# CS230 Hands-on session 10: “Grading criteria / examples of great projects”

Kian Katanforoosh, Andrew Ng

Fall 2018

The goal of this session is to give you an idea of what makes a successful CS230 project.

## Part I: Grading criteria for final project

Here are the grading criteria we use to grade the final report:

**Problem description (Why the problem matters, (limitations of prior work))**

**Description of the dataset**

**Hyperparameters tuning & Architecture search**

**Paper writing**

**Explanations of choices and decisions (architecture, loss, metrics, data)**

**Data cleaning and preprocessing (optional)**

**How much code you wrote on your own**

**Insights and discussions (including next steps, and interpretation of results)**

**Results: Accuracy (or other metric) satisfaction**

**References**

**Penalty for more than 5 pages (except References/contribution/theory-proofs)**

Other tips:

- The choices you make in the Hyperparameter tuning phase should be explained and justified. A short sentence is oftentimes enough.

## Part II: Grading criteria for poster presentation

You will be asked for a 3 minutes pitch. A good pitch includes:

- Brief introduction of the problem that you’re trying to solve, including the dataset used. You should explain what makes your project useful/interesting.

- The models that you tried on this problem
- Which models gave the best performance/ any insights that you had
- Next steps for the project

The 3 minutes pitch will be followed by 2 minutes of questions and answers. Be prepared for this.

Other criteria which matter:

- Poster organization/ outline

## Part III: Reporting results

- Evaluation metrics for various tasks: Precision, Recall, F1, mAP, etc.
- Class imbalance: Let's say you have 10 patients. 9 patients are normal and 1 has a disease. If your classifier predicts everyone as normal, the accuracy is 90%. However, this is not a useful model. This shows that accuracy is not a good metric for this task.)
- Compare your results to the SOTA (state of the art) for your task. For instance, remember what YOLO reported about Fast RCNN, etc. https://arxiv.org/pdf/1506.02640.pdf
- Train / Validation / Test split.
- Illustrate results qualitatively. Ex: generating synthetic images, generating text.
- It is good to show failure cases.

## Part IV: Organizing your Github repository

When grading your project, we will also look at your code. Please add **`cs230-stanford`** as a collaborator on Github if the repo is private. Your github repository should be as organized as possible and folders' names should be relevant. Here is an example of organized repository: https://github.com/cs230-stanford/cs230-code-examples/tree/master/tensorflow/vision

**Readme**. You should have a nice Readme. The Readme should contain instructions about how to run your code.

**Dataset**. If it is small (<100MB), the dataset can be stored on the repository. Otherwise, please include at least a few examples and labels as well as a link to the dataset. If you can't share the dataset with us, because of privacy issues, please indicate it in the Readme.

**Problem description**
- **Why the problem matters**
- **Limitations of prior work (Literature Review)**

**Description of the dataset**
- **Class imbalance**
- **Describe features, show examples**
- **Describe the difficulty of dataset**

**Hyperparameters tuning & Architecture search**
- **Hyperparam Tuning: learning rate, number of layers… (explain why you choose these hyparparams to tune)**
- **Architecture Search: ResNet vs. VGG vs. your own network, Adding Attention etc. (recommend 3)**

**Paper writing**

**Explanations of choices and decisions (architecture, loss, metrics, data)**
- **E.g. Dataset small, so do transfer learning**
- **E.g. We faced an optimization problem and therefore we tuned learning rate**
- **E.g. Class imbalance, therefore we do negative sampling**

**Data cleaning and preprocessing (optional)**
- **Scraping data**
- **Modifying dataset to fit into model (e.g. time series data -> modifying it to work better for your model)**
- **Data Augmentation**

**How much code you wrote on your own**

**Insights and discussions (including next steps, and interpretation of results)**
- **Error Analysis**
- **Reasoning why a certain model works better than another**
- **Visualizing CNNs (saliency maps, occlusion sensitivity) -> Chexnet**

**Results: Accuracy (or other metric) satisfaction**

**Results should make sense (e.g. better than random)**
**Evaluation metric and loss function should make sense (F1 score, precision, recall)**

**References**

**Penalty for more than 5 pages (except References/contribution/theory-proofs)**