test

April 19, 2023

[8]:
```
from sklearn.model_selection import train_test_split

import numpy as np
import pandas as pd
import os
import sys
import pickle

sys.path.insert(0, 'src/models')
sys.path.insert(0, 'src/explanation')

from CNN_models import *
from DCAM import *

import matplotlib.pyplot as plt
from random import randint
from tqdm import tqdm_notebook as tqdm
```

[9]:
```
PATH_TO_UCR = "data/UCR_UEA/"

with open(PATH_TO_UCR + "FingerMovements.pickle",'rb') as f:
    X,y = pickle.load(f)

# Convert the UCR-UEA format into a list of list
def generate_list_instance(x):
    res = []
    for i in range(len(x)):
        res.append(list(x[i]))
    return np.array(res)

dict_label = {}
count = 0
for val in set(y.values):
    dict_label[val] = count
    count += 1

all_class_all = []
```

```
all_label = []
for i in range(len(X)):
    all_class_all.append(generate_list_instance(X.values[i]))
    all_label.append(dict_label[y.values[i]])

original_length = len(all_class_all[0][0])
num_classes = len(set(y.values))
original_dim = len(all_class_all[0])
nb_instance = len(all_class_all)

all_class, all_class_test, label, label_test = train_test_split(all_class_all,␣
↪all_label,stratify=all_label, test_size=1-0.8,random_state=11081994)

# Generate C-wised input for d-based models (i.e., dCNN, dResNet, and␣
↪dInceptionTime)
def gen_cube(instance):
    result = []
    for i in range(len(instance)):
        result.append([instance[(i+j)%len(instance)] for j in␣
↪range(len(instance))])
    return result

x = np.array([gen_cube(acl) for acl in all_class])
dataset_mat = TSDataset(x,label)
dataloader_cl1 = data.DataLoader(dataset_mat, batch_size=32, shuffle=True)

x = np.array([gen_cube(acl) for acl in all_class_test])
dataset_mat_test = TSDataset(x,label_test)
dataloader_cl1_test = data.DataLoader(dataset_mat_test, batch_size=1,␣
↪shuffle=True)
```

[14]:
```
# This is dInceptionTime
#modelarch = dInceptionModel(num_blocks=3, in_channels=original_dim,␣
↪out_channels=64,
#                            bottleneck_channels=64, kernel_sizes=[10,20,40],
#                            use_residuals=True, num_pred_classes=num_classes).
↪to('cpu')

# dResNet gives the same error
modelarch =␣
↪dResNetBaseline(original_dim,mid_channels=128,num_pred_classes=num_classes).
↪to('cpu')

model = ModelCNN(modelarch,'cpu')
```

```
model.
 ↪train(num_epochs=70,dataloader_cl1=dataloader_cl1,dataloader_cl1_test=dataloader_cl1_test)
```

```
Epoch [1/70], Loss Train: 0.6951,Loss Test: 0.6939, Accuracy Train: 49.70%,
Accuracy Test: 50.00%
```

```
      ␣
↪---------------------------------------------------------------------------

       KeyboardInterrupt                         Traceback (most recent call␣
↪last)

       <ipython-input-14-43f84fc7d137> in <module>
         9 model = ModelCNN(modelarch,'cpu')
        10
   ---> 11 model.
↪train(num_epochs=70,dataloader_cl1=dataloader_cl1,dataloader_cl1_test=dataloader_cl1_test)


       ~/Desktop/LIPADE/dCAM/src/models/CNN_models.py in train(self,␣
↪num_epochs, dataloader_cl1, dataloader_cl1_test, model_name, verbose)
        94                                                                  #␣
↪==================backward====================
        95                                                                 ␣
↪loss_train = self.criterion(output_train.float(), v_label_train.long())
   ---> 96                                                                 ␣
↪loss_train.backward()
        97                                                                  self.
↪optimizer.step()
        98                                                                  #␣
↪================eval on train==================


       /opt/anaconda3/lib/python3.8/site-packages/torch/tensor.py in␣
↪backward(self, gradient, retain_graph, create_graph)
       219                 retain_graph=retain_graph,
       220                 create_graph=create_graph)
   --> 221         torch.autograd.backward(self, gradient, retain_graph,␣
↪create_graph)
       222
       223     def register_hook(self, hook):


       /opt/anaconda3/lib/python3.8/site-packages/torch/autograd/__init__.py in␣
↪backward(tensors, grad_tensors, retain_graph, create_graph, grad_variables)
       128         retain_graph = create_graph
```

```
    129
--> 130     Variable._execution_engine.run_backward(
    131         tensors, grad_tensors_, retain_graph, create_graph,
    132         allow_unreachable=True)  # allow_unreachable flag


    KeyboardInterrupt:
```

[ ]: