CIS 6930/4930 Deep Learning for Computer Graphics
Dr. Corey Toler-Franklin

Course Project
DUE: October 5th, 11:59 pm Part I
DUE: October 19th, 11:59 pm Proposal
DUE: October 23rd, 11:59 pm Part II
DUE: November 15th, 11:59 pm Midpoint Evaluation
DUE: December 9th, 11:59 pm Part III

Overview

Collaboration Rules: The entire course project may be completed in groups of one, two or three. Groups are strongly encouraged. The complexity and effort required for the third part of your project should be commensurate with the number of people in your group. Thus, a larger group will be expected to have projects where Part III is larger in scope than a project with only one individual. Although you may discuss concepts between groups, each group must submit their own original code.

Deep learning is becoming more prevalent in computer graphics applications. This course covers deep learning algorithms that are popular in computer graphics (CG). In this semester long project, you will implement a system that incorporates the key theoretical deep learning concepts presented in the first part of the course and apply them to solve a CG problem. We will cover applications of deep learning in CG in more detail in the second half of the course. Throughout the course we will also discuss research papers at the intersection of deep learning and CG.

This document provides an overview of the project scope and expectations. The project is divided into three parts which increase in complexity as your knowledge of the topic progresses. You will be provided with additional details and specific instructions for Parts I, II and III in subsequent documents. Within the structure of this project, you will have the ability to be creative by customizing your final deep learning CG application in Part III.

Details

Part I: In the first part of the project, you will evaluate the performance of classifiers and regressors on a simple game of *Tic Tac Toe*. Your single classifier program will generate statistical accuracy and confusion matrices for several classifiers and regressors (linear regression, linear SVM, k-nearest neighbors, multilayer perceptron) using provided datasets. Each game play session will be synthetically generated. The two players in your games are both AI players. Your program will learn from these games to produce reasonable player moves for one of the players in the game.

Part II: In the second part of the project, you will start to work with CG concepts while exploring several fundamental concepts in deep learning. You will train a CNN regressor to add color to images. The process converts a grayscale image to a colorful version by learning global and local image priors. The goal is to generate a script that will train and test an automatic deep image colorization model. You will learn about perceptual color spaces, chromaticity and denoising. You will also have to make decisions about the design of your network architecture and resolve issues related to overfitting. The final implementation will be optimized to improve overall learning performance while creating different color effects.

Part III: In the third part of the project, you will use your system to create a CG application of your choice. To narrow the project scope, you will focus on one of two application categories; analysis or synthesis. Recall from class that analysis approaches examine and refine data (e.g. smoothing) while synthesis algorithms produce new information (realistic image synthesis from neural networks). Some examples might include implementing an image editing tool that creates new faces from a collection of portraits. Your system will have to learn the best fit features for creating a new face. Any deep learning algorithm that synthesizes realistic images from observations is also a good choice. Using a convolutional neural network to denoise images is another popular recent example. Painterly rendering of curved brushstrokes is another example but would require some non-trivial additions to make the project complex enough for a final project. You will be provided with publication references to help you make your decision. Implementing aspects of SIGGRAPH papers is a good place to start. We will discuss different project examples in class. A word of caution: Part III is more complex than parts one and two combined so please do not wait until the last minute to get started.

Proposal: You are required to submit a project proposal (1-2 pages) to describe the work you will do in Part III. The proposal requirements will be posted to canvas. Requirements will include a description of your project, a schedule for completing key components, an explanation of how the work will be distributed among team members, expected results and a list any third-party APIs and datasets you will use. You must also include related references (research papers, websites). I will evaluate each team proposal and provide constructive feedback for how to scope your work to be successful.

Midpoint Evaluation: Each team will submit a 5-7 minute video screen tour (just like the ones submitted with Parts I and II) at a mid checkpoint. The goal of this submission is to verify you are making progress on your project. I would like everyone to have enough time to address any issues before the final submission due date. You will be evaluated on your progress, your code thus far and your initial results.

Final Write-Up: You will be required to submit your code as well as a written technical write-up. Graduate students are encouraged to use latex and the ACM Tog paper template (but this is not a requirement). Rather than summarize your project in a report as you did for parts I and II, you must follow the techniques of technical writing which we will cover. Each write-up should have an abstract, introduction, previouswork, results and analysis and conclusion section. The main body of the paper should have sections appropriately titled to discuss the technical details of your project. Your paper must have a teaser, at least one table and examples of results computed directly from your system. In addition, you must incorporate a minimum of one of each of the diagram types discussed in class - analytical, illustrative and didactic.

Virtual Presentation: Due to the class size and virtual format, we will not have final class presentations. Instead, each group will record a 20-minute presentation to be included with your submission. Throughout the course, I will provide you with tips on how to present your project.

Grading: Each part of the project is worth 100 points. You will receive an explanation of how grading points are distributed with each of these assignments. In general your work will be evaluated based on the following criteria (1) source code correct-ness and completeness (2) complexity (more challenging projects will be rewarded) (3) novelty and strategy for approaching the problem (4) results (5) recorded presentation or video screen tour (6) write-up.

Code Base and Resources: You will be provided with direction for how to set up your systems and how to access UF resources to complete your project. Parts I and II of the program will be developed using Python, Torch, the scikit-learn module, LuaJIT and related dependencies. For part III, you may use any API you choose. However, be cautious about using APIs with high-level implementations. Projects that simply call high level functions without non-trivial implementation or evaluation strategies will be down-graded significantly. Your project proposal submission is the time to verify your approach. See me if you need datasets or advice on API frameworks or debugging interfaces.

Timeline

OUT: September 21st Part I Classifier and Regressor Performance
DUE: October 5th Part I Classifier and Regressor Performance
OUT: October 9th Part II Deep Learning for Image Editing
OUT: October 12th Part III, Proposal Instructions, Midpoint Evaluation Criteria
DUE: October 19th Proposal
DUE: October 23rd Part II Deep Learning for Image Editing
DUE: Novemeber15th Midpoint Evaluation
DUE: December9th Part III, Final Submission

Submitting

Upload one zip file to Canvas. If your zip file is too large for a canvas upload, you must submit your written report to canvas by the due date and upload your zip file to a file share provided by your instructor by the due date. All submissions (Canvas or otherwise) must be completely uploaded by the due date and time. Parts I and II of the course project are subject to the late policy for assignments. No late submissions for Part III will be accepted.

Your submission should include everything needed to test, run and understand your code including:

- The complete source code including any third party libraries.
- An example of your dataset that can be used to test your code. For projects with large datasets, a link to your dataset is acceptable. This may be a publicly available dataset or a dataset you have hosted on a server or other site that is accessible to the course instructor.
- All input data (images and meshes are examples) used by your program (in the data folder in your zip folder). This refers to any input needed to run your code that is not included with your dataset link. For example some programs need images icons or other support data to run out-of-the-box.
- A written report. The size and requirements of your report will vary for each part of the project. You may be asked for specific results in Part I that are not applicable to Part II for example. In general, each report will include:
  - An explanation of your implementation.
  - Your results (screen shots, tables, graphs).
  - Instructions on how to run your code.
  - Anything you would like us to know (bugs, difficulties).
- All of your original results in a subfolder of your zip file named results. For example, although you may include a screen shot of an image your program generated in your written report, you must also include the original image with your submission.

- 5-7 minute video screen tour walking us through your code and explaining your implementation. It should be located in a folder in your zip called videotour. You should explain your code and step through the key parts of your code. You also need to run the system so that we can see the results being generated.