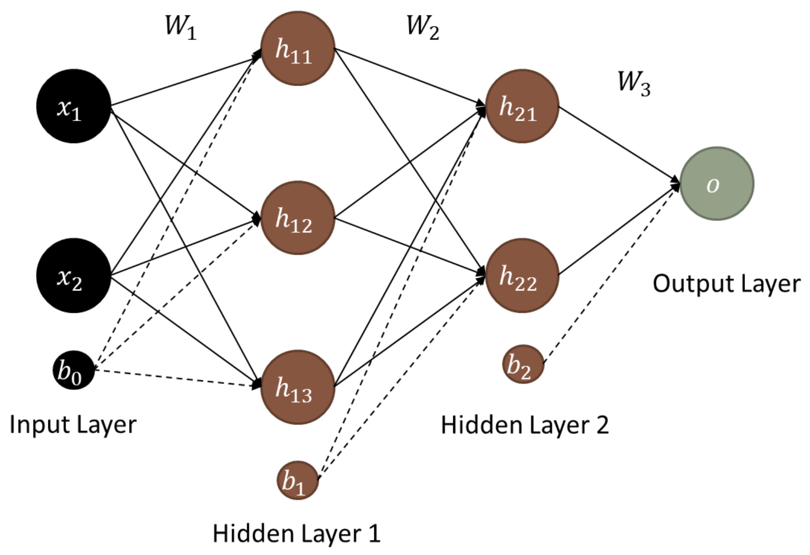
1101 Deep Learning – Homework 3

Due: Dec 15, 2021, 11:59pm

Please turn in your code and report. (Total: 120%)

1. (15%) Please use a neural net (with its architecture shown below) to find the decision boundary based on 'train.mat.' The activation function must be used in the two hidden layers and the output layer. You can use any off-the-shelf functions to construct and optimize your network model. Report the test error on the test set 'test.mat' (percentage of misclassified test samples).



2. (45%) Please design a neural net to predict network intrusion types for the datasets "train_DefenseSystem.csv" and "test_DefenseSystem.csv." Please read the dataset description in the file "DefenseSystem_Readme.docx."

   2.1. (30%) Randomly select 80% of the training set as your training set and the rest 20% as your validation set. Use the validation set to design your network structure and choose possible hyper-parameters (try 3 different network structures at least and list all the parameters you may use). You need to report the training and validation accuracy for each network.

   2.2. (15%) Use your best network (out of 3 at least) to predict the intrusion type for the test set. Please report your results.

   Please detail your models in the report. For neural net, your model must at least have **four** hidden layers, meaning including the input and output layers, in total, it has at least **six** layers.

3. (60%) The mnist dataset contains handwritten digits, where it has a training set of 60,000 examples, and a test set of 10,000 examples.
   Please download it here: http://yann.lecun.com/exdb/mnist/

3.1 (10%) You are asked to construct a classification model based on convolutional neural networks for digit recognition. Please report the prediction accuracy for the testing set.

3.2 (30%) Please add 10%, 20%, 30% of salt-and-pepper noise to the test images, and test them using the model you trained on clean images from 3.1. Report the prediction **accuracies**. Compare your results with those from 3.1. What do you find?

3.3 (20%) Please use VGG-16 as a backbone pre-trained on ImageNet to fine-tune your model for the classification and redo Q. 3.1 and Q. 3.2.

To add noise to the images, you could use the example code below:

```
import random
import numpy as np

noise_lv = 0.1    % 0.1, 0.2, 0.3
img_size = 28*28
for i in range(len(X_train)):
  ran_seq = random.sample([n for n in range(img_size)], np.int(img_size*noise_lv))
  x = X_train[i].reshape(-1, img_size)
  x[0, ran_seq]=255
```