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):**

In [1]:

```
!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: decorator<5.0,>=4.0.2 in /opt/conda/lib/python3.7/site-pac
kages (from moviepy) (4.4.2)
Requirement already satisfied: requests<3.0,>=2.8.1 in /opt/conda/lib/python3.7/site-pack
ages (from moviepy) (2.28.1)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/lib/python3.7/site-packages (f
rom moviepy) (1.21.6)
Requirement already satisfied: imageio-ffmpeg>=0.2.0 in /opt/conda/lib/python3.7/site-pac
kages (from moviepy) (0.4.7)
Requirement already satisfied: imageio<3.0,>=2.5 in /opt/conda/lib/python3.7/site-package
s (from moviepy) (2.19.3)
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-package
s (from moviepy) (4.64.0)
Requirement already satisfied: pillow>=8.3.2 in /opt/conda/lib/python3.7/site-packages (f
rom imageio<3.0,>=2.5->moviepy) (9.1.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.7/site-pac
kages (from requests<3.0,>=2.8.1->moviepy) (1.26.12)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.7/site-packages (fr
om requests<3.0,>=2.8.1->moviepy) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/site-packag
es (from requests<3.0,>=2.8.1->moviepy) (2022.9.24)
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)
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
```

```
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: tqdm in /opt/conda/lib/python3.7/site-packages (from gdown
) (4.64.0)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.7/site-packages (
from gdown) (4.11.1)
Requirement already satisfied: filelock in /opt/conda/lib/python3.7/site-packages (from g
down) (3.7.1)
Requirement already satisfied: requests[socks] in /opt/conda/lib/python3.7/site-packages
(from gdown) (2.28.1)
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 (f
rom 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: idna<4,>=2.5 in /opt/conda/lib/python3.7/site-packages (fr
om requests[socks]->gdown) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/site-packag
es (from requests[socks]->gdown) (2022.9.24)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.7/site-pac
kages (from requests[socks]->gdown) (1.26.12)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /opt/conda/lib/python3.7/site-pa
ckages (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 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:

In [2]:

```
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 tensorflow as tf
import matplotlib.pyplot as plt
import cv2
```

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

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**.**

In [3]:

```
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;

In [4]:

```
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 * r
ad + 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 = 48 #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.

In [5]:

```
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), o
utline=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.

In [6]:

```
import gdown
```

In [7]:

```
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 p
ool 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_wei
ghts)[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, ...]
        if stack_s > 1:
            frames_stack = np.squeeze(np.split(frames_stack, indices_or_sections=self.st
ack_s, axis=0))
        #if self.move_axis:
            #frames_stack = np.concatenate(frames_stack, axis=0)
        #else:
        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)
            #new_masks = np.zeros((masks_batch.shape[0], 384, 640))
            #new_masks[:, 12:372, :] = masks_batch
            masks_batch[masks_batch > 0.2] = 1
            masks_batch[masks_batch <= 0.2] = 0
            yield imgs_batch, masks_batch.reshape((masks_batch.shape[0], -1, 1))
```

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**.**

In [ ]:

```
stack_s = 3
batch_s = 4
train_gen = DataGenerator(Path('../input/tennistrackingassignment/train/'), [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)
```

In [ ]:

```
stack_s = 3
train_gen = DataGenerator(Path('../input/tennistrackingassignment/train/'), [1], stack_s=
stack_s, downscale=True, pool_s=10, pool_update_s=4, quiet=False)
stack, mask = train_gen.get_random_stack(12)
print(stack.shape, mask.shape)
mask[mask < 0.4] = 0
mask[mask >= 0.4] = 1
```

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

In [8]:

```
import tensorflow as tf
import tensorflow.keras.backend as K
import numpy as np
# weighted loss functions


def weighted_binary_cross_entropy(weights: dict, from_logits: bool = False):

    def weighted_cross_entropy_fn(y_true, y_pred):
        tf_y_true = tf.cast(y_true, dtype=tf.float32)
        tf_y_pred = tf.cast(y_pred, dtype=tf.float32)

        weights_v = tf.where(tf.equal(tf_y_true, 1), weights[1], weights[0])
        weights_v = tf.cast(weights_v, dtype=tf.float32)
        ce = K.binary_crossentropy(tf_y_true, tf_y_pred, from_logits=from_logits)
        loss = K.mean(tf.multiply(ce, weights_v))
        return loss

    return weighted_cross_entropy_fn
```

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.**

In [9]:

```
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 rang
e(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
```

Класс **CustomDatagen**

Класс **CustomDatagen** нужен для корректной подачи даннных в нейронную сеть при обучении классификатора. К сожалению, из этой затеи мало что вышло.

In [10]:

```
from sklearn.utils import shuffle
```

In [11]:

```
class CustomDatagen(tf.keras.utils.Sequence):

    def __init__(self, generator, length, batch_size):
        self.generator = generator
        self.length = length
        self.batch_size = batch_size
        self.window_size = 80

    def __len__(self):
        return self.length

    def __call__(self):
        for i in range(self.__len__()):
            im, l = self.__getitem__()
            images, labels =  self.create_image_batches(im, l)
            for j in range(images.shape[0]):
                yield images[j], labels[j]

    def create_image_batches(self, image, labels):
        shape = np.array([720, 1280])
        if self.generator.downscale:
            shape //= 2
        data = np.empty((50, self.window_size, self.window_size, 3))
        images_with_balls = np.empty((image.shape[0], shape[0], shape[1], 3))
        labels_balls = []
        cnt = 0
        for i in range(15):
            if labels[i][0] == 1 or labels[i][0] == 2:
                images_with_balls[cnt, :, :, :] = image[i, :, :, :]
                labels_balls.append(labels[i])
                cnt += 1
        bin_labels = np.empty((50,))
        for i in range(cnt):
            y = labels_balls[i][2]
            x = labels_balls[i][1]
            k = 0
            if 0 <= shape[0] - y <= self.window_size // 2:
                y -= (self.window_size // 2 - (shape[0] - y))
                k = 1
            elif y <= self.window_size // 2:
                y += (self.window_size // 2 - y)
                k = 1
            if 0 <= shape[1] - x <= self.window_size // 2:
                x -= (self.window_size // 2 - (shape[1] - x))
                k = 1
            elif x <= self.window_size // 2:
                x += (self.window_size // 2 - x)
                k = 1
            if k == 0:
                if self.window_size <= x <= shape[1] - self.window_size:
                    x += np.random.randint(-int(self.window_size / 2.3), int(self.window
_size / 2.3))
                if self.window_size <= y <= shape[0] - self.window_size:
                    y += np.random.randint(-int(self.window_size / 2.3), int(self.window
_size / 2.3))
            window = images_with_balls[i, y - self.window_size // 2:y  + self.window_siz
e // 2, x - self.window_size // 2:x + self.window_size // 2, :]
            data[2 * i] = window
            bin_labels[2 * i] = 1
            bin_labels[2 * i + 1] = 1
            data[2 * i + 1] = cv2.flip(window, 1)
```

```
        for i in range(2 * cnt, 50):
            image_num = np.random.randint(0, 15)
            y = np.random.randint(0, shape[0] - self.window_size, size=None)
            x = np.random.randint(0, shape[1] - self.window_size, size=None)
            data[i] = image[image_num, y:y + self.window_size, x:x + self.window_size, :
]
            if 1 <= labels[image_num][0] <= 2 and x + 5 <= labels[image_num][1] <= x +
self.window_size - 5 and y + 5 <= labels[image_num][2] <= y + self.window_size - 5:
                bin_labels[i] = 1
            else:
                bin_labels[i] = 0
        data, bin_labels = shuffle(data, bin_labels)
        return data, bin_labels

    def __getitem__(self):
        image, _, labels = self.generator.get_random_batch(15)
        shape = np.array([720, 1280])
        if self.generator.downscale:
            shape //= 2
        image = image.reshape(15, shape[1], shape[0],  3)
        image = image.astype('float32')
        image = image / 255.0
        image1 = np.zeros((15, shape[0], shape[1],  3))
        for i in range(15):
            for j in range(3):
                image1[i, :, :, j] = image[i, :, :, j].T
        return image1, labels
```

Класс **CustomDatagenForUnet**

Класс **CustomDatagenForUnet** нужен для корректной подачи данных при обучении сегментирующей нейронной сети. К сожалению, из этой затеи мало что вышло.

In [12]:

```
class CustomDatagenForUnet(tf.keras.utils.Sequence):

    def __init__(self, generator, length, batch_size):
        self.generator = generator
        self.length = length
        self.batch_size = batch_size
        self.window_size = 80

    def __len__(self):
        return self.length

    def __call__(self):
        for i in range(self.__len__()):
            im, m, l = self.__getitem__()
            images, masks =  self.create_image_batches(im, m, l)
            for j in range(images.shape[0]):
                yield images[j], masks[j]

    def create_image_batches(self, image, masks, labels):
        shape = np.array([720, 1280])
        if self.generator.downscale:
            shape //= 2
        images_with_balls = np.empty((image.shape[0], shape[0], shape[1], 3))
        labels_balls = []
        cnt = 0
        for i in range(15):
            if labels[i][0] == 1 or labels[i][0] == 2:
                images_with_balls[cnt, :, :, :] = image[i, :, :, :]
                labels_balls.append(labels[i])
                cnt += 1
        new_masks = np.empty((2 * cnt, self.window_size, self.window_size, 1))
        data = np.empty((2 * cnt, self.window_size, self.window_size, 3))
        for i in range(cnt):
```

```
            y = labels_balls[i][2]
            x = labels_balls[i][1]
            k = 0
            if 0 <= shape[0] - y <= self.window_size // 2:
                y -= (self.window_size // 2 - (shape[0] - y))
                k = 1
            elif y <= self.window_size // 2:
                y += (self.window_size // 2 - y)
                k = 1
            if 0 <= shape[1] - x <= self.window_size // 2:
                x -= (self.window_size // 2 - (shape[1] - x))
                k = 1
            elif x <= self.window_size // 2:
                x += (self.window_size // 2 - x)
                k = 1
            if k == 0:
                if self.window_size <= x <= shape[1] - self.window_size:
                    x += np.random.randint(-int(self.window_size / 2.3), int(self.window
_size / 2.3))
                if self.window_size <= y <= shape[0] - self.window_size:
                    y += np.random.randint(-int(self.window_size / 2.3), int(self.window
_size / 2.3))
            window = images_with_balls[i, y - self.window_size // 2:y  + self.window_siz
e // 2, x - self.window_size // 2:x + self.window_size // 2, :]
            data[2 * i] = window
            mask = masks[i, y - self.window_size // 2:y  + self.window_size // 2, x - se
lf.window_size // 2:x + self.window_size // 2].reshape(self.window_size, self.window_siz
e, 1)
            new_masks[2 * i] = mask
            new_masks[2 * i + 1] = cv2.flip(mask.reshape(self.window_size, self.window_s
ize), 1).reshape(self.window_size, self.window_size, 1)
            data[2 * i + 1] = cv2.flip(window, 1)

        data, new_masks = shuffle(data, new_masks)
        return data, new_masks

    def __getitem__(self, i):
        image, masks, labels = self.generator.get_random_batch(15)
        shape = np.array([720, 1280])
        if self.generator.downscale:
            shape //= 2
        image = image.reshape(15, shape[1], shape[0],  3)
        image = image.astype('float32')
        image = image / 255.0
        image1 = np.zeros((15, shape[0], shape[1],  3))
        for i in range(15):
            for j in range(3):
                image1[i, :, :, j] = image[i, :, :, j].T
        masks[masks < 0.6] = 0
        masks[masks >= 0.6] = 1
        return image1, masks, labels
```

Класс **Datagen1**

Еще один самописный генератор, который использовался при тестировании и который оказался ненужным. Решил не удалять.

In [13]:

```
class Datagen1(tf.keras.utils.Sequence):

    def __init__(self, generator, length, batch_size):
        self.generator = generator
        self.length = length
        self.batch_size = batch_size

    def __len__(self):
        return self.length
```

```
    def __call__(self):
        for i in range(self.__len__()):
            yield self.__getitem__()

    def __getitem__(self):
        image, labels = self.generator.get_random_stack()
        shape = np.array([720, 1280])
        if self.generator.downscale:
            shape //= 2
        image = image.astype('float32')
        image = image / 255.0
        labels[labels > 0.2] = 1
        labels[labels <= 0.2] = 0
        return image, labels.reshape((-1, 1))
```

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_batch** 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!

In [14]:

```
import cv2
from tensorflow.keras.layers import *
from keras.models import *
```

In [15]:

```
#нужно было для одной из моделей
!pip install -q git+https://github.com/tensorflow/examples.git
```

```
  WARNING: Built wheel for tensorflow-examples is invalid: Metadata 1.2 mandates PEP 440
version, but '6ae97eaf3dbd607ed3eccf18f7dc05d7a3b677e3-' is not
  DEPRECATION: tensorflow-examples was installed using the legacy 'setup.py install' meth
od, because a wheel could not be built for it. A possible replacement is to fix the wheel
build issue reported above. Discussion can be found at https://github.com/pypa/pip/issues
/8368
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
```

In [16]:

```
import tensorflow_addons as tfa
from tensorflow_examples.models.pix2pix import pix2pix
```

In [17]:

```
up_stack = [
    pix2pix.upsample(512, 3),  # 4x4 -> 8x8
    pix2pix.upsample(256, 3),  # 8x8 -> 16x16
    pix2pix.upsample(128, 3),  # 16x16 -> 32x32
    pix2pix.upsample(64, 3),   # 32x32 -> 64x64
```

```
]
```

```
2022-12-29 22:04:01.431226: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2022-12-29 22:04:01.525687: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2022-12-29 22:04:01.526495: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2022-12-29 22:04:01.528248: I tensorflow/core/platform/cpu_feature_guard.cc:142] This Ten
sorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the f
ollowing CPU instructions in performance-critical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flag
s.
2022-12-29 22:04:01.528601: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2022-12-29 22:04:01.529323: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2022-12-29 22:04:01.529989: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2022-12-29 22:04:03.835398: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2022-12-29 22:04:03.836273: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2022-12-29 22:04:03.836960: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2022-12-29 22:04:03.837590: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Crea
ted 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
```

In [18]:

```
#Одна из моделей, которую я тестировал. Нужной точности не показала.
def Model2(input_size):
    base_model = tf.keras.applications.MobileNetV2(input_shape=input_size, include_top=F
alse)

    # Use the activations of these layers
    layer_names = [
    'block_1_expand_relu',   # 64x64
    'block_3_expand_relu',   # 32x32
    'block_6_expand_relu',   # 16x16
    'block_13_expand_relu',  # 8x8
    'block_16_project',      # 4x4
    ]
    base_model_outputs = [base_model.get_layer(name).output for name in layer_names]

    # Create the feature extraction model
    down_stack = tf.keras.Model(inputs=base_model.input, outputs=base_model_outputs)

    down_stack.trainable = False
    inputs = tf.keras.layers.Input(shape=input_size)

   # Downsampling through the model
    skips = down_stack(inputs)
    x = skips[-1]

   # Upsampling and establishing the skip connections
    for up in up_stack:
        x = up(x)

   # This is the last layer of the model
    last = tf.keras.layers.Conv2DTranspose(filters=1, kernel_size=3, strides=2, padding=
'same')  #64x64 -> 128x128
```

```
    x = last(x)
    x = (Reshape(( -1, 1)))(x)

    return tf.keras.Model(inputs=inputs, outputs=x)
```

In [19]:

```
#Одна из моделей, которую я тестировал. Нужной точности не показала (или у кого то руки н
е из того места растут...).
def UNet(input_size):

    #Build argminthe model
    inputs = tf.keras.layers.Input(input_size)

    #Contraction path

    c2 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_no
rmal', padding='same')(inputs)
    c2 = tf.keras.layers.Dropout(0.1)(c2)
    c2 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_no
rmal', padding='same')(c2)
    p2 = tf.keras.layers.MaxPooling2D((2, 2))(c2)

    c3 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_no
rmal', padding='same')(p2)
    c3 = tf.keras.layers.Dropout(0.2)(c3)
    c3 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_no
rmal', padding='same')(c3)
    p3 = tf.keras.layers.MaxPooling2D((2, 2))(c3)

    c4 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_n
ormal', padding='same')(p3)
    c4 = tf.keras.layers.Dropout(0.2)(c4)
    c4 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_n
ormal', padding='same')(c4)
    p4 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))(c4)

    c5 = tf.keras.layers.Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_n
ormal', padding='same')(p4)
    c5 = tf.keras.layers.Dropout(0.3)(c5)
    c5 = tf.keras.layers.Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_n
ormal', padding='same')(c5)

    #Expansive path
    u6 = tf.keras.layers.Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(c5
)
    u6 = tf.keras.layers.concatenate([u6, c4])
    c6 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_n
ormal', padding='same')(u6)
    c6 = tf.keras.layers.Dropout(0.2)(c6)
    c6 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_n
ormal', padding='same')(c6)

    u7 = tf.keras.layers.Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(c6)
    u7 = tf.keras.layers.concatenate([u7, c3])
    c7 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_no
rmal', padding='same')(u7)
    c7 = tf.keras.layers.Dropout(0.2)(c7)
    c7 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_no
rmal', padding='same')(c7)

    u8 = tf.keras.layers.Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(c7)
    u8 = tf.keras.layers.concatenate([u8, c2])
    c8 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_no
rmal', padding='same')(u8)
    c8 = tf.keras.layers.Dropout(0.1)(c8)
    c8 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_no
rmal', padding='same')(c8)

    outputs = tf.keras.layers.Conv2D(1, (1, 1), activation='sigmoid')(c8)
```

```
    x = (Reshape(( -1, 1)))(outputs)

    model = tf.keras.Model(inputs=[inputs], outputs=[x])

    return model
```

In [21]:

```
#Был сделан выбор в пользу этой модели.
def TrackNet(input_size):

    imgs_input = Input(shape=input_size)
    x = ( BatchNormalization())(imgs_input)
    #layer1
    x = Conv2D(64, (3, 3), kernel_initializer='random_uniform', padding='same' )(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer2
    x = Conv2D(64, (3, 3), kernel_initializer='random_uniform', padding='same' )(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer3
    x = MaxPooling2D((2, 2), strides=(2, 2) )(x)

    #layer4
    x = Conv2D(128, (3, 3), kernel_initializer='random_uniform', padding='same' )(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer5
    x = Conv2D(128, (3, 3), kernel_initializer='random_uniform', padding='same')(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer6
    x = MaxPooling2D((2, 2), strides=(2, 2) )(x)

    #layer7
    x = Conv2D(256, (3, 3), kernel_initializer='random_uniform', padding='same' )(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer8
    x = Conv2D(256, (3, 3), kernel_initializer='random_uniform', padding='same' )(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer9
    x = Conv2D(256, (3, 3), kernel_initializer='random_uniform', padding='same' )(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer10
    x = MaxPooling2D((2, 2), strides=(2, 2) )(x)

    #layer11
    x = ( Conv2D(512, (3, 3), kernel_initializer='random_uniform', padding='same'))(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer12
    x = ( Conv2D(512, (3, 3), kernel_initializer='random_uniform', padding='same'))(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer13
    x = ( Conv2D(512, (3, 3), kernel_initializer='random_uniform', padding='same'))(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)
```

```
    #layer14
    x = ( UpSampling2D( (2,2)))(x)

    #layer15
    x = ( Conv2D( 256, (3, 3), kernel_initializer='random_uniform', padding='same'))(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer16
    x = ( Conv2D( 256, (3, 3), kernel_initializer='random_uniform', padding='same'))(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer17
    x = ( Conv2D( 256, (3, 3), kernel_initializer='random_uniform', padding='same'))(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer18
    x = ( UpSampling2D( (2,2)))(x)

    #layer19
    x = ( Conv2D( 128 , (3, 3), kernel_initializer='random_uniform', padding='same'))(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer20
    x = ( Conv2D( 128 , (3, 3), kernel_initializer='random_uniform', padding='same' ))(x
)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer21
    x = ( UpSampling2D( (2,2)))(x)

    #layer22
    x = ( Conv2D( 64 , (3, 3), kernel_initializer='random_uniform', padding='same'))(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer23
    x = ( Conv2D( 64 , (3, 3), kernel_initializer='random_uniform', padding='same'))(x)
    x = ( Activation('relu'))(x)
    x = ( BatchNormalization())(x)

    #layer24
    x =  Conv2D(1 , (3, 3) , kernel_initializer='random_uniform', padding='same')(x)
    x = ( Activation('sigmoid'))(x)
    x = (Reshape(( -1, 1)))(x)

    model = Model(imgs_input , x )
    return model
```

In [22]:

```
from skimage import img_as_ubyte
from skimage.transform import hough_circle, hough_circle_peaks
from skimage.feature import canny
```

In [31]:

```
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
        IMG_HEIGHT = 720
        IMG_WIDTH = 1280
```

```
        if downscale:
            IMG_HEIGHT //= 2
            IMG_WIDTH //= 2
        self.INPUT_SIZE = (IMG_HEIGHT, IMG_WIDTH)
        self.model = TrackNet((self.INPUT_SIZE[0], self.INPUT_SIZE[1], 3 * self.stack_s)
)
        #self.model = UNet((self.INPUT_SIZE[0], self.INPUT_SIZE[1], 3 * self.stack_s))
        #self.model = Model2((self.INPUT_SIZE[0], self.INPUT_SIZE[1], 3 * self.stack_s))
        self.epochs = 20
        self.length = 1000

    def save(self, name: str):
        self.model.save_weights(f'/kaggle/working/{name}_save.h5', save_format='h5')

    def load(self, d: dict, name: str):
        # todo: add code for loading model here
        print('Running stub for loading model ...')
        output = f'/kaggle/working/{name}_load.h5'
        gdown.download(f"https://drive.google.com/uc?export=download&confirm=pbef&id={d[n
ame]}", output, quiet=False)
        self.model.load_weights(output)
        print('Loading model done.')

    def predict_on_batch(self, batch: np.ndarray) -> np.ndarray:
        # todo: add code for batch mask prediction here
        predictions = self.model.predict(batch).reshape((batch.shape[0], batch.shape[1],
batch.shape[2]))
        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, ...]
            if self.stack_s > 1:
                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:
        # todo: get ball coordinates from predicted masks
        # remember to upscale predicted coords if you use downscaled images
```

```
        n_frames = pred_prob.shape[0]
        coords = np.zeros([n_frames, 3])
        for i in range(n_frames):
            code, x, y = self.get_blob_centre(pred_prob[i])
            if upscale_coords:
                x, y = 2 * x, 2 * y
            coords[i] = [code, x, y]
        return coords

    def get_blob_centre(self, mask):
        if mask.sum() < 1:
            return 0, -1, -1
        mask[mask < 0.5] = 0
        mask[mask >= 0.5] = 1
        code = 0
        x, y = -1, -1

        image = img_as_ubyte(mask)
        edges = canny(image, sigma=3, low_threshold=10, high_threshold=50)
        hough_radii = np.arange(15, 30, 2)
        hough_res = hough_circle(edges, hough_radii)
        acc, cx, cy, rad = hough_circle_peaks(hough_res, hough_radii,
                                              total_num_peaks=1)
        if cx:
            x = cx[0]
            y = cy[0]
            code = 1
        return code, x, y

    def predict(self, clip: np.ndarray, upscale_coords) -> np.ndarray:
        prob_pr = self._predict_prob_on_clip(clip)
        print(prob_pr.shape)
        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) if se
lf.downscale else data
            labels_gt = load_clip_labels(data_path, game, clip, downscale=False)
            labels_pr, prob_pr = self.predict(data, self.downscale)
            SIBATRACC_per_frame, SIBATRACC_total = Metrics.evaluate_predictions(labels_g
t, 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{cli
p}')
                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_generator, val_gen):
        # todo: implement model training here
        print('Running stub for training model...')
        self.model.compile(optimizer=tf.keras.optimizers.Adadelta(learning_rate=1.0),
              loss=tf.keras.losses.BinaryCrossentropy())
        self.model.fit(train_generator, validation_data=val_gen, epochs=self.epochs, ste
ps_per_epoch = self.length // self.batch_s, validation_steps=self.length // 5 // self.ba
tch_s)
        print('training done.')
```

Пример пайплайна для обучения модели:

In [25]:

```
batch_s = 6
stack_s = 3
downscale = True

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

In [26]:

```
model = SuperTrackingModel(batch_s, stack_s, out_path=output_path, downscale=downscale)
model.model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 360, 640, 9)]     0
_________________________________________________________________
batch_normalization_4 (Batch (None, 360, 640, 9)       36
_________________________________________________________________
conv2d (Conv2D)              (None, 360, 640, 64)      5248
_________________________________________________________________
activation (Activation)      (None, 360, 640, 64)      0
_________________________________________________________________
batch_normalization_5 (Batch (None, 360, 640, 64)      256
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 360, 640, 64)      36928
_________________________________________________________________
activation_1 (Activation)    (None, 360, 640, 64)      0
_________________________________________________________________
batch_normalization_6 (Batch (None, 360, 640, 64)      256
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 180, 320, 64)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 180, 320, 128)     73856
_________________________________________________________________
activation_2 (Activation)    (None, 180, 320, 128)     0
_________________________________________________________________
batch_normalization_7 (Batch (None, 180, 320, 128)     512
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 180, 320, 128)     147584
_________________________________________________________________
activation_3 (Activation)    (None, 180, 320, 128)     0
_________________________________________________________________
batch_normalization_8 (Batch (None, 180, 320, 128)     512
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 90, 160, 128)      0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 90, 160, 256)      295168
_________________________________________________________________
activation_4 (Activation)    (None, 90, 160, 256)      0
_________________________________________________________________
batch_normalization_9 (Batch (None, 90, 160, 256)      1024
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 90, 160, 256)      590080
_________________________________________________________________
activation_5 (Activation)    (None, 90, 160, 256)      0
_________________________________________________________________
batch_normalization_10 (Batc (None, 90, 160, 256)      1024
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 90, 160, 256)      590080
_________________________________________________________________
activation_6 (Activation)    (None, 90, 160, 256)      0
_________________________________________________________________
batch_normalization_11 (Batc (None, 90, 160, 256)      1024
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 45, 80, 256)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 45, 80, 512)       1180160
_________________
```

```
activation_7 (Activation)    (None, 45, 80, 512)       0
_________________________________________________________________
batch_normalization_12 (Batc (None, 45, 80, 512)       2048
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 45, 80, 512)       2359808
_________________________________________________________________
activation_8 (Activation)    (None, 45, 80, 512)       0
_________________________________________________________________
batch_normalization_13 (Batc (None, 45, 80, 512)       2048
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 45, 80, 512)       2359808
_________________________________________________________________
activation_9 (Activation)    (None, 45, 80, 512)       0
_________________________________________________________________
batch_normalization_14 (Batc (None, 45, 80, 512)       2048
_________________________________________________________________
up_sampling2d (UpSampling2D) (None, 90, 160, 512)      0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 90, 160, 256)      1179904
_________________________________________________________________
activation_10 (Activation)   (None, 90, 160, 256)      0
_________________________________________________________________
batch_normalization_15 (Batc (None, 90, 160, 256)      1024
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 90, 160, 256)      590080
_________________________________________________________________
activation_11 (Activation)   (None, 90, 160, 256)      0
_________________________________________________________________
batch_normalization_16 (Batc (None, 90, 160, 256)      1024
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 90, 160, 256)      590080
_________________________________________________________________
activation_12 (Activation)   (None, 90, 160, 256)      0
_________________________________________________________________
batch_normalization_17 (Batc (None, 90, 160, 256)      1024
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 180, 320, 256)     0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 180, 320, 128)     295040
_________________________________________________________________
activation_13 (Activation)   (None, 180, 320, 128)     0
_________________________________________________________________
batch_normalization_18 (Batc (None, 180, 320, 128)     512
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 180, 320, 128)     147584
_________________________________________________________________
activation_14 (Activation)   (None, 180, 320, 128)     0
_________________________________________________________________
batch_normalization_19 (Batc (None, 180, 320, 128)     512
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 360, 640, 128)     0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 360, 640, 64)      73792
_________________________________________________________________
activation_15 (Activation)   (None, 360, 640, 64)      0
_________________________________________________________________
batch_normalization_20 (Batc (None, 360, 640, 64)      256
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 360, 640, 64)      36928
_________________________________________________________________
activation_16 (Activation)   (None, 360, 640, 64)      0
_________________________________________________________________
batch_normalization_21 (Batc (None, 360, 640, 64)      256
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 360, 640, 1)       577
_________________________________________________________________
activation_17 (Activation)   (None, 360, 640, 1)       0
_________________________________________________________________
reshape (Reshape)            (None, 230400, 1)         0
=================================================================
Total params: 10,568,101
Trainable params: 10,560,403
```

```
Non-trainable params: 7,698
_________________________________________________________________
```

In [ ]:

```
train_gen = DataGenerator(Path('../input/tennistrackingassignment/train/'), [1, 2, 3, 4]
, stack_s=stack_s, downscale=downscale, pool_s=5, pool_update_s=4, quiet=False)
val_gen = DataGenerator(Path('../input/tennistrackingassignment/train/'), [5, 6], stack_
s=stack_s, downscale=downscale, pool_s=5, pool_update_s=4, quiet=True)
```

In [ ]:

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

In [ ]:

```
model.save('model_stacks_3')
```

In [27]:

```
name_to_id_dict = {
            'model': '1Gd-e3BmL5u0chntDJGmR76AwEe5ytJul',
            'model8':'1q7AVOO0bIlal1ox5wvP0efvstkJ_ENuh',
            'model9': '1hBshVwnO4pt17L1vPgnNMAy63LwRWhJY',
            'best_model_stacks_3':'1CihHd7M85XpzgfqttQxBK500vN00Hg_i'
        }
```

In [28]:

```
model.load(name_to_id_dict, 'best_model_stacks_3')
```

```
Running stub for loading model ...
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1CihHd7M85XpzgfqttQxBK5
00vN00Hg_i
To: /kaggle/working/best_model_stacks_3_load.h5
100%|██████████| 42.4M/42.4M [00:03<00:00, 13.2MB/s]
Loading model done.
```

In [ ]:

```
output_path = prepare_experiment(Path('/kaggle/working'))
sibatracc_final = model.test(Path('../input/tennistrackingassignment/test/'), [1,2], do_
visualization=False, test_name='test')
print(f'SiBaTrAcc final value: {sibatracc_final}')
```

In [29]:

```
output_path = prepare_experiment(Path('/kaggle/working'))
new_model = SuperTrackingModel(batch_s, stack_s, out_path=output_path, downscale=downscal
e)
new_model.load(name_to_id_dict, 'best_model_stacks_3')
sibatracc_final = new_model.test(Path('../input/tennistrackingassignment/test/'), [1,2],
do_visualization=False, test_name='test')
print(f'SiBaTrAcc final value: {sibatracc_final}')
```

```
Running stub for loading model ...
Downloading...
From: https://drive.google.com/uc?export=download&confirm=pbef&id=1CihHd7M85XpzgfqttQxBK5
00vN00Hg_i
To: /kaggle/working/best_model_stacks_3_load.h5
100%|██████████| 42.4M/42.4M [00:00<00:00, 149MB/s]
Loading model done.
loading clip data (game 1, clip 1) downscaled
loading clip labels (game 1, clip 1)
doing predictions
2022-12-29 22:05:52.803535: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:18
```

```
5] None of the MLIR Optimization Passes are enabled (registered 2)
2022-12-29 22:05:54.093206: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuD
NN version 8005
```

```
predictions are made
(361, 360, 640)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:101: DeprecationWarning: The
truth value of an empty array is ambiguous. Returning False, but in future this will resu
lt in an error. Use `array.size > 0` to check that an array is not empty.
```

```
loading clip data (game 1, clip 2) downscaled
loading clip labels (game 1, clip 2)
doing predictions
predictions are made
(199, 360, 640)
loading clip data (game 1, clip 3) downscaled
loading clip labels (game 1, clip 3)
doing predictions
predictions are made
(36, 360, 640)
loading clip data (game 1, clip 4) downscaled
loading clip labels (game 1, clip 4)
doing predictions
predictions are made
(45, 360, 640)
loading clip data (game 1, clip 5) downscaled
loading clip labels (game 1, clip 5)
doing predictions
predictions are made
(196, 360, 640)
loading clip data (game 1, clip 6) downscaled
loading clip labels (game 1, clip 6)
doing predictions
predictions are made
(551, 360, 640)
loading clip data (game 1, clip 7) downscaled
loading clip labels (game 1, clip 7)
doing predictions
predictions are made
(189, 360, 640)
loading clip data (game 1, clip 8) downscaled
loading clip labels (game 1, clip 8)
doing predictions
predictions are made
(645, 360, 640)
loading clip data (game 2, clip 1) downscaled
loading clip labels (game 2, clip 1)
doing predictions
predictions are made
(83, 360, 640)
loading clip data (game 2, clip 2) downscaled
loading clip labels (game 2, clip 2)
doing predictions
predictions are made
(258, 360, 640)
loading clip data (game 2, clip 3) downscaled
loading clip labels (game 2, clip 3)
doing predictions
predictions are made
(359, 360, 640)
loading clip data (game 2, clip 4) downscaled
loading clip labels (game 2, clip 4)
doing predictions
predictions are made
(106, 360, 640)
loading clip data (game 2, clip 5) downscaled
loading clip labels (game 2, clip 5)
doing predictions
predictions are made
(292, 360, 640)
loading clip data (game 2, clip 6) downscaled
loading clip labels (game 2, clip 6)
```

```
doing predictions
predictions are made
(109, 360, 640)
loading clip data (game 2, clip 7) downscaled
loading clip labels (game 2, clip 7)
doing predictions
predictions are made
(87, 360, 640)
loading clip data (game 2, clip 8) downscaled
loading clip labels (game 2, clip 8)
doing predictions
predictions are made
(56, 360, 640)
loading clip data (game 2, clip 9) downscaled
loading clip labels (game 2, clip 9)
doing predictions
predictions are made
(223, 360, 640)
SiBaTrAcc final value: 0.7454047963575194
```

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**:**

In [ ]:

```
%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**:**

In [ ]:

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

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_TlwtBjuPcEfsPV0Jz7lPA/view?usp=sharing id** файла заключен между **...d/ b /view?...** и равен **1kZ8CC-zfkB_TlwtBjuPcEfsPV0Jz7lPA**

In [ ]:

```
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)
```