НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №4

з дисципліни «Системи глибинного навчання»

на тему

“Розпізнавання двовимірних кольорових об’єктів за допомогою згорткової

нейронної мережі”

|  |  |
| --- | --- |
| Виконав: | Викладач: |
| студент групи КМ-03 | Професор кафедри ПМА |
| Шаповалов Г. Г. | Терейковський І. А. |

Київ — 2023

Зміст

[Теоретичні відомості 3](#_Toc152438751)

[Основна частина 4](#_Toc152438752)

[Додаток А – Код програми 7](#_Toc152438753)

## Теоретичні відомості

Згорткові нейронні мережі (ЗНМ) є потужним інструментом у галузі глибокого навчання, особливо в обробці зображень та відео. Основним компонентом ЗНМ є згорткові шари, які дозволяють ефективно взаємодіяти з просторовою структурою вхідних даних.

Кожен згортковий шар включає набір фільтрів (ядро), які рухаються по вхідному зображенню та виконують операцію згортки. Ці фільтри виявляють різні особливості, такі як границі, форми або текстури, що дозволяє ЗНМ ефективно витягувати важливі ознаки з вхідних даних.

Після згортки використовуються шари пулінгу, які зменшують просторовий розмір отриманих карт ознак, зберігаючи при цьому важливі інформаційні особливості. Це допомагає зменшити кількість параметрів та обчислювальний обсяг, що поліпшує ефективність навчання та прискорює роботу мережі.

Після згортки та пулінгу використовують повнозв'язані шари, які об'єднують отримані ознаки для прийняття рішення або класифікації. Згорткові нейронні мережі успішно використовуються в різних задачах, таких як розпізнавання об'єктів, визначення обличчя, аналіз зображень та багато інших, завдяки їх здатності автоматично вивчати ієрархічні функції та властивості вхідних даних.

# Основна частина

Для початку імпортуємо потрібні бібліотеки:

Numpy – для математичних розрахунків

Tensorflow – для створення НМ

Keras – для завантаження датасету, перекодування міток

Sklearn – для розбиття наборів даних та знаходження метрик у кінці

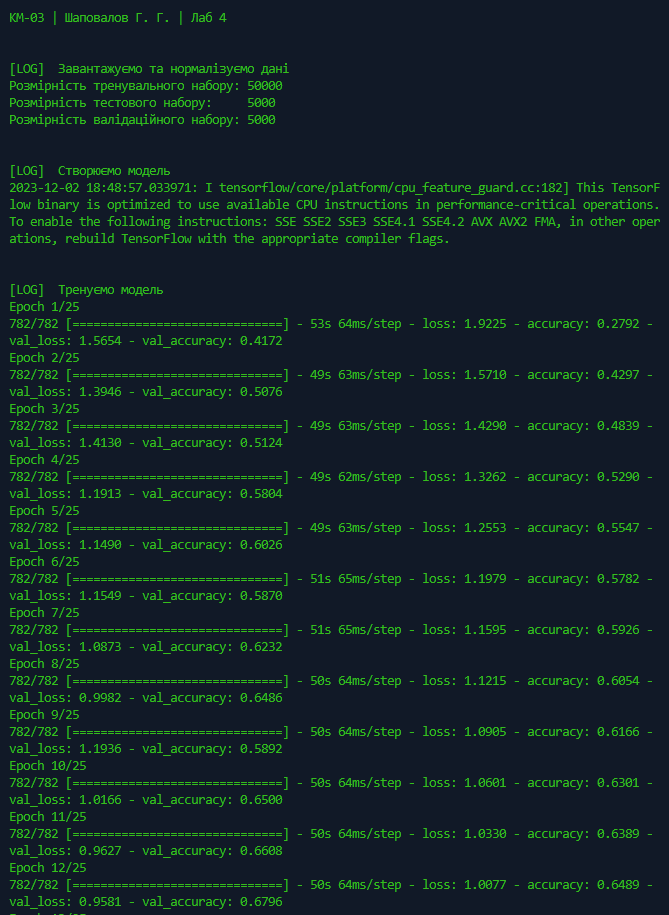
***Валідаційні дані*** – дані для тестування після кожної епохи. Походження даних – розбиття тестового набору на 2 рівні частини.

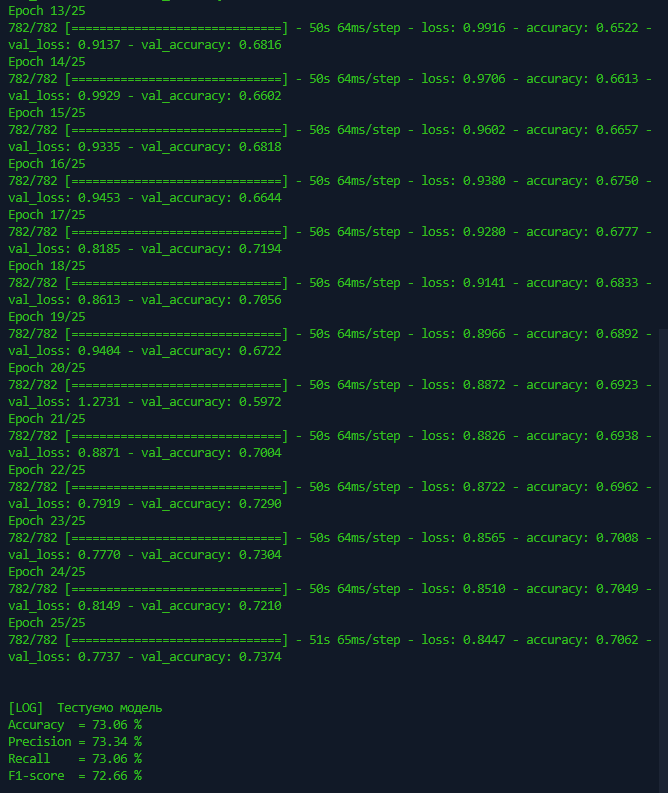
В якості оптимізатора моделі вибрано SGD з параметрами: ***learning\_rate=0.01, momentum=0.9, nesterov=True***

Датасет має 10 класів зображень, тому використовувалась ***categorical\_crossentropy*** в якості ***loss функції***

Модель навчається 25 епох.

Скріншоти роботи програми:





Модель натренувалась на 25-ти епохах. Досягла точності в 0.7374 на валідаційному наборі на останній епосі. В результаті, метрики показали непогані результати, отже модель змогла більш-менш навчитись.

# Додаток А – Код програми

import numpy as np

import tensorflow as tf

from keras import layers, models

from keras.optimizers import SGD

from keras.datasets import cifar10

from keras.utils import to\_categorical

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import precision\_score, recall\_score, f1\_score

print('\n\nКМ-03 | Шаповалов Г. Г. | Лаб 4')

print(f'\n\n[LOG]  Завантажуємо та нормалізуємо дані')

batch\_size = 64

(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()

X\_train = X\_train / 255.0

X\_test  = X\_test / 255.0

y\_train = to\_categorical(y\_train, 10)

y\_test  = to\_categorical(y\_test,  10)

X\_test, X\_val, y\_test, y\_val = train\_test\_split(X\_test, y\_test, test\_size=0.5, random\_state=42)

print(f'Розмірність тренувального набору: {len(X\_train)}')

print(f'Розмірність тестового набору:     {len(X\_test)}')

print(f'Розмірність валідаційного набору: {len(X\_val)}')

print(f'\n\n[LOG]  Створюємо модель')

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(32, 32, 3)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dropout(0.5))

model.add(layers.Dense(10, activation='softmax'))  # 10 класів виводу

sgd = SGD(learning\_rate=0.01, momentum=0.9, nesterov=True)

model.compile(optimizer=sgd,

              loss='categorical\_crossentropy',

              metrics=['accuracy'])

print(f'\n\n[LOG]  Тренуємо модель')

try:

    model = models.load\_model('my\_model.keras')

except:

    history = model.fit(X\_train, y\_train, epochs=25, batch\_size=batch\_size, validation\_data=(X\_val, y\_val))

    model.save('my\_model.keras')

print(f'\n\n[LOG]  Тестуємо модель')

test\_results = model.evaluate(X\_test, y\_test, verbose=0)

test\_accuracy = test\_results[1]

predictions = np.argmax(model.predict(X\_test, verbose=0), axis=1)

test\_precision = precision\_score(np.argmax(y\_test, axis=1), predictions, average='weighted')

test\_recall = recall\_score(np.argmax(y\_test, axis=1), predictions, average='weighted')

test\_f1\_score = f1\_score(np.argmax(y\_test, axis=1), predictions, average='weighted')

print(f'Accuracy  = {test\_accuracy \* 100:.2f} %')

print(f'Precision = {test\_precision \* 100:.2f} %')

print(f'Recall    = {test\_recall \* 100:.2f} %')

print(f'F1-score  = {test\_f1\_score \* 100:.2f} %')