МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра комп'ютерних наук та програмної інженерії

МЕТОДИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ ПРОГРАМИ ДЛЯ РОЗПІЗНАВАННЯ ГРАФІЧНИХ ОБРАЗІВ.

Кваліфікаційна робота
на здобуття ступеня вищої освіти «бакалавр»

Виконав: студент 4 курсу 441 групи
Спеціальності: 121 Інженерія
програмного забезпечення
Освітньо-професійної програми:
Інженерія програмного забезпечення
Левченко Дмитро Олександрович
Керівник: Доцент кафедри
комп'ютерних наук та програмної
інженерії
Кравцов Геннадій Михайлович
Рецензент: Григоренко Ярослав
Сергійович

Івано-Франківськ - 2023

ЗМІСТ

ВСТУП **3**

РОЗДІЛ 1. Теоретичні основи використання нейромереж у різних сферах. **7**

1.1 Поняття нейромережі **7**

1.2 Історія та тенденції розвитку нейромереж **7**

1.2.2 Історія розвитку нейромереж **7**

1.2.2 Тенденції розвитку нейромереж **11**

1.3 Сфери використання нейромереж у наш час **14**

РОЗДІЛ 2. Аналіз існуючих видів і архітектур нейромереж. **18**

2.1 Аналіз наявних видів нейромереж **18**

2.1.1 Штучні нейронні мережі(ANN) **20**

2.1.2 Повторювана нейронна мережа (RNN) **22**

2.1.3 Згорточна нейронна мережа (CNN) **24**

2.2. Порівняння існуючих моделей CNN нейромережі. **26**

РОЗДІЛ 3. Проектування і розробка програми. **30**

3.1 Вибір мови розробки та платформи розробки **30**

ВИСНОВКИ **40**

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **41**

ДОДАТОК А **44**

ДОДАТОК Б **46**

ДОДАТОК В **48**

ДОДАТОК Г **49**

ДОДАТОК Д **51**

ВСТУП

Технологічний прогрес не стоїть на місці і кожного року ми все більше залежимо від інформаційних систем. Сьогодні важко уявити повсякденне життя, робочі процеси, комунікації та виробництво без інформаційних технологій, оскільки автоматизація вже давно стала одним з фундаментальних стовпів сучасного суспільства. Різноманітні програми, веб-сайти, соціальні мережі та інші інструменти стали невід'ємними складовими нашого життя, спрощуючих задоволення наших щоденних потреб і вирішення проблем.

Одним з основних привілеїв які нам дає світ інформаційних технологій є оптимізація, делегування процесів та інкапсюляція їх від користувача. Завдяки зручним інтерфейсам та алгоритмам автоматизації, прості алгоритми поступаються місцем складнішим рішенням. Зараз програми можуть не лише оптимізувати процеси доставки та вибору товарів, але й допомагати нам здійснювати вибір, ґрунтуючись на наших попередніх покупках. Камери спостереження можуть ідентифікувати осіб, а програми обробки тексту не тільки виправляти помилки, але й генерувати унікальні тексти на задану тему. Такі можливості надають передові технології, такі як машинне навчання та нейронні мережі.

У сучасному світі швидкого розвитку технологій та зростаючої ролі штучного інтелекту, нейронні мережі стали ключовим інструментом у вирішенні різноманітних завдань

Вони широко використовуються в різних сферах основні з яких: економіка і бізнес, медицина авіоніка, автоматизація виробництв, системах безпеки та охорони, виконуючи ряд завдань, таких як розпізнавання, оброблення, трансформація даних та прогнозування змін на основі наявних даних.

Одним з найпоширеніших застосувань нейронних мереж є розпізнавання зображень та машинний зір, коли нейромережа може швидко опрацьовувати великі обсяги даних та виявляти задані об'єкти. Цей підхід відкриває широкий спектр можливостей, від відстеження порушень за камерами спостереження в реальному часі до аналізу якості виробництва.

Тож, дипломна робота зосереджується на розробці саме такого типу нейромережі і саме - нейромережі для розпізнавання маски на обличчі людини. З огляду на недавні світові події, такі як пандемія COVID-19, використання захисних масок стало обов'язковим елементом повсякденного життя для забезпечення безпеки населення. Крім того, розробка ефективних систем розпізнавання масок може мати широке застосування у різних галузях, включаючи медицину, безпеку, та робототехніку.

Отже, метою цієї роботи є створення моделі, здатної швидко та ефективно визначати наявність маски на людині, що може полегшити контроль та підтримку дотримання заходів безпеки.

Актуальність дослідження обумовлюється постійною потребою модернізації способів швидкого розпізнавання образів для автоматизації процесів верифікації.
У зв'язку з пандемічною ситуацією останніх років, застосування такого додатку набуває особливої актуальності.

Об'єкт дослідження нейромережі як технологія реалізації систем для розпізнавання образів.

Предмет дослідження технології реалізації нейронних мереж.

Мета і завдання дослідження спроектувати та реалізувати систему для розпізнавання образів, а саме медичної маски на обличчі людини.

Для реалізації поставленої мети були визначені наступні завдання:

1. Виконати аналіз теоретичних основ використання нейромереж у різних сферах.
2. Проаналізувати існуючі типи нейронних мереж та обрати найбільш релевантну.
3. Оглянути архітектури нейромереж та визначити найбільш релевантну для реалізації поставленої мети.
4. Розглянути інструменти та мови програмування для розробки нейромереж.
5. Розробити базову модель нейромережі та оцінити її точність на різних наборах даних.
6. Зробити висновки про виконану роботу.

Методи дослідження: загальнонаукові методи дослідженяня а саме: постереження, порівняння, рахунок, вимірювання, експеримент, узагальнення, абстрагування, формалізація,

Практичне значення одержаних результатів

У результаті роботи була отримана навчена нейносеть з точністю ~96.24%.

Ця нейромережа надалі може бути використана як частина системи для автоматизації розпізнавання наявності масок на обличчі людини в реальному часі на місцях контролю.

Наукова новизна одержаних результатів

Підсумкова архітектура була написана на основі tf-efficientetv2_I_in21K, але для кращої її роботи як функцію активації було використано

функцію SoftMax

Були написані кастомні фільтри, які в процесі випробування показали кращий результат, ніж вбудовані стандартні фільтри (Див. Додаток А). У результаті аналізу різних конфігурацій нейромережі було визначено оптимальну кількість шарів навчання і кількість нейронів на кожному з них. Завдяки цим змінам підсумкова архітектура дала результати краще, ніж у стандартної архітектури ~93.4% до ~96.24%.

РОЗДІЛ 1. Теоретичні основи використання нейромереж у різних сферах.

1.1 Поняття нейромережі

Нейронна мережа - це система штучного інтелекту, яка імітує спосіб обробки даних людським мозком. Вона є варіантом машинного навчання, відомим як глибоке навчання, та використовує міжзв'язані елементи, зазвичай називаються нейронами, розташовані в багатошаровій структурі, що співставляється з архітектурою мозку. Нейромережі розвиваються через адаптивний процес, де комп'ютерні системи навчаються на основі своїх помилок і постійно поліпшують свою роботу[1].

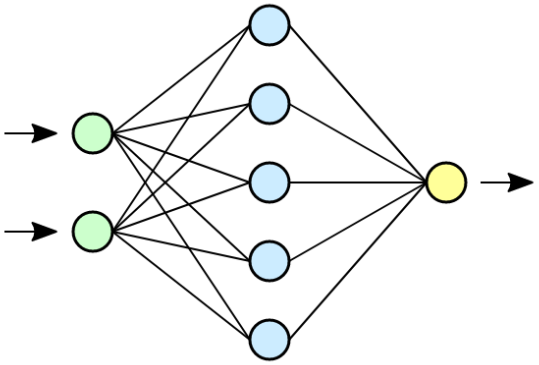


Рис 1.1.1

1.2 Історія та тенденції розвитку нейромереж

1.2.2 Історія розвитку нейромереж

[1] [7][24]

Розвиток нейромереж почався з досліджень нейробіології та сприйняття мозку людини в 1940-х роках. У 1943 році Варрен Маккалок та Уолтер Піттс вперше запропонували математичну модель нейрона, що стала основою для розвитку нейромереж .

У 1958 році Френк Розенблатт розробив персептрон - просту одношарову нейронну мережу, яка могла навчатися за допомогою алгоритму зворотного розповсюдження. Однак, через обмеження одношарових нейромереж, їхнє використання у вирішенні складних завдань було обмеженим.

У 1980-х роках науковці розробили нові алгоритми навчання та архітектури нейромереж, які значно покращили їхню ефективність. Завдяки роботам Румельхарта, Хінтона та Вільямса було розроблено алгоритм зворотного розповсюдження помилки (backpropagation), який дозволив багатошаровим нейромережам ефективно навчатися.

У 90-х роках XX століття активізувався розвиток нейронних мереж за рахунок покращення алгоритмів навчання і збільшення обчислювальної потужності комп'ютерів. Однією з ключових технологій, що з'явилася у цей період, була методика зворотного розповсюдження помилки (Backpropagation), яка значно підвищила ефективність навчання багатошарових персептронів (MLP).

У 1998 році Ян Лекун та його співавтори представили архітектуру згорткових нейронних мереж (CNN), яка була створена спеціально для розпізнавання образів. Вони стали основою для більшості сучасних систем розпізнавання образів, таких як розпізнавання облич, рукописного тексту та сцен.[2]

[2] [22,23]

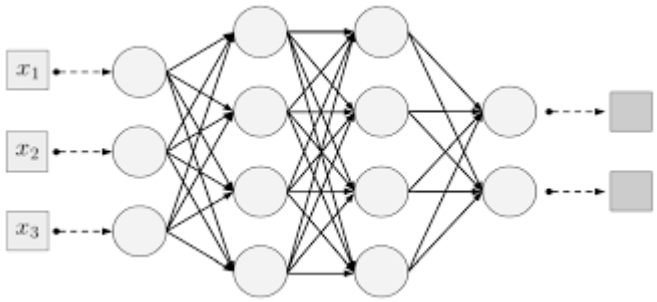


Рис 1.1.2

Починаючи з 2000-х років, вчені почали активно досліджувати рекурентні нейронні мережі (RNN) для обробки послідовностей та часових рядів. Однією з найвідоміших архітектур RNN є довгострокова пам'ять (LSTM), запропонована Хохрайтером та Шмідхубером в 1997 році. LSTM забезпечила рішення проблеми затухання градієнта, яка обмежувала ефективність навчання традиційних RNN.

В останні роки однією з найбільш революційних архітектур стали трансформери, вперше представлені у 2017 році в статті "Attention is All You Need". Трансформери використовують механізм уваги для встановлення зв'язків між елементами вхідної послідовності, що дозволяє досягти кращої роботи у задачах

Нижче наведено таблицю перерахування основних моделей нейромереж, які показували кращий результат для свого часу і зробили важливий внесок у розвиток нейроніх мереж.

Назва	**Рік**	**Точність**	**Опис**
LeNet-5	1998	98.2% на MNIST[див. додаток В]	Перша успішна нейромережа для розпізнавання рукописних цифр.

Sparse Coding	1999	~99.1% на MNIST	Одна з перших моделей глибокого навчання, що використовувала нелінійність для ефективного кодування даних.
Echo State Network	2001	N/A	Перша нейромережа з випадковим зв'язком, що використовувалася для моделювання динамічних систем
Deep Belief Network	2006	99.3% на MNIST	Нейромережа, що перша використовувала передтренування для покращення стабільності та точності моделі.
AlexNet	2012	63.3%	Модель, яка виграла конкурс ImageNet 2012 та спричинила популярність глибокого навчання.
ZFNet	2013	62.5%	Виграла конкурс ImageNet 2013 та була модифікацією архітектури AlexNet.
VGG-1	2014	74.4%	Відрізнялася дуже глибокою архітектурою, включаючи 16 шарів змішування.
DenseNet	2016	77.3%	Використовує щільні з'єднання між шарами для поліпшення градієнтного потоку.
NASNet	2017	82.7%	Розроблена за допомогою алгоритмів “пошуку” нейромережі.
SENet	2017	83.1%	Вводить "селективні блоки зв'язку"

EfficientNet	2019	84.4%	Масштабується на рівні шарів, глибини та розміру зображень для покращення точності та ефективност
TResNet	2020	84.6%	Оптимізована версія ResNet, яка включає швидші шари та усунення плями.
ViT-Large	2021	87.1%	Різке маштабування кількості параметрів збільшило точність.[3]

Табл 1.1

1.2.2 Тенденції розвитку нейромереж

За останні десятиліття, нейромережі пройшли шлях від простих одношарових структур до глибоких архітектур зі складними функціями активації та оптимізації. Такі глибокі нейромережі (DNN) показали вражаючі результати у вирішенні завдань класифікації зображень, розпізнавання мови та багатьох інших сферах[4].

З розвитком обчислювальної потужності та збільшенням обсягів даних, створюються нові архітектури нейромеромереж, такі як згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та трансформери[5].

Згорткові нейронні мережі стали особливо популярними у задачах комп'ютерного зору, таких як класифікація та сегментація зображень. Робота Яна ЛеКуна над LeNet-5 в 1990-х роках стала основою для

[3] [2], [21]
[4] [11]
[5] [12]

розвитку сучасних архітектур CNN, таких як AlexNet, VGG, ResNet та інші.

Рекурентні нейронні мережі розроблені для роботи з послідовними даними, такими як текст чи часові ряди, і використовуються для задач розпізнавання мови, машинного перекладу та аналізу емоцій. LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit) - це дві популярні архітектури RNN, які допомагають вирішувати проблему затухання градієнту[6].

Трансформери, запропоновані Васвані та іншими у 2017 році, використовують механізм уваги (attention) для паралельної обробки послідовних даних та забезпечують більш ефективне навчання глибоких моделей (8). Трансформери швидко стали основою для розвитку сучасних масштабованих моделей NLP, таких як BERT, GPT та T5.

Перспективи розвитку у технічному аспекті

У майбутньому розвиток нейромереж та глибокого навчання може слідувати декільком важливим напрямкам. Ось деякі перспективні то можливі тенденції:

Ефективність: Швидкість та енергоефективність нейромереж є важливими аспектами для розгортання моделей на різних пристроях, особливо на мобільних та вбудованих системах. В майбутньому розробники можуть зосередитися на створенні компактних моделей, які мають високу точність та низьке споживання енергії.

Інтерпретація та пояснення: Навіть найкращі нейромережі можуть бути складними для інтерпретації. Це може призвести до проблем з

[6] [13]

довірою та нездатністю зрозуміти, як модель приймає рішення. У майбутньому дослідники можуть розробляти методи для створення інтерпретованих моделей, які можуть пояснювати свої рішення.

Масштабованість та автоматизація: У майбутньому розробка нейромереж може стати ще більш автоматизованою та масштабованою. Це означає, що алгоритми та інструменти можуть створювати моделі глибокого навчання на основі великих наборів даних без значної людської втручання. Використання автоматичного машинного навчання (AutoML) та нейромережевого архітектурного пошуку (NAS) може сприяти реалізації цього напрямку.

Трансферне навчання та метанавчання: Трансферне навчання дозволяє моделям використовувати знання, отримані з одного завдання, для рішення інших завдань. У майбутньому трансферне навчання може відігравати важливу роль у створенні більш загальних моделей нейромереж, які можуть легко адаптуватися до нових задач. Метанавчання, процес, який спрямований на вивчення алгоритмів навчання, також може стати важливим елементом розвитку нейромереж. Це може допомогти розробити моделі, які можуть швидко навчитися з малим обсягом даних.[7]

Перспективи розвитку у аспекті використання

Медицина: нейромережі допомагають у діагностиці захворювань, аналізі медичних зображень, та розробці індивідуальних планів лікування для пацієнтів.

Автономні транспортні засоби: нейромережі використовуються для управління автономними транспортними засобами, такими як безпілотні автомобілі та дрони.

[7] [20]

Розпізнавання мови: поліпшення систем розпізнавання та синтезу мови, що дозволяє розуміти та генерувати природну мову.

Генеративні моделі: створення нових зображень, відео, музики або тексту, використовуючи генеративні нейромережі.

Біотехнологія: використання нейромережі для передбачення структури білків, проектування лікарських препаратів та оптимізації мікроорганізмів для промислових застосувань.

Фінанси: аналіз фінансових даних, передбачення трендів ринку та автоматизація процесів управління портфелем за допомогою нейромереж

Кібербезпека: використання нейромережі для виявлення аномалій, зловмисних програм та атак на комп'ютерні системи.[8]

1.3 Сфери використання нейромереж у наш час

Нейромережі в різних предметних галузях зазвичай використовують для виконання наступних завдань:

Класифікація. ІНС визначає, чи відповідає аналізований об'єкт заданим параметрам, і відносить його до тієї чи іншої групи.

Прогнозування. На основі вивчення вхідних даних ІНС передбачає поведінку досліджуваної об'єктної області в майбутньому

Розпізнавання. можливість ІНС виділяти об'єкт серед безлічі подібних.

У багатьох предметних галузях при найближчому розгляді можна знайти постановки задач для нейронних мереж

[8] [16], [17]

Схема 1.2.1

Економіка і бізнес: прогнозування часових рядів (курсів валют, цін на сировину, попиту, обсягів продажів), автоматичний трейдинг (торгівля на валютній, фондовій або товарній біржі)

Медицина та охорона здоров'я: постановка діагнозу хворому (діагностика захворювань), обробка і розпізнавання медичних зображень (рентгенівських знімків, томограм і т.д.)

Біотехнології: Нейромережі застосовуються для аналізу біологічних даних, таких як генетичні послідовності та структури білків, що сприяє розвитку нових ліків та терапій.

Авіоніка: автопілоти, що навчаються, розпізнавання сигналів радарів, адаптивне пілотування сильно пошкодженого літака, безпілотні літальні апарати (дрони), розпізнавання/детекція об'єктів на фото/відеозйомці з дрона.

Автоматизація виробництва: оптимізація режимів виробничого процесу, контроль якості продукції, моніторинг і візуалізація

багатовимірної диспетчерської інформації, попередження аварійних ситуацій.

Робототехніка: Нейромережі використовуються в робототехніці для навчання роботів виконувати складні задачі, такі як маніпуляція предметами або навігація.

Безпека, охоронні системи: розпізнавання облич; ідентифікація особи за відбитками пальців, голосом або підписом; розпізнавання автомобільних номерів.

Введення та обробка інформації: розпізнавання рукописних текстів, відсканованих поштових, платіжних, фінансових і бухгалтерських документів; розпізнавання мовних команд, мовне введення тексту в комп'ютер.

Геологорозвідка: аналіз сейсмічних даних, асоціативні методики пошуку корисних копалин, оцінка ресурсів родовищ.

Ігри: нейромережі використовуються для створення штучного інтелекту в іграх, що може адаптуватися до стратегій гравця та навчатися з досвіду.

Генерація музики: нейромережі використовуються для компонування музики, створюючи нові мелодії та аранжування на основі відомих музичних тв.

Прогнозування погоди: нейромережі допомагають у прогнозуванні погоди, аналізуючи великі масиви даних з різних джерел, що дозволяє робити точніші прогнози.

За прикладним функціональним застосуванням можна виділити такі способи використання:

- Системи розпізнавання та впорядкування об'єктів на зображеннях;
- Голосові інтерфейси взаємодії для IoT-пристроїв;
- Системи контролю якості обслуговування в кол-центрах;

- Системи виявлення несправностей (зокрема, прогнозування технічного обслуговування), відхилень та кіберфізичних загроз;
- Інтелектуальні системи безпеки та моніторингу;
- Системи аналізу відеоданих;
- Самонавчальні системи оптимізації управління матеріальними потоками та розміщення об'єктів (на складах, у транспорті);
- Розумні, самонавчальні системи управління виробничими процесами та пристроями (зокрема, робототехнічними);
- Системи універсального миттєвого перекладу для конференцій та особистого користування;
- Чат-боти технічної підтримки або персональні асистенти, наближені за функціональністю до людини.[9]

Отже, можна стверджувати, що в сучасному світі нейромережі відіграють важливу роль у багатьох різноманітних сферах, сприяючи автоматизації складних процесів аналізу даних, виявленню закономірностей, прогнозуванню ринкових тенденцій та розробці складних інтерфейсів.

Тенденції розвитку нейромереж свідчать про те, що їх значення продовжуватиме зростати з кожним роком. Зокрема, нейромережі будуть все активніше застосовуватися в нових областях, а у сферах, де вони вже використовуються, роль цього буде зростати.

[9] [3],[7]

РОЗДІЛ 2. Аналіз існуючих видів і архітектур нейромереж.

2.1 Аналіз наявних видів нейромереж

Архітектура базової нейронної мережі

Базова нейронна мережа містить три шари взаємопов'язаних штучних нейронів:

Вхідний шар: цей шар приймає дані зі зовнішнього джерела та обробляє їх для подальшого аналізу або класифікації. Після обробки дані передаються наступному шару.

Прихований шар: цей шар отримує вхідні дані від попереднього шару або інших прихованих шарів. Нейромережі можуть мати багато прихованих шарів. Кожен прихований шар аналізує вихідні дані попереднього шару, обробляє їх та передає наступному шару.

Вихідний шар: цей шар надає кінцевий результат обробки даних нейромережею. Він може мати один або декілька вузлів, в залежності від завдання.

Архітектура глибокої нейронної мережі

Глибокі нейромережі, також відомі як мережі глибокого навчання, містять кілька прихованих шарів з мільйонами взаємопов'язаних штучних нейронів. Вага між вузлами визначає силу зв'язку між ними. Позитивна вага вказує на збудливий зв'язок, тоді як негативна вага вказує на інгібіторний зв'язок. Вузли з більш високими значеннями ваги мають більший вплив на інші вузли в мережі.

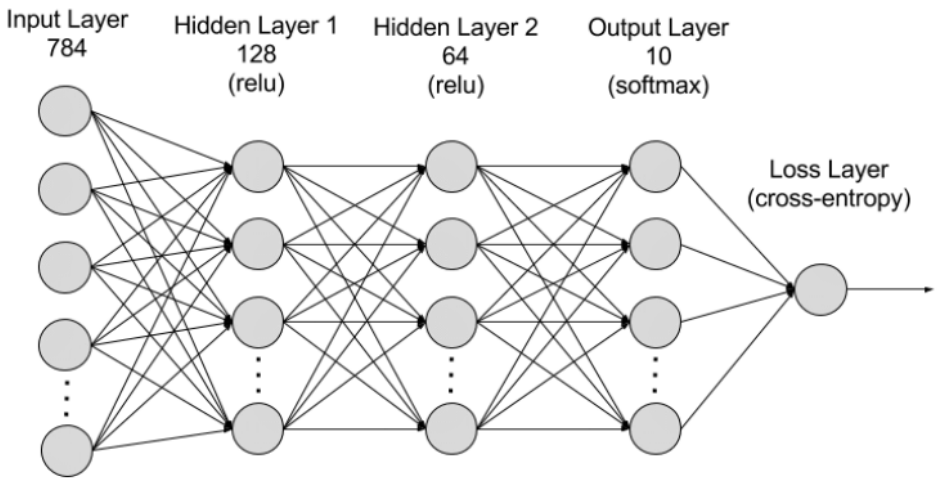


Схема 2.1.1

Додатково до архітектури фундаментальної нейромережі та глибокої нейромережі, існує велика кількість спеціалізованих архітектур нейромереж, кожна з яких розроблена для різних завдань або проблем. Ось основні з них:

Адаптивні нейромережі (ANN) відрізняються своєю здатністю адаптуватися та самоорганізуватися під час навчання. Вони автоматично оптимізують свою структуру та параметри для покращення точності передбачень або класифікації.

Конволюційні нейромережі (CNN): ці мережі розроблені спеціально для обробки зображень, аудіо та інших сігналів, що мають просторову структуру. Вони включають конволюційні шари, які допомагають мережі автоматично виявляти значимі ознаки з даних.

Рекурентні нейромережі (RNN): ці мережі призначені для обробки послідовностей даних, таких як часові ряди або текст. Вони включають рекурентні шари, які дозволяють мережі зберігати інформацію про попередні кроки обробки для використання в майбутніх кроках.

Спільні нейромережі (SNN): ці мережі використовуються для порівняння двох або більше вхідних даних, наприклад для визначення схожості між зображеннями або текстами. Вони містять спільні шари,

які дозволяють мережі ефективно обробляти та порівнювати вхідні дані.[10]

Автоенкодери (AE): ці мережі призначені для стиснення даних або виявлення структури в даних. Вони складаються з двох частин: енкодера, який стискає вхідні дані, та декодера, який реконструює дані зі стиснутого представлення

Генеративно-змагальні мережі (GANs): ці мережі використовуються для генерації нових даних, які схожі на дані навчання. Вони складаються з двох мереж, генератора та дискримінатора. Генератор створює нові дані, а дискримінатор намагається визначити, чи є сгенеровані дані справжніми або штучними. Протягом процесу навчання генератор поступово покращує свою здатність створювати реалістичні зображення, в той час як дискримінатор навчається краще відрізняти справжні дані від штучних. Це змагання між генератором та дискримінатором стимулює покращення обох мереж, що призводить до створення дедалі більш реалістичних зображень[11]

За визначенням **Автоенкодери** та **Генеративно-змагальні мережі** є допоміжними та не підходять для реалізації поставленої мети.

Спільні нейромережі (SNN): мають інший вид вхідних даних, що теж не підходить у нашому випадку.

Розглянемо особливості ANN, CNN, RNN і підберемо найбільше оптимальне для вирішення задачі розпізнавання об'єктів на зображенні.

2.1.1 Штучні нейронні мережі(ANN)

Нейромережі прямого розповсюдження обробляють інформацію, направляючи її в одному напрямку від входів до виходів. Всі вузли

[10] [7]
[11] [25]

одного шару пов'язані з вузлами наступного шару. Такі нейромережі використовують зворотний зв'язок для покращення прогнозів з часом.[12]

Принцип роботи

ANN складається з 3 рівнів - входів, прихованих вузлів та виходів. Вхідний рівень приймає дані, прихований рівень обробляє їх, а вихідний рівень видає результат. Кожен шар намагається навчитися певним вагам.

Штучні нейронні мережі можуть навчитися будь-якій нелінійній функції, тому їх часто називають універсальними апроксиматорами функцій. ANN вивчає ваги, що відображають будь-які вхідні та вихідні дані. Однією з основних причин універсальної апроксимації є активаційні функції, які надають мережі нелінійних властивостей, дозволяючи вивчити будь-які складні зв'язки між входами та виходами.

Однією з головних причин універсальної апроксимації є функція активації. Функції активації надають мережі нелінійні властивості. Це допомагає мережі вивчати будь-який складний зв’язок між входом і виходом.

Проблеми використання

Оптимізаційна проблема: при вирішенні задачі класифікації зображень за допомогою ANN, спочатку 2D-зображення перетворюється на 1D-вектор перед тренуванням моделі. Це має два недоліки: Кількість параметрів, які можуть бути навчені, стрімко зростає зі збільшенням розміру зображення. Наприклад, при розмірі зображення 224x224, кількість параметрів, які можна навчити на першому прихованому шарі з лише 4 нейронами, становить 602 112, що є надзвичайно великою

[12] [5]

кількістю. Також ANN втрачає просторові характеристики зображення, які стосуються розташування пікселів на зображенні.

Іншою поширеною проблемою є зникаючий та вибухаючий градієнт. Ця проблема пов'язана з алгоритмом зворотного розповсюдження. Ваги нейронної мережі оновлюються за допомогою цього алгоритму зворотного розповсюдження шляхом визначення градієнтів.

Таким чином, у випадку дуже глибоких нейронних мереж (мережі з великою кількістю прихованих шарів) градієнт може зникнути або вибухнути, коли розповсюджується назад, що призводить до зникнення або вибуху градієнта.

Отже, у випадку дуже глибокої нейронної мережі (мережа з великою кількістю прихованих шарів) градієнт зникає або вибухає, коли він поширюється назад, що призводить до зникнення та вибуху градієнта.

2.1.2 Повторювана нейронна мережа (RNN)

RNN - принцип дії алгоритму схожий на ANN, проте має періодичне з'єднання в прихованому стані. Це обмеження циклу гарантує, що послідовна інформація фіксується у вхідних даних.

Тобто якщо порівняти RNN та ANN схематично буде такий вигляд:[13]

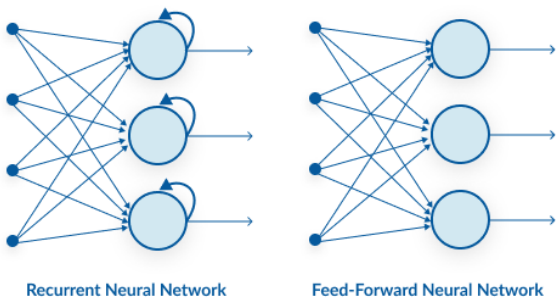


Рис. 2.2.1

[13] [4]

Цей підхід має ряд переваг перед ANN

RNN розпізнає послідовність у вхідних даних, враховуючи залежності між словами у тексті під час прогнозування:

RNN використовують спільні параметри для різних часових інтервалів. Тобто має спільне використання параметрів. Це зменшує кількість параметрів для навчання та знижує обчислювальні витрати.

Проблеми використання:

Глибокі RNN (RNN з великою кількістю часових інтервалів) також зіткнулися з проблемою зникнення та вибухового градієнта, яка є поширеною проблемою для багатьох видів нейронних мереж.

2.1.3 Згорточна нейронна мережа (CNN)

Будівельними блоками CNN є фільтри, вони ж ядра. Ядра використовуються для отримання відповідних функцій із вхідних даних за допомогою операції згортки.

Переваги згорткової нейронної мережі (CNN)

CNN вивчає фільтри автоматично, не згадуючи це явно. Ці фільтри допомагають витягувати правильні та релевантні функції з вхідних даних

CNN фіксує просторові характеристики із зображення. Просторові характеристики стосуються розташування пікселів і зв'язку між ними на зображенні. Вони допомагають нам точно ідентифікувати об'єкт, розташування об'єкта, а також його зв'язок з іншими об'єктами на зображенні.

CNN також дотримується концепції спільного використання параметрів. Один фільтр застосовується до різних частин вхідних даних для створення карти об'єктів

В підсумку порівняльна характеристика цих типів нейромереж наступна:[14]

Параметр	**ANN**	**RNN**	**CNN**	**AE**	**SNN**	**GAN**
Оброблювані дані	Зображення, текст, сигнали	Послідовності даних, текст, часові ряди	Зображення, аудіо, сигнали	Зображення, текст, сигнали	Зображення, текст	Зображення, текст, сигнали(генерація)
Повторювані з'єднання	Ні	Так	Ні	Ні	Залежить від архітектури	Залежить від архітектури

[14] [7]

Спільний доступ до параметрів	Ні	Так	Так	Так	Так	Так
Просторове відношення	Ні	Ні	Так	Ні	Залежить від архітектури	Залежить від архітектури
Зникаючий вибуховий градієнт	Так	Так	Так	Ні	Ні	Ні
Швидкість навчання	Середня	Низька	Висока	Низька	Середня	Дуже висока
Точність розпізнавання зображень	Відносно низька	Середня	Висока	-	-	Генерація нових[15]

Табл 2.1.1

Отже за результатами порівнння можна зродити висновок, що для розробки додатку, краще використовувати конволюційні нейронні мережі (CNN). Причини цього такі:

CNN ефективно обробляють зображення, оскільки вони розуміють просторові відношення між пікселями і можуть вивчити риси на різних рівнях абстракції.

Спільний доступ до параметрів дозволяє CNN навчатися від меншої кількості даних, а також забезпечує меншу кількість параметрів, що сприяє швидкій оптимізації та зменшенню ризику перенавчання.

Конволюційні мережі добре справляються з варіаціями в позиції, орієнтації та освітленні об'єктів на зображенні, що є важливим для розпізнавання маски на обличчі людини.

[15] [27][7]

2.2. Порівняння існуючих моделей CNN нейромережі.

Серед CNN нейромереж зараз такі основні групи архитектур:

- EfficientNet: Ця модель - результат автоматичного збільшення мережі, який оптимізує ширину, глибину та роздільну здатність одночасно.
- ResNet: Ця архітектура нейромережі відома своїм застосуванням "шихти" для глибокого навчання, дозволяючи мережі легко навчатися навіть з десятками шарів.
- ResNeXt (реалізація ResNet): ResNeXt є поліпшеною версією ResNet, яка використовує структуру розділених та повторюваних груп з'єднань. ResNeXt надає кращу точність та ефективність для розпізнавання зображень.
- VGG: Ця архітектура нейромережі характеризується послідовними конволюційними шарами та дуже глибокими структурами. VGG забезпечує високу точність у задачах розпізнавання зображень, але може бути відносно повільною та ресурсомісткою.
- DenseNet: DenseNet відрізняється своїм зв'язком кожного шару з усіма іншими шарами після нього, що поліпшує потік інформації та градієнта.

Для порівняння як метрика якості використовувалася частка правильних відповідей (у відсотках) у топ1 (acc), і частка правильних відповідей у топ5 (acc5) найближчих векторів.

Як вхідні дані бралися кольорові зображення 224x224 пікселя. Якщо вхід у нейромережі відрізнявся від зазначеного дозволу, зображення попередньо масштабувалися до необхідного.

Група мереж VGG.

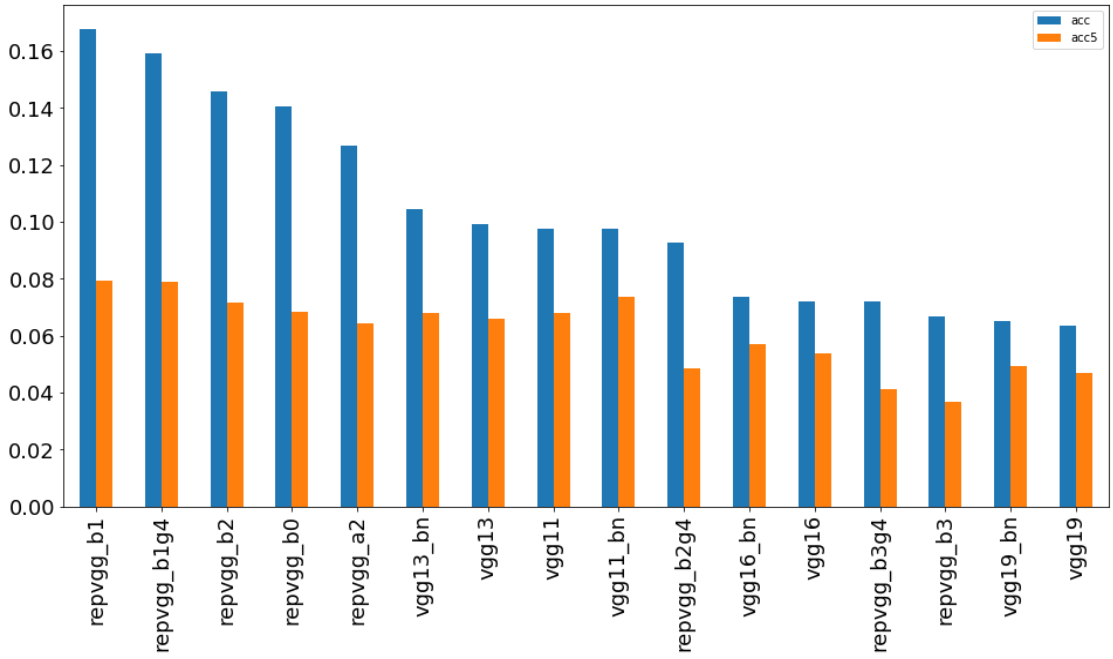


Рис 2.2.1

Група мереж EfficientNet

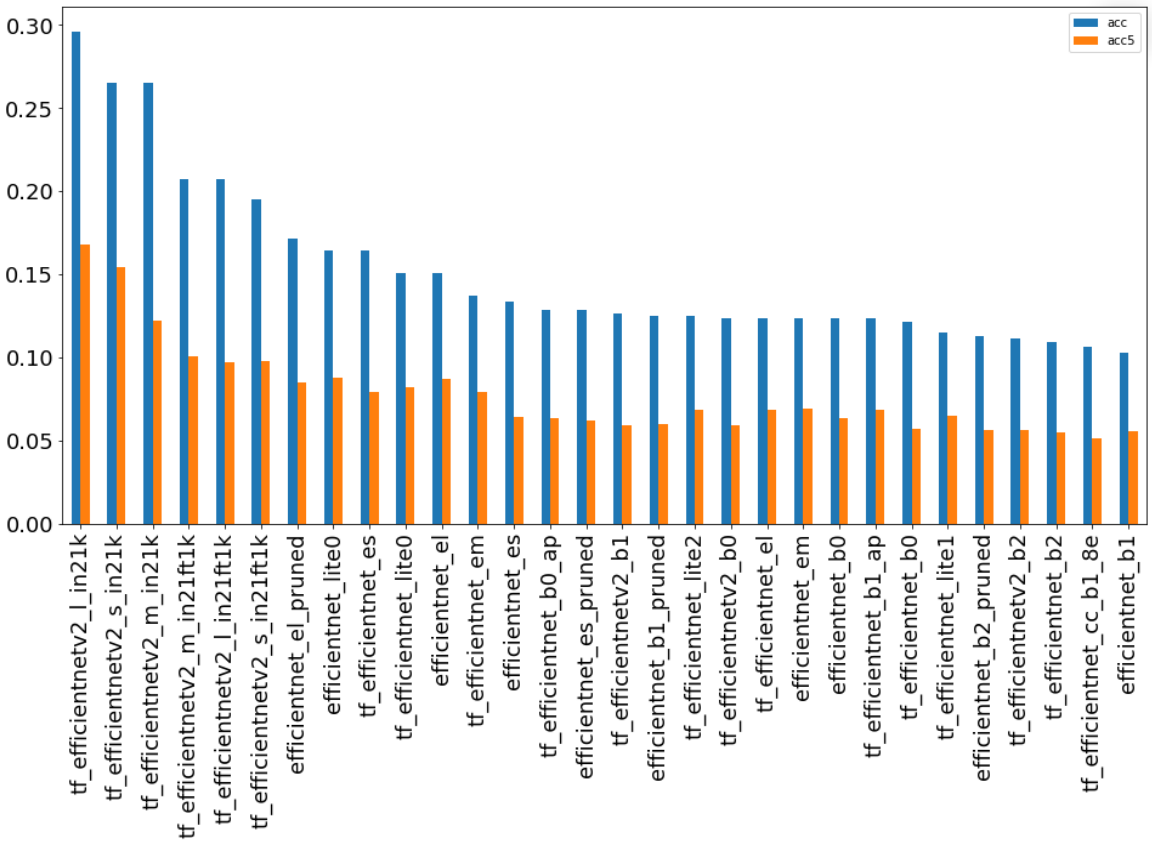


Рис 2.2.2

Група мереж ResNet.

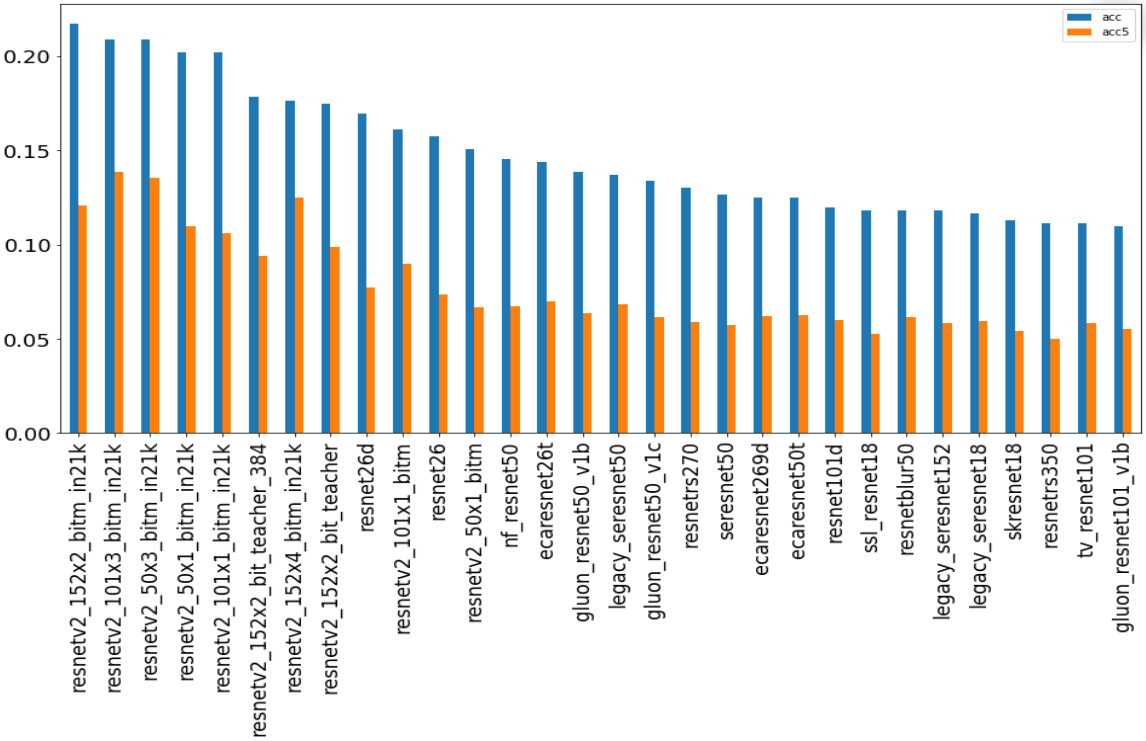


Рис 2.2.3

Группа сетей ResNeXt

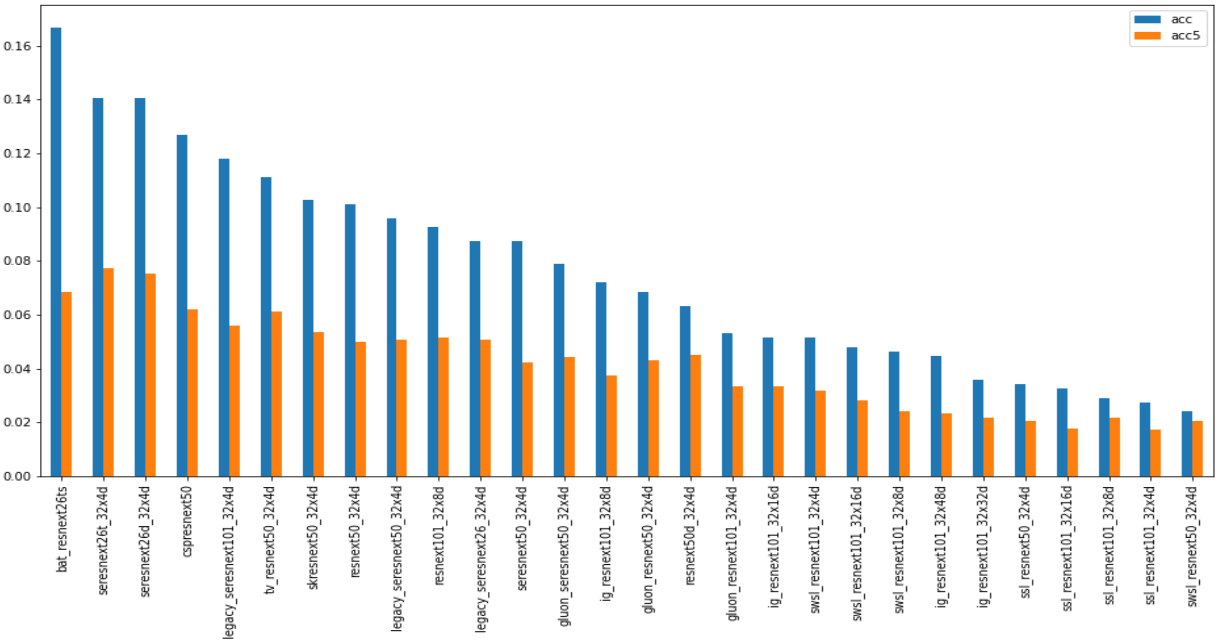


Рис 2.2.4

Група мереж DenseNet

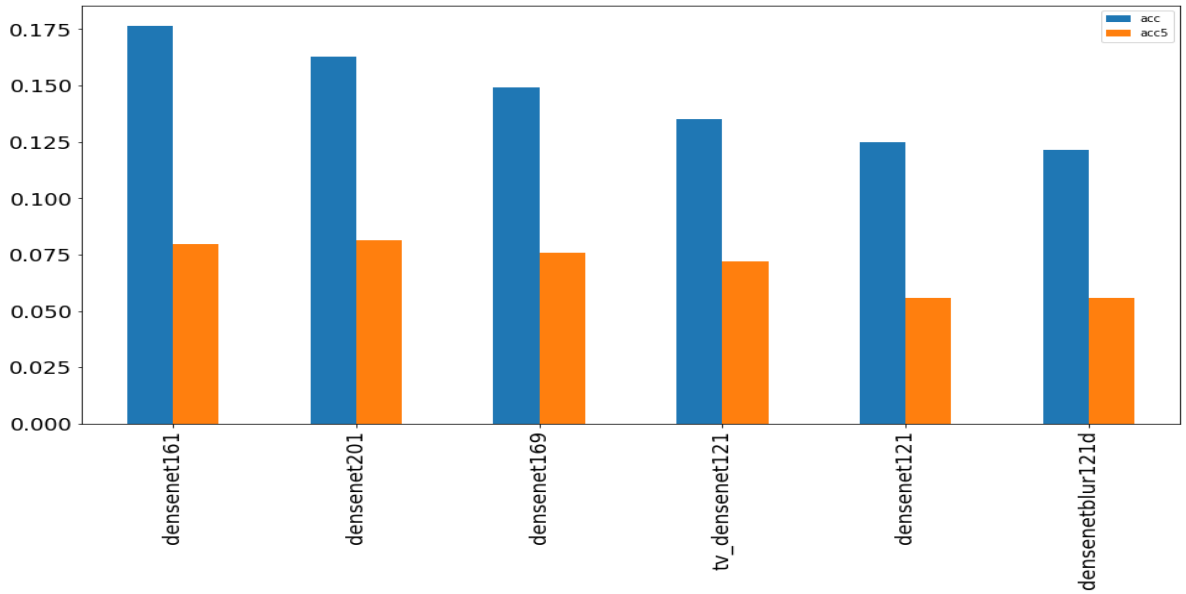


Рис 2.2.5

Отже, можна зробити висновок що для реалізації поставленого завдання найкраще підійде архітеткутра з типу EfficientNet, а саме tf-efficientetv2_I_in21K.

РОЗДІЛ 3. Проектування і розробка програми.

Створення додатку складалось з декількох етапів: постановки задачі, дослідження предметної області, постановки технічного завдання, аналізу інтсрументів та мов програмування, проектування додатку, написання коду, аналізу роботи програми, доопрацювання програми на основі тестування.

3.1 Вибір мови розробки та платформи розробки

Найпопулярнішими мовами для розроблення нейромереж на даний момент є Python, C#, C++, Js. Було проведено порівняльний аналіз розробки нейромереж на цих мовах за такими критеріями:

- **Спільнота** - оцінка популярності підходу у сфері, велике сом'юніті сприяють більш простому пошуку рішення проблеми
- **Швидкість розробки** - наявність зручних інструментів для розробки
- **Швидкість виконання**
- **Синтаксис –** складність ситнтаксису
- **Бібліотеки для нейромереж**
- **Портативність**[16]

Мова	**C#**	**Javascript**	**C++**	**Python**
Спільнота	Середня	Велика	Велика	Найбільша
Швидкість виконання	Висока	Середня	Висока	Висока
Швидкість розробкти	Середня	Середня	Низька	Висока

[16] [29]

Бібліотеки для нейромереж	Обиежені	Обмежені	Багато	Багато
Портативність	Висока	Висока	Висока	Висока
Синтаксис	Середня складність	Легкий	Складний	Легкий

Отже у зв'язку з тим, що нейромережа не вимагатиме високого ступеня оптимізації процесу навчання у зв'язку з невеликим набором даних, було прийнято рішення вибрати мову Python. З урахуванням порівняльної таблиці, Python є найкращим вибором. З легким синтаксисом, високою швидкістю розробки, великою кількістю бібліотек для машинного навчання та великою спільнотою.

Вибір бібліотеки для розробки нейромережі.

Є три основні бібіліотеки для розроблення нейромереж на Python: Keras Pytorch TensoFlow. Було проведено порівняльну харастектистику цих бібліотек за такими параметрами:

- **Рівень апі** - рівень інкапсуляції бібліотеки.
- **Обробка наборів даних** - які набори даних може обробляти бібліотека за розміром.
- **Налагодження та візуалізація** - рівень зручності внутрішніх інструментів налагодження та візуалізації під час роботи з даними.
- **Швидкість**.
- **Легкість використання**.
- **Динамічний граф обчислень** - це концепція в області нейромереж та глибокого навчання, в якій граф обчислень будується та змінюється в

режимі реального часу в процесі навчання. У такому випадку граф обчислень не фіксований та може адаптуватися в залежності від вхідних даних або змінних умов навчання.

- **Процес навчання** - Keras має змішаний (спрощений) процес навчання, що забезпечує швидке створення та навчання моделей, але з меншими налаштуваннями. PyTorch та TensorFlow пропонують гнучкі можливості навчання з більшою кількістю налаштувань.
- **Підтримка наукового python**.
- **Популярність**.

Параметр	**Keras**	**Pytorch**	**TensorFlow**
Рівень апі	високий	Середній	Середній
Обробка наборів данних	Середня	Великі	Великі
Швидкість	Середня	Висока	Висока
Налагодження та візуалізація	Висока	Середня	Низька
Легкість використання	Висока	Середня	Середня
Динамічний граф обчислень	Ні	Так	Так(З Tensor Flow 2.0)
Популярність	Середня	Висока	Висока
Підтримка наукового python	Частково	Повна	Частково
Процес навчання	Спрощена	Гнучка	Гнучка

У результаті аназизу було вирішено вибрати TensoFlow через можливість маштабованості в розмірі датасета, хорошу продуктивність і поплуярність цієї бібіліотеки.

Середовище розробки

В якості середовище для розроблення було обрано Jupyter, який спеціально розроблено для поліпшення зручності роботи з великими обсягами даних, код можна поділити на кластери(ноди), кожен з яких після виконання буде мемоізуватися, а під час перезапуску програми результати виконання лежатимуть у кеші пристрою, що дуже прискорить роботу з обробкою великої кількості даних.

3.2 Створення нейромережі

Створення нейромережі складається з трьох основних етапів:

- Збір, нормалізація та сортування даних для набору даних для навчання
- Написання архітектури нейронної мережі
- Навчання неронної мережі та подальше калібрування конфігурацій навчання

3.2.1 Збір даних для датасету

Збір та нормалізація даних для датасету є критично важливими кроками у процесі розробки нейромережі. Перед початком збору даних потрібно визначити кількість класів, на які будуть розбиті дані. У нашому випадку ми маємо два класи: особи з масками та без масок.

Під час створення датасету необхідно зібрати зображення з різноманітними характеристиками, такими як:

• Різні кути зйомки об'єкта

• Різні погодні умови

• Різне розташування об'єкта на зображенні

• Відмінні варіації об'єкта, наприклад кольори масок

Важливо також забезпечити рівномірний баланс між кількістю зображень кожного класу, таким чином, щоб кількість зображень без маски дорівнювала кількості зображень з маскою.

Після збору даних необхідно розділити датасет на дві частини: тренувальний датасет, який буде використовуватися для навчання нейромережі, та тестовий датасет, який буде застосовуватися для оцінки її точності та ефективності. Такий підхід дозволяє отримати більш достовірну оцінку роботи нейромережі на реальних даних.
Також для збільшення кількості зображень у датасеті було прийнято рішення написати Gans нейромережу для генерації асетів на основі існуючих(Див. Додаток Г)

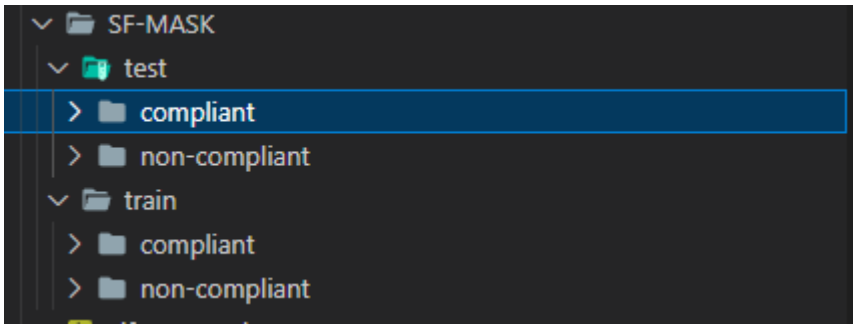


Рис 3.2.1

Таким чином до підсумкового датасету для навчання увійшли 4500 зображень без маски і така сама кількість із маскою. Датасет для тесту був 150.

3.2.2 Написання архітектури нейронної мережі

Кінцева архітектура була розроблена на основі архітектури tf-efficientnetv2_I_in21K, проте деякі параметри базової архітектури були модифіковані для поліпшення:

• Нейромережа мала багатокласову структуру, тому в якості активаційної функції використовувалася функція SoftMax.
• Застосовувались спеціальні фільтри, які під час експериментів продемонстрували кращі результати порівняно зі стандартними вбудованими фільтрами (детальніше див. Додаток А).
• В результаті дослідження різних конфігурацій нейромережі було встановлено оптимальну кількість навчальних шарів та кількість нейронів у кожному з них (детальніше див. Додаток Б).

3.2.3 Тестування нейронної мережі

У результаті навчання нейромережа пройшла 10 епох навчання і мала максимальну точність 96.24%.[17]

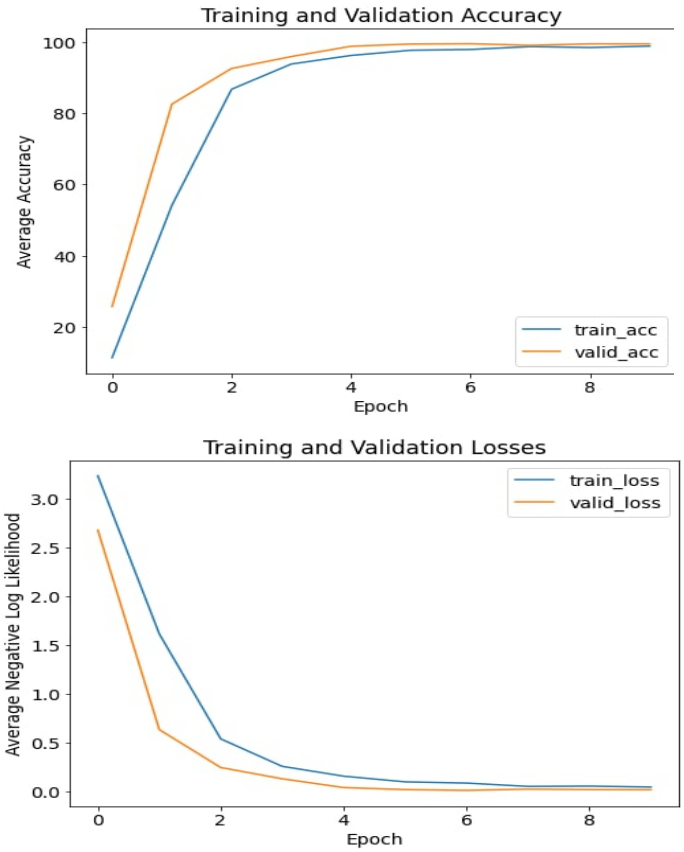
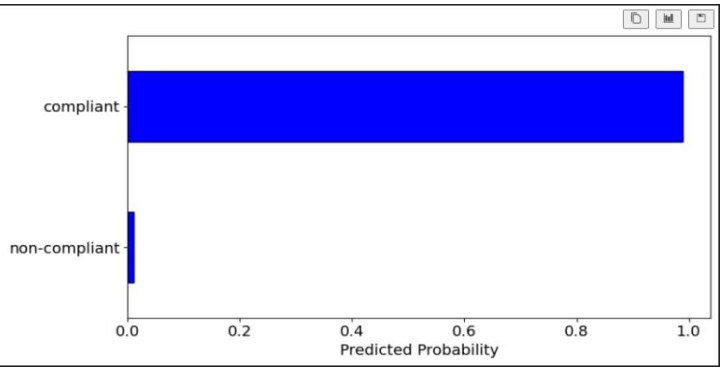


Схема 3.2.2

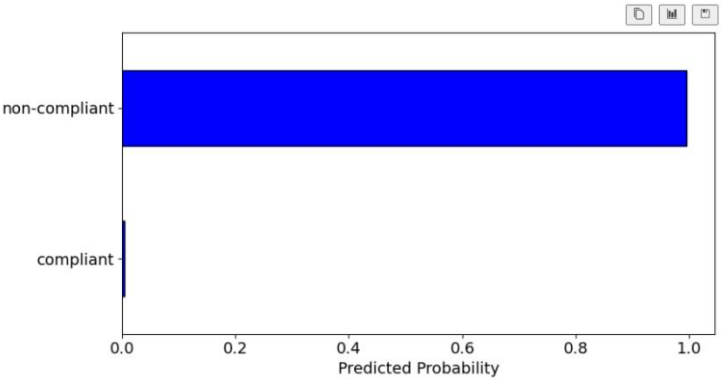
Так само було проведено ручне тестування нейромережі та зі 200 запропонованих варіантів нейромережа правильно визначила 197 з них. Приклади тестування наведено нижче:

[17] [31]

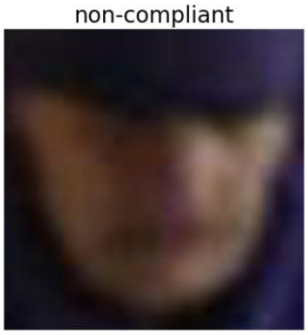
compliant

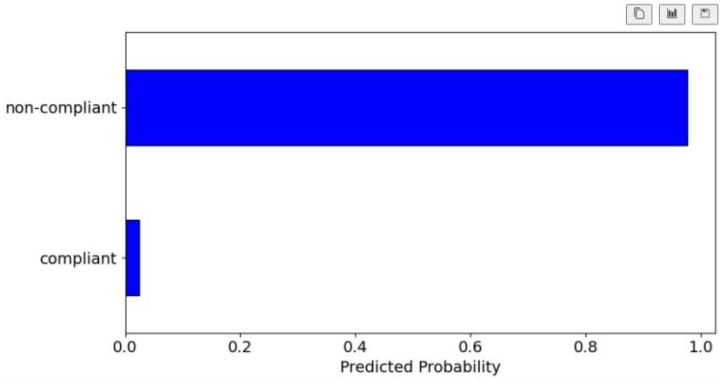


non-compliant

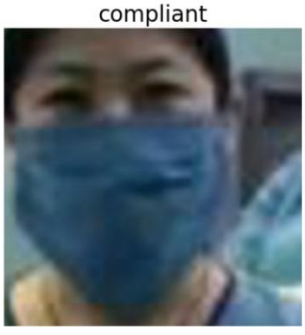


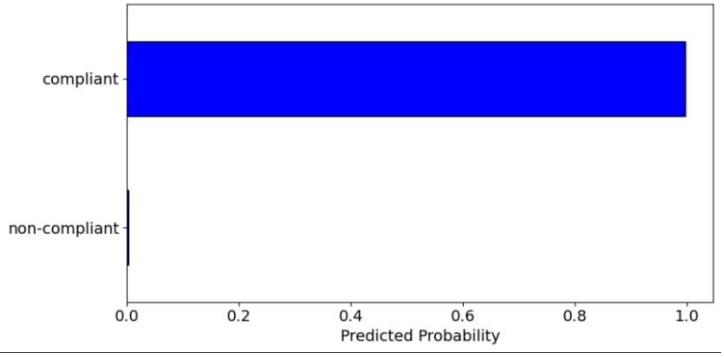
non-compliant

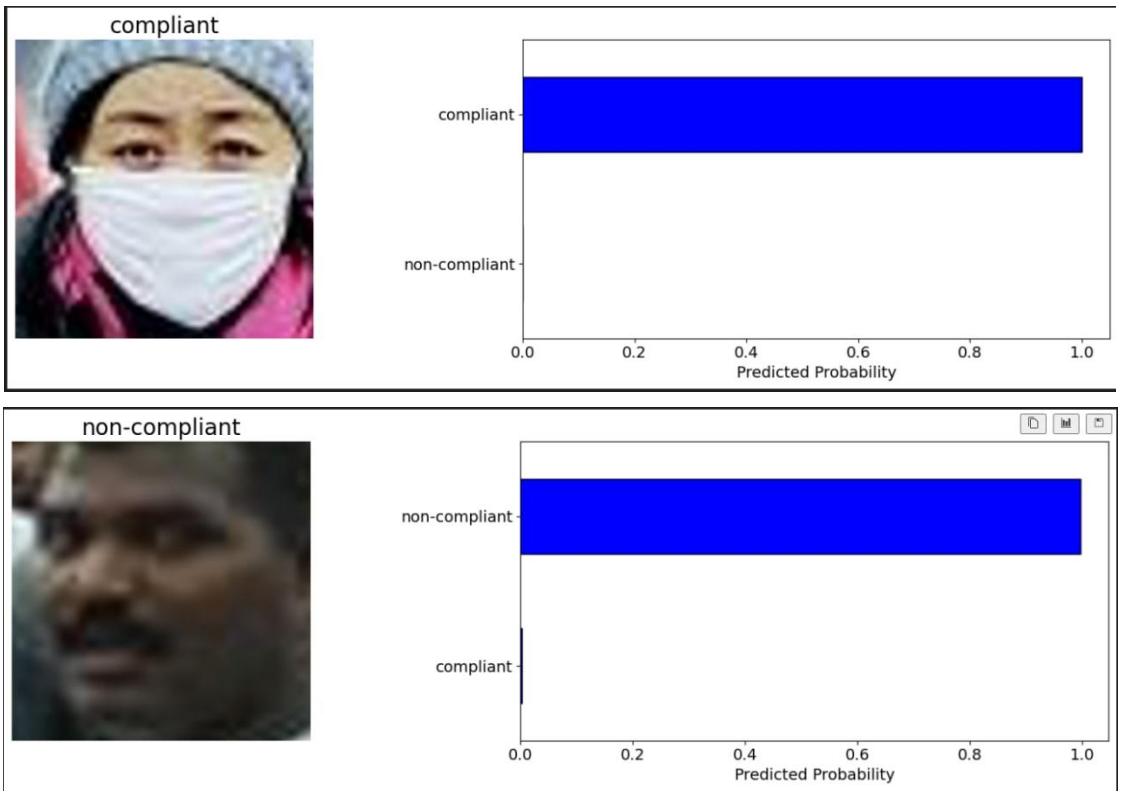




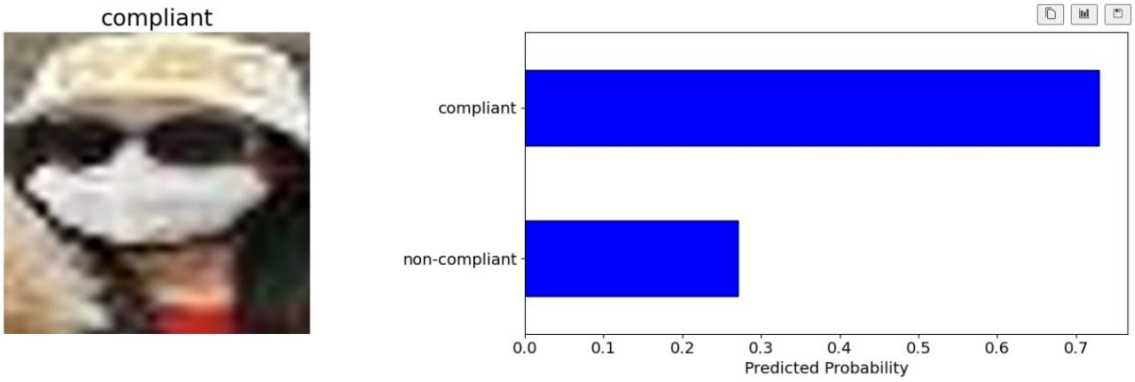
compliant

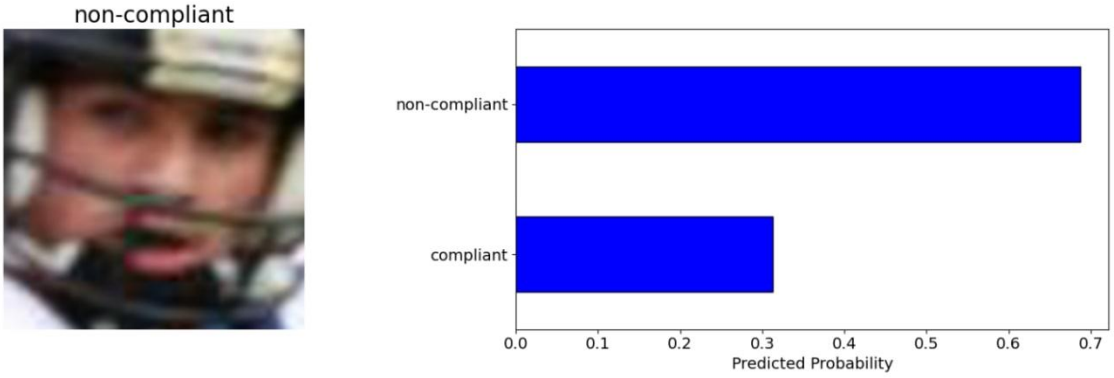






Найбільші труднощі у нейромережі виникають під час оцінювання людей, у яких є шоломи або головні убори, що закривають частину обличчя, але, як правило, нейромережа і в цьому разі правильно визначає наявність маски.





Таким чином у процесі розробки було написано та навчено нейромережу з точністю 96.24%. Було проведено тестування нейромережі з результатом 197/200 правильних відповідей. Було виявлено, що на даному етапі нейромережа найгірше справляється з визначенням макскі на людях, які мають головні убори, що закривають частину обличчя.

ВИСНОВКИ

В ході даної роботи було виконано наступні завдання:

- Виконано аналіз теоретичних основ використання нейромереж у різних сферах. Було зроблено висновок, що вже зараз нейромережі широко використовують у різних сферах для оптимізації різних складних процесів передбачення та іншого.
- Проаналізовано існуючі типи нейронних мереж та за результатами дослідження було з'ясовано, що найбільше для вирішення поставленого завдання підходить нейромережа CNN типу.
- Проаналізовано існуючи архітектури нейромереж та обрана найточніша tf-efficientetv2_I_in21K.
- Проаналізувати інструменти написання нейромереж у результаті аналізу було прийнято вести розробку на мові Python на бібліотеці TensoFlow в середовищі Jupyter .
- Написати модель нейромережі з точністю роботи 96.24% та проаналізувати її точність на різних наборах даних було виявлено основні складнощі нейромережі під час розпізнавання образів.

Отже, за результатами виконаної роботи всі поставлені завдання були виконані. Розбролений додаток має актуальність дослідження яка обумовлюється постійною потребою модернізації способів швидкого розпізнавання образів для автоматизації процесів верифікації.

Перспективи розвитку додатку

- Збільшення точності неронної мережі
- Додання веб/мобайл інтерфесу для зручної взаємодії з неромережею
- Реалізація можливості машинного зору (розрізнення об'єктів у реальному часі)

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Recurrent Neural Networks [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/cloud/learn/recurrent-neural-networks
2. An Introduction to Recurrent Neural Networks and the Math That Powers Them [Електронний ресурс] – Режим доступу до ресурсу: https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/
3. 9 Types of Neural Networks: Applications, Pros, and Cons [Електронний ресурс] – Режим доступу до ресурсу: https://www.knowledgehut.com/blog/data-science/types-of-neural-networks
4. Recurrent Neural Network (RNN) [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn
5. Artificial Neural Network Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/artificial-neural-network
6. Artificial Neural Network [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/artificial-neural-network
7. CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning [Електронний ресурс] – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/
8. Convolutional neural network [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Convolutional_neural_network
9. Нейронные сети: практическое применение [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/322392/
10. Область применения искусственных нейронных сетей и перспективы их развития [Електронний ресурс] – Режим доступу до ресурсу:

https://cyberleninka.ru/article/n/oblast-primeneniya-iskusstvennyh-neyronnyh-setey-i-perspektivy-ih-razvitiy

11. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.
12. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11).
13. Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In Eleventh Annual Conference of the International Speech Communication Association.
14. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
15. Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115-118.
16. Chen, J., Wang, C., & Dong, Y. (2020). Deep learning for financial applications: A survey. Applied Soft Computing, 93, 106384.
17. Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018). Progressive growing of gans for improved quality, stability, and variation. In Proceedings of the International Conference on Learning Representations (ICLR).
18. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
19. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
20. Chollet, F. (2018). Deep learning with Python. Manning Publications.
21. Image Classification on ImageNet [Електронний ресурс] – Режим доступу до ресурсу: https://paperswithcode.com/sota/image-classification-on-imagenet?p=xception-deep-learning-with-depthwise

22. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133.
23. History of Artificial Neural Network [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/history-of-artificial-neural-network
24. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117.
25. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, Y. (2014). Generative adversarial nets. Advances in neural information processing systems, 27.
26. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training GANs. Advances in neural information processing systems, 29.
27. Anomaly detection by using a combination of generative adversarial networks and convolutional autoencoders [Електронний ресурс] – Режим доступу до ресурсу: https://asp-eurasipjournals.springeropen.com/articles/10.1186/s13634-022-00943-7
28. EfficientNet: Mingxing Tan, Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." arXiv:1905.11946v5 [cs.CV] (2019).
29. Best Programming Languages for AI [Електронний ресурс] – Режим доступу до ресурсу: https://www.springboard.com/blog/data-science/best-programming-language-for-ai/
30. What is the best programming language for Machine Learning [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7
31. Левченко Д.О. Розробка програми для розпізнавання графічних образів. Курсова робота на здобуття ступеня вищої освіти “бакалавр” Левченко Д.О. Доцент кафедри комп’ютерних наук та програмної інженерії Кравцов Г.М; Міністерство освіти і науки України ; Херсонський держ. ун-т, ф-т комп’ютерних наук, фізики та математики, кафедра комп’ютерних наук та програмної інженерії. – Херсон: ХДУ, 2023. – 46 с.

ДОДАТОК А

```
class FusedMBConv(nn.Module):
    def __init__(self,
                 kernel_size: int,
                 in_channels: int,
                 out_channels: int,
                 expand_ratio: int,
                 stride: int,
                 se_ratio: float,
                 drop_rate: float,
                 norm_layer: Callable[..., nn.Module]):
        super(FusedMBConv, self).__init__()


        assert stride in [1, 2]
        assert se_ratio == 0

        self.out_channels = out_channels
        self.drop_rate = drop_rate

        self.has_shortcut = (stride == 1 and in_channels == out_channels)

        activation_layer = SiLU

        expanded_channels = in_channels * expand_ratio

        self.has_expansion = (expand_ratio != 1)


        ########################################################################## * Expansion
!= 1 * ################################################################
        if self.has_expansion:

            #================================================== * Expansion
convolution(3x3, s1/S2) * ==========================================#
            # in:*para-IN* || out:*para-IN* x ratio || kernel: *para* || stride: *para* || group: 1 || Nomormal:
*para*(BN) || Act: SiLu
            self.expand_conv = ConvBNAct(in_channels      = in_channels,
                                         out_channels     = expanded_channels,
                                         kernel_size      = kernel_size,
                                         stride           = stride,
                                         norm_layer       = norm_layer,
                                         activation_layer = activation_layer)
```

```
            #================================================ * Point-wise linear projection(1x1, s1) * ==========================================#
            # in:*para-IN* x ratio || out:*para-OUT* || kernel: 1 || stride: 1 || group: 1 || Nomormal: *para*(BN) || Act: None
            self.project_conv = ConvBNAct(in_channels      = expanded_channels,
                                          out_channels     = out_channels,
                                          kernel_size      = 1,
                                          norm_layer       = norm_layer,
                                          activation_layer = nn.Identity)  # There is no activation function, so use Identity (empty layer)


        ######################################################################## * Expansion = 1 * ###########################################################
        # When only project_conv exists.
        else:

            #================================================ * Point-wise linear projection(3x3, s1/s2) * ========================================#
            # in:*para-IN* || out:*para-OUT* || kernel: *para* || stride: *para* || group: 1 || Nomormal: *para*(BN) || Act: SiLu
            self.project_conv = ConvBNAct(in_channels      = in_channels,
                                          out_channels     = out_channels,
                                          kernel_size      = kernel_size,
                                          stride           = stride,
                                          norm_layer       = norm_layer,
                                          activation_layer = activation_layer)  # There is activation function.


        #================================================================== * Drop Path * ============================================================#
        # Use the dropout layer only when using the shortcut.
        if self.has_shortcut and drop_rate > 0:
            self.dropout = DropPath(drop_rate)


    def forward(self, x: Tensor) -> Tensor:
        if self.has_expansion:
            result = self.expand_conv(x)
            result = self.project_conv(result)
        else:
            result = self.project_conv(x)

        if self.has_shortcut:
            if self.drop_rate > 0:
                result = self.dropout(result)
            result += x

        return result
```

ДОДАТОК Б

Кастомні фільтри для стискання зображення

```python
def efficientnetv2_s(num_classes: int = 1000):


    model_config = [[2, 3, 1, 1, 24, 24, 0, 0],
                    [4, 3, 2, 4, 24, 48, 0, 0],
                    [4, 3, 2, 4, 48, 64, 0, 0],
                    [6, 3, 2, 4, 64, 128, 1, 0.25],
                    [9, 3, 1, 6, 128, 160, 1, 0.25],
                    [15, 3, 2, 6, 160, 256, 1, 0.25]]

    model = EfficientNetV2(model_cnf    = model_config,
                           num_classes  = num_classes,
                           dropout_rate = 0.2)
    return model
```

✓ 0.4s

```python
def efficientnetv2_m(num_classes: int = 1000):

    model_config = [[3, 3, 1, 1, 24, 24, 0, 0],
                    [5, 3, 2, 4, 24, 48, 0, 0],
                    [5, 3, 2, 4, 48, 80, 0, 0],
                    [7, 3, 2, 4, 80, 160, 1, 0.25],
                    [14, 3, 1, 6, 160, 176, 1, 0.25],
                    [18, 3, 2, 6, 176, 304, 1, 0.25],
                    [5, 3, 1, 6, 304, 512, 1, 0.25]]

    model = EfficientNetV2(model_cnf    = model_config,
                           num_classes  = num_classes,
                           dropout_rate = 0.3)
    return model
```

✓ 0.6s

```python
def efficientnetv2_l(num_classes: int = 1000):

    model_config = [[4, 3, 1, 1, 32, 32, 0, 0],
                    [7, 3, 2, 4, 32, 64, 0, 0],
                    [7, 3, 2, 4, 64, 96, 0, 0],
                    [10, 3, 2, 4, 96, 192, 1, 0.25],
                    [19, 3, 1, 6, 192, 224, 1, 0.25],
                    [25, 3, 2, 6, 224, 384, 1, 0.25],
                    [7, 3, 1, 6, 384, 640, 1, 0.25]]

    model = EfficientNetV2(model_cnf     = model_config,
                           num_classes   = num_classes,
                           dropout_rat   = 0.4)
    return model
```

✓ 0.3s

ДОДАТОК В

MNIST (Modified National Institute of Standards and Technology database) - це набір даних, що складається з рукописних цифр від 0 до 9, який використовується для оцінки алгоритмів розпізнавання зображень. MNIST є одним з найпопулярніших і найбільш відомих наборів даних в галузі машинного навчання і широко використовується для порівняння різних алгоритмів класифікації та навчання нейромереж.

ДОДАТОК Г

Реалізація генератора

```
import tensorflow as tf
from tensorflow.keras.layers import Dense, Reshape, Conv2D, Conv2DTranspose, Flatten
from tensorflow.keras.models import Sequential
import numpy as np
import matplotlib.pyplot as plt
```

```
# normalize [-1, 1]
train_images = (train_images - 127.5) / 127.5
```

```
def create_generator():
    model = Sequential()
    model.add(Dense(8 * 8 * 256, input_dim=100))
    model.add(Reshape((8, 8, 256)))
    model.add(Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same'))
    model.add(Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same'))
    model.add(Conv2DTranspose(3, (5, 5), strides=(2, 2), padding='same', activation='tanh'))
    return model

def create_discriminator():
    model = Sequential()
    model.add(Conv2D(64, (5, 5), strides=(2, 2), padding='same', input_shape=(64, 64, 3)))
    model.add(Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
    model.add(Flatten())
    model.add(Dense(1))
    return model
```

```
generator = create_generator()
discriminator = create_discriminator()

gan = Sequential()
gan.add(generator)
gan.add(discriminator)

discriminator.compile(optimizer='adam', loss=tf.keras.losses.BinaryCrossentropy(from_logits=True))
discriminator.trainable = False

gan.compile(optimizer='adam', loss=tf.keras.losses.BinaryCrossentropy(from_logits=True))
```

```
def train_gan(gan, dataset, batch_size, epochs):
    generator, discriminator = gan.layers
    for epoch in range(epochs):
        for real_images in dataset.batch(batch_size):
            # Генеруємо шум
            noise = tf.random
```

```
BATCH_SIZE = 32
EPOCHS = 1000

dataset = tf.data.Dataset.from_tensor_slices(train_images).shuffle(buffer_size=1000)

train_gan(gan, dataset, BATCH_SIZE, EPOCHS)
```

```
def generate_images(generator, num_images):
    noise = tf.random.normal([num_images, 100])
    generated_images = generator(noise, training=False)
    return (generated_images + 1) / 2

new_images = generate_images(generator, 1000)

new_images = (new_images * 255).numpy().astype(np.uint8)
```

ДОДАТОК Д

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Левченко Дмитро Олександрович,
учасник(ця) освітнього процесу Херсонського державного університету, УСВІДОМЛЮЮ, що академічна
доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.
ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності ЗОБОВ'ЯЗУЮСЯ:
– дотримуватися:
 вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту
Університету;
 принципів та правил академічної доброчесності;
 нульової толерантності до академічного плагіату;
 моральних норм та правил етичної поведінки;
 толерантного ставлення до інших;
 дотримуватися високого рівня культури спілкування;
– надавати згоду на:
 безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за
допомогою спеціалізованих програмних продуктів;
 оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному
репозитарії;
 використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з
метою виявлення можливих ознак академічного плагіату;
– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;
– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності,
використаних методик досліджень та джерел інформації;
– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;
– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного
іміджу;
– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;
– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;
– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан,
соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи
іншою належністю;
– відповідально ставитися до своїх обов’язків, вчасно та сумлінно виконувати необхідні навчальні та
науководослідницькі завдання;
– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і
родинних зв’язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;
– не брати участі в будь-якій діяльності, пов’язаній із обманом, нечесністю, списуванням, фабрикацією;
– не підроблювати документи;
– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і
співробітників;
– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення
впливу на зміну отриманої академічної оцінки;
– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;
– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності
інших студентів та/або працівників;
– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов’язаних з
діяльністю університету;
– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних
методів досягати власних корисних цілей;
– не завдавати загрози власному здоров’ю або безпеці іншим студентам та/або працівникам.
УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної
доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані
заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

5.04.2023 Левченко Дмитро Олександрович