Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №6

з дисципліни «Вступ до технології Data Science»

на тему

«РЕАЛІЗАЦІЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ (Artificial Neural Networks)»

Виконала:
студентка групи ІС-12
Павлова Софія

Перевірив:
Баран Д. Р.

Київ – 2023

1. Постановка задачі

Мета роботи:

Виявити дослідити та узагальнити особливості підготовки різних типів даних, синтезу, навчання та застосування штучних нейронних мереж (Artificial Neural Networks).

Завдання III рівня:

1. Відповідно до технічних умов, рис. 1, але в якості Data Set – обрати реальні дані у форматі числових / часових рядів, наприклад, як результат виконання лабораторних робіт із статистичного навчання (парсинг самостійно обраного сайту).
2. Класифікувати та ідентифікувати об’єкти в обраному відеопотоці з використанням технологій штучних нейронних мереж. Об’єкти, що підлягають ідентифікації та кокретику відеопотоку обрати самостійно, наприклад як вихідні дані лабораторної роботи із машинного навчання (обробка цифрових зображень).

1	Розробити програмний скрипт, що реалізує: 1. Модель зміни досліджуваного процесу – dataset за лінійним законом, у 1000 дискретних вимірів, з нормальним шумом (нульове середнє та СКВ похибки 10) та 40% аномальних вимірів з рівномірним розподілом в межах вибірки; 2. Штучну нейронну мережу з прогнозування даних відповідно до параметрів заданого в п.1 dataset; 3. Відображення процесу прогнозування у формі графіків. 4. Дослідити залежність точності прогнозування числового ряду від структури мережі.

Рисунок 1 – Технічні умови завдання 1

2. Виконання

2.1. Штучна нейронна мережа з прогнозування даних

Постановка задачі

Сформулюємо задачу з власного досвіду моделювання нейронних мереж для задачі прогнозування даних:

Задача:

Створити, навчити і апробувати багатошарову нейронну мережу для ухвалення рішення про зарахування до Університету абітурієнтів, які здали вступні іспити з математики, англійської та української мови.

Технічні умови задачі:

1. Модель зміни досліджуваного процесу — dataset з балами вступних іспитів абітурієнтів, які здали вступні іспити з математики, англійської та української мови.
2. Штучну нейронну мережу для ухвалення рішення про зарахування до Університету абітурієнтів, відповідно до параметрів заданого в п.1 dataset.
3. Відображення процесу прогнозування у формі графіків.
4. Дослідити залежність точності прогнозування числового ряду від структури мережі.

Правила прийому наступні:

1. Рейтинг абітурієнтів формується за формулою 0,4 БМ+0,3БА+0,3БУ, де БМ-бал з іспиту з математики, БА-бал з іспиту з англійської мови, БУ-бал з іспиту з української мови.
2. Мінімальний прохідний бал на вступ 160 для абітурієнтів без пільг.
3. З математики для абітурієнтів без пільг мінімальний бал іспиту не може бути менший 140 балів.
4. Абітурієнти, які мають пільги, зараховуються при мінімумі 120 балів з усіх іспитів і їх рейтинг не може бути меншим ніж 144 бали

5. Університет може прийняти на навчання 350 абітурієнтів, з них не більше 10% це абітурієнти з пільгами.
6. Статистика минулих років показує, що в середньому до Університету подають документи 1500 абітурієнтів.

Створення датасету

Так, як підходящого дата сету не було знайдено, створимо його самостійно.

Передбачимо для програми 2 режими роботи:

- генерація тренувального датасету;
- генерація тесту вального датасету.

Для нашого дата сету створимо поля: ПІБ, оцінка з математики, англійської та української, рейтинг та рішення. Рішення поставимо всім «відхилено».

Тут також наведені функції для генерації та визначення унікальності ПІБ, оскільки на стадії перевірки було помічено, що так, як нижче код видаляє з загального списку людей, які вже пройшли по квотах, опираючись на їх ПІБ, самі ПІБ не мають бути однакові, бо видаляться зайві люди. Утім надалі поле ПІБ не буде використовуватись,оскільки воно заважає навчанню моделей.

Генеруємо оцінки абітурієнтам від 100 до 200, для того, щоб була достатня кількість тих, хто набирає достатні бали для зарахування.

Лістинг коду:

```
import random
import pandas as pd

# Параметр кількості абітурієнтів
kilkist_abiturientiv = 1500

# Функція для перевірки унікальності ПІБ
def is_unique_name(pib, dataset):
    for abiturient in dataset:
        if pib == abiturient[0]:
            return False
    return True
```

```python
# Функція для генерації ПІБ
def generate_name():
    first_names = ['Павло', 'Оксана', 'Дмитро', 'Ольга', 'Михайло', 'Анастасія', 'Ірина',
                   'Оксана', 'Наталія', 'Марія', 'Анна', 'Катерина', 'Вікторія', 'Олена',
                   'Тетяна', 'Людмила', 'Ярослава', 'Іванна', 'Ганна', 'Лідія', 'Ніна',
                   'Софія', 'Ірина', 'Ольга', 'Марта', 'Юлія', 'Іван', 'Олександр',
                   'Михайло', 'Андрій', 'Володимир', 'Віктор', 'Сергій', 'Петро',
                   'Дмитро', 'Олег', 'Максим', 'Роман', 'Ярослав', 'Ігор', 'Богдан',
                   'Тарас', 'Артем', 'Ілля', 'Олексій', 'Анатолій']
    last_names = ['Дмитришенко', 'Загвойко', 'Неборачко', 'Товтиш', 'Ковтун', 'Алібіба',
                  'Іваниш', 'Оксаниш', 'Наталіїш', 'Маріїш', 'Анниш', 'Катериниш',
                  'Вікторіїш', 'Оленіш', 'Тетяниш', 'Людмилиш', 'Ярославиш', 'Іванниш',
                  'Ганниш', 'Лідіїш', 'Ніниш', 'Софіїш', 'Іриниш', 'Дмитришенко',
                  'Неборачко', 'Ковтун', 'Іваниш', 'Олександріш', 'Михайлиш', 'Андріїш',
                  'Володимириш', 'Вікторіш', 'Сергіїш', 'Петріш', 'Дмитріш', 'Олегіш',
                  'Максиміш', 'Романіш', 'Ярославіш', 'Ігоріш', 'Богданіш', 'Тарасіш',
                  'Артеміш', 'Ілляш', 'Олексіїш', 'Анатоліїш']

    return random.choice(first_names) + ' ' + random.choice(last_names)

# Функція для генерації датасетів
def generate_dataset(file_name):
    # Генерація датасету з 1500 абітурієнтів
    dataset = []
    for _ in range(kilkist_abiturientiv):
        pib = generate_name()
        while not is_unique_name(pib, dataset):
            pib = generate_name()
        pilga = random.choice([True, False])
        matematyka = random.randint(100, 200)
        angl_mova = random.randint(100, 200)
        ukr_mova = random.randint(100, 200)
        rating = 0.4 * matematyka + 0.3 * angl_mova + 0.3 * ukr_mova
        decision = 'незараховано'
        dataset.append([pib, pilga, matematyka, angl_mova, ukr_mova, rating, decision])

    [...]

# Головні виклики
descriptions = ['згенерувати тренувальний датасет', 'згенерувати тестовий датасет']

# Вибір режиму роботи програми
print('Оберіть режим роботи програми:')
for i in range(len(descriptions)):
    print(i + 1, '-', descriptions[i])
data_mode = int(input('mode:'))
# Якщо джерело даних існує
if data_mode in range(1, len(descriptions) + 1):

    # Тренувальний датасет
    if (data_mode == 1):
        for i in range(1, 4):
            generate_dataset(i)
            [...]

    # Тестовий датасет
    if (data_mode == 2):
        generate_dataset('test')
```

Перетворимо наш масив на дата фрейм і відсортуємо абітурієнтів в порядку спадання їх загального рейтингу .

Лістинг коду:

```
# Функція для генерації датасетів
def generate_dataset(file_name):
    # Генерація датасету з 1500 абітурієнтів
    [...]

    # Створення DataFrame з датасету
    columns = ['ПІБ', 'Пільги', 'Бал з математики', 'Бал з англійської', 'Бал з
української', 'Рейтинг', 'Рішення']
    df = pd.DataFrame(dataset, columns=columns)

    # Сортування за рейтингом у порядку спадання
    df.sort_values(by='Рейтинг', ascending=False, inplace=True)
```

Далі визначимо кількість абітурієнтів з пільгами та без, які можуть бути зараховані відповідно до умови.

Так, як пільговиків може бути до 10% від 350 (макс. кількість зарахованих), то їх може бути до 35. Решту місць, тобто «*350 – пільговики*» займають абітурієнти з загального конкурсу.

Лістинг коду:

```
# Функція для генерації датасетів
def generate_dataset(file_name):
    # Генерація датасету з 1500 абітурієнтів
    [...]

    # Створення DataFrame з датасету
    [...])

    # Сортування за рейтингом у порядку спадання
    [...]

    # Визначення кількість абітурієнтів з пільгами, які можуть бути зараховані
    kilkist_pilgovukiv = (df['Пільги'] == True).count()
    if kilkist_pilgovukiv > 35: kilkist_pilgovukiv = 35
    # Визначення загальної кількость абітурієнтів, які можуть бути зараховані
    if kilkist_abiturientiv >= 350:
        kilkist_zagalna = 350 - kilkist_pilgovukiv
    else:
        kilkist_zagalna = kilkist_abiturientiv - kilkist_pilgovukiv
```

Далі відберемо потрібну кількість пільговиків і виключимо з загального списку тих, хто не може бути зарахований, відповідно до умов та тих, хто вже отримав своє місце як пільговик. Потім за необхідності доберемо необхідну кількість абітурієнтів, щоб заповнити вступні місця до 350.

Лістинг коду:

```
# Функція для генерації датасетів
def generate_dataset(file_name):
    # Генерація датасету з 1500 абітурієнтів
    [...]

    # Створення DataFrame з датасету
    [...]

    # Сортування за рейтингом у порядку спадання
    [...]

    # Визначення кількість абітурієнтів з пільгами, які можуть бути зараховані
    [...]
    # Визначення загальної кількость абітурієнтів, які можуть бути зараховані
    [...]

    # Відберемо до 35 пільговиків
    pilgovyky_list = df[(df['Пільги'] == True) &
                        (df['Бал з математики'] >= 120) &
                        (df['Бал з англійської'] >= 120) &
                        (df['Бал з української'] >= 120) &
                        (df['Рейтинг'] >= 144)].head(kilkist_pilgovukiv)
    # Відберемо з загального списку тих, хто може бути зарахований як безпільговик
    general_list = df[(df['Бал з математики'] >= 140) &
                      (df['Рейтинг'] >= 160)]
    # Виключимо абітурієнтів, які вже пройшли по пільгах
    general_list = general_list[~general_list['ПІБ'].isin(pilgovyky_list['ПІБ'])]
    # Відберемо решту з загального списку, щоб у сумі було до 350 вступивших
    general_list = general_list.head(kilkist_zagalna)
```

Об’єднаємо список пільговиків з тими, хто пройшов без пільг у один список і призначимо їм рішення «зараховано». Таким чином ми отримуємо бажаний список абітурієнтів, яких зараховано.

Лістинг коду:

```
# Функція для генерації датасетів
def generate_dataset(file_name):
    # Генерація датасету з 1500 абітурієнтів
    [...]

    # Створення DataFrame з датасету
    [...]
```

```
    # Сортування за рейтингом у порядку спадання
    [...]

    # Визначення кількість абітурієнтів з пільгами, які можуть бути зараховані
    [...]
    # Визначення загальної кількость абітурієнтів, які можуть бути зараховані
    [...]

    # Відберемо до 35 пільговиків
    [...]

    # Відберемо з загального списку тих, хто може бути зарахований як безпільговик
    [...]
    # Виключимо абітурієнтів, які вже пройшли по пільгах
    [...]
    # Відберемо решту з загального списку, щоб у сумі було до 350 вступивших
    [...]

    # Об'єднання двох списків
    combined_list = pd.concat([general_list, pilgovyky_list])

    # Зміна рішення для всіх абітурієнтів з комбінованого списку
    combined_list['Рішення'] = 'зараховано'
```

Далі об’єднаємо список зарахований і не зарахованих студентів, щоб отримати бажаний дата сет з мітками, підходящий для навчання і тестування.

Для цього видалимо з початкового дата фрейму абітурієнтів, які вже зараховані і об’єднаємо дата фрейм з отриманим списком 350 зарахованих студентів.

Потім, для того, щоб відокремити дані від міток, видалимо останній стовпець новоствореного списку, щоб прибрати рішення про зарахування. Рішення буде зберігатись окремим файлом і використовуватись як мітка.

Лістинг коду:

```
# Функція для генерації датасетів
def generate_dataset(file_name):
    # Генерація датасету з 1500 абітурієнтів
    [...]

    # Створення DataFrame з датасету
    [...]
    # Сортування за рейтингом у порядку спадання
    [...]

    # Визначення кількість абітурієнтів з пільгами, які можуть бути зараховані
    [...]
    # Визначення загальної кількость абітурієнтів, які можуть бути зараховані
    [...]

    # Відберемо до 35 пільговиків
    [...]
    # Відберемо з загального списку тих, хто може бути зарахований як безпільговик
    [...]
```

```
    # Виключимо абітурієнтів, які вже пройшли по пільгах
    [...]
    # Відберемо решту з загального списку, щоб у сумі було до 350 вступивших
    [...]

    # Об'єднання двох списків
    [...]

    # Зміна рішення для всіх абітурієнтів з комбінованого списку
    [...]
    # Створення фінального датасету з мітками
    # Виключимо абітурієнтів, які вже пройшли
    ne_proysli = df[~df['ПІБ'].isin(combined_list['ПІБ'])]
    total_list = pd.concat([combined_list, ne_proysli])
    # Видалення останнього стовпця
    df_without_last_column = total_list.iloc[:, :-1]
```

Збережемо набір даних у таблицях **excel** (для наочності результатів) та **csv** (для роботи).

Лістинг коду:

```
# Функція для генерації датасетів
def generate_dataset(file_name):
    # Генерація датасету з 1500 абітурієнтів
    [...]

    # Створення DataFrame з датасету
    [...]

    # Сортування за рейтингом у порядку спадання
    [...]

    # Визначення кількість абітурієнтів з пільгами, які можуть бути зараховані
    [...]
    # Визначення загальної кількость абітурієнтів, які можуть бути зараховані
    [...]

    # Відберемо до 35 пільговиків
    [...]
    # Відберемо з загального списку тих, хто може бути зарахований як безпільговик
    [...]
    # Виключимо абітурієнтів, які вже пройшли по пільгах
    [...]
    # Відберемо решту з загального списку, щоб у сумі було до 350 вступивших
    [...]

    # Об'єднання двох списків
    [...]

    # Зміна рішення для всіх абітурієнтів з комбінованого списку
    [...]
    # Створення фінального датасету з мітками
    # Виключимо абітурієнтів, які вже пройшли
    [...]
    # Видалення останнього стовпця
    [...]
```

```
    # Збереження набору даних
    df_without_last_column.to_csv('dataset/{0}_dataset.csv'.format(file_name),
index=False)
    df_without_last_column.to_excel('dataset/{0}_dataset.xlsx'.format(file_name),
index=False, engine='openpyxl')
```

Аналогічно зробимо для останнього стовпця, тобто мітки набору даних збережемо окремо.

Лістинг коду:

```
# Функція для генерації датасетів
def generate_dataset(file_name):
    # Генерація датасету з 1500 абітурієнтів
    [...]

    # Створення DataFrame з датасету
    [...]

    # Сортування за рейтингом у порядку спадання
    [...]

    # Визначення кількість абітурієнтів з пільгами, які можуть бути зараховані
    [...]
    # Визначення загальної кількость абітурієнтів, які можуть бути зараховані
    [...]

    # Відберемо до 35 пільговиків
    [...]
    # Відберемо з загального списку тих, хто може бути зарахований як безпільговик
    [...]
    # Виключимо абітурієнтів, які вже пройшли по пільгах
    [...]
    # Відберемо решту з загального списку, щоб у сумі було до 350 вступивших
    [...]

    # Об'єднання двох списків
    [...]

    # Зміна рішення для всіх абітурієнтів з комбінованого списку
    [...]
    # Створення фінального датасету з мітками
    # Виключимо абітурієнтів, які вже пройшли
    [...]
    # Видалення останнього стовпця
    [...]

    # Збереження набору даних
    [...]

    # Вибір останнього стовпця
    last_column = total_list.iloc[:, -1]

    # Збереження міток
    last_column.to_csv('dataset/{0}_mark.csv'.format(file_name), index=False)
    last_column.to_excel('dataset/{0}_mark.xlsx'.format(file_name), index=False,
engine='openpyxl')

    return
```

Оскільки в завданні сказано, що зазвичай в університет подають документи близько 1500 вступників і зараховують з них лише 350 (10% з них квота), то для наочності тестувати нашу майбутню модель ми будемо саме на цій кількості абітурієнтів.

Тобто *x_test* та *y_test* будуть складатися з 1500 елементів масиву.

Тоді, *x_train* та *y_train* мають бути в 3 рази більші.

Тому, щоб не порушувати співвідношення, створимо функцію, що об'єднує 3 датасети з мітками по 1500 елементів у кожному у тренувальний.

Лістинг коду:

```
# Функція об'єднання датасетів у тренувальний
def concat_train_dataset():
    # Зчитування датасету
    df_1 = pd.read_csv('dataset/1_dataset.csv', header=None)
    df_2 = pd.read_csv('dataset/2_dataset.csv', header=None, skiprows=1)
    df_3 = pd.read_csv('dataset/3_dataset.csv', header=None, skiprows=1)

    # Об'єднання даних без повторення заголовків
    df_combined = pd.concat([df_1, df_2, df_3], ignore_index=True)
    # Збереження об'єднаного результату в новому CSV-файлі
    df_combined.to_csv('dataset/train_dataset.csv', index=False, header=None)
    df_combined.to_excel('dataset/train_dataset.xlsx', index=False, header=None)

    # Зчитування міток
    mark_1 = pd.read_csv('dataset/1_mark.csv', header=None)
    mark_2 = pd.read_csv('dataset/2_mark.csv', header=None, skiprows=1)
    mark_3 = pd.read_csv('dataset/3_mark.csv', header=None, skiprows=1)

    # Об'єднання міток без повторення заголовків
    mark_combined = pd.concat([mark_1, mark_2, mark_3], ignore_index=True)
    # Збереження об'єднаного результату в новому CSV-файлі
    mark_combined.to_csv('dataset/train_mark.csv', index=False, header=None)
    mark_combined.to_excel('dataset/train_mark.xlsx', index=False, header=None)

    return

# Головні виклики
[...]

# Вибір режиму роботи програми
[...]
# Якщо джерело даних існує
[...]

    # Тренувальний датасет
    if (data_mode == 1):
        for i in range(1, 4):
            generate_dataset(i)
        concat_train_dataset()

    [...]
```

Результат:

У результаті отримуємо наступні датасети.

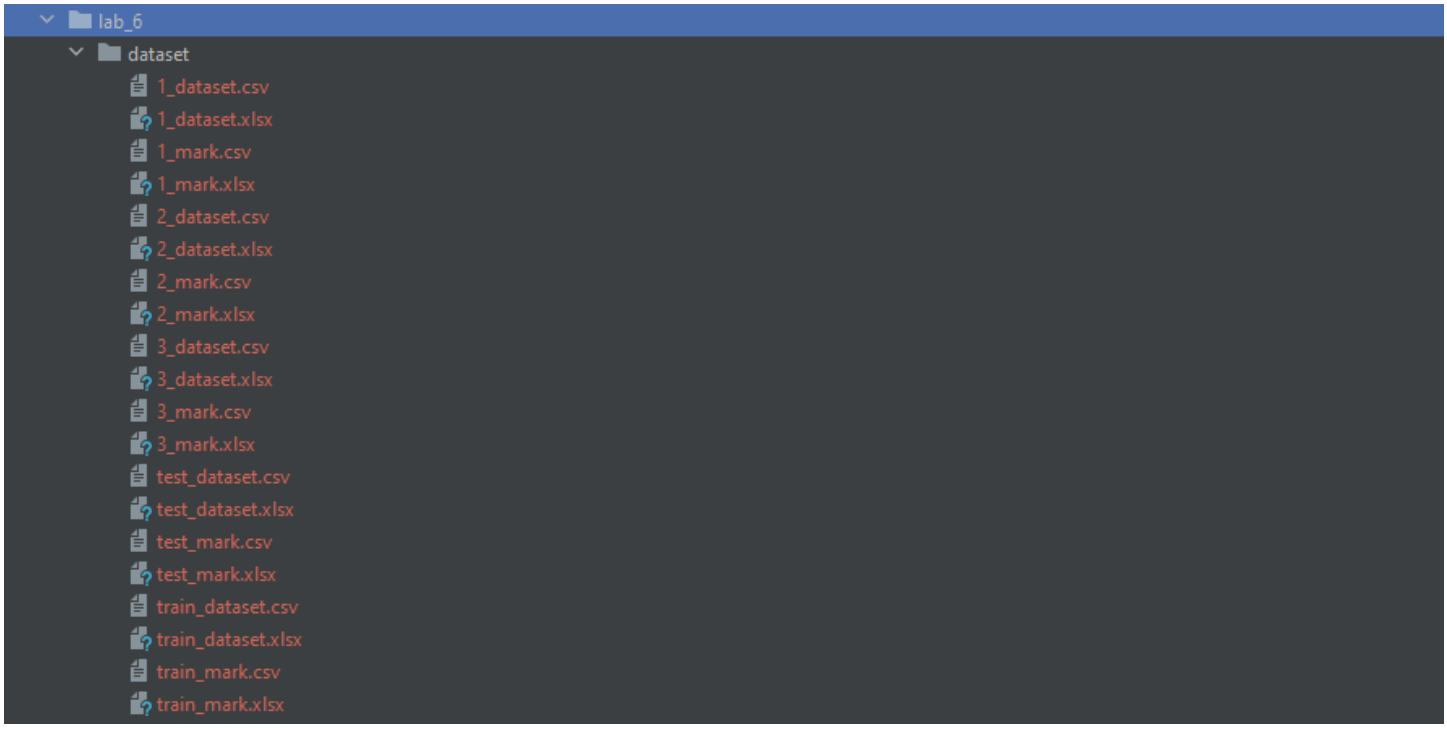


Рисунок 2 – Дерево датасетів

Датасет та мітки для тренування з 4500 абітурієнтів.

	A	B	C	D	E	F	G	H
1	ПІБ	Пільги	Бал з математики	Бал з англійської	Бал з української	Рейтинг		
2	Тарас Загвойко	False	199	193	199	197.2		
3	Андрій Петріш	False	186	200	198	193.8		
4	Андрій Товтиш	False	197	192	185	191.9		
5	Анастасія Анниш	False	197	187	189	191.6		
6	Олександр Іванниш	False	196	193	184	191.5		
7	Артем Андріїш	False	191	195	188	191.3		
8	Михайло Тетяниш	False	197	183	189	190.4		
9	Олександр Олексіїш	False	190	193	185	189.4		
10	Сергій Марїїш	False	178	198	196	189.39999999999998		
11	Оксана Тетяниш	False	191	174	198	188.0		
12	Юлія Тарасіш	False	200	175	182	187.1		
13	Вікторія Дмитришенко	False	200	157	200	187.1		
14	Олег Ярославиш	False	200	191	163	186.20000000000002		
15	Ігор Олегіш	False	195	167	192	185.7		

Рисунок 3 – Файл **train_dataset.xlsx**

	A	B	C	D	E	F	G	H	I	J	K	L
1	Рішення											
2	зараховано											
3	зараховано											
4	зараховано											
5	зараховано											
6	зараховано											
7	зараховано											
8	зараховано											
9	зараховано											
10	зараховано											
11	зараховано											
12	зараховано											
13	зараховано											
14	зараховано											
15	зараховано											

Рисунок 4 – Файл **train_mark.xlsx**

Датасет та мітки для тестування з 1500 абітурієнтів.

	A	B	C	D	E	F	G	H
1	**ПІБ**	**Пільги**	**Бал з математики**	**Бал з англійської**	**Бал з української**	**Рейтинг**		
2	Ольга Дмитріш	ЛОЖЬ	197	187	194	193,1		
3	Ірина Лідіїш	ЛОЖЬ	195	195	181	190,8		
4	Павло Лідіїш	ЛОЖЬ	191	182	193	188,9		
5	Олексій Неборачко	ЛОЖЬ	196	196	172	188,8		
6	Ніна Олексіїш	ЛОЖЬ	200	195	167	188,6		
7	Михайло Неборачко	ЛОЖЬ	192	170	198	187,2		
8	Роман Алібіба	ЛОЖЬ	192	195	173	187,2		
9	Тетяна Неборачко	ЛОЖЬ	188	196	177	187,1		
10	Анатолій Вікторіш	ЛОЖЬ	199	177	181	187		
11	Олексій Оксаниш	ЛОЖЬ	190	172	191	184,9		
12	Людмила Олениш	ЛОЖЬ	194	192	165	184,7		
13	Петро Ярославиш	ЛОЖЬ	182	175	197	184,4		
14	Віктор Оксаниш	ЛОЖЬ	193	170	186	184		
15	Павло Романіш	ЛОЖЬ	197	175	175	183,8		

Рисунок 4 – Файл **test_dataset.xlsx**

	A	B	C	D	E	F	G	H	I	J	K	L
1	**Рішення**											
2	зараховано											
3	зараховано											
4	зараховано											
5	зараховано											
6	зараховано											
7	зараховано											
8	зараховано											
9	зараховано											
10	зараховано											
11	зараховано											
12	зараховано											
13	зараховано											
14	зараховано											
15	зараховано											

Рисунок 5 – Файл **test_mark.xlsx**

Створення та навчання штучної нейронної мережі

Для даної задачі оберемо 4 типи нейронної мережі:

- **Sequential**,
- **Functional API**,
- **Recurrent Neural Networks**,
- **Convolutional Neural Networks**.

Пройдемось коротко по їх особливостям:

Sequential Model (Послідовна модель): Це найпростіший тип нейронної мережі, де шари послідовно додаються один за одним.

Functional API (Функціональний API): Цей підхід дає більшу гнучкість при створенні нейронних мереж з складнішою архітектурою, такою як моделі з відгалуженнями або з'єднаннями.

Convolutional Neural Networks (CNN): Цей тип нейронної мережі спеціально призначений для роботи з великими об'ємами даних, такими як зображення. Шари CNN використовуються для виявлення локальних шаблонів у вхідних даних.

Recurrent Neural Networks (RNN): Цей тип нейронної мережі зазвичай використовується для роботи з послідовними даними, такими як текст або часові ряди. Шари RNN мають зв'язки з попередніми станами, що дозволяє нейронній мережі утримувати інформацію про попередні елементи послідовності.

Створимо, навчимо та протестуємо всі 4 мережі.

Підготовка даних

Виконаємо завантаження, первинну обробку та нормалізацію даних у межах від 0 до 1.

Реалізуємо програму таким чином, щоб користувач міг обирати, який тип нейронної мережі використовувати.

Лістинг коду:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from keras import models
from keras import layers
from keras.models import Model, Sequential
from keras.layers import Input, Dense, LSTM, Conv1D, MaxPooling1D, Flatten
import matplotlib.pyplot as plt
import numpy as np

# Усунення warning
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

# Завантаження даних
def data_load():
    x_train = pd.read_csv(r'dataset/train_dataset.csv')
    y_train = pd.read_csv(r'dataset/train_mark.csv')
    x_test = pd.read_csv(r'dataset/test_dataset.csv')
    y_test = pd.read_csv(r'dataset/test_mark.csv')

    return x_train, y_train, x_test, y_test

# Первинна обробка даних
def data_procesing(x_train, x_test, y_train, y_test):
    # Видалення стовпця з іменами
    x_train = x_train.iloc[:, 1:]
    x_test = x_test.iloc[:, 1:]
    # Перетворення рядкових значень на числові
    y_train = y_train.replace({'зараховано': 0, 'незараховано': 1}).to_numpy()
    y_test = y_test.replace({'зараховано': 0, 'незараховано': 1}).to_numpy()
    # Перетворення типу даних
    x_train = x_train.astype('float32').to_numpy()
    x_test = x_test.astype('float32').to_numpy()
```

```
    return x_train, y_train, x_test, y_test

# Нормалізація даних
def normalize_data(x_train, x_test):
    scaler = StandardScaler()
    x_train = scaler.fit_transform(x_train)
    x_test = scaler.transform(x_test)

    return x_train, x_test

# Головні виклики

descriptions = ['Sequential', 'Functional', 'CNN', 'RNN']

# Вибір режиму роботи програми
print('Оберіть тип нейронної мережі:')
for i in range(len(descriptions)):
    print(i + 1, '-', descriptions[i])
data_mode = int(input('mode:'))
# Якщо джерело даних існує
if data_mode in range(1, len(descriptions) + 1):

    # Завантаження даних
    x_train, y_train, x_test, y_test = data_load()

    # Первинна обробка даних
    x_train, y_train, x_test, y_test = data_procesing(x_train, x_test, y_train, y_test)

    # Нормалізація даних
    x_train, x_test = normalize_data(x_train, x_test)
```

Архітектура мереж

Створимо всі 4 мережі, що відрізнятимуться архітектурою з метою подальшого аналізу впливу архітектури мережі на результати.

Лістинг коду:

```
# Архітектура моделі
def network_architecture(data_mode):
    # Архітектура - Sequential
    if (data_mode == 1):
        network = models.Sequential()
        network.add(layers.Dense(16, activation='relu', input_dim=5))
        network.add(layers.Dense(8, activation='relu'))
        network.add(layers.Dense(1, activation='sigmoid'))
    # Архітектура - Functional
    elif (data_mode == 2):
        input_layer = Input(shape=(5,))
        hidden_layer1 = Dense(16, activation='relu')(input_layer)
        hidden_layer2 = Dense(8, activation='relu')(hidden_layer1)
        output_layer = Dense(1, activation='sigmoid')(hidden_layer2)
        network = Model(inputs=input_layer, outputs=output_layer)
    # Архітектура - CNN
    elif (data_mode == 3):
        network = Sequential()
        network.add(Conv1D(32, 2, activation='relu', input_shape=(x_train.shape[1], 1)))
```

```
        network.add(MaxPooling1D(1))
        network.add(Conv1D(64, 2, activation='relu'))
        network.add(Flatten())
        network.add(Dense(64, activation='relu'))
        network.add(Dense(1, activation='sigmoid'))
    # Архітектура - RNN
    else:
        network = Sequential()
        network.add(LSTM(32, activation='relu', input_shape=(x_train.shape[1], 1)))
        network.add(Dense(1, activation='sigmoid'))
    print('\nАрхітектура моделі')
    network.summary()

    return network
```

Результат:

У результаті отримаємо 4 мережі з різною архітектурою та призначенням.

```
Архітектура моделі
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 16)                96

 dense_1 (Dense)             (None, 8)                 136

 dense_2 (Dense)             (None, 1)                 9

=================================================================
Total params: 241 (964.00 Byte)
Trainable params: 241 (964.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рисунок 6 – Архітектура мережі **Sequential**

Створено нейромережу яка має **вхідний шар з 5 вхідними ознаками**, **два приховані шари з 16 та 8 нейронами** відповідно і **вихідний шар з одним нейроном** для бінарної класифікації.

```
Архітектура моделі
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 5)]               0

 dense (Dense)               (None, 16)                96

 dense_1 (Dense)             (None, 8)                 136

 dense_2 (Dense)             (None, 1)                 9

=================================================================
Total params: 241 (964.00 Byte)
Trainable params: 241 (964.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рисунок 7 – Архітектура мережі **Functional**

Створено мережу з **трьома повнозв'язними шарами**, де **вихід одного шару є входом для наступного**. Вхідні дані передаються через вхідний шар, проходять через **два приховані шари** з функцією активації **'relu'** та виходять через вихідний шар з функцією активації **'sigmoid'.**

```
Архітектура моделі
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 4, 32)             96

 max_pooling1d (MaxPooling1  (None, 4, 32)             0
 D)

 conv1d_1 (Conv1D)           (None, 3, 64)             4160

 flatten (Flatten)           (None, 192)               0

 dense (Dense)               (None, 64)                12352

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 16673 (65.13 KB)
Trainable params: 16673 (65.13 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рисунок 8 – Архітектура мережі **CNN**

Створено мережу з використанням **1D згорткових шарів** для обробки послідовних даних. Після згорткових шарів використовуються **шар пулінгу** та **повнозв'язні шари**.

```
Архітектура моделі
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 32)                4352

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 4385 (17.13 KB)
Trainable params: 4385 (17.13 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рисунок 9 – Архітектура мережі **RNN**

Створено мережу з **LSTM шаром** та після нього є **повнозв'язний шар з одним нейроном**.

Компіляція та навчання мереж

Навчимо мережі на 10 епохах. Відобразимо процес навчання у вигляді графіків та виведемо графік зміни помилки.

Лістинг коду:

```
# Компіляція та навчання мережі
def compile_train(network, x_train, y_train, x_test, y_test):
    network.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    print('\nНавчання моделі')

    epochs = 10
    batch_size = 32

    mse_train_values = []  # Зберігатиме значення MSE для тренування
    mse_test_values = []   # Зберігатиме значення MSE для тестування
    iterations = []        # Зберігатиме номери ітерацій

    for epoch in range(epochs):
        # Навчання моделі протягом кожної епохи
        history = network.fit(x_train, y_train, epochs=1, batch_size=batch_size,
verbose=0)

        # Отримання значення MSE для тренування та тестування
        mse_train = history.history['loss'][0]
        mse_test = network.evaluate(x_test, y_test, verbose=0)[0]

        # Зберігання значень MSE та номерів ітерацій
        mse_train_values.append(mse_train)
        mse_test_values.append(mse_test)
        iterations.append(epoch)

        # Відображення інформації про поточну епоху
        print(f'Epoch {epoch + 1}/{epochs},\tТренувальна MSE: {mse_train},\tТестова MSE:
{mse_test}')

        # Відображення графіка навчання та прогнозування
        pred = network.predict(x_test)
        plt.plot(y_test, label='Реальні')
        plt.plot(pred, label='Прогнозовані')
        plt.title(f'Епоха {epoch + 1}')
        plt.xlabel('Кількість абітурінтів')
        plt.ylabel('Рішення')
        plt.legend()
        plt.show()

    # Відображення графіків
    plt.plot(iterations, mse_train_values, label='Тренувальна MSE')
    plt.plot(iterations, mse_test_values, label='Тестова MSE')
    plt.title('Динаміка навчання')
    plt.xlabel('Епохи')
    plt.ylabel('MSE')
```

```
    plt.legend()
    plt.show()

    return network

# Головні виклики

[...]

# Вибір режиму роботи програми
[...]
# Якщо джерело даних існує
    [...]

    # Компіляція та навчання мережі
    network = compile_train(network, x_train, y_train, x_test, y_test)
```

Результат:

У результаті отримаємо візулізацію навчання мереж у графіках.

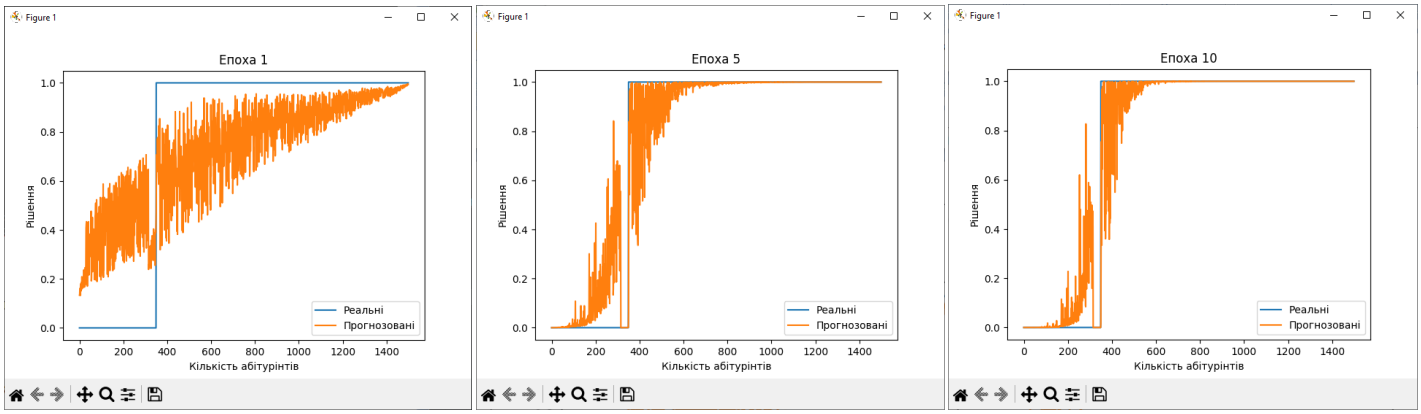


Рисунок 10 – Навчання мережі **Sequential**

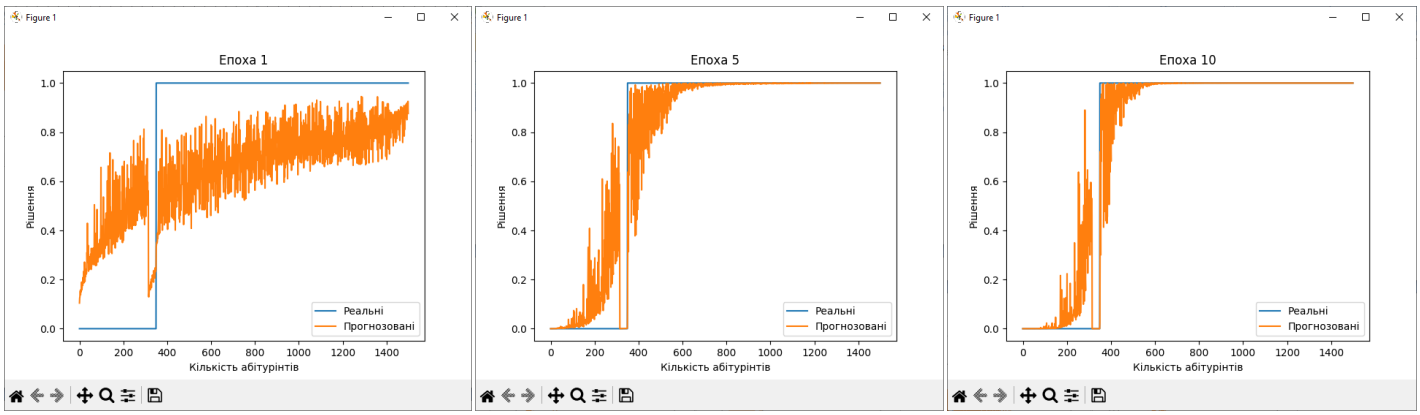


Рисунок 11 – Навчання мережі **Functional**

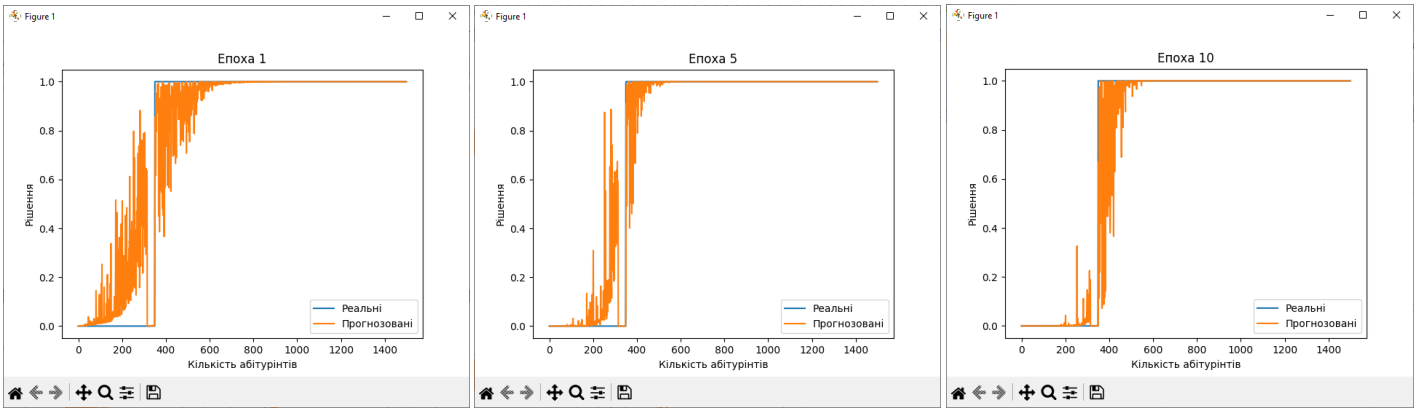


Рисунок 12 – Навчання мережі **CNN**

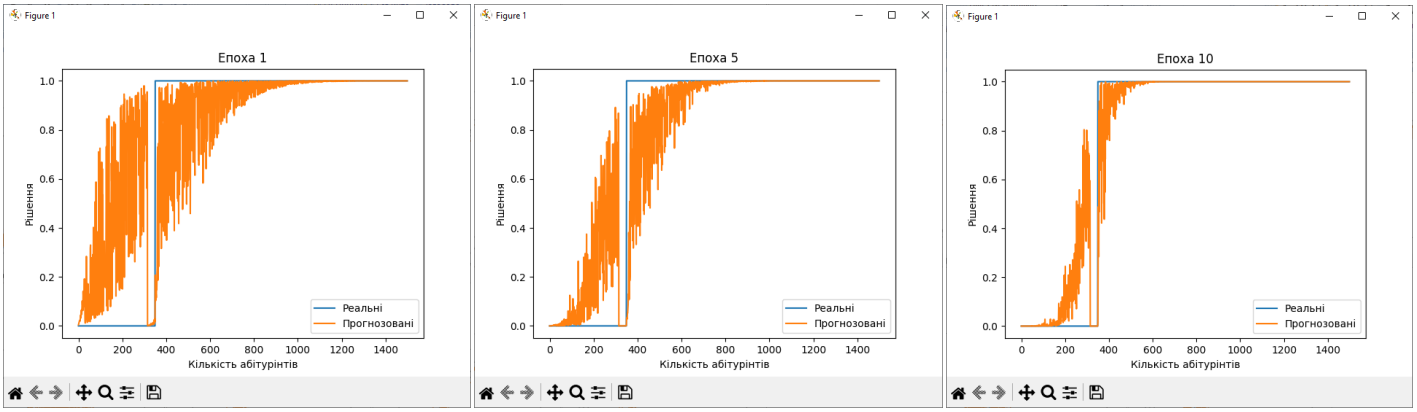


Рисунок 13 – Навчання мережі **RNN**

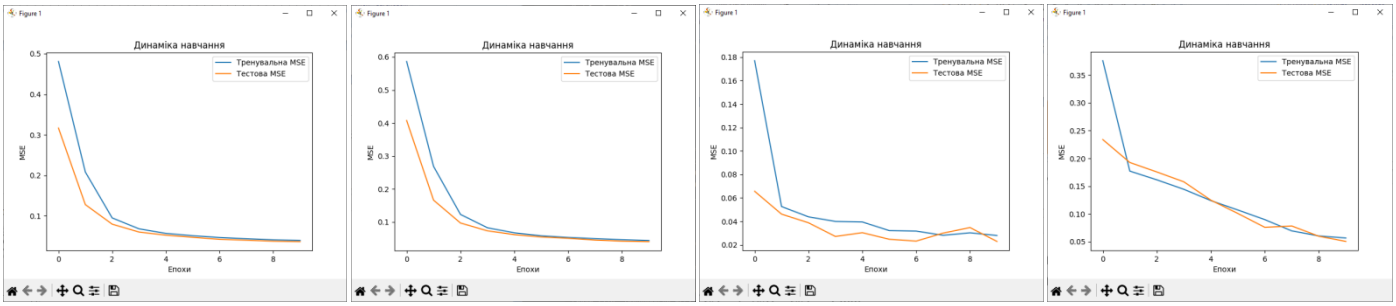


Рисунок 14 – Графіки втрат мереж **Sequential, Functional, CNN, RNN**

Тестування мереж

Протестуємо всі 4 мережі на тестовому датасеті та порівняємо результати.

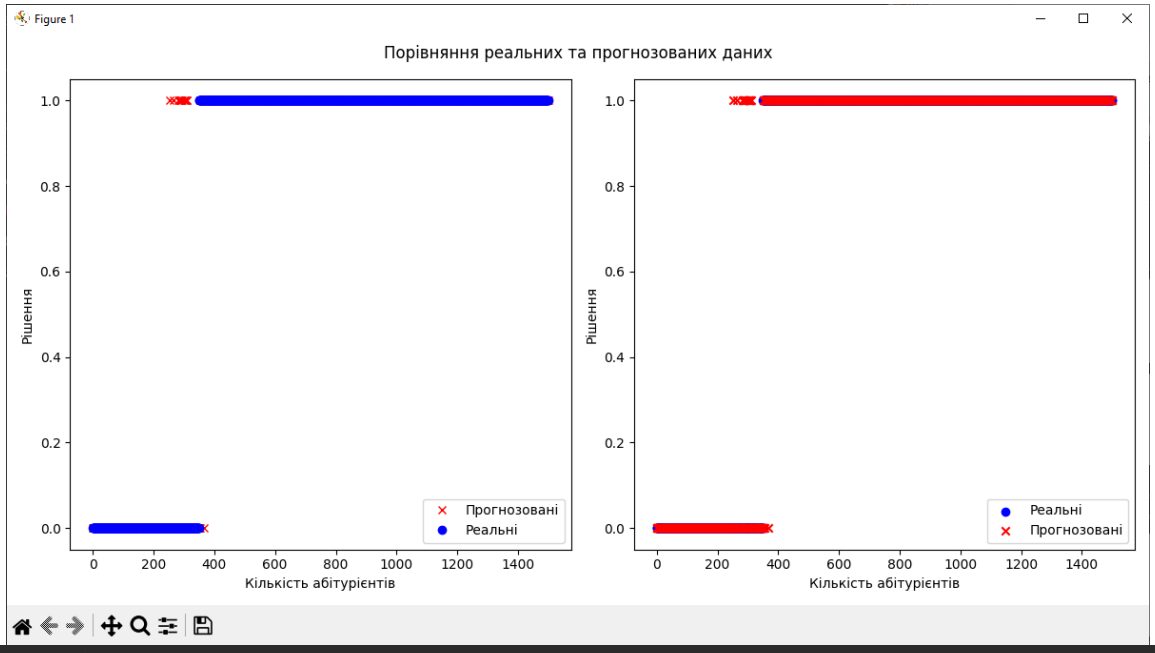
Лістинг коду:

```
# Перевірка моделі
def test(network, data_mode):
    print('\nПеревірка моделі')
```

```
    predicted_probabilities = network.predict(x_test)
    predicted_classes = (predicted_probabilities > 0.5).astype(int)
    test_results = pd.DataFrame({'Predicted': predicted_classes.flatten(), 'Actual':
y_test.flatten()})
    test_loss, test_acc = network.evaluate(x_test, y_test)

    # Збереження результатів у таблицю
    test_results.to_excel('model_{0}.xlsx'.format(data_mode), index=False)

    # Графік для оцінки точності
    plt.figure(figsize=(12, 6))
    # Перший графік
    plt.subplot(1, 2, 1)
    plt.plot(test_results.index, test_results['Predicted'], label='Прогнозовані',
marker='x', linestyle='None', color='r')
    plt.plot(test_results.index, test_results['Actual'], label='Реальні', marker='o',
linestyle='None', color='b')
    plt.xlabel('Кількість абітурієнтів')
    plt.ylabel('Рішення')
    plt.legend()
    # Другий графік
    plt.subplot(1, 2, 2)
    plt.scatter(np.arange(len(y_test)), y_test, label='Реальні', marker='o', color='b')
    plt.scatter(np.arange(len(predicted_classes)), predicted_classes,
label='Прогнозовані', marker='x', color='r')
    plt.xlabel('Кількість абітурієнтів')
    plt.ylabel('Рішення')
    plt.legend()
    # Відображення графіків
    plt.suptitle('Порівняння реальних та прогнозованих даних')
    plt.tight_layout()
    plt.show()

    return

# Головні виклики

[...]

# Вибір режиму роботи програми
[...]
# Якщо джерело даних існує
    [...]

    # Перевірка мережі
    test(network, data_mode)
```

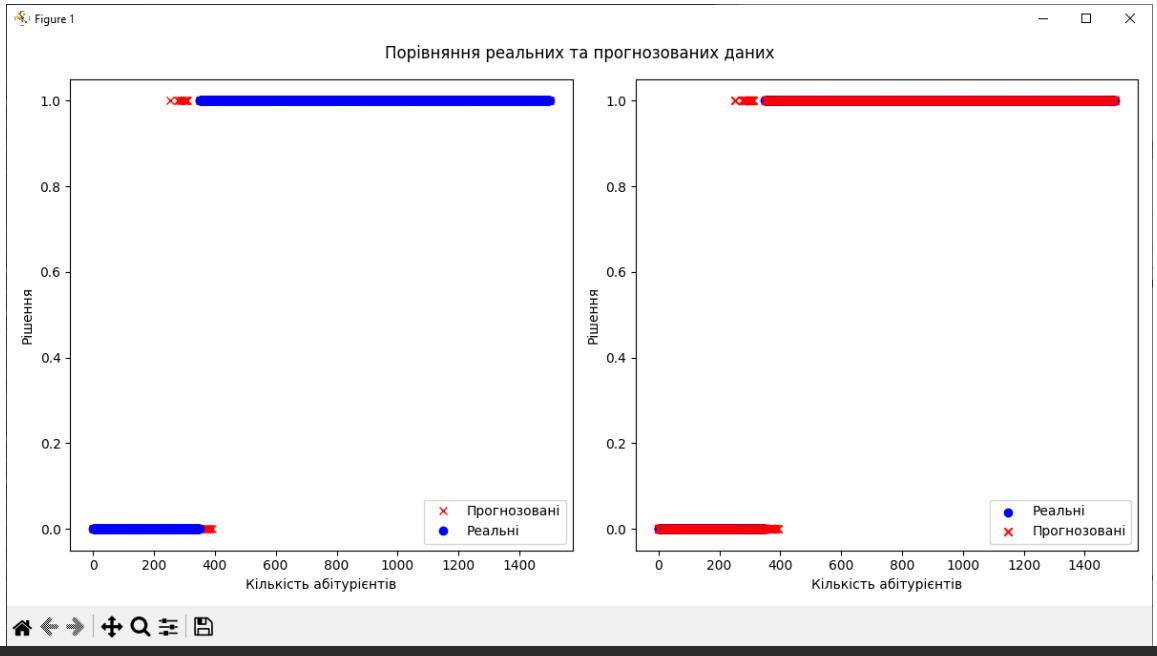
Результат:

У результаті отримуємо наступну точність моделі та порівняльні графіки результатів.



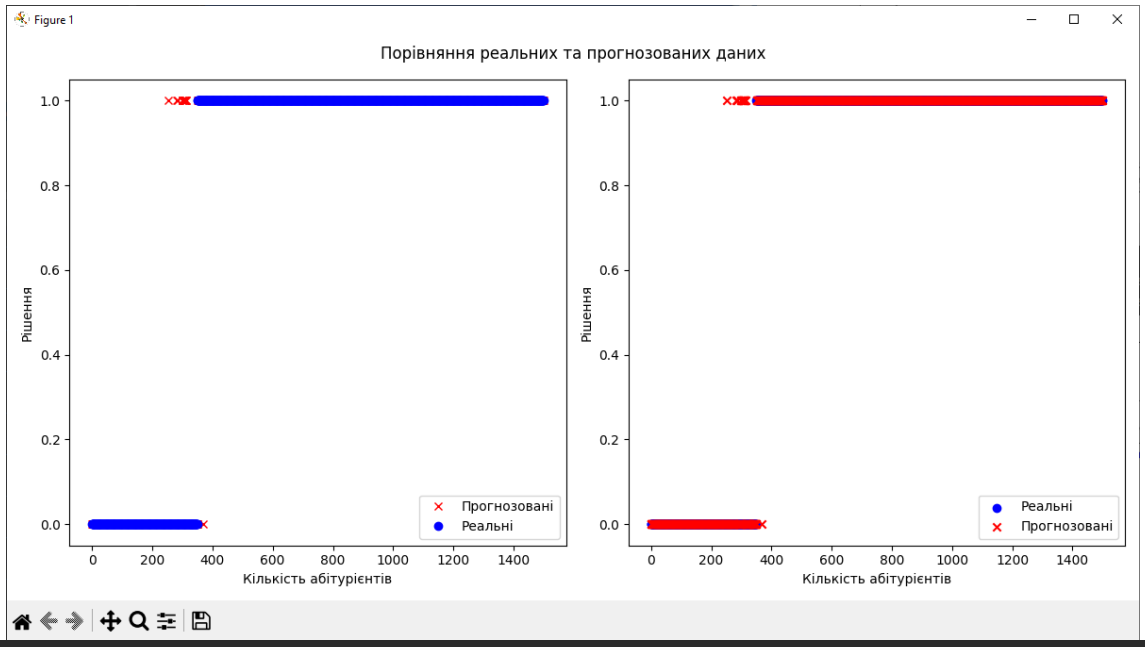

```
Перевірка моделі
47/47 [==============================] - 0s 1ms/step
47/47 [==============================] - 0s 1ms/step - loss: 0.0460 - accuracy: 0.9880
```

Рисунок 15 – Точність мережі **Sequential**



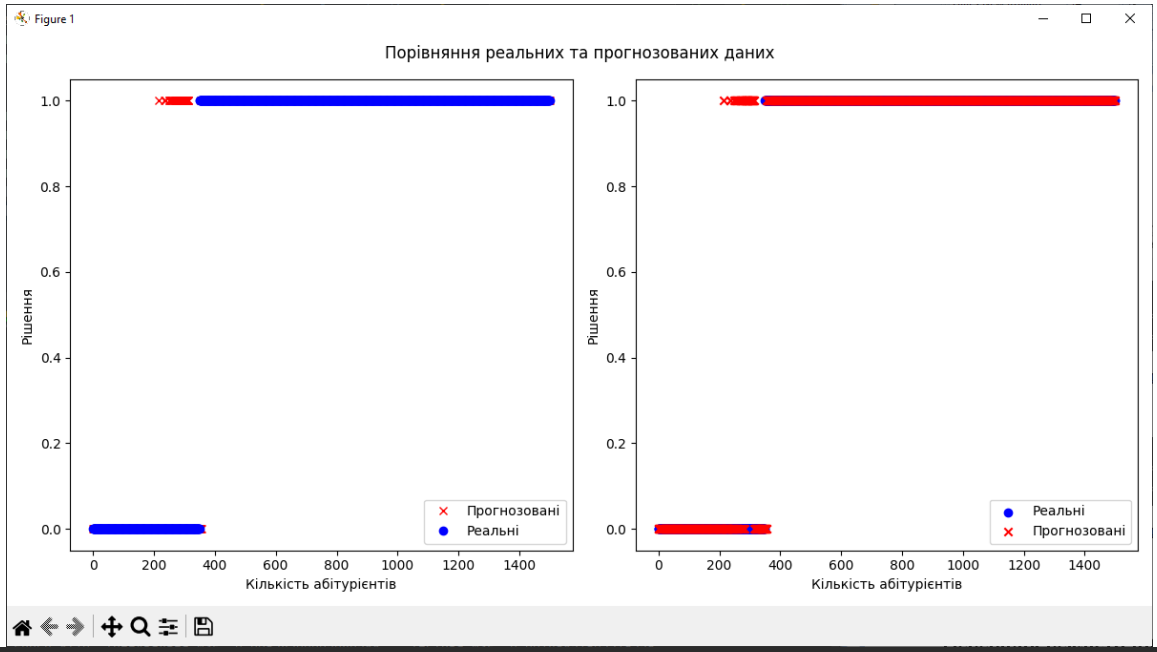

```
Перевірка моделі
47/47 [==============================] - 0s 1ms/step
47/47 [==============================] - 0s 1ms/step - loss: 0.0402 - accuracy: 0.9847
```

Рисунок 16 – Точність мережі **Functional**




```
Перевірка моделі
47/47 [==============================] - 0s 1ms/step
47/47 [==============================] - 0s 1ms/step - loss: 0.0253 - accuracy: 0.9893
```

Рисунок 17 – Точність мережі **CNN**




```
Перевірка моделі
47/47 [==============================] - 0s 2ms/step
47/47 [==============================] - 0s 2ms/step - loss: 0.0771 - accuracy: 0.9607
```

Рисунок 18 – Точність мережі **RNN**

Бачимо, що в середньому найкраще справилась **CNN** мережа, але слід також перевірити кожен клас прогнозування на відповідність вимогам задачі.

Перевірка реалістичності розв'язку

Після виконання програми отримуємо 4 файли з результатами класифікації.

Рисунок 19 – Дерево результатів класифікації

Файл з результатами виглядає наступним чином.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Predict	Actual										
255	1	0										
267	1	0										
284	1	0										
288	1	0										
289	1	0										
292	1	0										

Рисунок 20 – Вигляд файлу з результатами

Перевіримо, **скільки абітурієнтів мали бути зараховані**, але модель класифікувала їх як «*не зараховано*». Для цього виберемо по стовпцю «*Actual*» за допомогою фільтру лише «*0*» (0 – зараховано, 1 – не зараховано) і просумуємо обидва стовпці по фільтру. Таким чином ми побачимо скількох абітурієнтів помилково не зарахували.

348	0	0											
349	0	0											
350	0	0											
351	0	0											
1502													
1503	14	0		14									

Рисунок 21 – Помилково **не зараховані** абітурієнти – мережа **Sequential**

348	0	0											
349	0	0											
350	0	0											
351	0	0											
1502													
1503	13	0		13									

Рисунок 22 – Помилково **не зараховані** абітурієнти – мережа **Functional**

348	0	0											
349	0	0											
350	0	0											
351	0	0											
1502													
1503	14	0		14									

Рисунок 23 – Помилково **не зараховані** абітурієнти – мережа **CNN**

348	0	0											
349	0	0											
350	0	0											
351	0	0											
1502													
1503	53	0		53									

Рисунок 24 – Помилково **не зараховані** абітурієнти – мережа **RNN**

Перевіримо, **скільки абітурієнтів мали бути не зараховані**, але модель класифікувала їх як «*зараховано*». Для цього виберемо по стовпцю «*Actual*» за допомогою фільтра лише «*1*» (0 – зараховано, 1 – не зараховано) і просумуємо обидва стовпці по фільтру. У правій клітинці знайдемо різницю двох сум. Таким чином ми побачимо скількох абітурієнтів помилково зарахували.

1498	1	1											
1499	1	1											
1500	1	1											
1501	1	1											
1502													
1503	1146	1150		-4									

Рисунок 25 – Помилково **зараховані** абітурієнти – мережа **Sequential**

1498	1	1											
1499	1	1											
1500	1	1											
1501	1	1											
1502													
1503	1140	1150		-10									

Рисунок 26 – Помилково **зараховані** абітурієнти – мережа **Functional**

1498	1	1											
1499	1	1											
1500	1	1											
1501	1	1											
1502													
1503	1148	1150		-2									

Рисунок 27 – Помилково **зараховані** абітурієнти – мережа **CNN**

1498	1	1											
1499	1	1											
1500	1	1											
1501	1	1											
1502													
1503	1144	1150		-6									

Рисунок 28 – Помилково **зараховані** абітурієнти – мережа **RNN**

Отже помилка мереж складає:

Таблиця 1 – Помилки мереж

Тип мережі:	***Sequential***	***Functional***	***CNN***	***RNN***
Помилково **не зараховані**	14	13	14	53
Помилково **зараховані**	4	10	2	6
Сумарна помилка	18	23	16	59

Бачимо, що результати порівняння даних підтверджують результат тестування *accuracy*. Найкращою для даної задачі виявилась мережа типу **CNN**.

2.2. Класифікація та ідентифікація об’єктів у відеопотоці

Візьмемо задачу з попередньої лабораторної роботи, де було створено **object tracker** для допомоги поліції міста Нью-Йорка відловлювати крадіїв.

Отже задача для покращення реалізованої програми – ідентифікувати людей у натовпі на вулицях.

Модель

Для цього використаємо попередньо навчену модель **YOLOv8N**.

Датасет

Для додаткового навчання використаємо датасет **COCO128**.

Навчання

Довчимо модель на 3-х епохах.

Лістинг коду:

```python
import os
os.environ['KMP_DUPLICATE_LIB_OK'] = 'True'
from ultralytics import YOLO

# Зававантаження попередньо навченої моделі
model = YOLO('yolov8n.pt')

# Донавчання моделі
results = model.train(data='coco128.yaml', epochs=3)
```

Результат:

```
Logging results to runs\detect\train4
Starting training for 3 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
        1/3         0G      1.096      1.365      1.202        201        640: 100%|██████████| 8/8 [01:23<00
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/
                   all        128        929      0.647      0.518      0.595      0.441
  0%|          | 0/8 [00:00<?, ?it/s]
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
        2/3         0G      1.216      1.443      1.268        136        640: 100%|██████████| 8/8 [01:08<00
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/
                   all        128        929       0.66      0.542      0.614      0.455
  0%|          | 0/8 [00:00<?, ?it/s]
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
        3/3         0G      1.193      1.342      1.243        206        640: 100%|██████████| 8/8 [01:07<00
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/
                   all        128        929      0.664      0.548      0.624      0.464

3 epochs completed in 0.086 hours.
Optimizer stripped from runs\detect\train4\weights\last.pt, 6.5MB
Optimizer stripped from runs\detect\train4\weights\best.pt, 6.5MB
```

Рисунок 29 – Донавчання моделі

У результаті навчання модель зберігла натреновані ваги до файлів **last.pt** та **best.pt**.

Валідація

Проведемо валідацію моделі за допомогою трекера **bytetracker**.

Лістинг коду:

```python
[...]

# Оцінка продуктивності моделі на валідаційному наборі
results = model.val()
```

Результат:

```
Validating runs\detect\train4\weights\best.pt...
Ultralytics YOLOv8.0.226 🚀 Python-3.9.5 torch-2.1.1+cpu CPU (Intel Core(TM) i7-9750H 2.60GHz)
Model summary (fused): 168 layers, 3151904 parameters, 0 gradients, 8.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████|
                   all        128        929      0.664      0.548      0.623      0.463
                person        128        254      0.796      0.669      0.762      0.546
               bicycle        128          6      0.613      0.333      0.321      0.279
                   car        128         46      0.701      0.217      0.278      0.171
            motorcycle        128          5      0.675      0.834      0.938      0.764
              airplane        128          6      0.749      0.667      0.846      0.588
                   bus        128          7      0.529      0.645      0.657      0.595
                 train        128          3      0.538      0.667      0.806      0.698
                 truck        128         12      0.904      0.333      0.485      0.296
                  boat        128          6      0.299      0.167      0.379      0.242
         traffic light        128         14      0.691      0.214      0.206       0.14
             stop sign        128          2          1       0.94      0.995      0.703
```

Рисунок 30 – Валідація моделі

Результати валідації збережені у файлах **val_batch_pred.jpg** та **val_batch_lables.jpg**.

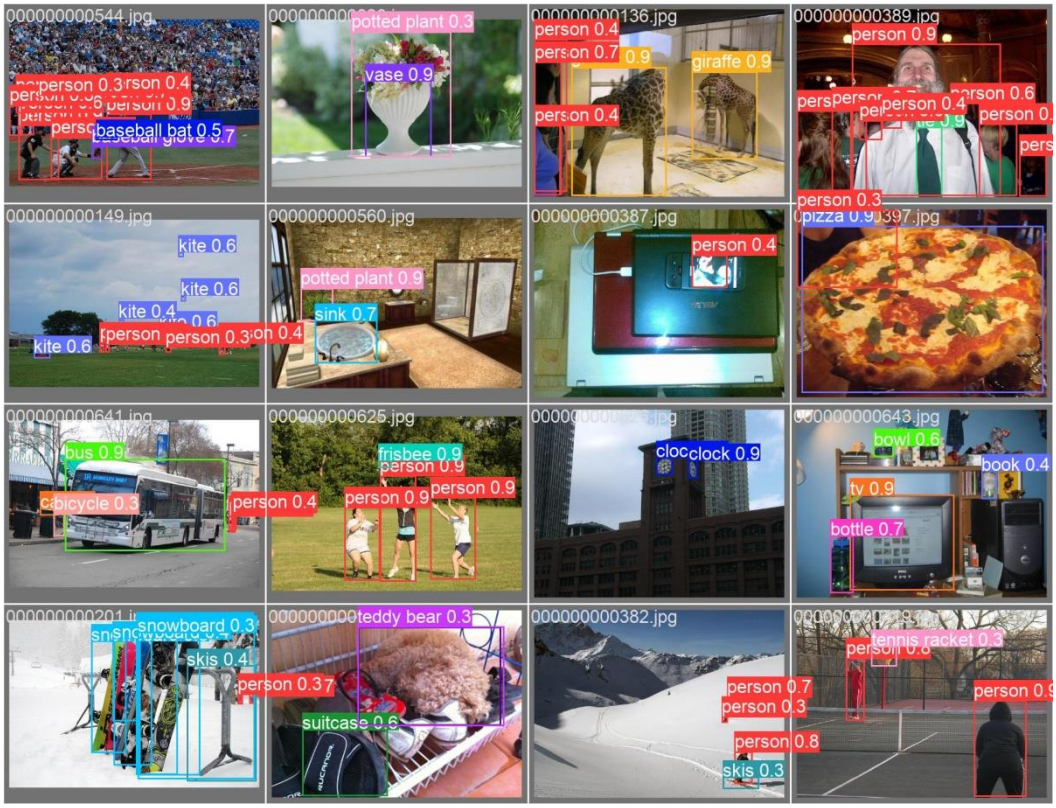


Рисунок 31 – Файл **val_batch_pred.jpg**

Рисунок 32 – Файл **val_batch_pred.jpg**

Бачимо, що тепер модель концентрується на розпізнаванні людей, що добре для нашого завдання.

Тестування

Протестуємо модель на відео з попередньої лабораторної роботи. Щоб зрозуміти, як модель впоралась з завданням ідентифікації людей у натовпі.

Лістинг коду:

```
[...]

# Тестування на відео та збереження моделі
results = model.track(source='V_6.mp4', show=True, tracker='bytetrack.yaml',
save_txt=True, save_conf=True, save_crop=True)
```

Результат:

Рисунок 33 – Розпізнавання людей у натовпі

Бачимо, що програма доволі непогано розпізнає людей з натовпу, але при цьому все ще має труднощі з місцями великого скупчення людей.

Ще з цікавих результатів саме для цього відео можна виділити те, що інколи програма розпізнає зебру пішохідного переходу як лижі.

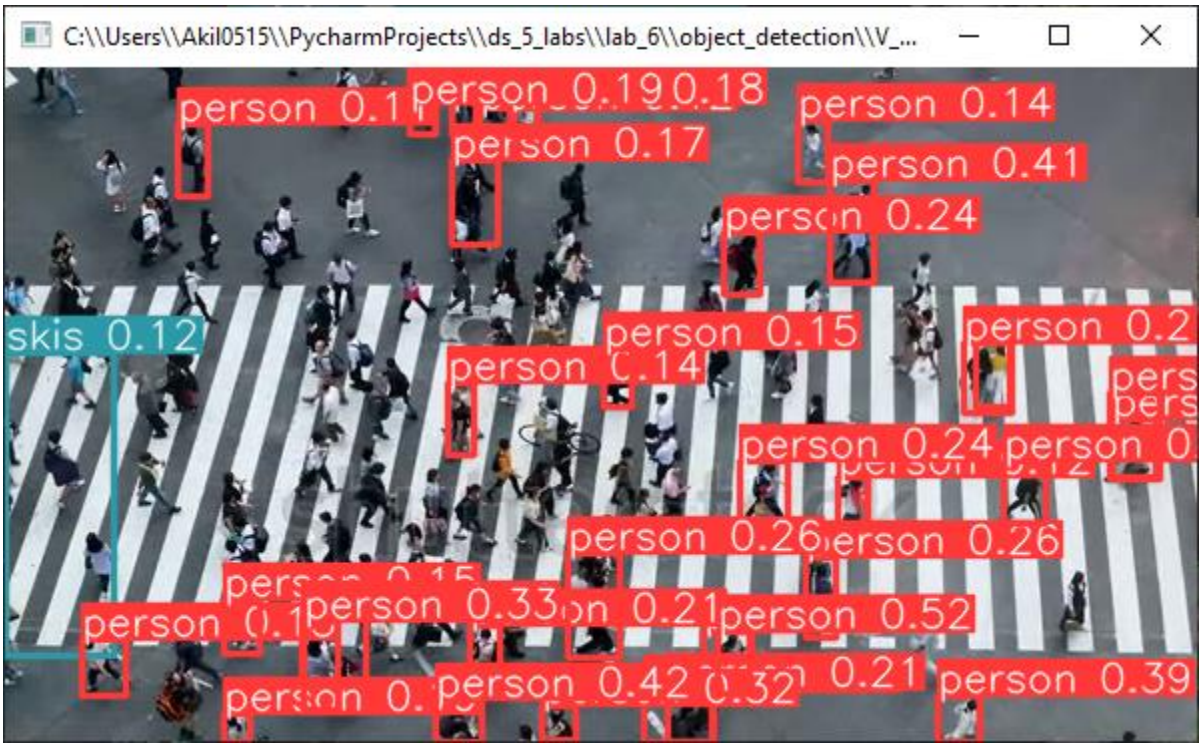


Рисунок 34 – Розпізнавання зебри як лиж

Також програма зберігає результати ідентифікації в кеш, які також можна переглянути.

2.3. Аналіз отриманих результатів

Штучна нейронна мережа:

У результаті створення 4-х штучних нейронних мереж для задачі прогнозування результату бінарної класифікації було отримано наступні результати.

Створено датасет розподілу балів за тести абітурієнтів. Датасет розбито на тренувальний та тестовий у співвідношенні 3:1. Підтвердження – створені файли з даними.

	A	B	C	D	E	F	G	H
1	ПІБ	Пільги	Бал з математики	Бал з англійської	Бал з української	Рейтинг		
2	Тарас Загвойко	False	199	193	199	197.2		
3	Андрій Петріш	False	186	200	198	193.8		
4	Андрій Товтиш	False	197	192	185	191.9		
5	Анастасія Анниш	False	197	187	189	191.6		
6	Олександр Іванниш	False	196	193	184	191.5		
7	Артем Андрїіш	False	191	195	188	191.3		
8	Михайло Тетяниш	False	197	183	189	190.4		
9	Олександр Олексїіш	False	190	193	185	189.4		
10	Сергій Марїіш	False	178	198	196	189.39999999999998		
11	Оксана Тетяниш	False	191	174	198	188.0		
12	Юлія Тарасіш	False	200	175	182	187.1		
13	Вікторія Дмитришенко	False	200	157	200	187.1		
14	Олег Ярославиш	False	200	191	163	186.20000000000002		
15	Ігор Олегіш	False	195	167	192	185.7		

	A	B	C	D	E	F	G	H	I	J	K	L
1	Рішення											
2	зараховано											
3	зараховано											
4	зараховано											
5	зараховано											
6	зараховано											
7	зараховано											
8	зараховано											
9	зараховано											
10	зараховано											
11	зараховано											
12	зараховано											
13	зараховано											
14	зараховано											
15	зараховано											

Рисунок 35 – Тренувальний датасет

Створено 4 мережі різної архітектури та навчено їх на тренувальному датасеті. Підтвердження – візуалізація процесу навчання за допомогою графіків.

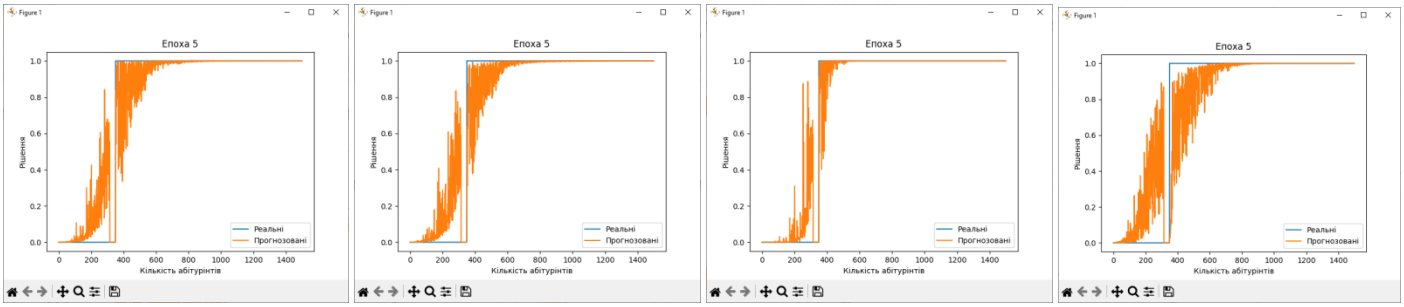


Рисунок 36 – Візуалізація навчання мереж **Sequential, Functional, CNN, RNN**

Проаналізовано реалістичність і точність отриманих результатів. Підтвердження – таблиця порівняння мереж.

Таблиця 2 – Порівняння мереж

Тип мережі:	***Sequential***	***Functional***	***CNN***	***RNN***
Помилково **не зараховані**	14	13	14	53
Помилково **зараховані**	4	10	2	6
Сумарна помилка	18	23	16	59
accuracy	0.9880	0.9847	0.9893	0.9607

З отриманих результатів можна зробити висновок, що архітектура мережі впливає на результати прогнозування. Для даної задачі та датасету найкращою виявилась мережа типу **CNN**, яка призначена для роботи з великими обсягами даних, яким і є наш датасет.

Найгіршою – мережа типу **RNN**, призначена для роботи з текстами та послідовними даними. Ця мережа запам’ятовує попередні стани, що ймовірно заважає їх правильно класифікувати абітурієнтів.

Ідентифікація та класифікація об’єктів у відеопотоці:

У результаті створення нейронної мережі, яка допомагає ідентифікувати та класифікувати об’єкти у відеопотоці отримано задовільні результати. Підтвердження – відео-результат класифікації.

Рисунок 37 – Стопкадр з відео розпізнання об’єктів у відеопотоці

Утім модель все ще не дуже добре розпізнає людей у великих натовпах. Тому можна зробити висновки, що для підвищення точності розпізнавання великого скупчення людей необхідно тренувати модель на специфічному датасеті, де наявна велика кількість зображень людей у натовпі.

Висновок:

Створено модель зміни досліджуваного процесу (бінарної класифікації) – датасет для задачі прийняття рішення щодо вступу абітурієнтів.

Розроблено на навчено 4 штучні нейронні мережі з різною архітектурою: **Sequential, Functional, CNN, RNN**.

Процес навчання мереж відображено у формі графіків.

Проведено дослідження залежності точності кожної з мереж від її структури. У результаті дослідження виявлено, що структура нейронної мережі кількісно впливає на результати прогнозування.