```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

class Dataset_raw:
    def __init__(self,N,x,y,t):
        self.N = N
        self.x = x
        self.y = y
        self.t = t

def readDataset(path):
    d1 = []
    d2 = []
    d3 = []
    with open(path,'r') as file:
        for line in file:
            data = line.split()
            d1.append(float(data[0]))
            d2.append(float(data[1]))
            d3.append(float(data[2]))

    dataset = Dataset_raw(len(d1),[],[],[])
    for i in range(len(d1)):
        dataset.x.append(d1[i])
        dataset.y.append(d2[i])
        dataset.t.append(d3[i])
    return dataset

def next_batch(dataset, batch_size):
    batch_x = np.array([[0 for i in range(2)] for j in range(batch_size)])
    batch_y = np.array([[0 for i in  range(2)]for j in range(batch_size)])

    # Shuffle dataset
    k = np.random.permutation(dataset.N)
    for i in range(batch_size):
        batch_x[i][0] = dataset.x[k[i]]
        batch_x[i][1] = dataset.y[k[i]]
        label = dataset.t[k[i]]
        if label == 0:
            batch_y[i][0] = 1
            batch_y[i][1] = 0
        else:
            batch_y[i][0] = 0
            batch_y[i][1] = 1

    return batch_x, batch_y

def plot_dataset(dataset):
    for i in range(dataset.N):
        if dataset.t[i] == 1.0:
            plt.scatter(dataset.x[i], dataset.y[i], c='r', label='Class 0')
        else:
            plt.scatter(dataset.x[i], dataset.y[i], c='b', label='Class 1')

    plt.title('Training dataset')


def write_data(loss, acc, epoch_list, filename):
    with open('Dataset/loss_' + filename,'w') as f:
        for s in loss:
```

```
            f.write(str(s) + '\n')
    with open('Dataset/acc_' + filename,'w') as f:
        for s in acc:
            f.write(str(s) + '\n')
    with open('Dataset/epoch_list_' + filename,'w') as f:
        for s in epoch_list:
            f.write(str(s) + '\n')

def read_data(filename):
    with open('Dataset/acc_' + filename,'r') as f:
        acc = [float(line.rstrip('\n')) for line in f]

    with open('Dataset/loss_' + filename,'r') as f:
        loss = [float(line.rstrip('\n')) for line in f]

    with open('Dataset/epoch_list_' + filename,'r') as f:
        epoch_list = [float(line.rstrip('\n')) for line in f]

    return acc, loss, epoch_list


def plot_data():
    # Read files
    acc_hidden_1_1mean, loss_hidden_1_1mean, epoch_list_hidden_1_1mean = read_data('hidden_1_1me
    acc_hidden_2_1mean, loss_hidden_2_1mean, epoch_list_hidden_2_1mean = read_data('hidden_2_1me
    acc_hidden_1_2mean, loss_hidden_1_2mean, epoch_list_hidden_1_2mean = read_data('hidden_1_2me
    acc_hidden_2_2mean, loss_hidden_2_2mean, epoch_list_hidden_2_2mean = read_data('hidden_2_2me


    plt.figure(2)
    plt.plot(epoch_list_hidden_1_1mean,loss_hidden_1_1mean, label='1 hidden layer')
    plt.plot(epoch_list_hidden_2_1mean,loss_hidden_2_1mean, label = '2 hidden layers')
    plt.plot(epoch_list_hidden_1_2mean,loss_hidden_1_2mean, label='1 hidden layer, double mean')
    plt.plot(epoch_list_hidden_2_2mean,loss_hidden_2_2mean, label = '2 hidden layers, double me
    plt.legend()
    plt.xlim([2, epoch_list_hidden_2_2mean[len(epoch_list_hidden_1_1mean)-1]])
    plt.title('Loss propagation on training set')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')

    plt.figure(3)
    plt.plot(epoch_list_hidden_1_1mean, acc_hidden_1_1mean, label='1 hidden layer')
    plt.plot(epoch_list_hidden_2_1mean,acc_hidden_2_1mean, label = '2 hidden layers')
    plt.plot(epoch_list_hidden_1_2mean, acc_hidden_1_2mean, label='1 hidden layer, double mean')
    plt.plot(epoch_list_hidden_2_2mean,acc_hidden_2_2mean, label = '2 hidden layers, double mean
    plt.legend()
    plt.xlim([0, epoch_list_hidden_2_2mean[len(epoch_list_hidden_1_1mean)-1]])
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.title('Accuracy propagation on training set')

def normalize_dataset(dataset):
    dataset.x = tf.keras.utils.normalize(x=dataset.x,order=2)
    dataset.x = dataset.x[0]
    dataset.y = tf.keras.utils.normalize(x=dataset.y, order=2)
    dataset.y = dataset.y[0]
    return dataset
```