Final Project

Computer Vision and Deep Learning
Håkon Hukkelås
hakon.hukkelas@ntnu.no

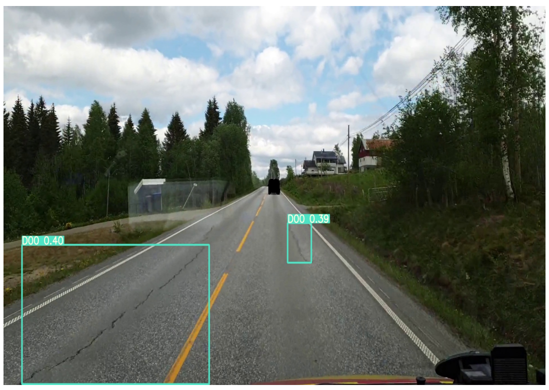
Delivery Deadline:
Sunday, April 25th, 23:59PM.
Presentation Days:
Monday 26th and Tuesday 27th of April.
This project counts **50%** of your final grade.
You are allowed to work in groups of up to **3 persons**.

Introduction. In this project you will select between to different projects to work on. The first project option will be about detecting wear and tear on roads, where you train an object detector to perform road damage detection. The second project option will be about medical image segmentation for diagnostic of heart chambers. It will not be possible to select both projects.

1 Introduction

About. Roads are a vital infrastructure of a country. They require periodic checks and maintenance to ensure unhindered travel between cities and communities which is imperative for a functioning society. Given the limited resources, the decision of when and where to delegate them is very important. Traditionally road maintenance is done by manual inspection or through specialized cars which is costly. Given the modern technological advances in the field of computer vision and artificial intelligence, an intelligent solution can be built that can detect and classify the road defects. Cameras are cheap and a lot of surveillance data is already available. This solution will help the road maintenance authorities to make decisions and save time and cost on manual inspection.

In this project, our goal is to train a network that can detect different types of road damages from videos. This can be solved as an object detection task, where we require labed ground truth images (similar to MNIST in Assignment 4). Here, we will detect 4 common types of road defects, where their labels are given in Table 1.



Damage Type	Class Label
Longitudinal Crack	D00
Lateral Crack	D10
Alligator Crack / Complex Crack	D20
Pothole	D40

Table 1: Road Damage Types and their Class labels [Maeda et al., 2018].

To solve this task, we have gathered two datasets for you. A public road damage detection dataset from [Maeda et al., 2018] (called *RDD2020*) that is collected outside of Norway. To ensure that our model works well on Norwegian roads, we have collected our own dataset from Norway (called *TDT4265-Dataset*). Your task will be to build upon these datasets and solve the following subtasks:

1. **Annotate data collected from Norway.** The TDT4265-Dataset has no pre-defined labels. Therefore, it's your task (together with the rest of the class) to label the different videos. We have built a custom annotation server for this project. See section 2.

2. **Developing and building your model.** Your starting point will be the SSD model from assignment 4 and your task is to improve your model for data related to Road Damage Detection. See section 3.
3. **Evaluating your model.** The last step is to evaluate the model on our data. You will report the mean Average Precision on the test dataset and score your model. As a motivating factor, we have setup a public leaderboard to compare yourself against your classmates. See section 4.
4. **Documenting your approach**. In a real-world scenario documenting your approach is important. Either you're going to hand-off the project to someone else, or reporting about the amazing work you have done to your boss. For this project, you will deliver a short report and give a presentation about the project. See section 5.

Project Rules

To achieve a fair project (and grading..), we ask you to follow these rules for all submssions to the leaderboard on tdt4265-annotering.idi.ntnu.no/leaderboard [1].

1. The code should be developed by yourself (except what is given in assignment 4). Of course, finding inspiration from open-source repositories is allowed. If you take code from anywhere else, please attribute the original authors in your source code and write a notice in your report. It is not allowed to use open source repositories "out of the box", such as detectron2.
2. Annotating the test data by yourself and using it to train/validate your model is NOT allowed.
3. It is allowed to use any pre-trained **classification** model found in torchvision.models repository. Any pre-trained object detection (and segmentation) models are not allowed to use. If you plan to use tensorflow/keras, it should be possible to find equivalent models to those provided by torchvision.
4. It is not allowed to train your model on any other data except the data provided by us. However, it is allowed to use backbone networks pre-trained on the ImageNet dataset, but you are not allowed to perform this training yourself. This is to prevent that the winning solution is the one who use the most amount of data/compute.
5. Training your model should not take more than **12 hours** on the tdt4265.idi.ntnu GPUs (or the cybele/tulipan computers) **per dataset**. In total you can train your model for 24 hours. This means that you can train a maximum for 12 hours on the RDD2020 dataset and 12 hours on the TDT4265-Dataset.

If you are unsure if something is allowed, please post on piazza or contact the teaching assistant through email: hakon.hukkelas@ntnu.no. If we believe that you've broken any of these rules, we will train your model following the steps in your report, and validate that we achieve a similar mAP. Breaking rules will be considered cheating on the project.

Grading (50 points)

1. [4*pts*] **Annotating Data**: Your group is required to annotate a minimum of *three videos per person in the group.* Therefore, if you are a group of four, you are required to annotate a minimum of 12 videos to receive full score.
2. [8*pts*] **Model Development Requirements:** Note that these are not requirements for your final model, but requirements for you to atleast implement and test it.
    - [2*pts*] Test the model on larger than 300×300 resolution images [2].

[1] Password for login is given in section 2
[2] You can either train on larger resolutions, perform inference on larger resolutions, or both.

- [4*pts*] Implemented a pre-trained model from torchvision.models as the SSD backbone (or equivalent tensorflow model).
- [2*pts*] Train with random sample crop and random horizontal mirror data augmentation.

Please document that you have performed the steps above in the technical report or in the final presentation.

3. [16*pts*] **Model Performance**: This grade is determined by the following:
   - [3*pts*] mAP higher than baseline 1
   - [7*pts*] mAP higher than baseline 2
   - [6*pts*] Placement and final mAP score depending on how you are doing compared to your classmates.
4. [20*pts*] **Presentation, approach and documentation**: See section 5 for more information.
5. [2*pts*] **Bonus**: Annotate more data! You will get bonus points per additional video you annotate, following:

$$\text{Bonus points} = \frac{0.5 * \text{Number of additional videos annotated}}{\text{Number of students in group}}$$

It's not possible to get more than 2 points bonus.

2 Annotating Data

Annotating data precisely is an important part of any deep learning project. To get you started rapdily, we've built an annotation server on top of opencv/cvat, at tdt4265-annotering.idi.ntnu.no.

You can login on the server using

- username: group[blackboard group number] (for example: group1)
- password: o4QTyrEznri!JbqB

If you do not have a blackboard group, you can login with the username: "test". Before you start to annotate, make sure to:

1. Select a **project group** on blackboard. This is not the same as the assignment group from previous assignments.
2. Login to the annotation system, and validate that the group number in the right corner matches the project group number on blackboard.
3. Before you start to annotate a video, set yourself as the "asignee" of the task before starting. This will register that your group is the one that annotated this video.
4. Make sure to mark the video as finished before assigning yourself to a new video. It is possible that several of the persons in the group annotate several videos in parallel.

We will validate that you have annotated the data by looking into the server database.

Before you start to annotate the data, make sure to read the following tutorial:
`https://github.com/hukkelas/TDT4265-StarterCode/blob/master/assignment4/SSD/tutorials/annotation_tutorial.md`

3 Building the Model

In this section we will describe how to get started on building your model, including datasets, what code modifications we've done from Assignment 4, and some recommendations on what to do to improve your initial model.

Datasets

You will only use the *RDD2020 dataset* and the TDT4265 dataset. An issue with the TDT4265 dataset is that it's quite small (and this is a common case for real-world scenarios). A solution to this issue is transfer learning: pre-train the model on the RDD2020 dataset, then fine-tune the model on the TDT4265 dataset.

Note! In the start of the project, there will be very little (or none) annotated training data for fine tuning. Therefore, develop your model on the RDD2020 dataset initially.

To get started with this dataset, see
https://github.com/hukkelas/TDT4265-StarterCode/blob/master/assignment4/SSD/tutorials/dataset.md

Code modifications from Assignment 4

We've done several modifications from the assignment 4 code. You can view all changes here:
https://github.com/hukkelas/TDT4265-StarterCode/commit/0b432b5c263c0d90927e359a05745bbacbc8e238

In particular, we did the following

- Add support for TDT4265/RDD2020 dataset.
- Add a script to submit evaluation results.
- Include two config files for the RDD2020 and TDT4265 dataset.
- Add a script to perform runtime analysis of your model.

Model development

For this project you will use what you've learned from previous assignments to improve the model from assignment 4. There are no strict requirements or guidelines on the development, and you're free to use whatever means necessary to achieve a high mAP (as long as you follow the competition rules).

Here are a couple of recommendations to get you started:

- Change the backbone from assignment 4. Look back at the lectures on common backbones and see if you can find any of these in torchvision.models.
- Pre-train your model on the RDD2020 dataset.
- Look at various types of data augmentation to improve training (especially on the TDT4265 dataset). The SSD paper mentions several augmentations which they achieved good results with.
- Customize your model to use another image ratio. The original images has the resolution 1920×1080 and in the starter code we do a naive approach of directly resizing the image to 300×300. This causes artifacts in the image (such as a car would be much shorter in the resized image) and might hamper training and final mean average precision. Maybe you could change the model to process a non-square image?

4 Model Evaluation

To evaluate your model and compare your model to the rest of the class, we have created a leaderboard reporting the mean Average Precision for every group. For information about how to evaluate your model, see:
github.com/hukkelas/TDT4265-StarterCode/blob/master/assignment4/SSD/tutorials/evaluation_tdt4265.md

Furthermore, we've created a couple of benchmarks for you to beat, which are the following:

- **Baseline 0**: This is the solution to assignment 4 task 4a.
- **Baseline 1:** This is the solution to assignment 4 task 4c.
- **Baseline 2:** A solution following the model recommendations given above.

Note that the mAP for each baseline will be updated iteratively as more data is annotated. The last baseline update will happen three days prior to the project deadline.

Qualitative Evaluation

To evaluate your model on real scenarios, test the model on two video files. You can download these from Google Drive: https://s.ntnu.no/tdt4265-project-video-1 and https://s.ntnu.no/tdt4265-project-video-2

We've created a script to help you test your model on a video, see:
github.com/hukkelas/TDT4265-StarterCode/blob/master/assignment4/SSD/tutorials/evaluation_tdt4265.md

Show both videos in the presentation.

5 Documenting your Approach

Documenting and reporting your approach is an important part of any deep learning project. This will consist of a presentation and a short and concise report.

Presentation

Students working alone will have 9 minutes to present, groups of two will have 10, and groups of three or more will have 11 minutes to present. The presentation will most likely be online. More information on blackboard will be published after easter.

Topics you should cover in the presentation are:

1. **Development:** The approach you decided on and what steps you did to improve your model. It should clearly describe the reasons to the changes you did, and what kind of improvements you noticed from these changes. Remember from previous assignments, your reasoning should be supported by either theoretical arguments, previous experiences made from reading the curriculum, or empirical experiments. An example of this could be

    > ResNet is known to improve gradient flow and diminish the problem of the degradation problem, therefore, we decided to use ResNet as the backbone. By replacing the backbone we notice a significant improvement that is shown in the loss curve as you see here on our powerpoint slide.

    Of course, doing this for every improvement will take a lot of time, but we expect analysis like this for the major improvements you made.

2. **Final model:** Describe your final model. This should be short since we should have a clear picture of your final model from the development process. Things you might include:

- A brief overview of the final architecture.
- How you trained it, for example how did you pre-train it? What kind of data augmentation did you use? What kind of data pre-processing did you do?
- Amount of time required to train it.

3. **Results:** Show results of your model. Include quantitative and qualitative results. Quantitative results are for example loss curves, mean average precision, and average precision for specific classes. Qualitative analysis could be testing your model on different images in the dataset and see what kind of objects it's good at detecting, and what it struggles with.

4. **Runtime Analysis:** Perform a runtime analysis of your model (see info here). We've included a script to check how fast your model is over N images.
   - Did you make any effort to improve the runtime of your model?
   - If not, what could you potentially do to improve the runtime?

5. **Discussion** Discuss your approach for solving this task and the final results you achieved. Examples of questions you might want to answer is:
   - Did you test something that did not work?
   - Was there any unexpected results?
   - Looking back at the task, is there anything you would want to do differently?
   - If you did not have the limitation of the rules in this assignment, what would you do?
   - Is there any further work you would like to do? (for example, things you didn't get time for)

6. **Group member contribution:** In the end of the presentation, shortly describe every group members contribution to the project and what they were responsible for.

The final presentation will be a significant part of your grade. I recommend you to start early and prepare yourself well for the actual presentation. For general guidelines for presenting a project, Professor Charles Elkan has some good advice.

With these guidelines we are trying to help yo to show your knowledge about the curriculum in the course, and prevent you from waste time on nitty, gritty details. Often students struggle with managing the time during the presentation and spend all their time on describing the initial model, architecture etc. This leads to a very rushed discussion and result section, making it hard for the evaluators to understand the work gone into the project and the student's understanding of the underlying concepts. Also, we are not interested in what learning rate you used, what batch size you used or any kind of hyperparameters you chose; we can read up on this ourself if we are interested.

Documentation of Details

The report should be short and concise. What we expect you to include in this is anything "boring" and **note that we will not read this document except** if we are interested in technical details of your model, or we want to re-run/validate your experiments. Note that anything included in the presentation should not be included in the report. What we expect is that the report includes anything required to re-produce your results. Such as:

- Hyperparameters. This can be referred to as "We used the config file `our_amazing_model.yml` and all hyperparameters are there". Nothing else is required.
- How to train your model. Assume that we want to re-run all your experiments. Document clearly how we should be able to do this. An example of this could be

To setup your environment, install the additional packages "some-package" (Not required if you used the default environment used in the assignments). Then, you can train the model on cityscapes by running the file "some_train.py". Furthermore, fine-tune the model on TDT4265 dataset by running "some_train2.py. Finally, run the evaluation script.

- Specific details of your model architecture. Examples of this can be the tables with models given in previous assignments.
- Any additional results that you did not have place for in the report. However, we do not want any discussion of this result in the report.

The reason we want such a short report is that we do not have enough staff resources to read through everything. Even though we truly enjoy reading your assignments and reports, it would take us way too much time getting through all of your reports! **Delivery** We ask you to follow these guidelines:

- **Report:** Deliver your answers as a **single PDF file**. Include all tasks in the report, and mark it clearly with the task you are answering (Task 1.a, Task1.b, Task 2.c etc). There is no need to include your code in the report.
- **Plots in report:** For the plots in the report, ensure that they are large and easily readable. You might want to use the "ylim" function in the matplotlib package to "zoom" in on your plots. Label the different graphs such that it is easy for us to see which graphs correspond to the train, validation and test set.
- **Source code:** Upload your code as a zip file. In the assignment starter code, we have included a script (`create_submission_zip.py`) to create your delivery zip. **Please use this**, as this will structure the zipfile as we expect. (Run this from the same folder as all the python files).

  To use the script, simply run: `python3 create_submission_zip.py`
- **Upload to blackboard:** Upload the ZIP file with your source code and the report to blackboard before the delivery deadline.
- The delivered code is taken into account for the evaluation. Ensure your code is well documented and as readable as possible.

Any group who does not follow these guidelines or delivers late will be subtracted in points.

References

[Maeda et al., 2018] Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., and Omata, H. (2018). Road damage detection and classification using deep neural networks with smartphone images. *Computer-Aided Civil and Infrastructure Engineering*, 33(12):1127–1141.