НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

лабораторної роботи №4

з дисципліни «[СИСТЕМИ ГЛИБИННОГО НАВЧАННЯ](https://classroom.google.com/c/NjIxMzk4NjAyNjI4)»

на тему

«Розпізнавання двовимірних  
кольорових об’єктів за допомогою згорткової нейронної мережі»

|  |  |
| --- | --- |
| Виконав:  Студент групи КМ-03  Мітченко А.Д. | Перевірив:  доцент  Терейковський І. А. |
|  |  |
|  |  |
|  |  |

Київ — 2023

ЗМІСТ

[Теоретичні відомості 3](#_Toc150269750)

[Умова завдання: 5](#_Toc150269751)

[Скріншоти та опис програми 6](#_Toc150269752)

[ДОДАТОК А ТЕКСТ ПРОГРАМИ 7](#_Toc150269753)

[Завдання : 7](#_Toc150269754)

# Теоретичні відомості

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

## Згорткові нейронні мережі (CNN)

Згорткові нейронні мережі (CNN) - це тип нейронних мереж, які особливо ефективні у роботі з великими об'ємами даних, такими як зображення. Основною їхньою перевагою є здатність розпізнавати властивості вхідних даних, зокрема просторові та ієрархічні особливості, що робить їх дуже корисними для задач обробки зображень.

### Основні компоненти CNN:

**1. Згорткові Шари (Convolutional Layers):**

- Ядро (Kernel) та Згортка (Convolution): Згорткові шари використовують невеликі фільтри або ядра, які проходять по вхідних даних для виділення різних властивостей. Згортка представляє собою операцію множення і накладання фільтра на вхідні дані.

- Стрід (Stride) та Падіння (Padding): Стрід визначає крок переміщення ядра під час згортки, а падіння додає нульові значення навколо вхідних даних, щоб зберегти розмірність виводу.

**2. Пулінгові Шари (Pooling Layers):**

- Пулінг (Pooling): Ця операція використовується для зменшення розмірності зображення та виділення ключових ознак. Зазвичай використовується максимальне або середнє значення в області пулінгу.

**3. Повнозв'язані Шари (Fully Connected Layers):**

- Повнозв'язані нейрони: Ці шари приймають векторизовані властивості вихідних даних з попередніх шарів та використовують їх для класифікації чи регресії.

**4. Функції Активації:**

- ReLU (Rectified Linear Unit): Часто використовується функція активації, яка призводить до нуля від'ємні значення і залишає позитивні без змін.

**5. Занурення (Dropout):**

- Випадкове відключення нейронів: Використовується для запобігання перенавчанню шляхом випадкового відключення певного відсотка нейронів під час тренування.

**6. Функція Втрат:**

- Категоріальна крос-ентропія (Categorical Cross-Entropy): Зазвичай використовується для задач класифікації, де є багато класів.

Згорткові нейронні мережі (CNN) мають свої плюси та мінуси, як і будь-яка інша технологія. Давайте розглянемо деякі з них:

### Плюси Згорткових Нейронних Мереж (CNN):

**1. Ефективність у обробці зображень:**

- CNN виявляються дуже ефективними для задач обробки зображень, завдяки здатності виділяти просторові особливості та ієрархічні структури в зображеннях.

**2. Здатність до автоматичного вивчення властивостей:**

- Завдяки проходженню через згорткові та пулінгові шари, CNN можуть автоматично вивчати важливі фільтри та властивості, що робить їх відмінними для завдань візуального врядкування.

**3. Варіативність вхідних розмірів:**

- CNN можуть ефективно обробляти вхідні дані різних розмірів завдяки використанню згорткових шарів та пулінгу.

**4. Можливість використання передньо навчених моделей:**

- Можна використовувати передньо навчені моделі (transfer learning), що дозволяє використовувати ваги, навчені на великих обсягах даних, для покращення результатів на обмежених даних.

### Мінуси Згорткових Нейронних Мереж (CNN):

**1. Вимоги до обчислень:**

- Тренування та використання великих CNN може вимагати значних обчислювальних ресурсів, що може бути важкодоступним для деяких застосувань.

**2. Потреба в об'ємних навчальних даних:**

- CNN ефективні, коли є достатньо об'ємний набір даних для тренування. У випадках обмежених даних може виникнути перенавчання.

**3. Неякісне визначення особливостей:**

- Якщо вхідні дані мають неякісні або невідповідні особливості, це може негативно позначитися на результативності CNN.

**4. Неефективність для деяких завдань:**

- В деяких завданнях, де важливі просторові структури менш важливі, CNN може бути менш ефективним, і інші типи нейронних мереж можуть бути більш придатними.

**5. Чутливість до перекосу в навчальних даних:**

- CNN можуть бути чутливими до перекосів в навчальних даних та шумів, що може призводити до неправильних класифікацій.

## Умова завдання:

**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 рис. 1.

Изображение выглядит как Шрифт, текст, белый, дизайн

Автоматически созданное описание

**2.1. Створити програму для реалізації навчання нейромережевої моделі (код наведений в додатку 1).**

**2.2. Створити програму для реалізації режиму розпізнавання нейромережевої моделі (код наведений в додатку 2).**

**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.**

# Скріншоти та опис програми

## Программа 1:

Спочатку встановлюємо потрібні нам бібліотеки.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Завантажуємо дата сет, та розбиваємо його на навчальну та тестувальні вибірки. А також приводимо фотографії у потрібний нам вид.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Створюємо модель та задаємо їй параметри.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Тренуємо нашу модель.

Изображение выглядит как текст, снимок экрана, Шрифт

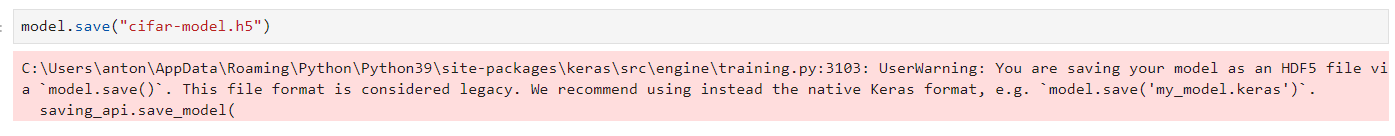
Автоматически созданное описание

Як видно в нас відрізняється точність моделі на тренувальній виборці та валідаційній в середньому час витрачений на 1 епоху – 26 сек. В кінці у нас виходить 0.93 на тренувальній та 0.7 на тестовій.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Зберігаємо нашу модель.



## Программа 2:

Знову портуємо бібліотеки та завантажуємо нашу модель.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Розбиваємо данні на той формат що нам підходить.

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Обираємо потрібну нам картинку.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Закидуємо в модель та дивимось, що вийшло.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Як видно, модель обрала не тот клас. Спробуємо ще раз

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

На другому прикладі модель правильно поставила лейбл.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

# ДОДАТОК А ТЕКСТ ПРОГРАМИ

## Программа 1:

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten

from tensorflow.keras.layers import BatchNormalization

from tensorflow.keras.layers import Conv2D, MaxPooling2D

from tensorflow.keras.regularizers import l2

from tensorflow.keras.optimizers import Adam

from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

from tensorflow.keras import datasets

from keras.utils import to\_categorical

from keras.models import load\_model

import numpy as np

import os

import pandas as pd

import matplotlib.pyplot as plt

(training\_images, training\_labels), (testing\_images, testing\_labels) = datasets.cifar10.load\_data()

training\_labels\_one\_hot = to\_categorical(training\_labels, num\_classes=10)

testing\_labels\_one\_hot = to\_categorical(testing\_labels, num\_classes=10)

print('x\_train shape => ' , training\_images.shape)

print('x\_test shape => ' , testing\_images.shape)

print('training\_labels shape => ' , training\_labels.shape)

print('testing\_labels shape => ' , testing\_labels.shape)

training\_images, testing\_images = training\_images / 255, testing\_images / 255

test\_files = ['/kaggle/input/cifar-10/test/' + str(i) + '.png' for i in range(1, 300001)]

model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(32, (3,3), activation='relu', input\_shape=(32, 32, 3)),

tf.keras.layers.MaxPooling2D(2, 2),

tf.keras.layers.Conv2D(64, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(512, activation='relu'),

tf.keras.layers.Dense(10, activation='softmax')

])

#Train the model

model.compile(optimizer=Adam(learning\_rate=0.001),

loss='categorical\_crossentropy',

metrics=['accuracy'])

model.fit(training\_images, training\_labels\_one\_hot, epochs=10, batch\_size=64, validation\_data=(testing\_images, testing\_labels\_one\_hot))

model.save("cifar-model.h5")

## Программа 2:

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten

from tensorflow.keras.layers import BatchNormalization

from tensorflow.keras.layers import Conv2D, MaxPooling2D

from tensorflow.keras.regularizers import l2

from tensorflow.keras.optimizers import Adam

from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

from tensorflow.keras import datasets

from keras.utils import to\_categorical

from keras.models import load\_model

import numpy as np

import os

import pandas as pd

import matplotlib.pyplot as plt

model = load\_model('cifar-model.h5')

(training\_images, training\_labels), (testing\_images, testing\_labels) = datasets.cifar10.load\_data()

training\_labels\_one\_hot = to\_categorical(training\_labels, num\_classes=10)

testing\_labels\_one\_hot = to\_categorical(testing\_labels, num\_classes=10)

print('x\_train shape => ' , training\_images.shape)

print('x\_test shape => ' , testing\_images.shape)

print('training\_labels shape => ' , training\_labels.shape)

print('testing\_labels shape => ' , testing\_labels.shape)

training\_images, testing\_images = training\_images / 255, testing\_images / 255

index\_to\_display = 2

image\_to\_display = testing\_images[index\_to\_display]

label\_to\_display = testing\_labels[index\_to\_display]

image\_to\_display = image\_to\_display / 255.0

image\_to\_display = np.reshape(image\_to\_display, (1, 32, 32, 3))

predictions = model.predict(image\_to\_display)

predicted\_label = np.argmax(predictions)

plt.imshow(testing\_images[index\_to\_display])

plt.title(f"True Label: {label\_to\_display}, Predicted Label: {predicted\_label}")

plt.show()