TDT4265: Computer Vision and Deep Learning

Final Project

Håkon Hukkelås
hakon.hukkelas@ntnu.no
Department of Computer Science
Norwegian University of Science and Technology (NTNU)

March 31, 2020

- **Important dates:** [1]
  - Delivery of project: Monday, April 27th by 23:59.
  - Presentation dates: Tuesday and Wednesday April 28th/ 29th. (The sign-up sheet for the presentations will be available after easter.)
- **This project count towards 26% of your final grade.**
- You can work on your own or in groups of up to 4 people.
- Upload your code as a single ZIP file.
- Upload your report and presentation as a single PDF file to blackboard.
- The delivered code is taken into account with the evaluation. Ensure your code is well documented and as readable as possible.

[1]Deadlines might be extended due to COVID-19 situation.

Practical Information

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 1.1, Task1.2, Task 2.1a etc).
- **Plots in presentation:** 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.

Late Delivery & Other Penalties

We have a strict policy on late deliveries. Any delivery submitted after the deadline will be limited in maximum score which you can achieve on the assignment. See "Course Information" on blackboard for more information.

Also, if you fail to follow our delivery guidelines we will **subtract 0.2 points** from your final score. This includes if you do not deliver your report as a PDF file, if you don't deliver your code as .ZIP file, or if you don't use the `create_submission_zip.py` to generate your .zip file.

Grading

We will grade your project on the following:

1. [4*pts*] **Annotating Data** Your group is required to annotate a minimum of *four 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. [13*pts*] **Model Performance**: This grade is determined by the following:
   - [1*pts*] mAP higher than baseline 1
   - [3*pts*] mAP higher than baseline 2
   - [9*pts*] Placement and final mAP score depending on how you are doing compared to your classmates.
3. [9*pts*] **Presentation, approach and documentation:** See section 4 for more information.
4. [1*pts*] **Bonus**: Annotate more data! You will get bonus points per additional video you annotate, following:

   $$\text{Bonus points} = \frac{0.25 * \text{Number of additional videos annotated}}{\text{Number of students in group}}$$

   It's not possible to get more than 1 point bonus.

Introduction

In this project we will go through a typical computer vision workflow often seen in real world scenarios. The main task in the project will be object detection for autonomous vehicles, where all the data is collected from the streets in Trondheim.

To achieve this, we need to solve several tasks;

1. **Annotate data collected from Trondheim.** To achieve the best possible performance on a custom dataset, the typical approach is to pre-train a model on a similar dataset, then fine-tune this model on our dataset. Therefore, we have built a custom annotation server for this project, found at tdt4265-annotering.idi.ntnu.no. See section 1 for more information.
2. **Developing and building your model.** Your starting point will be the model from assignment 4 and your task is to improve your model for data related to autonomous vehicles. See section 2 for more information.
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 3 for more information.
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 4 for more information.

Competition Rules

To achieve a fair competition (and grading..), we ask you to follow these rules for all submssions to the leaderboard on tdt4265-annotering.idi.ntnu.no/leaderboard [2].

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. Therefore, 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. Note that you are not allowed to train for more than 12 hours for a specific dataset. For example, training for 14 hours on Waymo and 10 hours on TDT4265 is not allowed.

[2] Password for login is given in section 1

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 retrain your model following the steps in your report, and validate that we achieve a similar mAP. Breaking rules will be considered cheating and we will subtract a large amount of your grade on the project.

1 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: jK4C8G7LKbBJKdP5yoH5RShjoC

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 finish the annotation of the video 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`

2 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

We recommend to use two datasets in this task: the *Waymo dataset* and our custom dataset (we will refer to this as 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). However, from the curriculum we've learned about transfer learning; pre-training our model on a the Waymo dataset, then fine-tuning the model on the TDT4265 dataset.

In the start of the project, there will be very little (or none) annotated training data. Therefore, develop your model on the Waymo dataset initially.

To get started with these datasets, 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/74730448694bdbedb75096c1708afd4d2a072027

In particular, we did the following

- Add support for TDT4265/Waymo dataset.
- Add a script to submit evaluation results.
- Add a script for download of TDT4265 datasets.
- Add a script to visualize dataset and labels.
- Include two config files for the waymo and tdt4265 dataset.

Also, we've added a couple of tutorials to get you starter. See:
https://github.com/hukkelas/TDT4265-StarterCode/blob/master/assignment4/README.md

Model development

In previous assignments we've told you quite explicitly what you should do. For this project, however, we will give you the freedom to use whatever means necessary to achieve a high mean average precision for your object detection model (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 at common backbones and see if you can find any of these in torchvision.models.
- Pre-train your model on the waymo 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, and several of these transforms are included in the starter code. You can find this in the folder `ssd/data/transforms`.
- Customize your model to use another image ratio. The original images has the resolution 1280×960 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? (For example 320×240).

3 Evaluating the Model

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 assignmetn 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: s.ntnu.no/tdt4265-video1 and s.ntnu.no/tdt4265-video2.

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 one of the videos in the presentation. We might ask you to show the other video as well, therefore, have both ready for the presentation.

4 Reporting 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 should be:

1. **Initial model:** Shortly describe the model you started with from Assignment 4.
2. **Development:** The approach you decided on and what steps you did to improve the 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, which you can see from the loss curve which we are currently pointing to on the our powerpoint slide.

Of course, doing this for every improvement will take a lot of time, but *please* do it for the major improvements you made.

3. **Final model:** Describe your final model. This might be short if a lot of it is covered in the "Development" section. Interesting things to include is:
   - 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.
4. **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.
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 I truly enjoy reading some of your assignments and reports, it would take me way too much time getting through all of your reports!

References

[LeCun et al., 2012] LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.