data processing pjf 0325

March 25, 2022

[1]:
```
import torch
import torchvision
import torchvision.transforms as transforms
import torch.optim as optim
import time
from itertools import count
import natsort
import datetime
import numpy as np
```

[2]:
```
from torch.utils.data import Dataset, DataLoader
import albumentations as A
from albumentations.pytorch import ToTensorV2
import cv2
import glob
import numpy
import random
import pandas as pd
import tqdm
```

[3]:
```
print(f"Is CUDA supported by this system? {torch.cuda.is_available()}")
print(f"CUDA version: {torch.version.cuda}")
# Storing ID of current CUDA device
cuda_id = torch.cuda.current_device()
print(f"ID of current CUDA device: {torch.cuda.current_device()}")
print(f"Name of current CUDA device: {torch.cuda.get_device_name(cuda_id)}")

device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
print(device)
```

```
Is CUDA supported by this system? True
CUDA version: 11.3
ID of current CUDA device: 0
Name of current CUDA device: NVIDIA GeForce RTX 2070 Super
cuda:0
```

[13]:
```
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
```

```
print(device)
```

```
cuda:0
```

1 Video Preprocessing

[9]:
```
for i in range(1,10):
    vidcap = cv2.VideoCapture(r"C:
↪\Users\panji\EECS6691_Advanced_DL\Assignment2\video\RALIHR_surgeon01_fps01_000%d.
↪mp4"% i)
    success, image = vidcap.read()
    success, image = vidcap.read()
    count = 0
    while success:
        cv2.imwrite(r'C:
↪\Users\panji\EECS6691_Advanced_DL\Assignment2\training_data_images\Video%dframe%d.
↪jpg'% (i, count), image)
        success,image = vidcap.read()
        count += 1
```

[8]:
```
# Cut frames from videos
# Cut frames from videos
for i in range(1,71):
    vidcap = cv2.VideoCapture(r"C:
↪\Users\panji\EECS6691_Advanced_DL\Assignment2\video\RALIHR_surgeon01_fps01_00%d.
↪mp4"% i)
    success, image = vidcap.read()
    success, image = vidcap.read()
    count = 0
    while success:
        cv2.imwrite(r'C:
↪\Users\panji\EECS6691_Advanced_DL\Assignment2\training_data_images\Video%dframe%d.
↪jpg'% (i, count), image)
        success,image = vidcap.read()
        count += 1
```

2 Label processing

[78]:
```
#Get labels
a =  pd.read_csv("video.phase.trainingData.clean.StudentVersion.csv")
a1 = a[a.PhaseName !="Access"]
a2 = a1
print(a2.head)

for i in range(a2.shape[0]):
    if any([c.isdigit() for c in a2.iat[i,1]]):
```

```
        a2.iat[i,1] = a2.iat[i,1][:-1]
    a2.iat[i,1] = a2.iat[i,1].lower()
b1 = a2["PhaseName"].unique()
```

```
<bound method NDFrame.head of                          videoName
PhaseName  Start    End
1     RALIHR_surgeon01_fps01_0001      Stationary Idle1  00:00  00:16
2     RALIHR_surgeon01_fps01_0001   Transitionary Idle1  00:16  00:35
3     RALIHR_surgeon01_fps01_0001           Out of body  00:35  01:05
4     RALIHR_surgeon01_fps01_0001   Transitionary Idle2  01:05  01:59
5     RALIHR_surgeon01_fps01_0001    Peritoneal Scoring  01:59  02:55
…                             …                    …    …    …
2020  RALIHR_surgeon01_fps01_0070   Peritoneal closure8  51:24  52:23
2021  RALIHR_surgeon01_fps01_0070  Positioning suture11  52:23  54:08
2022  RALIHR_surgeon01_fps01_0070   Peritoneal closure9  54:08  57:27
2023  RALIHR_surgeon01_fps01_0070  Positioning suture12  57:27  57:55
2024  RALIHR_surgeon01_fps01_0070  Transitionary Idle11  57:55  58:18

[2022 rows x 4 columns]>
```

[80]:
```
b2 = list(b1)
unique_classes = {}
for c in b1:
    if c[:3] == 'adh':
        unique_classes[c] = 0
        b2.remove(c)
    elif c[-7:] == 'scoring':
        unique_classes[c] = 1
        b2.remove(c)
    elif c[:6] == 'preper':
        unique_classes[c] = 2
        b2.remove(c)
    elif c[:3] == 'red':
        unique_classes[c] = 3
        b2.remove(c)
    elif c[:7] == 'mesh po':
        unique_classes[c] = 4
        b2.remove(c)
    elif c[:7] == 'mesh pl':
        unique_classes[c] = 5
        b2.remove(c)
    elif c == 'positioning of suture':
        unique_classes[c] = 6
        b2.remove(c)
    elif c == 'positioning suture':
        unique_classes[c] = 7
        b2.remove(c)
```

```
    elif c[:6] == 'direct':
        unique_classes[c] = 8
        b2.remove(c)
    elif c[:4] == 'cath':
        unique_classes[c] = 9
        b2.remove(c)
    elif c == 'peritoneal closure':
        unique_classes[c] = 10
        b2.remove(c)
    elif c == 'transitory idle':
        unique_classes[c] = 11
        b2.remove(c)
    elif c == 'stationary idle':
        unique_classes[c] = 12
        b2.remove(c)
    elif c == 'out of body':
        unique_classes[c] = 13
        b2.remove(c)
    elif c == 'blurry':
        unique_classes[c] = 14
        b2.remove(c)
```

[84]:
```
#print(len(b1))
print(b1)
print(b2)
print(len(b2))
print(len(unique_classes))
print(unique_classes)
```

```
['stationary idle' 'transitionary idle' 'out of body' 'peritoneal scoring'
 'preperioneal dissection' 'reduction of hernia' 'blurry'
 'positioning of suture' 'mesh placement' 'acquiring suture'
 'peritoneal closure' 'preperitoneal dissection' 'mesh positioning'
 'positioning suture' 'transitionary idle1' 'direct hernia repair'
 'catheter insertion' 'suture positioning' 'suture positioining'
 'prepreitoneal dissection' 'prepetioneal dissection' 'transtionary idle'
 'adhesiolysis' 'positioining suture' 'prepertioneal dissection'
 'cathter insertion' 'transtiionary idle' 'mesh positioning '
 'transitioning idle' 'primary hernia repair' 'transitiionary idle1'
 'transtitionary idle' 'positioning suture1' 'preperitoneal dissection1'
 'perperitoneal dissection']
['transitionary idle', 'acquiring suture', 'transitionary idle1', 'suture
positioning', 'suture positioining', 'prepreitoneal dissection', 'prepetioneal
dissection', 'transtionary idle', 'positioining suture', 'transtiionary idle',
'transitioning idle', 'primary hernia repair', 'transitiionary idle1',
'transtitionary idle', 'positioning suture1', 'perperitoneal dissection']
16
19
```

```
{'stationary idle': 12, 'out of body': 13, 'peritoneal scoring': 1,
'preperioneal dissection': 2, 'reduction of hernia': 3, 'blurry': 14,
'positioning of suture': 6, 'mesh placement': 5, 'peritoneal closure': 10,
'preperitoneal dissection': 2, 'mesh positioning': 4, 'positioning suture': 7,
'direct hernia repair': 8, 'catheter insertion': 9, 'adhesiolysis': 0,
'prepertioneal dissection': 2, 'cathter insertion': 9, 'mesh positioning ': 4,
'preperitoneal dissection1': 2}
```

[88]:
```
unique_classes['transitionary idle'] = 11
unique_classes['acquiring suture'] = unique_classes['positioning of suture']
unique_classes['transitionary idle1'] = 11
# the two below are suspicious
unique_classes['suture positioning'] = unique_classes['positioning suture']
unique_classes['suture positioining'] = unique_classes['positioning suture']

unique_classes['prepreitoneal dissection'] = unique_classes['preperitoneal␣
 ↪dissection']
unique_classes['prepetioneal dissection'] = unique_classes['preperitoneal␣
 ↪dissection']
unique_classes['positioining suture'] = unique_classes['positioning suture']
unique_classes['transtiionary idle'] = 11
unique_classes['transitioning idle'] = 11
unique_classes['transtionary idle'] = 11
unique_classes['primary hernia repair'] = unique_classes['direct hernia repair']
unique_classes['transitiionary idle1'] = 11
unique_classes['transtitionary idle'] = 11
unique_classes['positioning suture1'] = unique_classes['positioning suture']
unique_classes['perperitoneal dissection'] = unique_classes['preperitoneal␣
 ↪dissection']
print(len(unique_classes))
print(unique_classes)
# neglecting the 'Access' phase
```

```
35
{'stationary idle': 12, 'out of body': 13, 'peritoneal scoring': 1,
'preperioneal dissection': 2, 'reduction of hernia': 3, 'blurry': 14,
'positioning of suture': 6, 'mesh placement': 5, 'peritoneal closure': 10,
'preperitoneal dissection': 2, 'mesh positioning': 4, 'positioning suture': 7,
'direct hernia repair': 8, 'catheter insertion': 9, 'adhesiolysis': 0,
'prepertioneal dissection': 2, 'cathter insertion': 9, 'mesh positioning ': 4,
'preperitoneal dissection1': 2, 'transitionary idle': 11, 'acquiring suture': 6,
'transitionary idle1': 11, 'suture positioning': 7, 'suture positioining': 7,
'prepreitoneal dissection': 2, 'prepetioneal dissection': 2, 'positioining
suture': 7, 'transtiionary idle': 11, 'transitioning idle': 11, 'primary hernia
repair': 8, 'transitiionary idle1': 11, 'transtitionary idle': 11, 'positioning
suture1': 7, 'perperitoneal dissection': 2, 'transtionary idle': 11}
```

[66]:
```
def TimeChange(a):
    if len(a) != 8:
        x = time.strptime(a,'%M:%S')
        x1 = int(datetime.timedelta(minutes = x.tm_min, seconds = x.tm_sec).
↪total_seconds())
    elif len(a) == 8 and a[:2] == "00":
        a = a[3:]
        x = time.strptime(a,'%M:%S')
        x1 = int(datetime.timedelta(minutes = x.tm_min, seconds = x.tm_sec).
↪total_seconds())
    else:
        x = time.strptime(a,'%H:%M:%S')
        x1 = int(datetime.timedelta(hours = x.tm_hour, minutes = x.tm_min,␣
↪seconds = x.tm_sec).total_seconds())
    return x1
```

[91]:
```
indices= []
phases =[]
for i in range(a2.shape[0]):
    index = int(a2.iloc[i,0][-2:])
    start_time = int(TimeChange(a2.iloc[i,2]))
    end_time = int(TimeChange(a2.iloc[i,3]))
    indices1 = [index]*(end_time-start_time)
    indices.extend(indices1)
    phases1 = [int(unique_classes[a2.iloc[i,1]])]*(end_time-start_time)
    phases.extend(phases1)

x2 = {"Video":indices, "Phases": phases}
df = pd.DataFrame(x2, columns = ["Video", "Phases"])
df.to_csv("Processed_data.csv")
```