**CSCE 823: Machine Learning**

**Summer 2019**

**HW3**

Due Wed, 31 July at 2359

Submit via Canvas

**(**This Homework is worth 10 points toward your final grade**)**

This homework assignment focuses on classifying the 10-class small-image-size CIFAR dataset. (for details, see <https://www.cs.toronto.edu/~kriz/cifar.html> ). The dataset is part of Keras and a loading utility is already provided in the HW3 codebase. Your goal is to build, document, train, and evaluate a deep learning model of your design to classify the images.

In this assignment there are additional components to incorporate in your ipython notebook. Include a literature review & architecture/design description and rationale for all of your decisions. Your submitted ipython notebook will include these things and will be set up (using global Booleans) such that it *loads your saved model and uses your model for prediction and model performance comparison* (**BUT NOT TRAINING**). Your full model training process can be accomplished in a separate notebook(s) or directly in through python (.py) code if you wish. All code, notebooks, images, data and saved models and written documentation should be provided in a single directory (possibly with subdirectories as needed) and zipped before submission.

**Provided Codebase vs. Student Code**

Some starter code is included for you, including utilities for loading data, saving and loading models, and showing confusion matrices. Two CNN starter models are also provided. Explore the existing code before building new code. This code is provided as-is. You are free to modify the existing code (except the structure of the provided models), but if you do, please provide comments in the code and summarize the changes & reasons for the changes in your writeup.

**A Note on Training Time**

Some small sample network architectures have been included for you to explore. Training time on a decent GPU for a small network might take 30-60 minutes. Note that you don’t *need* to use a GPU for this assignment, but training time may be VERY long if you don’t have one – so plan accordingly. Since you will be exploring architectures, you will likely be iterating coding different architectures and training/evaluating them. Keep in mind that you can train multiple architectures in sequence and save the models under different filenames, then load them later – this might be a way to train several architectures when you are doing something else (like sleeping).

**Student Model Submission**

In previous homework assignments, you provided code for models which the instructor ran to train and evaluate your model. In this HW, you must submit at least one **trained** model which is your best-performing model (the instructor won’t have time to train your untrained models). This model can be saved using the provided utilities. This best model should be named “keras\_cifar10\_LASTNAME\_FIRSTNAME\_final\_model.h5”. The instructor will run evaluation code on your trained model to confirm its performance. Optionally - You may submit other models, but the instructor may not have time to evaluate them.

**Student Tasks**

Each step listed below should correspond to code and/or text in your file. Make it easy for the instructor to find your work by using the step identifiers in both your code and your writeup. Use keras with tensorflow backend on this assignment to make it possible for your instructor to grade the HW. You will also need packages hdf5 and h5py installed in your environment to load and save files even though they don’t appear as imports.

1. Existing Architecture Documentation: You have inherited 2 Keras CNN architectures (shallow, deep). Your goal in this step is to document the models. Examine the layers of each model in the code (using keras’s tools like .summary), and develop a diagram (e.g. using Powerpoint) of the layers like you’ve seen in class or on the internet. Save the diagram as an image that can be imported/displayed in your notebook with a code cell near the model’s code. In a markdown cell, describe the architectures and their parameterization. How many parameters are there? What other features are present in these models? How do they differ? What is your conjecture about their expected performance on the CIFAR10 dataset?
2. Training Modification: Modify the provided training code to enable recording and examining additional information per epoch such as training loss and accuracy and validation loss and accuracy. Implement early stopping based on the validation set. You may need to use Keras callbacks in this step. If you are making decisions using the validation data (i.e. early stopping), remember to not use the same validation data to also evaluate the model’s performance.
3. Existing Architecture Training Check: Using the provided code, separately train each of the provided Keras CNNs (shallow, deep) on the CIFAR10 dataset on a **small** number of epochs (e.g. 10). Make sure data\_augmentation = False. Confirm that training works. Note that you do not need to show evidence of this step in your notebook.
4. Existing Architecture Performance Evaluation: Using the provided code, train and evaluate the two provided Keras CNNs (shallow, deep) on the CIFAR10 dataset on 200 epochs. Make sure data\_augmentation = False. Produce and discuss the training and validation curves. What can you tell about these models from these curves? Do they have enough capacity? Were they trained long enough? You may desire to increase the number of training epochs beyond 200 if you have time and the training performance was still improving. Report standard classification metrics including loss and accuracy as well as a confusion matrix of performance on the test data. Compare the performances of these simple models with best architectures on CIFAR10 (state of the art performance): <https://en.wikipedia.org/wiki/CIFAR-10>
5. Literature Review: Using the Wikipedia link listed above, and/or a literature search you conduct, locate and discuss **at least 3 papers** (conference or journal) which provide *different* approaches to tackling classification on the CIFAR10 dataset and have very good performance. In a markdown cell in your ipython notebook, compare and contrast their architectures, data augmentation, training methods… and rationale for the researcher’s choices. One or more paragraphs per article is satisfactory. Include images or use other ways to describe the architectures. Compare/contrast model performances. Emphasize ideas that you plan to adopt for your own architecture in the next step. Include properly formatted bibliography entries for each of the references (either IEEE or APA format).
6. New Architecture Design & Documentation: Design your own architecture. Your architecture can be *inspired* by other architectures you might find on the web, but it should be your own. Consider using/altering the various layers (e.g. conv2D, activation, pooling, dropout) as well as the optimizer selection and settings. You may want to explore several different architectures and/or other improvements for training, including changing the optimizer, and using the data augmentation (data\_augmentation = True ) built into the training code. You can also modify the way the generator augments data. Cite any sources that inspired your design. Develop a diagram (e.g. using architecture display tools or Powerpoint, saved as an image that you load via a code cell) of the layers like you’ve seen in class or on the internet. Document your design and your decision rationale. In a markdown cell, describe the architecture and their parameters. How many parameters are there per layer? (hint: use model.summary()) What other features are present in these models? How do they differ? What is your conjecture about their expected performance on the CIFAR10 dataset compared to the instructor-provided architectures? How do you think they will do compared with state-of-the-art performance? Code your model in Keras in the provided location.
7. New Architecture Training Check: Train your new Keras model on the CIFAR10 dataset on a small number of epochs (e.g. 10). Make sure data\_augmentation = False. Confirm that training works (training loss dropping over epochs) and track how long training takes. You may need to adjust your model architecture based on the time to run a small number of epochs since your actual training will take 200+ epochs (don’t build something that you don’t have time to train). Note that you do not need to show evidence of this step in your notebook.
8. New Architecture Performance Evaluation: Train and evaluate your architecture on the CIFAR10 dataset on at least 200 epochs. Make sure data\_augmentation = False. Examine the training and validation curves during training and include them in your document. What can you tell about these models from these curves? Do they have enough capacity? Were they trained long enough? Report standard classification metrics including loss and accuracy as well as a confusion matrix. Compare your performance to both the instructor-provided models and the state-of-the-art performers. Compare your trained model to the instructor-provided model(s) and the model you implemented from the literature (be sure to use the data augmenter to train each model the same way to have a fair comparison). Document the results of the comparison in your report.

**Rules of Engagement for this Homework Assignment:**

**Using external sources:**

The use of pre-existing solutions to answer assignments is not allowed. This includes the use of other students’ answers, solution manuals, and any other source of information which does not reflect your own work.

You may get inspiration from the internet (see above) but your work should be your own. Cite all sources you used.

You may use the internet or get help from peers when determining basic things like “how do I add points to a plot in python?”

You may use any pseudocode or concepts learned in class to solve the problem.

The code & documentation you write must be original work unless otherwise specified by the instructor.

**Programming Conventions**

In the code chunks, good software engineering principles apply: self-documenting code (meaningful function & variable names), additional comments and whitespace should be the standard for all code you turn in. If your code is not understandable it may not receive full credit.

IMPORTANT: In your python notebooks you should explain what you are doing in text as well as in the comments to a code cells. A rule of thumb is to have line-level comments in the code chunks and save the larger high level comments & observations for the text in your homework writeup. Include step numbers as comments in your code to make it easy to locate them. Make mention of the names of your function names in your writeup so it is clear how you are using them

**Code & Model File Structure and Naming Conventions**

Ensure that your document, ipython notebook, python code, model and any other files are contained entirely in **one directory** (possibly with subdirectories). Your main homework code file(s) should be python code with the name: “LASTAME\_FIRSTNAME\_HW3.ipynb”. Your final CNN model should be named 'keras\_cifar10\_ LASTAME\_FIRSTNAME \_final\_model.h5'. If you write functions which get called your notebook, the name of the function will specify the filename.

**Document Naming Convention:**

Your written document should be a MS Word or PDF file with the name: “LASTAME\_FIRSTNAME\_HW3.XXX”, where XXX is docx, doc, or PDF.

**Pre-submission Checklist:**

**Code/Data/Model:** Ensure all of your code, data, and model is located in a single directory (you can put your document in the same directory). Avoid using absolute pathnames when loading data in scripts as your pathname will not be the same as the instructor’s pathname.  
Perform a final preflight code execution check by clearing your variables and figures, then ensure that your code still runs (due to time-constraints you should not re-train your model though…).

**Document:** Make sure you proofread your document carefully to ensure your final product reflects what you intend to submit. You don’t need a separate title/name page, but make sure your name is easy to find in the text of the document (ex: first line, or in the header). Include page numbering on your document.

**What to Submit: Your group should have one person submit the group’s deliverable.** Submit your **zip** file of the folder containing your ipython notebook and your model save files to Canvas.

Note that if you discover an error and change a problem solution and re-submit, keep in mind that your instructor will only review your *latest* submission on Canvas – make sure it is complete.