```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.
import kagglehub
xubiker_tennistrackingassignment_path = kagglehub.dataset_download('xubiker/tennistrackingassignment')

print('Data source import complete.')
```

Практическое задание №2

Общая терминология по используемым данным

Предоставляемые данные для разработки моделей и алгоритмов трекинга мяча в теннисе представляют собор набор игр (game), состоящих из нескольких клипов (clip), каждый из которых состоит из набора кадров (frame). Обратите внимание на структуру организации файлов внутри предоставляемого датасета для полного понимания.

Большинство алгоритмов трекинга объектов работают с несколькими последовательными кадрами, и в данном задании также подразумевается использование этого приема. Последовательность нескольких кадров будем именовать стопкой (stack), размер стопки (stack_s) является гиперпараметром разрабатываемого алгоритма.

Заготовка решения

Загрузка датасета

Для работы с данными в ноутбуке kaggle необходимо подключить датасет. File -> Add or upload data, далее в поиске написать tennis-tracking-assignment и выбрать датасет. Если поиск не работает, то можно добавить датасет по url: https://www.kaggle.com/xubiker/tennistrackingassignment. После загрузки данные датасета будут примонтированы в ../input/tennistrackingassignment.

Установка и импорт зависимостей

Установка необходимых пакетов (не забудьте "включить интернет" в настройках ноутбука kaggle):

```
!pip install moviepy --upgrade
!pip install gdown
```

```
Requirement already satisfied: moviepy in /usr/local/lib/python3.10/dist-packages (2.1.1)
Requirement already satisfied: decorator<6.0,>=4.0.2 in /usr/local/lib/python3.10/dist-packages (from moviepy) (4.4.2)
Requirement already satisfied: imageio<3.0,>=2.5 in /usr/local/lib/python3.10/dist-packages (from moviepy) (2.35.1)
Requirement already satisfied: imageio_ffmpeg>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from moviepy) (0.5.1)
Requirement already satisfied: numpy>=1.25.0 in /usr/local/lib/python3.10/dist-packages (from moviepy) (1.26.4)
Requirement already satisfied: proglog<=1.0.0 in /usr/local/lib/python3.10/dist-packages (from moviepy) (0.1.10)
Requirement already satisfied: python-dotenv>=0.10 in /usr/local/lib/python3.10/dist-packages (from moviepy) (1.0.1)
Requirement already satisfied: pillow<11.0,>=9.2.0 in /usr/local/lib/python3.10/dist-packages (from moviepy) (10.4.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from imageio_ffmpeg>=0.2.0->moviepy) (71.0.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from proglog<=1.0.0->moviepy) (4.66.5)
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (5.2.0)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.12.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from gdown) (3.16.1)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages (from gdown) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.5)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->gdown) (2.6)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2024.8.
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (1.7
```

После установки пакетов для корректной работы надо обязательно перезагрузить ядро. Run -> Restart and clear cell outputs. Без сего действа будет ошибка при попытке обращения к библиотеке moviepy при сохранении визуализации в виде видео. Может когда-то авторы библиотеки это починят...

Импорт необходимых зависимостей:

```
from pathlib import Path
from typing import List, Tuple, Sequence

import numpy as np
from numpy import unravel_index
from PIL import Image, ImageDraw, ImageFont
from tqdm import tqdm, notebook

from moviepy.video.io.ImageSequenceClip import ImageSequenceClip

import math
from scipy.ndimage import gaussian_filter

import gc
import time
import random
import csv
```

Набор функций для загрузки данных из датасета

Функция load_clip_data загружает выбранный клип из выбранной игры и возвращает его в виде numpy массива [n_frames, height, width, 3] типа uint8. Для ускорения загрузки используется кэширование - однажды загруженные клипы хранятся на диске в виде npz архивов, при последующем обращении к таким клипам происходит загрузка npz архива.

Также добавлена возможность чтения клипа в половинном разрешении 640x360, вместо оригинального 1280x720 для упрощения и ускорения разрабатываемых алгоритмов.

Функция load_clip_labels загружает референсные координаты мяча в клипе в виде numpy массива [n_frames, 4], где в каждой строке массива содержатся значения [code, x, y, q]. x, y соответствуют координате центра мяча на кадре, q не используется в данном задании, code описывает статус мяча:

- code = 0 - мяча в кадре нет
- code = 1 - мяч присутствует в кадре и легко идентифицируем
- code = 2 - мяч присутствует в кадре, но сложно идентифицируем
- code = 3 - мяч присутствует в кадре, но заслонен другими объектами.

При загрузке в половинном разрешении координаты x, y делятся на 2.

Функция load_clip загружает выбранный клип и соответствующий массив координат и возвращает их в виде пары.

```
def get_num_clips(path: Path, game: int) -> int:
    return len(list((path / f'game{game}/').iterdir()))


def get_game_clip_pairs(path: Path, games: List[int]) -> List[Tuple[int, int]]:
    return [(game, c)  for game in games for c in range(1, get_num_clips(path, game) + 1)]


def load_clip_data(path: Path, game: int, clip: int, downscale: bool, quiet=False) -> np.ndarray:
    if not quiet:
        suffix = 'downscaled' if downscale else ''
        print(f'loading clip data (game {game}, clip {clip}) {suffix}')
    cache_path = path / 'cache'
    cache_path.mkdir(exist_ok=True)
    resize_code = '_ds2' if downscale else ''
    cached_data_name = f'{game}_{clip}{resize_code}.npz'
    if (cache_path / cached_data_name).exists():
        clip_data = np.load(cache_path / cached_data_name)['clip_data']
    else:
        clip_path = path / f'game{game}/clip{clip}'
        n_imgs = len(list(clip_path.iterdir())) - 1
        imgs = [None] * n_imgs
        for i in notebook.tqdm(range(n_imgs)):
            img = Image.open(clip_path / f'{i:04d}.jpg')
            if downscale:
                img = img.resize((img.width // 2, img.height // 2),)
            imgs[i] = np.array(img, dtype=np.uint8)
        clip_data = np.stack(imgs)
        cache_path.mkdir(exist_ok=True, parents=True)
        np.savez_compressed(cache_path / cached_data_name, clip_data=clip_data)
    return clip_data


def load_clip_labels(path: Path, game: int, clip: int, downscale: bool, quiet=False):
    if not quiet:
```

```
        print(f'loading clip labels (game {game}, clip {clip})')
    clip_path = path / f'game{game}/clip{clip}'
    labels = []
    with open(clip_path / 'labels.csv') as csvfile:
        lines = list(csv.reader(csvfile))
        for line in lines[1:]:
            values = np.array([-1 if i == '' else int(i) for i in line[1:]])
            if downscale:
                values[1] //= 2
                values[2] //= 2
            labels.append(values)
    return np.stack(labels)


def load_clip(path: Path, game: int, clip: int, downscale: bool, quiet=False):
    data = load_clip_data(path, game, clip, downscale, quiet)
    labels = load_clip_labels(path, game, clip, downscale, quiet)
    return data, labels
```

Набор дополнительных функций

Еще несколько функций, немного облегчающих выполнение задания:

- prepare_expariment создает новую директорию в out_path для хранения результатов текущего эксперимента. Нумерация выполняется автоматически, функция возвращает путь к созданной директории эксперимента;
- ball_gauss_template - создает "шаблон" мяча, может быть использована в алгоритмах поиска мяча на изображении по корреляции;
- create_masks - принимает набор кадров и набор координат мяча, и генерирует набор масок, в которых помещает шаблон мяча на заданные координаты. Может быть использована при обучении нейронной сети семантической сегментации;

```
def prepare_experiment(out_path: Path) -> Path:
    out_path.mkdir(parents=True, exist_ok=True)
    dirs = [d for d in out_path.iterdir() if d.is_dir() and d.name.startswith('exp_')]
    experiment_id = max(int(d.name.split('_')[1]) for d in dirs) + 1 if dirs else 1
    exp_path = out_path / f'exp_{experiment_id}'
    exp_path.mkdir()
    return exp_path


def ball_gauss_template(rad, sigma):
    x, y = np.meshgrid(np.linspace(-rad, rad, 2 * rad + 1), np.linspace(-rad, rad, 2 * rad + 1))
    dst = np.sqrt(x * x + y * y)
    gauss = np.exp(-(dst ** 2 / (2.0 * sigma ** 2)))
    return gauss


def create_masks(data: np.ndarray, labels: np.ndarray, resize):
    rad = 64 #25
    sigma = 10
    if resize:
        rad //= 2
    ball = ball_gauss_template(rad, sigma)
    n_frames = data.shape[0]
    sh = rad
    masks = []
    for i in range(n_frames):
        label = labels[i, ...]
        frame = data[i, ...]
        if 0 < label[0] < 3:
            x, y = label[1:3]
            mask = np.zeros((frame.shape[0] + 2 * rad + 2 * sh, frame.shape[1] + 2 * rad + 2 * sh), np.float32)
            mask[y + sh : y + sh + 2 * rad + 1, x + sh : x + sh + 2 * rad + 1] = ball
            mask = mask[rad + sh : -rad - sh, rad + sh : -rad - sh]
            masks.append(mask)
        else:
            masks.append(np.zeros((frame.shape[0], frame.shape[1]), dtype=np.float32))
    return np.stack(masks)
```

Набор функций, предназначенных для визуализации результатов

Функция visualize_prediction принимает набор кадров, набор координат детекции мяча (можно подавать как референсные значения, так и предсказанные) и создает видеоклип, в котором отрисовывается положение мяча, его трек, номер кадра и метрика качества

трекинга (если она была передана в функцию). Видеоклип сохраняется в виде mp4 файла. Кроме того данная функция создает текстовый файл, в который записывает координаты детекции мяча и значения метрики качества трекинга.

Функция visualize_prob принимает набор кадров и набор предсказанных карт вероятности и создает клип с наложением предсказанных карт вероятности на исходные карты. Области "подсвечиваются" желтым, клип сохраняется в виде mp4 видеофайла. Данная функция может быть полезна при наличии в алгоритме трекинга сети, осуществляющей семантическую сегментацию.

```
def _add_frame_number(frame: np.ndarray, number: int) -> np.ndarray:
    fnt = ImageFont.load_default() # ImageFont.truetype("arial.ttf", 25)
    img = Image.fromarray(frame)
    draw = ImageDraw.Draw(img)
    draw.text((10, 10), f'frame {number}', font=fnt, fill=(255, 0, 255))
    return np.array(img)


def _vis_clip(data: np.ndarray, lbls: np.ndarray, metrics: List[float] = None, ball_rad=5, color=(255, 0, 0), track_length=10):
    print('perfoming clip visualization')
    n_frames = data.shape[0]
    frames_res = []
    fnt = ImageFont.load_default() # ImageFont.truetype("arial.ttf", 25)
    for i in range(n_frames):
        img = Image.fromarray(data[i, ...])
        draw = ImageDraw.Draw(img)
        txt = f'frame {i}'
        if metrics is not None:
            txt += f', SiBaTrAcc: {metrics[i]:.3f}'
        draw.text((10, 10), txt, font=fnt, fill=(255, 0, 255))
        label = lbls[i]
        if label[0] != 0: # the ball is clearly visible
            px, py = label[1], label[2]
            draw.ellipse((px - ball_rad, py - ball_rad, px + ball_rad, py + ball_rad), outline=color, width=2)
            for q in range(track_length):
                if lbls[i-q-1][0] == 0:
                    break
                if i - q > 0:
                    draw.line((lbls[i - q - 1][1], lbls[i - q - 1][2], lbls[i - q][1], lbls[i - q][2]), fill=color)
        frames_res.append(np.array(img))
    return frames_res


def _save_clip(frames: Sequence[np.ndarray], path: Path, fps):
    assert path.suffix in ('.mp4', '.gif')
    clip = ImageSequenceClip(frames, fps=fps)
    if path.suffix == '.mp4':
        clip.write_videofile(str(path), fps=fps, logger=None)
    else:
        clip.write_gif(str(path), fps=fps, logger=None)


def _to_yellow_heatmap(frame: np.ndarray, pred_frame: np.ndarray, alpha=0.4):
    img = Image.fromarray((frame * alpha).astype(np.uint8))
    maskR = (pred_frame * (1 - alpha) * 255).astype(np.uint8)
    maskG = (pred_frame * (1 - alpha) * 255).astype(np.uint8)
    maskB = np.zeros_like(maskG, dtype=np.uint8)
    mask = np.stack([maskR, maskG, maskB], axis=-1)
    return img + mask


def _vis_pred_heatmap(data_full: np.ndarray, pred_prob: np.ndarray, display_frame_number):
    n_frames = data_full.shape[0]
    v_frames = []
    for i in range(n_frames):
        frame = data_full[i, ...]
        pred = pred_prob[i, ...]
        hm = _to_yellow_heatmap(frame, pred)
        if display_frame_number:
            hm = _add_frame_number(hm, i)
        v_frames.append(hm)
    return v_frames


def visualize_prediction(data_full: np.ndarray, labels_pr: np.ndarray, save_path: Path, name: str, metrics=None, fps=15):
    with open(save_path / f'{name}.txt', mode='w') as f:
        if metrics is not None:
            f.write(f'SiBaTrAcc: {metrics[-1]} \n')
        for i in range(labels_pr.shape[0]):
            f.write(f'frame {i}: {labels_pr[i, 0]}, {labels_pr[i, 1]}, {labels_pr[i, 2]} \n')

    v = _vis_clip(data_full, labels_pr, metrics)
    _save_clip(v, save_path / f'{name}.mp4', fps=fps)
```

```
def visualize_prob(data: np.ndarray, pred_prob: np.ndarray, save_path: Path, name: str, frame_number=True, fps=15):
    v_pred = _vis_pred_heatmap(data, pred_prob, frame_number)
    _save_clip(v_pred, save_path / f'{name}_prob.mp4', fps=fps)
```

Класс DataGenerator

Класс, отвечающий за генерацию данных для обучения модели. Принимает на вход путь к директории с играми, индексы игр, используемые для генерации данных, и размер стопки. Хранит в себе автоматически обновляемый пул с клипами игр.

В пуле содержится pool_s клипов. DataGenerator позволяет генерировать батч из стопок (размера stack_s) последовательных кадров. Выбор клипа для извлечения данных взвешенно-случайный: чем больше длина клипа по сравнению с другими клипами в пуле, тем вероятнее, что именно из него будет сгенерирована стопка кадров. Выбор стопки кадров внтури выбранного клипа полностью случаен. Кадры внутри стопки конкатенируются по последнему измерению (каналам).

После генерирования количества кадров равного общему количеству кадров, хранимых в пуле, происходит автоматическое обновление пула: из пула извлекаются pool_update_s случайных клипов, после чего в пул загружается pool_update_s случайных клипов, не присутствующих в пуле. В случае, если размер пула pool_s больше или равен суммарному количеству клипов в играх, переданных в конструктор, все клипы сразу загружаются в пул, и автообновление не производится.

Использование подобного пула позволяет работать с практически произвольным количеством клипов, без необходимости загружать их всех в оперативную память.

Для вашего удобства функция извлечения стопки кадров из пула помимо самой стопки также создает и возвращает набор сгенерированных масок с мячом исходя из референсных координат мяча в клипе.

Функция random_g принимает гиперпараметр размера стопки кадров и предоставляет генератор, возвращающий стопки кадров и соответствующие им маски. Данный генератор может быть использован при реализации решения на tensorflow. Обновление пула происходит автоматически, об этом беспокоиться не нужно.

```
class DataGenerator:

    def __init__(self, path: Path, games: List[int], stack_s, downscale, pool_s=30, pool_update_s=10, pool_autoupdate=True, quiet=False)
        self.path = path
        self.stack_s = stack_s
        self.downscale = downscale
        self.pool_size = pool_s
        self.pool_update_size = pool_update_s
        self.pool_autoupdate = pool_autoupdate
        self.quiet = quiet
        self.data = []
        self.masks = []

        self.frames_in_pool = 0
        self.produced_frames = 0
        self.game_clip_pairs = get_game_clip_pairs(path, list(set(games)))
        self.game_clip_pairs_loaded = []
        self.game_clip_pairs_not_loaded = list.copy(self.game_clip_pairs)
        self.pool = {}

        self._first_load()

    def _first_load(self):
        # --- if all clips can be placed into pool at once, there is no need to refresh pool at all ---
        if len(self.game_clip_pairs) <= self.pool_size:
            for gcp in self.game_clip_pairs:
                self._load(gcp)
            self.game_clip_pairs_loaded = list.copy(self.game_clip_pairs)
            self.game_clip_pairs_not_loaded.clear()
            self.pool_autoupdate = False
        else:
            self._load_to_pool(self.pool_size)
        self._update_clip_weights()

    def _load(self, game_clip_pair):
        game, clip = game_clip_pair
        data, labels = load_clip(self.path, game, clip, self.downscale, quiet=self.quiet)
        masks = create_masks(data, labels, self.downscale)
        weight = data.shape[0] if data.shape[0] >= self.stack_s else 0
        self.pool[game_clip_pair] = (data, labels, masks, weight)
        self.frames_in_pool += data.shape[0] - self.stack_s + 1
        # print(f'items in pool: {len(self.pool)} - {self.pool.keys()}')

    def _remove(self, game_clip_pair):
        value = self.pool.pop(game_clip_pair)
```

```
        self.frames_in_pool -= value[0].shape[0] - self.stack_s + 1
        del value
        # print(f'items in pool: {len(self.pool)} - {self.pool.keys()}')

    def _update_clip_weights(self):
        weights = [self.pool[pair][-1] for pair in self.game_clip_pairs_loaded]
        tw = sum(weights)
        self.clip_weights = [w / tw for w in weights]
        # print(f'clip weights: {self.clip_weights}')

    def _remove_from_pool(self, n):
        # --- remove n random clips from pool ---
        if len(self.game_clip_pairs_loaded) >= n:
            remove_pairs = random.sample(self.game_clip_pairs_loaded, n)
            for pair in remove_pairs:
                self._remove(pair)
                self.game_clip_pairs_loaded.remove(pair)
                self.game_clip_pairs_not_loaded.append(pair)
            gc.collect()

    def _load_to_pool(self, n):
        # --- add n random clips to pool ---
        gc.collect()
        add_pairs = random.sample(self.game_clip_pairs_not_loaded, n)
        for pair in add_pairs:
            self._load(pair)
            self.game_clip_pairs_not_loaded.remove(pair)
            self.game_clip_pairs_loaded.append(pair)

    def update_pool(self):
        self._remove_from_pool(self.pool_update_size)
        self._load_to_pool(self.pool_update_size)
        self._update_clip_weights()

    def get_random_stack(self):
        pair_idx = np.random.choice(len(self.game_clip_pairs_loaded), 1, p=self.clip_weights)[0]
        game_clip_pair = self.game_clip_pairs_loaded[pair_idx]
        d, _, m, _ = self.pool[game_clip_pair]
        start = np.random.choice(d.shape[0] - self.stack_s, 1)[0]
        frames_stack = d[start : start + self.stack_s, ...]
        frames_stack = np.squeeze(np.split(frames_stack, indices_or_sections=self.stack_s, axis=0))
        frames_stack = np.concatenate(frames_stack, axis=-1)
        mask = m[start + self.stack_s - 1, ...]
        return frames_stack, mask

    def get_random_batch(self, batch_s):
        imgs, masks = [], []
        while len(imgs) < batch_s:
            frames_stack, mask = self.get_random_stack()
            imgs.append(frames_stack)
            masks.append(mask)
        if self.pool_autoupdate:
            self.produced_frames += batch_s
            # print(f'produced frames: {self.produced_frames} from {self.frames_in_pool}')
            if self.produced_frames >= self.frames_in_pool:
                self.update_pool()
                self.produced_frames = 0
        return np.stack(imgs), np.stack(masks)

    def random_g(self, batch_s):
        while True:
            imgs_batch, masks_batch = self.get_random_batch(batch_s)
            yield imgs_batch, masks_batch
```

Пример использования DataGenerator

Рекомендованный размер пула pool_s=10 в случае использования уменьшенных вдвое изображений. При большем размере пула есть большая вероятность нехватки имеющихся 13G оперативной памяти. Используйте параметр quiet=True в конструкторе DataGenerator, если хотите скрыть все сообщения о чтении данных и обновлении пула.

```
stack_s = 3
batch_s = 4
train_gen = DataGenerator(Path('../input/train/'), [1, 2, 3, 4], stack_s=stack_s, downscale=True, pool_s=10, pool_update_s=4, quiet=Fals
for i in range(10):
    imgs, masks = train_gen.get_random_batch(batch_s)
    print(imgs.shape, imgs.dtype, masks.shape, masks.dtype)
```

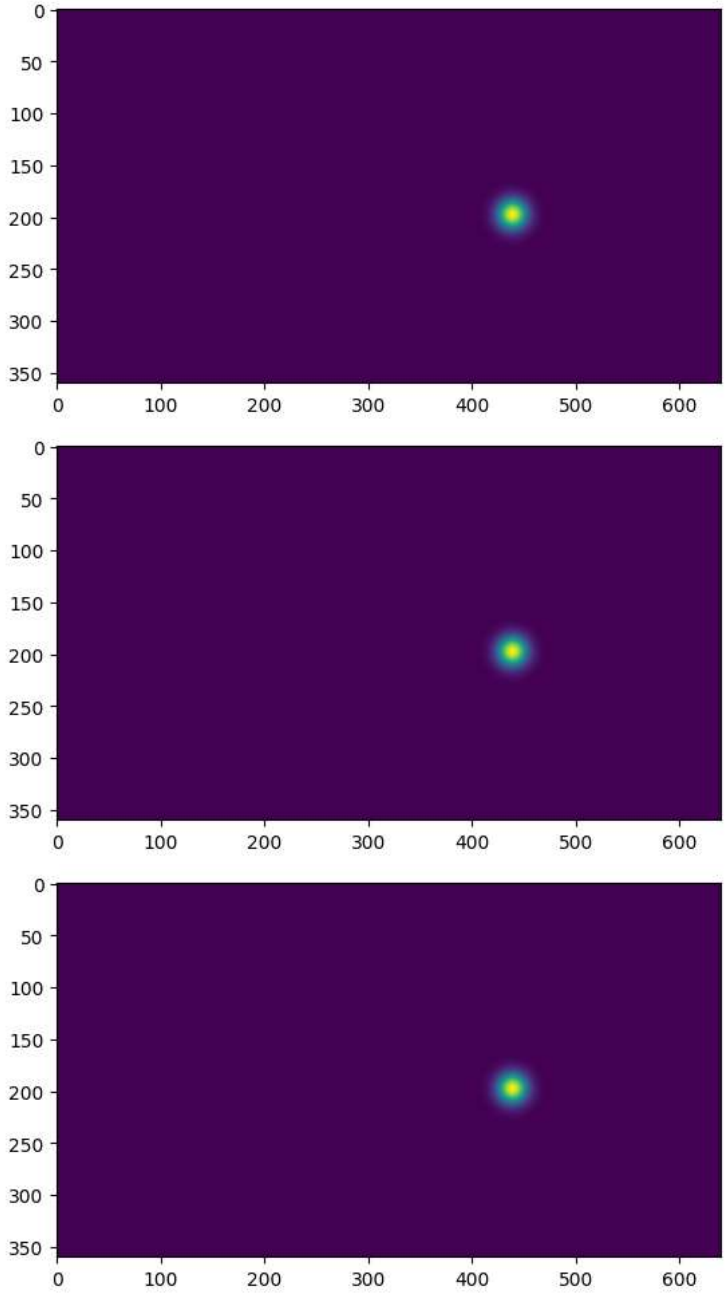
```
loading clip data (game 3, clip 1) downscaled
loading clip labels (game 3, clip 1)
loading clip data (game 2, clip 3) downscaled
loading clip labels (game 2, clip 3)
loading clip data (game 4, clip 4) downscaled
loading clip labels (game 4, clip 4)
loading clip data (game 1, clip 6) downscaled
loading clip labels (game 1, clip 6)
loading clip data (game 1, clip 12) downscaled
loading clip labels (game 1, clip 12)
loading clip data (game 4, clip 13) downscaled
loading clip labels (game 4, clip 13)
loading clip data (game 2, clip 4) downscaled
loading clip labels (game 2, clip 4)
loading clip data (game 4, clip 5) downscaled
loading clip labels (game 4, clip 5)
loading clip data (game 2, clip 5) downscaled
loading clip labels (game 2, clip 5)
loading clip data (game 4, clip 10) downscaled
loading clip labels (game 4, clip 10)
(4, 360, 640, 9) uint8 (4, 360, 640) float32
(4, 360, 640, 9) uint8 (4, 360, 640) float32
(4, 360, 640, 9) uint8 (4, 360, 640) float32
(4, 360, 640, 9) uint8 (4, 360, 640) float32
(4, 360, 640, 9) uint8 (4, 360, 640) float32
(4, 360, 640, 9) uint8 (4, 360, 640) float32
(4, 360, 640, 9) uint8 (4, 360, 640) float32
(4, 360, 640, 9) uint8 (4, 360, 640) float32
(4, 360, 640, 9) uint8 (4, 360, 640) float32
(4, 360, 640, 9) uint8 (4, 360, 640) float32
```

```
import matplotlib.pyplot as plt

stack_s = 3
train_gen = DataGenerator(Path('../input/train/'), [2], stack_s=stack_s, downscale=True, pool_s=10, pool_update_s=4, quiet=False)
stack, mask = train_gen.get_random_stack()
print(stack.shape, mask.shape)

for i in range(stack_s):
    plt.figure()
    # plt.imshow(stack[:, :, 3 * i: 3 * i + 3])
    plt.imshow(mask[:, :])
```

```
loading clip data (game 2, clip 1) downscaled
loading clip labels (game 2, clip 1)
loading clip data (game 2, clip 2) downscaled
loading clip labels (game 2, clip 2)
loading clip data (game 2, clip 3) downscaled
loading clip labels (game 2, clip 3)
loading clip data (game 2, clip 4) downscaled
loading clip labels (game 2, clip 4)
loading clip data (game 2, clip 5) downscaled
loading clip labels (game 2, clip 5)
loading clip data (game 2, clip 6) downscaled
loading clip labels (game 2, clip 6)
loading clip data (game 2, clip 7) downscaled
loading clip labels (game 2, clip 7)
loading clip data (game 2, clip 8) downscaled
loading clip labels (game 2, clip 8)
loading clip data (game 2, clip 9) downscaled
loading clip labels (game 2, clip 9)
(360, 640, 9) (360, 640)
```



Класс Metrics

Класс для вычисления метрики качества трекинга SiBaTrAcc. Функция evaluate_predictions принимает массив из референсных и предсказанных координат мяча для клипа и возвращает массив аккумулированных значений SiBaTrAcc (может быть полезно для визуализации результатов предсказания) и итоговое значение метрики SiBaTrAcc.

```
class Metrics:
```

```
    @staticmethod
    def position_error(label_gt: np.ndarray, label_pr: np.ndarray, step=8, alpha=1.5, e1=5, e2=5):
        # gt codes:
        # 0 - the ball is not within the image
        # 1 - the ball can easily be identified
        # 2 - the ball is in the frame, but is not easy to identify
        # 3 - the ball is occluded
        if label_gt[0] != 0 and label_pr[0] == 0:
            return e1
        if label_gt[0] == 0 and label_pr[0] != 0:
            return e2
        print(f"Gt: {label_gt}, pred: {label_pr}")
        dist = math.sqrt((label_gt[1] - label_pr[1]) ** 2 + (label_gt[2] - label_pr[2]) ** 2)
        pe = math.floor(dist / step) ** alpha
        pe = min(pe, 5)
        return pe

    @staticmethod
    def evaluate_predictions(labels_gt, labels_pr) -> Tuple[List[float], float]:
        pe = [Metrics.position_error(labels_gt[i, ...], labels_pr[i, ...]) for i in range(len(labels_gt))]
        SIBATRACC = []
        for i, _ in enumerate(pe):
            SIBATRACC.append(1 - sum(pe[: i + 1]) / ((i + 1) * 5))
        SIBATRACC_total = 1 - sum(pe) / (len(labels_gt) * 5)
        return SIBATRACC, SIBATRACC_total
```

Основной класс модели SuperTrackingModel

Реализует всю логику обучения, сохранения, загрузки и тестирования разработанной модели трекинга. Этот класс можно и нужно расширять.

В качестве примера вам предлагается заготовка модели, в которой трекинг осуществляется за счет предсказания маски по входному батчу и последующему предсказанию координат мяча по полученной маски. В данном варианте вызов функции предсказания координат по клипу (predict) повлечет за собой разбиение клипа на батчи, вызов предсказания маски для каждого батча, склеивание результатов в последовательность масок, вызов функции по вычислению координат мяча по маскам и возвращения результата. Описанные действия уже реализованы, вам остается только написать функции predict_on_bath и get_labels_from_prediction. Эта же функция predict используется и в вызове функции test, дополнительно вычисляя метрику качества трекинга и при необходимости визуализируя результат тестирования. Обратите внимание, что в результирующем numpy массиве с координатами помимо значений x и y первым значением в каждой строке должно идти значение code (0, если мяча в кадре нет и > 0, если мяч в кадре есть) для корректного вычисления качества трекинга.

Вам разрешается менять логику работы класса модели, (например, если решение не подразумевает использование масок), но при этом логика и работа функций load и test должна остаться неизменной!

```
from tensorflow.keras import backend as K# исправить


def IoU(y_true, y_pred, epsilon=1e-7):
    y_true_fl = K.flatten(y_true)
    y_pred_fl = K.flatten(y_pred)
    intersection = K.sum(y_true_fl * y_pred_fl)
    union = K.sum(y_true_fl + y_pred_fl) - intersection
    return intersection / (union + epsilon)

def IoU_loss(y_true, y_pred):
    return 1 - IoU(y_true, y_pred)


def smooth_loss(y_true, y_pred):
    diff = tf.abs(y_pred[:, 1:] - y_pred[:, :-1])
    return tf.reduce_mean(diff)

def IoU_loss_with_smoothness(y_true, y_pred):
    iou_loss = IoU_loss(y_true, y_pred)
    smoothness_penalty = smooth_loss(y_true, y_pred)
    return iou_loss + 0.1 * smoothness_penalty
```

Пример пайплайна для обучения модели:

```
import numpy as np
import gdown
import keras
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import backend as K
```

```
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from tensorflow.keras.callbacks import ModelCheckpoint
import os
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt
from tensorflow.keras.models import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D, concatenate,BatchNormalization, Dropout,Conv2DTranspose
import tensorflow as tf
class SuperTrackingModel:

    def __init__(self, batch_s, stack_s, out_path, downscale):
        self.batch_s = batch_s
        self.stack_s = stack_s
        self.out_path = out_path
        self.downscale = downscale
        self.model = self.build_model()

    def build_model(self):
        inputs = Input(shape=(360, 640, 15))#  проверить формат
        c1 = Conv2D(32, (3, 3), activation='relu', padding='same', strides=1)(inputs)
        c1 = Conv2D(32, (3, 3), activation='relu', padding='same', strides=1)(c1)
        c1 = BatchNormalization()(c1)
        p1 = MaxPooling2D((2, 2))(c1)

        c2 = Conv2D(64, (3, 3), activation='relu', padding='same', strides=1)(p1)
        c2 = Conv2D(64, (3, 3), activation='relu', padding='same', strides=1)(c2)
        c2 = BatchNormalization()(c2)
        p2 = MaxPooling2D((2, 2))(c2)
        p2 = Dropout(0.3)(p2)

        c4 = Conv2D(128, (3, 3), activation = "relu", padding = "same", strides=1)(p2)
        c4 = Conv2D(128, (3, 3), activation = "relu", padding = "same", strides=1)(c4)
        c4 = BatchNormalization()(c4)

        u2 = Conv2DTranspose(16, (3, 3), strides=(2, 2), padding="same")(c4)
        u2 = concatenate([u2, c2])
        u2 = Dropout(0.3)(u2)
        c6 = Conv2D(64, (3, 3), activation = "relu", padding = "same",strides=1)(u2)
        c6 = Conv2D(64, (3, 3), activation = "relu", padding = "same",strides=1)(c6)
        c6 = BatchNormalization()(c6)


        u3 = Conv2DTranspose(8, (3, 3), strides=(2, 2), padding="same")(c6)
        u3 = concatenate([u3, c1])
        c7 = Conv2D(64, (3, 3), activation = "relu", padding = "same", strides=1)(u3)
        c7 = Conv2D(64, (3, 3), activation = "relu", padding = "same", strides=1)(c7)
        c7 = BatchNormalization()(c7)
        #attention = MultiHeadAttention(num_heads=2, key_dim=16)(c7, c7)
        #c7 = concatenate([c7, attention])

        outputs = Conv2D(1, (1, 1), activation='sigmoid')(c7)
        model = Model(inputs=[inputs], outputs=[outputs])

        return model


    def load(self):
        print('Running stub for loading model ...')
        id = '1DlgGHnHu7KnKLOFKcEX3JlkECNbyervh'
        url = f'https://drive.google.com/uc?id={id}'
        output = 'model111.weights.h5'
        gdown.download(url, output, quiet=False)
        self.model.load_weights("/kaggle/working/model111.weights.h5")
        print('Loading model done.')

    def predict_on_batch(self, batch: np.ndarray) -> np.ndarray:
        predictions = self.model.predict(batch)
        return predictions

    def _predict_prob_on_clip(self, clip: np.ndarray) -> np.ndarray:
        print('doing predictions')
        n_frames = clip.shape[0]
        # --- get stacks ---
        stacks = []
        for i in range(n_frames - self.stack_s + 1):
            stack = clip[i : i + self.stack_s, ...]
            stack = np.squeeze(np.split(stack, self.stack_s, axis=0))
```

```
        stack = np.concatenate(stack, axis=-1)
        stacks.append(stack)
    # --- round to batch size ---
    add_stacks = 0
    while len(stacks) % self.batch_s != 0:
        stacks.append(stacks[-1])
        add_stacks += 1
    # --- group into batches ---
    batches = []
    for i in range(len(stacks) // self.batch_s):
        batch = np.stack(stacks[i * self.batch_s : (i + 1) * self.batch_s])
        batches.append(batch)
    stacks.clear()
    # --- perform predictions ---
    predictions = []
    for batch in batches:
        pred = np.squeeze(self.predict_on_batch(batch))
        predictions.append(pred)
    # --- crop back to source length ---
    predictions = np.concatenate(predictions, axis=0)
    if (add_stacks > 0):
        predictions = predictions[:-add_stacks, ...]
    batches.clear()
    # --- add (stack_s - 1) null frames at the begining ---
    start_frames = np.zeros((stack_s - 1, predictions.shape[1], predictions.shape[2]), dtype=np.float32)
    predictions = np.concatenate((start_frames, predictions), axis=0)
    print('predictions are made')
    return predictions

def get_labels_from_prediction(self, pred_prob: np.ndarray, upscale_coords: bool) -> np.ndarray:
    n_frames = pred_prob.shape[0]
    coords = np.zeros([n_frames, 3])
    for i in range(n_frames):
        mask = pred_prob[i]
        x, y = mask.sum(axis=0).argmax(), mask.sum(axis=1).argmax()
        h, w = mask.sum(axis=0).max(), mask.sum(axis=1).max()
        code = 0 if (h < 10) or (w < 10) else 1
        if upscale_coords:
            x, y = x * 2, y * 2
        coords[i] = [code, x, y]
    return coords

def postprocess_labels(self, labels):
    threshold = 300
    filtered_x, filtered_y = [labels[0][1]], [labels[0][2]]
    for i in range(1, len(labels)-1):
        if labels[i][0] == 0 and labels[i-1][0] == 1 and labels[i+1][0] == 1:
            labels[i][0] = 1
        '''
        if labels[i][0] == 1:
            if abs(labels[i][1] - labels[i-1][1]) > threshold and abs(labels[i][2] - labels[i-1][2]) > threshold:
                labels[i][1] = labels[i-1][1]
                labels[i][2] = labels[i-1][2]'''

    return labels

def predict(self, clip: np.ndarray, upscale_coords=True) -> np.ndarray:
    prob_pr = self._predict_prob_on_clip(clip)
    labels_pr = self.get_labels_from_prediction(prob_pr, upscale_coords)
    labels = self.postprocess_labels(labels_pr)
    return labels, prob_pr

def test(self, data_path: Path, games: List[int], do_visualization=False, test_name='test'):
    game_clip_pairs = get_game_clip_pairs(data_path, games)
    SIBATRACC_vals = []
    for game, clip in game_clip_pairs:
        data = load_clip_data(data_path, game, clip, downscale=self.downscale)
        if do_visualization:
            data_full = load_clip_data(data_path, game, clip, downscale=False) if self.downscale else data
        labels_gt = load_clip_labels(data_path, game, clip, downscale=False)
        labels_pr, prob_pr = self.predict(data)
        SIBATRACC_per_frame, SIBATRACC_total = Metrics.evaluate_predictions(labels_gt, labels_pr)
        SIBATRACC_vals.append(SIBATRACC_total)
        if do_visualization:
            visualize_prediction(data_full, labels_pr, self.out_path, f'{test_name}_g{game}_c{clip}', SIBATRACC_per_frame)
            visualize_prob(data, prob_pr, self.out_path, f'{test_name}_g{game}_c{clip}')
            del data_full
        del data, labels_gt, labels_pr, prob_pr
        gc.collect()
    SIBATRACC_final = sum(SIBATRACC_vals) / len(SIBATRACC_vals)
    return SIBATRACC_final
```

```
    def train(self, train, val):
        print('Running stub for training model...')
        self.model.compile(Adam(learning_rate=4e-4), loss=IoU_loss_with_smoothness, metrics=[IoU])
        file_path_best = "/kaggle/working/model111.weights.h5"

        modelcheckpoint_best = keras.callbacks.ModelCheckpoint(file_path_best,
                                                               monitor='val_loss',
                                                               mode='auto',
                                                               verbose=1,
                                                               save_best_only=True,
                                                               save_weights_only=True)


        history = self.model.fit(train(self.batch_s),
                                 steps_per_epoch=200,
                                 epochs=50,
                                 callbacks=[modelcheckpoint_best],
                                 validation_data=val(self.batch_s),
                                 validation_steps=50)
        print('training done.')
        plt.figure(figsize=(10, 5))
        plt.plot(history.history['loss'], color='purple', label='Training Loss')
        plt.plot(history.history['val_loss'], color='violet', label='Validation Loss')
        plt.title('Training and Validation Loss')
        plt.xlabel('Epochs')
        plt.ylabel('Loss')
        plt.legend()
        plt.grid()
        plt.show()


batch_s = 4
stack_s = 5
downscale = True

output_path = prepare_experiment(Path('/kaggle/working'))

model = SuperTrackingModel(batch_s, stack_s, out_path=output_path, downscale=downscale)

train_gen = DataGenerator(Path('../input/train/'), [1,2,3,4,5,6], stack_s=stack_s, downscale=True, pool_s=10, pool_update_s=7, quiet=Fa
val_gen = DataGenerator(Path('../input/test/'), [1,2], stack_s=stack_s, downscale=True, pool_s=4, pool_update_s=3, quiet=False)

model.train(train_gen.random_g, val_gen.random_g)
```

```
loading clip data (game 4, clip 4) downscaled
loading clip labels (game 4, clip 4)
loading clip data (game 3, clip 5) downscaled
loading clip labels (game 3, clip 5)
loading clip data (game 1, clip 2) downscaled
loading clip labels (game 1, clip 2)
loading clip data (game 5, clip 9) downscaled
loading clip labels (game 5, clip 9)
loading clip data (game 5, clip 1) downscaled
loading clip labels (game 5, clip 1)
loading clip data (game 6, clip 6) downscaled
loading clip labels (game 6, clip 6)
loading clip data (game 4, clip 2) downscaled
loading clip labels (game 4, clip 2)
loading clip data (game 3, clip 4) downscaled
loading clip labels (game 3, clip 4)
loading clip data (game 6, clip 9) downscaled
loading clip labels (game 6, clip 9)
loading clip data (game 6, clip 4) downscaled
loading clip labels (game 6, clip 4)
loading clip data (game 2, clip 8) downscaled
loading clip labels (game 2, clip 8)
loading clip data (game 1, clip 8) downscaled
loading clip labels (game 1, clip 8)
loading clip data (game 2, clip 5) downscaled
loading clip labels (game 2, clip 5)
loading clip data (game 2, clip 6) downscaled
loading clip labels (game 2, clip 6)
Running stub for training model...
Epoch 1/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 222ms/step - io_u: 0.0023 - loss: 0.9999
Epoch 1: val_loss improved from inf to 0.99750, saving model to /kaggle/working/model111.weights.h5
200/200 ━━━━━━━━━━━━━━━━━━━━ 86s 275ms/step - io_u: 0.0023 - loss: 0.9999 - val_io_u: 0.0027 - val_loss: 0.9975
Epoch 2/50
160/200 ━━━━━━━━━━━━━━━━──── 8s 222ms/step - io_u: 0.0031 - loss: 0.9971loading clip data (game 3, clip 3) downscaled
161/200 ━━━━━━━━━━━━━━━━──── 8s 224ms/step - io_u: 0.0031 - loss: 0.9971loading clip labels (game 3, clip 3)
loading clip data (game 4, clip 7) downscaled
loading clip labels (game 4, clip 7)
loading clip data (game 3, clip 2) downscaled
loading clip labels (game 3, clip 2)
loading clip data (game 5, clip 8) downscaled
loading clip labels (game 5, clip 8)
loading clip data (game 1, clip 2) downscaled
loading clip labels (game 1, clip 2)
loading clip data (game 6, clip 2) downscaled
loading clip labels (game 6, clip 2)
loading clip data (game 6, clip 7) downscaled
loading clip labels (game 6, clip 7)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 264ms/step - io_u: 0.0037 - loss: 0.9965
Epoch 2: val_loss improved from 0.99750 to 0.99398, saving model to /kaggle/working/model111.weights.h5
200/200 ━━━━━━━━━━━━━━━━━━━━ 56s 279ms/step - io_u: 0.0038 - loss: 0.9965 - val_io_u: 0.0061 - val_loss: 0.9940
Epoch 3/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 223ms/step - io_u: 0.0490 - loss: 0.9513
Epoch 3: val_loss improved from 0.99398 to 0.96940, saving model to /kaggle/working/model111.weights.h5
200/200 ━━━━━━━━━━━━━━━━━━━━ 48s 239ms/step - io_u: 0.0493 - loss: 0.9509 - val_io_u: 0.0306 - val_loss: 0.9694
Epoch 4/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 232ms/step - io_u: 0.2250 - loss: 0.7750
Epoch 4: val_loss improved from 0.96940 to 0.74574, saving model to /kaggle/working/model111.weights.h5
200/200 ━━━━━━━━━━━━━━━━━━━━ 50s 248ms/step - io_u: 0.2252 - loss: 0.7748 - val_io_u: 0.2543 - val_loss: 0.7457
Epoch 5/50
 37/200 ━━━━──────────────── 37s 229ms/step - io_u: 0.3175 - loss: 0.6825loading clip data (game 6, clip 3) downscaled
 39/200 ━━━━──────────────── 37s 235ms/step - io_u: 0.3186 - loss: 0.6814loading clip labels (game 6, clip 3)
loading clip data (game 6, clip 1) downscaled
loading clip labels (game 6, clip 1)
loading clip data (game 1, clip 9) downscaled
loading clip labels (game 1, clip 9)
loading clip data (game 4, clip 11) downscaled
loading clip labels (game 4, clip 11)
loading clip data (game 2, clip 6) downscaled
loading clip labels (game 2, clip 6)
loading clip data (game 3, clip 6) downscaled
loading clip labels (game 3, clip 6)
loading clip data (game 1, clip 10) downscaled
loading clip labels (game 1, clip 10)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 261ms/step - io_u: 0.3295 - loss: 0.6705
Epoch 5: val_loss did not improve from 0.74574
200/200 ━━━━━━━━━━━━━━━━━━━━ 55s 276ms/step - io_u: 0.3296 - loss: 0.6705 - val_io_u: 1.4801e-05 - val_loss: 1.0000
Epoch 6/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3286 - loss: 0.6714loading clip data (game 1, clip 8) downscaled
loading clip labels (game 1, clip 8)
loading clip data (game 1, clip 2) downscaled
loading clip labels (game 1, clip 2)
loading clip data (game 2, clip 1) downscaled
loading clip labels (game 2, clip 1)

Epoch 6: val_loss did not improve from 0.74574
200/200 ━━━━━━━━━━━━━━━━━━━━ 54s 269ms/step - io_u: 0.3287 - loss: 0.6714 - val_io_u: 0.0499 - val_loss: 0.9501
Epoch 7/50
127/200 ━━━━━━━━━━━──────── 16s 229ms/step - io_u: 0.3437 - loss: 0.6563loading clip data (game 6, clip 6) downscaled
130/200 ━━━━━━━━━━━──────── 16s 230ms/step - io_u: 0.3438 - loss: 0.6562loading clip labels (game 6, clip 6)
```

```
loading clip data (game 4, clip 14) downscaled
131/200 ━━━━━━━━━━━━━━━━━━━━ 15s 230ms/step - io_u: 0.3438 - loss: 0.6562loading clip labels (game 4, clip 14)
loading clip data (game 1, clip 10) downscaled
loading clip labels (game 1, clip 10)
loading clip data (game 4, clip 12) downscaled
loading clip labels (game 4, clip 12)
loading clip data (game 6, clip 5) downscaled
loading clip labels (game 6, clip 5)
loading clip data (game 3, clip 4) downscaled
loading clip labels (game 3, clip 4)
loading clip data (game 2, clip 1) downscaled
loading clip labels (game 2, clip 1)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 245ms/step - io_u: 0.3455 - loss: 0.6545
Epoch 7: val_loss improved from 0.74574 to 0.62959, saving model to /kaggle/working/model111.weights.h5
200/200 ━━━━━━━━━━━━━━━━━━━━ 52s 260ms/step - io_u: 0.3455 - loss: 0.6545 - val_io_u: 0.3704 - val_loss: 0.6296
Epoch 8/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 227ms/step - io_u: 0.3355 - loss: 0.6645
Epoch 8: val_loss improved from 0.62959 to 0.62527, saving model to /kaggle/working/model111.weights.h5
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.3356 - loss: 0.6645 - val_io_u: 0.3748 - val_loss: 0.6253
Epoch 9/50
138/200 ━━━━━━━━━━━━━━━━━━━━ 14s 229ms/step - io_u: 0.3597 - loss: 0.6403loading clip data (game 4, clip 10) downscaled
140/200 ━━━━━━━━━━━━━━━━━━━━ 13s 230ms/step - io_u: 0.3595 - loss: 0.6406loading clip labels (game 4, clip 10)
141/200 ━━━━━━━━━━━━━━━━━━━━ 13s 230ms/step - io_u: 0.3594 - loss: 0.6407loading clip data (game 4, clip 5) downscaled
142/200 ━━━━━━━━━━━━━━━━━━━━ 13s 230ms/step - io_u: 0.3593 - loss: 0.6408loading clip labels (game 4, clip 5)
143/200 ━━━━━━━━━━━━━━━━━━━━ 13s 230ms/step - io_u: 0.3592 - loss: 0.6409loading clip data (game 6, clip 3) downscaled
146/200 ━━━━━━━━━━━━━━━━━━━━ 12s 230ms/step - io_u: 0.3589 - loss: 0.6412loading clip labels (game 6, clip 3)
loading clip data (game 4, clip 6) downscaled
loading clip labels (game 4, clip 6)
loading clip data (game 5, clip 7) downscaled
loading clip labels (game 5, clip 7)
loading clip data (game 3, clip 5) downscaled
loading clip labels (game 3, clip 5)
loading clip data (game 2, clip 3) downscaled
loading clip labels (game 2, clip 3)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 262ms/step - io_u: 0.3539 - loss: 0.6461
Epoch 9: val_loss did not improve from 0.62527
200/200 ━━━━━━━━━━━━━━━━━━━━ 55s 277ms/step - io_u: 0.3538 - loss: 0.6462 - val_io_u: 0.2511 - val_loss: 0.7489
Epoch 10/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 227ms/step - io_u: 0.3494 - loss: 0.6506
Epoch 10: val_loss improved from 0.62527 to 0.60369, saving model to /kaggle/working/model111.weights.h5
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.3494 - loss: 0.6507 - val_io_u: 0.3963 - val_loss: 0.6037
Epoch 11/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3465 - loss: 0.6535loading clip data (game 1, clip 1) downscaled
loading clip labels (game 1, clip 1)
loading clip data (game 2, clip 9) downscaled
loading clip labels (game 2, clip 9)
loading clip data (game 2, clip 6) downscaled
loading clip labels (game 2, clip 6)

Epoch 11: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 52s 260ms/step - io_u: 0.3465 - loss: 0.6535 - val_io_u: 0.3206 - val_loss: 0.6795
Epoch 12/50
 55/200 ━━━━━━━━━━━━━━━━━━━━ 32s 226ms/step - io_u: 0.3488 - loss: 0.6512loading clip data (game 5, clip 7) downscaled
 58/200 ━━━━━━━━━━━━━━━━━━━━ 32s 229ms/step - io_u: 0.3490 - loss: 0.6510loading clip labels (game 5, clip 7)
loading clip data (game 4, clip 13) downscaled
 59/200 ━━━━━━━━━━━━━━━━━━━━ 32s 229ms/step - io_u: 0.3491 - loss: 0.6510loading clip labels (game 4, clip 13)
loading clip data (game 1, clip 9) downscaled
 61/200 ━━━━━━━━━━━━━━━━━━━━ 31s 229ms/step - io_u: 0.3491 - loss: 0.6509loading clip labels (game 1, clip 9)
loading clip data (game 2, clip 1) downscaled
 64/200 ━━━━━━━━━━━━━━━━━━━━ 31s 229ms/step - io_u: 0.3491 - loss: 0.6509loading clip labels (game 2, clip 1)
 65/200 ━━━━━━━━━━━━━━━━━━━━ 30s 229ms/step - io_u: 0.3491 - loss: 0.6509loading clip data (game 1, clip 1) downscaled
 68/200 ━━━━━━━━━━━━━━━━━━━━ 30s 229ms/step - io_u: 0.3491 - loss: 0.6510loading clip labels (game 1, clip 1)
loading clip data (game 6, clip 7) downscaled
 71/200 ━━━━━━━━━━━━━━━━━━━━ 29s 229ms/step - io_u: 0.3491 - loss: 0.6509loading clip labels (game 6, clip 7)
loading clip data (game 6, clip 6) downscaled
loading clip labels (game 6, clip 6)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 240ms/step - io_u: 0.3537 - loss: 0.6463
Epoch 12: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 51s 255ms/step - io_u: 0.3537 - loss: 0.6463 - val_io_u: 3.5077e-07 - val_loss: 1.0000
Epoch 13/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 227ms/step - io_u: 0.3540 - loss: 0.6460
Epoch 13: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 48s 242ms/step - io_u: 0.3540 - loss: 0.6460 - val_io_u: 0.0531 - val_loss: 0.9469
Epoch 14/50
 78/200 ━━━━━━━━━━━━━━━━━━━━ 27s 228ms/step - io_u: 0.3738 - loss: 0.6263loading clip data (game 3, clip 7) downscaled
 89/200 ━━━━━━━━━━━━━━━━━━━━ 25s 230ms/step - io_u: 0.3724 - loss: 0.6276loading clip labels (game 3, clip 7)
 90/200 ━━━━━━━━━━━━━━━━━━━━ 25s 230ms/step - io_u: 0.3723 - loss: 0.6277loading clip data (game 2, clip 7) downscaled
 95/200 ━━━━━━━━━━━━━━━━━━━━ 24s 230ms/step - io_u: 0.3719 - loss: 0.6282loading clip labels (game 2, clip 7)
 96/200 ━━━━━━━━━━━━━━━━━━━━ 23s 230ms/step - io_u: 0.3718 - loss: 0.6283loading clip data (game 1, clip 4) downscaled
 98/200 ━━━━━━━━━━━━━━━━━━━━ 23s 230ms/step - io_u: 0.3716 - loss: 0.6284loading clip labels (game 1, clip 4)
loading clip data (game 3, clip 6) downscaled
103/200 ━━━━━━━━━━━━━━━━━━━━ 22s 230ms/step - io_u: 0.3712 - loss: 0.6288loading clip labels (game 3, clip 6)
loading clip data (game 6, clip 3) downscaled
110/200 ━━━━━━━━━━━━━━━━━━━━ 20s 229ms/step - io_u: 0.3708 - loss: 0.6293loading clip labels (game 6, clip 3)
loading clip data (game 6, clip 5) downscaled
loading clip labels (game 6, clip 5)
loading clip data (game 2, clip 2) downscaled
loading clip labels (game 2, clip 2)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 235ms/step - io_u: 0.3657 - loss: 0.6343
Epoch 14: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 50s 250ms/step - io_u: 0.3657 - loss: 0.6344 - val_io_u: 0.0721 - val_loss: 0.9279
```

```
200/200 ━━━━━━━━━━━━━━━━━━━━ 50s 250ms/step - io_u: 0.3697 - loss: 0.6544 - val_io_u: 0.0721 - val_loss: 0.9279
Epoch 15/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 227ms/step - io_u: 0.3447 - loss: 0.6553
Epoch 15: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 48s 242ms/step - io_u: 0.3447 - loss: 0.6553 - val_io_u: 0.1782 - val_loss: 0.8218
Epoch 16/50
142/200 ━━━━━━━━━━━━━━━━━━━━ 13s 228ms/step - io_u: 0.3586 - loss: 0.6414loading clip data (game 3, clip 5) downscaled
147/200 ━━━━━━━━━━━━━━━━━━━━ 12s 229ms/step - io_u: 0.3581 - loss: 0.6419loading clip labels (game 3, clip 5)
148/200 ━━━━━━━━━━━━━━━━━━━━ 11s 229ms/step - io_u: 0.3580 - loss: 0.6420loading clip data (game 4, clip 3) downscaled
149/200 ━━━━━━━━━━━━━━━━━━━━ 11s 229ms/step - io_u: 0.3579 - loss: 0.6421loading clip labels (game 4, clip 3)
150/200 ━━━━━━━━━━━━━━━━━━━━ 11s 229ms/step - io_u: 0.3578 - loss: 0.6422loading clip data (game 4, clip 10) downscaled
152/200 ━━━━━━━━━━━━━━━━━━━━ 10s 229ms/step - io_u: 0.3576 - loss: 0.6425loading clip labels (game 4, clip 10)
loading clip data (game 6, clip 7) downscaled
161/200 ━━━━━━━━━━━━━━━━━━━━ 8s 229ms/step - io_u: 0.3566 - loss: 0.6434loading clip labels (game 6, clip 7)
162/200 ━━━━━━━━━━━━━━━━━━━━ 8s 229ms/step - io_u: 0.3565 - loss: 0.6435loading clip data (game 4, clip 9) downscaled
165/200 ━━━━━━━━━━━━━━━━━━━━ 8s 229ms/step - io_u: 0.3562 - loss: 0.6438loading clip labels (game 4, clip 9)
166/200 ━━━━━━━━━━━━━━━━━━━━ 7s 229ms/step - io_u: 0.3561 - loss: 0.6439loading clip data (game 3, clip 7) downscaled
176/200 ━━━━━━━━━━━━━━━━━━━━ 5s 229ms/step - io_u: 0.3552 - loss: 0.6449loading clip labels (game 3, clip 7)
177/200 ━━━━━━━━━━━━━━━━━━━━ 5s 229ms/step - io_u: 0.3551 - loss: 0.6449loading clip data (game 2, clip 7) downscaled
182/200 ━━━━━━━━━━━━━━━━━━━━ 4s 229ms/step - io_u: 0.3547 - loss: 0.6453loading clip labels (game 2, clip 7)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3537 - loss: 0.6463
Epoch 16: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 244ms/step - io_u: 0.3537 - loss: 0.6464 - val_io_u: 0.3508 - val_loss: 0.6492
Epoch 17/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3525 - loss: 0.6475loading clip data (game 2, clip 7) downscaled
loading clip labels (game 2, clip 7)
loading clip data (game 1, clip 3) downscaled
loading clip labels (game 1, clip 3)
loading clip data (game 2, clip 8) downscaled
loading clip labels (game 2, clip 8)

Epoch 17: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 244ms/step - io_u: 0.3525 - loss: 0.6475 - val_io_u: 0.0924 - val_loss: 0.9076
Epoch 18/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 227ms/step - io_u: 0.3760 - loss: 0.6240loading clip data (game 1, clip 8) downscaled

Epoch 18: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 244ms/step - io_u: 0.3760 - loss: 0.6240 - val_io_u: 0.3405 - val_loss: 0.6595
Epoch 19/50
 12/200 ━━━━━━━━━━━━━━━━━━━━ 42s 228ms/step - io_u: 0.4003 - loss: 0.5998loading clip labels (game 1, clip 8)
 16/200 ━━━━━━━━━━━━━━━━━━━━ 42s 229ms/step - io_u: 0.3996 - loss: 0.6004loading clip data (game 1, clip 5) downscaled
 20/200 ━━━━━━━━━━━━━━━━━━━━ 41s 229ms/step - io_u: 0.3983 - loss: 0.6017loading clip labels (game 1, clip 5)
loading clip data (game 2, clip 5) downscaled
 26/200 ━━━━━━━━━━━━━━━━━━━━ 39s 229ms/step - io_u: 0.3963 - loss: 0.6037loading clip labels (game 2, clip 5)
 92/200 ━━━━━━━━━━━━━━━━━━━━ 24s 228ms/step - io_u: 0.3858 - loss: 0.6142loading clip data (game 6, clip 5) downscaled
 94/200 ━━━━━━━━━━━━━━━━━━━━ 24s 230ms/step - io_u: 0.3857 - loss: 0.6143loading clip labels (game 6, clip 5)
loading clip data (game 5, clip 7) downscaled
 97/200 ━━━━━━━━━━━━━━━━━━━━ 23s 230ms/step - io_u: 0.3855 - loss: 0.6145loading clip labels (game 5, clip 7)
loading clip data (game 3, clip 7) downscaled
107/200 ━━━━━━━━━━━━━━━━━━━━ 21s 230ms/step - io_u: 0.3849 - loss: 0.6151loading clip labels (game 3, clip 7)
108/200 ━━━━━━━━━━━━━━━━━━━━ 21s 230ms/step - io_u: 0.3849 - loss: 0.6151loading clip data (game 1, clip 3) downscaled
109/200 ━━━━━━━━━━━━━━━━━━━━ 20s 230ms/step - io_u: 0.3849 - loss: 0.6152loading clip labels (game 1, clip 3)
loading clip data (game 4, clip 13) downscaled
110/200 ━━━━━━━━━━━━━━━━━━━━ 20s 230ms/step - io_u: 0.3848 - loss: 0.6152loading clip labels (game 4, clip 13)
loading clip data (game 2, clip 1) downscaled
113/200 ━━━━━━━━━━━━━━━━━━━━ 20s 230ms/step - io_u: 0.3848 - loss: 0.6153loading clip labels (game 2, clip 1)
loading clip data (game 3, clip 3) downscaled
116/200 ━━━━━━━━━━━━━━━━━━━━ 19s 230ms/step - io_u: 0.3847 - loss: 0.6153loading clip labels (game 3, clip 3)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3820 - loss: 0.6180
Epoch 19: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 244ms/step - io_u: 0.3820 - loss: 0.6180 - val_io_u: 0.3783 - val_loss: 0.6217
Epoch 20/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3947 - loss: 0.6053
Epoch 20: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.3947 - loss: 0.6054 - val_io_u: 0.3516 - val_loss: 0.6484
Epoch 21/50
170/200 ━━━━━━━━━━━━━━━━━━━━ 6s 227ms/step - io_u: 0.3860 - loss: 0.6140loading clip data (game 2, clip 5) downscaled
173/200 ━━━━━━━━━━━━━━━━━━━━ 6s 229ms/step - io_u: 0.3861 - loss: 0.6139loading clip labels (game 2, clip 5)
loading clip data (game 3, clip 7) downscaled
183/200 ━━━━━━━━━━━━━━━━━━━━ 3s 229ms/step - io_u: 0.3862 - loss: 0.6138loading clip labels (game 3, clip 7)
184/200 ━━━━━━━━━━━━━━━━━━━━ 3s 229ms/step - io_u: 0.3862 - loss: 0.6138loading clip data (game 1, clip 11) downscaled
187/200 ━━━━━━━━━━━━━━━━━━━━ 2s 229ms/step - io_u: 0.3862 - loss: 0.6138loading clip labels (game 1, clip 11)
loading clip data (game 3, clip 2) downscaled
197/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3863 - loss: 0.6137loading clip labels (game 3, clip 2)
199/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3863 - loss: 0.6137loading clip data (game 3, clip 4) downscaled
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3863 - loss: 0.6137loading clip labels (game 3, clip 4)
loading clip data (game 4, clip 7) downscaled
loading clip labels (game 4, clip 7)
loading clip data (game 1, clip 3) downscaled
loading clip labels (game 1, clip 3)

Epoch 21: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 244ms/step - io_u: 0.3863 - loss: 0.6137 - val_io_u: 0.3861 - val_loss: 0.6140
Epoch 22/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3822 - loss: 0.6178
Epoch 22: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.3822 - loss: 0.6178 - val_io_u: 0.3128 - val_loss: 0.6872
Epoch 23/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3937 - loss: 0.6063
Epoch 23: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 244ms/step - io_u: 0.3937 - loss: 0.6064 - val_io_u: 0.2818 - val_loss: 0.7182
```

```
Epoch 24/50
 74/200 ━━━━━━━━━━━━━━━━━━━━ 28s 229ms/step - io_u: 0.3759 - loss: 0.6241loading clip data (game 3, clip 6) downscaled
 80/200 ━━━━━━━━━━━━━━━━━━━━ 27s 232ms/step - io_u: 0.3765 - loss: 0.6235loading clip labels (game 3, clip 6)
loading clip data (game 5, clip 8) downscaled
 81/200 ━━━━━━━━━━━━━━━━━━━━ 27s 232ms/step - io_u: 0.3766 - loss: 0.6234loading clip labels (game 5, clip 8)
loading clip data (game 2, clip 4) downscaled
 88/200 ━━━━━━━━━━━━━━━━━━━━ 25s 231ms/step - io_u: 0.3773 - loss: 0.6227loading clip labels (game 2, clip 4)
 89/200 ━━━━━━━━━━━━━━━━━━━━ 25s 231ms/step - io_u: 0.3774 - loss: 0.6226loading clip data (game 4, clip 6) downscaled
 93/200 ━━━━━━━━━━━━━━━━━━━━ 24s 231ms/step - io_u: 0.3778 - loss: 0.6223loading clip labels (game 4, clip 6)
loading clip data (game 3, clip 2) downscaled
104/200 ━━━━━━━━━━━━━━━━━━━━ 22s 231ms/step - io_u: 0.3786 - loss: 0.6214loading clip labels (game 3, clip 2)
105/200 ━━━━━━━━━━━━━━━━━━━━ 21s 231ms/step - io_u: 0.3786 - loss: 0.6214loading clip data (game 6, clip 2) downscaled
107/200 ━━━━━━━━━━━━━━━━━━━━ 21s 231ms/step - io_u: 0.3787 - loss: 0.6213loading clip labels (game 6, clip 2)
loading clip data (game 5, clip 5) downscaled
111/200 ━━━━━━━━━━━━━━━━━━━━ 20s 231ms/step - io_u: 0.3789 - loss: 0.6211loading clip labels (game 5, clip 5)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 230ms/step - io_u: 0.3802 - loss: 0.6198loading clip data (game 1, clip 4) downscaled

Epoch 24: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 247ms/step - io_u: 0.3802 - loss: 0.6198 - val_io_u: 0.3395 - val_loss: 0.6605
Epoch 25/50
loading clip labels (game 1, clip 4)
loading clip data (game 1, clip 2) downscaled
  5/200 ━━━━━━━━━━━━━━━━━━━━ 44s 231ms/step - io_u: 0.3826 - loss: 0.6175loading clip labels (game 1, clip 2)
loading clip data (game 1, clip 3) downscaled
  6/200 ━━━━━━━━━━━━━━━━━━━━ 44s 230ms/step - io_u: 0.3871 - loss: 0.6129loading clip labels (game 1, clip 3)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3831 - loss: 0.6169
Epoch 25: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 245ms/step - io_u: 0.3831 - loss: 0.6170 - val_io_u: 0.2888 - val_loss: 0.7112
Epoch 26/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3887 - loss: 0.6113loading clip data (game 2, clip 2) downscaled

Epoch 26: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 246ms/step - io_u: 0.3887 - loss: 0.6113 - val_io_u: 0.2358 - val_loss: 0.7642
Epoch 27/50
  1/200 ━━━━━━━━━━━━━━━━━━━━ 45s 227ms/step - io_u: 0.4595 - loss: 0.5405loading clip labels (game 2, clip 2)
  2/200 ━━━━━━━━━━━━━━━━━━━━ 45s 230ms/step - io_u: 0.4401 - loss: 0.5599loading clip data (game 1, clip 6) downscaled
 13/200 ━━━━━━━━━━━━━━━━━━━━ 42s 230ms/step - io_u: 0.4191 - loss: 0.5809loading clip labels (game 1, clip 6)
 14/200 ━━━━━━━━━━━━━━━━━━━━ 42s 230ms/step - io_u: 0.4183 - loss: 0.5817loading clip data (game 2, clip 3) downscaled
 20/200 ━━━━━━━━━━━━━━━━━━━━ 41s 230ms/step - io_u: 0.4143 - loss: 0.5858loading clip labels (game 2, clip 3)
 28/200 ━━━━━━━━━━━━━━━━━━━━ 39s 230ms/step - io_u: 0.4113 - loss: 0.5887loading clip data (game 6, clip 3) downscaled
 37/200 ━━━━━━━━━━━━━━━━━━━━ 38s 235ms/step - io_u: 0.4095 - loss: 0.5906loading clip labels (game 6, clip 3
loading clip data (game 2, clip 8) downscaled
 38/200 ━━━━━━━━━━━━━━━━━━━━ 38s 235ms/step - io_u: 0.4093 - loss: 0.5907loading clip labels (game 2, clip 8)
loading clip data (game 5, clip 7) downscaled
 40/200 ━━━━━━━━━━━━━━━━━━━━ 37s 235ms/step - io_u: 0.4091 - loss: 0.5909loading clip labels (game 5, clip 7)
 41/200 ━━━━━━━━━━━━━━━━━━━━ 37s 235ms/step - io_u: 0.4090 - loss: 0.5910loading clip data (game 1, clip 9) downscaled
 43/200 ━━━━━━━━━━━━━━━━━━━━ 36s 234ms/step - io_u: 0.4087 - loss: 0.5913loading clip labels (game 1, clip 9)
loading clip data (game 6, clip 4) downscaled
 45/200 ━━━━━━━━━━━━━━━━━━━━ 36s 234ms/step - io_u: 0.4083 - loss: 0.5918loading clip labels (game 6, clip 4)
 46/200 ━━━━━━━━━━━━━━━━━━━━ 36s 234ms/step - io_u: 0.4081 - loss: 0.5919loading clip data (game 1, clip 3) downscaled
 47/200 ━━━━━━━━━━━━━━━━━━━━ 35s 234ms/step - io_u: 0.4079 - loss: 0.5921loading clip labels (game 1, clip 3)
loading clip data (game 5, clip 2) downscaled
 50/200 ━━━━━━━━━━━━━━━━━━━━ 35s 234ms/step - io_u: 0.4073 - loss: 0.5928loading clip labels (game 5, clip 2)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 231ms/step - io_u: 0.3976 - loss: 0.6024
Epoch 27: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 246ms/step - io_u: 0.3976 - loss: 0.6024 - val_io_u: 0.3310 - val_loss: 0.6690
Epoch 28/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3891 - loss: 0.6109
Epoch 28: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 244ms/step - io_u: 0.3891 - loss: 0.6110 - val_io_u: 0.0104 - val_loss: 0.9896
Epoch 29/50
 36/200 ━━━━━━━━━━━━━━━━━━━━ 37s 229ms/step - io_u: 0.3515 - loss: 0.6486loading clip data (game 6, clip 2) downscaled
 38/200 ━━━━━━━━━━━━━━━━━━━━ 38s 235ms/step - io_u: 0.3520 - loss: 0.6481loading clip labels (game 6, clip 2)
 39/200 ━━━━━━━━━━━━━━━━━━━━ 37s 235ms/step - io_u: 0.3522 - loss: 0.6479loading clip data (game 4, clip 14) downscaled
loading clip labels (game 4, clip 14)
loading clip data (game 1, clip 12) downscaled
 41/200 ━━━━━━━━━━━━━━━━━━━━ 37s 235ms/step - io_u: 0.3525 - loss: 0.6475loading clip labels (game 1, clip 12)
 42/200 ━━━━━━━━━━━━━━━━━━━━ 37s 234ms/step - io_u: 0.3527 - loss: 0.6473loading clip data (game 6, clip 6) downscaled
 44/200 ━━━━━━━━━━━━━━━━━━━━ 36s 234ms/step - io_u: 0.3531 - loss: 0.6469loading clip labels (game 6, clip 6)
loading clip data (game 5, clip 4) downscaled
 60/200 ━━━━━━━━━━━━━━━━━━━━ 32s 233ms/step - io_u: 0.3563 - loss: 0.6437loading clip labels (game 5, clip 4)
 62/200 ━━━━━━━━━━━━━━━━━━━━ 32s 233ms/step - io_u: 0.3569 - loss: 0.6432loading clip data (game 2, clip 4) downscaled
 69/200 ━━━━━━━━━━━━━━━━━━━━ 30s 232ms/step - io_u: 0.3587 - loss: 0.6414loading clip labels (game 2, clip 4)
 71/200 ━━━━━━━━━━━━━━━━━━━━ 29s 232ms/step - io_u: 0.3591 - loss: 0.6409loading clip data (game 2, clip 6) downscaled
 75/200 ━━━━━━━━━━━━━━━━━━━━ 29s 232ms/step - io_u: 0.3600 - loss: 0.6400loading clip labels (game 2, clip 6)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 230ms/step - io_u: 0.3744 - loss: 0.6256
Epoch 29: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 245ms/step - io_u: 0.3745 - loss: 0.6255 - val_io_u: 0.1471 - val_loss: 0.8529
Epoch 30/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3874 - loss: 0.6126
Epoch 30: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 244ms/step - io_u: 0.3874 - loss: 0.6126 - val_io_u: 0.3707 - val_loss: 0.6293
Epoch 31/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3864 - loss: 0.6136
Epoch 31: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.3864 - loss: 0.6136 - val_io_u: 0.3112 - val_loss: 0.6889
Epoch 32/50
  8/200 ━━━━━━━━━━━━━━━━━━━━ 43s 228ms/step - io_u: 0.3923 - loss: 0.6078loading clip data (game 6, clip 6) downscaled
 11/200 ━━━━━━━━━━━━━━━━━━━━ 46s 249ms/step - io_u: 0.3849 - loss: 0.6151loading clip labels (game 6, clip 6)
loading clip data (game 1, clip 4) downscaled
 13/200 ━━━━━━━━━━━━━━━━━━━━ 45s 245ms/step - io_u: 0.3834 - loss: 0.6166loading clip labels (game 1, clip 4)
```

```
 15/200 ━━━━━━━━━━━━━━━━━━━━ 43s 245ms/step - io_u: 0.3854 - loss: 0.6166loading clip labels (game 1, clip 4)
loading clip data (game 2, clip 1) downscaled
 16/200 ━━━━━━━━━━━━━━━━━━━━ 44s 242ms/step - io_u: 0.3817 - loss: 0.6183loading clip labels (game 2, clip 1)
 17/200 ━━━━━━━━━━━━━━━━━━━━ 44s 241ms/step - io_u: 0.3811 - loss: 0.6189loading clip data (game 1, clip 8) downscaled
 18/200 ━━━━━━━━━━━━━━━━━━━━ 43s 240ms/step - io_u: 0.3806 - loss: 0.6194loading clip labels (game 1, clip 8)
loading clip data (game 4, clip 3) downscaled
 19/200 ━━━━━━━━━━━━━━━━━━━━ 43s 239ms/step - io_u: 0.3804 - loss: 0.6196loading clip labels (game 4, clip 3)
loading clip data (game 2, clip 3) downscaled
 27/200 ━━━━━━━━━━━━━━━━━━━━ 40s 236ms/step - io_u: 0.3794 - loss: 0.6206loading clip labels (game 2, clip 3)
 28/200 ━━━━━━━━━━━━━━━━━━━━ 40s 236ms/step - io_u: 0.3794 - loss: 0.6206loading clip data (game 1, clip 6) downscaled
 29/200 ━━━━━━━━━━━━━━━━━━━━ 40s 235ms/step - io_u: 0.3792 - loss: 0.6208loading clip labels (game 1, clip 6)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3819 - loss: 0.6181loading clip data (game 1, clip 3) downscaled
loading clip labels (game 1, clip 3)
loading clip data (game 2, clip 4) downscaled
loading clip labels (game 2, clip 4)
loading clip data (game 2, clip 3) downscaled

Epoch 32: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 246ms/step - io_u: 0.3819 - loss: 0.6181 - val_io_u: 0.1950 - val_loss: 0.8050
Epoch 33/50
  5/200 ━━━━━━━━━━━━━━━━━━━━ 44s 229ms/step - io_u: 0.3792 - loss: 0.6208loading clip labels (game 2, clip 3)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3902 - loss: 0.6098
Epoch 33: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.3902 - loss: 0.6098 - val_io_u: 0.3564 - val_loss: 0.6437
Epoch 34/50
188/200 ━━━━━━━━━━━━━━━━━━━━ 2s 228ms/step - io_u: 0.3874 - loss: 0.6127loading clip data (game 1, clip 1) downscaled
192/200 ━━━━━━━━━━━━━━━━━━━━ 1s 229ms/step - io_u: 0.3874 - loss: 0.6126loading clip labels (game 1, clip 1)
loading clip data (game 1, clip 8) downscaled
193/200 ━━━━━━━━━━━━━━━━━━━━ 1s 229ms/step - io_u: 0.3874 - loss: 0.6126loading clip labels (game 1, clip 8)
loading clip data (game 1, clip 5) downscaled
195/200 ━━━━━━━━━━━━━━━━━━━━ 1s 229ms/step - io_u: 0.3874 - loss: 0.6126loading clip labels (game 1, clip 5)
196/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3874 - loss: 0.6126loading clip data (game 4, clip 12) downscaled
199/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3874 - loss: 0.6126loading clip labels (game 4, clip 12)
loading clip data (game 3, clip 7) downscaled
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step - io_u: 0.3874 - loss: 0.6126loading clip labels (game 3, clip 7)

Epoch 34: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 244ms/step - io_u: 0.3874 - loss: 0.6127 - val_io_u: 0.0631 - val_loss: 0.9369
Epoch 35/50
loading clip data (game 1, clip 4) downscaled
  1/200 ━━━━━━━━━━━━━━━━━━━━ 44s 224ms/step - io_u: 0.4587 - loss: 0.5414loading clip labels (game 1, clip 4)
  2/200 ━━━━━━━━━━━━━━━━━━━━ 45s 229ms/step - io_u: 0.4576 - loss: 0.5424loading clip data (game 4, clip 2) downscaled
  3/200 ━━━━━━━━━━━━━━━━━━━━ 45s 229ms/step - io_u: 0.4515 - loss: 0.5485loading clip labels (game 4, clip 2)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3668 - loss: 0.6332
Epoch 35: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.3669 - loss: 0.6332 - val_io_u: 0.3761 - val_loss: 0.6239
Epoch 36/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 227ms/step - io_u: 0.3866 - loss: 0.6134loading clip data (game 2, clip 3) downscaled
loading clip labels (game 2, clip 3)
loading clip data (game 2, clip 9) downscaled
loading clip labels (game 2, clip 9)
loading clip data (game 1, clip 4) downscaled
loading clip labels (game 1, clip 4)

Epoch 36: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 51s 254ms/step - io_u: 0.3867 - loss: 0.6134 - val_io_u: 0.3487 - val_loss: 0.6513
Epoch 37/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 227ms/step - io_u: 0.3860 - loss: 0.6141
Epoch 37: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 48s 242ms/step - io_u: 0.3860 - loss: 0.6140 - val_io_u: 0.3577 - val_loss: 0.6424
Epoch 38/50
 71/200 ━━━━━━━━━━━━━━━━━━━━ 29s 227ms/step - io_u: 0.3824 - loss: 0.6177loading clip data (game 4, clip 3) downscaled
 73/200 ━━━━━━━━━━━━━━━━━━━━ 29s 230ms/step - io_u: 0.3826 - loss: 0.6174loading clip labels (game 4, clip 3)
loading clip data (game 4, clip 13) downscaled
 74/200 ━━━━━━━━━━━━━━━━━━━━ 28s 230ms/step - io_u: 0.3827 - loss: 0.6173loading clip labels (game 4, clip 13)
loading clip data (game 1, clip 4) downscaled
 75/200 ━━━━━━━━━━━━━━━━━━━━ 28s 229ms/step - io_u: 0.3828 - loss: 0.6172loading clip labels (game 1, clip 4)
 76/200 ━━━━━━━━━━━━━━━━━━━━ 28s 229ms/step - io_u: 0.3830 - loss: 0.6171loading clip data (game 6, clip 9) downscaled
 80/200 ━━━━━━━━━━━━━━━━━━━━ 27s 229ms/step - io_u: 0.3834 - loss: 0.6166loading clip labels (game 6, clip 9)
 81/200 ━━━━━━━━━━━━━━━━━━━━ 27s 229ms/step - io_u: 0.3835 - loss: 0.6165loading clip data (game 1, clip 13) downscaled
 85/200 ━━━━━━━━━━━━━━━━━━━━ 26s 229ms/step - io_u: 0.3839 - loss: 0.6162loading clip labels (game 1, clip 13)
 86/200 ━━━━━━━━━━━━━━━━━━━━ 26s 229ms/step - io_u: 0.3840 - loss: 0.6161loading clip data (game 3, clip 3) downscaled
 88/200 ━━━━━━━━━━━━━━━━━━━━ 25s 229ms/step - io_u: 0.3841 - loss: 0.6159loading clip labels (game 3, clip 3)
loading clip data (game 4, clip 5) downscaled
 90/200 ━━━━━━━━━━━━━━━━━━━━ 25s 229ms/step - io_u: 0.3843 - loss: 0.6158loading clip labels (game 4, clip 5)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3897 - loss: 0.6103
Epoch 38: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.3898 - loss: 0.6103 - val_io_u: 0.3004 - val_loss: 0.6996
Epoch 39/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 227ms/step - io_u: 0.3921 - loss: 0.6079
Epoch 39: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 48s 242ms/step - io_u: 0.3921 - loss: 0.6079 - val_io_u: 0.2287 - val_loss: 0.7714
Epoch 40/50
 47/200 ━━━━━━━━━━━━━━━━━━━━ 34s 226ms/step - io_u: 0.3766 - loss: 0.6234loading clip data (game 6, clip 4) downscaled
 50/200 ━━━━━━━━━━━━━━━━━━━━ 34s 231ms/step - io_u: 0.3771 - loss: 0.6229loading clip labels (game 6, clip 4)
loading clip data (game 4, clip 11) downscaled
 51/200 ━━━━━━━━━━━━━━━━━━━━ 34s 231ms/step - io_u: 0.3773 - loss: 0.6228loading clip labels (game 4, clip 11)
loading clip data (game 4, clip 4) downscaled
 55/200 ━━━━━━━━━━━━━━━━━━━━ 33s 230ms/step - io_u: 0.3779 - loss: 0.6221loading clip labels (game 4, clip 4)
 56/200 ━━━━━━━━━━━━━━━━━━━━ 33s 230ms/step - io_u: 0.3780 - loss: 0.6220loading clip data (game 1, clip 2) downscaled
 58/200 ━━━━━━━━━━━━━━━━━━━━ 32s 230ms/step - io_u: 0.3784 - loss: 0.6217loading clip labels (game 1, clip 2)
```

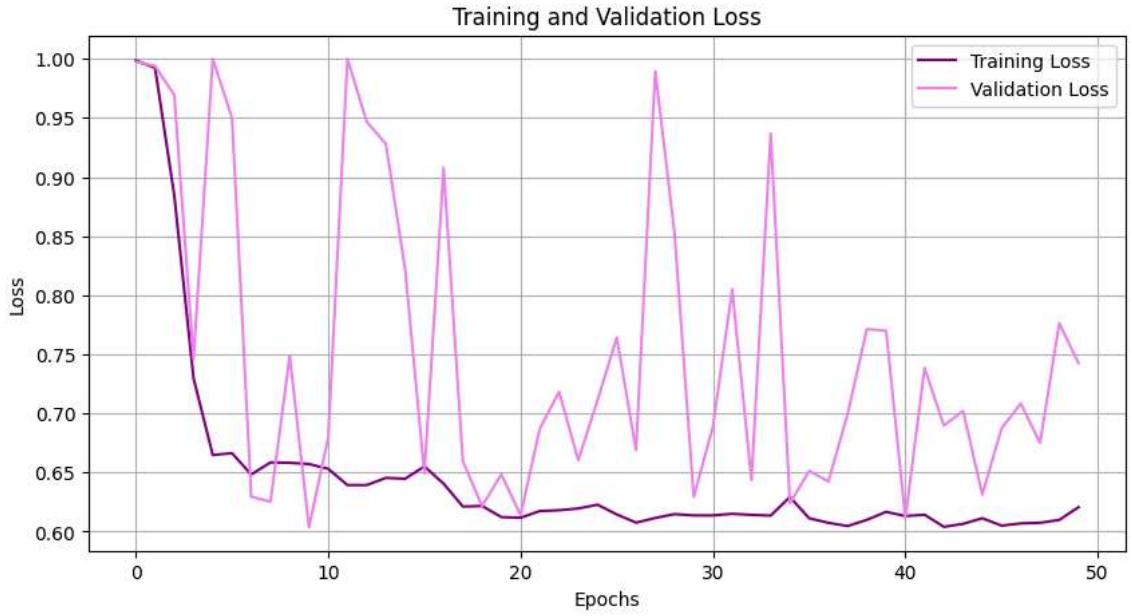
```
loading clip data (game 2, clip 1) downscaled
 61/200 ━━━━━━━━━━━━━━━━━━━━ 31s 230ms/step - io_u: 0.3788 - loss: 0.6212loading clip labels (game 2, clip 1)
 62/200 ━━━━━━━━━━━━━━━━━━━━ 31s 230ms/step - io_u: 0.3789 - loss: 0.6211loading clip data (game 1, clip 6) downscaled
 63/200 ━━━━━━━━━━━━━━━━━━━━ 31s 230ms/step - io_u: 0.3791 - loss: 0.6209loading clip labels (game 1, clip 6)
loading clip data (game 1, clip 9) downscaled
 65/200 ━━━━━━━━━━━━━━━━━━━━ 31s 230ms/step - io_u: 0.3794 - loss: 0.6206loading clip labels (game 1, clip 9)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3843 - loss: 0.6157loading clip data (game 1, clip 8) downscaled

Epoch 40: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 244ms/step - io_u: 0.3843 - loss: 0.6157 - val_io_u: 0.2301 - val_loss: 0.7699
Epoch 41/50
  6/200 ━━━━━━━━━━━━━━━━━━━━ 44s 227ms/step - io_u: 0.3748 - loss: 0.6252loading clip labels (game 1, clip 8)
  7/200 ━━━━━━━━━━━━━━━━━━━━ 43s 227ms/step - io_u: 0.3752 - loss: 0.6248loading clip data (game 1, clip 5) downscaled
 12/200 ━━━━━━━━━━━━━━━━━━━━ 42s 227ms/step - io_u: 0.3746 - loss: 0.6254loading clip labels (game 1, clip 5)
loading clip data (game 2, clip 3) downscaled
 18/200 ━━━━━━━━━━━━━━━━━━━━ 41s 227ms/step - io_u: 0.3762 - loss: 0.6238loading clip labels (game 2, clip 3)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 227ms/step - io_u: 0.3868 - loss: 0.6132
Epoch 41: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 48s 242ms/step - io_u: 0.3868 - loss: 0.6132 - val_io_u: 0.3886 - val_loss: 0.6114
Epoch 42/50
  8/200 ━━━━━━━━━━━━━━━━━━━━ 43s 227ms/step - io_u: 0.4029 - loss: 0.5971loading clip data (game 1, clip 8) downscaled
  9/200 ━━━━━━━━━━━━━━━━━━━━ 47s 250ms/step - io_u: 0.4026 - loss: 0.5975loading clip labels (game 1, clip 8)
loading clip data (game 5, clip 8) downscaled
 10/200 ━━━━━━━━━━━━━━━━━━━━ 47s 247ms/step - io_u: 0.4007 - loss: 0.5994loading clip labels (game 5, clip 8)
loading clip data (game 4, clip 5) downscaled
 12/200 ━━━━━━━━━━━━━━━━━━━━ 45s 244ms/step - io_u: 0.3978 - loss: 0.6022loading clip labels (game 4, clip 5)
loading clip data (game 4, clip 8) downscaled
 14/200 ━━━━━━━━━━━━━━━━━━━━ 44s 241ms/step - io_u: 0.3962 - loss: 0.6039loading clip labels (game 4, clip 8)
 15/200 ━━━━━━━━━━━━━━━━━━━━ 44s 240ms/step - io_u: 0.3957 - loss: 0.6043loading clip data (game 1, clip 1) downscaled
 18/200 ━━━━━━━━━━━━━━━━━━━━ 43s 238ms/step - io_u: 0.3952 - loss: 0.6048loading clip labels (game 1, clip 1)
loading clip data (game 4, clip 12) downscaled
 21/200 ━━━━━━━━━━━━━━━━━━━━ 42s 237ms/step - io_u: 0.3947 - loss: 0.6053loading clip labels (game 4, clip 12)
 22/200 ━━━━━━━━━━━━━━━━━━━━ 42s 236ms/step - io_u: 0.3946 - loss: 0.6054loading clip data (game 3, clip 5) downscaled
 26/200 ━━━━━━━━━━━━━━━━━━━━ 40s 235ms/step - io_u: 0.3939 - loss: 0.6061loading clip labels (game 3, clip 5)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3853 - loss: 0.6148
Epoch 42: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.3853 - loss: 0.6148 - val_io_u: 0.2616 - val_loss: 0.7384
Epoch 43/50
172/200 ━━━━━━━━━━━━━━━━━━━━ 6s 227ms/step - io_u: 0.4002 - loss: 0.5998loading clip data (game 2, clip 8) downscaled
173/200 ━━━━━━━━━━━━━━━━━━━━ 6s 228ms/step - io_u: 0.4002 - loss: 0.5999loading clip labels (game 2, clip 8)
loading clip data (game 3, clip 3) downscaled
176/200 ━━━━━━━━━━━━━━━━━━━━ 5s 228ms/step - io_u: 0.4001 - loss: 0.5999loading clip labels (game 3, clip 3)
loading clip data (game 4, clip 6) downscaled
180/200 ━━━━━━━━━━━━━━━━━━━━ 4s 228ms/step - io_u: 0.4001 - loss: 0.5999loading clip labels (game 4, clip 6)
loading clip data (game 1, clip 11) downscaled
182/200 ━━━━━━━━━━━━━━━━━━━━ 4s 228ms/step - io_u: 0.4001 - loss: 0.6000loading clip labels (game 1, clip 11)
183/200 ━━━━━━━━━━━━━━━━━━━━ 3s 228ms/step - io_u: 0.4000 - loss: 0.6000loading clip data (game 4, clip 7) downscaled
186/200 ━━━━━━━━━━━━━━━━━━━━ 3s 228ms/step - io_u: 0.4000 - loss: 0.6000loading clip labels (game 4, clip 7)
loading clip data (game 2, clip 3) downscaled
193/200 ━━━━━━━━━━━━━━━━━━━━ 1s 228ms/step - io_u: 0.3999 - loss: 0.6001loading clip labels (game 2, clip 3)
194/200 ━━━━━━━━━━━━━━━━━━━━ 1s 228ms/step - io_u: 0.3999 - loss: 0.6001loading clip data (game 6, clip 8) downscaled
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3998 - loss: 0.6002loading clip labels (game 6, clip 8)

Epoch 43: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.3998 - loss: 0.6002 - val_io_u: 0.3103 - val_loss: 0.6898
Epoch 44/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 227ms/step - io_u: 0.3837 - loss: 0.6163
Epoch 44: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 48s 242ms/step - io_u: 0.3838 - loss: 0.6162 - val_io_u: 0.2979 - val_loss: 0.7021
Epoch 45/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 227ms/step - io_u: 0.3824 - loss: 0.6177
Epoch 45: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 48s 242ms/step - io_u: 0.3824 - loss: 0.6176 - val_io_u: 0.3687 - val_loss: 0.6313
Epoch 46/50
 39/200 ━━━━━━━━━━━━━━━━━━━━ 36s 227ms/step - io_u: 0.4192 - loss: 0.5808loading clip data (game 4, clip 6) downscaled
 43/200 ━━━━━━━━━━━━━━━━━━━━ 36s 231ms/step - io_u: 0.4176 - loss: 0.5824loading clip labels (game 4, clip 6)
 44/200 ━━━━━━━━━━━━━━━━━━━━ 36s 231ms/step - io_u: 0.4173 - loss: 0.5827loading clip data (game 4, clip 7) downscaled
 47/200 ━━━━━━━━━━━━━━━━━━━━ 35s 231ms/step - io_u: 0.4163 - loss: 0.5837loading clip labels (game 4, clip 7)
loading clip data (game 5, clip 5) downscaled
 51/200 ━━━━━━━━━━━━━━━━━━━━ 34s 230ms/step - io_u: 0.4151 - loss: 0.5849loading clip labels (game 5, clip 5)
loading clip data (game 3, clip 1) downscaled
 65/200 ━━━━━━━━━━━━━━━━━━━━ 30s 230ms/step - io_u: 0.4119 - loss: 0.5881loading clip labels (game 3, clip 1)
 67/200 ━━━━━━━━━━━━━━━━━━━━ 30s 230ms/step - io_u: 0.4115 - loss: 0.5885loading clip data (game 4, clip 9) downscaled
 70/200 ━━━━━━━━━━━━━━━━━━━━ 29s 229ms/step - io_u: 0.4109 - loss: 0.5891loading clip labels (game 4, clip 9)
 71/200 ━━━━━━━━━━━━━━━━━━━━ 29s 229ms/step - io_u: 0.4108 - loss: 0.5892loading clip data (game 1, clip 11) downscaled
 73/200 ━━━━━━━━━━━━━━━━━━━━ 29s 229ms/step - io_u: 0.4105 - loss: 0.5895loading clip labels (game 1, clip 11)
loading clip data (game 3, clip 4) downscaled
 75/200 ━━━━━━━━━━━━━━━━━━━━ 28s 229ms/step - io_u: 0.4102 - loss: 0.5898loading clip labels (game 3, clip 4)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.4035 - loss: 0.5965
Epoch 46: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.4035 - loss: 0.5966 - val_io_u: 0.3127 - val_loss: 0.6873
Epoch 47/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3909 - loss: 0.6091loading clip data (game 2, clip 4) downscaled
loading clip labels (game 2, clip 4)
loading clip data (game 1, clip 1) downscaled

Epoch 47: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 245ms/step - io_u: 0.3909 - loss: 0.6091 - val_io_u: 0.2916 - val_loss: 0.7084
Epoch 48/50
  5/200 ━━━━━━━━━━━━━━━━━━━━ 44s 229ms/step - io_u: 0.3041 - loss: 0.6959loading clip labels (game 1, clip 1)
  7/200 ━━━━━━━━━━━━━━━━━━━━ 44s 229ms/step - io_u: 0.3192 - loss: 0.6808loading clip data (game 2, clip 5) downscaled
```

```
 12/200 ━━━━━━━━━━━━━━━━━━━━ 43s 229ms/step - io_u: 0.3400 - loss: 0.6600loading clip labels (game 2, clip 5)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3821 - loss: 0.6179
Epoch 48: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.3821 - loss: 0.6179 - val_io_u: 0.3250 - val_loss: 0.6751
Epoch 49/50
 68/200 ━━━━━━━━━━━━━━━━━━━━ 30s 229ms/step - io_u: 0.3753 - loss: 0.6247loading clip data (game 3, clip 2) downscaled
 79/200 ━━━━━━━━━━━━━━━━━━━━ 27s 231ms/step - io_u: 0.3771 - loss: 0.6229loading clip labels (game 3, clip 2)
 80/200 ━━━━━━━━━━━━━━━━━━━━ 27s 231ms/step - io_u: 0.3773 - loss: 0.6228loading clip data (game 4, clip 15) downscaled
 83/200 ━━━━━━━━━━━━━━━━━━━━ 27s 231ms/step - io_u: 0.3776 - loss: 0.6225loading clip labels (game 4, clip 15)
loading clip data (game 6, clip 9) downscaled
 88/200 ━━━━━━━━━━━━━━━━━━━━ 25s 231ms/step - io_u: 0.3778 - loss: 0.6222loading clip labels (game 6, clip 9)
loading clip data (game 6, clip 4) downscaled
 91/200 ━━━━━━━━━━━━━━━━━━━━ 25s 231ms/step - io_u: 0.3779 - loss: 0.6222loading clip labels (game 6, clip 4)
loading clip data (game 2, clip 5) downscaled
 94/200 ━━━━━━━━━━━━━━━━━━━━ 24s 231ms/step - io_u: 0.3779 - loss: 0.6221loading clip labels (game 2, clip 5)
loading clip data (game 4, clip 5) downscaled
 96/200 ━━━━━━━━━━━━━━━━━━━━ 23s 231ms/step - io_u: 0.3780 - loss: 0.6220loading clip labels (game 4, clip 5)
loading clip data (game 1, clip 8) downscaled
 97/200 ━━━━━━━━━━━━━━━━━━━━ 23s 231ms/step - io_u: 0.3781 - loss: 0.6220loading clip labels (game 1, clip 8)
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 230ms/step - io_u: 0.3824 - loss: 0.6176
Epoch 49: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 245ms/step - io_u: 0.3825 - loss: 0.6176 - val_io_u: 0.2238 - val_loss: 0.7763
Epoch 50/50
200/200 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - io_u: 0.3781 - loss: 0.6219
Epoch 50: val_loss did not improve from 0.60369
200/200 ━━━━━━━━━━━━━━━━━━━━ 49s 243ms/step - io_u: 0.3781 - loss: 0.6219 - val_io_u: 0.2575 - val_loss: 0.7425
training done.
```



Пример пайплайна для тестирования обученной модели:

```
batch_s = 4
stack_s = 5
downscale = True
output_path = prepare_experiment(Path('/kaggle/working'))
new_model = SuperTrackingModel(batch_s, stack_s, out_path=output_path, downscale=downscale)
new_model.load()
sibatracc_final = new_model.test(Path('../input/test/'), [1,2], do_visualization=True, test_name='test')
print(f'SiBaTrAcc final value: {sibatracc_final}')
```

```
Running stub for loading model ...
Downloading...
From: https://drive.google.com/uc?id=1DlgGHnHu7KnKLOFKcEX3JlkECNbyervh
To: /kaggle/working/model111.weights.h5
100%| | 5.59M/5.59M [00:00<00:00, 254MB/s]
Loading model done.
loading clip data (game 1, clip 1) downscaled
loading clip data (game 1, clip 1)
loading clip labels (game 1, clip 1)
doing predictions
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 429ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
```