.classifier(x)
   return x
```

3.2.6 Навчання методу машинного навчання HOG-SVM

Навчання методу відбувається у методі train_model. На вхід отримується вектор ознак, який оптимізується та передається до методу опорних векторів. На відміну від нейронних мереж, SVM проходить по даних лише один раз та функція втрат для нього не розраховується.

Досягнуто 74,7% точності на тренувальній вибірці зображень для метода HOG-SVM (рис. 3.2).

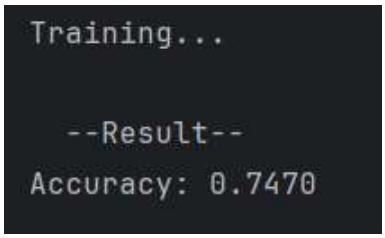


Рисунок 3.2 – Результати навчання методу HOG-SVM

3.2.7 Навчання побудованих моделей згорткових мереж

Під час проведеного дослідження побудовано універсальний скрипт для тренування кожної з моделей. За допомогою нього відбувалося тренування моделей із виведенням значень точності та втрат для кожної епохи.

Набір даних попередньо обробляється задля поліпшення різноманіття класифікації та зменшення ризику виникнення проблеми перенавчання нейронної мережі, що особливо небезпечно на невеликих об’ємах даних. Розмір зображення змінюється, застосовується відзеркалення, а також повертання.

Лістинг 3.13 Попереднє оброблення вхідних даних:

```
train_transform = transforms.Compose([
    transforms.Resize((image_size, image_size)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor(),
])
```

Для моделей нейронних мереж визначено рівні умови навчання:

- застосовано один набір даних;
- обрано оптимізатор Adam;
- розмір батчів 16;
- темп навчання (learning rate) 0,0003;
- 20 епох навчання.

Для моделей користовувався різний розмір зображення: для ResNet-50 та Xception *image_size* = 224, а для EfficientNet-B4 та HybridEfficientNet – 380. Така зміна зумовлена специфікою входу для моделей лінійки EfficientNet.

Під час навчання модель зберігається за умови досягнення кращого результату відносно попередній епох, що дозволяє в кінці навчання отримати найкращу версію моделі. У свою чергу, такий підхід надає можливість подальшого використання найкращої конфігурації нейронної мережі для тестування або практичного застосування.

Лістинг 3.14 Налаштування методу train_model:

```
train_model(
    model=model,
    train_dir="yoga16/train",
    val_dir="yoga16/valid",
    save_path="model_name.pth",
    batch_size=16,
    lr=0.0003,
    num_epochs=20,
    image_size=380
)
```

Процес навчання здійснено у середовищі Google Colab. Набір даних розміщено у хмарне сховище Google Disk, з якого він завантажуватиметься до colab-ноутбуку із вихідним кодом. Для навчання моделі інтегровано клас моделі, що навчається, а також універсальний скрипт для тренування моделі.

Для зменшення тривалості навчання віддано перевагу графічним процесорам. Найбільш ефективним серед представлених у розглянутому середовищі вважається A100, тож саме його застосовано під час тренування. Під’єднано режим cuda з модуля torchvision бібліотеки PyTorch для перемикання режиму навчання з центрального процесора на графічний.

Проведено тренування моделі ResNet-50 (рис. 3.3), за результатами проходження 20 епох отримано модель з 88,19% точністю на валідаційній вибірці, втрати склали 0,47.

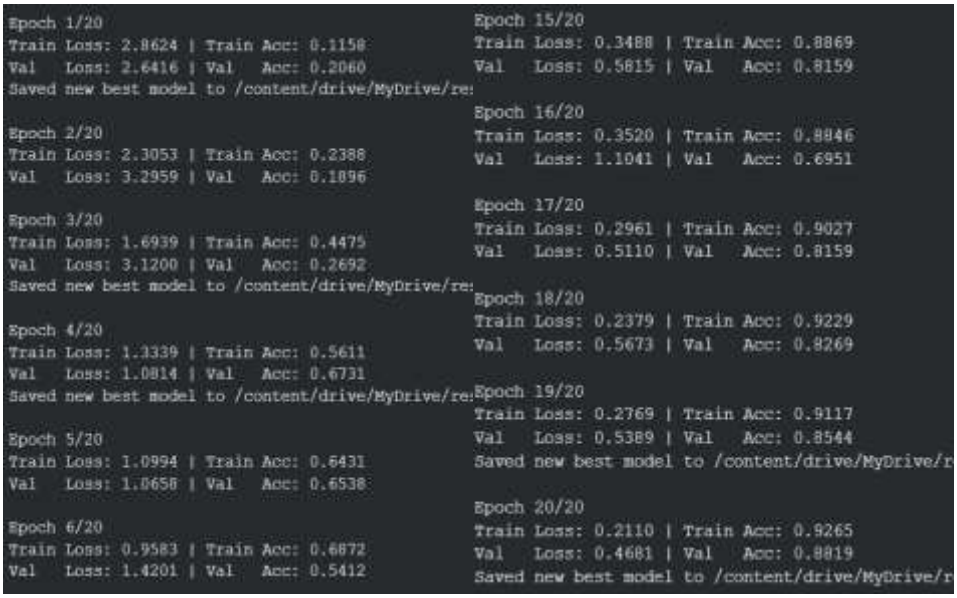


Рисунок 3.3 – Результати тренування ResNet-50

Проведено навчання моделі Xception (рис. 3.4). У результаті проходження всіх епох отримано результат у 91,21% точності на валідаційній вибірці, втрати склали 0,32. Модель Xception показала кращі результати відносно ResNet-50.

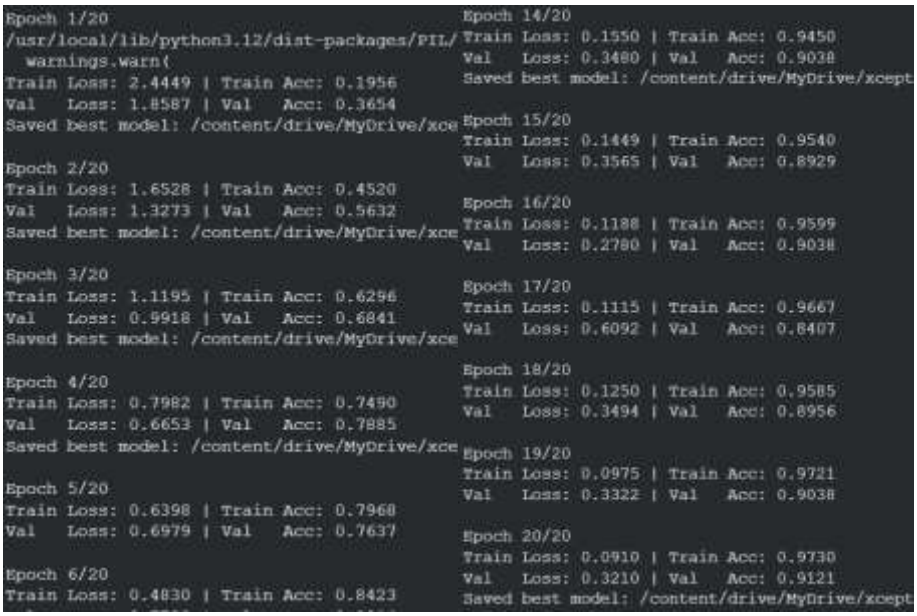


Рисунок 3.4 – Результати тренування Xception

Після проведення навчання перших двох моделей при *image_size* = 224, дане значення було збільшено до 380 задля навчання моделей EfficientNet та HybridEfficientNet. Потім проведено навчання моделі EfficientNet-B4 (рис. 3.5).

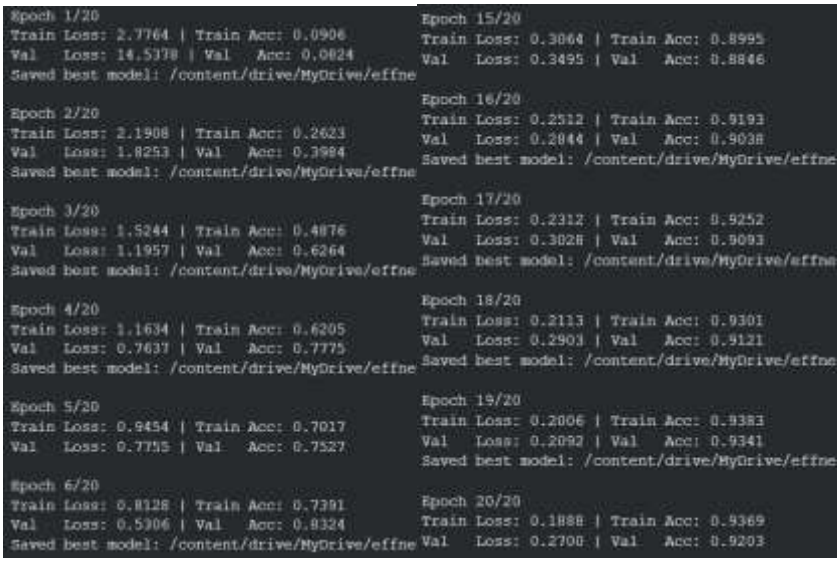


Рисунок 3.5 – Результати тренування EfficientNet-B4

У результаті проходження всіх епох було досягнуто 93,41% точності на валідаційній вибірці, втрати склали 0,21.

Навчання модифікованої моделі HybridEfficientNet показало після проходження всіх епох – 81,87% точність, а втрати становлять 0,62 (рис. 3.6).

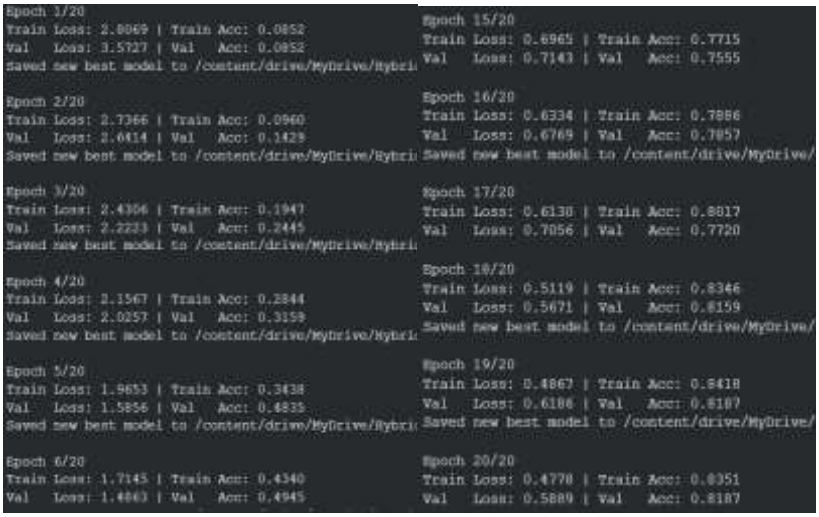


Рисунок 3.6 – Результати навчання модифікованої нейронної мережі HybridEfficientNet

Таким чином, навчено досліджені метод та моделі та отримано словник ваг для застосування у задачах класифікації, а також отримано важливі результати навчання (табл. 3.1).

Таблиця 3.1 – Результати навчання методів та моделей

Модель	**Втрати**	**Точність, %**
HOG-SVM	–	74,70
ResNet-50	0,47	88,19
Xception	0,32	91,21
EfficientNet-B4	0,21	93,41
HybridEfficientNet	0,62	81,78

3.2.8 Створення графічного інтерфейсу

Для візуалізації роботи натренованих згорткових нейронних мереж застосовано модулі фреймворку PyQt5, що має низку зручних та інтуїтивно зрозумілих методів для гнучкого налаштування та оформлення. Розроблено простий кросплатформний інтерфейс, що має декілька об’єктів, таких як:

- кнопка для завантаження зображення;
- контейнер для виведення зображення;
- таблиця з отриманими результатами.

Лістинг 3.15 Генерація вікна та об’єктів:

```
self.setWindowTitle("Yoga pose classifier")
self.setFixedSize(800, 600)
layout = QVBoxLayout()
self.button = QPushButton("Завантажити зображення")
self.button.clicked.connect(self.load_image)
layout.addWidget(self.button)
```

```
self.image_label = QLabel()
self.image_label.setFixedSize(400, 300)
layout.addWidget(self.image_label, alignment=Qt.AlignCenter)
self.table = QTableWidget(4, 4)
self.table.setFixedSize(545, 185)
self.table.setHorizontalHeaderLabels(["Модель", "Клас", "Ймовірність",
"Час (сек)"])
layout.addWidget(self.table, alignment=Qt.AlignCenter)
```

При ініціалізації графічного вікна також завантажуються усі моделі, а саме метод машинного навчання та чотири побудовані згорткові нейронні мережі, отримані під час тренування ваги, що збережені в форматі .pth, що використовуються для швидкого та легкого збереження стану мережі.

Під час завантаження зображення з пристрою воно класифікується, отримані результати заносяться у таблицю: обраний клас, ймовірність вибору обраного класу, а також час класифікації (рис. 3.7).

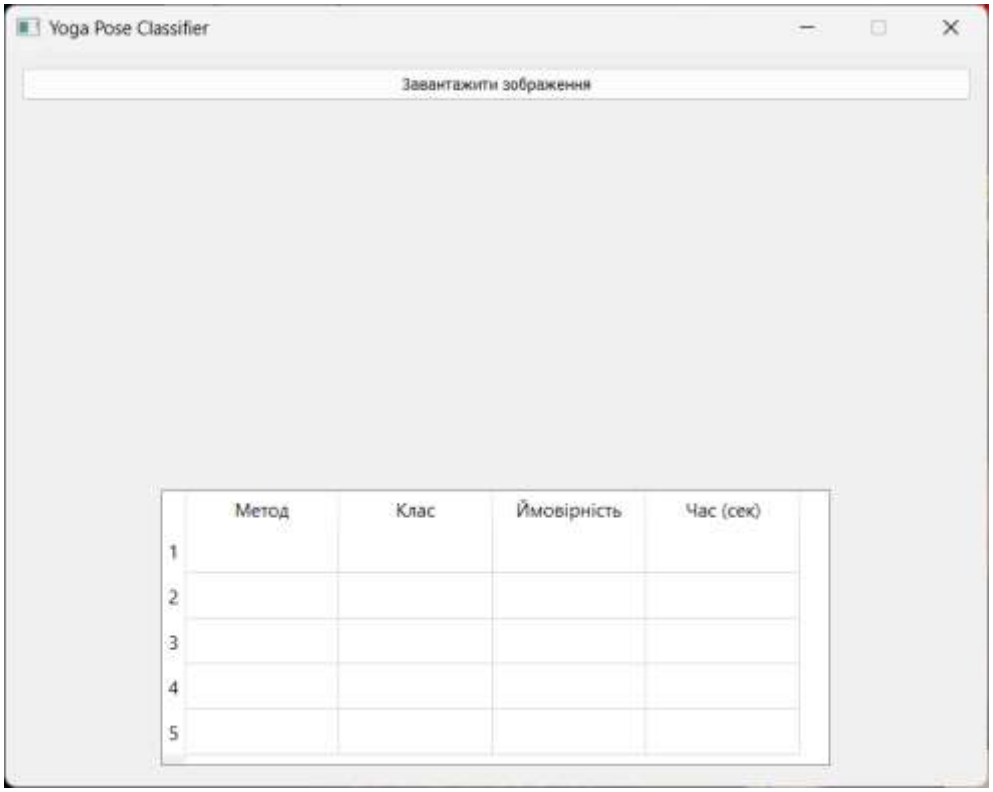


Рисунок 3.7 – Початковий стан графічного інтерфейсу

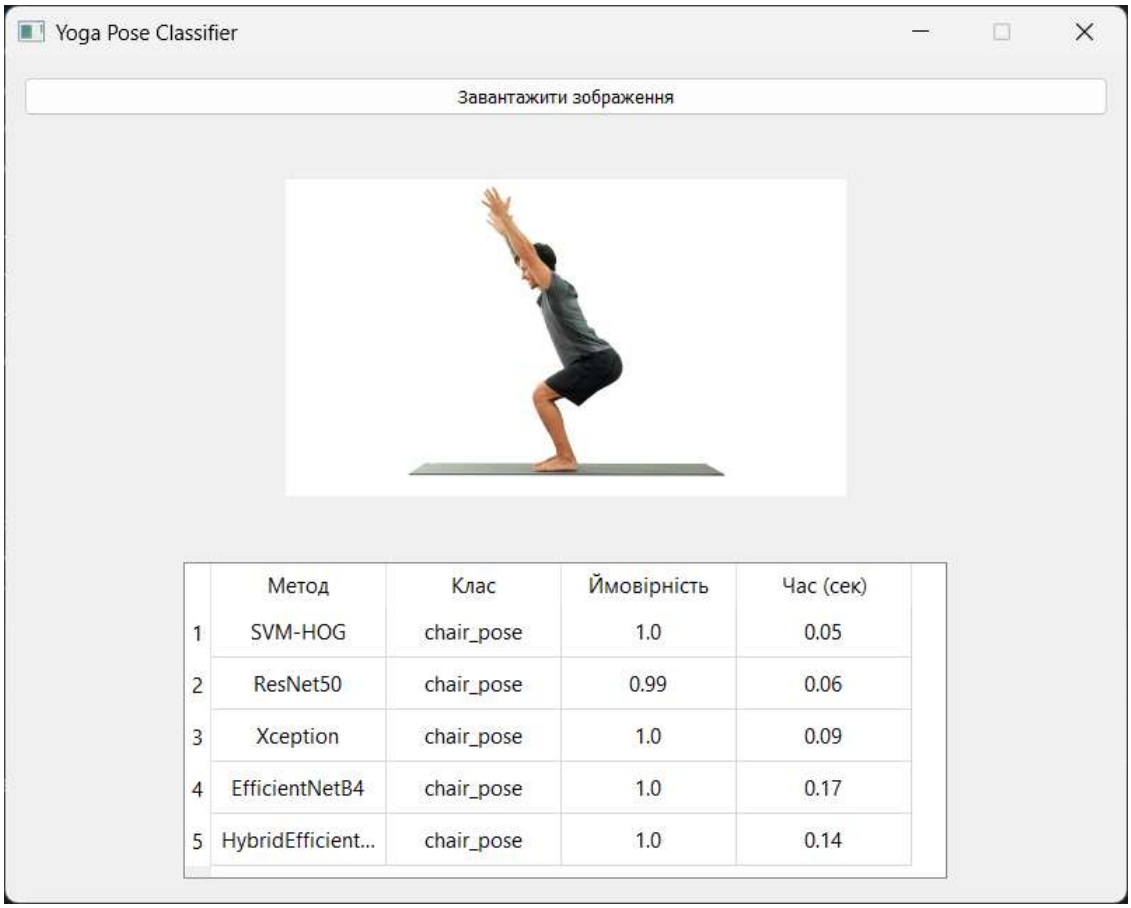
Лістинг 3.16 Завантаження навчених моделей:

```
model1 = resnet50(num_classes=16)
model2 = xception(num_classes=16)
model3 = efficientnet_b4(num_classes=16)
model4 = HybridEfficientNet(num_classes=16)
model1.load_state_dict(torch.load("resnet50.pth", map_location="cpu"))
model2.load_state_dict(torch.load("xception_best.pth", map_location="cpu"))
model3.load_state_dict(torch.load("effnet.pth", map_location="cpu"))
model4.load_state_dict(torch.load("HybridEEffNet__.pth",
map_location="cpu"))
model1.eval()
model2.eval()
model3.eval()
model4.eval()
```

Таким чином, за допомогою графічного інтерфейсу (рис. 3.6) надається можливість наглядно побачити та оцінити роботу побудованих нейронних мереж на конкретних зображеннях поз йоги.

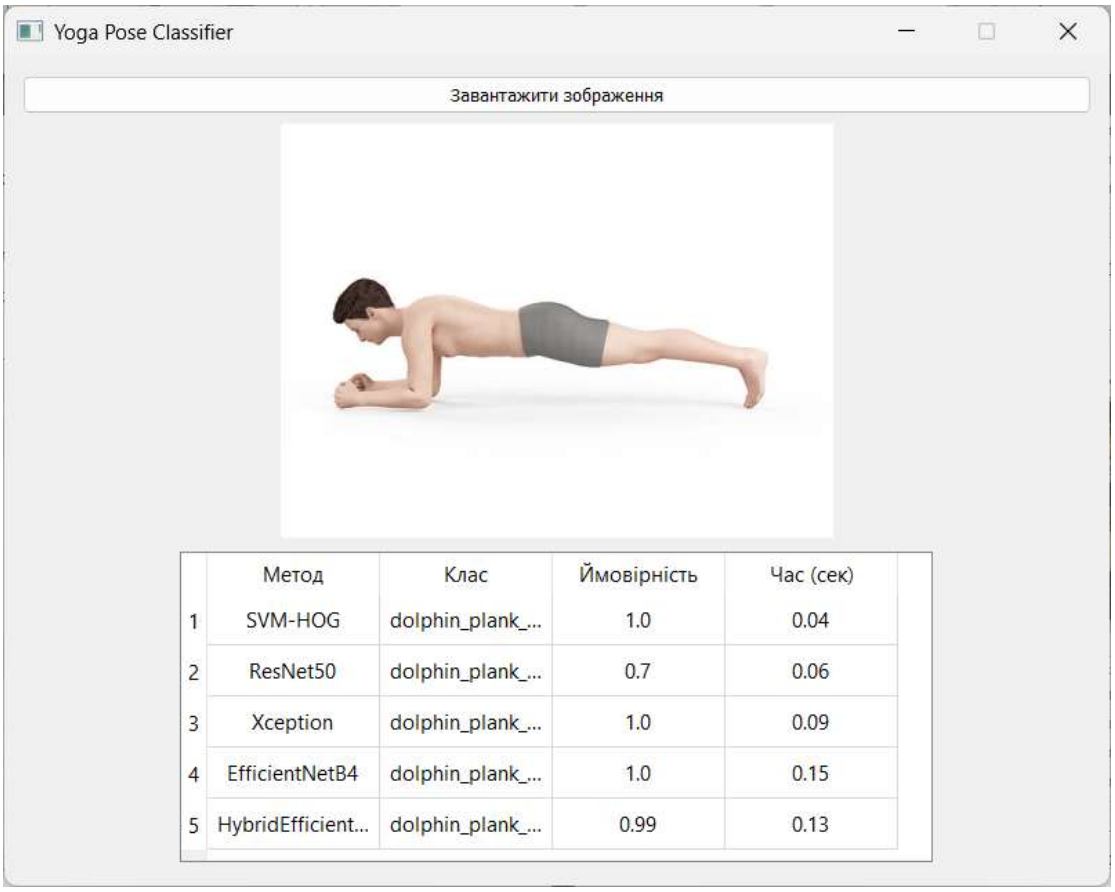
3.3 Тестування методів класифікації вправ йоги

Проведено тестову класифікацію еталонних зображень, отримано результати тестування та час класифікації (рис. 3.8 – рис. 3.23).



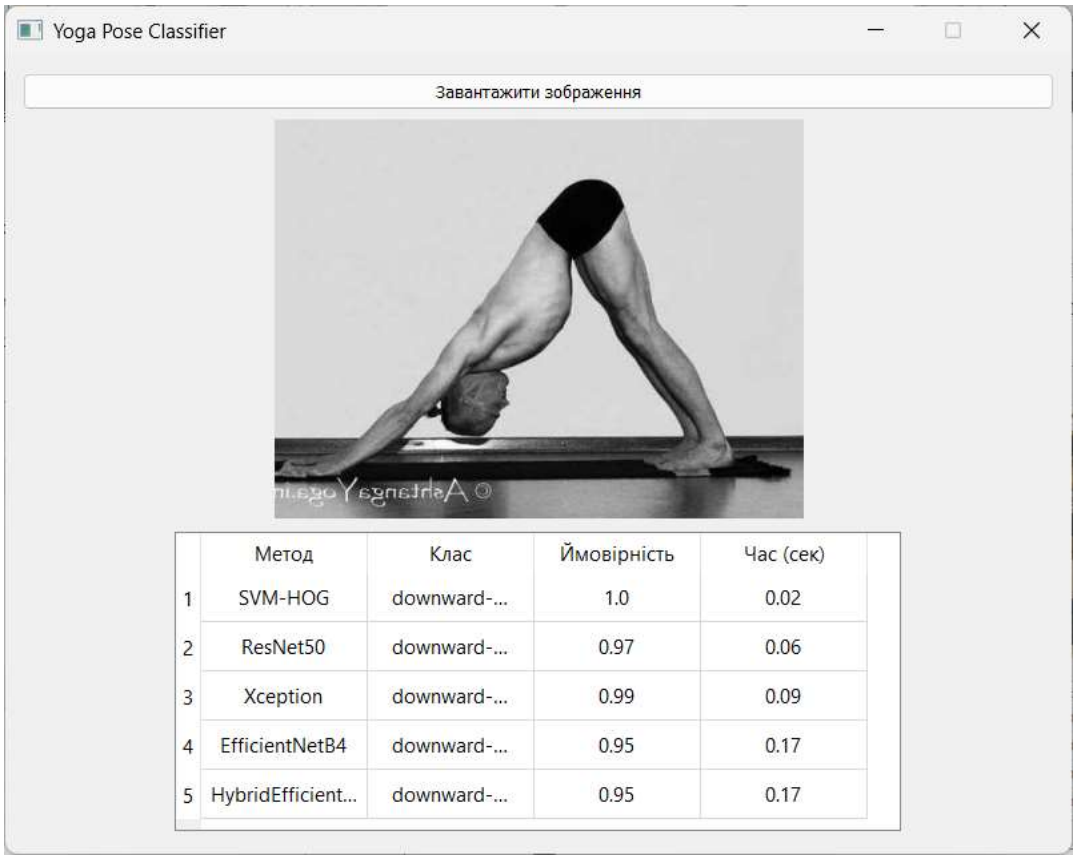
	Метод	Клас	Ймовірність	Час (сек)
1	SVM-HOG	chair_pose	1.0	0.05
2	ResNet50	chair_pose	0.99	0.06
3	Xception	chair_pose	1.0	0.09
4	EfficientNetB4	chair_pose	1.0	0.17
5	HybridEfficient...	chair_pose	1.0	0.14

Рисунок 3.8 – Результат класифікації зображення класу chair_pose



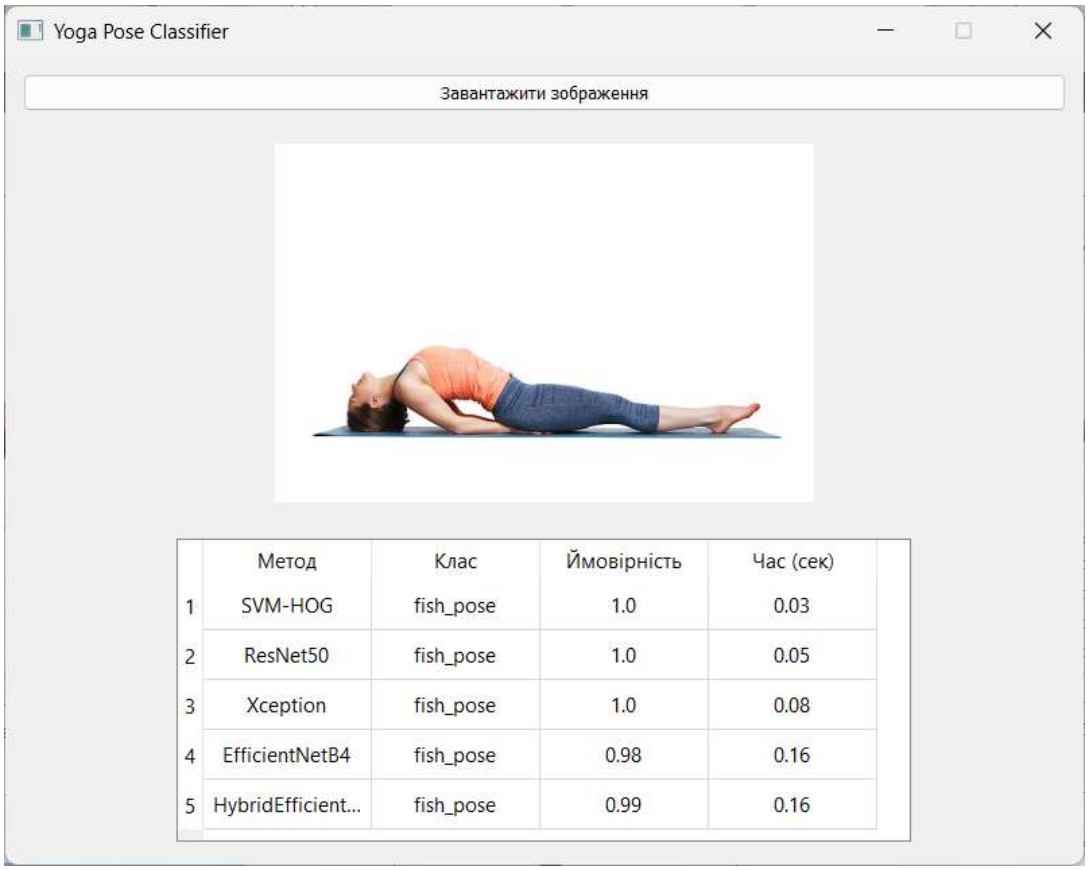
	Метод	Клас	Ймовірність	Час (сек)
1	SVM-HOG	dolphin_plank_...	1.0	0.04
2	ResNet50	dolphin_plank_...	0.7	0.06
3	Xception	dolphin_plank_...	1.0	0.09
4	EfficientNetB4	dolphin_plank_...	1.0	0.15
5	HybridEfficient...	dolphin_plank_...	0.99	0.13

Рисунок 3.9 – Результат класифікації зображення класу dolphin_plank_pose



	Метод	Клас	Ймовірність	Час (сек)
1	SVM-HOG	downward-...	1.0	0.02
2	ResNet50	downward-...	0.97	0.06
3	Xception	downward-...	0.99	0.09
4	EfficientNetB4	downward-...	0.95	0.17
5	HybridEfficient...	downward-...	0.95	0.17

Рисунок 3.10 – Результат класифікації зображення класу downward-facing_dog_pose



	Метод	Клас	Ймовірність	Час (сек)
1	SVM-HOG	fish_pose	1.0	0.03
2	ResNet50	fish_pose	1.0	0.05
3	Xception	fish_pose	1.0	0.08
4	EfficientNetB4	fish_pose	0.98	0.16
5	HybridEfficient...	fish_pose	0.99	0.16

Рисунок 3.11 – Результат класифікації зображення класу fish_pose

Рисунок 3.12 – Результат класифікації зображення класу goddess_pose

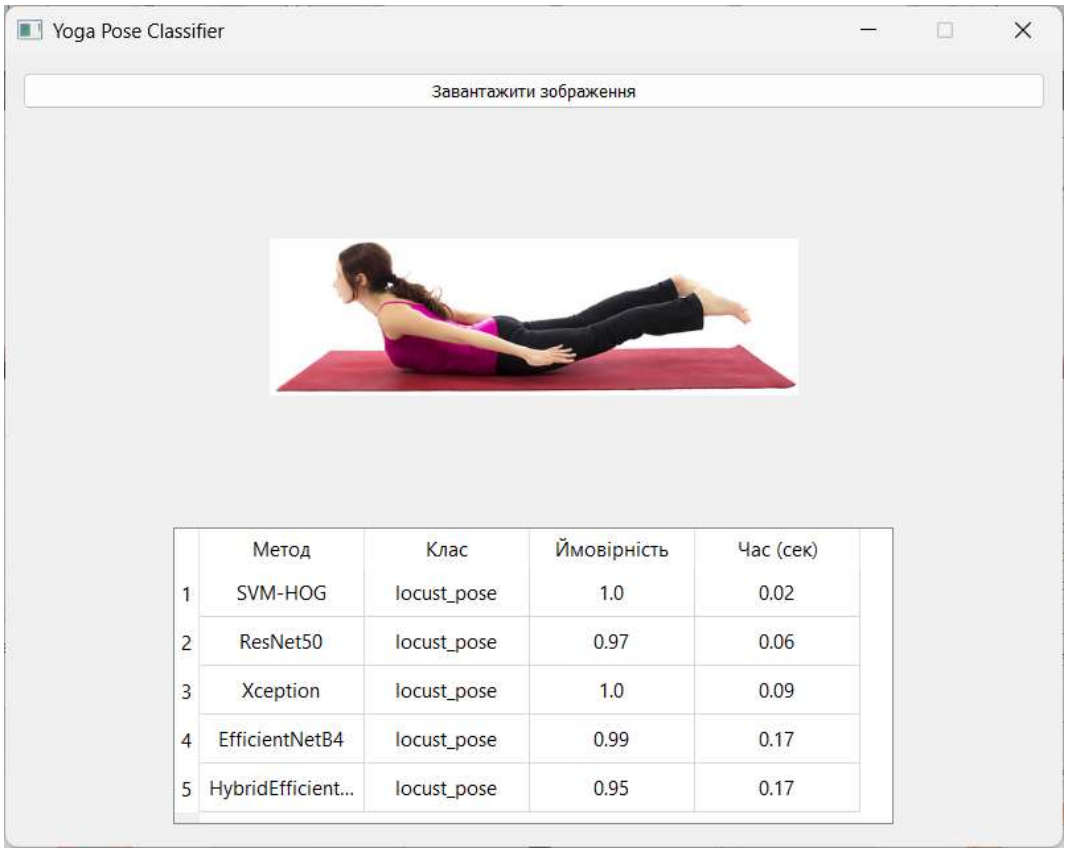
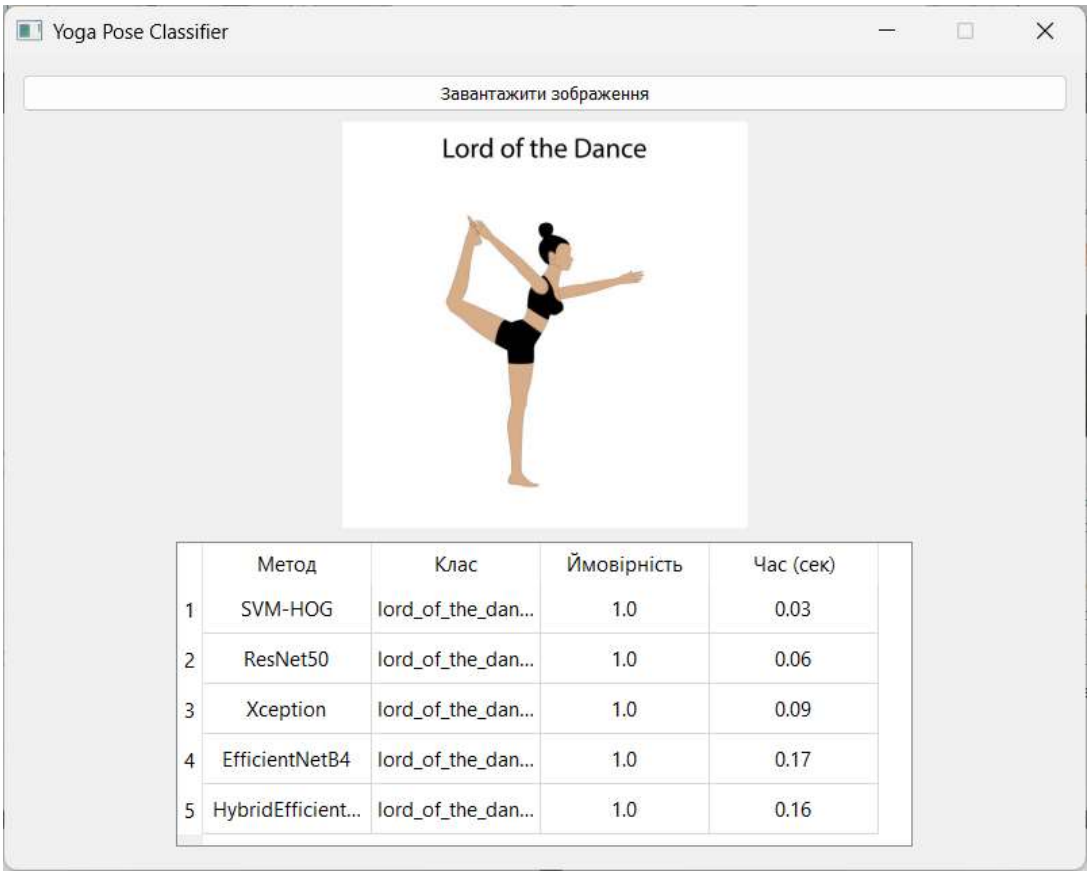
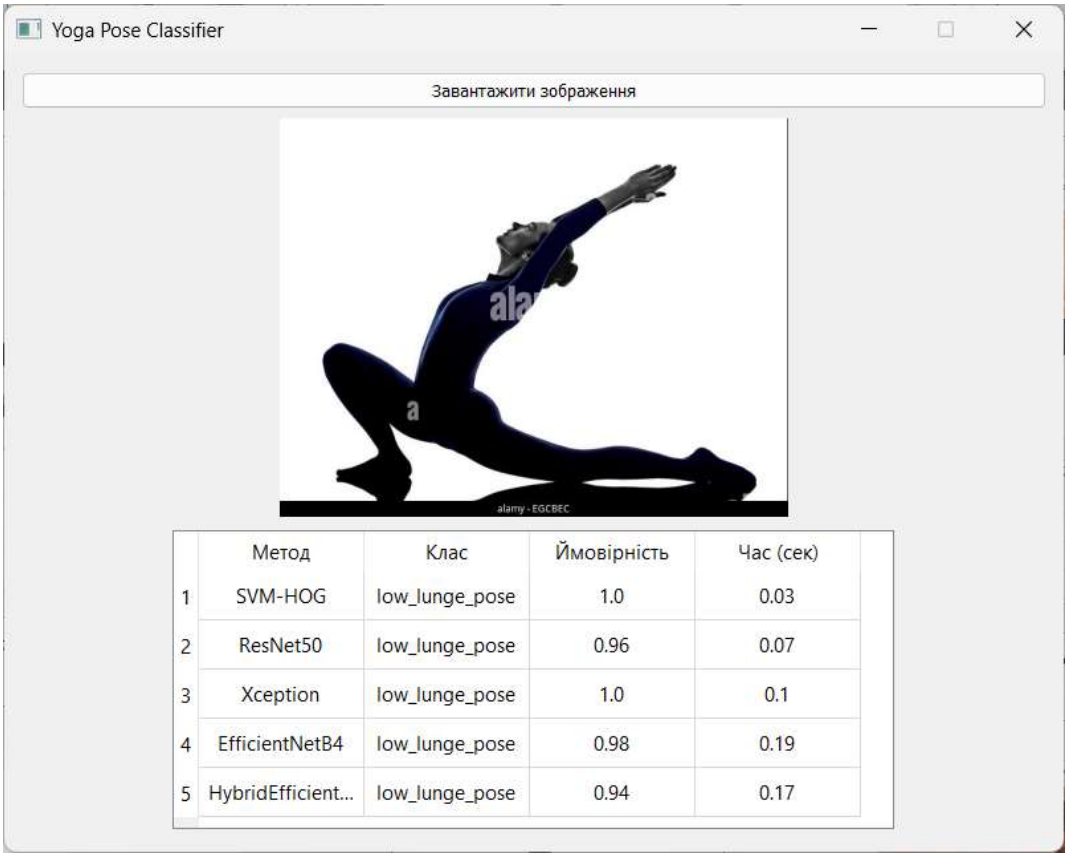


Рисунок 3.13 – Результат класифікації зображення класу locust_pose



	Метод	Клас	Ймовірність	Час (сек)
1	SVM-HOG	lord_of_the_dan...	1.0	0.03
2	ResNet50	lord_of_the_dan...	1.0	0.06
3	Xception	lord_of_the_dan...	1.0	0.09
4	EfficientNetB4	lord_of_the_dan...	1.0	0.17
5	HybridEfficient...	lord_of_the_dan...	1.0	0.16

Рисунок 3.14 – Результат класифікації зображення класу lord_of_the_dance_pose



	Метод	Клас	Ймовірність	Час (сек)
1	SVM-HOG	low_lunge_pose	1.0	0.03
2	ResNet50	low_lunge_pose	0.96	0.07
3	Xception	low_lunge_pose	1.0	0.1
4	EfficientNetB4	low_lunge_pose	0.98	0.19
5	HybridEfficient...	low_lunge_pose	0.94	0.17

Рисунок 3.15 – Результат класифікації зображення класу low_lunge_pose

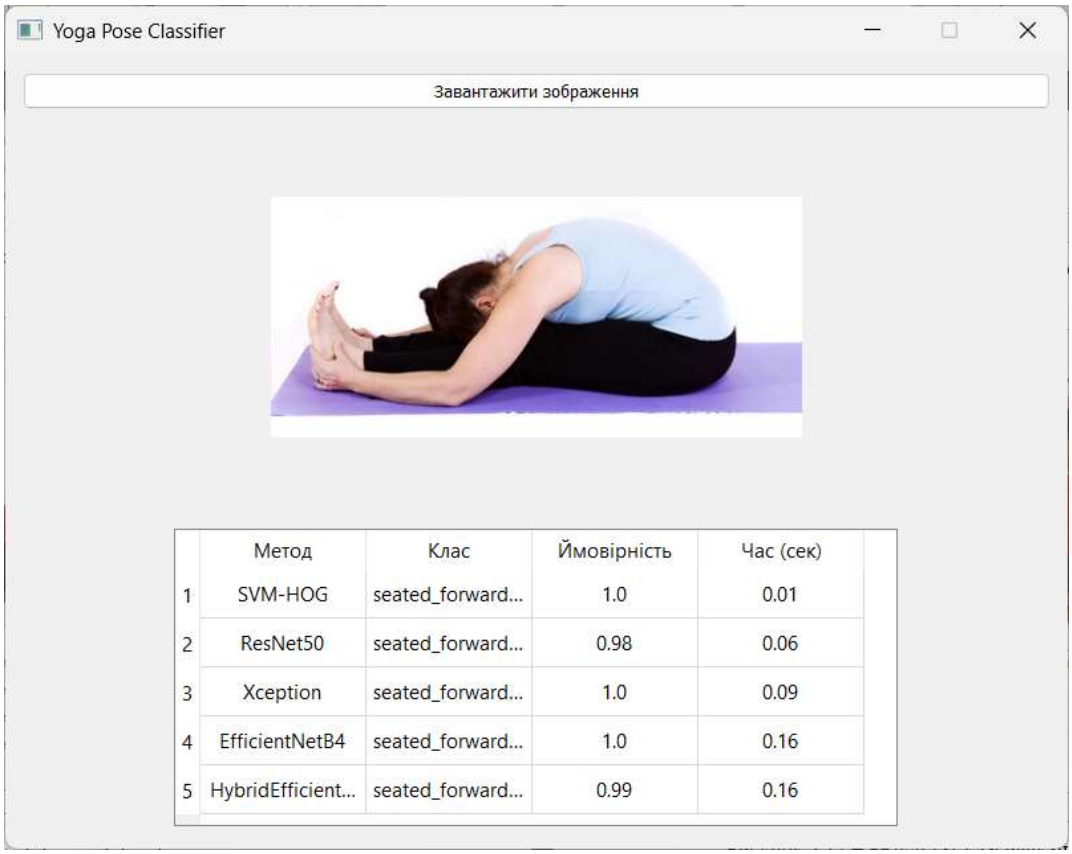


Рисунок 3.16 – Результат класифікації зображення класу seated_forward_bend_pose

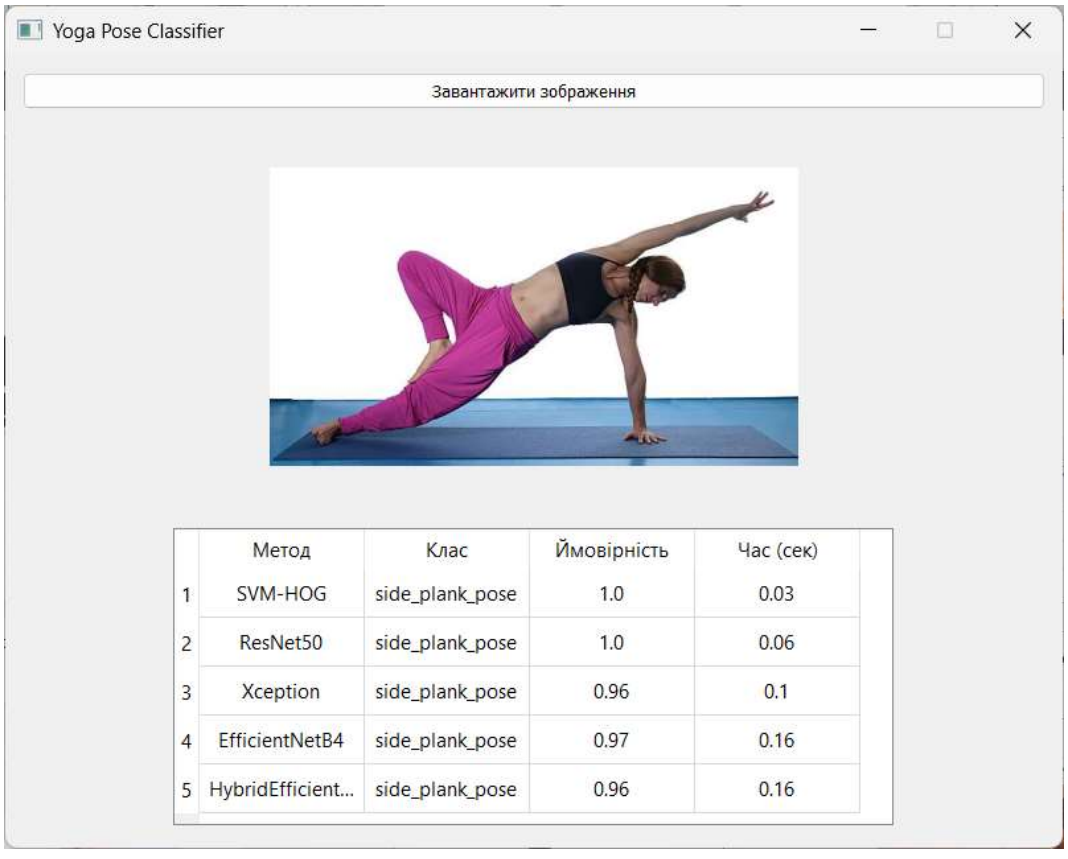
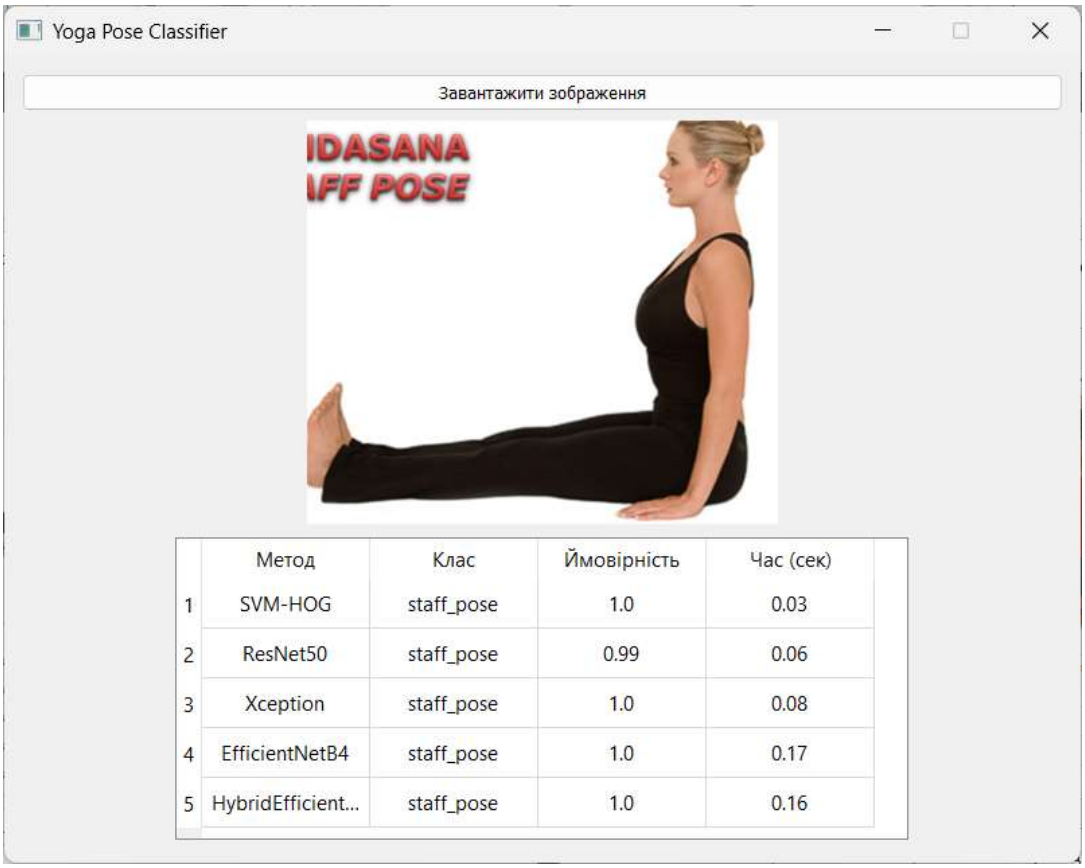
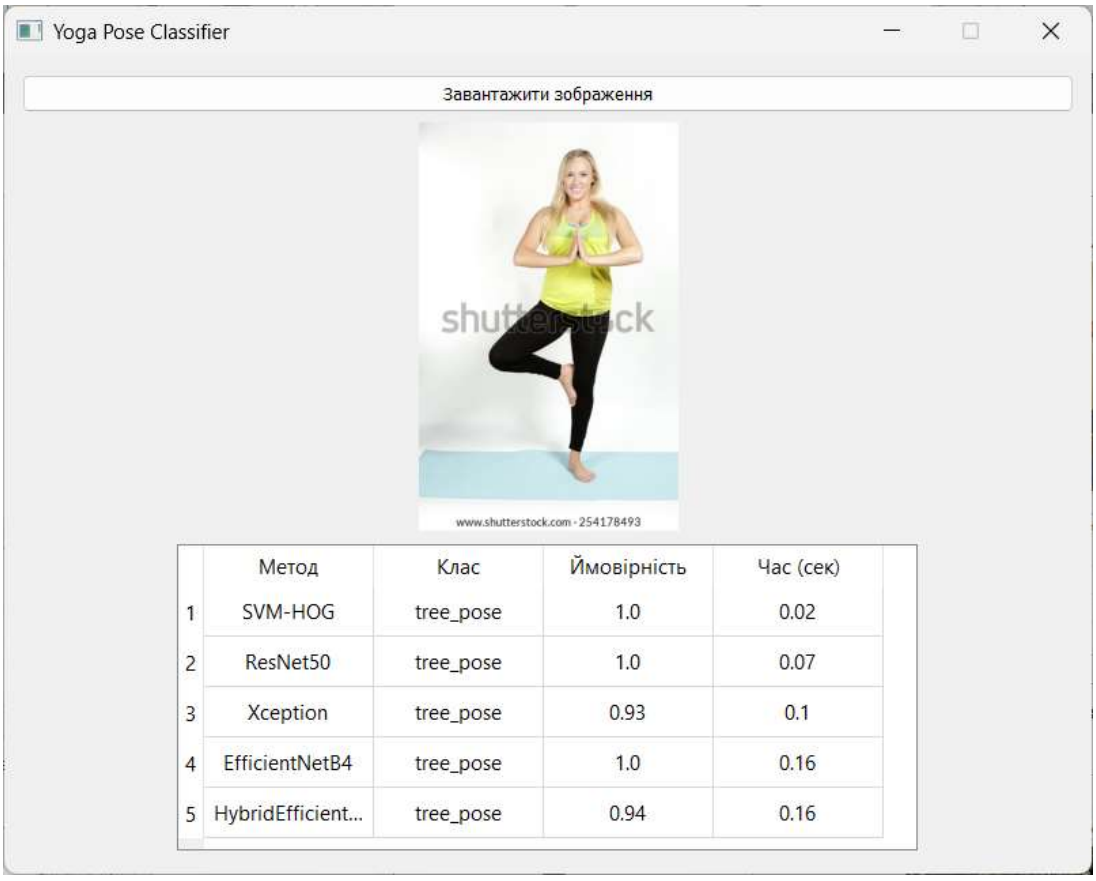


Рисунок 3.17 – Результат класифікації зображення класу side_plank_pose



	Метод	Клас	Ймовірність	Час (сек)
1	SVM-HOG	staff_pose	1.0	0.03
2	ResNet50	staff_pose	0.99	0.06
3	Xception	staff_pose	1.0	0.08
4	EfficientNetB4	staff_pose	1.0	0.17
5	HybridEfficient...	staff_pose	1.0	0.16

Рисунок 3.18 – Результат класифікації зображення класу staff_pose



	Метод	Клас	Ймовірність	Час (сек)
1	SVM-HOG	tree_pose	1.0	0.02
2	ResNet50	tree_pose	1.0	0.07
3	Xception	tree_pose	0.93	0.1
4	EfficientNetB4	tree_pose	1.0	0.16
5	HybridEfficient...	tree_pose	0.94	0.16

Рисунок 3.19 – Результат класифікації зображення класу tree_pose

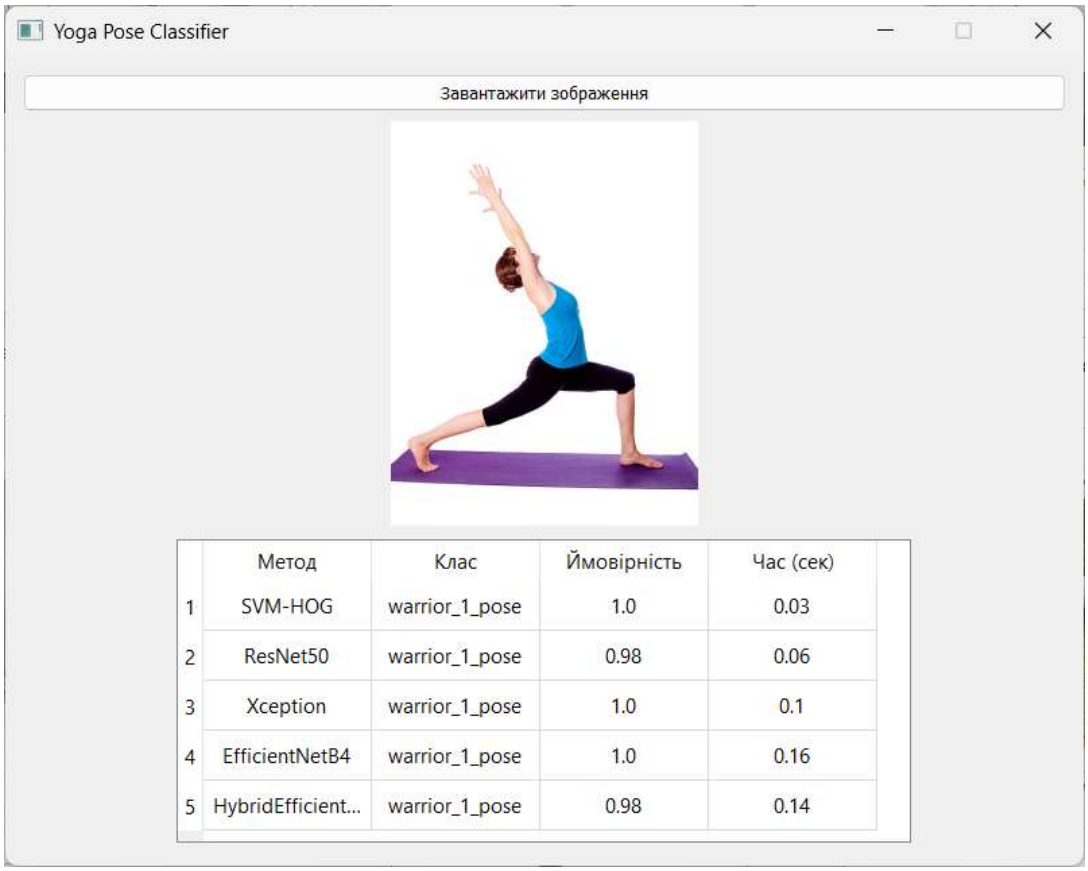


Рисунок 3.20 – Результат класифікації зображення класу warrior_1_pose

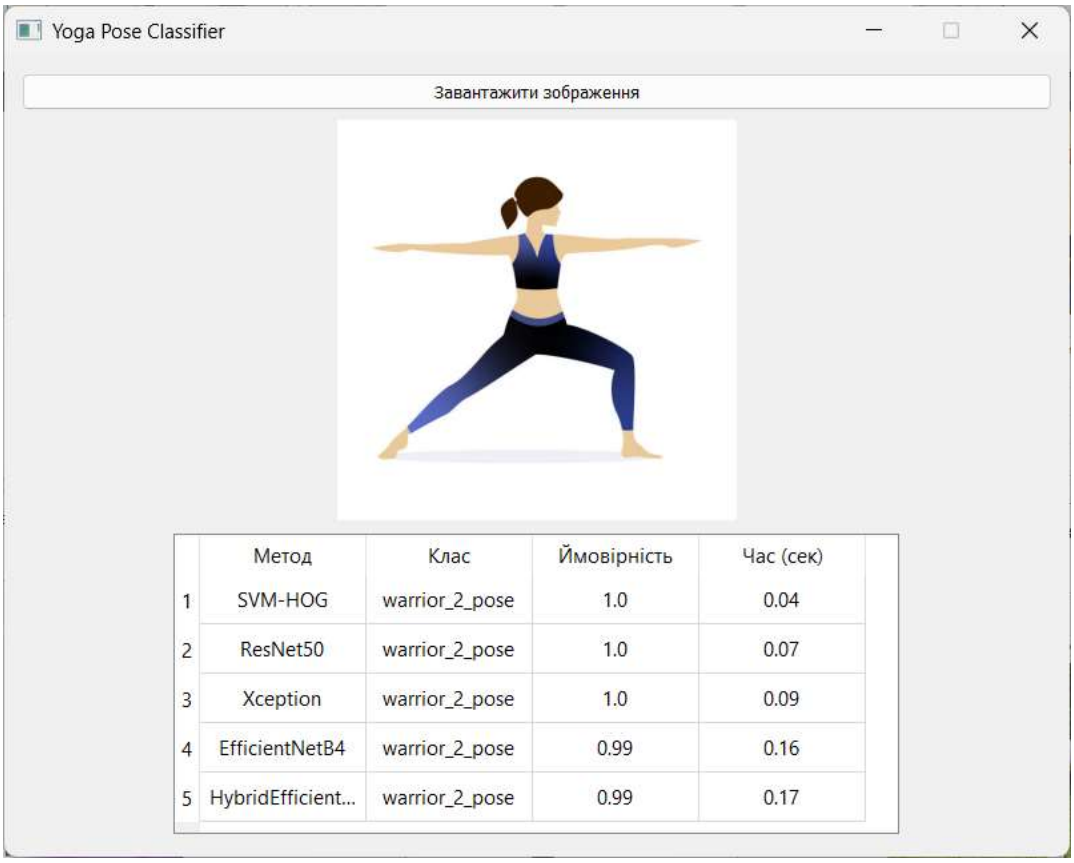
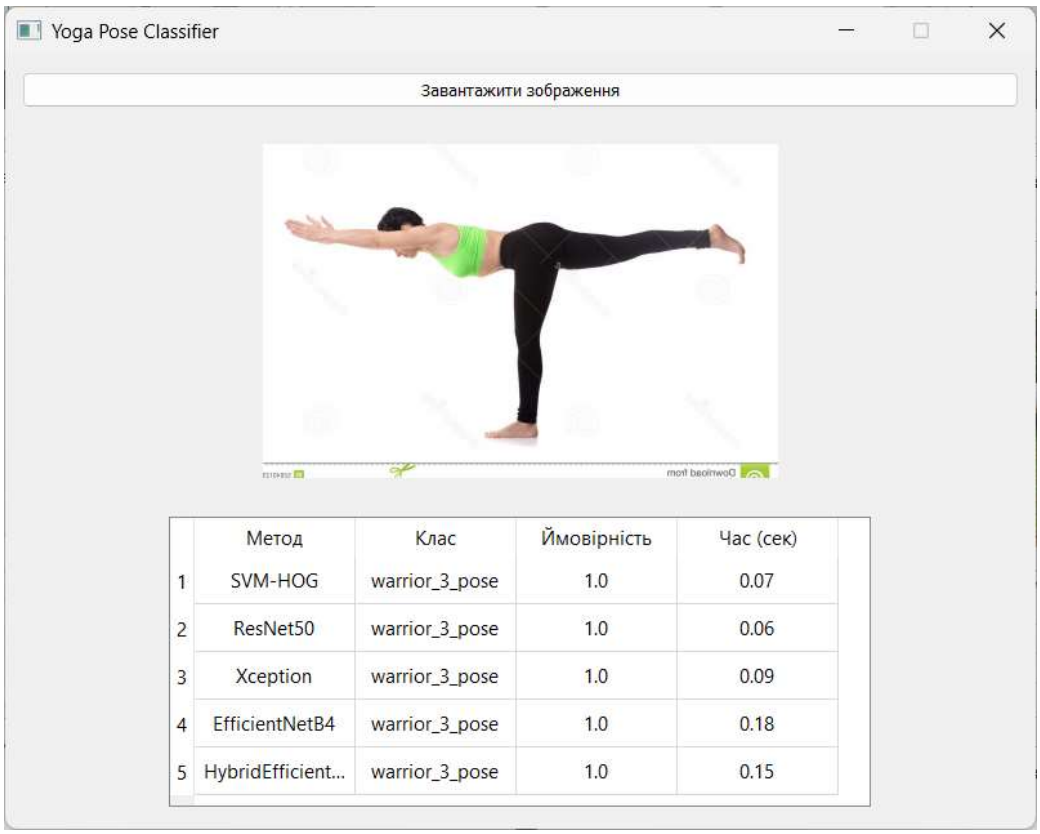
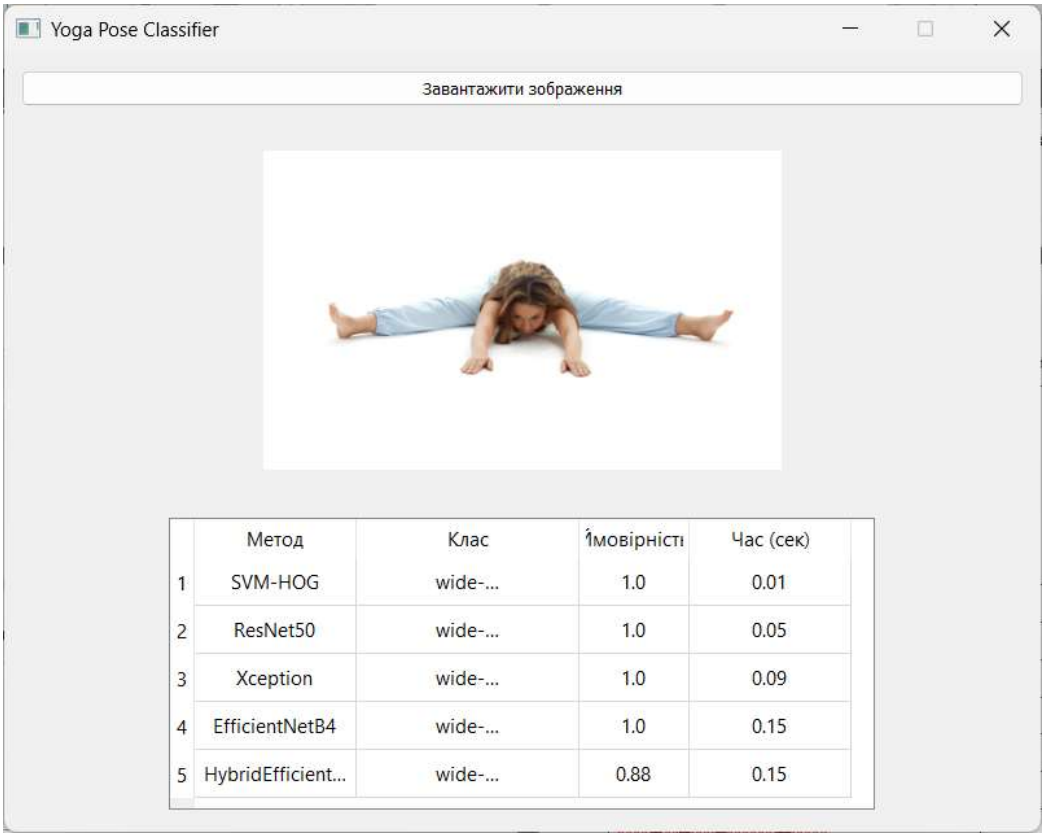


Рисунок 3.21 – Результат класифікації зображення класу warrior_2_pose



	Метод	Клас	Ймовірність	Час (сек)
1	SVM-HOG	warrior_3_pose	1.0	0.07
2	ResNet50	warrior_3_pose	1.0	0.06
3	Xception	warrior_3_pose	1.0	0.09
4	EfficientNetB4	warrior_3_pose	1.0	0.18
5	HybridEfficient...	warrior_3_pose	1.0	0.15

Рисунок 3.22 – Результат класифікації зображення класу warrior_3_pose



	Метод	Клас	Ймовірність	Час (сек)
1	SVM-HOG	wide-...	1.0	0.01
2	ResNet50	wide-...	1.0	0.05
3	Xception	wide-...	1.0	0.09
4	EfficientNetB4	wide-...	1.0	0.15
5	HybridEfficient...	wide-...	0.88	0.15

Рисунок 3.23 – Результат класифікації зображення класу wide-angle_seated_forward_bend_pose

3.4 Порівняльний аналіз досліджених методів класифікації вправ йоги

Результати тестування занесено до таблиці 3.2.

Найбільш ефективною з точки зору точності виявилася модель EfficientNet-B4 із середньою точністю на вибірці 93,41%. Такий результат був очікуваним, оскільки лінійка моделей EfficientNet вважається найбільш досконалою у порівнянні з ResNet-50 та Xception. До її архітектури було залучено переваги інших моделей, що розглядалися, а також використовуються вдосконалені функції активації.

У той же час, модифікована версія EfficientNet-B4, HybridEfficientNet**,** показала найменший відсоток правильної класифікації зображень виконання вправ йоги порівняно з іншими моделями, причому класична версія має перевагу в 12%.

Таблиця 3.2 – Розрахунок точності моделей для кожного класу

Назва класу	**HOG-SVM**	**ResNet-50**	**Xception**	**EfficientNet-B4**	**HybridEfficientNet**
1	**2**	**3**	**4**	**5**	**6**
chair_pose	0,74	0,91	0,91	0,96	0,91
dolphin_plank_pose	0,58	0,88	0,88	0,92	0,88
downward-facing_dog_pose	0,88	0,83	0,92	0,88	0,88
fish_pose	0,54	0,96	0,83	0,96	0,75
goddess_pose	0,58	0,96	0,83	0,96	0,75
locust_pose	0,75	0,75	1	1	0,88
lord_of_the_dance_pose	0,83	0,96	0,88	0,83	0,88
low_lunge_pose	0,83	0,63	0,79	0,79	0,54

Продовження таблиці 3.2

1	2	3	4	5	6
seated_forward_bend_pose	0,79	0,88	1	0,96	0,83
side_plank_pose	0,71	0,79	0,92	0,96	0,71
staff_pose	0,75	0,92	0,92	1	0,96
tree_pose	0,79	1	0,92	0,92	0,79
warrior_1_pose	0,79	0,96	1	1	1
warrior_2_pose	0,71	0,92	0,96	0,83	0,88
warrior_3_pose	0,79	0,79	1	0,92	0,83
wide-angle_seated_forward_bend_pose	0,67	0,90	0,90	0,86	0,62
Середнє значення	**0,75**	**0,88**	**0,91**	**0,93**	**0,82**

Це може бути пов’язано з недостатнім рівнем налаштування гіперпараметрів або невірно підібраними розмірами вхідних зображень.

Вирішення подібних проблем можна віднести до перспектив для подальшої роботи.

Проаналізувавши дані про швидкодію моделей (табл. 3.3) можна прийти до висновку, що найбільш швидкими виявилися моделі ResNet-50 та Xception. Це зумовлено тим, що їх архітектура простіша за архітектуру моделі EfficientNet-B4.

На збільшення швидкості класифікації відносно інших двох моделей вплинули початкові розміри вхідних зображень. Під час класифікації моделями ResNet-50 та Xception використовувалися дані меншого розміру у порівнянні з іншими двома моделями.

Найбільш повільною виявилася модель EfficientNet-B4 із середнім результатом у 0,2 секунди. У той же час її модифікована версія впоралася за 0,15 секунд, що на 0,05 секунд у середньому менше.

Таблиця 3.3 – Розрахунок часу класифікації моделей для кожного класу, с

Назва класу	**HOG-SVM**	**ResNet-50**	**Xception**	**EfficientNet-B4**	**HybridEfficientNet**
chair_pose	0,05	0,15	0,09	0,23	0,18
dolphin_plank_pose	0,03	0,06	0,1	0,19	0,16
downward-facing_dog_pose	0,06	0,07	0,09	0,22	0,15
fish_pose	0,03	0,07	0,09	0,21	0,15
goddess_pose	0,03	0,06	0,09	0,18	0,15
locust_pose	0,04	0,06	0,09	0,18	0,15
lord_of_the_dance_pose	0,03	0,06	0,09	0,18	0,14
low_lunge_pose	0,03	0,06	0,09	0,19	0,15
seated_forward_bend_pose	0,02	0,06	0,09	0,19	0,14
side_plank_pose	0,05	0,06	0,09	0,21	0,18
staff_pose	0,06	0,06	0,09	0,21	0,16
tree_pose	0,03	0,06	0,09	0,21	0,16
warrior_1_pose	0,03	0,06	0,09	0,19	0,16
warrior_2_pose	0,03	0,06	0,09	0,21	0,18
warrior_3_pose	0,03	0,07	0,09	0,21	0,14
wide-angle_seated_forward_bend_pose	0,03	0,07	0,1	0,21	0,14
Середнє значення	**0,03**	**0,07**	**0,09**	**0,2**	**0,15**

Таким чином, HybridEfficientNet має швидкодію в 1,3 рази більшу за свою класичну версію та за швидкодією наближається до перших двох моделей, не зважаючи на те, що початковий розмір зображення для модифікована версії також становить 380.

3.5 Перспективи подальшої роботи

Перспективами до подальшого вдосконалення результатів даного дослідження можуть бути:

– збільшення класів у тестовий вибірці;

– удосконалення модифікованої моделі HybridEfficientNet;

– вдосконалення інтерфейсу застосунку.

Пропозиції для подальшої роботи вдосконалення модифікованої моделі HybridEfficientNet:

– дослідження зміни точності та швидкості при різних розмірах вхідних зображень;

– дослідження зміни точності та швидкості при додаванні додаткових блоків;

– можлива імплементація блоків з інших моделей, що не були досліджені у цій роботі.

ВИСНОВКИ

Таким чином, у кваліфікаційній роботі досліджено методи класифікації вправ йоги, реалізованих засобами машинного навчання та нейронних мереж та вирішено такі завдання:

– проведено аналіз літературних джерел щодо апробації методів класифікації об’єктів на зображеннях, що дало можливість виявити сучасний стан дослідженої проблематики, недоліки та переваги аналогічних напрямків;

– проведено аналіз сучасних методів та готових комерційних сервісів класифікації положення людини на зображеннях, що дало можливість детально вивчити їх недоліки та переваги для подальшого вибору найкращих для вирішення поставленої задачі;

– деталізовано кожний із вибраних методів класифікації об’єктів на зображеннях, основаних на згорткових нейронних мережах, що дало можливість запрограмувати кожний із вибраних методів;

– візуалізовано структуру блоків та структуру кожного із вибраних методів, що дозволило глибше зрозуміти архітектуру методів та, за можливості, оптимізувати або удосконалити їх;

– розроблено програмний застосунок, що надав змогу класифікувати вправи йоги на зображенні одним методом машинного навчання та чотирма методами згорткових нейронних мереж, це дозволило провести необхідне дослідження та задовольнити мету кваліфікаційної роботи.

У рамках кваліфікаційної роботи проведено дослідження та порівняння методів HOG-SVM, ResNet-50, Xception та EfficientNet-B4 шляхом програмної реалізації застосунку, що надало змогу класифікувати вправи йоги на зображенні. Побудовано модифіковану версію EfficientNet-B4 із застосуванням блоків інших моделей, яку під час дослідження порівняно з класичними моделями.

Підібрано набір зображень поз йоги, що включає 15 класів. Розширено його з 1500 до 3000 зображень та проведено на ньому навчання.

Метод HOG-SVM показав найнижчий результат у плані навчання. Це зумовлено тим, що, можливо, пози йоги є складними для класифікації нескладними методами. Порівняно з ним, згорткові нейронні мережі показали більш високі результати, серед них найвищий показала модель EfficientNet-B4.

Здійснено тестування застосунку та порівняно результати класифікації зображень виконання поз йоги кожним із п’яти методів.

Результат порівняння даних показав:

– найбільш швидким методом виявився HOG-SVM, однак він показав найнижчий відсоток точності класифікації;

– найбільш точним методом виявилася згорткова нейронна мережа EfficientNet-B4, однак вона виявилася найповільнішою серед усіх розглянутих методів;

– модифікована згорткова мережа HybridEfficientNet показала нижчий результат точності відносно інших нейронних мереж, хоча і вищий за HOG-SVM, однак виявилася у 1,3 рази швидше за свою класичну версію.

Наукова новизна роботи полягає у результатах порівняння методів класифікації зображень, основаних на машинному навчанні та нейронних мережах.

Таким чином, розвиток інформаційних технологій включає використання комп’ютерних систем для оброблення, зберігання та передачі інформації, що кардинально змінило способи спілкування, роботи, навчання та відпочинку, надавши доступ до величезних обсягів знань [24–43].

Результати роботи апробовано у вигляді 2 тез доповідей під час IV Міжнародної науково-практичної конференції «Технології, теорії та розробки: сучасне наукове викладання» [44] та XII Міжнародної науково-практичної конференції «Основні тенденції в науці, викладанні та сучасному навчанні» [45].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 1280px-Plank.jpg (1280×853). URL: https://upload.wikimedia.org/wikipedia/commons/thumb/e/ee/Plank.jpg/1280px-Plank.jpg (дата звернення 10.11.2025).

2. Trikonasana_Yoga-Asana_Nina-Mel.jpg (591×678). URL: https://upload.wikimedia.org/wikipedia/commons/9/9d/Trikonasana_Yoga-Asana_Nina-Mel.jpg (дата звернення 10.11.2025).

3. Balasana.JPG (1737×870). URL: https://upload.wikimedia.org/wikipedia/commons/0/0b/Balasana.JPG (дата звернення 10.11.2025).

4. Vriksasana_Yoga-Asana_Nina-Mel.jpg (493×922). URL: https://upload.wikimedia.org/wikipedia/commons/7/72/Vriksasana_Yoga-Asana_Nina-Mel.jpg (дата звернення 10.11.2025).

5. Virasana_Yoga-Asana_Nina-Mel.jpg (556×610). URL: https://upload.wikimedia.org/wikipedia/commons/1/11/Virasana_Yoga-Asana_Nina-Mel.jpg (дата звернення 10.11.2025).

6. Viparita-Karani_Yoga-Asana_Nina-Mel.jpg (523×753). URL: https://upload.wikimedia.org/wikipedia/commons/3/3b/Viparita-Karani_Yoga-Asana_Nina-Mel.jpg (дата звернення 10.11.2025).

7. Matsyasana_Yoga-Asana_Nina-Mel.jpg (967×528). URL: https://upload.wikimedia.org/wikipedia/commons/e/ea/Matsyasana_Yoga-Asana_Nina-Mel.jpg (дата звернення 10.11.2025).

8. Bhujangasana_Yoga-Asana_Nina-Mel.jpg (979×552). URL: https://upload.wikimedia.org/wikipedia/commons/a/ae/Bhujangasana_Yoga-Asana_Nina-Mel.jpg (дата звернення 10.11.2025).

9. Virabhadrasana_III_in_Egypt.jpg (4016×4042). URL: https://upload.wikimedia.org/wikipedia/commons/4/4e/Virabhadrasana_III_in_Egypt.jpg (дата звернення 10.11.2025).

10. Bound_wheel_pose.jpg (507×420). URL: https://upload.wikimedia.org/wikipedia/commons/f/fb/Bound_wheel_pose.jpg (дата звернення 10.11.2025).

11. KinesteX AI - Personal Fitness Trainer. URL: https://www.kinestex.com (дата звернення 15.10.2025).

12. Motion Tracking Exercise Platform for Physio and Fitness | Kemtai. URL; https://kemtai.com (дата звернення 15.10.2025).

13. GitHub - AlessandroDiPatria/GymAI: Full stack Web application that use AI to track and recognize user's movements. URL; https://github.com/AlessandroDiPatria/GymAI (дата звернення 15.10.2025).

14. Kumar, K. A., & Geethakalyani, V. (2025). Analysis of Yoga Position using Machine Learning based Approach. *In 2025 3rd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)* (pp. 345-351). IEEE.

15. Bhandage, V., Prabhu, S., Hadimani, B. S., Chadaga, K., Sampathila, N., & Shetty, S. (2024). Classification of Surya Namaskar Yoga Asanas: A Sequential Combination of Predominant Poses for Physical and Mental Health. *IEEE Access*.

16. Rajendran, A. K., & Sethuraman, S. C. (2024). YogiCombineDeep: Enhanced yogic posture classification using combined deep fusion of VGG16 and VGG19 features. *IEEE Access*, 12, 139165-139180.

17. Talaat, A. S. (2023). Novel deep learning models for yoga pose estimator. *SN Applied Sciences*, 5(12), 341.

18. Swain, D., Satapathy, S., Acharya, B., Shukla, M., Gerogiannis, V. C., Kanavos, A., & Giakovis, D. (2022). Deep learning models for yoga pose monitoring. *Algorithms*, 15(11), 403.

19. Maddukuri, N., & Ummity, S. R. (2023). Yoga Pose prediction using Transfer Learning Based Neural Networks.

20. Rawat, A., Balasundaram, A., & Vaithilingam, C. A. (2025). Angle-based regularized deep learning model for gauging effectiveness in performing yoga postures. *Sport Sciences for Health*, 1-17.

21. Narayanan, S. S., Misra, D. K., Arora, K., & Rai, H. (2021, May). Yoga pose detection using deep learning techniques. *In Proceedings of the International Conference on Innovative Computing & Communication (ICICC)*.

22. Adeyemi, J. O. (2024). Enhancing Elderly Wellness through AI-Powered Yoga and Exercise Support Systems.

23. Bang, G. S., & Park, S. B. (2024). Workout classification using a convolutional neural network in ensemble learning. *Sensors*, 24(10), 3133.

24. Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. *In International conference on machine learning* (pp. 6105-6114). PMLR.

25. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

26. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, Сучасні інформаційні системи, 6(3), С. 5-12.

27. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.

28. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Engineering Research*, 7(9), pp. 64-72.

29. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, *IEEE Access*, vol. 11, pp. 126938-126949.

30. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2025) Image description compression in classification structural methods, *IEEE Access*, vol. 13, pp. 43631-43641.

31. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25-36.

32. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57-70.

33. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 1, pp. 113-125.

34. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, *IEEE Access*, vol. 12, pp. 73376-73385.

35. Tvoroshenko I., Pomazan V., Gorokhovatskyi V., and Kobylin O. (2023) Application of video data classification models using convolutional neural networks, *International Journal of Academic and Applied Research*, 7(11), pp. 134-145.

36. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2024) Improving the effectiveness of image classification structural methods by compressing the description according to the information content criterion, *Computers, Materials & Continua*, vol. 80, no. 2, pp. 3085-3106.

37. Gorokhovatskyi V., Chmutov Y., Tvoroshenko I., and Kobylin O. (2025) Reducing computational costs by compressing the structural description in image classification methods, *Advanced Information Systems*, vol. 9, no. 1, pp. 5–12.

38. Yakovleva O., Matúšová S., Tvoroshenko I., and Isaiev Y. (2024) Visitor counting based on video stream analysis from surveillance cameras to solve various business problems, *Verejná správa a regionálny rozvoj ekonómia, manažment a marketing*, XX(1), pp. 67-87.

39. Bohdan N., Tvoroshenko I., Gorokhovatskyi V., and Kobylin O. (2025) Development of a hybrid method to enhance context memory for a chatbot application based on large language models, *International Journal of Academic Information Systems Research*, 9(10), pp. 7-18.

40. Suprun A., Tvoroshenko I., Gorokhovatskyi V., and Yakovleva O. (2025) Development and research of a method for the combined use of large language models for text generation, *International Journal of Academic and Applied Research*, 9(10), pp. 249-263.

41. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. International scientific symposium «INTELLIGENT SOLUTIONS-S». *Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium, September 28, 2023*, Kyiv-Uzhorod, Ukraine, pp. 25-27.

42. Larin I., Tvoroshenko I., Gorokhovatskyi V., and Liubchenko V. (2025) Research and comparison of facial color normalization methods relative to an etalon image, *International Journal of Academic and Applied Research*, 9(11), pp. 36-47.

43. Гороховатський В.О., Творошенко І.С. (2025) Оцінювання значущості ознак для підвищення продуктивності структурних методів класифікації зображень. *Проблеми інформатики та моделювання (ПІМ-2025). Тези двадцять п'ятої міжнародної науково-технічної конференції (25 – 28 вересня 2025 року)*. Харків: НТУ «ХПІ», С. 38-43.

44. Podshyvalova O. (2025) Review of the methods of classification of yoga exercises implemented by machine learning and neural networks, *Abstracts of IV International scientific and practical conference «Technologies, theories and developments: modern scientific teaching», (September 23 – 26, 2025). Valencia, Spain*, pp. 19-22.

45. Podshyvalova O. (2025) Review of the ResNet-50 convolutional neural network architecture, *Abstracts of XII International scientific and practical conference «Main trends in science, teaching and modern learning», (November 18 – 21, 2025). Hamburg, Germany*, pp. 19-24.