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
!pip install gdown
```

```
Collecting moviepy
  Downloading moviepy-1.0.3.tar.gz (388 kB)
     |████████████████████████████████| 388 kB 904 kB/s eta 0:00:01
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: tqdm<5.0,>=4.11.2 in /opt/conda/lib/python3.7/site-package
s (from moviepy) (4.59.0)
Requirement already satisfied: requests<3.0,>=2.8.1 in /opt/conda/lib/python3.7/site-pack
ages (from moviepy) (2.25.1)
Collecting proglog<=1.0.0
  Downloading proglog-0.1.9.tar.gz (10 kB)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/lib/python3.7/site-packages (f
rom moviepy) (1.19.5)
Requirement already satisfied: imageio<3.0,>=2.5 in /opt/conda/lib/python3.7/site-package
s (from moviepy) (2.9.0)
Collecting imageio_ffmpeg>=0.2.0
  Downloading imageio_ffmpeg-0.4.4-py3-none-manylinux2010_x86_64.whl (26.9 MB)
     |████████████████████████████████| 26.9 MB 381 kB/s  eta 0:00:01    |█████
| 4.6 MB 6.7 MB/s eta 0:00:04
Requirement already satisfied: pillow in /opt/conda/lib/python3.7/site-packages (from ima
geio<3.0,>=2.5->moviepy) (7.2.0)
Requirement already satisfied: chardet<5,>=3.0.2 in /opt/conda/lib/python3.7/site-package
s (from requests<3.0,>=2.8.1->moviepy) (4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/site-packag
es (from requests<3.0,>=2.8.1->moviepy) (2020.12.5)
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) (1.26.4)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/lib/python3.7/site-packages (fr
```

```
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/lib/python3.7/site-packages (fr
om requests<3.0,>=2.8.1->moviepy) (2.10)
Building wheels for collected packages: moviepy, proglog
  Building wheel for moviepy (setup.py) ... done
  Created wheel for moviepy: filename=moviepy-1.0.3-py3-none-any.whl size=110726 sha256=2
0e02038541f3778e3791392b68d8c2f5026519f6371f4d3c456f076532b33ec
  Stored in directory: /root/.cache/pip/wheels/56/dc/2b/9cd600d483c04af3353d66623056fc03f
aed76b7518faae4df
  Building wheel for proglog (setup.py) ... done
  Created wheel for proglog: filename=proglog-0.1.9-py3-none-any.whl size=6147 sha256=f73
d225683aca138692f2b500836dcbc7d2919227ade6e75f5dd99736e1d0516
  Stored in directory: /root/.cache/pip/wheels/12/36/1f/dc61e6ac10781d63cf6fa045eb09fa613
a667384e12cb6e6e0
Successfully built moviepy proglog
Installing collected packages: proglog, imageio-ffmpeg, moviepy
Successfully installed imageio-ffmpeg-0.4.4 moviepy-1.0.3 proglog-0.1.9
Collecting gdown
  Downloading gdown-3.13.0.tar.gz (9.3 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
    Preparing wheel metadata ... done
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-packages (from gdown
) (4.59.0)
Requirement already satisfied: requests[socks]>=2.12.0 in /opt/conda/lib/python3.7/site-p
ackages (from gdown) (2.25.1)
Requirement already satisfied: filelock in /opt/conda/lib/python3.7/site-packages (from g
down) (3.0.12)
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-packages (from gdown)
(1.15.0)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/lib/python3.7/site-packages (fr
om requests[socks]>=2.12.0->gdown) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.7/site-pac
kages (from requests[socks]>=2.12.0->gdown) (1.26.4)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/site-packag
es (from requests[socks]>=2.12.0->gdown) (2020.12.5)
Requirement already satisfied: chardet<5,>=3.0.2 in /opt/conda/lib/python3.7/site-package
s (from requests[socks]>=2.12.0->gdown) (4.0.0)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /opt/conda/lib/python3.7/site-pa
ckages (from requests[socks]>=2.12.0->gdown) (1.7.1)
Building wheels for collected packages: gdown
  Building wheel for gdown (PEP 517) ... done
  Created wheel for gdown: filename=gdown-3.13.0-py3-none-any.whl size=9034 sha256=65d635
ffc7c1fd980fb4b6e40a710fe07a6472c48f671f8a49d0eb87584605ec
  Stored in directory: /root/.cache/pip/wheels/2f/2a/2f/86449b6bdbaa9aef873f68332b68be6bf
bc386b9219f47157d
Successfully built gdown
Installing collected packages: gdown
Successfully installed gdown-3.13.0
```

Импорт необходимых зависимостей:

In [2]:

```python
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
```

In [3]:

```
import keras
from keras import layers
from keras import backend as K
import matplotlib.pyplot as plt
import gdown
```

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**.**

In [4]:

```
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 [5]:

```
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()]
    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 = 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.

In [6]:

```
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

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 [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, ...]
        frames_stack = np.squeeze(np.split(frames_stack, indices_or_sections=self.stack_
s, axis=0))
        frames_stack = np.concatenate(frames_stack, axis=-1)
        mask = m[start + self.stack_s - 1, ...]
        return frames_stack, mask

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

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

Пример использования **DataGenerator**

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

In [ ]:

```
stack_s = 5
batch_s = 4
train_gen = DataGenerator(Path('../input/tennistrackingassignment/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)
```

In [ ]:

```
import matplotlib.pyplot as plt

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()
print(stack.shape, mask.shape)

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

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 [8]:

```
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
```

Основной класс модели **SuperTrackingModel**

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

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

code (0, если мяча в кадре нет и **> 0,** если мяч в кадре есть**)** для корректного вычисления качества трекинга**.**

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

In [9]:

```
def dice_coef(y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection) / (K.sum(y_true_f) + K.sum(y_pred_f))

def dice_coef_loss(y_true, y_pred):
    return -dice_coef(y_true, y_pred)
```

In [10]:

```
def coord(img):
    max_y = 0
    max_sum_y = 0
    for y in range(img.shape[0]):
        sum_y = 0
        for x in range(img.shape[1]):
            sum_y += img[y,x]
        if (sum_y > max_sum_y):
            max_sum_y = sum_y
            max_y = y
    max_x = 0
    max_sum_x = 0
    for x in range(img.shape[1]):
        sum_x = 0
        for y in range(img.shape[0]):
            sum_x += img[y,x]
        if (sum_x > max_sum_x):
            max_sum_x = sum_x
            max_x = x
    return max_x, max_y, max_sum_x, max_sum_y
```

In [11]:

```
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

    def build_unet(self):
        inputs = keras.Input(shape=(360, 640, 9))
        conv1 = layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_norm
al', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(inputs)
        conv1 = layers.BatchNormalization()(conv1)
        conv1 = layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_norm
al', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(conv1)
        pool1 = layers.MaxPooling2D(pool_size=(2, 2))(conv1)
        pool1 = layers.Dropout(0.1)(pool1)

        conv2 = layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_norm
al', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(pool1)
        conv2 = layers.BatchNormalization()(conv2)
        conv2 = layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_norm
al', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(conv2)
        pool2 = layers.MaxPooling2D(pool_size=(2, 2))(conv2)
        pool2 = layers.Dropout(0.1)(pool2)

        conv3 = layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_norm
al', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(pool2)
        conv3 = layers.BatchNormalization()(conv3)
        conv3 = layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_norm
```

```
al', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(conv3)
        pool3 = layers.MaxPooling2D(pool_size=(2, 2))(conv3)
        pool3 = layers.Dropout(0.1)(pool3)

        conv4 = layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_nor
mal', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(pool3)
        conv4 = layers.BatchNormalization()(conv4)
        conv4 = layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_nor
mal', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(conv4)

        up5 = layers.Conv2DTranspose(32, (3, 3), strides = (2, 2), padding = 'same')(con
v4)
        up5 = layers.concatenate([up5, conv3])
        up5 = layers.Dropout(0.1)(up5)
        conv5 = layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_norm
al', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(up5)
        conv5 = layers.BatchNormalization()(conv5)
        conv5 = layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_norm
al', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(conv5)

        up6 = layers.Conv2DTranspose(16, (3, 3), strides = (2, 2), padding = 'same')(con
v5)
        up6 = layers.concatenate([up6, conv2])
        up6 = layers.Dropout(0.1)(up6)
        conv6 = layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_norm
al', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(up6)
        conv6 = layers.BatchNormalization()(conv6)
        conv6 = layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_norm
al', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(conv6)

        up7 = layers.Conv2DTranspose(8, (3, 3), strides = (2, 2), padding = 'same')(conv
6)
        up7 = layers.concatenate([up7, conv1])
        up7 = layers.Dropout(0.1)(up7)
        conv7 = layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_norm
al', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(up7)
        conv7 = layers.BatchNormalization()(conv7)
        conv7 = layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_norm
al', kernel_regularizer = keras.regularizers.l2(0.001), padding='same')(conv7)

        outputs = layers.Conv2D(1, (1, 1), activation='sigmoid')(conv7)

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

        model.compile(keras.optimizers.Adam(lr=0.0001), loss="binary_crossentropy", metr
ics=[dice_coef])
        self.unet = model

    def save(self):
        self.unet.save_weights('/kaggle/working/my_model_weights.h5')

    def load(self):
        #50 epochs
        #id = '1UlprriYKDynnLYhKbqNl4k8B-zALYfY5'
        #80 epochs
        id = '1RHG_027l_TDKpEYwupjZnC0gUVVmSgLr'
        url = f'https://drive.google.com/uc?id={id}'
        output = 'loaded_weights.h5'
        gdown.download(url, output, quiet=False)
        self.build_unet()
        self.unet.load_weights("/kaggle/working/loaded_weights.h5")

    def predict_on_batch(self, batch: np.ndarray) -> np.ndarray:
        predictions = self.unet.predict(batch)
        return predictions.reshape(self.batch_s, 360, 640)

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

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

    def get_labels_from_prediction(self, pred_prob: np.ndarray, upscale_coords: bool) ->
np.ndarray:
        n_frames = pred_prob.shape[0]
        coords = np.zeros([n_frames, 3])
        for i in range(n_frames):
            x, y, sum_x, sum_y = coord(pred_prob[i])
            if (sum_x < 10.) or (sum_y < 10.):
                code = 0
            else:
                code = 1
            if upscale_coords:
                x, y = x * 2, y * 2
            coords[i] = [code, 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) if se
lf.downscale else data
            labels_gt = load_clip_labels(data_path, game, clip, downscale=False)
            labels_pr, prob_pr = self.predict(data)
            SIBATRACC_per_frame, SIBATRACC_total = Metrics.evaluate_predictions(labels_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, param_1=None, param_2=None, param_3=None, param_4=None, param_5=None
, param_6=None):
        print('Running stub for training model...')
        file_path_best = "/kaggle/working/exp_1/model_weights_best.h5"
        #LBL2 автоматическое сохранение модели на обучении
        modelcheckpoint_best = keras.callbacks.ModelCheckpoint(file_path_best,
                                                               monitor='val_loss',
                                                               mode='auto',
                                                               verbose=1,
                                                               save_best_only=True,
                                                               save_weights_only=True)
        callbacks = [modelcheckpoint_best]
        EPOCHS = 120
        #LBL1 валидация модели на тестовой выборке
        #LBL3 вывод различных показателей в процессе обучения
        history = self.unet.fit_generator(param_1(self.batch_s),
                                          steps_per_epoch=150,
                                          epochs=EPOCHS,
                                          callbacks=callbacks,
                                          validation_data=param_2(self.batch_s),
                                          validation_steps=50)
        print('training done.')
        #LBL4 построение графиков, визуализирующих процесс обучения
        loss = history.history['loss']
        val_loss = history.history['val_loss']
        epochs_range = range(EPOCHS)
        plt.figure(figsize=(8,8))
        plt.subplot(1, 2, 1)
        plt.plot(epochs_range, loss, label='Потери на обучении')
        plt.legend(loc='lower right')
        plt.title('Потери на обучающих данных')

        plt.subplot(1, 2, 2)
        plt.plot(epochs_range, val_loss, label='Потери на валидации')
        plt.legend(loc='upper right')
        plt.title('Потери на валидационных данных')
        plt.show()
```

In [12]:

```
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)
model.build_unet()
model.unet.summary()
```

```
Model: "model"
__________________________________________________________________________________________
_________
Layer (type)                    Output Shape         Param #     Connected to

==========================================================================================
=========
input_1 (InputLayer)            [(None, 360, 640, 9) 0


__________________________________________________________________________________________
_________
conv2d (Conv2D)                 (None, 360, 640, 32) 2624        input_1[0][0]


__________________________________________________________________________________________
_________
batch_normalization (BatchNorma (None, 360, 640, 32) 128         conv2d[0][0]


__________________________________________________________________________________________
_________
```

```
conv2d_1 (Conv2D)               (None, 360, 640, 32) 9248        batch_normalization[0][0
]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 180, 320, 32) 0           conv2d_1[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 180, 320, 32) 0           max_pooling2d[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 180, 320, 32) 9248        dropout[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 180, 320, 32) 128         conv2d_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 180, 320, 32) 9248        batch_normalization_1[0]
[0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 90, 160, 32)  0           conv2d_3[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 90, 160, 32)  0           max_pooling2d_1[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 90, 160, 64)  18496       dropout_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 90, 160, 64)  256         conv2d_4[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 90, 160, 64)  36928       batch_normalization_2[0]
[0]
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 45, 80, 64)   0           conv2d_5[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 45, 80, 64)   0           max_pooling2d_2[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 45, 80, 128)  73856       dropout_2[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 45, 80, 128)  512         conv2d_6[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 45, 80, 128)  147584      batch_normalization_3[0]
[0]
__________________________________________________________________________________________________
conv2d_transpose (Conv2DTranspo (None, 90, 160, 32)  36896       conv2d_7[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 90, 160, 96)  0           conv2d_transpose[0][0]
                                                                 conv2d_5[0][0]
```

```
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 90, 160, 96)  0           concatenate[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 90, 160, 64)  55360       dropout_3[0][0]
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 90, 160, 64)  256         conv2d_8[0][0]
__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 90, 160, 64)  36928       batch_normalization_4[0][0]
__________________________________________________________________________________________________
conv2d_transpose_1 (Conv2DTrans (None, 180, 320, 16) 9232        conv2d_9[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 180, 320, 48) 0           conv2d_transpose_1[0][0]
                                                                 conv2d_3[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 180, 320, 48) 0           concatenate_1[0][0]
__________________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, 180, 320, 32) 13856       dropout_4[0][0]
__________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 180, 320, 32) 128         conv2d_10[0][0]
__________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 180, 320, 32) 9248        batch_normalization_5[0][0]
__________________________________________________________________________________________________
conv2d_transpose_2 (Conv2DTrans (None, 360, 640, 8)  2312        conv2d_11[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 360, 640, 40) 0           conv2d_transpose_2[0][0]
                                                                 conv2d_1[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 360, 640, 40) 0           concatenate_2[0][0]
__________________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 360, 640, 32) 11552       dropout_5[0][0]
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 360, 640, 32) 128         conv2d_12[0][0]
__________________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 360, 640, 32) 9248        batch_normalization_6[0][0]
__________________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, 360, 640, 1)  33          conv2d_13[0][0]
```

```
==========================================================================================
=========
Total params: 493,433
Trainable params: 492,665
Non-trainable params: 768
__________________________________________________________________________________________
_________
```

In [13]:

```
train_gen = DataGenerator(Path('../input/tennistrackingassignment/train/'), [1, 2, 3, 4]
, stack_s=stack_s, downscale=True, pool_s=10, pool_update_s=4, quiet=True)
val_gen = DataGenerator(Path('../input/tennistrackingassignment/test/'), [1, 2], stack_s
=stack_s, downscale=True, pool_s=4, pool_update_s=2, quiet=True)
```

In [14]:

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

```
Running stub for training model...
Epoch 1/120
```

```
/opt/conda/lib/python3.7/site-packages/tensorflow/python/keras/engine/training.py:1844: U
serWarning: `Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
150/150 [==============================] - 98s 593ms/step - loss: 2.0671 - dice_coef: 0.0
051 - val_loss: 1.2154 - val_dice_coef: 0.0040

Epoch 00001: val_loss improved from inf to 1.21540, saving model to /kaggle/working/exp_1
/model_weights_best.h5
Epoch 2/120
150/150 [==============================] - 88s 587ms/step - loss: 1.1186 - dice_coef: 0.0
051 - val_loss: 0.8750 - val_dice_coef: 0.0045

Epoch 00002: val_loss improved from 1.21540 to 0.87498, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 3/120
150/150 [==============================] - 88s 586ms/step - loss: 0.8179 - dice_coef: 0.0
050 - val_loss: 0.6660 - val_dice_coef: 0.0042

Epoch 00003: val_loss improved from 0.87498 to 0.66604, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 4/120
150/150 [==============================] - 100s 668ms/step - loss: 0.6264 - dice_coef: 0.
0058 - val_loss: 0.5228 - val_dice_coef: 0.0037

Epoch 00004: val_loss improved from 0.66604 to 0.52282, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 5/120
150/150 [==============================] - 92s 611ms/step - loss: 0.4941 - dice_coef: 0.0
095 - val_loss: 0.4211 - val_dice_coef: 0.0031

Epoch 00005: val_loss improved from 0.52282 to 0.42112, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 6/120
150/150 [==============================] - 88s 586ms/step - loss: 0.3993 - dice_coef: 0.0
125 - val_loss: 0.3473 - val_dice_coef: 0.0026

Epoch 00006: val_loss improved from 0.42112 to 0.34731, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 7/120
150/150 [==============================] - 91s 608ms/step - loss: 0.3302 - dice_coef: 0.0
206 - val_loss: 0.2933 - val_dice_coef: 0.0023

Epoch 00007: val_loss improved from 0.34731 to 0.29329, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 8/120
150/150 [==============================] - 93s 619ms/step - loss: 0.2792 - dice_coef: 0.0
245 - val_loss: 0.2522 - val_dice_coef: 0.0027
```

```
Epoch 00008: val_loss improved from 0.29329 to 0.25217, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 9/120
150/150 [==============================] - 88s 586ms/step - loss: 0.2408 - dice_coef: 0.0
251 - val_loss: 0.2213 - val_dice_coef: 0.0028

Epoch 00009: val_loss improved from 0.25217 to 0.22131, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 10/120
150/150 [==============================] - 98s 656ms/step - loss: 0.2101 - dice_coef: 0.0
480 - val_loss: 0.1970 - val_dice_coef: 0.0023

Epoch 00010: val_loss improved from 0.22131 to 0.19699, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 11/120
150/150 [==============================] - 98s 656ms/step - loss: 0.1858 - dice_coef: 0.0
730 - val_loss: 0.1765 - val_dice_coef: 0.0132

Epoch 00011: val_loss improved from 0.19699 to 0.17647, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 12/120
150/150 [==============================] - 88s 586ms/step - loss: 0.1649 - dice_coef: 0.1
423 - val_loss: 0.1557 - val_dice_coef: 0.0727

Epoch 00012: val_loss improved from 0.17647 to 0.15566, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 13/120
150/150 [==============================] - 88s 586ms/step - loss: 0.1478 - dice_coef: 0.2
111 - val_loss: 0.1399 - val_dice_coef: 0.1749

Epoch 00013: val_loss improved from 0.15566 to 0.13991, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 14/120
150/150 [==============================] - 88s 588ms/step - loss: 0.1336 - dice_coef: 0.2
432 - val_loss: 0.1253 - val_dice_coef: 0.2639

Epoch 00014: val_loss improved from 0.13991 to 0.12528, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 15/120
150/150 [==============================] - 88s 586ms/step - loss: 0.1218 - dice_coef: 0.2
496 - val_loss: 0.1142 - val_dice_coef: 0.1977

Epoch 00015: val_loss improved from 0.12528 to 0.11424, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 16/120
150/150 [==============================] - 102s 679ms/step - loss: 0.1112 - dice_coef: 0.
2464 - val_loss: 0.1046 - val_dice_coef: 0.2548

Epoch 00016: val_loss improved from 0.11424 to 0.10462, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 17/120
150/150 [==============================] - 88s 587ms/step - loss: 0.1018 - dice_coef: 0.2
558 - val_loss: 0.1524 - val_dice_coef: 0.0339

Epoch 00017: val_loss did not improve from 0.10462
Epoch 18/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0933 - dice_coef: 0.2
775 - val_loss: 0.0874 - val_dice_coef: 0.3123

Epoch 00018: val_loss improved from 0.10462 to 0.08736, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 19/120
150/150 [==============================] - 95s 635ms/step - loss: 0.0857 - dice_coef: 0.2
860 - val_loss: 0.0821 - val_dice_coef: 0.2565

Epoch 00019: val_loss improved from 0.08736 to 0.08206, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 20/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0789 - dice_coef: 0.2
849 - val_loss: 0.0748 - val_dice_coef: 0.2700

Epoch 00020: val_loss improved from 0.08206 to 0.07484, saving model to /kaggle/working/e
```

```
xp_1/model_weights_best.h5
Epoch 21/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0726 - dice_coef: 0.3
043 - val_loss: 0.0701 - val_dice_coef: 0.1885

Epoch 00021: val_loss improved from 0.07484 to 0.07010, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 22/120
150/150 [==============================] - 93s 623ms/step - loss: 0.0667 - dice_coef: 0.3
115 - val_loss: 0.0645 - val_dice_coef: 0.2431

Epoch 00022: val_loss improved from 0.07010 to 0.06446, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 23/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0616 - dice_coef: 0.3
196 - val_loss: 0.0604 - val_dice_coef: 0.2641

Epoch 00023: val_loss improved from 0.06446 to 0.06037, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 24/120
150/150 [==============================] - 88s 588ms/step - loss: 0.0570 - dice_coef: 0.3
330 - val_loss: 0.0554 - val_dice_coef: 0.2475

Epoch 00024: val_loss improved from 0.06037 to 0.05545, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 25/120
150/150 [==============================] - 98s 655ms/step - loss: 0.0530 - dice_coef: 0.3
128 - val_loss: 0.0528 - val_dice_coef: 0.1937

Epoch 00025: val_loss improved from 0.05545 to 0.05276, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 26/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0487 - dice_coef: 0.3
464 - val_loss: 0.0501 - val_dice_coef: 0.2017

Epoch 00026: val_loss improved from 0.05276 to 0.05010, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 27/120
150/150 [==============================] - 98s 653ms/step - loss: 0.0453 - dice_coef: 0.3
471 - val_loss: 0.0468 - val_dice_coef: 0.2533

Epoch 00027: val_loss improved from 0.05010 to 0.04677, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 28/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0427 - dice_coef: 0.3
398 - val_loss: 0.0427 - val_dice_coef: 0.2485

Epoch 00028: val_loss improved from 0.04677 to 0.04268, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 29/120
150/150 [==============================] - 93s 618ms/step - loss: 0.0393 - dice_coef: 0.3
682 - val_loss: 0.0412 - val_dice_coef: 0.2268

Epoch 00029: val_loss improved from 0.04268 to 0.04125, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 30/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0372 - dice_coef: 0.3
557 - val_loss: 0.0434 - val_dice_coef: 0.1922

Epoch 00030: val_loss did not improve from 0.04125
Epoch 31/120
150/150 [==============================] - 94s 630ms/step - loss: 0.0349 - dice_coef: 0.3
534 - val_loss: 0.0382 - val_dice_coef: 0.2921

Epoch 00031: val_loss improved from 0.04125 to 0.03823, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 32/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0326 - dice_coef: 0.3
531 - val_loss: 0.0349 - val_dice_coef: 0.2073

Epoch 00032: val_loss improved from 0.03823 to 0.03491, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
```

```
Epoch 33/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0306 - dice_coef: 0.3
762 - val_loss: 0.0341 - val_dice_coef: 0.2465

Epoch 00033: val_loss improved from 0.03491 to 0.03411, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 34/120
150/150 [==============================] - 92s 614ms/step - loss: 0.0288 - dice_coef: 0.3
695 - val_loss: 0.0326 - val_dice_coef: 0.2069

Epoch 00034: val_loss improved from 0.03411 to 0.03259, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 35/120
150/150 [==============================] - 92s 618ms/step - loss: 0.0274 - dice_coef: 0.3
805 - val_loss: 0.0300 - val_dice_coef: 0.2704

Epoch 00035: val_loss improved from 0.03259 to 0.03000, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 36/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0260 - dice_coef: 0.3
893 - val_loss: 0.0274 - val_dice_coef: 0.2775

Epoch 00036: val_loss improved from 0.03000 to 0.02743, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 37/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0251 - dice_coef: 0.3
766 - val_loss: 0.0279 - val_dice_coef: 0.1951

Epoch 00037: val_loss did not improve from 0.02743
Epoch 38/120
150/150 [==============================] - 94s 630ms/step - loss: 0.0239 - dice_coef: 0.3
794 - val_loss: 0.0263 - val_dice_coef: 0.2717

Epoch 00038: val_loss improved from 0.02743 to 0.02632, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 39/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0227 - dice_coef: 0.3
680 - val_loss: 0.0358 - val_dice_coef: 0.0597

Epoch 00039: val_loss did not improve from 0.02632
Epoch 40/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0214 - dice_coef: 0.3
959 - val_loss: 0.0260 - val_dice_coef: 0.2037

Epoch 00040: val_loss improved from 0.02632 to 0.02602, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 41/120
150/150 [==============================] - 92s 611ms/step - loss: 0.0207 - dice_coef: 0.3
960 - val_loss: 0.0302 - val_dice_coef: 0.0888

Epoch 00041: val_loss did not improve from 0.02602
Epoch 42/120
150/150 [==============================] - 92s 615ms/step - loss: 0.0197 - dice_coef: 0.3
870 - val_loss: 0.0212 - val_dice_coef: 0.3415

Epoch 00042: val_loss improved from 0.02602 to 0.02124, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 43/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0189 - dice_coef: 0.3
928 - val_loss: 0.0280 - val_dice_coef: 0.1420

Epoch 00043: val_loss did not improve from 0.02124
Epoch 44/120
150/150 [==============================] - 92s 612ms/step - loss: 0.0182 - dice_coef: 0.3
878 - val_loss: 0.0253 - val_dice_coef: 0.2340

Epoch 00044: val_loss did not improve from 0.02124
Epoch 45/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0175 - dice_coef: 0.3
961 - val_loss: 0.0252 - val_dice_coef: 0.2203

Epoch 00045: val_loss did not improve from 0.02124
```

```
Epoch 46/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0172 - dice_coef: 0.3
871 - val_loss: 0.0227 - val_dice_coef: 0.2617

Epoch 00046: val_loss did not improve from 0.02124
Epoch 47/120
150/150 [==============================] - 99s 664ms/step - loss: 0.0166 - dice_coef: 0.3
988 - val_loss: 0.0191 - val_dice_coef: 0.2449

Epoch 00047: val_loss improved from 0.02124 to 0.01905, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 48/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0163 - dice_coef: 0.3
737 - val_loss: 0.0234 - val_dice_coef: 0.1286

Epoch 00048: val_loss did not improve from 0.01905
Epoch 49/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0155 - dice_coef: 0.3
984 - val_loss: 0.0321 - val_dice_coef: 0.0100

Epoch 00049: val_loss did not improve from 0.01905
Epoch 50/120
150/150 [==============================] - 90s 601ms/step - loss: 0.0152 - dice_coef: 0.3
927 - val_loss: 0.0232 - val_dice_coef: 0.1528

Epoch 00050: val_loss did not improve from 0.01905
Epoch 51/120
150/150 [==============================] - 91s 608ms/step - loss: 0.0150 - dice_coef: 0.3
880 - val_loss: 0.0167 - val_dice_coef: 0.2819

Epoch 00051: val_loss improved from 0.01905 to 0.01673, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 52/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0145 - dice_coef: 0.3
789 - val_loss: 0.0257 - val_dice_coef: 0.1192

Epoch 00052: val_loss did not improve from 0.01673
Epoch 53/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0140 - dice_coef: 0.3
870 - val_loss: 0.0195 - val_dice_coef: 0.2232

Epoch 00053: val_loss did not improve from 0.01673
Epoch 54/120
150/150 [==============================] - 90s 602ms/step - loss: 0.0138 - dice_coef: 0.3
924 - val_loss: 0.0172 - val_dice_coef: 0.2358

Epoch 00054: val_loss did not improve from 0.01673
Epoch 55/120
150/150 [==============================] - 88s 589ms/step - loss: 0.0138 - dice_coef: 0.3
834 - val_loss: 0.0190 - val_dice_coef: 0.1919

Epoch 00055: val_loss did not improve from 0.01673
Epoch 56/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0132 - dice_coef: 0.4
146 - val_loss: 0.0276 - val_dice_coef: 0.0603

Epoch 00056: val_loss did not improve from 0.01673
Epoch 57/120
150/150 [==============================] - 94s 626ms/step - loss: 0.0130 - dice_coef: 0.4
089 - val_loss: 0.0228 - val_dice_coef: 0.0928

Epoch 00057: val_loss did not improve from 0.01673
Epoch 58/120
150/150 [==============================] - 90s 600ms/step - loss: 0.0128 - dice_coef: 0.3
979 - val_loss: 0.0195 - val_dice_coef: 0.2126

Epoch 00058: val_loss did not improve from 0.01673
Epoch 59/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0127 - dice_coef: 0.4
095 - val_loss: 0.0259 - val_dice_coef: 0.0322

Epoch 00059: val_loss did not improve from 0.01673
```

```
Epoch 60/120
150/150 [==============================] - 90s 601ms/step - loss: 0.0125 - dice_coef: 0.3
955 - val_loss: 0.0210 - val_dice_coef: 0.1784

Epoch 00060: val_loss did not improve from 0.01673
Epoch 61/120
150/150 [==============================] - 90s 604ms/step - loss: 0.0119 - dice_coef: 0.4
197 - val_loss: 0.0155 - val_dice_coef: 0.2985

Epoch 00061: val_loss improved from 0.01673 to 0.01552, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 62/120
150/150 [==============================] - 92s 616ms/step - loss: 0.0119 - dice_coef: 0.4
103 - val_loss: 0.0196 - val_dice_coef: 0.1858

Epoch 00062: val_loss did not improve from 0.01552
Epoch 63/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0118 - dice_coef: 0.4
228 - val_loss: 0.0181 - val_dice_coef: 0.2539

Epoch 00063: val_loss did not improve from 0.01552
Epoch 64/120
150/150 [==============================] - 93s 622ms/step - loss: 0.0118 - dice_coef: 0.4
152 - val_loss: 0.0176 - val_dice_coef: 0.2197

Epoch 00064: val_loss did not improve from 0.01552
Epoch 65/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0113 - dice_coef: 0.4
149 - val_loss: 0.0152 - val_dice_coef: 0.2989

Epoch 00065: val_loss improved from 0.01552 to 0.01524, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 66/120
150/150 [==============================] - 89s 592ms/step - loss: 0.0107 - dice_coef: 0.4
095 - val_loss: 0.0199 - val_dice_coef: 0.2657

Epoch 00066: val_loss did not improve from 0.01524
Epoch 67/120
150/150 [==============================] - 91s 607ms/step - loss: 0.0110 - dice_coef: 0.3
986 - val_loss: 0.0166 - val_dice_coef: 0.2888

Epoch 00067: val_loss did not improve from 0.01524
Epoch 68/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0108 - dice_coef: 0.4
213 - val_loss: 0.0226 - val_dice_coef: 0.0540

Epoch 00068: val_loss did not improve from 0.01524
Epoch 69/120
150/150 [==============================] - 90s 602ms/step - loss: 0.0110 - dice_coef: 0.4
058 - val_loss: 0.0154 - val_dice_coef: 0.2684

Epoch 00069: val_loss did not improve from 0.01524
Epoch 70/120
150/150 [==============================] - 93s 622ms/step - loss: 0.0107 - dice_coef: 0.4
195 - val_loss: 0.0181 - val_dice_coef: 0.1854

Epoch 00070: val_loss did not improve from 0.01524
Epoch 71/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0106 - dice_coef: 0.4
106 - val_loss: 0.0117 - val_dice_coef: 0.3724

Epoch 00071: val_loss improved from 0.01524 to 0.01169, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 72/120
150/150 [==============================] - 89s 593ms/step - loss: 0.0104 - dice_coef: 0.4
091 - val_loss: 0.0153 - val_dice_coef: 0.2403

Epoch 00072: val_loss did not improve from 0.01169
Epoch 73/120
150/150 [==============================] - 92s 617ms/step - loss: 0.0105 - dice_coef: 0.3
963 - val_loss: 0.0147 - val_dice_coef: 0.2767
```

```
Epoch 00073: val_loss did not improve from 0.01169
Epoch 74/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0102 - dice_coef: 0.4
203 - val_loss: 0.0246 - val_dice_coef: 0.1404

Epoch 00074: val_loss did not improve from 0.01169
Epoch 75/120
150/150 [==============================] - 90s 600ms/step - loss: 0.0104 - dice_coef: 0.3
994 - val_loss: 0.0125 - val_dice_coef: 0.2293

Epoch 00075: val_loss did not improve from 0.01169
Epoch 76/120
150/150 [==============================] - 90s 602ms/step - loss: 0.0099 - dice_coef: 0.4
197 - val_loss: 0.0175 - val_dice_coef: 0.1822

Epoch 00076: val_loss did not improve from 0.01169
Epoch 77/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0098 - dice_coef: 0.4
113 - val_loss: 0.0217 - val_dice_coef: 0.1377

Epoch 00077: val_loss did not improve from 0.01169
Epoch 78/120
150/150 [==============================] - 89s 593ms/step - loss: 0.0096 - dice_coef: 0.4
167 - val_loss: 0.0239 - val_dice_coef: 0.0362

Epoch 00078: val_loss did not improve from 0.01169
Epoch 79/120
150/150 [==============================] - 94s 628ms/step - loss: 0.0095 - dice_coef: 0.4
232 - val_loss: 0.0268 - val_dice_coef: 0.0389

Epoch 00079: val_loss did not improve from 0.01169
Epoch 80/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0098 - dice_coef: 0.4
047 - val_loss: 0.0230 - val_dice_coef: 0.1670

Epoch 00080: val_loss did not improve from 0.01169
Epoch 81/120
150/150 [==============================] - 92s 614ms/step - loss: 0.0094 - dice_coef: 0.4
216 - val_loss: 0.0125 - val_dice_coef: 0.3223

Epoch 00081: val_loss did not improve from 0.01169
Epoch 82/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0094 - dice_coef: 0.4
257 - val_loss: 0.0125 - val_dice_coef: 0.2886

Epoch 00082: val_loss did not improve from 0.01169
Epoch 83/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0096 - dice_coef: 0.4
023 - val_loss: 0.0152 - val_dice_coef: 0.2726

Epoch 00084: val_loss did not improve from 0.01169
Epoch 85/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0094 - dice_coef: 0.4
101 - val_loss: 0.0118 - val_dice_coef: 0.3544

Epoch 00085: val_loss did not improve from 0.01169
Epoch 86/120
150/150 [==============================] - 93s 619ms/step - loss: 0.0093 - dice_coef: 0.4
084 - val_loss: 0.0202 - val_dice_coef: 0.1668

Epoch 00086: val_loss did not improve from 0.01169
Epoch 87/120
150/150 [==============================] - 90s 603ms/step - loss: 0.0096 - dice_coef: 0.3
880 - val_loss: 0.0117 - val_dice_coef: 0.3501

Epoch 00087: val_loss did not improve from 0.01169
Epoch 88/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0088 - dice_coef: 0.4
284 - val_loss: 0.0151 - val_dice_coef: 0.2375

Epoch 00088: val_loss did not improve from 0.01169
Epoch 89/120
```

```
150/150 [==============================] - 91s 608ms/step - loss: 0.0090 - dice_coef: 0.4
196 - val_loss: 0.0143 - val_dice_coef: 0.2203

Epoch 00089: val_loss did not improve from 0.01169
Epoch 90/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0090 - dice_coef: 0.4
169 - val_loss: 0.0200 - val_dice_coef: 0.2148

Epoch 00090: val_loss did not improve from 0.01169
Epoch 91/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0089 - dice_coef: 0.4
128 - val_loss: 0.0194 - val_dice_coef: 0.2064

Epoch 00091: val_loss did not improve from 0.01169
Epoch 92/120
150/150 [==============================] - 89s 592ms/step - loss: 0.0088 - dice_coef: 0.4
090 - val_loss: 0.0129 - val_dice_coef: 0.2711

Epoch 00092: val_loss did not improve from 0.01169
Epoch 93/120
150/150 [==============================] - 91s 608ms/step - loss: 0.0084 - dice_coef: 0.4
347 - val_loss: 0.0123 - val_dice_coef: 0.3586

Epoch 00093: val_loss did not improve from 0.01169
Epoch 94/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0085 - dice_coef: 0.4
348 - val_loss: 0.0113 - val_dice_coef: 0.3691

Epoch 00094: val_loss improved from 0.01169 to 0.01132, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 95/120
150/150 [==============================] - 89s 597ms/step - loss: 0.0086 - dice_coef: 0.4
219 - val_loss: 0.0197 - val_dice_coef: 0.0940

Epoch 00095: val_loss did not improve from 0.01132
Epoch 96/120
150/150 [==============================] - 92s 612ms/step - loss: 0.0085 - dice_coef: 0.4
235 - val_loss: 0.0166 - val_dice_coef: 0.1906

Epoch 00096: val_loss did not improve from 0.01132
Epoch 97/120
150/150 [==============================] - 88s 585ms/step - loss: 0.0085 - dice_coef: 0.4
133 - val_loss: 0.0243 - val_dice_coef: 0.0623

Epoch 00097: val_loss did not improve from 0.01132
Epoch 98/120
150/150 [==============================] - 91s 610ms/step - loss: 0.0083 - dice_coef: 0.4
227 - val_loss: 0.0113 - val_dice_coef: 0.3340

Epoch 00098: val_loss improved from 0.01132 to 0.01126, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 99/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0084 - dice_coef: 0.4
201 - val_loss: 0.0140 - val_dice_coef: 0.2887

Epoch 00099: val_loss did not improve from 0.01126
Epoch 100/120
150/150 [==============================] - 96s 641ms/step - loss: 0.0085 - dice_coef: 0.4
071 - val_loss: 0.0124 - val_dice_coef: 0.3214

Epoch 00100: val_loss did not improve from 0.01126
Epoch 101/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0082 - dice_coef: 0.4
225 - val_loss: 0.0117 - val_dice_coef: 0.2709

Epoch 00101: val_loss did not improve from 0.01126
Epoch 102/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0084 - dice_coef: 0.4
231 - val_loss: 0.0117 - val_dice_coef: 0.2943

Epoch 00102: val_loss did not improve from 0.01126
Epoch 103/120
```

```
150/150 [==============================] - 90s 598ms/step - loss: 0.0082 - dice_coef: 0.4
175 - val_loss: 0.0124 - val_dice_coef: 0.3127

Epoch 00103: val_loss did not improve from 0.01126
Epoch 104/120
150/150 [==============================] - 90s 604ms/step - loss: 0.0081 - dice_coef: 0.4
305 - val_loss: 0.0118 - val_dice_coef: 0.3329

Epoch 00104: val_loss did not improve from 0.01126
Epoch 105/120
150/150 [==============================] - 88s 586ms/step - loss: 0.0082 - dice_coef: 0.4
124 - val_loss: 0.0202 - val_dice_coef: 0.2012

Epoch 00105: val_loss did not improve from 0.01126
Epoch 106/120
150/150 [==============================] - 90s 601ms/step - loss: 0.0079 - dice_coef: 0.4
335 - val_loss: 0.0134 - val_dice_coef: 0.2984

Epoch 00106: val_loss did not improve from 0.01126
Epoch 107/120
150/150 [==============================] - 88s 587ms/step - loss: 0.0078 - dice_coef: 0.4
347 - val_loss: 0.0209 - val_dice_coef: 0.1200

Epoch 00107: val_loss did not improve from 0.01126
Epoch 108/120
150/150 [==============================] - 91s 605ms/step - loss: 0.0083 - dice_coef: 0.4
112 - val_loss: 0.0101 - val_dice_coef: 0.4107

Epoch 00108: val_loss improved from 0.01126 to 0.01012, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 109/120
150/150 [==============================] - 88s 585ms/step - loss: 0.0082 - dice_coef: 0.4
142 - val_loss: 0.0142 - val_dice_coef: 0.2050

Epoch 00109: val_loss did not improve from 0.01012
Epoch 110/120
150/150 [==============================] - 91s 607ms/step - loss: 0.0082 - dice_coef: 0.4
126 - val_loss: 0.0145 - val_dice_coef: 0.2895

Epoch 00110: val_loss did not improve from 0.01012
Epoch 111/120
150/150 [==============================] - 88s 584ms/step - loss: 0.0080 - dice_coef: 0.4
142 - val_loss: 0.0098 - val_dice_coef: 0.4223

Epoch 00111: val_loss improved from 0.01012 to 0.00982, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 112/120
150/150 [==============================] - 90s 604ms/step - loss: 0.0080 - dice_coef: 0.4
218 - val_loss: 0.0118 - val_dice_coef: 0.2950

Epoch 00112: val_loss did not improve from 0.00982
Epoch 113/120
150/150 [==============================] - 88s 585ms/step - loss: 0.0079 - dice_coef: 0.4
012 - val_loss: 0.0182 - val_dice_coef: 0.0708

Epoch 00113: val_loss did not improve from 0.00982
Epoch 114/120
150/150 [==============================] - 90s 604ms/step - loss: 0.0080 - dice_coef: 0.4
186 - val_loss: 0.0179 - val_dice_coef: 0.2144

Epoch 00114: val_loss did not improve from 0.00982
Epoch 115/120
150/150 [==============================] - 93s 618ms/step - loss: 0.0077 - dice_coef: 0.4
208 - val_loss: 0.0138 - val_dice_coef: 0.2437

Epoch 00115: val_loss did not improve from 0.00982
Epoch 116/120
150/150 [==============================] - 87s 584ms/step - loss: 0.0078 - dice_coef: 0.4
118 - val_loss: 0.0158 - val_dice_coef: 0.1651

Epoch 00116: val_loss did not improve from 0.00982
Epoch 117/120
```

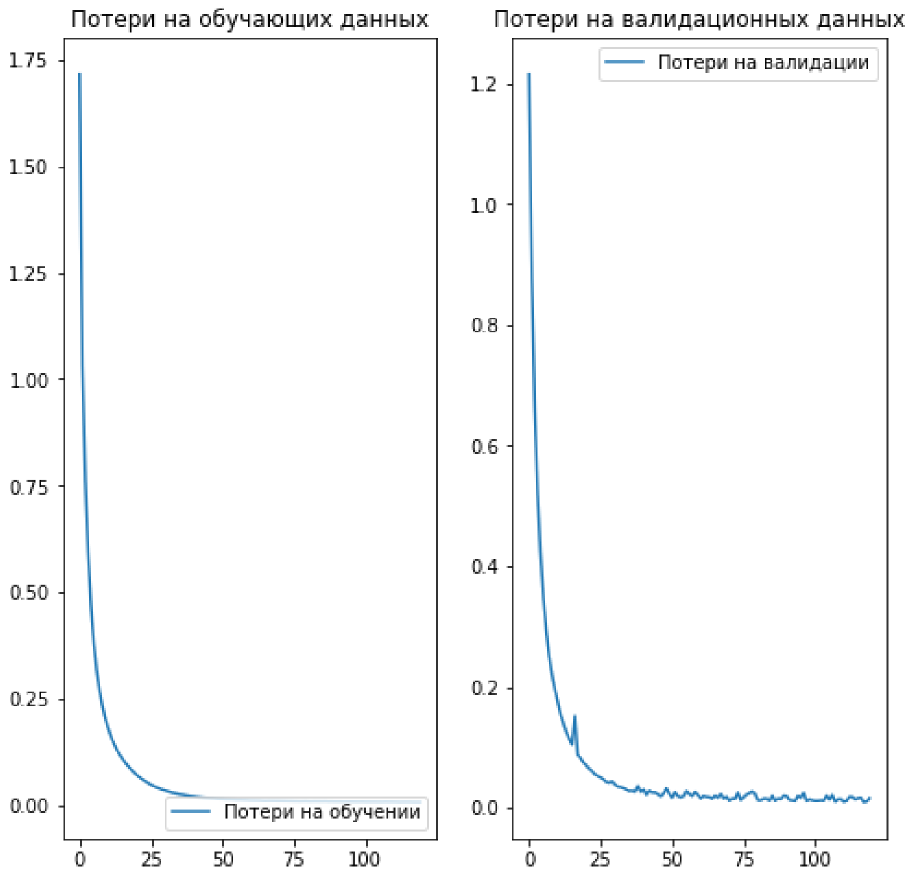
```
150/150 [==============================] - 88s 584ms/step - loss: 0.0078 - dice_coef: 0.4
191 - val_loss: 0.0166 - val_dice_coef: 0.1902

Epoch 00117: val_loss did not improve from 0.00982
Epoch 118/120
150/150 [==============================] - 94s 625ms/step - loss: 0.0073 - dice_coef: 0.4
351 - val_loss: 0.0093 - val_dice_coef: 0.4183

Epoch 00118: val_loss improved from 0.00982 to 0.00934, saving model to /kaggle/working/e
xp_1/model_weights_best.h5
Epoch 119/120
150/150 [==============================] - 88s 585ms/step - loss: 0.0075 - dice_coef: 0.4
308 - val_loss: 0.0103 - val_dice_coef: 0.3495

Epoch 00119: val_loss did not improve from 0.00934
Epoch 120/120
150/150 [==============================] - 88s 585ms/step - loss: 0.0075 - dice_coef: 0.4
309 - val_loss: 0.0156 - val_dice_coef: 0.2456

Epoch 00120: val_loss did not improve from 0.00934
training done.
```



In [15]:

```
model.unet.save_weights('/kaggle/working/my_model_weights2.h5')
```

In [17]:

```
batch_s = 4
stack_s = 3
downscale = True
output_path = prepare_experiment(Path('/kaggle/working'))
loaded_model = SuperTrackingModel(batch_s, stack_s, out_path=output_path, downscale=down
scale)
loaded_model.load()
```

```
Downloading...
From: https://drive.google.com/uc?id=1RHG_027l_TDKpEYwupjZnC0gUVVmSgLr
To: /kaggle/working/loaded_weights.h5
100%|██████████| 2.07M/2.07M [00:00<00:00, 92.0MB/s]
```

In [18]:

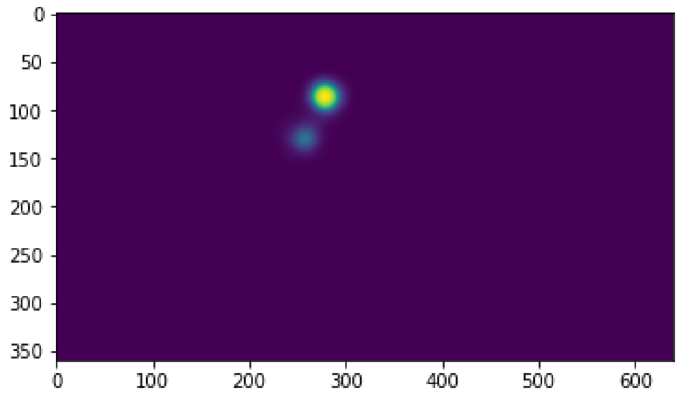
```python
sibatracc_final = loaded_model.test(Path('../input/tennistrackingassignment/test/'), [1, 2], do_visualization=True, test_name='test')
print(f'SiBaTrAcc final value: {sibatracc_final}')
```

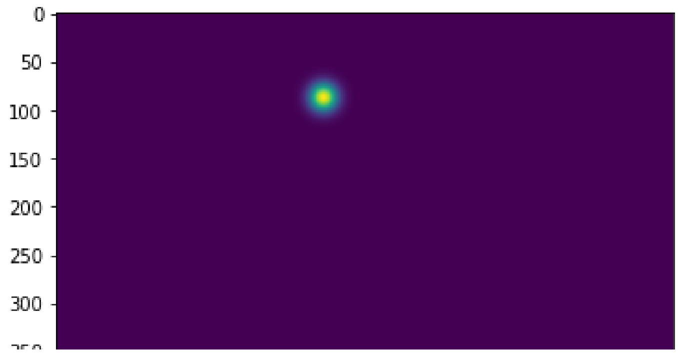
```
loading clip data (game 1, clip 1) downscaled
loading clip data (game 1, clip 1)
loading clip labels (game 1, clip 1)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 1, clip 2) downscaled
loading clip data (game 1, clip 2)
loading clip labels (game 1, clip 2)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 1, clip 3) downscaled
loading clip data (game 1, clip 3)
loading clip labels (game 1, clip 3)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 1, clip 4) downscaled
loading clip data (game 1, clip 4)
loading clip labels (game 1, clip 4)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 1, clip 5) downscaled
loading clip data (game 1, clip 5)
loading clip labels (game 1, clip 5)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 1, clip 6) downscaled
loading clip data (game 1, clip 6)
loading clip labels (game 1, clip 6)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 1, clip 7) downscaled
loading clip data (game 1, clip 7)
loading clip labels (game 1, clip 7)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 1, clip 8) downscaled
loading clip data (game 1, clip 8)
loading clip labels (game 1, clip 8)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 2, clip 1) downscaled
loading clip data (game 2, clip 1)
loading clip labels (game 2, clip 1)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 2, clip 2) downscaled
loading clip data (game 2, clip 2)
loading clip labels (game 2, clip 2)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 2, clip 3) downscaled
loading clip data (game 2, clip 3)
loading clip labels (game 2, clip 3)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 2, clip 4) downscaled
loading clip data (game 2, clip 4)
```

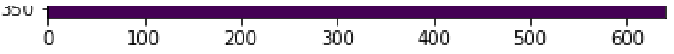
```
loading clip data (game 2, clip 4)
loading clip labels (game 2, clip 4)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 2, clip 5) downscaled
loading clip data (game 2, clip 5)
loading clip labels (game 2, clip 5)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 2, clip 6) downscaled
loading clip data (game 2, clip 6)
loading clip labels (game 2, clip 6)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 2, clip 7) downscaled
loading clip data (game 2, clip 7)
loading clip labels (game 2, clip 7)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 2, clip 8) downscaled
loading clip data (game 2, clip 8)
loading clip labels (game 2, clip 8)
doing predictions
predictions are made
perfoming clip visualization
loading clip data (game 2, clip 9) downscaled
loading clip data (game 2, clip 9)
loading clip labels (game 2, clip 9)
doing predictions
predictions are made
perfoming clip visualization
SiBaTrAcc final value: 0.6956884575440252
```

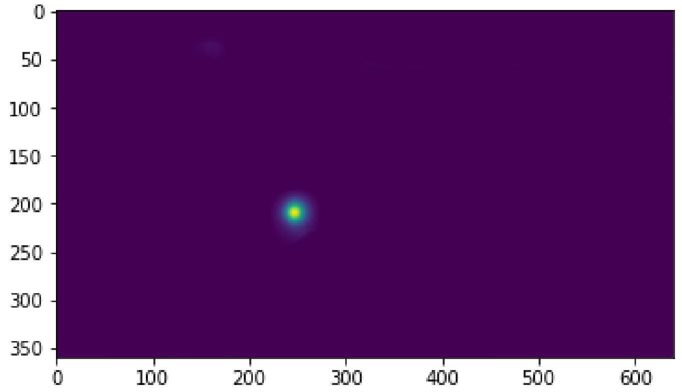
In [16]:

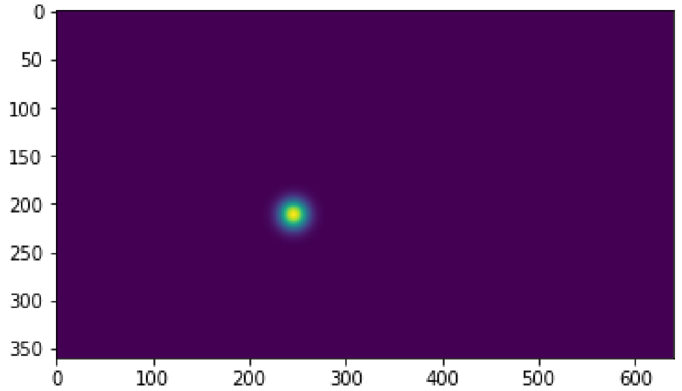
```python
imgs, masks = train_gen.get_random_batch(batch_s)
prediction = model.unet.predict(imgs)
for i in range(4):
    plt.figure()
    plt.imshow(prediction[i].reshape(360, 640))
    plt.figure()
    plt.imshow(masks[i])
```

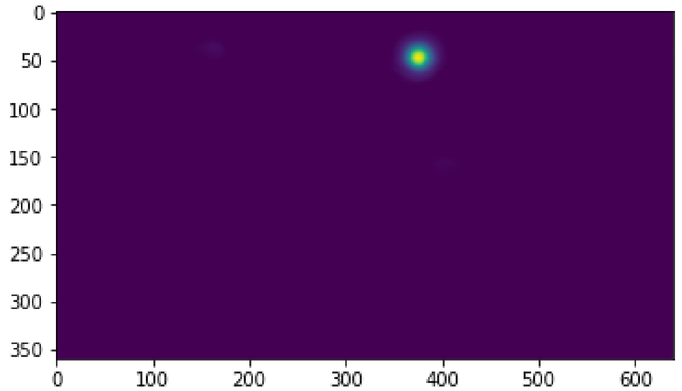


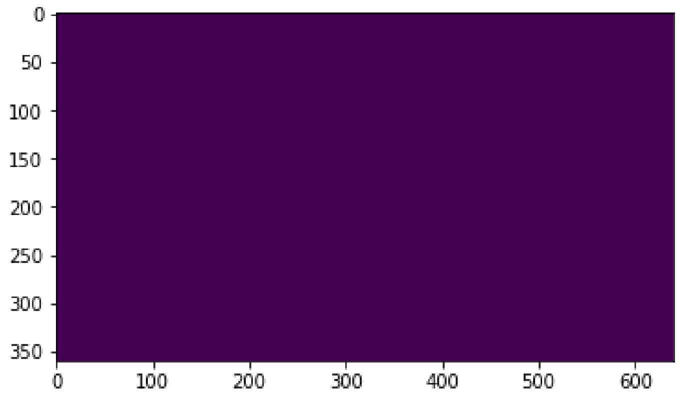


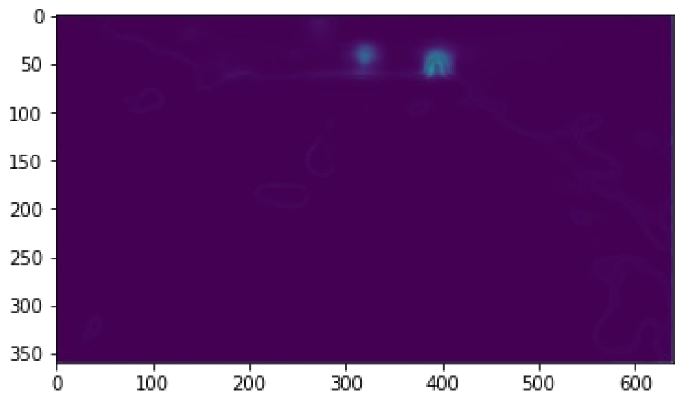

0 100 200 300 400 500 600

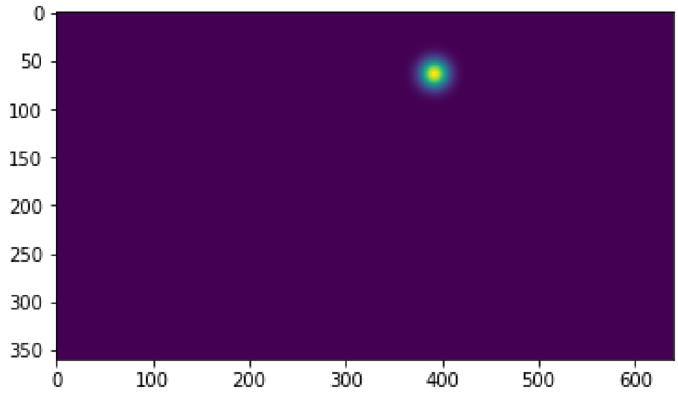

0 50 100 150 200 250 300 350
0 100 200 300 400 500 600


0 50 100 150 200 250 300 350
0 100 200 300 400 500 600


0 50 100 150 200 250 300 350
0 100 200 300 400 500 600

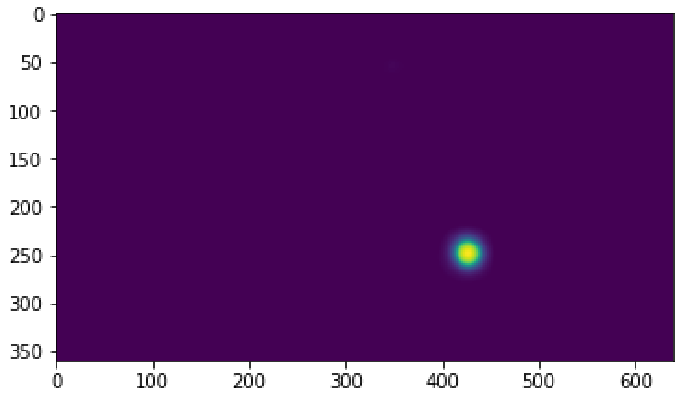

0 50 100 150 200 250 300 350
0 100 200 300 400 500 600

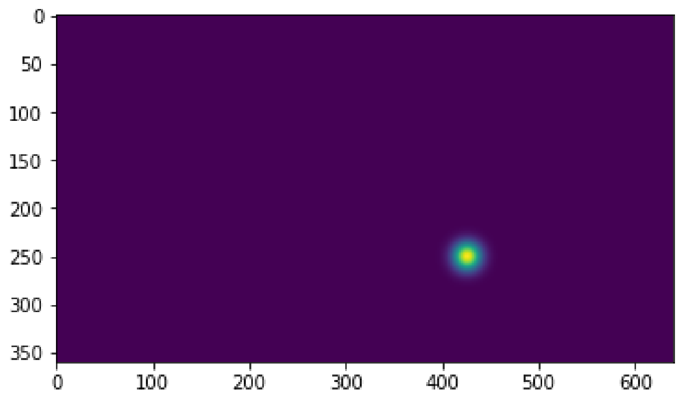

0 50 100 150 200 250 300 350
0 100 200 300 400 500 600

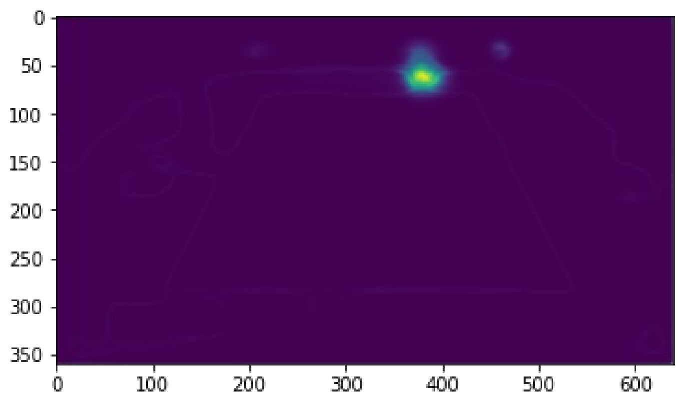


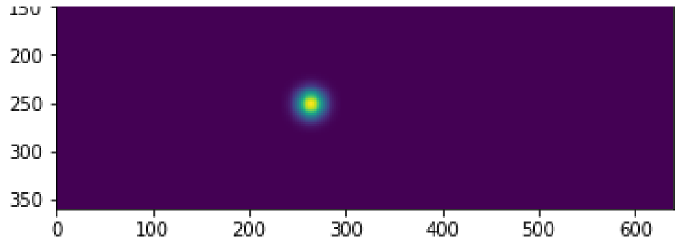
In [17]:

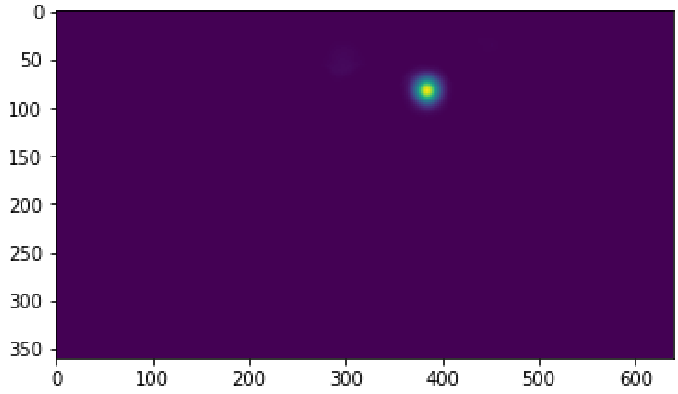
```
imgs, masks = val_gen.get_random_batch(batch_s)
prediction = model.unet.predict(imgs)
for i in range(4):
    plt.figure()
    plt.imshow(prediction[i].reshape(360, 640))
    plt.figure()
    plt.imshow(masks[i])
```

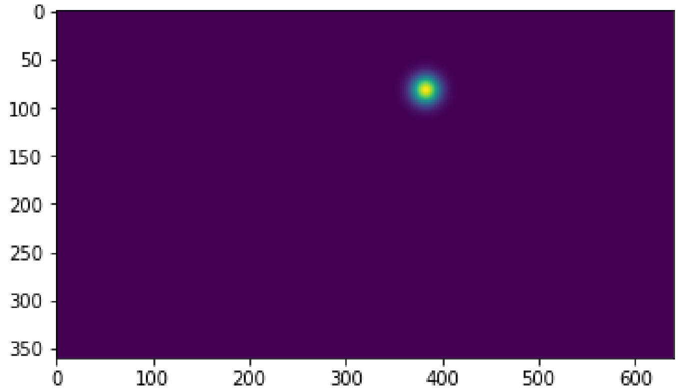


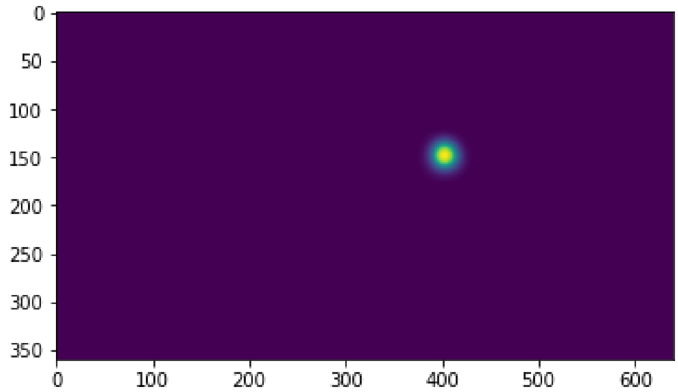


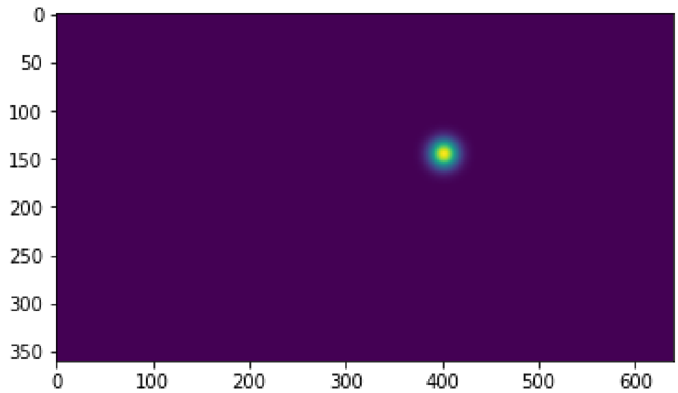












Пример пайплайна для обучения модели**:**

In [ ]:

```
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/tennistrackingassignment/train/'), [1, 2, 3, 4]
, stack_s=stack_s, downscale=True, pool_s=10, pool_update_s=4, quiet=True)
val_gen = DataGenerator(Path('../input/tennistrackingassignment/test/'), [1, 2], stack_s
=stack_s, downscale=True, pool_s=4, pool_update_s=2, quiet=True)

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

Пример пайплайна для тестирования обученной модели:

In [ ]:

```
new_model = SuperTrackingModel(batch_s, stack_s, out_path=output_path, downscale=downscal
e)
new_model.load()
sibatracc_final = new_model.test(Path('../input/tennistrackingassignment/test/'), [1,], d
o_visualization=True, test_name='test')
print(f'SiBaTrAcc final value: {sibatracc_final}')
```

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 [19]:

```
%cd /kaggle/working/
!zip -r "exp_2.zip" "exp_2"
from IPython.display import FileLink
FileLink(r'exp_2.zip')
```

```
/kaggle/working
  adding: exp_2/ (stored 0%)
  adding: exp_2/test_g2_c7_prob.mp4 (deflated 2%)
  adding: exp_2/test_g2_c4.mp4 (deflated 0%)
  adding: exp_2/test_g2_c9.mp4 (deflated 0%)
  adding: exp_2/test_g1_c3_prob.mp4 (deflated 2%)
  adding: exp_2/test_g1_c5_prob.mp4 (deflated 1%)
  adding: exp_2/test_g2_c2.txt (deflated 80%)
  adding: exp_2/test_g1_c5.txt (deflated 78%)
  adding: exp_2/test_g1_c1.mp4 (deflated 0%)
  adding: exp_2/test_g1_c7.txt (deflated 79%)
  adding: exp_2/test_g2_c5_prob.mp4 (deflated 1%)
  adding: exp_2/test_g1_c1_prob.mp4 (deflated 1%)
  adding: exp_2/test_g2_c6_prob.mp4 (deflated 1%)
  adding: exp_2/test_g1_c1.txt (deflated 78%)
  adding: exp_2/test_g2_c2.mp4 (deflated 0%)
  adding: exp_2/test_g2_c8_prob.mp4 (deflated 1%)
  adding: exp_2/test_g1_c2.txt (deflated 77%)
  adding: exp_2/test_g1_c2_prob.mp4 (deflated 1%)
  adding: exp_2/test_g2_c6.txt (deflated 78%)
  adding: exp_2/test_g1_c4_prob.mp4 (deflated 1%)
  adding: exp_2/test_g2_c8.txt (deflated 78%)
```

```
  adding: exp_2/test_g1_c7_prob.mp4 (deflated 1%)
  adding: exp_2/test_g1_c8.txt (deflated 80%)
  adding: exp_2/test_g2_c4.txt (deflated 80%)
  adding: exp_2/test_g1_c4.mp4 (deflated 0%)
  adding: exp_2/test_g2_c7.txt (deflated 79%)
  adding: exp_2/test_g1_c3.txt (deflated 73%)
  adding: exp_2/test_g2_c8.mp4 (deflated 0%)
  adding: exp_2/test_g1_c3.mp4 (deflated 0%)
  adding: exp_2/test_g2_c9_prob.mp4 (deflated 2%)
  adding: exp_2/test_g1_c5.mp4 (deflated 0%)
  adding: exp_2/test_g2_c5.mp4 (deflated 0%)
  adding: exp_2/test_g1_c2.mp4 (deflated 0%)
  adding: exp_2/test_g1_c6.txt (deflated 80%)
  adding: exp_2/test_g1_c6_prob.mp4 (deflated 1%)
  adding: exp_2/test_g2_c7.mp4 (deflated 0%)
  adding: exp_2/test_g2_c5.txt (deflated 80%)
  adding: exp_2/test_g2_c3.txt (deflated 80%)
  adding: exp_2/test_g2_c4_prob.mp4 (deflated 1%)
  adding: exp_2/test_g2_c1.txt (deflated 79%)
  adding: exp_2/test_g2_c3.mp4 (deflated 0%)
  adding: exp_2/test_g2_c1_prob.mp4 (deflated 2%)
  adding: exp_2/test_g2_c9.txt (deflated 80%)
  adding: exp_2/test_g2_c1.mp4 (deflated 0%)
  adding: exp_2/test_g1_c7.mp4 (deflated 0%)
  adding: exp_2/test_g1_c8.mp4 (deflated 0%)
  adding: exp_2/test_g1_c8_prob.mp4 (deflated 1%)
  adding: exp_2/test_g1_c6.mp4 (deflated 0%)
  adding: exp_2/test_g2_c6.mp4 (deflated 0%)
  adding: exp_2/test_g2_c2_prob.mp4 (deflated 1%)
  adding: exp_2/test_g1_c4.txt (deflated 75%)
  adding: exp_2/test_g2_c3_prob.mp4 (deflated 1%)
```

Out[19]:

exp_2.zip

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

In [ ]:

```
!rm -r /kaggle/working/exp_5
```

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 id** файла заключен между **...d/ b /view?...** и равен **1kZ8CC-zfkB_TlwtBjuPcEfsPV0Jz7IPA**

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)
```