ECE 5460:
Image Processing
Fall 2023
Ertin

Final Project

Due: Thursday 15:00, December 14, 2023 There is no final examination for this class. In lieu of the final you are asked to complete this project. You could work in teams of 2-3. Each member should submit the report (noting the name of their group members.)

We recommend you use the basic Python notebook discussed in the tutorials. You can run it locally or on Google Colab. To use Colab, upload it to Google Drive and double-click the notebook (or right-click and select Open with Google Colaboratory), which will allow you to complete the problems without setting up your own environment. Once you have finished, make sure all the cells are run before downloading the notebook).

Submission Instructions: Please submit two categories of files on Canvas: (1) the Python jupyter notebook with the relevant source code and with all the cells run, and (2) A typed !! brief(latex, word, etc) report saved as as a pdf file that includes all input and output images and a brief description the processing details and description of each input/output image.

Late Submission Policy: No late projects will be accepted, if you anticipate issues meeting the deadline, please contact me before the deadline to make arrangements.

In this exercise you are going to train a multilayer neural network classifier for the CIFAR10 dataset and interpret the resulting classifier. A comprehensive notebook that implements a convolutunial neural network given, feel free to use as many parts as you like.

1. Try improving the classification performance of the multilayer neural network classifier. You should be able to come close and exceed 65% accuracy. You should try *at least* **one** of the following ideas. Feel free to combine and add your own ideas.

    - Modify the network architecture. (e.g. try adding another ReLU layer before the final linear layer, or change the number of convolutional filters).
    - Try to increase the size of your training data through *data augmentation*. That is taking your training data and use transformations to get new training data images. For example, you could flip, rotate images, or shift them by few pixels left/right up/down(and appropriately pad them), apply homographies to tilt them slightly, change hue,contrast, brightness slightly, etc.
    - If you are adventurous implement a pre deep-learning era classifier that we discussed in class. That is instead of working on the pixels directly, compute some engineered features and then feed those features to the linear classifier we used in Project 4 . A powerful set of features is given by HoG (Histogram of Gradient Features) that we discused in class. You don't have to code this from scratch, scikit-image provides a full implementation. Values `pixels_per_cell=(8, 8)`, `cells_per_block=(2, 2)` and `orientations=9` should give you a descent set of features to try .

2. Tune the learning parameters to achive better validation performance. Experiment with different learning and momemntum parameters. Optionally consider L2 regularization we discussed in class, that is implemented by the `weight_decay` parameter in `torch.optim.SGD`.

3. Report final classification performance for each class and compare it with the performance you achieved in Project 4. I will tabulate the scores and announce the team that achieved the best performance for each class. Teams that make the list will get bonus points.