Submitted by: Md. Saifur Rahman
Email: mdsaifur.rahman.1@und.edu
Date: 04/05/2021

HOMEWORK #4: Robotics

CSCI 543: Assignment 4

Problem #1: Video inbetweening using 3D convolutions

Report:

Video inbetweening using 3D convolutions

Abstract

I implement and run a tutorial on the video inbetweening using 3D Convolution Neural Network. This 3D Convolution Neural Network is using only a start and an end frame of the given video to generate 14 plausible and diverse video sequences in between these two frames. I am using TensorFlow Hub to get the "Conv3d-based video inbetweening model" which is trained on the Berkley Artificial Intelligence Research (BAIR) Lab robot pushing dataset. This TensorFlow Hub has the following characteristics.

i. Has models for BAIR Robot pushing videos and KTH action video dataset (though this colab uses only BAIR)

ii. BAIR dataset already available in Hub (https://tfhub.dev/google/tweening_conv3d_bair/1).

iii. However, KTH videos need to be supplied by the users themselves. I can consider the KTH videos in AVI format from the following link (https://www.csc.kth.se/cvap/actions/)

iv. Only the evaluation (video generation) is considered for now

v. Batch size (16) and frame size (64x64 RGB frames) are hard-coded.

The original idea behind this implementation follow the approach and dataset of the following two published research project (i) From Here to There: Video Inbetweening Using Direct 3D Convolutions [1], (ii) Self-supervised visual planning with temporal skip connections [2]. I perform this implementation of video inbetweening using python programming language. Besides, I am using TensorFlow, NumPy, matplotlib library etc. for video frame plotting to show the results.

TensorFlow tutorial link: https://www.tensorflow.org/hub/tutorials/tweening_conv3d

My code in google drive (Colab):
https://colab.research.google.com/drive/1TXqiLUb155IVbUp8bgZhO5ZP87IjLZ1_?usp=sharing

Submitted by: Md. Saifur Rahman
Email: mdsaifur.rahman.1@und.edu
Date: 04/05/2021

Tools: Python programming language, TensorFlow, Google Colab, TensorFlow Hub, Google Drive, Google Cloud Storage.

Main Objectives

Generate meaningful and diverse filled video frames in between a start and an end frame of the given video using 3D Convolution Neural Network. This work generates 14 plausible and diverse video sequences in between these two frames (start & end). This model receives a start frame, an end frame, and a Gaussian noise vector as inputs. Then generate 14 meaningful video sequence. The total of video frames is 14 + 2(start, end) = 16.

Dataset Details

The original dataset comes from the TFDS- TensorFlow Datasets (tfds.load('bair_robot_pushing_small', split='test') which has 30 GB archive including training data for this work. The "Conv3d-based video inbetweening model" is already trained with BAIR pushing dataset which is found in the following link (https://tfhub.dev/google/tweening_conv3d_bair/1). To test this model, I use the following test dataset (https://storage.googleapis.com/download.tensorflow.org/data/bair_test_traj_0_to_255.tfrecords) which contains 190 MB of test data. These video datasets are known as unstructured data and required 3D modeling of neural network. Machine learning model usually required three types of dataset to perform any experimental analysis, (i) 'train dataset' to train the model, (ii) 'validation dataset' for evaluation the quality of the model, (iii) 'test dataset' to test the model after the model has gone through the validation process. I am using "Conv3d-based video inbetweening model" from TensorFlow Hub which is an already trained model. This trained model uses the BAIR dataset splits into train (sequences 2560 to remaining) and validation (sequences of 256 to 2559) dataset [1]. That's why I don't need the 'train dataset' and 'validation dataset' in this work. I use only the 'test dataset' to perform the test operation of this 3D Convolutional Neural Network.

Data Preprocessing

Since the dataset builder expects the 'train', 'validation' and 'test' split in the data to be downloaded, patch the dataset so that it only expects the test dataset. I am using *BairRobotPushingSmall()* and *SplitGenerator()* methods to perform this task. The pixel value range for each video frame is [0.0, 255.0]. Besides, to be consistent all video frame is considered as 64x64 RGB frames. The batch size of 32 samples is considered in this model

The ML Approach

I use TesorFlow Hub to get the "Conv3d-based video inbetweening model" which is an already trained model on BAIR robot pushing dataset. This model is designed based on 3D Convolutional

Submitted by: Md. Saifur Rahman
Email: mdsaifur.rahman.1@und.edu
Date: 04/05/2021

Neural Network. URL for this machine learning robotics model is - https://tfhub.dev/google/tweening_conv3d_bair/1. According to reference [1], this is a fully convolutional model. This ML model work through the following components.

i. "A **2D-convolutional image encoder**, which maps the input key frames to a latent space.
ii. A **3D-convolutional latent representation generator**, which learns how to incorporate the information contained in the input frames with progressively increasing temporal resolution.
iii. A **video generator**, which uses **transposed 3D-convolutions** to decode the latent representation into video frames." [1]

In addition to this 3D CNN, a **video discriminator** and an **image discriminator** are used to train this model using adversarial learning. Adversarial learning helps this model to generate realistic video frames. For training purpose, the ADAM optimizer with $\beta 1 = 0{:}5$, $\beta 2 = 0{:}999$, $\boldsymbol{\varepsilon} = 10^{-8}$ is used in this model. The batch size of 32 samples is considered in this model with a conservative learning rate of 5×10^{-5}. The details of this approach can be found in From Here to There: Video Inbetweening Using Direct 3D Convolutions. The complete procedure of this ML approach for generating in between video frames is shown below.

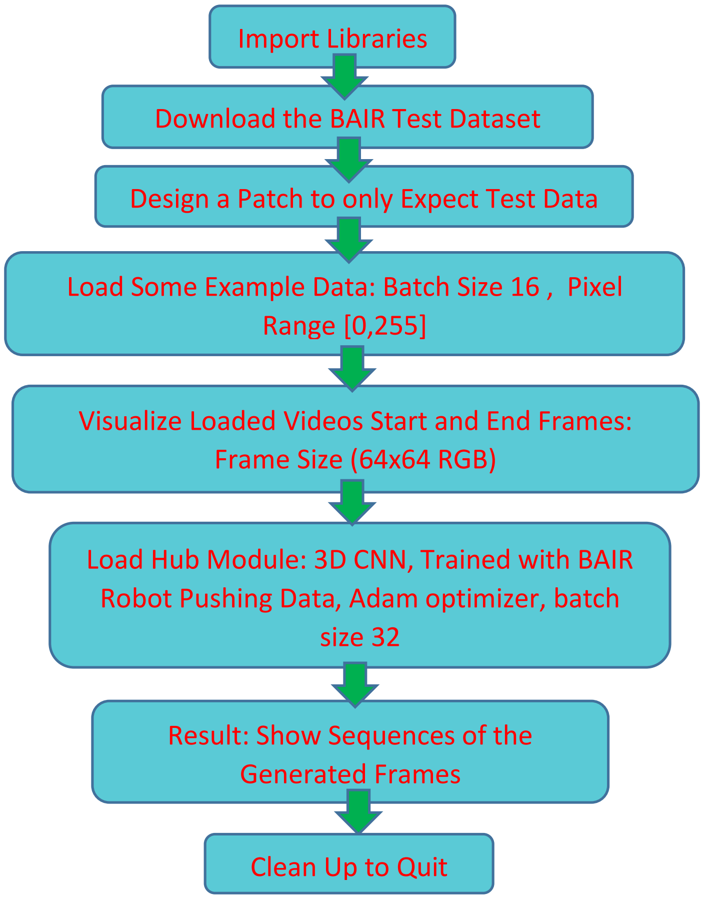

Import Libraries
Download the BAIR Test Dataset
Design a Patch to only Expect Test Data
Load Some Example Data: Batch Size 16 , Pixel Range [0,255]
Visualize Loaded Videos Start and End Frames: Frame Size (64x64 RGB)
Load Hub Module: 3D CNN, Trained with BAIR Robot Pushing Data, Adam optimizer, batch size 32
Result: Show Sequences of the Generated Frames
Clean Up to Quit

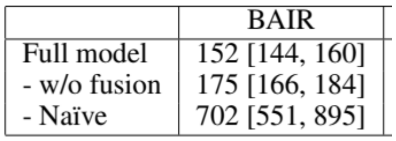
Figure 1: Complete procedure of this video inbetweening ML approach

Submitted by: Md. Saifur Rahman
Email: mdsaifur.rahman.1@und.edu
Date: 04/05/2021

ML Model Performance

“Experiments on several widely-used benchmark datasets show that this ML robotic approach is able to generate meaningful and diverse in-between video sequences, according to both quantitative and qualitative evaluations” [1]. According to reference [1], the **FDV** (**Fréchet video distance**) is used as primary evaluation metrics. Besides, two baseline functions; (i) **Baseline without fusion**, and (ii) **Naïve** methods are also used to make a performance comparison measurement. A lower mean value in FDV means higher quality in the generated videos. According to the table 1, this 3D CNN model generates higher quality videos.

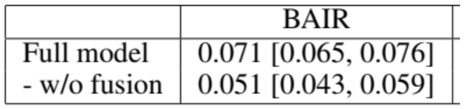
Table 1: The mean FVD for the 3D CNN model and two baselines



	BAIR
Full model	152 [144, 160]
- w/o fusion	175 [166, 184]
- Naïve	702 [551, 895]

Table 2 shows the diversity measurement by the average pairwise cosine distance in the FVD embedding space. A higher value means more diversion in videos. According to table 2, this 3D CNN creates more diversion in videos.

Table 2: Diversity measurement by the average pairwise cosine distance in FVD



	BAIR
Full model	0.071 [0.065, 0.076]
- w/o fusion	0.051 [0.043, 0.059]

References

[1] Yunpeng Li, Dominik Roblek, and Marco Tagliasacchi. From Here to There: Video Inbetweening Using Direct 3D Convolutions. arXiv preprint arXiv:1905.10240, 2019.

[2] Frederik Ebert, Chelsea Finn, Alex X. Lee, and Sergey Levine. Self-Supervised Visual Planning with Temporal Skip Connections. Conference on Robot Learning (CoRL), 2017

Codes with Comments:

I am giving explanation of each line segment from my implementation of this video inbetweening using 3D convolutional neural network from TF hub.

Colab link for the code:
https://colab.research.google.com/drive/1TXqiLUb155IVbUp8bgZhO5ZP87IjLZ1_?usp=sharing

Submitted by: Md. Saifur Rahman
Email: mdsaifur.rahman.1@und.edu
Date: 04/05/2021

```
# -*- coding: utf-8 -*-

"""Video inbetweening.ipynb

Automatically generated by Colaboratory.

Original file is located at

    https://colab.research.google.com/drive/1TXqiLUb155IVbUp8bgZhO5ZP87IjLZ1_

"""


# import libraries


import tensorflow as tf


import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

import tensorflow_hub as hub

import tensorflow_datasets as tfds


from tensorflow_datasets.core import SplitGenerator

from tensorflow_datasets.video.bair_robot_pushing import BairRobotPushingSmall


import tempfile

import pathlib


TEST_DIR = pathlib.Path(tempfile.mkdtemp()) / "bair_robot_pushing_small/softmotion30_44k/test/"


TEST_DIR


# Download the test split to $TEST_DIR

!mkdir -p $TEST_DIR
```

Submitted by: Md. Saifur Rahman
Email: mdsaifur.rahman.1@und.edu
Date: 04/05/2021

```
!wget -nv
https://storage.googleapis.com/download.tensorflow.org/data/bair_test_traj_0_to_255.tfrecords -O
$TEST_DIR/traj_0_to_255.tfrecords


# Since the dataset builder expects the train and test split to be downloaded, patch it so it only expects
the test data to be available.


builder = BairRobotPushingSmall()

test_generator = SplitGenerator(name='test', gen_kwargs={"filedir": str(TEST_DIR)})

builder._split_generators = lambda _: [test_generator]

builder.download_and_prepare()


# BAIR: Demo based on numpy array inputs

# @title Load some example data (BAIR).

batch_size = 16


# If unable to download the dataset automatically due to "not enough disk space", please download
manually to Google Drive and load using tf.data.TFRecordDataset.

ds = builder.as_dataset(split="test")

test_videos = ds.batch(batch_size)

first_batch = next(iter(test_videos))

input_frames = first_batch['image_aux1'][:, ::15]

input_frames = tf.cast(input_frames, tf.float32)


# @title Visualize loaded videos start and end frames.


print('Test videos shape [batch_size, start/end frame, height, width, num_channels]: ',
input_frames.shape)

sns.set_style('white')

plt.figure(figsize=(4, 2*batch_size))
```

```
for i in range(batch_size)[:4]:

  plt.subplot(batch_size, 2, 1 + 2*i)

  plt.imshow(input_frames[i, 0] / 255.0)

  plt.title('Video {}: First frame'.format(i))

  plt.axis('off')

  plt.subplot(batch_size, 2, 2 + 2*i)

  plt.imshow(input_frames[i, 1] / 255.0)

  plt.title('Video {}: Last frame'.format(i))

  plt.axis('off')


# Load Hub Module

# Conv3d-based video inbetweening model trained on the BAIR robot pushing dataset (with 64x64 RGB frames). This module stochastically generates 14 in-between video frames given the start and end frames.

# It takes as input <Tensor(tf.float32, shape=[16, 2, 64, 64, 3])>, representing a (fixed-size) batch of 16 time-wise concatenated start/end frames of 64x64 with 3 channels (RGB),

# and outputs <Tensor(tf.float32, shape=[16, 14, 64, 64, 3])>, which are the 14 in-between frames. The pixel value range is [0.0, 255.0].

# Note that the model is trained for this particular dataset. Therefore it will only work on video frames from the dataset or someting very similar in appearance.


hub_handle = 'https://tfhub.dev/google/tweening_conv3d_bair/1'

module = hub.load(hub_handle).signatures['default']


# Generate and show the videos

#  The pixel value range is [0.0, 255.0].


filled_frames = module(input_frames)['default'] / 255.0
```

Submitted by: Md. Saifur Rahman
Email: mdsaifur.rahman.1@und.edu
Date: 04/05/2021

```
# Show sequences of generated video frames.

# Experiments on several widely-used benchmark datasets show that it is able to generate meaningful
and diverse in-between video sequences, according to both quantitative and qualitative evaluations.

# Concatenate start/end frames and the generated filled frames for the new videos.


generated_videos = np.concatenate([input_frames[:, :1] / 255.0, filled_frames, input_frames[:, 1:] /
255.0], axis=1)


for video_id in range(4):
  fig = plt.figure(figsize=(10 * 2, 2))
  for frame_id in range(1, 16):
    ax = fig.add_axes([frame_id * 1 / 16., 0, (frame_id + 1) * 1 / 16., 1],
                      xmargin=0, ymargin=0)
    ax.imshow(generated_videos[video_id, frame_id])
    ax.axis('off')


# terminate the kernel and free memory resources


import os, signal
os.kill(os.getpid(), signal.SIGKILL)
```