Final Project

Computer Vision and Deep Learning

Håkon Hukkelås
hakon.hukkelas@ntnu.no

Delivery deadlines:
Annotation (Task 1.2): Friday, April 8th, 23:59PM.
Poject delivery: Friday, May 6th, 23:59PM.
This project counts **50%** of your final grade.
You are allowed to work in groups of up to **3 persons**.

About. In this project, our goal is to detect objects in traffic from LIDAR images collected around the campus (we will call this dataset the **TDT4265 dataset**). The figure below shortly describes the dataset. The project is separated into 4 separate parts, following a typical computer vision development cycle: dataset exploration and annotation (**part 1**), model development (**part 2**), model evaluation (**part 3**), and "going beyond" (**part 4**). It is not required to solve these parts sequentially, but we recommend you to start with part 1. Note that all groups have to finish dataset annotation **by the 8th of April**.

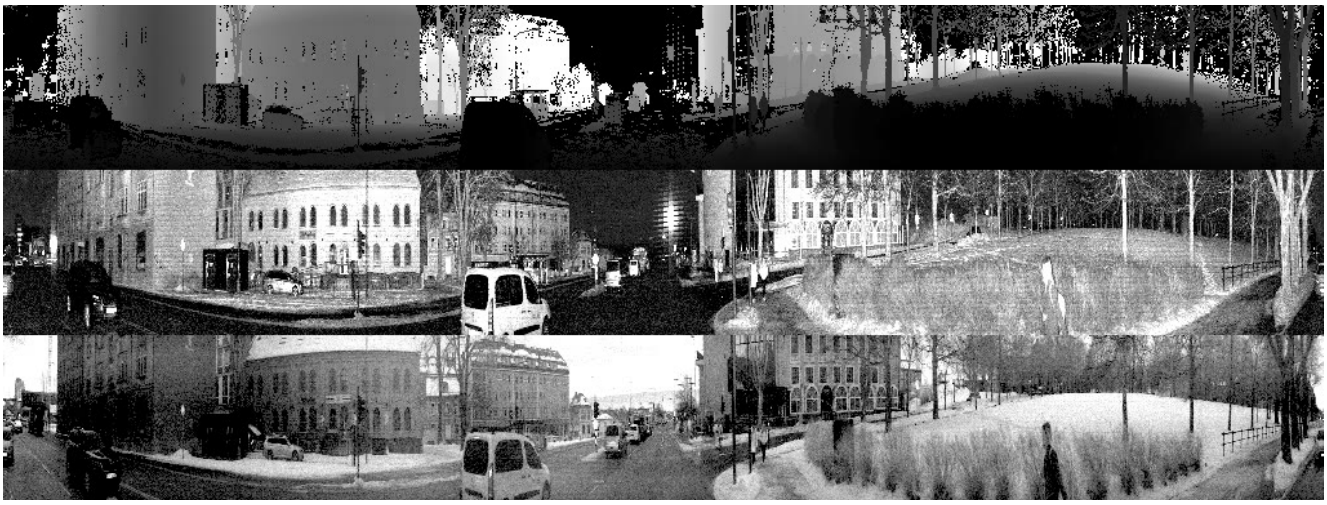

Figure 1: The figure shows a sample image from the TDT4265 dataset. The image is a 3-channel image from a 360° 128-channel LIDAR. The image is separated into depth of the scene (top row), ambience, and intensity (the return of the laser beam, bottom row). The images have a resolution of 128 × 1024 and are stored as 8-bit 3-channel images. Note that we have concatenated the 3-channels into an "RGB" image in the starter code.

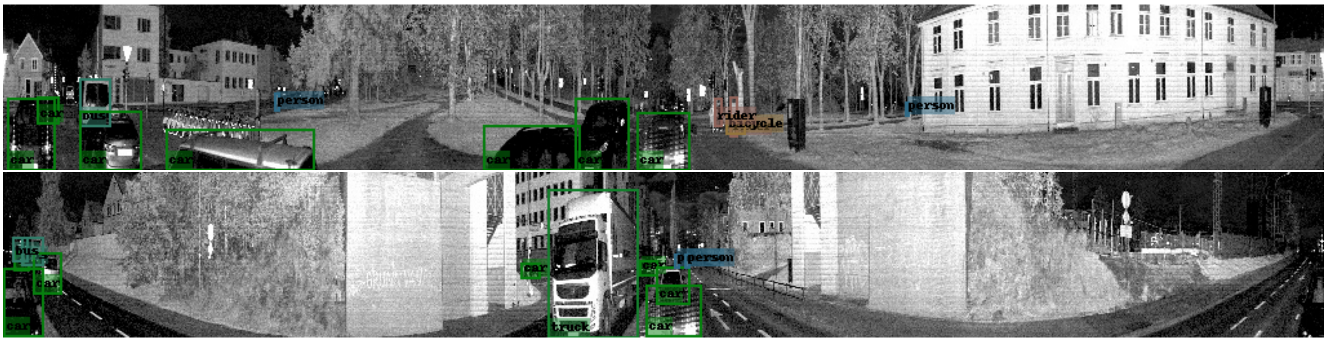

Figure 2: Parts of the TDT4265 dataset is already annotated, where every moving object in the traffic is annotated into the following categories: car, truck, bus, motorcycle, bicycle, scooter, person, and rider. All other object categories are ignored. **Note.** This visualization shows only the intensity of the LIDAR image.

Easter break. This project timeline intersects the Easter break (April 8th-April 18th). The student assistants will not actively answer questions during this break, unless there are issues with provided resources (dataset/server).

Delivery Guidelines

You are required to deliver a report, your code, and a video for this project. Each task is marked with clear delivery instructions.

Report.

- Only tasks marked with Report should be included.
- Divide the report into the respective subtasks and mark each subtask with a header.
- If a task require a quantitative analysis, include the analysis under the respective subtask.
- If a task require a "technical description" you can shortly describe what you changed in the report, and refer to a specific spot in the code to see details. Preferably, each model iteration for each subtask should have a separate config file, such that you can be brief in the report and refer to the code for details. For example, a "perfect" answer for task 2.2 would be: "we applied random horizontal flip and mirror for the model in task 2.1. See the config "configs/task2_2.py for details". An answer such as this allows us to easily inspect what you actually implemented, without requiring you to duplicate and spend time on documentation in the report that is already in the code.

Video. The video delivery should be a **single** video that presents the entirety of the project. Feel free to format the video as you'd like, but we recommend you to back up your analysis with figures/visualizations that are presented in the video. For example, a screen (or physical) recording of a powerpoint presentation is perfectly fine. We've detailed which tasks that we expect you to present in the video and what you should discuss for each subtask.

- Only the tasks marked with Video should be included.
- The video should be a maximum of 6 minutes for a single person, and 8 minutes for a group of 2 or more.
- Follow the video guidelines for each subtask.
- You are free to switch the order of subtask presentation if you find that it improves the flow of the video.

Code.

- The delivered code will be taken into account for grading. Make sure that your code is readable and in a functional state.
- You can use create_submission_zip.py given in the assignment 4 starter code.

Starter Code

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, except for part 4.

Important changes to the starter code The starter code is based on the SSD repository from assignment 4. It is slightly changed for this project, and you can view all changes in the commit history (all changes since March 14th). The important changes are:

- Updated the dataset tutorial with information about downloading the TDT4265 dataset.
- Bugfix of RandomSampleCrop.
- Improve evaluation to print per-class AP and change area thresholds for box categorization into small/medium/large.
- Several quality of life changes.
- Update readme to include descriptions of new scripts that can be useful for the tasks in the project.

We recommend everyone to include all of these changes into your new code. For those not familiar with git merge, we recommend you to delete the old code and copy the basic.py model from assignment 4.

Configuration files. We recommend you to create separate configuration files for each experiment in your project. The configuration files used for SSD are designed for iterative development, such that you can change small parts of the code without having to duplicate configs. For example, if you want to change the learning rate to 1, you can simply change a few lines of code (see the example here). Or, if you want to run experiments for different anchor boxes, you can take a look at this config file.

Recommended resources:

- A Recipe for Training Neural Networks by Andrej Karpathy.

Part 1. Dataset Exploration (9 points)

Downloading the dataset: See here.

Video Task 1.1 (6 points) - Getting to know your dataset

Perform an exhaustive dataset analysis to explore the attributes of the TDT4265 dataset. The analysis should highlight commonalities, limitations and perhaps interesting samples in the dataset (both images and labels). You are free to choose how to present the analysis, and the analysis can be both statistical and qualitative. To get you started, you can try to analyze the size of the objects in the dataset.

This step is (arguably) the most important part of any computer vision project. The insights you bring from the dataset analysis will guide your moodeling decisions and help you build an efficient model tailored to the needs of the task.

Delivery guideline. We expect you to present several plots/figures in the video. The visualizations that you find most insightful should be included in the video, and optionally, include any additional analysis in the report.

Note. We have included a couple of scripts to get you started on the dataset visualization. See the readme.

Task 1.2 (3 points) - Annotating Data

Delivery date: April 8th

Read the tutorial about annotation found here.

Your group is required to annotate one video per person in the group.

The username and password is:

- Username: group[blackboard group number] (*e.g.*: group1)
- password: SR5cF8RFqQF9A3aF

NOTE THAT YOU HAVE TO JOIN A PROJECT GROUP BEFORE STARTING TO ANNOTATE. THE PROJECT GROUP IS NOT THE SAME AS THE GROUP NUMBER FOR THE ASSIGNMENTS.

Delivery guideline. Mark your delivery as ”validation” when you are done with annotating the video, and we will review the annotation shortly after. You will receive full score if your video is marked as ”completed” in CVAT. There is no need to deliver anything in the report or video.

Part 2. Model Creation (23 points)

Report Task 2.1. (2 points) - Creating your first baseline.

Adjust your model from assignment 4 (task 4a) to support object detection on the TDT4265 dataset. You can keep the backbone convolutional network more or less the same, but you need to change the anchor boxes to support the new dataset resolution.

Note that the objective of this task is to create a ”dumb” baseline with no advanced modeling choices. This will give you a lower performance bound to adjust the expectations for a more complicated model.

Tip: The model from assignment 4 expects the resolution 300 × 300 and you need to alter your model to handle the image resolution of the dataset in this project (128 × 1024). We altered the priors to the following:

- Feature sizes: [[32, 256], [16, 128], [8, 64], [4, 32], [2, 16], [1, 8]]
- Strides: [[4, 4], [8, 8], [16, 16], [32, 32], [64, 64], [128, 128]]
- Min sizes: [[16, 16], [32, 32], [48, 48], [64, 64], [86, 86], [128, 128], [128, 400]]
- Aspect ratios: [[2, 3], [2, 3], [2, 3], [2, 3], [2], [2]]

In addition, we removed the second max-pool layer and set the kernel size of the last to convolutions to 2. Following this tip is not required, but if you do, you should expect an mAP@0.5:0.95 of 0.037.

Report guidelines. Include a technical description of the model, and include a quantitative analysis following the description in Task 3.1 [1].

Report Task 2.2 (3 points) - Augmenting the Data

Increase the training dataset by introducing data augmentation into your training pipeline. We recommend you to start with random horizontal flipping and random cropping.

Report guidelines. Report the data augmentations used in your report and perform a quantitative analysis. Compare the improvement of data augmentation compared to Task 2.1.

Tip: Costly data augmentation can often become the bottleneck of your training pipeline. We've included a script to benchmark the runtime of your dataset loading pipeline (benchmark_data_loading.py) to make sure that your augmentations are computationally efficient. Also, the inspect_dataset notebook visualizes images with augmentations, which can be useful to make sure that the augmented images looks ”realistic”. See the readme for more info.

Video Report Task 2.3. (11 points) - Implementing RetinaNet.

Read the paper ”Focal loss for dense object detection” [1]. The paper proposes a one-stage detector with several advances compared to the original SSD model. Their main contributions can be summarized into four different components. Compared to the SSD framework, they change the following modules:

1. Replacing the backbone with a feature pyramid network.
2. Replace hard-negative mining in the SSD loss with Focal loss.
3. Replace the single-layer regression and classification output heads with deeper convolutional nets.

[1] This task and the following tasks will ask you to include a quanitative/qualitative analysis in your report/video. You are required to include this analysis, following the description in Task 3.1/3.2.

4. Initialize the bias of the last convolutional layer of the classification head to account for background class-imbalance.

Iteratively develop your model (replacing one component at a time), and perform a quantitative analysis of the improvement of each component.

Video guidelines. Present the quantitative analysis of the iterative development. Do not spend time on explaining what each component does (unless needed to support an argument in your analysis). We will evaluate your understanding depending on your implementation and report.

Report guidelines. Refer to where in the code each component is implemented. Also, please report the config file for each iteration in the development if you have different config files for each of them.

1. Note on FPN: We recommend to implement FPN on top of a pre-trained backbone (*e.g.* one of the ResNet variants from torchvision model zoo). In addition, you can use the torchvision FPN helper function to ease the implementation.
2. Note on focal loss: The paper defines focal loss as the following (see eq. 2 in the paper for definition of p_t):

$$FL(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t). \tag{1}$$

This is a modification of the binary cross-entropy loss, where the authors use a sigmoid activation for every class (instead of a softmax). For those interested, we recommend this stackoverflow discussion on the reasoning behind. In short, this simplifies the implementation such that it is possible to remove the "background" class. The provided SSD implementation use Softmax classification (class "0" is treated as background), and altering the implementation to sigmoid classification requires several changes in the code. Instead, to simplify your task, you can still use softmax cross entropy loss and use the following focal-loss definition.

$$FL(p, y) = -\sum_{k=1}^{K} \alpha_k \cdot (1-p_k)^\gamma \cdot y_k \cdot log(p_k), \tag{2}$$

where α_k is the weight for class k, p_k is the softmax output for class k, and y is the ground truth one-hot encoded (1 if the ground truth correspond to class k else 0). This is simply a generalization of the equation above.
Note that we implemented the α-weighted version of Focal loss, where we set it to 1 for all classes except background, which is set to 0.01 (we're not saying that this is the best weighting).
3. Note on deeper regression heads. The authors share the parameters of the regression and classification heads across different feature resolutions. In other words, they use the same regression/classification head for all feature resolutions. To achieve this, you need to use the same number of anchor-box ratios at every feature map.
4. Note on weight initialization. Follow the weight initialization stated in the paper for the convolutional filter, and set the bias to 0 for all classes except the background. For the background class, you can set the bias to $b = \log(p \cdot \frac{K-1}{1-p})$ where p is the probability of an anchor belonging to a background class (we used $p = 0.99$).
Final note. You might struggle with "NaN" gradients when developing this model, which you can observe if the AMP gradient scale (logged in tensorboard) goes below 1. This is, most likely, caused by incorrect weight initiaization, where step 4 will significantly diminish this issue. Thus, if you struggle with this early, we recommend you to implement step 4 earlier (note that Step 4 can be implemented with the original SSD model).

Video Report Task 2.4 (5 points) - Using the Exploration Knowledge

Use the knowledge you learned from Task 1.1 and specialize your model to the dataset. You will receive points to any changes that you introduce in the code to specialize your model to this unique dataset [2]. To get you started, we recommend you to specialize the anchor boxes.

Video guidelines. Report the design decisions you made in the video and evaluate the new specialized model. We expect you to compare the improvement of your new model to a model without the specializations.

Report guidelines. Include a technical description of the modifications that you introduce to your implementation in this task. A comprehensive technical description is not required in the video (only the key/interesting aspects).

Note on evaluation: You're not required to complete task 2.3 before starting on this task. You can perform your experiments on whichever model you'd like (from task 2.1, 2.2, 2.3, or 2.5). But, make sure that you note in your report which model you compare against!

Report Task 2.5 (2 points) - Extending the Dataset Dataset released: April 20th

Extend your dataset with the data annotated by you and your classmates. Retrain your model from scratch with the new dataset and perform a quantitative evaluation to analyze the importance of a larger dataset.

Report guidelines. Include the quantitative analysis in your report.

Information regarding how to download the new dataset will be announced on blackboard later.

Part 3. Discussion and Evaluation (12 points)

Task 3.1 (5 points) - Quantitative Analysis

Note on grading. The grading of this task will be determined on how thorough and fitting your evaluation is for the tasks in part 2 and 4. Thus, your "delivery" for this task will be the delivered evaluation for Part 2/4. Note that you can achieve full score on this part even though you have not delivered all subtasks.

Perform a quantitative evaluation of your model. We expect you to report the following metrics, as a minimum:

- COCO mAP@0.5:0.95.
- Inference speed. You can evaluate this with the script runtime_analysis.py.
- Number of parameters.
- Plots of loss(es).

Feel free to extend the set of metrics with other relevant metrics/plots. Also, you're not required to report every metric that you use for every subtask, as some might not be relevant (*e.g.* inference speed will not change if you only change data augmentation used).

[2] Points will be given to any implementation detail that is grounded in the dataset analysis. Methods that are not unique to the dataset, such as "specializing" the learning rate, will not be awarded points.

Video Report Task 3.2 (5 points) - Discussion and Qualitative Analysis

Select three [3] of the most influential modeling decisions throughout the project and present a qualitative analysis. The qualitative analysis should highlight key aspects of your model and should be supported by qualitative examples.

Example questions that your analysis and discussion can answer are:

- What are the strengths of the model?
- What are the limitations of the model?
- What is the reason that a specific modeling decision improves metric X?
- Alternative methods to the modeling decision and why they are better/worse.

We've included a set of scripts to help you to visualize predictions for your qualitative analysis. See the project readme for more info.

Video Report Task 3.3 (2 points) - Final Discussion

What are the weaknesses of the approach selected in the project if you were to deliver this to a customer?

Task 3.4 (0 points) - Leaderboard Submission

To 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 the project readme.md.

Submitting to the leaderboard is not required.

Part 4. Going beyond (6 points)

The following tasks are difficult, not as restricted as previous ones, and we recommend everyone to finish part 1 and 2 before starting on this part. You can select which tasks you want to do freely, and we expect you to solve one task per group member. Thus, a group of 1 is expected to solve one task, a group of 2 is expected to solve two tasks, and three for a group of 3.

Video Report Task 4.1 - Implementing BiFPN

EfficientDet [2] propose a bi-directional feature pyramid network (BiFPN) to replace FPN (section 3.2). Implement BiFPN for your model and perform a quantitative analysis to compare it to regular FPN. Do you observe a similar improvement as the original authors?

Video guidelines. Present your comparison in the video.

Report guidelines. Include a technical description of the model you used.

Video Report Task 4.2 - Explaining the Model with CAM

Class activation maps (CAM) tries to visualize what a model attends to when detecting objects. Implement CAM for your object detection model and present the results.

Video guidelines. Present the CAM visualization in the video.

[3] Selecting more than three is OK, but the grading is dependent on the quality of your analysis and not the number of models compared.

Report guidelines. Include the implementation of CAM in your report/code. Describe how to reproduce your results.

Tip. To get started, we recommend you to take a look at this Github repsoitory

Video Report Task 4.3 - Following your own idea

This task is designed to give you the freedom to explore your own ideas. You will be awarded points for anything "extra" that you do outside the scope of the previous tasks, and we will award points to everything that is within the scope of this course. Remember that quality is often better than quantity, so we're not asking you to do endless of boring/thoughtless tasks to achieve these points.

Video guidelines. Present the key aspects/insights to your idea, and optionally, present a qualitative/quantitative analysis.

Report guidelines. Document the technical description of what you did, and include any additional analysis in the report that was not presented in the video.

Feel free to ask us on Piazza or in the lab hours if you are unsure if your idea is within the scope of this course.

Video Report Task 4.4 - Comparing to State-of-the-Art

Current state-of-the-art uses much more advanced tricks/techniques to improve object detection. Take a look at open-source object detection frameworks and try to transfer learn an object detection model to the TDT4265 dataset.

Video guidelines. Present the key aspects/insights to the model you choice and reasoning behind the choice. Present the qualitative/quantitative analysis in your video.

Report guidelines. Include a technical description of the model you used and source code to reproduce your result [4].

Tip. We can recommend the Detectron2 object detection repository. It is one of the most popular repositories out there and has pre-trained models for a wide range of object detection models.

References

[1] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 4

[2] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 7

[4] If you forked a Github repository, you can just include a link to the fork.