The vision task we will play with for the final project is **object detection**. While object classification only tell if a certain type of objects exist in a given image, object detection needs to further search for the specific locations of the objects. Detection is conceptually more difficult than classification, but more practically related with a lot of application, e.g. tracking, pose estimation.

**Data and Evaluation Metrics** You are expected to work on PASCAL VOC 2012 dataset, which is originally created for a challenge of recognizing objects from a number of categories in realistic scenes. The challenge is end nowaday, but the well annotated dataset is still a widely using benchmark for evaluating and comparing different methods. There are a variety of tasks proposed in the PASCAL VOC, however we only focus on object detection on the 2015 version of challenge. You can find all the information from [http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html](http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html).

In PASCAL VOC 2015 object detection challenge, the algorithm should output bounding boxes (rectangle region aligned with image axis) that it believes to tightly contain an object belonging to a specific category. A bounding box reported by an algorithm is considered correct if its area intersection over union with a ground truth bounding box is beyond 50%. If a lot of closely overlapping bounding boxes hitting on a same ground truth, only one of them is counted as correct, and all the others are treated as false alarms. After running on a bunch of images, an algorithm is majorly evaluated by two metrics:

1. **Precision**: The number of correct reports divided by the total number of reports from the algorithm.
2. **Recall**: The number of ground truth that have been hit by algorithm divided by the total number of ground truth.

Intuitively, precision checks how many reports are accurate, while recall measures how many ground truth can be found by the algorithm. There is usually a tradeoff between these two factors depending on how many bounding box generated from the algorithm. Conceptually, if we pursue high recall, we should encourage the algorithm to generate more boxes such that we have a larger chance to cover more ground truth. However, the precision will be hurt because of more inaccurate boxes. So a same algorithm can work on multiple precision/recall with different system setting, and all the precision/recall value pairs form a curve. The average precision is defined as the area under the curve, and a good system would have a comparatively larger average precision. Average precision is defined for one category, and the mean of multiple categories, i.e. mean average precision, mAP, is the ultimate metric measuring the overall performance of an algorithm for object detection. **PASCAL VOC has provided you a well documented development toolkit to calculate these metrics**.
**What to do?** You need to design and implement an object detection system trained and evaluated on PASCAL VOC 2012 dataset. PASCAL VOC 2012 contains 11,530 images on which objects from 20 categories are annotated with bounding box. It also provides a split to divide all the images into training and evaluation set. Following the tradition, your algorithm should train on the training split and test on the evaluation split. The training procedure should not see any of the image in the evaluation set, and we take the mAP on the evaluation set as the final performance of your whole detection system. Actually, such experiment setting is not ideal in that it provides a chance to over-tune the performance on the evaluation set. To be absolutely fair, algorithm should be evaluated on a testing set with ground truth unavailable to public, in which the images are not seen in either training or evaluation set. PASCAL VOC 2012 also provides a testing set with only images but no ground truth, however evaluating on it requires you to submit your result on to the official evaluation server, which is a little bit troublesome. **We only need you to submit your performance on the evaluation set.** However, if your performance is competitive with the state of the art, we strongly recommend you to evaluate your result on the testing set. If your method ranks high on the official leaderboard, you will gain extra bonus.

There is no limitation and requirement on how to design your system. You can make up your own idea and implement from scratch, or start from previous works and do refinement. You are allowed use the source codes of previous works as a part of your system. But you must cite the work and clearly mention which parts come from previous works and which parts are implemented by you. We will mostly focus on the part of codes written by yourself for the grading. Here are some baselines and previous works, from naive to complicated, you may want to start with:

1. No matter what the image is, always predict a fixed bounding box at the center of the image as a “cat”.
2. Perform a object classification as we did in PS5, and generate a box based on some statistics learned from training set.
3. Create a window sliding in the images and evaluate object existence for each window location. Deformable part model is such a model: [http://www.cs.berkeley.edu/~rbg/latent/](http://www.cs.berkeley.edu/~rbg/latent/).
4. Extract regions from image and perform classification on each region. RCNN is such a model: [http://arxiv.org/abs/1311.2524](http://arxiv.org/abs/1311.2524)
5. Here are more recent works, which was pushing the state of the art performance. You are recommended to try this methods:
Computational Resources Students from department of computer science can use the IONIC server, which contains 500+ cpu nodes and 2 gpu nodes. All the students can use machines in Friend Center 016/017.

Deadline, group, report You can form a group of 1-3 people to work together on the final project. You are required to submit a one page project proposal, including your group members and a brief introduction about your method before Dec 18, 2015. The deadline for submitting the source codes and the final report is the midnight of Jan 12, 2016.

Each group need to write a report in CVPR format. You can find the template (LaTeX/Word) here: http://www.pamitc.org/cvpr16/author_guidelines.php. The report should include sections about:

1. Group members and student IDs
2. Detailed explanation to your method. In general, you should talk about training, testing, post processing, and any engineering hacking to achieve the final performance. If your work is based on previous works, you should clearly state the modification made upon the previous works with proper citation.
3. Evaluation of the method. You should report your performance on evaluation set. This includes
   - Average precision on each of the 20 categories, and mAP over 20 categories.
   - Precision/recall curve of each category.
   - A visualization of detection results on some images, both successful and failure case.
   If you want to fight for extra bonus, you should write down your performance on the testing set and include a screenshot from their leaderboard.
4. Discussion. You can discuss any observation or lesson learned from this final project. Any extensive analysis on data, algorithm, or result is highly welcomed.
5. Job assignment. You should clearly mention what is borrowed from previous works and what has been done by each of the group member. People in a group may not get the same mark based on what they have done individually.

We prefer concise report. You should try to control the length of your report within 4 pages. To the most extreme case where valuable contents are hardly to be fit in 4 pages, your report must not exceed 6 pages. Any material beyonds 6 pages will not be considered for grading.

Besides the report, we require each group to submit a runnable source code package. You should build your code upon the development toolkit which is provided on the challenge webpage: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html#devkit. It greatly facilitate your development by providing you a lot of useful functions, e.g. loading dataset, standard evaluation. In your source codes, there should be two scripts to perform training and testing respectively following these instruction:
1. A script named “script_train.m” takes no input for training. The code should load PASCAL VOC 2012 training set, and whatever external data if any, and eventually output a model saved on disk.

2. A script named “script_test.m” takes no input for evaluation. The code should load PASCAL VOC 2012 evalutaion set, the model from the training code, and then perform testing on the evaluation dataset. The code should eventually output all the numbers, figures, and visualizations shown in the report.

**Grading** The final project will be grading according to the following criteria:

- **Implementation**: 40%. We will evaluate the amount of work that has been done by you. We will mainly check how many coding work, data/result analysis/visualization has been done. Reimplementing a previous paper is counted as your own work. However, any published source codes will not be considered during the grading. So in the most extreme case where you simply run some open sourced code, you would not get high score for implementation even your performance is high. In order to gain good implementation marks, make sure that you clearly mention in your report about what is done by you. If you decide to use deep learning as a part of your algorithm, you are strongly recommended to use Marvin.

- **Correctness**: 20%. Your codes should not contain obvious bugs, and interact properly with the dataset. You implementation should perform fairly well and reach an AP above 20% to be considered correct.

- **Report Writing**: 10%. Your report should be written in the right format and cover all the above mentioned aspects. If anything is borrowed from previous works in your report, it must be cited properly.

- **Code Clearness**: 10%. Your code should be clean, well-organized, and easy to read and understand.

- **Algorithm Novelty**: 10%. Although you are allowed to mostly re-implement upon some previous works, you are strongly encouraged to propose completely novel methods or do significant improvement on previous works, say 5% AP improvement. Again, you would get nothing for novelty if merely run a previous work.

- **Performance**: 10%. We will rank the performances (mAP on evaluation set) of all the groups. The group ranked top get the full performance marks.

- **Extra Bonus**: If your performance is very competitive, we encourage you to submit your result to the official evaluation server: [http://host.robots.ox.ac.uk:8080/](http://host.robots.ox.ac.uk:8080/) and get your performance on the testing set. If your method ranks high on the official leaderboard, we will give you a considerable amount of extra credit depending on your performance. For your reference, you should be able to get an AP at least higher than 70% to get the bonus.

**Submission format:** Any one in the group can submit on behalf of the whole group. You need to submit two files: one PDF file for the report that contains your names, Princeton NetIDs, all the aspects mentioned above; one ZIP file (not RAR or any other format) that
contains all source code for your system. Both the PDF and ZIP file should be named using the Princeton NetIDs of the group members underscore cos429fp. As an example where Professor and TAs form a group, they should be named “xj_yindaz_mingru_cos429fp.pdf” and “xj_yindaz_mingru_cos429fp.zip”. To verify your result and detect plagiarism to make sure there is no cheating, we will use an automatic program to run your code and compare your code with other groups’ and public available implementations (e.g. from Google, Bing, Github). Therefore, please follow the file format to make our grading job easier. Failure to follow these rules will result in losing your grade.