НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”
Факультет прикладної математики
Кафедра прикладної математики

ЛАБОРАТОРНА РОБОТА

з дисципліни “Системи глибинного навчання”

на тему: “Розпізнавання двовимірних кольорових об’єктів за допомогою згорткової нейронної мережі”

Керівник:
Терейковський І. А.

Студентки IV курсу, групи КМ-03
Пюстонен С.Р.

Київ – 2023

ЗМІСТ

ВСТУП

Постановка задачі

Завдання: отримання практичних навичок з розробки програмного забезпечення для реалізації згорткової нейронної мережі, призначеної для розпізнавання двовимірних кольорових об’єктів.

1. Підготовка інструментальних засобів. Для виконання лабораторної роботи рекомендується використовувати мову програмування Python (модуль numpy, бібліотеки Keras та TensorFlow), dataset cifar10. Комп’ютер має бути під’єднаний до мережі Internet.
2. Створення проєкту. В лабораторній роботі використовується згорткова нейрона мережа, структура якої показана на Рисункові 1.1..

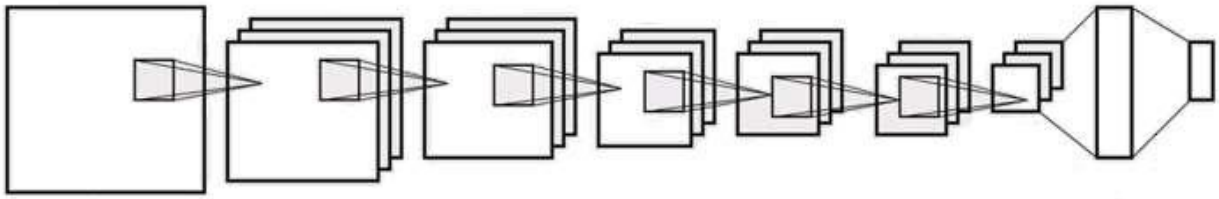

Рисунок 1.1. – Структура нейронної мережі лабораторної роботи

2.1. Створити програму для реалізації навчання нейромережевої моделі.

2.2. Створити програму для реалізації режиму розпізнавання нейромережевої моделі.

3. Навчання нейромережевої моделі. Запустити програму для навчання нейромережевої моделі. Зафіксувати термін та точність навченої нейромережевої моделі.
4. Тестування нейромережевої моделі. Запустити програму для розпізнавання. Дослідити можливості навченої моделі в аспекті розпізнавання різних зображень.
5. Оформити звіт з лабораторної роботи.

2Теоретична частина

Згорткова нейронна мережа (Convolutional Neural Network, CNN) — це клас нейронних мереж, які спеціально призначені для обробки вхідних даних зі структурованим сімптом, таким як зображення чи відео. Вони виявляються дуже ефективними в розпізнаванні образів та подальшому використанні цих образів у завданнях, таких як класифікація та локалізація об'єктів на зображеннях.

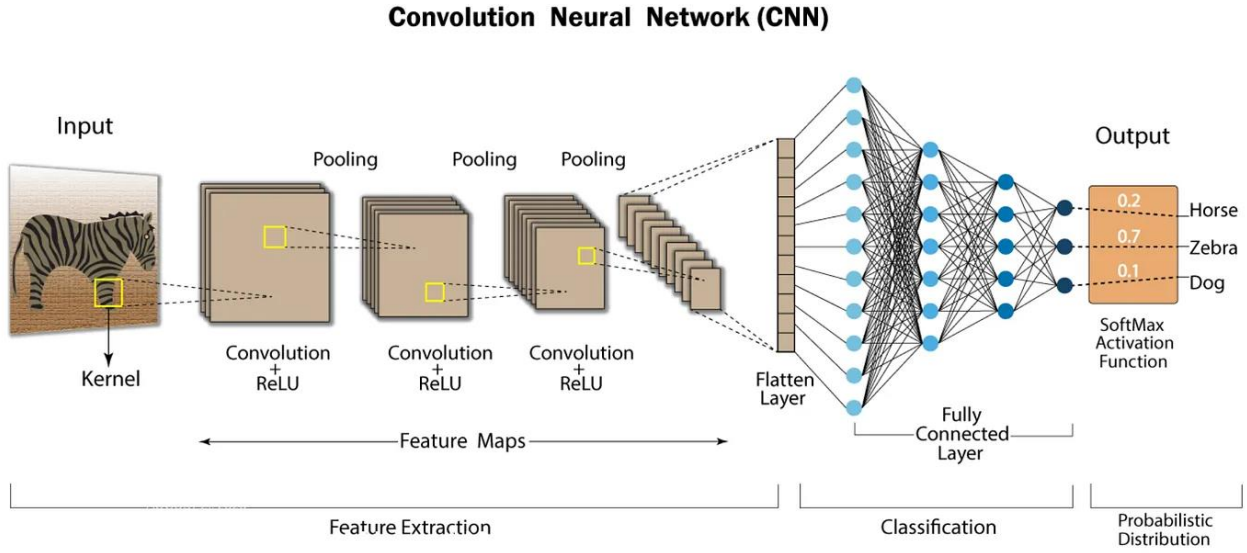

Рисунок 2.1. - Візуалізація роботи CNN

Згорткова нейронна мережа розпочинає роботу, приймаючи вхідне зображення, яке потім трансформується у карту ознак за допомогою серії згорткових та пулінгових шарів. Згортковий шар застосовує набір фільтрів до вхідного зображення, кожен фільтр створює карту ознак, яка виділяє конкретний аспект вхідного зображення. Пулінговий шар зменшує розмір карти ознак, зберігаючи при цьому найважливішу інформацію.

Карту ознак, створену згортковим шаром, потім проходять через кілька додаткових згорткових та пулінгових шарів, при цьому кожен шар вивчає все складніші ознаки вхідного зображення. Завершальним результатом мережі є передбачене класове позначення або ймовірнісний бал для кожного класу в залежності від завдання.

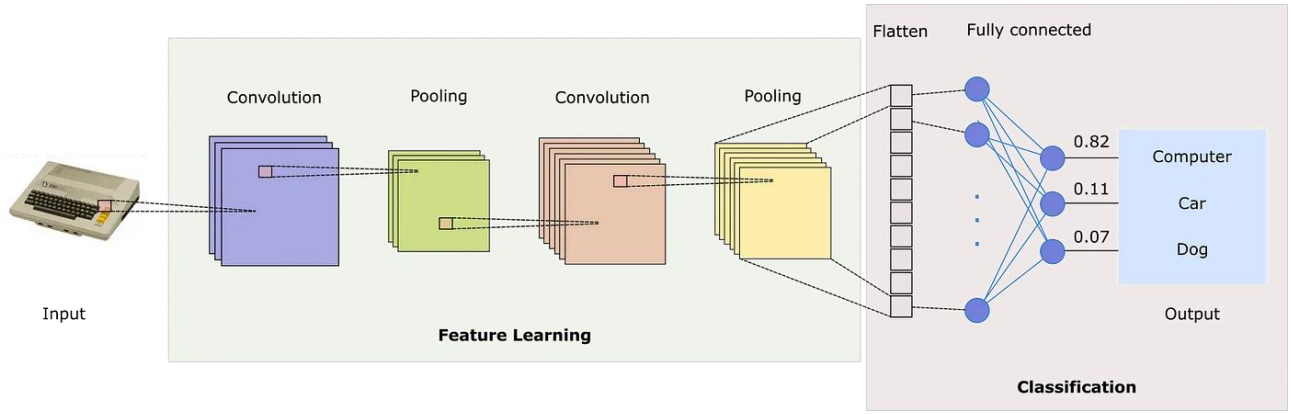

Рисунок 2.2. - Візуалізація шарів CNN

Звичайна архітектура згорткової нейронної мережі складається з трьох основних компонентів: вхідного шару, прихованих шарів та вихідного шару. Вхідний шар отримує вхідне зображення і передає його до прихованих шарів, які складаються з кількох згорткових та пулінгових шарів. Вихідний шар надає передбачене класове позначення або ймовірнісні бали для кожного класу.

Приховані шари є найважливішою частиною згорткової нейронної мережі, і кількість прихованих шарів та кількість фільтрів у кожному шарі можуть бути налаштовані для оптимізації продуктивності мережі. Загальна архітектура для згорткової нейронної мережі часто включає кілька згорткових шарів, за якими слідують один чи декілька пулінгових шарів, а потім повністю з'єднаний шар, який надає кінцевий вихід.

Шари згорткової нейронної мережі (CNN) можна широко класифікувати на наступні категорії:

1. **Згортковий шар**: згортковий шар відповідає за витягання ознак зі вхідного зображення. Він виконує операцію згортки на вхідному зображенні, де до зображення застосовується фільтр або ядро для ідентифікації та вилучення конкретних ознак;

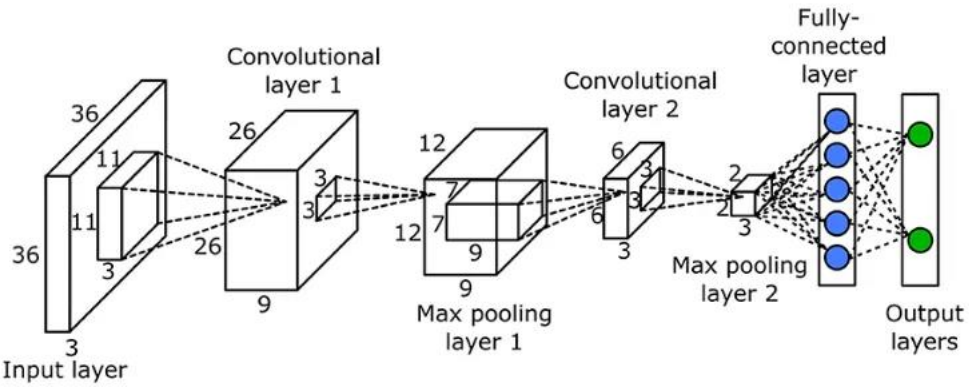

Рисунок 2.3. – Згортковий шар

2. **Шар пулінгу**: шар пулінгу відповідає за зменшення просторових розмірів карт ознак, створених згортковим шаром. Він виконує операцію підвибірки для зменшення розміру карт ознак та спрощення обчислювальної складності;

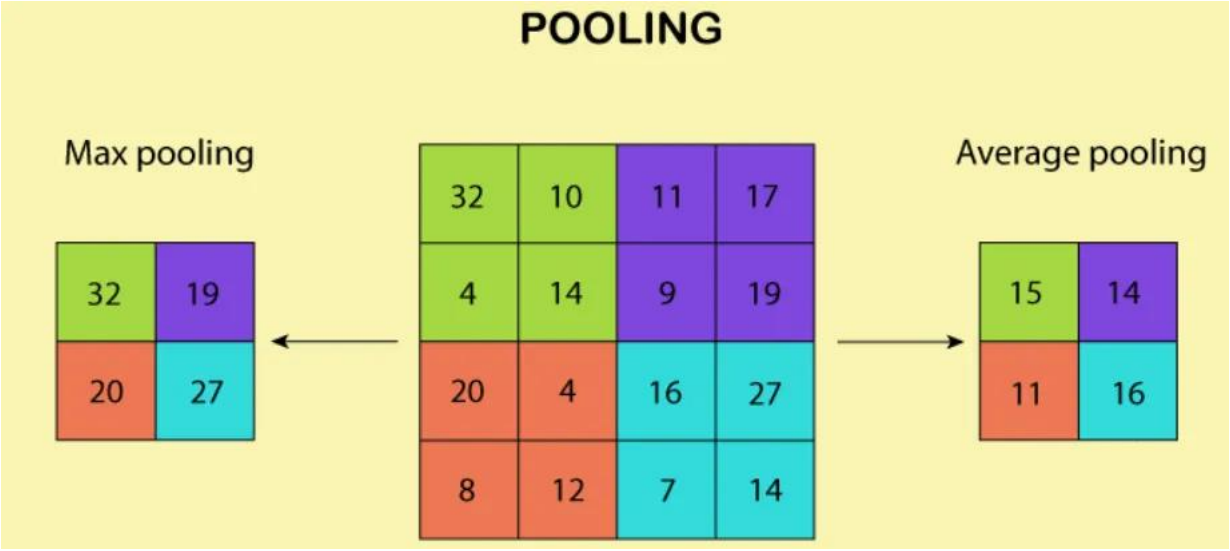

Рисунок 2.4. – Пулінговий шар

3. **Шар активації**: шар активації застосовує нелінійну активаційну функцію, таку як функція ReLU, до виходу пулінгового шару. Ця функція допомагає введенню нелінійності в модель, дозволяючи їй вчитися більш складних представлень вхідних даних;

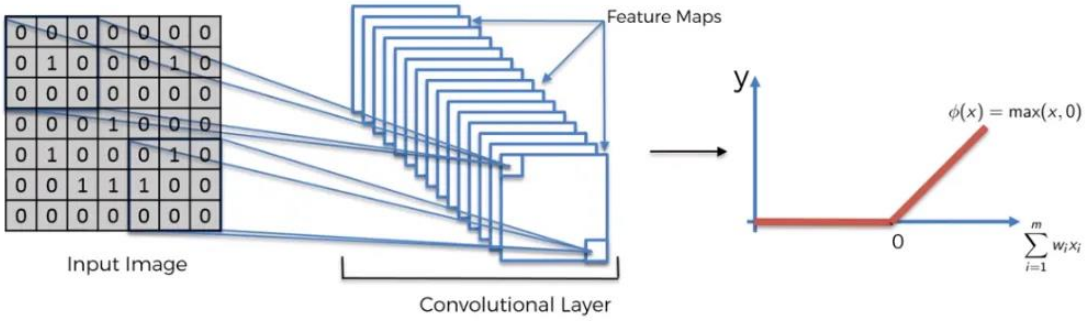

Рисунок 2.5. – Шар активації

4. **Fully connected layer**: повністю з'єднаний шар є традиційним шаром нейронної мережі, який з'єднує всі нейрони попереднього шару з усіма нейронами наступного шару. Цей шар відповідає за об'єднання ознак, вивчених згортковими та пулінговими шарами, для здійснення передбачення;

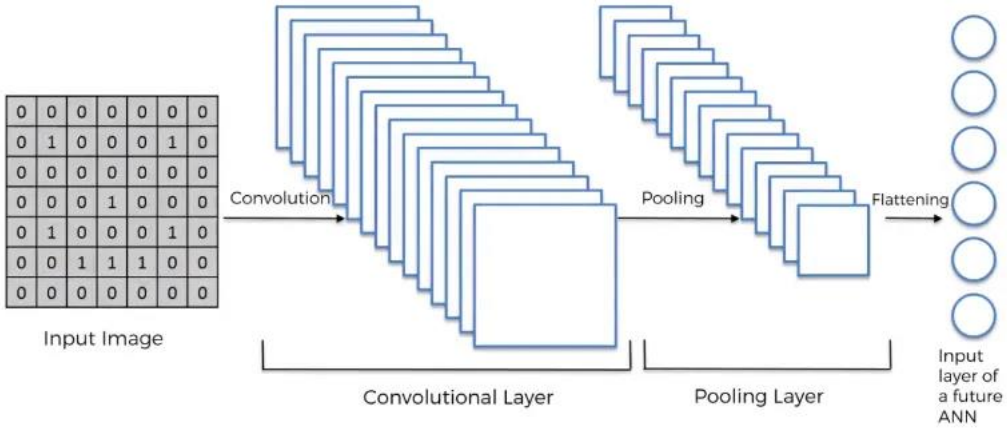

Рисунок 2.6. – Fully connected layer

5. **Normalization layer**: нормалізаційний шар виконує операції нормалізації, такі як нормалізація партії або нормалізація шару, щоб забезпечити, що активації кожного шару мають добрі умови та запобігти перенавчанню;
6. **Dropout layer**: шар випадкового відключення використовується для запобігання перенавчанню шляхом випадкового відключення нейронів під час тренування. Це допомагає забезпечити, що модель не меморізує тренувальні дані, а замість цього узагальнює їх до нових, невиданих даних;

7. **Dense layer**: після того як згорткові та пулінгові шари витягли ознаки з вхідного зображення, повністю з'єднаний шар може бути використаний для об'єднання цих ознак та здійснення остаточного передбачення. У згортковій нейронній мережі (CNN) повністю з'єднаний шар зазвичай є останнім шаром і використовується для генерації вихідних передбачень. Активації з попередніх шарів розгортаються та передаються як вхідні дані повністю з'єднаному шару, який виконує зважену суму вхідних даних та застосовує функцію активації для отримання остаточного виводу.

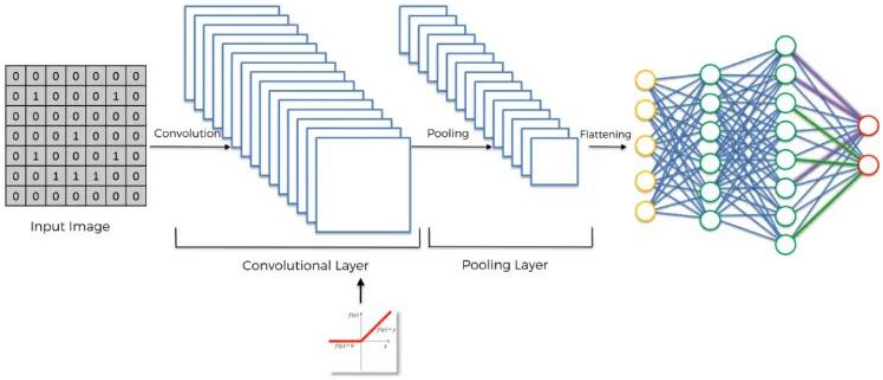

Рисунок 2.7. – Dense layer

Переваги згорткових нейронних мереж (CNN):

1. **Вилучення ознак**: CNN автоматично виділяє відповідні ознаки з вхідних зображень, зменшуючи потребу в ручному інженерінгу ознак;
2. **Просторова інваріантність**: CNN може впізнавати об'єкти на зображеннях незалежно від їхнього місця, розміру або орієнтації, що робить їх відмінними для завдань розпізнавання об'єктів;
3. **Стійкість до шуму**: CNN часто може обробляти зображення з шумом чи забрудненням, що робить їх корисними для застосувань у реальному світі, де якість зображення може варіюватися;

4. **Перенос навчання**: CNN може використовувати передньо-навчені моделі, зменшуючи кількість даних та обчислювальних ресурсів, необхідних для навчання нової моделі;
5. **Ефективність**: CNN демонструє сучасну ефективність в різних завданнях комп'ютерного зору, включаючи класифікацію зображень, виявлення об'єктів та семантичну сегментацію [1].

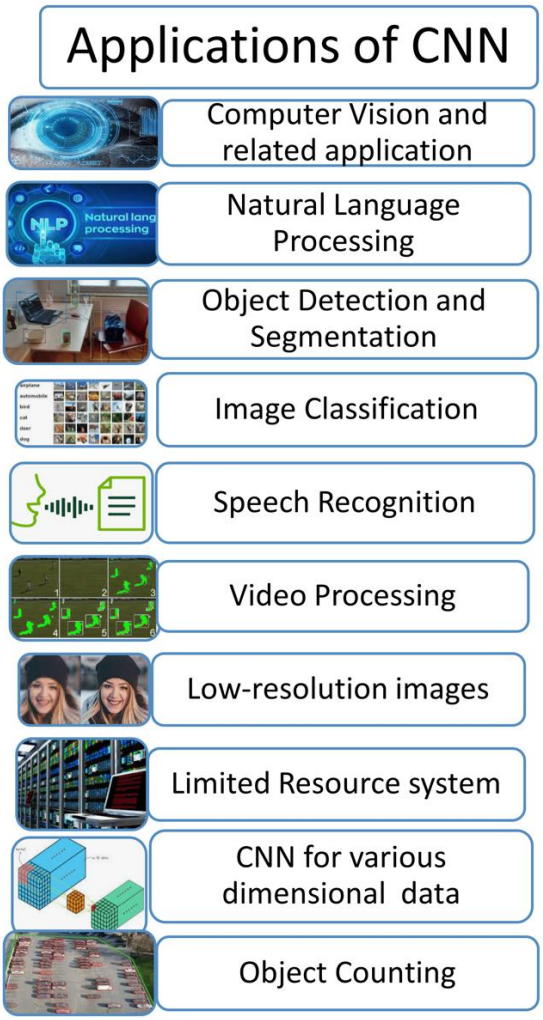

Рисунок 2.8. – Застосування CNN

3Практична частина

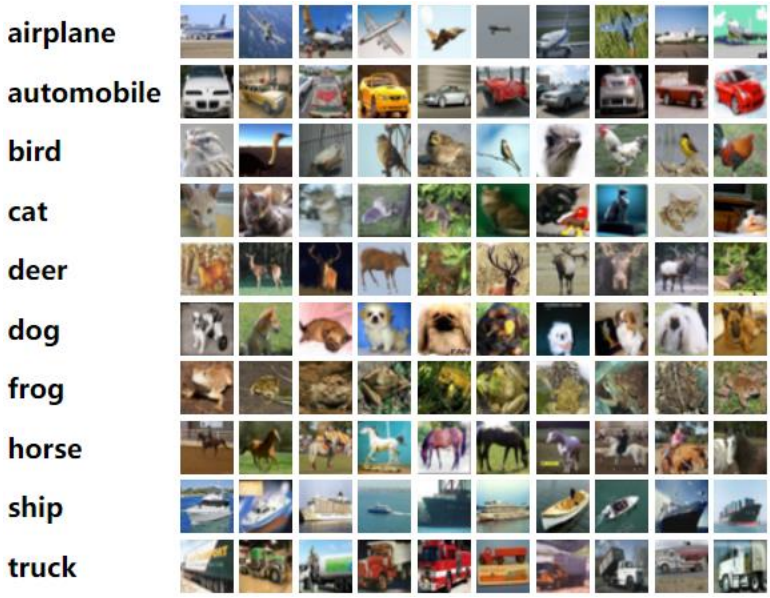

Рисунок 3.1. - Опис датасету

```
Epoch 1/25
1407/1407 - 230s - loss: 1.7465 - accuracy: 0.3570 - val_loss: 1.3703 - val_accuracy: 0.5068 - 230s/epoch - 164ms/ste
Epoch 2/25
1407/1407 - 216s - loss: 1.3240 - accuracy: 0.5222 - val_loss: 1.1781 - val_accuracy: 0.5858 - 216s/epoch - 154ms/ste
Epoch 3/25
1407/1407 - 218s - loss: 1.1425 - accuracy: 0.5931 - val_loss: 1.0150 - val_accuracy: 0.6488 - 218s/epoch - 155ms/ste
Epoch 4/25
1407/1407 - 217s - loss: 1.0184 - accuracy: 0.6362 - val_loss: 0.9245 - val_accuracy: 0.6724 - 217s/epoch - 154ms/ste
Epoch 5/25
1407/1407 - 218s - loss: 0.9252 - accuracy: 0.6743 - val_loss: 0.8851 - val_accuracy: 0.6982 - 218s/epoch - 155ms/ste
Epoch 6/25
1407/1407 - 217s - loss: 0.8609 - accuracy: 0.6994 - val_loss: 0.8234 - val_accuracy: 0.7166 - 217s/epoch - 154ms/ste
Epoch 7/25
1407/1407 - 218s - loss: 0.8129 - accuracy: 0.7156 - val_loss: 0.7957 - val_accuracy: 0.7300 - 218s/epoch - 155ms/ste
Epoch 8/25
1407/1407 - 218s - loss: 0.7756 - accuracy: 0.7294 - val_loss: 0.7160 - val_accuracy: 0.7548 - 218s/epoch - 155ms/ste
Epoch 9/25
1407/1407 - 219s - loss: 0.7391 - accuracy: 0.7416 - val_loss: 0.7308 - val_accuracy: 0.7490 - 219s/epoch - 156ms/ste
Epoch 10/25
1407/1407 - 218s - loss: 0.7045 - accuracy: 0.7559 - val_loss: 0.7092 - val_accuracy: 0.7562 - 218s/epoch - 155ms/ste
Epoch 11/25
1407/1407 - 221s - loss: 0.6839 - accuracy: 0.7631 - val_loss: 0.6697 - val_accuracy: 0.7710 - 221s/epoch - 157ms/ste
Epoch 12/25
1407/1407 - 219s - loss: 0.6608 - accuracy: 0.7710 - val_loss: 0.6923 - val_accuracy: 0.7720 - 219s/epoch - 156ms/ste
Epoch 13/25
1407/1407 - 216s - loss: 0.6451 - accuracy: 0.7764 - val_loss: 0.7058 - val_accuracy: 0.7674 - 216s/epoch - 153ms/ste
Epoch 14/25
1407/1407 - 219s - loss: 0.6296 - accuracy: 0.7832 - val_loss: 0.7118 - val_accuracy: 0.7618 - 219s/epoch - 156ms/ste
Epoch 15/25
1407/1407 - 244s - loss: 0.6138 - accuracy: 0.7862 - val_loss: 0.6984 - val_accuracy: 0.7666 - 244s/epoch - 173ms/ste
Epoch 16/25
1407/1407 - 268s - loss: 0.5986 - accuracy: 0.7923 - val_loss: 0.6787 - val_accuracy: 0.76
Epoch 17/25
1407/1407 - 252s - loss: 0.5979 - accuracy: 0.7939 - val_loss: 0.6717 - val_accuracy: 0.7842 - 252s/epoch - 179ms/ste
Epoch 18/25
1407/1407 - 228s - loss: 0.5886 - accuracy: 0.7983 - val_loss: 0.7056 - val_accuracy: 0.7576 - 228s/epoch - 162ms/ste
Epoch 19/25
1407/1407 - 220s - loss: 0.5939 - accuracy: 0.7948 - val_loss: 0.6907 - val_accuracy: 0.7702 - 220s/epoch - 156ms/ste
Epoch 20/25
1407/1407 - 218s - loss: 0.5823 - accuracy: 0.8000 - val_loss: 0.7031 - val_accuracy: 0.7702 - 218s/epoch - 155ms/ste
Epoch 21/25
1407/1407 - 224s - loss: 0.5752 - accuracy: 0.8015 - val_loss: 0.7021 - val_accuracy: 0.7668 - 224s/epoch - 160ms/ste
Epoch 22/25
1407/1407 - 231s - loss: 0.5668 - accuracy: 0.8051 - val_loss: 0.6991 - val_accuracy: 0.7656 - 231s/epoch - 164ms/ste
Epoch 23/25
1407/1407 - 234s - loss: 0.5609 - accuracy: 0.8090 - val_loss: 0.6912 - val_accuracy: 0.7756 - 234s/epoch - 166ms/ste
Epoch 24/25
1407/1407 - 220s - loss: 0.5523 - accuracy: 0.8104 - val_loss: 0.7304 - val_accuracy: 0.7670 - 220s/epoch - 156ms/ste
Epoch 25/25
1407/1407 - 218s - loss: 0.5579 - accuracy: 0.8089 - val_loss: 0.6698 - val_accuracy: 0.7746 - 218s/epoch - 155ms/ste
Accuracy on test data: 76.72%
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as
  saving_api.save_model(
```

Рисунок 3.2. - Процес навчання моделі

Навчання тривало протягом 1 години 36 хвилини.

Рисунок 3.3. - Зображення для тесту моделі (frog).

```
1/1 [==============================] - 0s 27ms/step
array([[0., 0., 0., 0., 0., 0., 1., 0., 0., 0.]], dtype=float32)
```

Рисунок 3.4. - Результат роботи програми для зображення frog (класифіковано правильно).

Рисунок 3.5. Зображення для тесту моделі (dog)

```
1/1 [==============================] - 0s 74ms/step
array([[0., 0., 0., 0., 0., 1., 0., 0., 0., 0.]], dtype=float32)
```

Рисунок 3.6. - Результат роботи програми для зображення dog (класифіковано правильно).

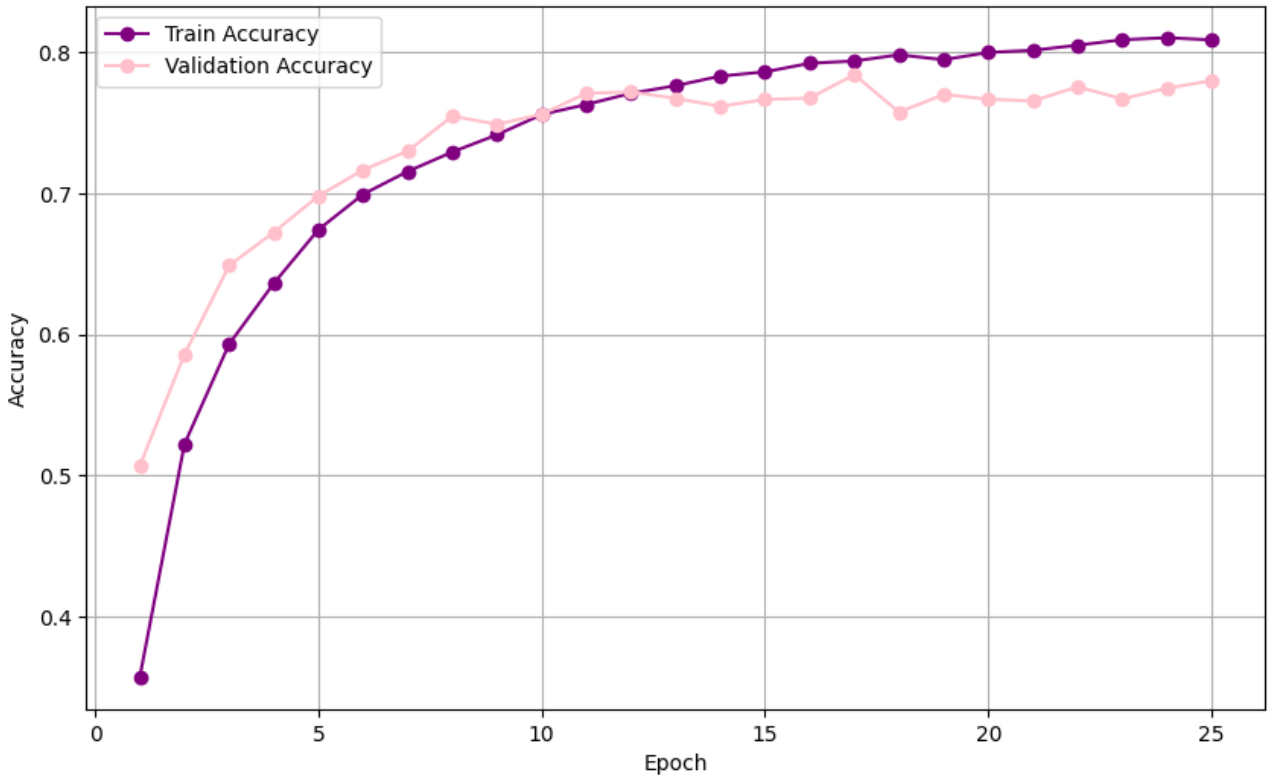

Рисунок 3.7. - Точність роботи моделі на тренувальній та тестовій (валідаційній) вибірці.

```
Train:
1563/1563 [==============================] - 65s 41ms/step
Accuracy:  0.91526
Precision:  0.9163793365520608
Recall:  0.91526
F1 :  0.915453387326389
Matthews corr:  0.9059282237288676
Balanced accuracy:  0.91526
Test:
313/313 [==============================] - 13s 41ms/step
Accuracy:  0.7672
Precision:  0.7702041305978197
Recall:  0.7672
F1 :  0.7676353384280197
Matthews corr:  0.7415671616054241
Balanced accuracy:  0.7672
```

Рисунок 3.8. - Показники ефективності на навчальному та тестовому наборах даних для навченої моделі.

ВИСНОВКИ

Модель налагоджена добре: класифікаційні метрики в порядку, перенавчання не спостерігається, тестові малюнки класифіковані правильно.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5

ДОДАТКИ

https://colab.research.google.com/drive/1J0XwtForRx6df2iBD8bnQpq0ChUQqqv9?usp=sharing