```
pip install mxnet
```

```
import matplotlib.pyplot as plt
import numpy as np
import mxnet as mx
from mxnet import gluon, nd, image
from mxnet.gluon.data.vision import transforms
from gluoncv.data.transforms import video
from gluoncv import utils
from gluoncv.model_zoo import get_model
```

```
url = 'https://github.com/bryanyzhu/tiny-ucf101/raw/master/ThrowDiscus.png'
im_fname = utils.download(url)

img = image.imread(im_fname)

plt.imshow(img.asnumpy())
plt.show()
```

```
transform_fn = transforms.Compose([
    video.VideoCenterCrop(size=224),
    video.VideoToTensor(),
    video.VideoNormalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
```

```
img_list = transform_fn([img.asnumpy()])
plt.imshow(np.transpose(img_list[0], (1,2,0)))
plt.show()
```

```
net = get_model('vgg16_ucf101', nclass=101, pretrained=True)
```

```
pred = net(nd.array(img_list[0]).expand_dims(axis=0))

classes = net.classes
```

```
classes = net.classes
topK = 5
ind = nd.topk(pred, k=topK)[0].astype('int')
print('The input video frame is classified to be')
for i in range(topK):
    print('\t[%s], with probability %.3f.'%
          (classes[ind[i].asscalar()], nd.softmax(pred)[0][ind[i]].asscalar()))
```

```
from gluoncv.utils import try_import_cv2
cv2 = try_import_cv2()

url = 'https://github.com/bryanyzhu/tiny-ucf101/raw/master/v_Basketball_g01_c01.avi'
video_fname = utils.download(url)

cap = cv2.VideoCapture(video_fname)
cnt = 0
video_frames = []
while(cap.isOpened()):
    ret, frame = cap.read()
    cnt += 1
    if ret and cnt % 25 == 0:
        video_frames.append(frame)
    if not ret: break

cap.release()
print('We evenly extract %d frames from the video %s.' % (len(video_frames), video_fname))
```

```
video_frames_transformed = transform_fn(video_frames)
final_pred = 0
for _, frame_img in enumerate(video_frames_transformed):
    pred = net(nd.array(frame_img).expand_dims(axis=0))
    final_pred += pred
final_pred /= len(video_frames)

classes = net.classes
topK = 5
ind = nd.topk(final_pred, k=topK)[0].astype('int')
print('The input video is classified to be')
```

```
print('The input video is classified to be')
for i in range(topK):
    print('\t[%s], with probability %.3f.'%
          (classes[ind[i].asscalar()], nd.softmax(final_pred)[0][ind[i]].asscalar()))
```

```
!pip install decord
```

```
Collecting decord
  Downloading https://files.pythonhosted.org/packages/c0/21/e077da18292be83f653d9a3f5b63c00712a917dc6ac6e15c044ac7319
     |████████████████████████████████| 12.0MB 345kB/s
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.6/dist-packages (from decord) (1.18.2)
Installing collected packages: decord
Successfully installed decord-0.3.6
```

```
from gluoncv import utils
url = 'https://github.com/bryanyzhu/tiny-ucf101/raw/master/abseiling_k400.mp4'
video_fname = utils.download(url)

from decord import VideoReader
vr = VideoReader(video_fname)
```

```
vr = VideoReader(video_fname, width=320, height=256)
```

```
duration = len(vr)
print('The video contains %d frames' % duration)
```

```
frame = vr[9]
print(frame.shape)
```

```
frame_id_list = range(0, 64, 2)
frames = vr.get_batch(frame_id_list).asnumpy()
print(frames.shape)
```

```
key_indices = vr.get_key_indices()
```

```
key_frames = vr.get_batch(key_indices)
print(key_frames.shape)
```

```
import cv2
import time
import numpy as np

frames_list = np.arange(duration)
np.random.shuffle(frames_list)

# Decord
for i in range(11):
    if i == 1:
        start_time = time.time()
    decord_vr = VideoReader(video_fname)
    frames = decord_vr.get_batch(frames_list)
end_time = time.time()
print('Decord takes %4.4f seconds.' % ((end_time - start_time)/10))

# OpenCV
for i in range(11):
    if i == 1:
        start_time = time.time()
    cv2_vr = cv2.VideoCapture(video_fname)
    for frame_idx in frames_list:
        cv2_vr.set(1, frame_idx)
        _, frame = cv2_vr.read()
    cv2_vr.release()
end_time = time.time()
print('OpenCV takes %4.4f seconds.' % ((end_time - start_time)/10))
```

```
1
```

```
!pip install gluoncv
```

```
Collecting gluoncv
  Downloading https://files.pythonhosted.org/packages/69/4d/d9d6b9261af8f7251977bb97be669a3908f72bdec9d3597e527712d38
     |████████████████████████████████| 696kB 4.9MB/s
Collecting portalocker
  Downloading https://files.pythonhosted.org/packages/91/db/7bc703c0760df726839e0699b7f78a4d8217fdc9c7fcb1b51b39c5a22
Requirement already satisfied: Pillow in /usr/local/lib/python3.6/dist-packages (from gluoncv) (7.0.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from gluoncv) (3.2.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from gluoncv) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from gluoncv) (1.18.2)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from gluoncv) (2.21.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from gluoncv) (4.38.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib->gluoncv) (0.1
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->gluoncv)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->gluon
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests->gluoncv)
Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests->gluoncv) (2.8
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests->gluoncv
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests->gluoncv
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from cycler>=0.10->matplotlib->gluoncv)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from kiwisolver>=1.0.1->matplotli
Installing collected packages: portalocker, gluoncv
Successfully installed gluoncv-0.6.0 portalocker-1.5.2
```

```
import matplotlib.pyplot as plt
import numpy as np
import mxnet as mx
from mxnet import gluon, nd, image
from mxnet.gluon.data.vision import transforms
from gluoncv.data.transforms import video
from gluoncv import utils
from gluoncv.model_zoo import get_model
from gluoncv.utils.filesystem import try_import_decord
```

```
from gluoncv.utils.filesystem import try_import_decord
decord = try_import_decord()
```

```
url = 'https://github.com/bryanyzhu/tiny-ucf101/raw/master/abseiling_k400.mp4'
video_fname = utils.download(url)
vr = decord.VideoReader(video_fname)
frame_id_list = range(0, 64, 2)
video_data = vr.get_batch(frame_id_list).asnumpy()
clip_input = [video_data[vid, :, :, :] for vid, _ in enumerate(frame_id_list)]
```

```
Downloading abseiling_k400.mp4 from https://github.com/bryanyzhu/tiny-ucf101/raw/master/abseiling_k400.mp4...
100%|██████████| 782/782 [00:00<00:00, 11292.21KB/s]
```

```
transform_fn = video.VideoGroupValTransform(size=224, mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
clip_input = transform_fn(clip_input)
clip_input = np.stack(clip_input, axis=0)
clip_input = clip_input.reshape((-1,) + (32, 3, 224, 224))
clip_input = np.transpose(clip_input, (0, 2, 1, 3, 4))
print('Video data is downloaded and preprocessed.')

```

```
Video data is downloaded and preprocessed.
```

```
model_name = 'i3d_inceptionv1_kinetics400'
net = get_model(model_name, nclass=400, pretrained=True)
print('%s model is successfully loaded.' % model_name)
```

```
Downloading /root/.mxnet/models/i3d_inceptionv1_kinetics400-81e0be10.zip from https://apache-mxnet.s3-accelerate.dual
51278KB [00:01, 38384.03KB/s]
i3d_inceptionv1_kinetics400 model is successfully loaded.
```

```
pred = net(nd.array(clip_input))

classes = net.classes
topK = 5
ind = nd.topk(pred, k=topK)[0].astype('int')
print('The input video clip is classified to be')
for i in range(topK):
    print('\t[%s], with probability %.3f.'%
```

```
    print('\t[%s], with probability %.3f.' %
          (classes[ind[i].asscalar()], nd.softmax(pred)[0][ind[i]].asscalar()))
```

```
The input video clip is classified to be
        [abseiling], with probability 0.991.
        [rock_climbing], with probability 0.009.
        [ice_climbing], with probability 0.000.
        [paragliding], with probability 0.000.
        [skydiving], with probability 0.000.
```

```
import matplotlib.pyplot as plt
import numpy as np
import mxnet as mx
from mxnet import gluon, nd, image
from mxnet.gluon.data.vision import transforms
from gluoncv.data.transforms import video
from gluoncv import utils
from gluoncv.model_zoo import get_model
```

Double-click (or enter) to edit

```
from gluoncv.utils.filesystem import try_import_decord
decord = try_import_decord()

url = 'https://github.com/bryanyzhu/tiny-ucf101/raw/master/abseiling_k400.mp4'
video_fname = utils.download(url)
vr = decord.VideoReader(video_fname)
fast_frame_id_list = range(0, 64, 2)
slow_frame_id_list = range(0, 64, 16)
frame_id_list = list(fast_frame_id_list) + list(slow_frame_id_list)
video_data = vr.get_batch(frame_id_list).asnumpy()
clip_input = [video_data[vid, :, :, :] for vid, _ in enumerate(frame_id_list)]
```

```
transform_fn = video.VideoGroupValTransform(size=224, mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
clip_input = transform_fn(clip_input)
clip_input = np.stack(clip_input, axis=0)
clip_input = clip_input.reshape((-1,) + (36, 3, 224, 224))
```

```
clip_input = np.transpose(clip_input, (0, 2, 1, 3, 4))
print('Video data is downloaded and preprocessed.')
```

```
Video data is downloaded and preprocessed.
```

```
model_name = 'slowfast_4x16_resnet50_kinetics400'
net = get_model(model_name, nclass=400, pretrained=True)
print('%s model is successfully loaded.' % model_name)
```

```
Downloading /root/.mxnet/models/slowfast_4x16_resnet50_kinetics400-9d650f51.zip from https://apache-mxnet.s3-accelera
100%|██████████| 134964/134964 [00:03<00:00, 43538.85KB/s]
slowfast_4x16_resnet50_kinetics400 model is successfully loaded.
```

```
pred = net(nd.array(clip_input))

classes = net.classes
topK = 5
ind = nd.topk(pred, k=topK)[0].astype('int')
print('The input video clip is classified to be')
for i in range(topK):
    print('\t[%s], with probability %.3f.'%
          (classes[ind[i].asscalar()], nd.softmax(pred)[0][ind[i]].asscalar()))
```

```
The input video clip is classified to be
        [abseiling], with probability 0.996.
        [rock_climbing], with probability 0.004.
        [ice_climbing], with probability 0.000.
        [paragliding], with probability 0.000.
        [climbing_a_rope], with probability 0.000.
```