Homework 3 **CSCE 633**
Due: 11:59pm on November 14, 2023

Instructions for homework submission
a) For each question, please explain your thought process, results, and observations in a markdown cell after the code cells. Please do not just include your code without justification.
b) **You can use any available libraries for this homework.**
c) The maximum grade for this homework, excluding bonus questions, is **100 points**. There are **2 bonus questions**.

Question 1: Machine learning for object recognition

In this problem, we will process images coming from the CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes (i.e., airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck) with 6000 images per class. There are 50000 training images and 10000 test images. More information about the data can be found here.

The CIFAR-10 dataset is included in most of the frameworks for deep neural networks e.g., Keras, Tensorflow, PyTorch. In Keras, CIFAR-10 can be downloaded by running:

```
from keras.datasets import cifar10

# load the dataset
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

Note: The original CIFAR-10 with color images can take a long time to train. Hence, we will convert the color images into grayscale images. One way to achieve this would be to use a library such as `scikit-image`.

```
from skimage.color import rgb2gray

# convert to grayscale images
X_train = rgb2gray(X_train)
X_test = rgb2gray(X_test)
```

(a) (6 points) Visualization: Randomly select and visualize 5-6 images (no need to include all the classes).
Note: You can find a useful link on image pre-processing here: `https://www.tensorflow.org/api_docs/python/tf/image/per_image_standardization`

(b) (6 points) Data exploration: Count the number of samples per class in the training data.

(c) (26 points) Image classification with FNNs: In this part, you will use a feedforward neural network (FNN) (also called "multilayer perceptron") to perform the object classification task. The input of the FNN comprises of all the pixels of the image. Use one of the five batches of the training data as a validation set.

(c.i) (20 points) Experiment on the validation set with different FNN hyper-parameters, e.g. # layers, #nodes per layer, activation function, dropout, weight regularization, etc. Choose 3

hyper-parameter combinations and for each combination, please do the following: (1) monitor the loss on the train and validation set across the epochs of the FNN training; (2) report the final classification accuracy on the training and validation sets; (3) report the running time for training the FNN; (4) report the # parameters that are learned for each FNN.
Note: If running the FNN takes a long time, you can subsample the training data (i.e., choose a random set of samples from training) or sub-sample the input images to a smaller size (e.g., 24×24).

(c.ii) (6 points) Run the best model that was found based on the validation set from question (**c.i**) on the testing set. Report the classification accuracy on the testing set. Report the confusion matrix for each class.
Note: The confusion matrix is a 10×10 matrix; its rows correspond to the actual labels for each class, while its columns correspond to the predicted classes. Element (i, j) includes the number of samples that belonged to the i^{th} class and were predicted as the j^{th} class. In a perfect classification task, the non-diagonal elements of the matrix will be all non-zero.

(d) (26 points) Image classification with CNNs: In this part, you will use a convolutional neural network (CNN) to perform the object classification task.

(d.i) (20 points) Experiment on the validation set with different CNN hyper-parameters, e.g. # layers, filter size, stride size, activation function, dropout, weight regularization, etc. Choose 3 hyper-parameter combinations and for each combination, please do the following: (1) monitor the loss on the train and validation set across the epochs of the CNN training; (2) report the final classification accuracy on the training and validation sets; (3) report the running time for training the CNN; (4) report the # parameters that are learned for each CNN. How do these metrics compare to the FNN?

(d.ii) (6 points) Run the best model that was found based on the validation set from question (**d.i**) on the testing set. Report the classification accuracy on the testing set. How does this metric compare to the FNN?

(e) (Bonus - 5 points) Bayesian optimization for hyper-parameter tuning: Instead of performing grid or random search to tune the hyper-parameters of the CNN, we can also try a model-based method for finding the optimal hyper-parameters through Bayesian optimization. This method performs a more intelligent search on the hyper-parameter space in order to estimate the best set of hyper-parameters for the data. Use publicly available libraries (e.g., hyperopt in Python) to perform a Bayesian optimization on the hyper-parameter space using the validation set. Report the emotion classification accuracy on the testing set.
Hint: Check this and this source.

(f) (Bonus - 5 points) Fine-tuning: Use a pre-trained CNN (e.g., the pre-trained example of the MNIST dataset that we saw in class, or any other available pre-trained CNN) and fine-tune it on the CIFAR-10 data. Please experiment with different fine-tuning hyper-parameters (e.g., #layers to fine-tune, regularization during fine-tuning) on the validation set. Report the classification accuracy for all hyper-parameter combinations on the validation set. Also report the classification accuracy with the best hyper-parameter combination on the testing set.

Question 2: SVM
In this problem, we will use university application data for the purpose of admission classification. The data can be found in the following link:
`https://github.com/selva86/datasets/blob/master/Admission.csv`

(a) (4 points) Data Preprocessing: Create a binary label based on the column "Chance of Admit ". Convert any values bigger than the mean to 1 and 0 otherwise.

(b) (4 points) Model Initialization: Initilize 4 different SVM models with the following kernels:

1. SVC with linear kernel
2. LinearSVC (linear kernel)
3. SVC with RBF kernel
4. SVC with polynomial (degree 3) kernel

(c) (10 points) Feature Selection and Model Training: Train each SVM Model with the following feature combinations to predict admission.

1. CGPA and SOP
2. CGPA and GRE Score
3. SOP with LOR
4. LOR with GRE Score

(d) (10 points) Result Visualization: Visualize the decision boundary for each model and for each input combination.
(e) (4 points) Result Analysis: Just by looking at the figures you generated before, answer the following question. Which of the feature + kernel combinations did give you the best result?
(f) (4 points) Result Postprocessing: Were there any outliers in the data? If yes, please explain how we can use a one-class SVM to detect them. (No need to implement/code)