1 Example of Moments in Time Model Inference

The following loads and parses a video from the "Skiing" class of the UCF101 dataset. This notebook shows a minimum example of inference on this video which the model has not previously seen.

View and (optionally) modify the TODOs below to try your own video on a selection of these models.

1.1 Setup

1.2 Get UCF101 video dataset

Go to https://www.crcv.ucf.edu/data/UCF101.php (https://www.crcv.ucf.edu/data/UCF101.php) and download the dataset. Locate teh 'Skiing' class and choose a video.

1.2.1 Import Packages and Methods

In [1]:
```
import os
import numpy as np
import re
import sys
import glob
import argparse
import functools
import subprocess
from PIL import Image
import time
import torch
from torchvision import transforms as trn
import h5py
import shutil
from tensorflow.python.keras.utils import to_categorical
```

```
/state/partition1/llgrid/pkg/anaconda/anaconda3-2020a/lib/python3.6/si
te-packages/tensorflow/python/framework/dtypes.py:516: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a
future version of numpy, it will be understood as (type, (1,)) / '(1,)
type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/state/partition1/llgrid/pkg/anaconda/anaconda3-2020a/lib/python3.6/si
te-packages/tensorflow/python/framework/dtypes.py:517: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a
future version of numpy, it will be understood as (type, (1,)) / '(1,)
type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/state/partition1/llgrid/pkg/anaconda/anaconda3-2020a/lib/python3.6/si
te-packages/tensorflow/python/framework/dtypes.py:518: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a
future version of numpy, it will be understood as (type, (1,)) / '(1,)
type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/state/partition1/llgrid/pkg/anaconda/anaconda3-2020a/lib/python3.6/si
te-packages/tensorflow/python/framework/dtypes.py:519: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a
future version of numpy, it will be understood as (type, (1,)) / '(1,)
type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/state/partition1/llgrid/pkg/anaconda/anaconda3-2020a/lib/python3.6/si
te-packages/tensorflow/python/framework/dtypes.py:520: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a
future version of numpy, it will be understood as (type, (1,)) / '(1,)
type'.
WARNING:tensorflow:From /state/partition1/llgrid/pkg/anaconda/anaconda
3-2020a/lib/python3.6/site-packages/horovod/tensorflow/__init__.py:11
7: The name tf.global_variables is deprecated. Please use tf.compat.v
1.global_variables instead.

WARNING:tensorflow:From /state/partition1/llgrid/pkg/anaconda/anaconda
3-2020a/lib/python3.6/site-packages/horovod/tensorflow/__init__.py:14
3: The name tf.get_default_graph is deprecated. Please use tf.compat.v
1.get_default_graph instead.


/state/partition1/llgrid/pkg/anaconda/anaconda3-2020a/lib/python3.6/si
te-packages/tensorboard/compat/tensorflow_stub/dtypes.py:541: FutureWa
rning: Passing (type, 1) or '1type' as a synonym of type is deprecate
d; in a future version of numpy, it will be understood as (type, (1,))
/ '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/state/partition1/llgrid/pkg/anaconda/anaconda3-2020a/lib/python3.6/si
te-packages/tensorboard/compat/tensorflow_stub/dtypes.py:542: FutureWa
rning: Passing (type, 1) or '1type' as a synonym of type is deprecate
d; in a future version of numpy, it will be understood as (type, (1,))
/ '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/state/partition1/llgrid/pkg/anaconda/anaconda3-2020a/lib/python3.6/si
te-packages/tensorboard/compat/tensorflow_stub/dtypes.py:543: FutureWa
rning: Passing (type, 1) or '1type' as a synonym of type is deprecate
d; in a future version of numpy, it will be understood as (type, (1,))
```

```
/ '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/state/partition1/llgrid/pkg/anaconda/anaconda3-2020a/lib/python3.6/si
te-packages/tensorboard/compat/tensorflow_stub/dtypes.py:544: FutureWa
rning: Passing (type, 1) or '1type' as a synonym of type is deprecate
d; in a future version of numpy, it will be understood as (type, (1,))
/ '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/state/partition1/llgrid/pkg/anaconda/anaconda3-2020a/lib/python3.6/si
te-packages/tensorboard/compat/tensorflow_stub/dtypes.py:545: FutureWa
rning: Passing (type, 1) or '1type' as a synonym of type is deprecate
d; in a future version of numpy, it will be understood as (type, (1,))
/ '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/state/partition1/llgrid/pkg/anaconda/anaconda3-2020a/lib/python3.6/si
te-packages/tensorboard/compat/tensorflow_stub/dtypes.py:550: FutureWa
rning: Passing (type, 1) or '1type' as a synonym of type is deprecate
d; in a future version of numpy, it will be understood as (type, (1,))
/ '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

1.2.2 Enable Model File Reading

In [2]:
```
1 os.putenv("HDF5_USE_FILE_LOCKING", "FALSE")
```

Out[2]: 0

1.2.3 Create Moments in Time One-Hot Class Labels

In [3]:
```
1  set_dir = '/home/gridsan/groups/Moments_in_Time/data-copy/data/parsed/
2  # Read folders in set directory. Folders should correspond to classes
3  categories = os.listdir(set_dir)
4  # Remove any extraneous .ipynb files
5  new_categories = []
6  for cat in categories:
7      if '.ipynb' not in cat:
8          new_categories.append(cat)
9  categories = new_categories
10 # Create one-hot vector labels
11 categories_labels = dict()
12 for i in range(len(categories)):
13     categories_labels[categories[i]] = to_categorical(np.array(i), num
```

```
339 total classes
```

1.3 Parse Video

1.3.1 Specify Input

In [4]:
```
1 # TODO:
```

```
# Specify input video and corresponding ground truth Moments in Time c
video_file = 'example_video.avi' ### TODO: add your video file from th
ucf101_class_gt = 'Skiing'
```

1.3.2 Load and Parse

In [5]:

```
def read_as_list(filename):
    """Read the file at filename and store its contents into a list.""
    with open(filename) as f:
        result = [line.rstrip() for line in f.readlines()]
        f.close()
        return result

def load_frames(frame_paths, num_frames):
    """Loads frames from file and returns a list of the frames."""
    #print('Running load_frames')
    frames = [Image.open(frame).convert('RGB') for frame in frame_path
    if len(frames) >= num_frames:
        return frames[::int(np.ceil(len(frames) / float(num_frames)))]
    else:
        raise ValueError('Video must have at least {} frames'.format(n

def extract_frames(video_file, framesFolder, framerate):
    """Takes a video and converts it into a list of frames."""
    if os.path.exists(framesFolder):
        subprocess.call(['rm', '-rf', framesFolder + '*.jpg'])
    try:
        os.makedirs(os.path.join(framesFolder))
    except OSError:
        pass
    output = subprocess.Popen(['ffmpeg', '-i', video_file], stderr=sub
    # Search and parse 'Duration: 00:05:24.13,' from ffmpeg stderr
    re_duration = re.compile('Duration: (.*?)\.')
    duration = re_duration.search(str(output[1])).groups()[0]
    seconds = functools.reduce(lambda x, y: x * 60 + y,
                               map(int, duration.split(':')))
    #rate = num_frames / float(seconds)
    rate = framerate
    num_frames = seconds * framerate
    output = subprocess.Popen(['ffmpeg', '-i', video_file,
                               '-vf', 'fps={}'.format(rate),
                               '-vframes', str(num_frames),
                               '-loglevel', 'panic',
                               os.path.join(framesFolder,'%d.jpg') ]).
    frame_paths = sorted([os.path.join(framesFolder, frame)
                          for frame in os.listdir(framesFolder)])
    frames = load_frames(frame_paths, num_frames)
    subprocess.call(['rm', '-rf', framesFolder + '*.jpg'])
    return frames

def remove(path):
    """ param <path> could either be relative or absolute. """
    if os.path.isfile(path) or os.path.islink(path):
        os.remove(path)  # remove the file
    elif os.path.isdir(path):
```

```
        shutil.rmtree(path)  # remove dir and all contains
    else:
        raise ValueError("file {} is not a file or dir.".format(path))

def parse(video_file):
    """Extracts a numpy array representation from a given video file."
    # Extract frames from video
    framerate = 25 # frames per second (fps)
    framesFolder = os.getcwd() + '/frames'
    frames = extract_frames(video_file, framesFolder, framerate)
    # Transform frames into tensors with values [0,1]
    # Load an image transformer
    transform = trn.ToTensor()
    tframes = []
    for frame in frames:
        #frame = frame.resize((224,224))
        tframes.append(frame)
    tmpdata = torch.stack([transform(frame) for frame in tframes] )
    tmpdata = tmpdata.numpy()
    remove(framesFolder)
    return tmpdata

video_array = parse(video_file)
```

```
Video shape: (200, 3, 240, 320)
```

1.4 Prepare Model(s)

1.4.1 Select Trained Model(s)

In [6]:

```
# TODO:
# Select models from: ['DenseNet169', 'DenseNet201', 'InceptionV3', '1
#                      'MobileNetV2', 'Resnet50', 'VGG19', 'Xception',
#                      'InceptionResnetV2-64avg']

model_names = ['DenseNet169', 'DenseNet201', 'InceptionV3', 'Inception
               'MobileNetV2', 'Resnet50', 'VGG19', 'Xception', 'I3D-In
               'InceptionResNetV2-64avg']

# TODO:
# Specify model directory
```

1.4.2 Load and Compile Model(s)

In [7]:

```
def get_model(model_name):
    if model_name == 'DenseNet169':
        return load_model(model_folder + 'D169-224x224x3-339-im.h5')
    elif model_name == 'DenseNet201':
        return load_model(model_folder + 'D201-224x224x3-339-im.h5')
    elif model_name == 'InceptionV3':
        return load_model(model_folder + 'Iv3-224x224x3-339-im.h5')
    elif model_name == 'InceptionResNetV2':
```

```
        return load_model(model_folder + 'IRv2-224x224x3-339-im.h5')
    elif model_name == 'MobileNet':
        return load_model(model_folder + 'M-224x224x3-339-im.h5')
    elif model_name == 'MobileNetV2':
        return load_model(model_folder + 'Mv2-224x224x3-339-im.h5')
    elif model_name == 'Resnet50':
        return load_model(model_folder + 'R50-224x224x3-339-im.h5')
    elif model_name == 'VGG19':
        return load_model(model_folder + 'VGG19-224x224x3-339-im.h5')
    elif model_name == 'Xception':
        return load_model(model_folder + 'X-224x224x3-339-im.h5')
    elif model_name == 'C3D':
        return load_model(model_folder + 'C3D-16x224x224x3-339-m.h5')
    elif model_name == 'I3D-InceptionV1':
        return load_model(model_folder + 'I3DIv1-16x224x224x3-339-ikm.
    elif model_name == 'LRCN':
        return load_model(model_folder + 'LRCN-16x224x224x3-339-m.h5')
    elif model_name == 'InceptionResNetV2-64avg':
        return load_model(model_folder + 'IRv2avg-64x224x224x3-339-im.
    print('Should not reach here')
    return None

def transformer(model_name):
    if model_name in ['DenseNet169', 'DenseNet201', 'InceptionV3', 'In
                      'MobileNetV2', 'Resnet50', 'VGG19', 'Xception']:
        return 1
    elif model_name in ['I3D-InceptionV1', 'C3D', 'LRCN']:
        return 2
    elif model_name in ['InceptionResNetV2-64avg']:
        return 3
    else:
        print('Should not reach here')
        return None

met = get_metrics()
models = [get_model(m) for m in model_names]
transforms = [transformer(m) for m in model_names]
for model in models:
    model.compile(optimizer=SGD(), loss='categorical_crossentropy', me
print('----------------------------')
```

```
WARNING:tensorflow:From /state/partition1/llgrid/pkg/anaconda/anaconda
3-2020a/lib/python3.6/site-packages/tensorflow/python/ops/init_ops.py:
97: calling GlorotUniform.__init__ (from tensorflow.python.ops.init_op
s) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing i
t to the constructor
WARNING:tensorflow:From /state/partition1/llgrid/pkg/anaconda/anaconda
3-2020a/lib/python3.6/site-packages/tensorflow/python/ops/init_ops.py:
1251: calling VarianceScaling.__init__ (from tensorflow.python.ops.ini
t_ops) with dtype is deprecated and will be removed in a future versio
n.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing i
t to the constructor
WARNING:tensorflow:From /state/partition1/llgrid/pkg/anaconda/anaconda
```

```
3-2020a/lib/python3.6/site-packages/tensorflow/python/ops/init_ops.py:
97: calling Zeros.__init__ (from tensorflow.python.ops.init_ops) with
dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing i
t to the constructor
WARNING:tensorflow:From /state/partition1/llgrid/pkg/anaconda/anaconda
3-2020a/lib/python3.6/site-packages/tensorflow/python/ops/init_ops.py:
97: calling Ones.__init__ (from tensorflow.python.ops.init_ops) with d
type is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing i
t to the constructor
WARNING:tensorflow:From /state/partition1/llgrid/pkg/anaconda/anaconda
3-2020a/lib/python3.6/site-packages/tensorflow/python/ops/init_ops.py:
97: calling Orthogonal.__init__ (from tensorflow.python.ops.init_ops)
with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing i
t to the constructor
WARNING:tensorflow:From /state/partition1/llgrid/pkg/anaconda/anaconda
3-2020a/lib/python3.6/site-packages/tensorflow/python/ops/math_grad.p
y:1250: add_dispatch_support.<locals>.wrapper (from tensorflow.python.
ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:No training configuration found in save file: the m
odel was *not* compiled. Compile it manually.
---------------------------
13 model(s) loaded and compiled
```

1.5 Inference

Note that there is still randomness in the frame(s) being selected for inference. Therefore, running this cell multiple times will give slightly different results.

In [8]:
```
# TODO:
# Specify number of top prediction classes to show
```

In [9]:
```
# Performs Inference and Displays Results

def label_to_cat(index, classes):
    for cat in categories_labels.keys():
        if categories_labels[cat][index]==1:
            return cat
def top_classes(tups, k=5):
    top_k = []
    for i in range(len(tups)):
        if tups[i][1] <= k:
            top_k.append(tups[i])
    return sorted(top_k, key=lambda tup: tup[1])


label = categories_labels[mit_class_gt]
```

```
datum = [video_array], 0, label

print('Model Name')
print('(class_name, prediction_rank, probability)')
print('-----------------------------------------')
print()

for i in range(len(models)):
    model_name = model_names[i]
    model = models[i]
    t = transforms[i]
    y_hats = []
    for i in range(5): # sample averaging
        arr, label = transform(datum, transform_num=t)
        X = np.array([arr])
        y = np.array([label])
        y_hat = model.predict(X, batch_size=1)
        y_hats.append(y_hat)
    y_hat = np.average(y_hats, axis=0)
    order = y_hat.argsort()
    rev_ranks = order.argsort()
    classes = len(y_hat[0])
    tups = []
    for i in range(classes):
        cat = label_to_cat(i, classes)
        tups.append( (cat, 339-rev_ranks[0][i], y_hat[0][i]) )
    short_list = top_classes(tups, k)
    print(model_name)
    for tup in short_list:
        print(tup)
```

```
Model Name
(class_name, prediction_rank, probability)
-----------------------------------------

DenseNet169
('skiing', 1, 0.9085125)
('boarding', 2, 0.025353933)
('sliding', 3, 0.0076259063)
('slipping', 4, 0.005177059)
('skating', 5, 0.0050613997)

DenseNet201
('skiing', 1, 0.58287513)
('sliding', 2, 0.05612815)
('slipping', 3, 0.043456513)
('skating', 4, 0.03366632)
('descending', 5, 0.018783813)

InceptionV3
('skiing', 1, 0.5834716)
('sliding', 2, 0.028677518)
('slipping', 3, 0.028165314)
('skating', 4, 0.023576682)
('boarding', 5, 0.021372344)
```

```
InceptionResNetV2
('skiing', 1, 0.6004424)
('boarding', 2, 0.053444456)
('jumping', 3, 0.034714162)
('skating', 4, 0.02377763)
('falling', 5, 0.023125973)

MobileNet
('skiing', 1, 0.2398886)
('officiating', 2, 0.14038408)
('slipping', 3, 0.10001288)
('boarding', 4, 0.095631294)
('sliding', 5, 0.038019247)

MobileNetV2
('skiing', 1, 0.16444364)
('sliding', 2, 0.07168311)
('boarding', 3, 0.06656089)
('slipping', 4, 0.06320612)
('skating', 5, 0.036344867)

Resnet50
('skiing', 1, 0.6975399)
('slipping', 2, 0.035883404)
('skating', 3, 0.023647005)
('balancing', 4, 0.015892975)
('officiating', 5, 0.015808614)

VGG19
('slipping', 1, 0.113963984)
('boarding', 2, 0.105429515)
('skiing', 3, 0.0875464)
('sliding', 4, 0.06347631)
('skating', 5, 0.043519128)

Xception
('skiing', 1, 0.45929837)
('slipping', 2, 0.13140288)
('racing', 3, 0.03646811)
('boarding', 4, 0.02920106)
('playing+sports', 5, 0.027381152)

I3D-InceptionV1
('skiing', 1, 0.6306972)
('boarding', 2, 0.04456868)
('sliding', 3, 0.03136454)
('skating', 4, 0.029156972)
('playing+sports', 5, 0.028053453)

C3D
('skiing', 1, 0.1877743)
('skating', 2, 0.07308701)
('crashing', 3, 0.032002218)
('boarding', 4, 0.02330643)
('flipping', 5, 0.02312484)
```

```
LRCN
('skiing', 1, 0.3637656)
('skating', 2, 0.07882427)
('slipping', 3, 0.041725338)
('jumping', 4, 0.03955111)
('sliding', 5, 0.035640597)

InceptionResNetV2-64avg
('skiing', 1, 0.57533085)
('boarding', 2, 0.074289046)
('slipping', 3, 0.035638716)
('jumping', 4, 0.028740693)
('sliding', 5, 0.019542733)
```

1.6 Questions

Any questions can be directed to Matthew Hutchinson at hutchinson@alum.mit.edu (mailto:hutchinson@alum.mit.edu).

Python license: https://docs.python.org/3/license.html (https://docs.python.org/3/license.html)

TensorFlow license: https://github.com/tensorflow/tensorflow/blob/master/LICENSE (https://github.com/tensorflow/tensorflow/blob/master/LICENSE)

NumPy license: https://numpy.org/doc/stable/license.html (https://numpy.org/doc/stable/license.html)

PIL license: http://www.pythonware.com/products/pil/license.htm (http://www.pythonware.com/products/pil/license.htm)

PyTorch license: https://github.com/pytorch/pytorch/blob/master/LICENSE (https://github.com/pytorch/pytorch/blob/master/LICENSE)

H5Py license: https://docs.h5py.org/en/stable/licenses.html (https://docs.h5py.org/en/stable/licenses.html)

In [ ]: