リスト 1　学習済みモデルの読み込み

```
import tensorflow as tf
feature_extractor = tf.keras.applications.MobileNetV3Large(include_top=False)
```

リスト 2　DAVIS データセットから動画を一つ取得

```
import tensorflow_datasets as tfds
davis = tfds.load('davis', split='train')

data = davis.skip(3).take(1).get_single_element()
frames = data['video']['frames']
segmentations = data['video']['segmentations']
n_classes = tf.cast(tf.reduce_max(segmentations), tf.int32)
```

リスト 3　ラベルを表示

```
!pip install mediapy
import mediapy

COLOR_PALETTE = tf.constant([
    [0., 0., 0.], # background (unused)
    [1., 0., 0.], [0., 1., 0.], [0., 0., 1.],
    [0., 1., 1.], [1., 0., 1.], [1., 1., 0.],
])

@tf.function
def show_labels(image, labels):
    bg = tf.cast(image, labels.dtype) / 255.
    color_palette = COLOR_PALETTE[:labels.shape[-1]]
    fg = tf.tensordot(labels, color_palette, [[-1],[0]])
    bg_weight = labels[...,:1]
    return bg_weight * bg + (1. - bg_weight) * fg

labels = tf.one_hot(segmentations[0,:,:,0], n_classes+1)
mediapy.show_image(show_labels(frames[0], labels), width=400)
```

リスト 4　物体領域の特徴量を抽出

```
IMAGE_SIZE = [448,832]

@tf.function
def extract_features(images):
    images = tf.image.resize(images, IMAGE_SIZE)
    feat = feature_extractor(images)
    feat = tf.nn.l2_normalize(feat, axis=-1)
    return feat

```

```
ref_frame = frames[0]
ref_feat = extract_features(ref_frame[None])[0]
feat_size = ref_feat.shape[:2].as_list()

ref_seg = tf.cast(segmentations[0] == 3, ref_feat.dtype)
ref_seg = tf.image.resize(ref_seg, feat_size)
ref_seg = ref_seg[...,0] == 1.

pos_feat = tf.boolean_mask(ref_feat, ref_seg)
```

リスト 5　物体領域の特徴量を抽出

```
tgt_frame = frames[5]
tgt_feat = extract_features(tgt_frame[None])[0]

pos_score = tf.tensordot(tgt_feat, pos_feat, [[-1],[-1]])
pos_score = tf.reduce_max(pos_score, axis=-1)

pos_score = tf.image.resize(pos_score[...,None], ref_frame.shape[:2], method='
    nearest')
mediapy.show_image(tf.where(pos_score > 0.5, [[[0., 0., 1.]]], tgt_frame/255),
    width=400)
```

リスト 6　物体領域の特徴量を抽出

```
neg_feat = tf.boolean_mask(ref_feat, tf.logical_not(ref_seg))
neg_score = tf.tensordot(tgt_feat, neg_feat, [[-1], [-1]])
neg_score = tf.reduce_max(neg_score, axis=-1)

neg_score = tf.image.resize(neg_score[...,None], ref_frame.shape[:2], method='
    nearest')
mediapy.show_image(tf.where(pos_score > neg_score, [[[0., 0., 1.]]], tgt_frame
    /255), width=400)
```

リスト 7　物体領域の特徴量を抽出

```
@tf.function
def copy_labels(ref_feat, tgt_feat, ref_labels, temperature=0.05):
    ref_feat_flat = tf.reshape(ref_feat, [-1, ref_feat.shape[-1]])
    tgt_feat_flat = tf.reshape(tgt_feat, [-1, tgt_feat.shape[-1]])
    ref_labels_flat = tf.reshape(ref_labels, [-1, ref_labels.shape[-1]])

    inner = tf.matmul(tgt_feat_flat, ref_feat_flat, transpose_b=True)
    weights = tf.nn.softmax(inner/temperature, axis=1)
    tgt_labels_flat = tf.matmul(weights, ref_labels_flat)
    tgt_labels = tf.reshape(tgt_labels_flat, [-1, *ref_labels.shape[1:]])
    return tgt_labels
```

```

ref_frames = frames[:2]
ref_feat = extract_features(ref_frames)

ref_labels = tf.one_hot(segmentations[:2,:,:,0], n_classes+1)
ref_labels = tf.image.resize(ref_labels, feat_size)

tgt_frames = frames[6:7]
tgt_feat = extract_features(tgt_frames)

tgt_labels = copy_labels(ref_feat, tgt_feat, ref_labels)
tgt_labels = tf.image.resize(tgt_labels, tgt_frames.shape[1:3], method='nearest')
mediapy.show_images(show_labels(tgt_frames, tgt_labels), width=400)
```

リスト 8 いくつかのフレームをキーフレーム（正解ラベルを与えるフレーム）として，動画全体にラベル付する

```
VIDEO_SIZE = [224,416]

ref_inds = [0, 40, 80]
ref_frames = tf.gather(frames, ref_inds)
ref_feat = extract_features(ref_frames)
ref_seg = tf.gather(segmentations, ref_inds)
ref_labels = tf.one_hot(ref_seg[...,0], n_classes+1)
ref_labels = tf.image.resize(ref_labels, feat_size)

result = []
for t in range(len(frames)):
    tgt_frames_t = frames[t:t+1]
    tgt_feat_t = extract_features(tgt_frames_t)
    tgt_labels_t = copy_labels(ref_feat, tgt_feat_t, ref_labels, temperature
        =0.02)

    ref_feat = tf.concat([ref_feat[:len(ref_inds)], tgt_feat_t], axis=0)
    ref_labels = tf.concat([ref_labels[:len(ref_inds)], tgt_labels_t], axis=0)

    result.append(
        show_labels(
            tf.image.resize(tgt_frames_t, VIDEO_SIZE),
            tf.image.resize(tgt_labels_t, VIDEO_SIZE, method='nearest')))

result = tf.concat(result, axis=0)
mediapy.show_video(result.numpy(), fps=5)
```

リスト 9 バウンディングボックスを描画する関数

```
def draw_bounding_boxes(image, boxes, colors=COLOR_PALETTE[1:]):
    size = image.shape[:2]
    # 座標値が 0-1 の範囲になるように規格化
    boxes /= tf.cast(tf.concat([size, size], axis=0), tf.float32)
    # サイズが 1/4 の画像にバウンディングボックスだけを描く
    box_image = tf.image.draw_bounding_boxes(
        tf.zeros([1, size[0]//4, size[1]//4, 3]),
        boxes[None], colors)[0]
    # サイズを 4 倍（元のサイズ）にする
    box_image = tf.image.resize(box_image, size)
    # 元の画像と合成する
    weight = tf.reduce_max(box_image, axis=-1, keepdims=True)
    return image * (1. - weight) + box_image * weight
```

リスト 10　推定されたクラス 5（右の人物）の領域のバウンディングボックスを求める

```
segment = tf.argmax(tgt_labels[0], axis=-1) == 5
points = tf.where(segment)
tgt_frame = tf.image.resize(tgt_frames[0], IMAGE_SIZE) / 255.
top_left = tf.reduce_min(points, axis=0)
bottom_right = tf.reduce_max(points, axis=0) + 1
box = tf.concat([top_left, bottom_right], axis=0)

box_image = draw_bounding_boxes(tgt_frame, tf.cast(box[None], tf.float32) * 32)
mediapy.show_image(box_image, width=400)
```

リスト 11　推定された領域からバウンディングボックスをサンプリングする

```
inds = tf.range(points.shape[0])

sampled_points = []
for _ in tf.range(10):
    shuffled_inds = tf.random.shuffle(inds)
    _points = tf.gather(points, shuffled_inds[:5])
    sampled_points.append(_points)
sampled_points = tf.stack(sampled_points, axis=0)

top_left = tf.reduce_min(sampled_points, axis=1)
bottom_right = tf.reduce_max(sampled_points, axis=1) + 1
boxes = tf.concat([top_left, bottom_right], axis=1)

box_image = draw_bounding_boxes(tgt_frame, tf.cast(boxes, tf.float32) * 32)
mediapy.show_image(box_image, width=400)
```

リスト 12　サンプリングした矩形の中から元の領域との重なり具合（IoU）が大きいものを選択

```
def points_in_boxes(points, boxes):
    in_box_tl = tf.reduce_all(points[:,None,:] >= boxes[None,:,:2], axis=2)
    in_box_br = tf.reduce_all(points[:,None,:] < boxes[None,:,2:], axis=2)
    in_box = tf.logical_and(in_box_tl, in_box_br)  # [n_points, n_boxes]
    return in_box

in_box = points_in_boxes(points, boxes)
n_points_in_box = tf.reduce_sum(tf.cast(in_box, tf.float32), axis=0)
n_points = tf.cast(points.shape[0], tf.float32)
areas = tf.cast(tf.reduce_prod(boxes[:,2:] - boxes[:,:2], axis=1), tf.float32)

union = areas + n_points - n_points_in_box
iou = n_points_in_box / union
good_boxes = tf.boolean_mask(boxes, iou > 0.5)

box_image = draw_bounding_boxes(tgt_frame, tf.cast(good_boxes, tf.float32) * 32)
mediapy.show_image(box_image, width=400)
```

リスト 13　選択した矩形全てを包含する矩形をバウンディングボックスとする

```
box = tf.concat([
    tf.reduce_min(good_boxes[:,:2], axis=0),
    tf.reduce_max(good_boxes[:,2:], axis=0)
], axis=0)

box_image = draw_bounding_boxes(tgt_frame, tf.cast(box[None], tf.float32) * 32)
mediapy.show_image(box_image, width=400)
```

リスト 14　物体検出モデルをダウンロード

```
!pip install tensorflow-hub
import tensorflow_hub as hub
model_url = 'https://tfhub.dev/tensorflow/centernet/resnet101v1_fpn_512x512/1'
detector = hub.load(model_url).signatures['serving_default']
```

リスト 15　DAVIS データセットから別の動画を取得

```
frames = None
for data in davis:
    if data['metadata']['video_name'] == 'lindy-hop':
        frames = data['video']['frames']

frames = tf.cast(tf.image.resize(frames, IMAGE_SIZE), tf.uint8)
```

リスト 16　物体検出モデルを用いて画像から人物のバウンディングボックスを抽出

```
def detect_persons(image, threshold=0.6):
    outputs = detector(image[None])
    cls = outputs['detection_classes'][0]
    scores = outputs['detection_scores'][0]
    boxes = outputs['detection_boxes'][0]

    is_person = tf.logical_and(cls == 1, scores > threshold)
    person_boxes = tf.boolean_mask(boxes, is_person)
    return person_boxes

frame1 = frames[10]
frame2 = frames[15,:,::-1]

person_boxes1 = detect_persons(frame1)
person_boxes2 = detect_persons(frame2)
```

リスト 17　検出した人物のバウンディングボックスを表示

```
# バウンディングボックスの値を [0, 1] から [0, H or W] にするための乗数
size = tf.cast(IMAGE_SIZE, tf.float32)
box_scale = tf.concat([size, size], axis=0)

box_image1 = draw_bounding_boxes(frame1/255, person_boxes1 * box_scale)
box_image2 = draw_bounding_boxes(frame2/255, person_boxes2 * box_scale)
mediapy.show_images([box_image1, box_image2], width=400)
```

リスト 18　バウンディングボックスをラベルに変換

```
def boxes_to_labels(boxes, size):
    H, W = size
    grid = tf.stack(tf.meshgrid(tf.range(H), tf.range(W), indexing='ij'), axis
        =-1)
    grid = tf.cast(grid, tf.float32)
    boxes = boxes * tf.cast([H,W,H,W], boxes.dtype)

    tl = tf.reduce_all(grid[None,:,:] >= boxes[:,:2][:,None,None], axis=-1)
    br = tf.reduce_all(grid[None,:,:] < (boxes[:,2:]+1.)[:,None,None], axis=-1)
    labels = tf.cast(tf.logical_and(tl, br), tf.float32)
    labels = tf.transpose(labels, [1,2,0])
    return labels

labels1 = boxes_to_labels(person_boxes1, [14,26])
labels2 = boxes_to_labels(person_boxes2, [14,26])

# 各ラベルに色を割り当てて図示する
labels1_vis = tf.concat([1-tf.reduce_max(labels1, axis=-1, keepdims=True),
    labels1], axis=-1)
labels1_vis = tf.image.resize(labels1_vis, IMAGE_SIZE, method='nearest')
mediapy.show_image(show_labels(frame1, labels1_vis), width=400)
```

リスト 19　特徴量の類似度に基づいてラベルを画像 1 から画像 2 に転移

```
feat1 = extract_features(frame1[None])[0]
feat2 = extract_features(frame2[None])[0]
labels1to2 = copy_labels(feat1, feat2, labels1, temperature=0.05)

# 各ラベルに色を割り当てて図示する
labels1to2_vis = tf.concat([1-tf.reduce_max(labels1to2, axis=-1, keepdims=True),
    labels1to2], axis=-1)
labels1to2_vis = tf.image.resize(labels1to2_vis, IMAGE_SIZE, method='nearest')
label_image = show_labels(frame2, labels1to2_vis)
label_image = draw_bounding_boxes(label_image, person_boxes2 * box_scale)
mediapy.show_image(label_image, width=400)
```

リスト 20　画像 1 から画像 2 へ転移したラベルと，画像 2 のバウンディングボックスとの重なり具合（IoU）を計算

```
intersections = tf.einsum('ijk,ijl->kl', labels1to2, labels2)
areas1 = tf.reduce_sum(labels1to2, axis=[0,1])
areas2 = tf.reduce_sum(labels2, axis=[0,1])
unions = areas1[:,None] + areas2[None,:] - intersections
ious = intersections / unions

# matplotlib で IoU の組み合わせを図示
import matplotlib.pyplot as plt
plt.imshow(ious)
plt.ylabel('Label from frame1')
plt.xlabel('Box on frame2')
plt.colorbar()
```

リスト 21　IoU の合計が最大になるようにペアを作る

```
from scipy.optimize import linear_sum_assignment
inds1, inds2 = linear_sum_assignment(ious.numpy(), maximize=True)

# IoU が小さいペアを除く
comb_ious = tf.gather_nd(ious, tf.stack([inds1, inds2], axis=1))
comb_valid = comb_ious > 0.15
inds1 = tf.boolean_mask(inds1, comb_valid)
inds2 = tf.boolean_mask(inds2, comb_valid)

person_boxes1_sorted = tf.gather(person_boxes1, inds1)
person_boxes2_sorted = tf.gather(person_boxes2, inds2)

mediapy.show_image(draw_bounding_boxes(frame1/255, person_boxes1_sorted * tf.
    concat([size, size], axis=0)), width=400)
mediapy.show_image(draw_bounding_boxes(frame2/255, person_boxes2_sorted * tf.
    concat([size, size], axis=0)), width=400)
```