# Summary

For your final project you will demonstrate mastery of the machine learning techniques you have learned throughout the course as applied to a machine-learning problem directly related to your thesis research. Each student will complete their own final project guided by the 7-step machine learning work-flow described in Chapter 4 of the Chollet textbook. The final project will use either real or simulated data to demonstrate learning in an area related to the student’s research area. The final project will be structured in such a way as to be scalable, so that as you make progress in your research (either getting real data for the first time or greatly scaling up the data you do have) you will be able to have your framework handle the quantity of data needed.

# Topic Area

Ideally, each student will do a project directly related to their thesis. There are reasons that this may not be possible for every student. In this case, students will use a substitute process with the same type of raw input data they expect to see with their thesis and use a similar ML architecture as they plan to use in their thesis.

# Week 3 Deliverables

In the third week of the course, each student will meet with the instructor one-on-one to discuss their topic area as well as their path forward to obtain data. This will consist of a 10-minute conversation with the instructor (student may use a few slides to assist if needed). At the end of the meeting, the instructors will either approve or disapprove of the project.

In the meeting, students will discuss:

1. What is your topic area?
2. What are the inputs and outputs for your problem?
3. How will you get data for your project?
   1. If collecting your own data, is it feasible you will actually get enough data in the next few weeks to complete the project?
   2. If utilizing an existing dataset, are you sure the dataset contain everything you need (is it a *labeled*) dataset?
   3. If simulating, what are your methods to create synthetic data? It is high-enough fidelity to actually be worth pursuing? Do you have a path forward (someone else’s simulation code?) or do you need to write your own simulation software?
4. What is the ANN architecture (Dense, CNN, RNN, 1D ConvNet, etc.)?
5. What will your baseline to compare against be (also include possible metrics)?
6. Are you all set to go from a hardware point of view?

# Final Project Deliverables

There will be three main final deliverables for this class final project.

1. A paper in the style of a journal article. See Canvas for the rubric on this paper. The paper should have minimal introduction/background and focus on the methodology. What you did to process/split your data, what was your ANN architecture and what did you do to train the ANN. Results should include any losses and metrics for your problem.
2. All the software for the project and all required data, correctly configured to run on the established course image so that it can easily be reproduced.
3. A short ~10-minute presentation that will be recorded. For this year, the recording will be submitted the same day as the paper, and the class will coordinate to view them after submission. The presentation should go over the topic, input/output data, the ANN used and the final results.
   1. The presentation videos will be watched in an “academy awards” fashion, with bonus points being given to the presentations with the best:
      1. Visual Affects
      2. Unique Problem set/ Solution
      3. Best Pilot Retention Solution (specific to just this year)

All deliverables can be uploaded to the GitHub Classroom for submission. Data should not be uploaded to the github and instead should be given to the instructor as a hard drive or uploaded to the ANT Center NAS on the CDN. Alternatively if the data is publicly available please explain where the data came from (include URLs).

# Creating Datasets

One of the most challenging issues for the final project will be obtaining a dataset. Most students are just starting their theses, and few have real data.

There are three major paths forward for obtaining your data.

1. Collect your own datasets needed for your thesis. Self-explanatory. This is the best option but also potentially the most difficult and time-consuming. It may require equipment that is not purchased yet, etc.
2. Utilize an existing dataset to show something original related to your thesis. For example, there are a large variety of existing imagery datasets at AFIT. You could provide this image dataset to an ANN in order to demonstrate something new – note you *cannot* take a dataset that was collected for a specific ML task and recreate the same task, you must do something original. This link may help: <https://towardsdatascience.com/top-sources-for-machine-learning-datasets-bb6d0dc3378b>
3. Simulate your data. Note that if you simulate your data it is unlikely the ANN will actually generalize well to real life. This is ok as long as your entire framework and approach is reasonable – the idea is that once you get real data it will require minimal changes to your overall workflow, and you should be able to drop in the real data as a replacement for the simulated data. Note that some things are *much* more difficult to simulate than others.

Requirements

Points will be deduced for each significant-requirement violation.

* Must employ parallel processing in algorithm
* Must store code in a Git repo (such as either git.antcenter.net or GitHub)
  + Must configure Git repo to execute your unit tests on code changes
  + Must have Git repo successfully run your unit tests for all capabilities that do not have bugs
  + Must use a .gitignore file with the proper configuration of the programming language used
* Must specify dependencies through a programming language standard, such as through a requirements.txt file or the newer pyproject.toml file
  + See [https://packaging.python.org/en/latest/tutorials/packaging-projects/ Links to an external site.](https://packaging.python.org/en/latest/tutorials/packaging-projects/)for advanced options to organize files for distribution
* Must demonstrate implementation with three datasets (two test datasets, one challenge dataset)
* Must allow hyper-parameter specification from users of module
* Must be free of major bugs that prevent itself from fulfilling purpose
* Must comment or document all unresolved minor bugs
* Must work with other datasets that may not have the same number of variables or types of variables
* Can review existing implementations of the algorithm to get ideas on how to implement
* Can use the following Python modules without restrictions: numpy, pandas, multiprocessing, mpi4py, numba, sympy
* Can receive extra point by ensuring algorithms works in a distributed manner
* Must present the following sections and content within a Readme.md file:
  + Algorithm Purpose
  + Hyperparameters
  + Background
    - History
    - Variations
  + Pseudo code
  + Example code to import and use module
  + Visualization or animation of algorithm steps or results
  + Benchmark Results
    - Comparison of efficiency and effectiveness
  + Lessons Learned
    - Such as new code snippets to support some computations
  + Unit-testing strategy
    - What steps of the algorithm were tested individually?
    - Code-coverage measurement
* Must present your Python module to the class and background on how the algorithm works, approximately 10 to 15 minute presentation