ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ
КАФЕДРА ЕОМ

Кваліфікаційна робота
Другий рівень (магістр)

МЕТОДИ ГЕНЕРАЦІЇ ГОЛОСОВИХ ПОВІДОМЛЕНЬ У СИСТЕМАХ КОМЕНТУВАННЯ СПОРТИВНИХ ЗМАГАНЬ

Автор: Біліченко О.Ю., ст. гр. СПм -22-5
Керівник: Барковська О.Ю., доц. каф. ЕОМ

ОГЛЯД ПРОБЛЕМНОЇ ОБЛАСТІ

- **Збір та обробка відео**
  - обробка затримок
  - передача та обробка в реальному часі
- **Відстеження та аналіз**
  - конкретність
  - розпізнавання ключових подій
- **Генерація опису**
  - натуральність
  - повнота тексту
  - мовні варіації
  - акцент

- **Генерація голосового повідомлення**
  - акустична подібність
  - звукова ідентичність
  - якість звуку
- **Виведення результатів**
  - гармонійність картинки та подій

АКТУАЛЬНІСТЬ ОБРАНОЇ ТЕМИ

Перспективи

- **Спрощення роботи коментатора**
- **Розширення кількості спортивних подій з коментуванням**
- **Покращення глядацького досвіду**
- **Індивідуальна кастомізація коментування**

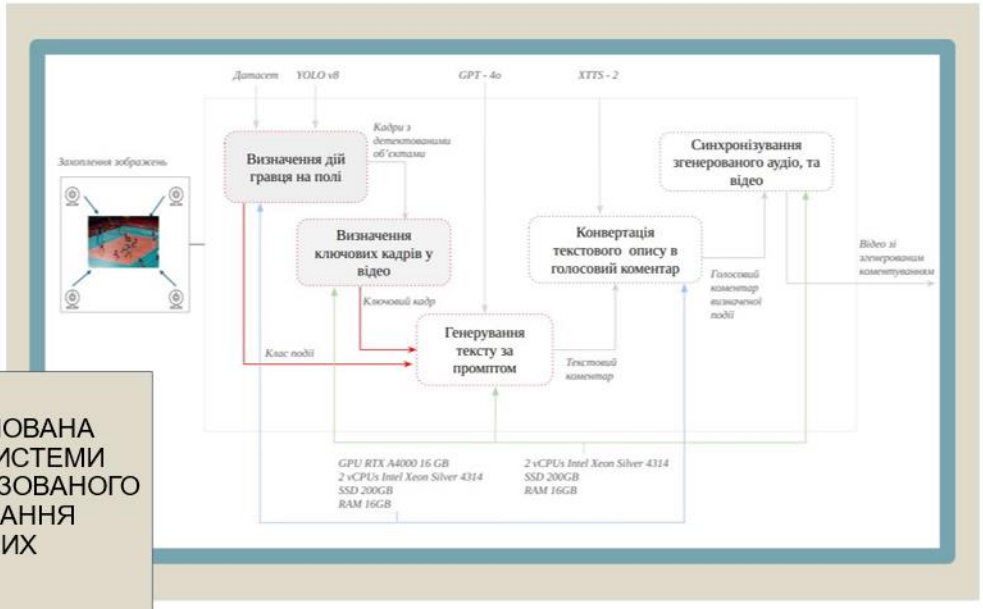
МЕТА ТА ЗАДАЧІ ДОСЛІДЖЕННЯ

Мета дослідження

Створення системи для подійного генерування голосових повідомлень, здатної імітувати коментатора спортивних трансляцій на основі вхідних відеоданих.

Задачі дослідження

- аналіз проблемної області;
- огляд існуючих технологій аналізу відео, генерування тексту та аудіо;
- створення моделі системи коментування спортивних змагань;
- розробка модулю детектування та класифікації подій;
- розробка модулю подійного генерування голосових повідомлень
- розробка модулю синхронізації згенерованого аудіо та відео
- проведення дослідження впливу оптимізатора на точність детектування
- проведення дослідження впливу мультимодальності на повноту і точність коментаря
- проведення дослідження впливу кількості ключових кадрів у батчі на синхронізацію та затримки в результуючому відео
- аналіз отриманих результатів



ЗАПРОПОНОВАНА МОДЕЛЬ СИСТЕМИ АВТОМАТИЗОВАНОГО КОМЕНТУВАННЯ СПОРТИВНИХ ЗМАГАНЬ

5

ОГЛЯД ОСНОВНИХ МОДУЛІВ.

Модуль детектування та класифікації подій (в контексті гравців) на майданчику

Задача модулю:

Попереднє визначення дій спортсменів під час матчу, для подальшої передачі анотованого зображення

Software stack

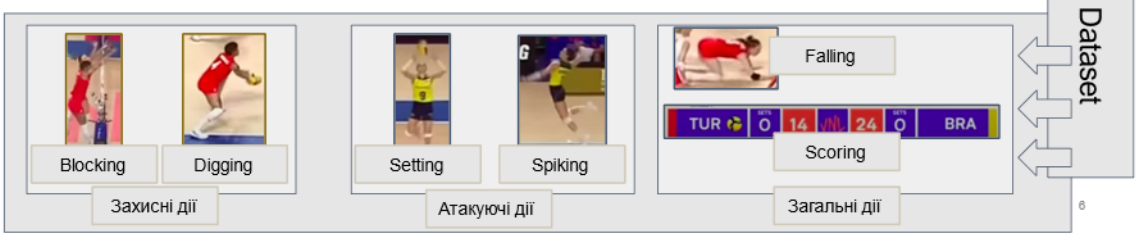
YOLOv8

Hardware stack (execution)

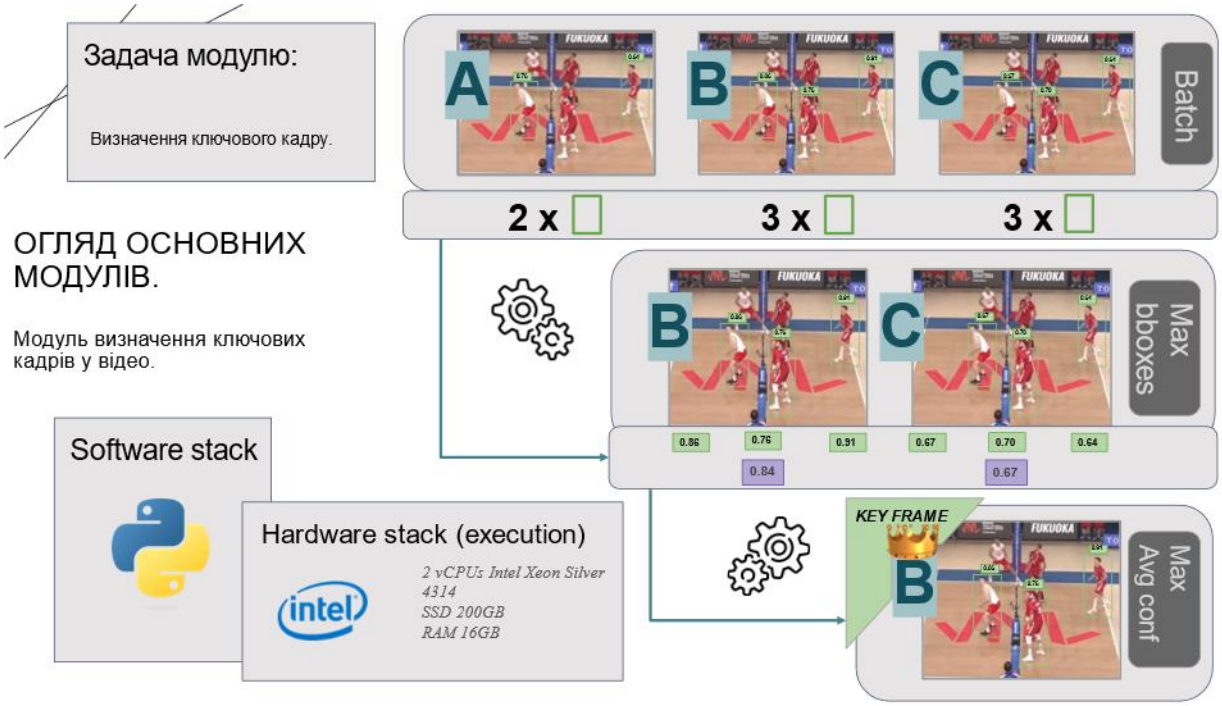
GPU RTX A4000 16 GB
2 vCPUs Intel Xeon Silver 4314
SSD 200GB
RAM 16GB

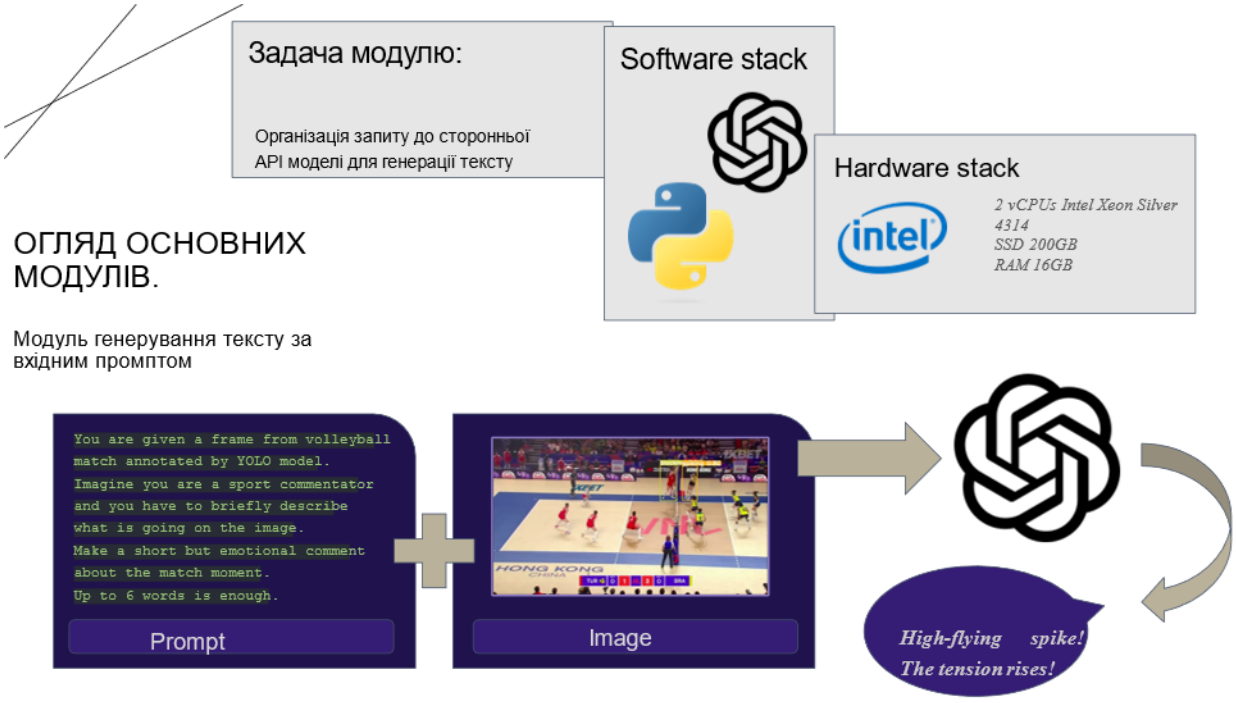
Hardware stack (training)

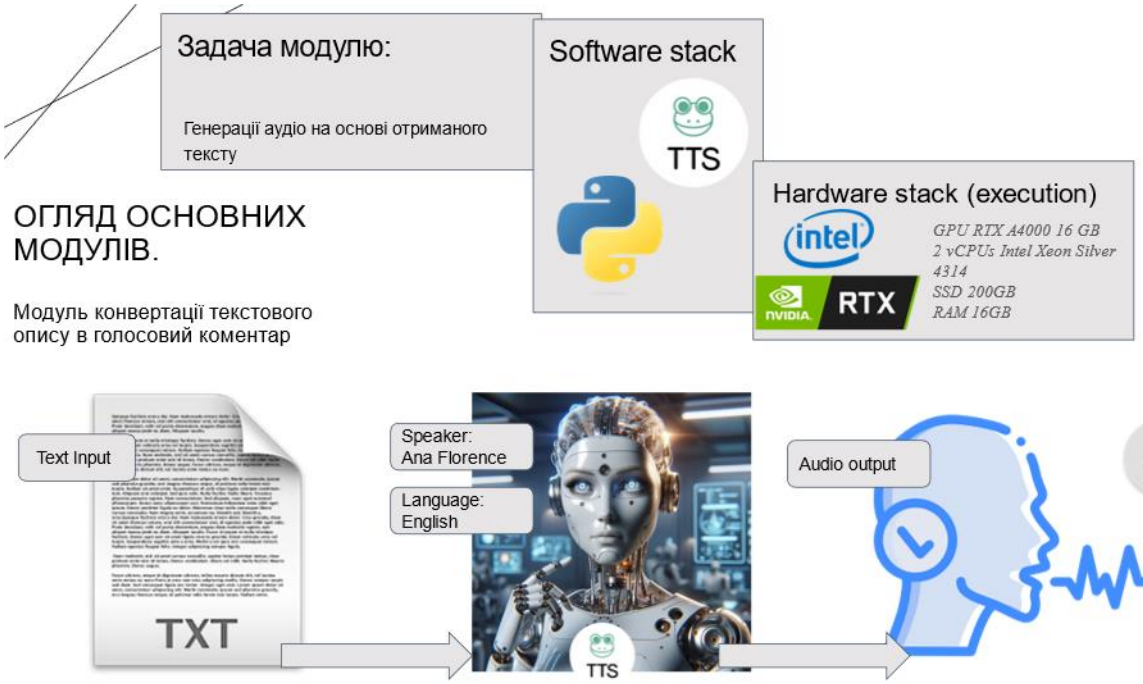
GPU Tesla T4 16 GB
8 CPUs Intel Xeon 2GHz
HDD 250GB
RAM 16GB

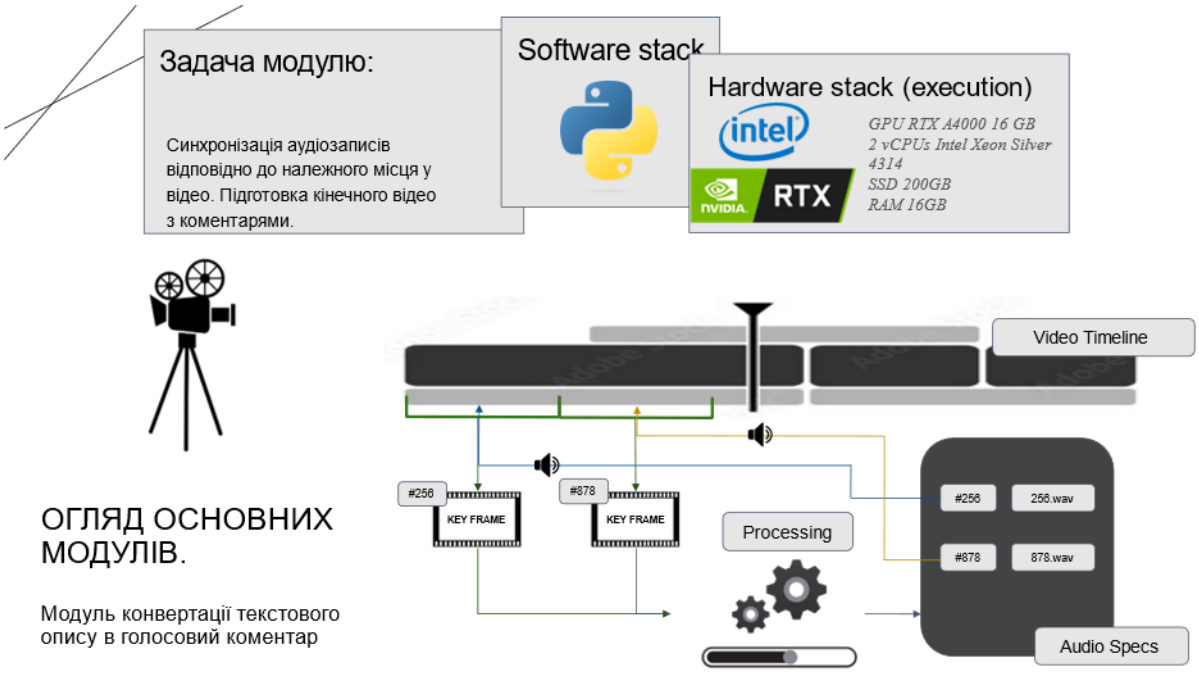


6


Задача модулю:
Визначення ключового кадру.
ОГЛЯД ОСНОВНИХ МОДУЛІВ.
Модуль визначення ключових кадрів у відео.
A
B
C
Batch
2 x
3 x
3 x
Max bboxes
0.86
0.76
0.91
0.67
0.70
0.64
0.84
0.67
KEY FRAME
Max Avg conf
Software stack
Hardware stack (execution)
2 vCPUs Intel Xeon Silver 4314
SSD 200GB
RAM 16GB

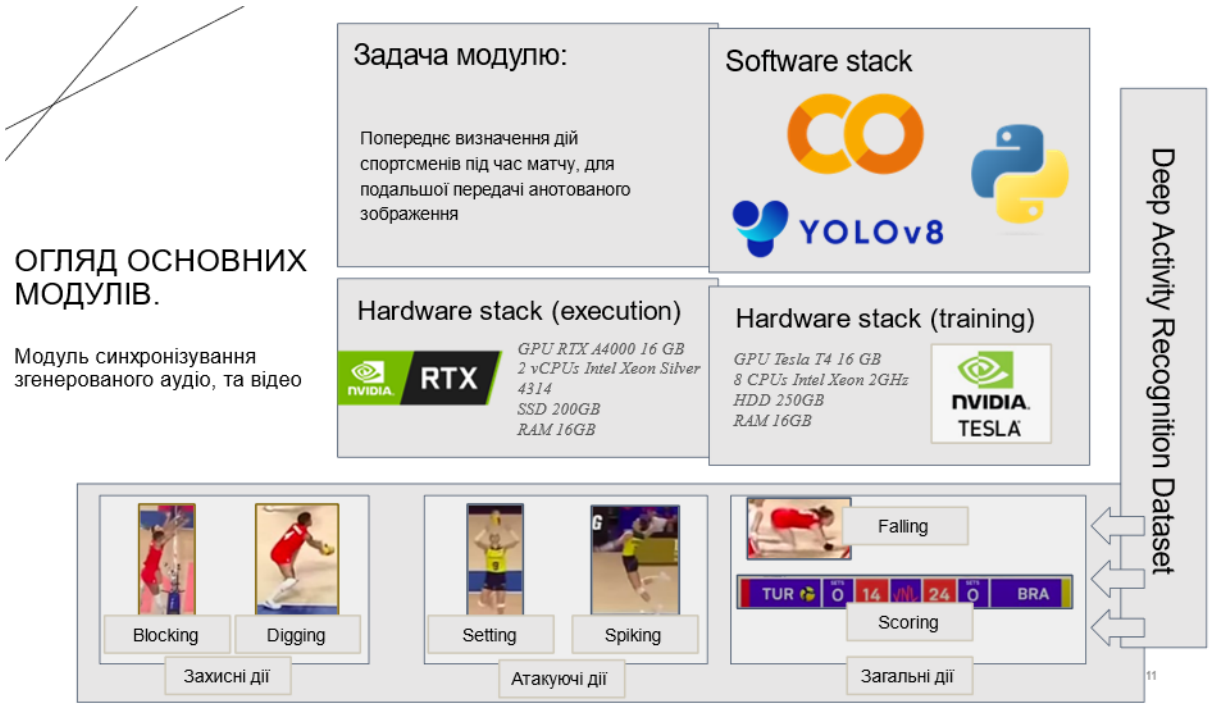

Задача модулю:
Організація запиту до сторонньої API моделі для генерації тексту
ОГЛЯД ОСНОВНИХ МОДУЛІВ.
Модуль генерування тексту за вхідним промптом
Software stack
Hardware stack
2 vCPUs Intel Xeon Silver 4314
SSD 200GB
RAM 16GB
You are given a frame from volleyball match annotated by YOLO model.
Imagine you are a sport commentator and you have to briefly describe what is going on the image.
Make a short but emotional comment about the match moment.
Up to 6 words is enough.
Prompt
Image
High-flying spike!
The tension rises!


Задача модулю:
Генерації аудіо на основі отриманого тексту
Software stack
TTS
Hardware stack (execution)
intel
NVIDIA
RTX
GPU RTX A4000 16 GB
2 vCPUs Intel Xeon Silver 4314
SSD 200GB
RAM 16GB
ОГЛЯД ОСНОВНИХ МОДУЛІВ.
Модуль конвертації текстового опису в голосовий коментар
Text Input
TXT
Speaker: Ana Florence
Language: English
TTS
Audio output


Задача модулю:
Синхронізація аудіозаписів відповідно до належного місця у відео. Підготовка кінечного відео з коментарями.
Software stack
Hardware stack (execution)
intel
NVIDIA
RTX
GPU RTX A4000 16 GB
2 vCPUs Intel Xeon Silver 4314
SSD 200GB
RAM 16GB
ОГЛЯД ОСНОВНИХ МОДУЛІВ.
Модуль конвертації текстового опису в голосовий коментар
Video Timeline
#256
KEY FRAME
#878
KEY FRAME
Processing
#256
256.wav
#878
878.wav
Audio Specs

ОГЛЯД ОСНОВНИХ МОДУЛІВ.

Модуль синхронізування згенерованого аудіо, та відео

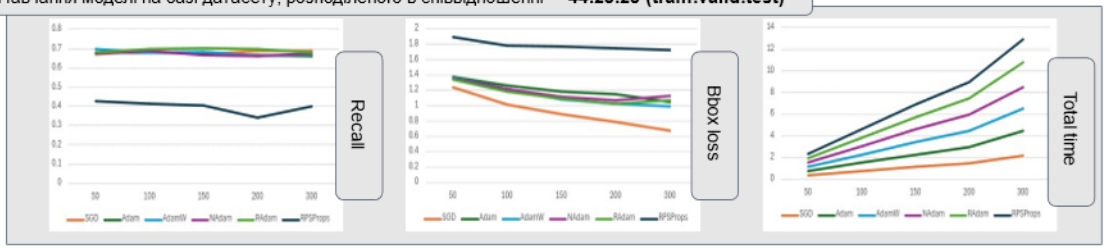


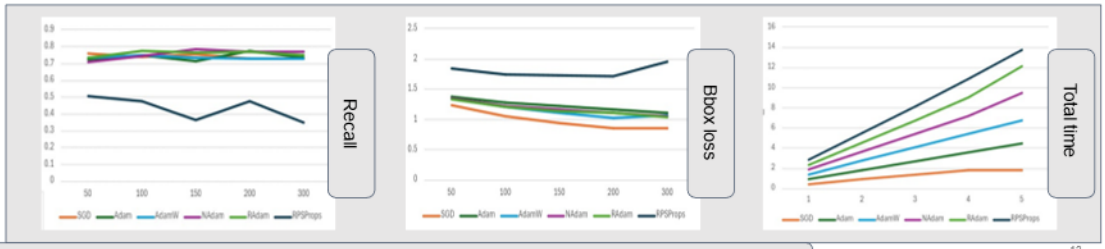
ОТРИМАНІ РЕЗУЛЬТАТИ

ОТРИМАНІ РЕЗУЛЬТАТИ

Дослідження характеристик та можливостей нейромережевого детектора та класифікатора подій у відеопослідовності

Навчання моделі на базі датасету, розподіленого в співвідношенні **44:28:28 (train:valid:test)**



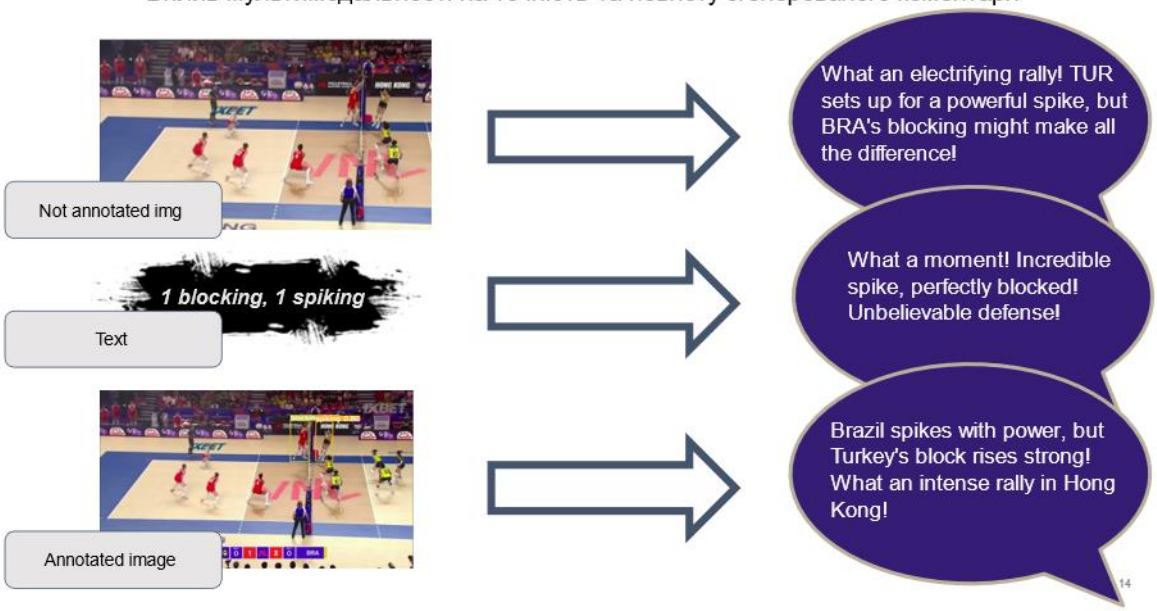


Навчання моделі на базі датасету, розподіленого в співвідношенні **70:15:15 (train:valid:test)**

13

ОТРИМАНІ РЕЗУЛЬТАТИ

Вплив мультимодальності на точність та повноту згенерованого коментаря



14

ОТРИМАНІ РЕЗУЛЬТАТИ

Вплив вибору кількості та частоти ключових кадрів на синхронізацію згенерованого коментаря та події в часі

Розмір батчу: 3
Помилки відповідності: 9%
Затримки коментування: 4%
Кількість коментарів: 22

Розмір батчу: 192
Помилки відповідності: 10%
Затримки коментування: 0%
Кількість коментарів: 10

15

ВИСНОВКИ

Побудована система пропонує вирішення наступних задач:

- розпізнавання гравців на волейбольному полі
- вибір ключового кадру з послідовності
- генерування тексту коментаря, на основі отриманих даних
- створення аудіо зі згенерованого тексту
- синхронізація аудіо з оригінальним відео

Потенційні покращення:

- збільшення рівню точності розпізнавання та класифікації
- розширення кількості моделей для комплексної обробки даних
- додавання класифікатора жестів рефері
- підключення аналізу аудіо
- розробка користувацького інтерфейсу
- інтеграція індивідуалізації коментування

16

АПРОБАЦІЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Стаття у фаховому науковому виданні

Olesia BARKOVSKA, **Oleksandr BILICHENKO**, Heorhii UVAROV, Tymur MAKUSHENKO Improved rendering method of skeletal animation on control points base. *Computer systems and information technologies*, 2024, (1), 71-81.

https://doi.org/10.31891/csit-2024-1-9

ДОДАТОК Б

Лістинги розробленого застосунку

Б.1 Функція вибору ключового кадру у батчі

```
        -> List[Tuple[int, ultralytics.engine.results.Results]]:
    def average_confidence(bounding_boxes):
        if not bounding_boxes:
            return 0
        return np.mean([bbox.conf for bbox in bounding_boxes])

    def select_best_frame(batch_):
        max_bboxes = 0
        best_frame_ = None
        frame_n = None

        for n, frame in batch_:
            bboxes = frame.boxes
            num_bboxes = len(bboxes)
            avg_conf = average_confidence(bboxes)

            # select the frame with the largest number of bboxes
and the highest average conf
            if num_bboxes > max_bboxes or (num_bboxes ==
max_bboxes and avg_conf > average_confidence(
                    best_frame_.boxes if best_frame_ else [])):
                best_frame_ = frame
                max_bboxes = num_bboxes
                frame_n = n
        return frame_n, best_frame_

    batch = []
    results = []

    for i, annotation in yolo_annotations_gen:
        batch.append((i, annotation))
        if len(batch) == batch_size:
            # Process the batch
            best_frame = select_best_frame(batch)
            results.append(best_frame)
            batch = []

    # Process remaining frames if batch is not full
    if batch:
        best_frame = select_best_frame(batch)
        results.append(best_frame)
```

```
    return [result for result in results if result[0] is not
None]
```

Б.2 Функція синхронізації згенерованого аудіо з оригінальним відео

```
def align_video_with_audios(video_path: str, audio_specs:
List[Tuple[str, int]], fps: float, output_path: str) -> None:
    # Get video duration
    probe = ffmpeg.probe(video_path)
    video_duration = float(probe['format']['duration'])

    video_input = ffmpeg.input(video_path).video

    inputs = [video_input]

    # Track the end time of the last audio
    last_audio_end_time = 0

    for audio_path, start_frame in audio_specs:
        # Calculate start time for the audio
        start_time = start_frame / fps
        # Get audio duration
        audio_probe = ffmpeg.probe(audio_path)
        audio_duration =
float(audio_probe['format']['duration'])

        # Calculate audio end time
        audio_end_time = start_time + audio_duration

        # If the start time of the current audio is before the
end time of the last audio, skip this audio
        if start_time < last_audio_end_time:
            continue

        audio_input =
ffmpeg.input(audio_path).audio.filter('adelay', int(start_time *
1000))
        inputs.append(audio_input)

        # Update the end time of the last audio
        last_audio_end_time = audio_end_time

    if last_audio_end_time > video_duration:
        inputs = inputs[:-1]

    merged_audio = ffmpeg.filter(inputs[1:], 'amix',
inputs=len(inputs[1:]))
    output = ffmpeg.output(inputs[0], merged_audio, output_path,
vcodec='copy', acodec='aac')
    output.run(overwrite_output=True)
```