



AM-Traffic I. Phase 2. Iteration 2

Project Report

About:

This project report shows the issues and problems that have been encountered during the “**AM-Traffic I. Phase 2. Iteration 2**”. Also, it contains some ideas, thoughts and future improvements that can be done.

Project Report:

This project is a combination of building a dataset pipeline and training it for two classification tasks (1st task regarding the Weather Conditions and the 2nd task regarding the Road Conditions). The data of this project will be extracted from Finland digitraffic [Link: <https://www.digitraffic.fi/en/road-traffic>].

The tasks were proceeded in the following way:

Task#4 to gather useful data and build a dataset so that it can be used for classification purpose in Task#1 and Task#2.

Task 4:

- Build a Crawler to gather data from digitraffic,
- Build Finland Map that:
 - pinpoints the locations of (Camera stations and Weather stations)
 - contains information related to each station,
 - tracks nearest (radius 200 meters) weather stations to each camera station and vice versa.
- And finally deploy the crawler on AWS.

The www.digitraffic.fi API has restrictions and an Anti-DDOS mechanism, which may cause a permanent ban to any abuse of requests, so the idea is to extract the useful information from the available JSON files that change every (1min ~ 5min, more details can be found on the website) kind of making **Recon-ai** own API.

In order to make the dataset much easier to use, extracting the information from the sensors and rename images according to their labels is needed.

Since **Recon-ai** will gather data several times a day: **Lambda function** was chosen over **EC2**, because it will be charged only for the execution time (around 19~20min). This lambda function is triggered 4 times a day by a CloudWatch.

The Lambda functions purposes in Task#4 are to download images from digitraffic.fi, to get their corresponding information and to put the extracted data in tables on the (DynamoDB).

Issues encountered:

1. Figuring out every possible information that the sensors deliver was time consuming since no documentation about sensors and their possible outputs was available.
2. Founding cases where the vote mechanism must be used (e.g.: road left condition (wet...) and road right condition (ice or dry...)).
3. On AWS, lambda function has a timeout limitation (15min). That's why the need of dividing the process and using two lambda functions instead of one.

The first one gather JSON files, parse the downloaded files, download images, create csv files that contain data to be dumped in the corresponding tables.

The second one, has one purpose which is to dump the sensors data into the corresponding table, which is time consuming due to the huge amount of data.

Task 1 & Task 2:

- Train a classification model that can distinguish the weather conditions for Task#1 and road conditions for Task#2.
The model should be light weighted enough to be deployed on a Jetson Nano.

The Resnet-50 and Resnet-34 has been chosen for transfer-learning, to deal with the compromise of Accuracy and Light weighted model.

Extra data from CCTV traffic camera from around the world was gathered so it can be used in the training phase.

Issues encountered:

1. The data provided by our source www.digitraffic.fi contains many mislabeled images: it's due to sensors errors, distance between sensors and camera station, external intervention like wind or vehicles ... Besides, the data contains images that are unrelated to the task (e.g.: natural view, parcs and sometime shut-downed camera).
Dealing with this issue was so much time consuming, since manual verification is a must for almost every image while building the training & test dataset.
2. Since our data source CCTV take pictures every few minutes and it's not a live streaming, in most cases the condition cannot be recognized even by a human especially regarding weather condition (e.g. for the weak rain (same goes for weak snow) condition, the image can be described as clear because it hardly shows any fallen rain or snow) and that effects so much the training phase.
3. Some labels do not even exist in our dataset, so external traffic CCTV images are needed in order to feed them to the dataset.

Common issues, ideas and suggestion (Task#1 & Task#2):

Both Taks#1 and Task#2 were done using **TensorFlow** and **Pytorch** frameworks, models will be compared to choose the best one.

- **Pytorch** model provided slightly better accuracy, but while making conversion from **Pytorch** (.pt file) to **ONNX** (.onnx file) in order to create a **TensorRT** engine, **ONNX** required a **static** input and does not support yet **dynamic** inputs, so the model get modified and lose accuracy. (For more details check: <https://github.com/onnx/onnx/issues/654>).

- The version of **TensorRT** installed on the Jetson Nano is v6.0 and for conversion from **ONNX** to **TensorRT** engine you should have v7.0 (For more details check: <https://github.com/onnx/onnx-tensorrt>). That may be helpful for future use and making the process much easier.
- Since **Recon-ai** is gathering data several times a day, the dataset is getting bigger and more diversified, so using manpower for labeling will make the future trained models for both Task#1 and Task#2 much more accurate.
- Possibility of unifying both tasks in one, by creating a multi-output model (Weather and Road Conditions).