

## Практическое задание №2

### Общая терминология по используемым данным

Предоставляемые данные для разработки моделей и алгоритмов трекинга мяча в теннисе представляют собой 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/tennistackingassignment>. После загрузки данные датасета будут примонтированы в ../input/tennistackingassignment.

#### 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
/opt/conda/lib/python3.7/site-packages (1.0.3)
Requirement already satisfied: requests<3.0,>=2.8.1 in
/opt/conda/lib/python3.7/site-packages (from moviepy) (2.28.1)
Requirement already satisfied: numpy>=1.17.3 in
/opt/conda/lib/python3.7/site-packages (from moviepy) (1.21.6)
Requirement already satisfied: proglog<=1.0.0 in
/opt/conda/lib/python3.7/site-packages (from moviepy) (0.1.10)
Requirement already satisfied: tqdm<5.0,>=4.11.2 in
/opt/conda/lib/python3.7/site-packages (from moviepy) (4.64.0)
Requirement already satisfied: decorator<5.0,>=4.0.2 in
/opt/conda/lib/python3.7/site-packages (from moviepy) (4.4.2)
```

Requirement already satisfied: imageio-ffmpeg>=0.2.0 in  
/opt/conda/lib/python3.7/site-packages (from moviepy) (0.4.7)  
Requirement already satisfied: imageio<3.0,>=2.5 in  
/opt/conda/lib/python3.7/site-packages (from moviepy) (2.19.3)  
Requirement already satisfied: pillow>=8.3.2 in  
/opt/conda/lib/python3.7/site-packages (from imageio<3.0,>=2.5-  
>moviepy) (9.1.1)  
Requirement already satisfied: charset-normalizer<3,>=2 in  
/opt/conda/lib/python3.7/site-packages (from requests<3.0,>=2.8.1-  
>moviepy) (2.1.0)  
Requirement already satisfied: idna<4,>=2.5 in  
/opt/conda/lib/python3.7/site-packages (from requests<3.0,>=2.8.1-  
>moviepy) (3.3)  
Requirement already satisfied: urllib3<1.27,>=1.21.1 in  
/opt/conda/lib/python3.7/site-packages (from requests<3.0,>=2.8.1-  
>moviepy) (1.26.12)  
Requirement already satisfied: certifi>=2017.4.17 in  
/opt/conda/lib/python3.7/site-packages (from requests<3.0,>=2.8.1-  
>moviepy) (2022.9.24)  
WARNING: Running pip as the 'root' user can result in broken  
permissions and conflicting behaviour with the system package manager.  
It is recommended to use a virtual environment instead:  
<https://pip.pypa.io/warnings/venv>  
Requirement already satisfied: gdown in /opt/conda/lib/python3.7/site-  
packages (4.6.0)  
Requirement already satisfied: filelock in  
/opt/conda/lib/python3.7/site-packages (from gdown) (3.7.1)  
Requirement already satisfied: requests[socks] in  
/opt/conda/lib/python3.7/site-packages (from gdown) (2.28.1)  
Requirement already satisfied: beautifulsoup4 in  
/opt/conda/lib/python3.7/site-packages (from gdown) (4.11.1)  
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-  
packages (from gdown) (4.64.0)  
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-  
packages (from gdown) (1.15.0)  
Requirement already satisfied: soupsieve>1.2 in  
/opt/conda/lib/python3.7/site-packages (from beautifulsoup4->gdown)  
(2.3.1)  
Requirement already satisfied: charset-normalizer<3,>=2 in  
/opt/conda/lib/python3.7/site-packages (from requests[socks]->gdown)  
(2.1.0)  
Requirement already satisfied: certifi>=2017.4.17 in  
/opt/conda/lib/python3.7/site-packages (from requests[socks]->gdown)  
(2022.9.24)  
Requirement already satisfied: urllib3<1.27,>=1.21.1 in  
/opt/conda/lib/python3.7/site-packages (from requests[socks]->gdown)  
(1.26.12)  
Requirement already satisfied: idna<4,>=2.5 in  
/opt/conda/lib/python3.7/site-packages (from requests[socks]->gdown)  
(3.3)

Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in  
/opt/conda/lib/python3.7/site-packages (from requests[socks]->gdown)  
(1.7.1)

WARNING: Running pip as the 'root' user can result in broken  
permissions and conflicting behaviour with the system package manager.  
It is recommended to use a virtual environment instead:

<https://pip.pypa.io/warnings/venv>

После установки пакетов для корректной работы надо обязательно  
перезагрузить ядро. Run -> Restart and clear cell outputs. Без сего действия 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
import numpy as np
import tensorflow as tf
import gdown
from skimage import data, color
from skimage.transform import hough_circle, hough_circle_peaks
from skimage.feature import canny
from skimage.draw import circle_perimeter
from skimage.util import img_as_ubyte
from keras.models import Model
from keras.layers import Input, concatenate, Dense, Conv2DTranspose,
GlobalAveragePooling2D, Dropout, UpSampling2D, Concatenate, Conv2D,
Activation, MaxPooling2D, BatchNormalization
from IPython.display import clear_output
```

## 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_experiment` 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 внутри 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) -> None:
        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, ...]
        mask[mask > 0.1] = 1
        mask[mask <= 0.1] = 0
        return(tf.cast(frames_stack, tf.float32) / 255.0, 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/tennistackingassignment/train/'), [1, 2,
3, 4], stack_s=stack_s, downscale=True, pool_s=10, pool_update_s=4,
quiet=False)
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 4, clip 2) downscaled
loading clip labels (game 4, clip 2)
loading clip data (game 3, clip 5) downscaled
loading clip labels (game 3, clip 5)
loading clip data (game 2, clip 6) downscaled
loading clip labels (game 2, clip 6)
loading clip data (game 4, clip 7) downscaled
loading clip labels (game 4, clip 7)
loading clip data (game 3, clip 1) downscaled
loading clip labels (game 3, clip 1)
loading clip data (game 3, clip 2) downscaled
loading clip labels (game 3, clip 2)
loading clip data (game 1, clip 4) downscaled
loading clip labels (game 1, clip 4)
loading clip data (game 1, clip 11) downscaled

```

loading clip labels (game 1, clip 11)  
loading clip data (game 1, clip 1) downscaled  
loading clip labels (game 1, clip 1)  
loading clip data (game 1, clip 13) downscaled  
loading clip labels (game 1, clip 13)

2022-12-30 22:49:36.538552: I  
tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful  
NUMA node read from SysFS had negative value (-1), but there must be  
at least one NUMA node, so returning NUMA node zero  
2022-12-30 22:49:36.644700: I  
tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful  
NUMA node read from SysFS had negative value (-1), but there must be  
at least one NUMA node, so returning NUMA node zero  
2022-12-30 22:49:36.645578: I  
tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful  
NUMA node read from SysFS had negative value (-1), but there must be  
at least one NUMA node, so returning NUMA node zero  
2022-12-30 22:49:36.648631: I  
tensorflow/core/platform/cpu\_feature\_guard.cc:142] This TensorFlow  
binary is optimized with oneAPI Deep Neural Network Library (oneDNN)  
to use the following CPU instructions in performance-critical  
operations: AVX2 AVX512F FMA  
To enable them in other operations, rebuild TensorFlow with the  
appropriate compiler flags.  
2022-12-30 22:49:36.649033: I  
tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful  
NUMA node read from SysFS had negative value (-1), but there must be  
at least one NUMA node, so returning NUMA node zero  
2022-12-30 22:49:36.650007: I  
tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful  
NUMA node read from SysFS had negative value (-1), but there must be  
at least one NUMA node, so returning NUMA node zero  
2022-12-30 22:49:36.650784: I  
tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful  
NUMA node read from SysFS had negative value (-1), but there must be  
at least one NUMA node, so returning NUMA node zero  
2022-12-30 22:49:38.701917: I  
tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful  
NUMA node read from SysFS had negative value (-1), but there must be  
at least one NUMA node, so returning NUMA node zero  
2022-12-30 22:49:38.702785: I  
tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful  
NUMA node read from SysFS had negative value (-1), but there must be  
at least one NUMA node, so returning NUMA node zero  
2022-12-30 22:49:38.703504: I  
tensorflow/stream\_executor/cuda/cuda\_gpu\_executor.cc:937] successful  
NUMA node read from SysFS had negative value (-1), but there must be  
at least one NUMA node, so returning NUMA node zero  
2022-12-30 22:49:38.704121: I

```
tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Created
device /job:localhost/replica:0/task:0/device:GPU:0 with 15401 MB
memory: -> device: 0, name: Tesla P100-PCIE-16GB, pci bus id:
0000:00:04.0, compute capability: 6.0
```

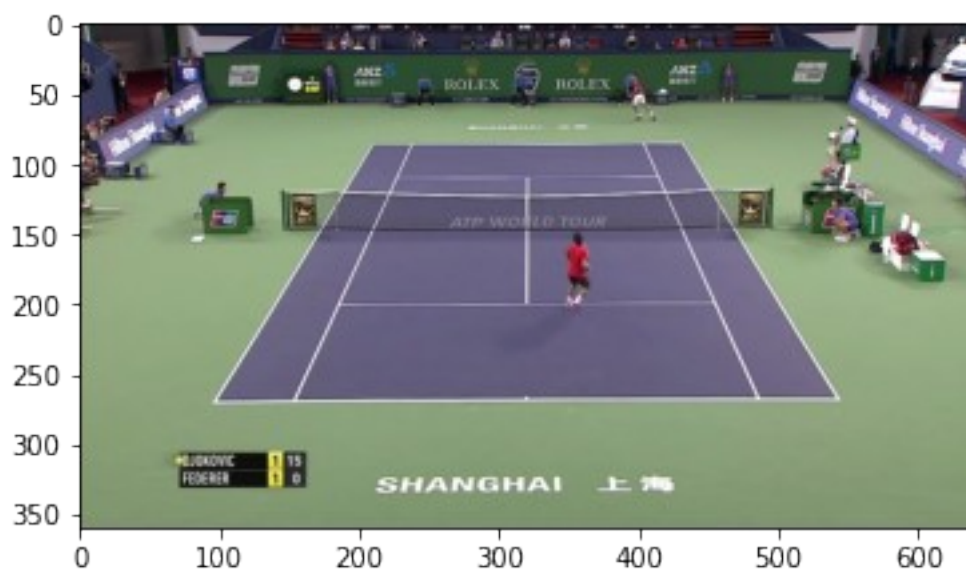
```
(4, 360, 640, 9) float32 (4, 360, 640) float32
(4, 360, 640, 9) float32 (4, 360, 640) float32
(4, 360, 640, 9) float32 (4, 360, 640) float32
(4, 360, 640, 9) float32 (4, 360, 640) float32
(4, 360, 640, 9) float32 (4, 360, 640) float32
(4, 360, 640, 9) float32 (4, 360, 640) float32
(4, 360, 640, 9) float32 (4, 360, 640) float32
(4, 360, 640, 9) float32 (4, 360, 640) float32
(4, 360, 640, 9) float32 (4, 360, 640) float32
(4, 360, 640, 9) float32 (4, 360, 640) float32
```

```
import matplotlib.pyplot as plt
```

```
stack_s = 3
train_gen =
DataGenerator(Path('../input/tennistackingassignment/train/'), [1],
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])
```

```
loading clip data (game 1, clip 2) downscaled
loading clip labels (game 1, clip 2)
loading clip data (game 1, clip 5) downscaled
loading clip labels (game 1, clip 5)
loading clip data (game 1, clip 13) downscaled
loading clip labels (game 1, clip 13)
loading clip data (game 1, clip 7) downscaled
loading clip labels (game 1, clip 7)
loading clip data (game 1, clip 1) downscaled
loading clip labels (game 1, clip 1)
loading clip data (game 1, clip 8) downscaled
loading clip labels (game 1, clip 8)
loading clip data (game 1, clip 12) downscaled
loading clip labels (game 1, clip 12)
loading clip data (game 1, clip 11) downscaled
loading clip labels (game 1, clip 11)
loading clip data (game 1, clip 10) downscaled
loading clip labels (game 1, clip 10)
loading clip data (game 1, clip 4) downscaled
loading clip labels (game 1, clip 4)
(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 аккумулярованных 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
    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 маске. 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!

```

def marinet(n_classes, input_height, input_width):
    input_layer = Input(shape=(input_height, input_width, 9))

    layer1 = Conv2D(64, (3, 3), padding='same')(input_layer)
    layer1 = BatchNormalization()(layer1)
    layer1 = Activation('relu')(layer1)

    layer2 = Conv2D(64, (3, 3), padding='same')(layer1)
    layer2 = BatchNormalization()(layer2)
    layer2 = Activation('relu')(layer2)
    layer2 = MaxPooling2D((2, 2), strides=(2, 2))(layer2)

    layer3 = Conv2D(128, (3, 3), padding='same')(layer2)
    layer3 = BatchNormalization()(layer3)
    layer3 = Activation('relu')(layer3)

```

```
layer4 = Conv2D(128, (3, 3), padding='same')(layer3)
layer4 = BatchNormalization()(layer4)
layer4 = Activation('relu')(layer4)
layer4 = MaxPooling2D((2, 2), strides=(2, 2))(layer4)

layer5 = Conv2D(256, (3, 3), padding='same')(layer4)
layer5 = BatchNormalization()(layer5)
layer5 = Activation('relu')(layer5)

layer6 = Conv2D(256, (3, 3), padding='same')(layer5)
layer6 = BatchNormalization()(layer6)
layer6 = Activation('relu')(layer6)

layer7 = Conv2D(256, (3, 3), padding='same')(layer6)
layer7 = BatchNormalization()(layer7)
layer7 = Activation('relu')(layer7)
layer7 = MaxPooling2D((2, 2), strides=(2, 2))(layer7)

layer8 = Conv2D(512, (3, 3), padding='same')(layer7)
layer8 = BatchNormalization()(layer8)
layer8 = Activation('relu')(layer8)

layer9 = Conv2D(512, (3, 3), padding='same')(layer8)
layer9 = BatchNormalization()(layer9)
layer9 = Activation('relu')(layer9)

layer10 = Conv2D(512, (3, 3), padding='same')(layer9)
layer10 = BatchNormalization()(layer10)
layer10 = Activation('relu')(layer10)
layer10 = UpSampling2D((2, 2))(layer10)

layer11 = Conv2D(256, (3, 3), padding='same')(layer10)
layer11 = BatchNormalization()(layer11)
layer11 = Activation('relu')(layer11)

layer12 = Conv2D(256, (3, 3), padding='same')(layer11)
layer12 = BatchNormalization()(layer12)
layer12 = Activation('relu')(layer12)

layer13 = Conv2D(256, (3, 3), padding='same')(layer12)
layer13 = BatchNormalization()(layer13)
layer13 = Activation('relu')(layer13)
layer13 = UpSampling2D((2, 2))(layer13)

layer14 = Conv2D(128, (3, 3), padding='same')(layer13)
layer14 = BatchNormalization()(layer14)
layer14 = Activation('relu')(layer14)

layer15 = Conv2D(128, (3, 3), padding='same')(layer14)
```

```

layer15 = BatchNormalization()(layer15)
layer15 = Activation('relu')(layer15)
layer15 = UpSampling2D((2,2))(layer15)

layer16 = Conv2D(64 , (3, 3), padding='same')(layer15)
layer16 = BatchNormalization()(layer16)
layer16 = Activation('relu')(layer16)

layer17 = Conv2D(64 , (3, 3), padding='same')(layer16)
layer17 = BatchNormalization()(layer17)
layer17 = Activation('relu')(layer17)

layer18 = Conv2D(n_classes, (3, 3), padding='same')(layer17)
layer18 = Activation('softmax')(layer18)

model = Model(input_layer, layer18)
return model

```

```

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 = marinet(2, 360, 640)

```

```

self.model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
loss=tf.keras.losses.sparse_categorical_crossentropy,
metrics=['accuracy'])

```

```

    def save(self, name: str):
        self.model.save(name)

```

```

    def load(self, name):
        name_to_id_dict = {
            'best': '1UhWS0wF8-tgs0JEEqBecBRLzHwa0pSc9'
        }
        url =
f'https://drive.google.com/drive/folders/{name_to_id_dict[name]}'
        gdown.download_folder(url, quiet=True, output=name,
use_cookies=False)
        self.model = tf.keras.models.load_model(name)

```

```

    def predict_on_batch(self, batch: np.ndarray) -> np.ndarray:
        norm_mask = tf.cast(batch, tf.float32) / 255.0
        mask = self.model.predict(norm_mask)
        result = np.zeros(mask.shape[:3])
        for i in range(mask.shape[0]):
            result[i] = tf.argmax(mask[i], axis=-1)

```

```

    return result

def _predict_prob_on_clip(self, clip: np.ndarray) -> np.ndarray:
    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 beginning ---
    start_frames = np.zeros((self.stack_s - 1,
predictions.shape[1], predictions.shape[2]), dtype=np.float32)
    predictions = np.concatenate((start_frames, predictions),
axis=0)
    return predictions

```

```

def find_circle_center(self, img):
    image = img_as_ubyte(img)
    edges = canny(image, sigma=3, low_threshold=10,
high_threshold=50)
    hough_radii = np.arange(20, 35, 2)
    hough_res = hough_circle(edges, hough_radii)
    accums, cx, cy, radii = hough_circle_peaks(hough_res,
hough_radii,
total_num_peaks=1)

```

```

        if cx.shape == (1,):
            return cx[0] * 2, cy[0] * 2
        else:
            return 0, 0

    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):
            curr_mask = pred_prob[i]
            if len(np.unique(curr_mask)) > 1:
                coords[i, 0] = 1
                x, y = self.find_circle_center(curr_mask)
                if x == 0 or y == 0 and i > 0:
                    coords[i, 1], coords[i, 2] = coords[i-1, 1],
coords[i-1, 2]
            else:
                coords[i, 1], coords[i, 2] = x, y
        return coords

    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)
        return labels_pr, 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, quiet=True) if self.downscale else data
                labels_gt = load_clip_labels(data_path, game, clip,
downscale=False, quiet=True)
                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, param_1=None, param_2=None, param_3=None,
param_4=None, param_5=None, param_6=None):
    print('Running stub for training model...')
    print(self.model)
    self.model.fit_generator(param_1, steps_per_epoch=40,
epochs=25)
    print('training done.')

```

Пример пайплайна для обучения модели:

```

batch_s = 4
stack_s = 3
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/tennistackingassignment/train/'), [1, 2,
3], stack_s=stack_s, downscale=True, pool_s=10, pool_update_s=4,
quiet=False)

model.train(train_gen.random_g(batch_s))

loading clip data (game 2, clip 5) downscaled
loading clip labels (game 2, clip 5)
loading clip data (game 2, clip 1) downscaled
loading clip labels (game 2, clip 1)
loading clip data (game 1, clip 5) downscaled
loading clip labels (game 1, clip 5)
loading clip data (game 3, clip 7) downscaled
loading clip labels (game 3, clip 7)
loading clip data (game 2, clip 2) downscaled
loading clip labels (game 2, clip 2)
loading clip data (game 1, clip 9) downscaled
loading clip labels (game 1, clip 9)
loading clip data (game 1, clip 1) downscaled
loading clip labels (game 1, clip 1)
loading clip data (game 3, clip 6) downscaled
loading clip labels (game 3, clip 6)
loading clip data (game 1, clip 3) downscaled

```

```
loading clip labels (game 1, clip 3)
loading clip data (game 2, clip 6) downscaled
loading clip labels (game 2, clip 6)
Running stub for training model...
<keras.engine.functional.Functional object at 0x7f9b3cb05110>
Epoch 1/25

2022-12-30 22:53:16.291254: I
tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of
the MLIR Optimization Passes are enabled (registered 2)
2022-12-30 22:53:18.712465: I
tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN version
8005

 2/40 [>.....] - ETA: 14s - loss: 1.1908 -
accuracy: 0.2302

2022-12-30 22:53:27.649087: W
tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of
33177600 exceeds 10% of free system memory.

 3/40 [=>.....] - ETA: 16s - loss: 1.1018 -
accuracy: 0.2620

2022-12-30 22:53:28.093015: W
tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of
33177600 exceeds 10% of free system memory.

 4/40 [==>.....] - ETA: 15s - loss: 1.0158 -
accuracy: 0.3269

2022-12-30 22:53:28.546610: W
tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of
33177600 exceeds 10% of free system memory.

 5/40 [==>.....] - ETA: 15s - loss: 0.9401 -
accuracy: 0.4037

2022-12-30 22:53:28.974467: W
tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of
33177600 exceeds 10% of free system memory.

 6/40 [===>.....] - ETA: 14s - loss: 0.8767 -
accuracy: 0.4734

2022-12-30 22:53:29.452980: W
tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of
33177600 exceeds 10% of free system memory.

40/40 [=====] - 28s 431ms/step - loss: 0.2683
- accuracy: 0.9120
Epoch 2/25
40/40 [=====] - 17s 430ms/step - loss: 0.0668
```

```
- accuracy: 0.9938
Epoch 3/25
40/40 [=====] - 17s 423ms/step - loss: 0.0511
- accuracy: 0.9939
Epoch 4/25
40/40 [=====] - 17s 420ms/step - loss: 0.0417
- accuracy: 0.9942
Epoch 5/25
40/40 [=====] - 17s 425ms/step - loss: 0.0377
- accuracy: 0.9942
Epoch 6/25
40/40 [=====] - 17s 422ms/step - loss: 0.0342
- accuracy: 0.9943
Epoch 7/25
40/40 [=====] - 17s 427ms/step - loss: 0.0319
- accuracy: 0.9943
Epoch 8/25
40/40 [=====] - 17s 426ms/step - loss: 0.0320
- accuracy: 0.9939
Epoch 9/25
40/40 [=====] - 17s 437ms/step - loss: 0.0283
- accuracy: 0.9943
Epoch 10/25
40/40 [=====] - 17s 430ms/step - loss: 0.0285
- accuracy: 0.9941
Epoch 11/25
40/40 [=====] - 17s 428ms/step - loss: 0.0247
- accuracy: 0.9941
Epoch 12/25
14/40 [=====>.....] - ETA: 11s - loss: 0.0226 -
accuracy: 0.9950loading clip data (game 3, clip 2) downscaled
loading clip labels (game 3, clip 2)
loading clip data (game 2, clip 7) downscaled
loading clip labels (game 2, clip 7)
loading clip data (game 3, clip 4) downscaled
loading clip labels (game 3, clip 4)
loading clip data (game 1, clip 13) downscaled
loading clip labels (game 1, clip 13)
40/40 [=====] - 24s 602ms/step - loss: 0.0225
- accuracy: 0.9947
Epoch 13/25
40/40 [=====] - 17s 423ms/step - loss: 0.0198
- accuracy: 0.9952
Epoch 14/25
40/40 [=====] - 17s 424ms/step - loss: 0.0181
- accuracy: 0.9955
Epoch 15/25
40/40 [=====] - 17s 426ms/step - loss: 0.0143
- accuracy: 0.9967
Epoch 16/25
```



```

40/40 [=====] - 17s 425ms/step - loss: 0.0154
- accuracy: 0.9964
Epoch 17/25
40/40 [=====] - 17s 423ms/step - loss: 0.0122
- accuracy: 0.9968
Epoch 18/25
40/40 [=====] - 17s 426ms/step - loss: 0.0109
- accuracy: 0.9971
Epoch 19/25
40/40 [=====] - 17s 427ms/step - loss: 0.0119
- accuracy: 0.9967
Epoch 20/25
40/40 [=====] - 17s 429ms/step - loss: 0.0108
- accuracy: 0.9971
Epoch 21/25
40/40 [=====] - 17s 426ms/step - loss: 0.0087
- accuracy: 0.9977
Epoch 22/25
40/40 [=====] - 17s 430ms/step - loss: 0.0083
- accuracy: 0.9977
Epoch 23/25
40/40 [=====] - 17s 429ms/step - loss: 0.0076
- accuracy: 0.9981
Epoch 24/25
40/40 [=====] - 17s 428ms/step - loss: 0.0081
- accuracy: 0.9978
Epoch 25/25
21/40 [=====>.....] - ETA: 8s - loss: 0.0092 -
accuracy: 0.9975loading clip data (game 1, clip 5) downscaled
loading clip labels (game 1, clip 5)
loading clip data (game 1, clip 1) downscaled
loading clip labels (game 1, clip 1)
loading clip data (game 3, clip 3) downscaled
loading clip labels (game 3, clip 3)
loading clip data (game 1, clip 8) downscaled
loading clip labels (game 1, clip 8)
40/40 [=====] - 20s 513ms/step - loss: 0.0080
- accuracy: 0.9979
training done.

```

```
model.save('best')
```

2022-12-30 23:02:27.541160: W tensorflow/python/util/util.cc:348] Sets are not currently considered sequences, but this may change in the future, so consider avoiding using them.

Пример пайплайна для тестирования обученной модели:

```

output_path = prepare_experiment(Path('/kaggle/working'))
new_model = SuperTrackingModel(batch_s, stack_s, out_path=output_path,
downscale=downscale)

```

```

new_model.load('best')
sibatracc_final =
new_model.test(Path('../input/tennistackingassignment/test/'), [2,],
do_visualization=False, test_name='test')
print(f'SiBaTrAcc final value: {sibatracc_final}')

loading clip data (game 2, clip 1) downscaled
loading clip data (game 2, clip 2) downscaled
loading clip data (game 2, clip 3) downscaled
loading clip data (game 2, clip 4) downscaled
loading clip data (game 2, clip 5) downscaled
loading clip data (game 2, clip 6) downscaled
loading clip data (game 2, clip 7) downscaled
loading clip data (game 2, clip 8) downscaled
loading clip data (game 2, clip 9) downscaled
SiBaTrAcc final value: 0.6007064666971883

```

Во время самостоятельного тестирования попробуйте хотя бы раз сделать тестирование с визуализацией (`do_visualization=True`), чтобы визуально оценить качество трекинга разработанной моделью.

Загрузка модели через функцию `load` должна происходить полностью автоматически без каких-либо действий со стороны пользователя! Один из вариантов подобной реализации с использованием `google drive` и пакета `gdown` приведен в разделе с дополнениями.

## Дополнения

Иногда при записи большого количества файлов в `output` директорию `kaggle` может "тупить" и не отображать корректно структуру дерева файлов в `output` и не показывать кнопки для скачивания выбранного файла. В этом случае удобно будет запаковать директорию с экспериментом и выкачать ее вручную. Пример для выкачивания директории с первым экспериментом приведен ниже:

```

%cd /kaggle/working/
!zip -r "exp_1.zip" "exp_1"
from IPython.display import FileLink
FileLink(r'exp_1.zip')

```

удалить лишние директории или файлы в `output` тоже легко:

```
!rm -r /kaggle/working/exp_1
```

Для реализации загрузки данных рекомендуется использовать облачное хранилище `google drive` и пакет `gdown` для скачивания файлов. Пример подобного использования приведен ниже:

1. загружаем файл в `google drive` (в данном случае, это `npz` архив, содержащий один `numpy` массив по ключу `'w'`)

2. в интерфейсе google drive открываем доступ на чтение к файлу по ссылке и извлекаем из ссылки id файла
3. формируем url для скачивания файла
4. с помощью gdown скачиваем файл
5. распаковываем npz архив и пользуемся numpy массивом

Обратите внимание, что для корректной работы нужно правильно определить id файла. В частности, в ссылке [https://drive.google.com/file/d/1kZ8CC-zfkB\\_TlwtBjuPcEfsPV0Jz7IPA/view?usp=sharing](https://drive.google.com/file/d/1kZ8CC-zfkB_TlwtBjuPcEfsPV0Jz7IPA/view?usp=sharing) id файла заключен между ...d/ b /view?... и равен 1kZ8CC-zfkB\_TlwtBjuPcEfsPV0Jz7IPA

```
import gdown
```

```
id = '1kZ8CC-zfkB_TlwtBjuPcEfsPV0Jz7IPA'  
url = f'https://drive.google.com/uc?id={id}'  
output = 'sample-weights.npz'  
gdown.download(url, output, quiet=False)
```

```
import numpy as np
```

```
weights = np.load('/kaggle/working/sample-weights.npz')['w']  
print(weights)
```