```
from __future__ import print_function
%tensorflow_version 1.x
import tensorflow as tf
import numpy as np
import time
import pickle
from sklearn import preprocessing
```

```
    TensorFlow 1.x selected.
```

```
#Bahaar's code cell - ignore
#source: https://stackoverflow.com/questions/51869713/how-to-read-edf-data-in-python-3
#import mne
#file = "my_path\\my_file.edf" #how do you add file
#data = mne.io.read_raw_edf(file)
#raw_data = data.get_data()
# you can get the metadata included in the file and a list of all channels:
#info = data.info
#channels = data.ch_names

from google.colab import drive
import os
drive.mount('/content/gdrive', force_remount=True)

root_dir = "/content/gdrive/My\ Drive/"

os.chdir("/content/gdrive/My Drive/unzipped_all_14sub.zip/")
os.listdir()
!pwd
!ls

#base_dir = root_dir + 'my-images/'
```

```
    Mounted at /content/gdrive
    /content/gdrive/My Drive/unzipped_all_14sub.zip
    all_14sub.p
```

```
# #Akarsh's code cell
# #!pwd
# #!ls gdrive/MyDrive/
# import os
# path = "/content/gdrive/MyDrive/"
# os.chdir(path)
```

```
def one_hot(y_):
    # Function to encode output labels from number indexes
    # e.g.: [[5], [0], [3]] --> [[0, 0, 0, 0, 0, 1], [1, 0, 0, 0, 0, 0], [0, 0, 0, 1,
```

```
    y_ = y_.reshape(len(y_))
    y_ = [int(x) for x in y_]
    n_values = np.max(y_) + 1
    return np.eye(n_values)[np.array(y_, dtype=np.int32)]

def extract(input, n_fea, time_window, moving):
    global n_classes
    xx = input[:, :n_fea]
    yy = input[:, n_fea:n_fea + 1]
    new_x = []
    new_y = []
    number = int((xx.shape[0] / moving) - 1)
    for i in range(number):
        ave_y = np.average(yy[int(i * moving):int(i * moving + time_window)])
        if ave_y in range(n_classes + 1):
            new_x.append(xx[int(i * moving):int(i * moving + time_window), :])
            new_y.append(ave_y)
        else:
            new_x.append(xx[int(i * moving):int(i * moving + time_window), :])
            new_y.append(0)

    new_x = np.array(new_x)
    new_x = new_x.reshape([-1, n_fea * time_window])
    new_y = np.array(new_y)
    new_y.shape = [new_y.shape[0], 1]
    data = np.hstack((new_x, new_y))
    data = np.vstack((data, data[-1]))  # add the last sample again, to make the samp
    return data


def compute_accuracy_t(v_xs, v_ys):  # this function only calculate the acc of CNN_tas
    global prediction_t
    y_pre = sess.run(prediction_t, feed_dict={xs: v_xs, keep_prob: keep})
    correct_prediction = tf.equal(tf.argmax(y_pre, 1), tf.argmax(v_ys, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
    result = sess.run(accuracy, feed_dict={xs: v_xs, ys_t: v_ys, keep_prob: keep})
    return result


def compute_accuracy_p(v_xs, v_ys):  # this function only calculate the acc of CNN_tas
    global prediction_p
    y_pre = sess.run(prediction_p, feed_dict={xs: v_xs, keep_prob: keep})
    correct_prediction = tf.equal(tf.argmax(y_pre,1), tf.argmax(v_ys,1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
    result = sess.run(accuracy, feed_dict={xs: v_xs, ys_p: v_ys, keep_prob: keep})
    return result


# # WoW! use this to limit the GPU number
# import os
# GPU_ID = 2
# os.environ['CUDA_VISIBLE_DEVICES'] = str(GPU_ID)
```

```
# print('Let`s start!, GPU:', GPU_ID)
!cd /home/
!pwd
!ls /content
```

```
/content/gdrive/My Drive/unzipped_all_14sub.zip
gdrive  sample_data
```

```
# data reading
# python 3: add ',encoding='iso-8859-1'' in the pickle.load.
import pickle
#Bahaar's path
all_data = pickle.load(open("/content/gdrive/My Drive/unzipped_all_14sub.zip/all_14sub
#Akarsh's path
#all_data = pickle.load(open("/content/gdrive/MyDrive/all_14sub.p", "rb" ), encoding='
print(type(all_data), all_data.shape, all_data[:, -1])

n_classes = 2
n_person_ = 13  # the number of training subjects
sample_persub = 250*500  # we have overlapping now
print(type(all_data), all_data.shape, all_data[:, -1])

no_fea = 21  # data.shape[-1] - 1
seg_length = 250  # # 255 for raw data, 96 for layer 23, 64 for layer 2, 32 for layer

scaler = preprocessing.MinMaxScaler()  # normalization
F = scaler.fit_transform(all_data[:, :no_fea])  # scale to [0, 1]

all_data = np.hstack((F, all_data[:, no_fea:no_fea+1]))  # only use the task ID
```

```
<class 'numpy.ndarray'> (1750000, 23) [ 2.  2.  2. ... 17. 17. 17.]
<class 'numpy.ndarray'> (1750000, 23) [ 2.  2.  2. ... 17. 17. 17.]
```

```
"""Make person label"""
n_sample_ = int(2*sample_persub/seg_length )  # the number of sampls of each subject a
ll = np.ones([n_sample_, 1])*0
for hh in range(1, n_person_):
    ll_new = np.ones([n_sample_, 1])*hh
    ll = np.vstack((ll, ll_new))
print('the shape of maked person label', ll.shape)

ll_test = np.ones([n_sample_, 1])*n_person_

ss_train = time.clock()
# Person Independent
```

```
the shape of maked person label (13000, 1)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:11: DeprecationWarn
```

```
    # This is added back by InteractiveShellApp.init_path()



for P_ID in range(14):  # n_person_++1
    if P_ID==0:
        reuse=False
    else:
        reuse=True
    """Select train and test subject"""
    data_ = all_data[sample_persub*P_ID:sample_persub*(P_ID+1)]

    list = range(sample_persub*P_ID, sample_persub*(P_ID+1))
    data = np.delete(all_data, list, axis=0)
    # overlap
    train_data = extract(data, n_fea=no_fea, time_window=seg_length, moving=(seg_lengt
    test_data = extract(data_, n_fea=no_fea, time_window=seg_length, moving=(seg_lengt
    # continue
    """Replace the original person data by the maked data"""
    no_fea_long = train_data.shape[-1] - 1  # here is - 2, because has two IDs
    print(train_data[:, :no_fea_long+1].shape, ll.shape)
    train_data = np.hstack((train_data[:, :no_fea_long+1], ll))
    test_data = np.hstack((test_data[:, :no_fea_long + 1], ll_test))
    print(train_data.shape)
    np.random.shuffle(train_data)
    np.random.shuffle(test_data)
    print(train_data.shape, test_data.shape)

    feature_train = train_data[:, :no_fea_long]
    feature_test = test_data[:, :no_fea_long]
    label_train_t = train_data[:, no_fea_long:no_fea_long + 1]
    label_test_t = test_data[:, no_fea_long:no_fea_long + 1]
    label_train_p = train_data[:, no_fea_long + 1:no_fea_long + 2]

    label_train_t = one_hot(label_train_t)
    label_test_t = one_hot(label_test_t)
    label_train_p = one_hot(label_train_p)


    n_class_t = 2  # 0-3
    n_class_p = n_person_  # 0-8
    keep = 0.8

    a = feature_train

    ## batch split
    batch_size = int(feature_test.shape[0])
    train_fea = []
    n_group = int(feature_train.shape[0]/feature_test.shape[0])
    for i in range(n_group):
        f = a[(0+batch_size*i):(batch_size+batch_size*i)]
```

```
        train_fea.append(f)
    print (train_fea[0].shape)

    train_label_t=[]
    for i in range(n_group):
        f = label_train_t[(0 + batch_size * i):(batch_size + batch_size * i), :]
        train_label_t.append(f)
    print (train_label_t[0].shape)

    train_label_p = []
    for i in range(n_group):
        f = label_train_p[(0 + batch_size * i):(batch_size + batch_size * i), :]
        train_label_p.append(f)
    print (train_label_p[0].shape)

    """Placeholder"""
    # define placeholder for inputs to network
    tf.compat.v1.disable_eager_execution()

    xs = tf.compat.v1.placeholder(tf.float32, [None, no_fea_long], name = 'xsss')  # 2
    ys_t = tf.compat.v1.placeholder(tf.float32, [None, n_class_t], name='ys_t')
    ys_p = tf.compat.v1.placeholder(tf.float32, [None, n_class_p], name='ys_p')
    keep_prob = tf.compat.v1.placeholder(tf.float32, name='keep')

    """AE code, whihc is divided into two represents"""
    """Use tf.nn.relu in the hidden layer if maxmin scaler; sigmoid if z-score;
    use maxmin, AE converge better but the classification training acc cannot reach 1(
    use z-score, the opposite. I perfer z-score. Or use maxmin, make the network deepe
    """
    """Convolutional AE"""
    with tf.compat.v1.variable_scope("AE", reuse=reuse):
        # dim_code = 1000
        input = tf.reshape(xs, [-1, seg_length, no_fea, 1])  # [200, 14]
        input = tf.contrib.layers.batch_norm(input, decay=0.9)
        input = tf.nn.dropout(input, keep_prob)
        l_AE, w_AE = 2, 1
        print(xs.shape)  # [n_samples, 28,28,1]

        depth_AE = 4  # default is 8
        conv1 = tf.layers.conv2d(inputs=input, filters=depth_AE, kernel_size=[2, 2], p
                                 activation=tf.nn.relu)
        h_t = tf.layers.max_pooling2d(inputs=conv1, pool_size=[l_AE, w_AE], strides=[1
        # pool1 = tf.contrib.layers.batch_norm(pool1, decay=0.9)

        conv1_p = tf.layers.conv2d(inputs=input, filters=depth_AE, kernel_size=[2, 2],
                                   activation=tf.nn.relu)
        h_p = tf.layers.max_pooling2d(inputs=conv1_p, pool_size=[l_AE, w_AE], strides=

        # decoder
        output_t = tf.layers.conv2d_transpose(h_t, kernel_size=5, filters=1, strides=|
        # output_t = tf.nn.relu(tf.contrib.layers.batch_norm(output_t, decay=0.9))
```

```
        output_p = tf.layers.conv2d_transpose(h_p, kernel_size=5, filters=1, strides=|
        # output_p = tf.nn.relu(tf.contrib.layers.batch_norm(output_p, decay=0.9))
        output = (output_t + output_p) / 2

        # #another decoder
        # h = (h_t + h_p)/2
        # output = tf.layers.conv2d_transpose(h, kernel_size=5, filters=1, strides=[l_
        # output = tf.nn.relu(tf.contrib.layers.batch_norm(output, decay=0.9))

        output = tf.reshape(output, [-1, seg_length * no_fea])

    """CNN code for task, maybe we can make it deeper? """
    l_1, w_1 = 2, 2
    l_2, w_2 = 2, 2
    l_3, w_3 = 2, 2
    l_4, w_4 = 2, 1

    with tf.compat.v1.variable_scope("class_t", reuse=reuse):
        # x_image_t = tf.reshape(h_t, [-1, 10, 10, 1])  # [200, 14]
        x_image_t = tf.contrib.layers.batch_norm(h_t, decay=0.9)

        # x_image_t = tf.nn.dropout(x_image_t, keep_prob)
        print(x_image_t.shape)  # [n_samples, 28,28,1]
        depth_1 = 16  # default is 8
        conv1 = tf.layers.conv2d(inputs=x_image_t, filters=depth_1, kernel_size=[3, 3]
        pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[l_1, w_1], strides=[]
        pool1 = tf.contrib.layers.batch_norm(pool1, decay=0.9)

        depth_2 = 32  # default is 32
        conv2 = tf.layers.conv2d(inputs=pool1, filters=depth_2, kernel_size=[3, 3], pa
        pool2 = tf.layers.max_pooling2d(inputs=conv2, pool_size=[l_2, w_2], strides=[]
        pool2 = tf.contrib.layers.batch_norm(pool2, decay=0.9)

        depth_3 = 64
        conv3 = tf.layers.conv2d(inputs=pool2, filters=depth_3, kernel_size=[2, 2], pa
        pool3 = tf.layers.max_pooling2d(inputs=conv3, pool_size=[l_3, w_3], strides=[]
        pool3 = tf.contrib.layers.batch_norm(pool3, decay=0.9)
        # print(pool1.get_shape(), pool2.get_shape(), pool3.get_shape(),)

        depth_4 = 128
        conv4 = tf.layers.conv2d(inputs=pool3, filters=depth_4, kernel_size=[2, 2], pa
        pool4 = tf.layers.max_pooling2d(inputs=conv4, pool_size=[l_4, w_4], strides=[]
        pool4 = tf.contrib.layers.batch_norm(pool4, decay=0.9)

        fc1 = tf.contrib.layers.flatten(pool4)  # flatten the pool 2
        print(pool1.get_shape(), pool2.get_shape(), pool3.get_shape(),  pool4.get_shap



        # """Add another FC layer"""
        fc1 = tf.layers.dense(fc1, units=300, activation=tf.nn.sigmoid)
        fc1 = tf.nn.dropout(fc1, keep_prob)
```

```
        dim_hidden = 21
        fc3 = tf.layers.dense(fc1, units=dim_hidden, activation=tf.nn.sigmoid)
        fc3 = tf.nn.dropout(fc3, keep_prob)

        # Attention layer
        att = tf.layers.dense(xs, units=fc3.shape[-1], activation=tf.nn.sigmoid)
        fc3 = tf.multiply(fc3, att)

        prediction_t = tf.layers.dense(fc3, units=n_class_t, activation=None)
        print('prediction_t', prediction_t.get_shape())

    """CNN code for person"""
    with tf.compat.v1.variable_scope("class_p", reuse=reuse):
        x_image_p = tf.contrib.layers.batch_norm(h_p, decay=0.9)

        conv1_p = tf.layers.conv2d(inputs=x_image_p, filters=depth_1, kernel_size=[3,
        pool1_p = tf.layers.max_pooling2d(inputs=conv1_p, pool_size=[l_1, w_1], stride
        pool1_p = tf.contrib.layers.batch_norm(pool1_p, decay=0.9)

        conv2_p = tf.layers.conv2d(inputs=pool1_p, filters=depth_2, kernel_size=[3, 3
        pool2_p = tf.layers.max_pooling2d(inputs=conv2_p, pool_size=[l_2, w_2], stride
        pool2_p = tf.contrib.layers.batch_norm(pool2_p, decay=0.9)

        conv3_p = tf.layers.conv2d(inputs=pool2_p, filters=depth_2, kernel_size=[2, 2
                                   activation=tf.nn.relu)
        pool3_p = tf.layers.max_pooling2d(inputs=conv3_p, pool_size=[l_3, w_3], stride
        pool3_p = tf.contrib.layers.batch_norm(pool3_p, decay=0.9)

        conv4_p = tf.layers.conv2d(inputs=pool3_p, filters=depth_4, kernel_size=[2, 2
        pool4_p = tf.layers.max_pooling2d(inputs=conv4_p, pool_size=[l_4, w_4], stride
        pool4_p = tf.contrib.layers.batch_norm(pool4_p, decay=0.9)

        fc1_p = tf.contrib.layers.flatten(pool4_p)  # flatten the pool 2

        dim_hidden_p = 200
        fc3_p = tf.layers.dense(fc1_p, units=dim_hidden_p, activation=tf.nn.sigmoid)

        fc3_p = tf.nn.dropout(fc3_p, keep_prob)

        prediction_p = tf.layers.dense(fc3_p, units=n_class_p, activation=None)
        print('prediction_p', tf.shape(prediction_p))

    def kl_divergence(p, q):
        return tf.reduce_sum(p * tf.log(p/q))


    """cost calculation"""
    train_vars = tf.trainable_variables()
    l2_AE = 0.005 * sum(tf.nn.l2_loss(var) for var in tf.trainable_variables() if var.
    l2_class = 0.005 * sum(tf.nn.l2_loss(var) for var in tf.trainable_variables() if v
```

```
    """multiply 5 to enhance the cross_entropy_t """
    cross_entropy_t = 10*tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits
    cross_entropy_p = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=pr
    """Add 0.1 to reduce the cost_AE"""
    cost_AE = tf.reduce_mean(tf.pow(xs - output, 2)) + l2_AE

    class_vars = [var for var in train_vars if var.name.startswith("class")]  # discri
    AE_vars = [var for var in train_vars if var.name.startswith("AE")]
    t_vars = [var for var in train_vars if var.name.startswith("class_t")]

    cost = cost_AE  + cross_entropy_t +   cross_entropy_p + l2_class + l2_AE
    lr = 0.00005  # use 0.0001 for parameter tuning
    with tf.compat.v1.variable_scope("optimization", reuse=reuse):
        train_step_task = tf.train.AdamOptimizer(lr).minimize(cost, )
        # train_step_AE = tf.train.AdamOptimizer(lr).minimize(cost_AE+l2_AE, var_list=
        train_step_t = tf.train.AdamOptimizer(lr).minimize(cross_entropy_t)
    # 1. AE learning rate= 0.00001 2. dim_code larger better, 3. add cost_AE on cost.

    con = tf.ConfigProto()
    con.gpu_options.allow_growth = True
    sess = tf.Session(config=con)
    init = tf.global_variables_initializer()
    sess.run(init)
    # History records of loss functions
    cost_his = []
    cost_AE_his = []
    cost_t_his = []
    cost_p_his = []

    test_cost_t_his = []

    start=time.clock()
    step = 1
    while step < 81:  # 251 iterations
        #print('iteration step', step)
        for i in range(n_group):
            feed = {xs: train_fea[i], ys_t: train_label_t[i], ys_p: train_label_p[i],
            sess.run(train_step_task, feed_dict=feed)
            sess.run(train_step_t, feed_dict=feed)

        if step % 10 == 0:
            #"""training cost"""
            cost_, cost_AE_, cross_entropy_p_, cross_entropy_t_=sess.run([cost, cost_A
                                                                         feed_dict={xs
                                                 ys_t: train_label_t[0], ys_p: train_label

            #"""testing cost"""
            cost_AE_test_, cross_entropy_t_test_=sess.run([cost_AE, cross_entropy_t],
                                                          feed_dict ={xs: feature_test

            print('person, step:',P_ID, step, 'train acc task', compute_accuracy_t(fea
```

```
                'train acc person', compute_accuracy_p(feature_train, label_train_p)
                ',the test acc task', compute_accuracy_t(feature_test, label_test_t)
                'testing: AE, t', cost_AE_test_, cross_entropy_t_test_)

            print('training cost: total, AE, t, p',cost_, cost_AE_,  cross_entropy_t_,
            cost_his.append(cost_)
            cost_AE_his.append(cost_AE_)
            cost_t_his.append(cross_entropy_t_)
            cost_p_his.append(cross_entropy_p_)

            test_cost_t_his.append(cross_entropy_t_test_)

        # save the attention weights for fine-grained analysis
        if step % 80 == 0:
            att_ = sess.run(att, feed_dict={xs: feature_test, ys_t: label_test_t, keep

            ss = time.clock()
            pred = sess.run(prediction_t, feed_dict={xs: feature_test, ys_t: label_tes
            print('training, testing time', time.clock()-ss_train, time.clock()-ss)

            pickle.dump(att_,  open('/content/gdrive/My Drive/TUH_attention_p'
                                    +str(step)+'_backup.p',  "wb"), protocol=2) #/home
            print('attention saved, person:', P_ID)

        step += 1

    # save the cost history values for convergence analysis
    pickle.dump(cost_his, open('/content/gdrive/My Drive/cost_his.p', "wb"))
    pickle.dump(cost_AE_his,
                open('/content/gdrive/My Drive/cost_AE_his.p', "wb")) #home/xiangzhan
    pickle.dump(cost_t_his,
                open('/content/gdrive/My Drive/cost_t_his.p', "wb")) #/home/xiangzhan
    pickle.dump(cost_p_his,
                open('/content/gdrive/My Drive/cost_p_his.p', "wb")) #/home/xiangzhan
    pickle.dump(test_cost_t_his,
                open('/content/gdrive/My Drive/test_cost_t_his.p', "wb")) #/home/xian
    print("five losses are saved at /home/xiangzhang/scratch/activity_recognition_prac
```

```
Instructions for updating:
Use keras.layers.MaxPooling2D instead.
WARNING:tensorflow:From <ipython-input-11-c565ddb28383>:99: conv2d_transpose (fro
Instructions for updating:
Use `tf.keras.layers.Conv2DTranspose` instead.
(?, 125, 21, 4)
WARNING:tensorflow:From /tensorflow-1.15.2/python3.7/tensorflow_core/contrib/lay
Instructions for updating:
Use keras.layers.flatten instead.
(?, 62, 10, 16) (?, 31, 5, 32) (?, 15, 2, 64) (?, 7, 2, 128) (?, 1792)
WARNING:tensorflow:From <ipython-input-11-c565ddb28383>:150: dense (from tensorfl
Instructions for updating:
Use keras.layers.Dense instead.
prediction_t (?, 2)
prediction_p Tensor("class_p/Shape:0", shape=(2,), dtype=int32)
WARNING:tensorflow:From <ipython-input-11-c565ddb28383>:204: softmax_cross_entrop
```

```
Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See `tf.nn.softmax_cross_entropy_with_logits_v2`.

WARNING:tensorflow:From /tensorflow-1.15.2/python3.7/tensorflow_core/python/ops/n
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:234: DeprecationWarn
person, step: 0 10 train acc task 0.7876923 train acc person 0.35084614 ,the tes
training cost: total, AE, t, p 10.153784 0.3794348 4.873458 1.962121
person, step: 0 20 train acc task 0.8147692 train acc person 0.441 ,the test acc
training cost: total, AE, t, p 9.217756 0.2814971 4.5745306 1.6207138
person, step: 0 30 train acc task 0.82453847 train acc person 0.5126154 ,the tes
training cost: total, AE, t, p 8.385537 0.21246253 4.203146 1.4139563
person, step: 0 40 train acc task 0.83715385 train acc person 0.56646156 ,the te
training cost: total, AE, t, p 7.6164184 0.16447286 3.8150368 1.247415
person, step: 0 50 train acc task 0.8499231 train acc person 0.6269231 ,the test
training cost: total, AE, t, p 7.0666633 0.13208799 3.6066773 1.0834851
person, step: 0 60 train acc task 0.85992306 train acc person 0.6866923 ,the tes
training cost: total, AE, t, p 6.6316586 0.11059286 3.45974 0.94181395
person, step: 0 70 train acc task 0.8671538 train acc person 0.7466923 ,the test
training cost: total, AE, t, p 6.3245735 0.096024126 3.3912966 0.82615626
person, step: 0 80 train acc task 0.8736154 train acc person 0.7810769 ,the test
training cost: total, AE, t, p 6.0230618 0.08601313 3.3175085 0.7047704
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:270: DeprecationWarn
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:272: DeprecationWarn

training, testing time 414.43655 0.1663239999999746
attention saved, person: 0
five losses are saved at /home/xiangzhang/scratch/activity_recognition_practice/l
(13000, 5251) (13000, 1)
(13000, 5252)
(13000, 5252) (1000, 5252)
(1000, 5250)
(1000, 2)
(1000, 13)
(?, 5250)
(?, 125, 21, 4)
(?, 62, 10, 16) (?, 31, 5, 32) (?, 15, 2, 64) (?, 7, 2, 128) (?, 1792)
prediction_t (?, 2)
```

```
#80 iterations
accuracies = [0.771, 0.83, 0.94, 0.772, 0.719, 0.707, 0.75, 0.834, 0.521, 0.616, 0.896
print("step 80")
print(sum(accuracies)/len(accuracies))

#70 iterations
accuracies = [0.771, 0.854, 0.961, 0.769, 0.689, 0.698, 0.76, 0.826, 0.505, 0.62, 0.90
print("step 70")
print(sum(accuracies)/len(accuracies))

#50 iterations
```

```
accuracies = [0.812, 0.828, 0.973, 0.767, 0.759, 0.715, 0.79, 0.804, 0.528, 0.61, 0.92
print("step 50")
print(sum(accuracies)/len(accuracies))
```

```
step 80
0.7467142857142858
step 70
0.7467142857142857
step 50
0.7567857142857145
```

Double-click (or enter) to edit

✓ 0s completed at 11:16 AM