Міністерство освіти і науки України
Національний технічний університет України «КПІ» імені Ігоря Сікорського
Кафедра обчислювальної техніки ФІОТ

ЗВІТ
з лабораторної роботи №7
з навчальної дисципліни «Computer Vision»

Тема:

ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ІДЕНТИФІКАЦІЇ ОБ’ЄКТІВ НА ЦИФРОВИХ ЗОБРАЖЕННЯХ ДЛЯ ЗАДАЧ COMPUTER VISION

Виконав:
Студент 3 курсу кафедри ІПІ ФІОТ,
Навчальної групи ІП-11
Панченко С.В.

Перевірив:
Професор кафедри ОТ ФІОТ
Писарчук О.О.

Київ 2024 р.

І. Мета:

Дослідити принципи та особливості підготовки даних, синтезу, навчання та застосування штучних нейронних мереж (Artificial Neural Networks) для практичних задач ідентифікації в технологіях Computer Vision

ІІ. Завдання.

Розробити програмний скрипт мовою Python що реалізує обчислювальний алгоритм ідентифікації об’єктів на цифрових зображеннях за технологіями штучних нейронних мереж (Artificial Neural Networks): підготовка даних; конструювання нейромережі; навчання штучної нейронної мережі; застосування нейромережі:

Варіант (місяць народження) 06.03.2004	Технічні умови завдання
3	Розробити програмний скрипт, що забезпечує ідентифікацію бінарних зображень 4 спеціальних знаків, заданих матрицею растра. Для ідентифікації синтезувати, навчити та застосувати штучну нейронну мережу в «сирому» вигляді реалізації матричних операцій. Обґрунтувати вибір архітектури та алгоритму навчання нейромережі. Довести працездатність та ефективність синтезованої нейронної мережі.
3	Із впровадженням технологій штучних нейронних мереж розробити програмний скрипт для виявлення людей зі зброєю.

ІІІ. Результати виконання лабораторної роботи.

3.1. Синтезована математична модель відповідно до завдання.

Математична модель навчання із вчителем:

1. Початкові значення вагових коефіцієнтів приймаються нульовими, або невеликими випадковими
числами;ss
2. Для кожного навчального зразка $x^{(i)}$:
    2.1. Обчислити вихідне значення $y^{(i)}$;
    2.2. Оновити вагові коефіцієнти за моделлю:

$$w_j := w_j + \Delta w_j,$$

$$\Delta w_j = \eta(y^{(i)} - y^{(i-1)})x_j^{(i)}$$

$$\Delta w_0 = \eta(y^{(i)} - y^{(i-1)}), \Delta w_1 = \eta(y^{(i)} - y^{(i-1)})x_1^i, \Delta w_2 = \eta(y^{(i)} - y^{(i-1)})x_2^i$$

Рисунок 1 — опис роботи нейронної мережі

3.2. Блок схема алгоритму завдання та її опис.

Застосування синтезованих моделей здійснюється у порядку, що відображає суть алгоритму реалізації завдання лабораторної роботи. Далі зображено схему для першого завдання:

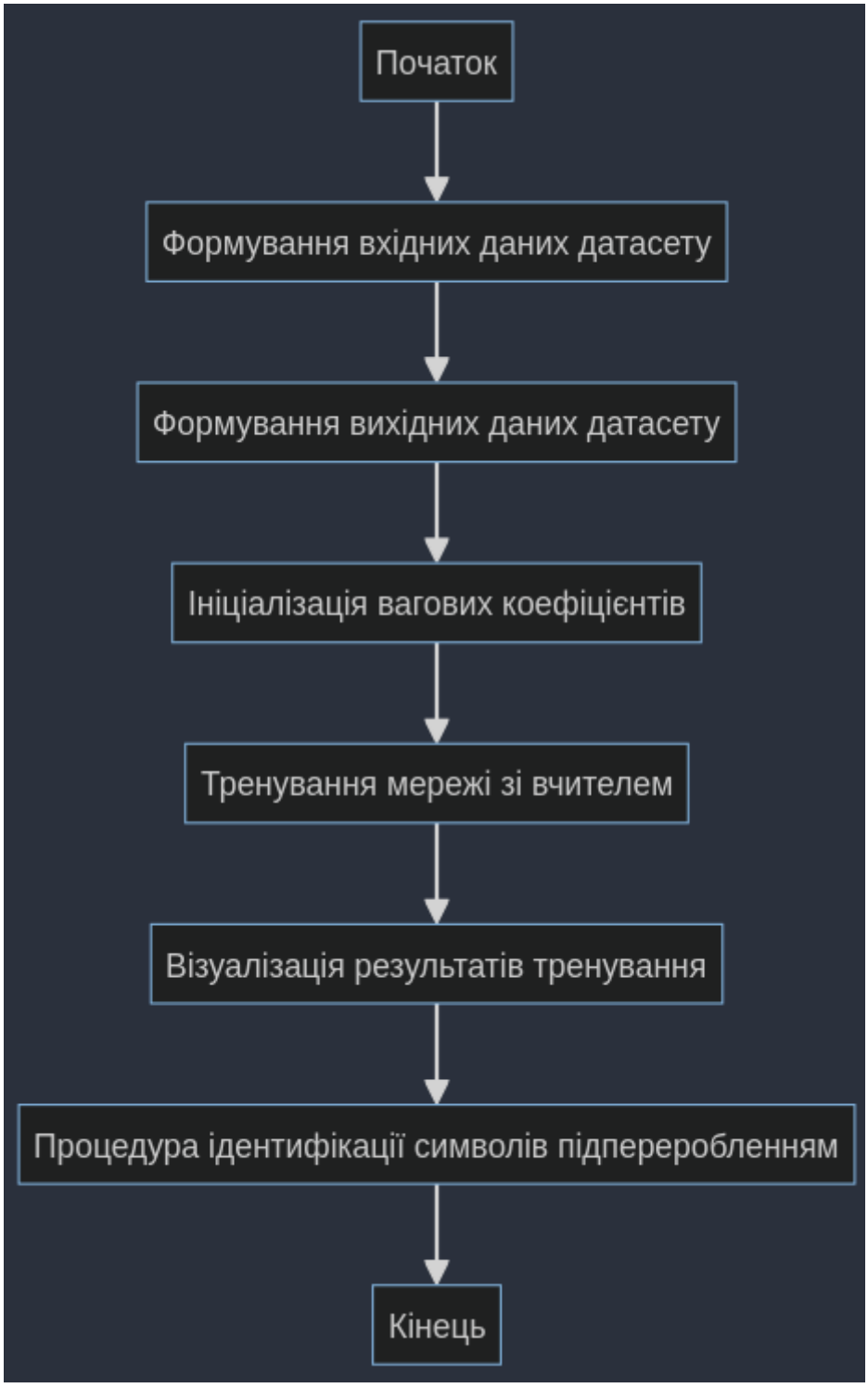

Рисунок 2 — схема роботи пергого завдання

Зображено схему для другого завдання:

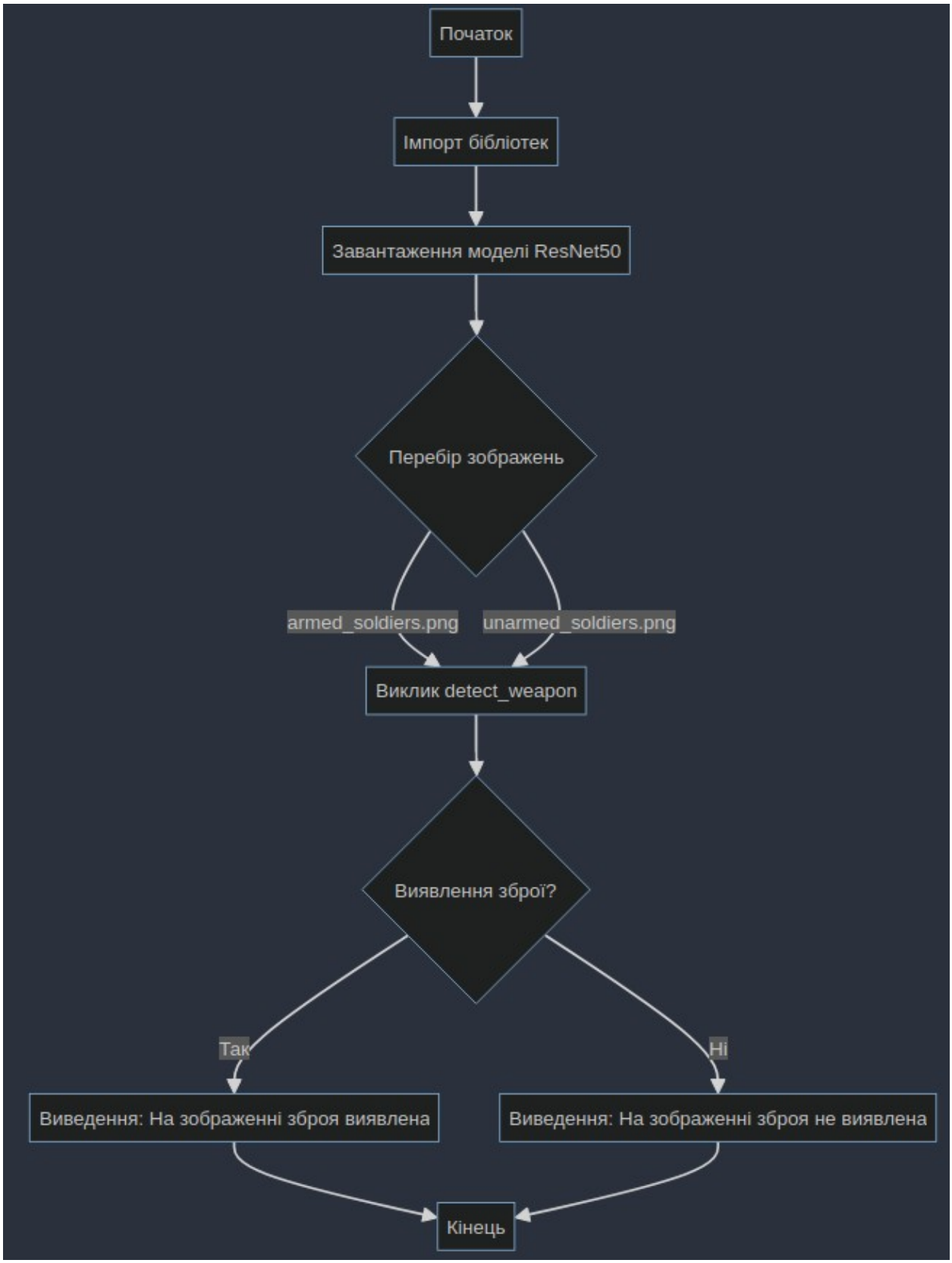

Рисунок 3 — схема роботи другого завдання

3.3. Опис структури проекту програми в середовищі PyCharm.

Для реалізації розробленого алгоритму мовою програмування Python з використанням можливостей інтегрованого середовища PyCharm сформовано проект.

Проект базується на лінійній бізнес-логіці функціонального програмування та має таку структуру.

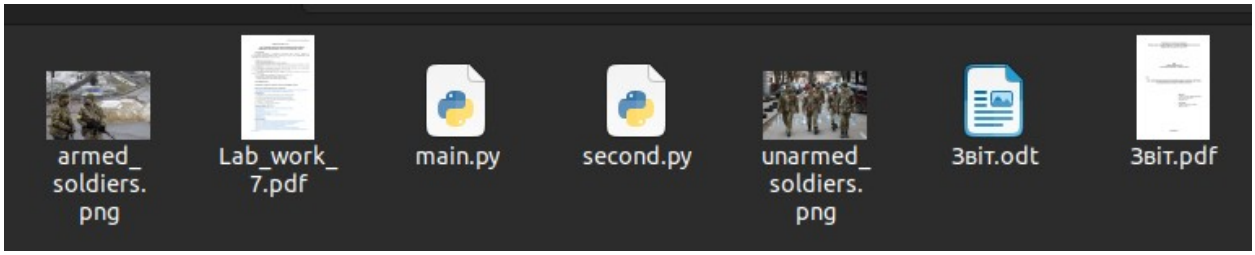

Рисунок 4 — структура проєкту

main.py – скрипт виконання 1-го завдання
second.py — реалізація 2-го завдання
Звіт.odt - звіт
armed_soldiers.png — солдати зі зброєю
unarmed_soldiers.png — солдати без зброї

3.4. Результати роботи програми відповідно до завдання.

Результатом роботи скрипта завдання є сукупність послідовності графічних вікон та результати передбачення відображені в консолі, що реалізують умови завдання лабораторної роботи.

Вхідна частина тренувального датасету відображена за допомогою функцій бібліотеки matplotlib. Обрані спеціальні символи : +, #, ?, &

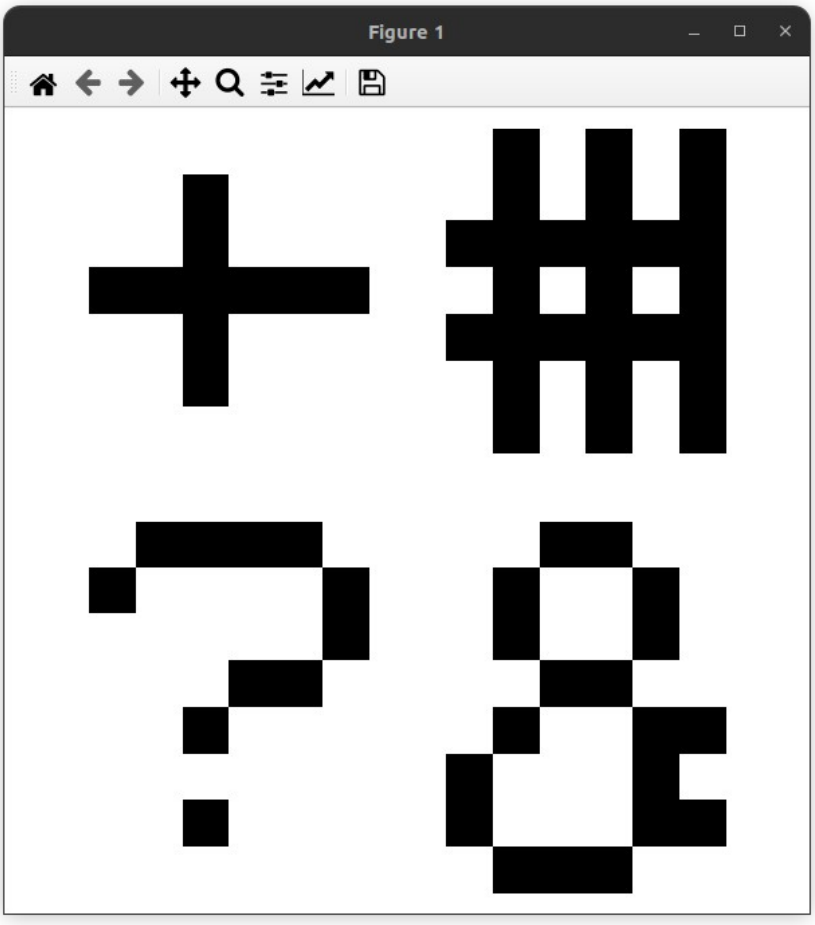

Рисунок 5 — зображення спеціальних

Візуалізація перебігу навчання. Точність мережі на епохах:

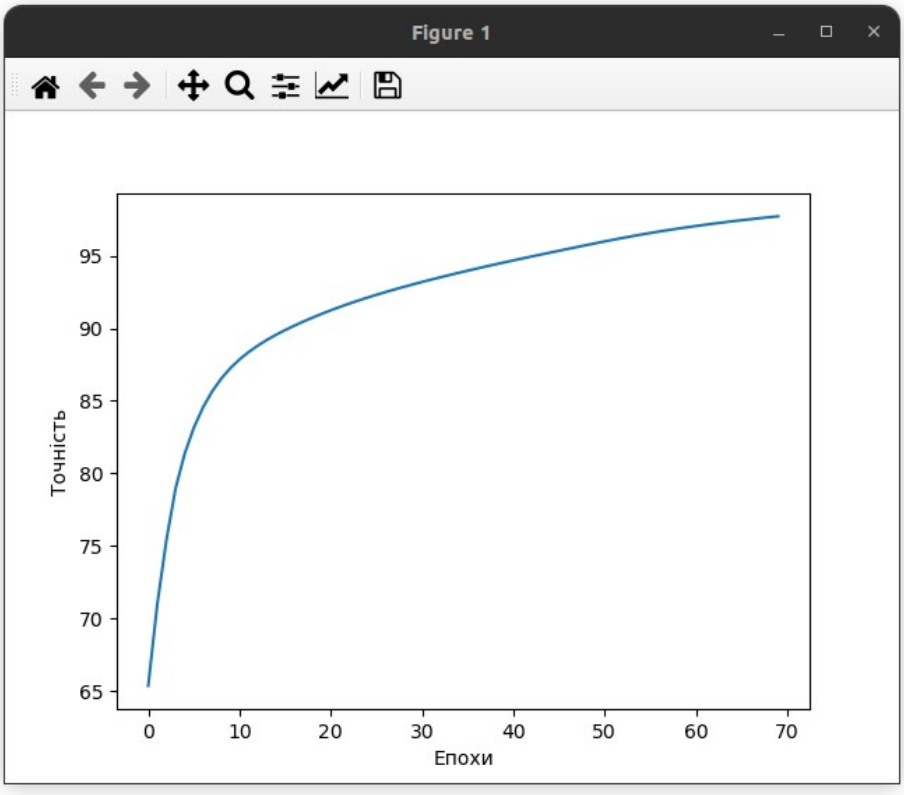

Рисунок 6 — залежність точності від кількості епох

Візуалізація перебігу навчання. Втрати мережі на епохах:

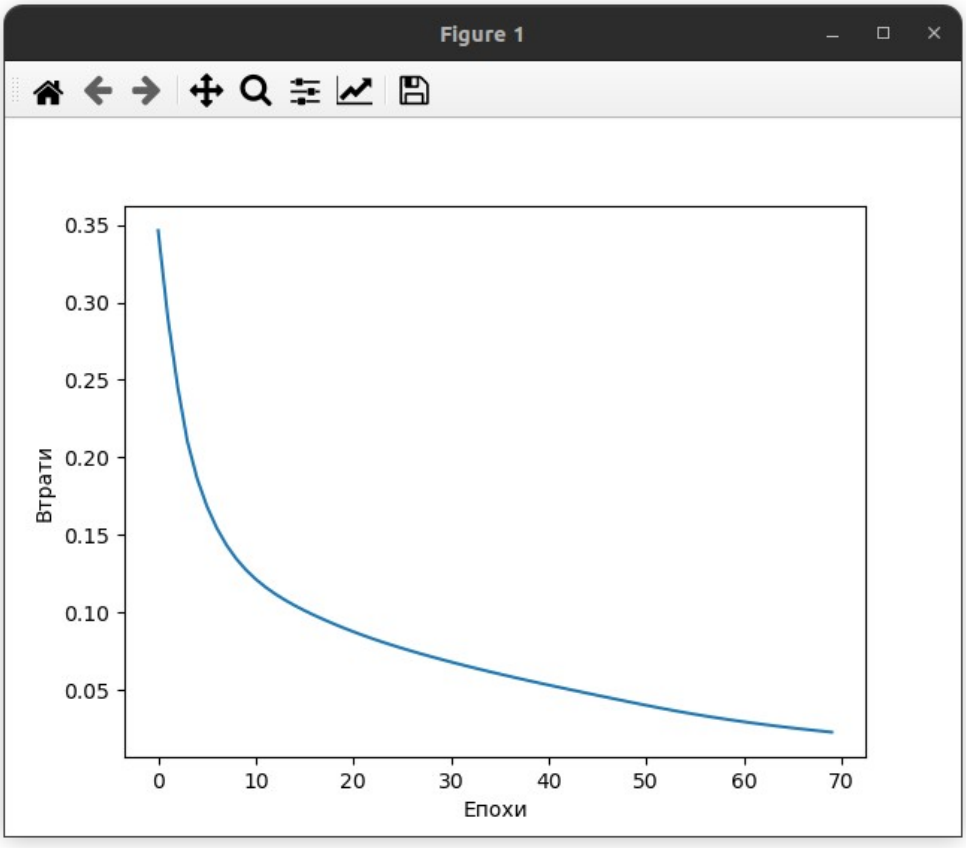

Рисунок 7 — залежність втрат від кількості епох

Результати ідентифікації символів:

```
Вхідні параметри відповідають символу "+"
Результат ідентифікації:
Зображення символу +.

Вхідні параметри відповідають символу "#"
Результат ідентифікації:
Зображення символу #.

Вхідні параметри відповідають символу "?"
Результат ідентифікації:
Зображення символу ?.

Вхідні параметри відповідають символу "&"
Результат ідентифікації:
Зображення символу &.
```

Рисунок 8 — результати ідентифікації символів

Результати демонструють, що навчена нейронна мережа ефективно розпізнає та класифікує задані спеціальні символи. Завдяки оптимальним параметрам навчання, таким як 70 епох та коефіцієнт швидкості навчання 0.1, вдалося досягти високої точності мережі на рівні 93.54%. Отримані результати повністю задовольняють поставлені вимоги та цілі, визначені в рамках лабораторної роботи.

На основі проведеного аналізу результатів роботи програми, яка використовує модель глибокого навчання ResNet50 для автоматичного виявлення зброї на зображеннях, можна зробити висновок про її ефективність у розпізнаванні об'єктів, які відповідають категоріям "гвинтівка" або "пістолет".

Зокрема, для зображення "armed_soldiers.png", де, як очікується, присутні зображення військових осіб зі зброєю, програма успішно ідентифікувала наявність зброї. Це свідчить про те, що модель здатна ефективно аналізувати зображення і визначати на ньому предмети, характерні для класів, пов'язаних із зброєю.

У той же час, на зображенні "unarmed_soldiers.png", де військові особи не мають при собі зброї, програма не виявила ознак зброї. Це демонструє, що модель не лише ефективна у виявленні зброї, але й точна у випадках її відсутності, не допускаючи хибнопозитивних результатів, коли зброя помилково ідентифікується на зображеннях без неї.

Таким чином, використання попередньо навченої моделі ResNet50 дозволяє ефективно вирішувати задачі автоматичного виявлення зброї на зображеннях.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
На зображенні armed_soldiers.png виявлено зброю!
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
На зображенні unarmed_soldiers.png не виявлено зброї.
```

Рисунок 9 — результати ідентифікації людей зі зброєю

Рисунок 10 — військові зі зброєю

Рисунок 11 — військові без зброї

3.5. Програмний код.

Програмний код було оптимізовано та спрощено шляхом використання функціональних підходів та створення окремих підпрограм для раціоналізації обчислень. Операції проводилися над матрицями пікселів, які представляють бінарні растрові зображення.

Для реалізації програми було застосовано функціональні можливості бібліотек Python, зокрема numpy та matplotlib.

Для кращого розуміння структури та призначення окремих частин програмного коду, було додано пояснювальні коментарі, які розкривають сутність відповідних скриптів.

Для другого завдання використовується бібліотека TensorFlow разом із її компонентом Keras та моделлю глибокого навчання ResNet50, яка була попередньо навчена на датасеті ImageNet. Використовується модель ResNet50 з вагами, навченими на датасеті ImageNet, для подальшого використання в задачі виявлення зброї.

main.py

```
"""
Розробити програмний скрипт, який забезпечує ідентифікацію бінарних зображень 4 спеціальних символів,
заданих растровою матрицею. Для ідентифікації синтезувати, навчити та застосувати штучну нейронну мережу
в "сирому" вигляді реалізації матричних операцій.
Вибрані символи: +, #, ?, &.
"""

import numpy as np
import matplotlib.pyplot as plt


# Вхідні дані DataSet масиву
def create_input_data():
    """
    Вхідна частина навчального DataSet масиву.
    Формування вхідних бінарних даних графічних примітивів.
    :return: x - np.array
    """

    characters = [
        np.array([
            0, 0, 0, 0, 0, 0,
            0, 0, 1, 0, 0, 0,
            0, 0, 1, 0, 0, 0,
            1, 1, 1, 1, 1, 1,
            0, 0, 1, 0, 0, 0,
            0, 0, 1, 0, 0, 0,
            0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0,
        ]).reshape(8, 6),
        np.array([
            0, 1, 0, 1, 0, 1,
            0, 1, 0, 1, 0, 1,
            1, 1, 1, 1, 1, 1,
```

```
            0, 1, 0, 1, 0, 1,
            1, 1, 1, 1, 1, 1,
            0, 1, 0, 1, 0, 1,
            0, 1, 0, 1, 0, 1,
            0, 0, 0, 0, 0, 0,
        ]).reshape(8, 6),
        np.array([
            0, 1, 1, 1, 1, 0,
            1, 0, 0, 0, 0, 1,
            0, 0, 0, 0, 0, 1,
            0, 0, 0, 1, 1, 0,
            0, 0, 1, 0, 0, 0,
            0, 0, 0, 0, 0, 0,
            0, 0, 1, 0, 0, 0,
            0, 0, 0, 0, 0, 0,
        ]).reshape(8, 6),
        np.array([
            0, 0, 1, 1, 0, 0,
            0, 1, 0, 0, 1, 0,
            0, 1, 0, 0, 1, 0,
            0, 0, 1, 1, 0, 0,
            0, 1, 0, 0, 1, 1,
            1, 0, 0, 0, 1, 0,
            1, 0, 0, 0, 1, 1,
            0, 1, 1, 1, 0, 0,
        ]).reshape(8, 6)
    ]

    # Візуалізація
    fig, axs = plt.subplots(2, 2, figsize=(6, 6))
    for i, ax in enumerate(axs.flatten()):
        ax.imshow(characters[i], cmap='binary')
        ax.axis('off')
    plt.tight_layout()
    plt.show()

    # Вхідна частина навчального DataSet масиву
    input_data = [char.reshape(1, 48) for char in characters]

    return input_data


def create_output_data():
    """
    Вихідна частина навчального DataSet масиву - відповідь.
    Формування кодових комбінацій бінарних відповідей у просторі 4 значень.
    :return: y - np.array
    """

    # Вихідна частина навчального DataSet масиву
    output_data = [[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]
    return np.array(output_data)


# Побудова нейронної мережі

# Функція активації - сигмоїда
def sigmoid(x):
    """
    :param x: - np.array DataSet in
```

```
    :return: функція активації - сигмоїда
    """

    return 1 / (1 + np.exp(-x))


# Побудова нейронної мережі
def forward_propagation(x, weights1, weights2):
    """
    Головний компонент побудови нейронної мережі.
    Це та наступне відрізняє цей приклад від звичайного перцептрона.
    Архітектурні залежності:
    1-й рівень: вхідний рівень (1, 48);
    2-й шар: прихований шар (1, 4);
    3-й шар: вихідний рівень (4, 4).
    Графічне представлення архітектури дивись Neural_Networks_numpy_2.jpg

    :param x: np.array -
    :param weights1: початкові вагові коефіцієнти шару 1 (вхідного)
    :param weights2: початкові вагові коефіцієнти шару 2 (прихованого)
    :return: output_vals - вектор вихідних параметрів мережі - 4 компоненти
    """

    # Структура вхідного шару визначається простором вхідних параметрів x

    # Прихований шар
    hidden_inputs = x.dot(weights1)  # зважені вхідні параметри вхідного шару 1
    hidden_outputs = sigmoid(hidden_inputs)  # адитивна згортка - вихід з шару 1 - вхід
до шару 2

    # Вихідний шар
    final_inputs = hidden_outputs.dot(weights2)  # зважені вхідні параметри шару 2 до
вихідного шару
    final_outputs = sigmoid(final_inputs)  # вихідні параметри нейронної мережі

    return final_outputs


# Ініціалізація початкових значень вагових коефіцієнтів мережі методом рандомізації
def initialize_weights(rows, cols):
    weights = []
    for _ in range(rows * cols):
        weights.append(np.random.randn())
    return np.array(weights).reshape(rows, cols)


# Контроль навчання мережі за допомогою середньоквадратичної помилки (MSE)
def calculate_loss(output, target):
    squared_error = np.square(output - target)
    loss = np.sum(squared_error) / len(target)
    return loss


# Зворотне поширення похибки
def backpropagation(x, y, weights1, weights2, learning_rate):
    # Прихований шар
    hidden_inputs = x.dot(weights1)  # зважені вхідні параметри вхідного шару 1
    hidden_outputs = sigmoid(hidden_inputs)  # адитивна згортка - вихід з шару 1 - вхід
до шару 2
```

```
    # Вихідний шар
    final_inputs = hidden_outputs.dot(weights2)  # зважені вхідні параметри шару 2 до вихідного шару
    final_outputs = sigmoid(final_inputs)  # вихідні параметри нейронної мережі

    # Похибка на вихідному шарі
    output_errors = final_outputs - y
    hidden_errors = np.multiply((weights2.dot((output_errors.transpose()))).transpose(),
                                (np.multiply(hidden_outputs, 1 - hidden_outputs)))

    # Градієнт для weights1 та weights2
    weights1_gradients = x.transpose().dot(hidden_errors)
    weights2_gradients = hidden_outputs.transpose().dot(output_errors)

    # Оновлення параметрів з контролем помилки learning_rate
    weights1 -= learning_rate * weights1_gradients
    weights2 -= learning_rate * weights2_gradients

    return weights1, weights2


# Навчання мережі з контролем помилки learning_rate на епоху
def train_network(x, y, weights1, weights2, learning_rate=0.01, num_epochs=10):
    def update_weights(inputs, targets, w1, w2, lr):
        output = forward_propagation(inputs, w1, w2)
        loss = calculate_loss(output, targets)
        updated_w1, updated_w2 = backpropagation(inputs, targets, w1, w2, lr)
        return loss, updated_w1, updated_w2

    def train_epoch(epoch, data, labels, w1, w2, lr):
        epoch_loss, updated_w1, updated_w2 = zip(*[update_weights(x, y, w1, w2, lr) for x, y in zip(data, labels)])
        avg_loss = sum(epoch_loss) / len(data)
        accuracy = (1 - avg_loss) * 100
        print(f"Епоха: {epoch + 1}, Точність: {accuracy:.2f}%")
        return accuracy, avg_loss, updated_w1[-1], updated_w2[-1]

    accuracies, losses, trained_weights1, trained_weights2 = zip(*[train_epoch(epoch, x, y, weights1, weights2, learning_rate) for epoch in range(num_epochs)])
    return accuracies, losses, trained_weights1[-1], trained_weights2[-1]


# Ідентифікація символів / прогнозування
def predict_symbol(x, weights1, weights2):
    """
    Функція прогнозування приймає наступні аргументи:
    :param x: матриця зображення
    :param weights1: натреновані ваги
    :param weights2: натреновані ваги
    :return: відображає ідентифікований символ - графічну форму
    """

    def get_predicted_class(output):
        return max(range(len(output[0])), key=lambda i: output[0][i])

    def get_symbol(predicted_class):
        symbols = ["+", "#", "?", "&"]
        return symbols[predicted_class]
```

```
    output = forward_propagation(x, weights1, weights2)
    predicted_class = get_predicted_class(output)
    symbol = get_symbol(predicted_class)

    print(f"Зображення символу {symbol}.\n")
    plt.imshow(x.reshape(8, 6), cmap='binary')
    plt.show()

    return


if __name__ == '__main__':
    # Вхідні дані
    input_data = create_input_data()
    output_data = create_output_data()
    print('Масив DataSet: навчальна пара для навчання з учителем')
    print('Вхідні дані:', input_data, '\n')
    print('Вихідні дані:', output_data, '\n')

    # Ініціалізація вагових коефіцієнтів для 2 шарів
    layer_sizes = [(48, 4), (4, 4)]
    weights = [initialize_weights(*size) for size in layer_sizes]
    print('Ініціалізація вагових коефіцієнтів для 2 шарів')

    # Навчання мережі з контролем помилки learning_rate на епоху
    print('Навчання мережі з контролем помилки learning_rate на епоху')
    accuracies, losses, *trained_weights = train_network(input_data, output_data, *weights,
0.1, 70)

    # Контроль / візуалізація параметрів навчання
    training_metrics = [
        ('Точність', accuracies),
        ('Втрати', losses)
    ]
    for metric, data in training_metrics:
        plt.figure()
        plt.plot(data)
        plt.ylabel(metric)
        plt.xlabel("Епохи")
        plt.show()

    # Ідентифікація символів / прогнозування
    symbols = ["+", "#", "?", "&"]
    for i, symbol in enumerate(symbols):
        print(f'Вхідні параметри відповідають символу "{symbol}"')
        print('Результат ідентифікації:')
        predict_symbol(input_data[i], *trained_weights)
```

 # second.py

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array, load_img

# Завантаження попередньо навченої моделі ResNet50
```

```python
model = ResNet50(weights='imagenet')


# Функція для передбачення наявності зброї на зображенні
def detect_weapon(image_path):
    # Завантаження і препроцесинг зображення
    image = load_img(image_path, target_size=(224, 224))
    image = img_to_array(image)
    image = preprocess_input(image)
    image = np.expand_dims(image, axis=0)

    # Отримання передбачень моделі
    preds = model.predict(image)

    # Декодування передбачень
    decoded_preds = tf.keras.applications.resnet50.decode_predictions(preds, top=5)[0]

    # Перевірка наявності класу "гвинтівка" або "пістолет" в топ-5 передбачених класах
    weapon_detected = False
    for a, class_name, b in decoded_preds:
        if class_name in ['rifle', 'revolver']:
            weapon_detected = True
            break

    return weapon_detected

for image_path in ['armed_soldiers.png', 'unarmed_soldiers.png']:
    # Виявлення наявності зброї на зображенні
    if detect_weapon(image_path):
        print(f"На зображенні {image_path} виявлено зброю!")
    else:
        print(f"На зображенні {image_path} не виявлено зброї.")
```

3.6. Аналіз результатів відлагодження та верифікації результатів роботи програми.

Результати від лагодження та тестування довели працездатність розробленого коду.

Верифікація функціоналу програмного коду, порівняння отриманих результатів з технічними умовами завдання на лабораторну роботу доводять, що усі завдання виконані у повному обсязі.

IV. Висновки.

Під час лабораторної роботи було вивчено основні принципи та методи підготовки даних, створення, тренування та використання штучних нейронних мереж для вирішення практичних завдань розпізнавання образів у галузі комп'ютерного зору. Для реалізації цих завдань було використано високорівневу мову програмування Python.

У рамках роботи створено набір даних, що складається з бінарних зображень шести спеціальних символів. На основі цього набору даних була розроблена та навчена штучна нейронна мережа з нуля, без використання готових бібліотек або фреймворків.

Експериментальним шляхом було визначено, що найкращі результати навчання та точності мережі досягаються при використанні 70 епох та значення параметра

регуляризації 0.1. Отримані результати роботи нейронної мережі повністю відповідають поставленим завданням лабораторної роботи та демонструють ефективність застосування штучних нейронних мереж для задач ідентифікації та розпізнавання образів.

Щодо другого завдання, то програма змогла правильно ідентифікувати наявність зброї на зображенні "armed_soldiers.png" та її відсутність на "unarmed_soldiers.png". Це свідчить про високу точність моделі в розрізненні об'єктів, які є зброєю, від інших предметів.

Виконав: студент Панченко С.В.