A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection - Reproduction

April 15, 2021

This blog post describes the replication of the experimental results of the paper "A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection" by Nobis et al.

1 Problems tackled by the paper

Nobis et al. combine data from both radar and camera sensors to achieve better performance for object detection using deep learning. Object detection using camera data in combination with computer vision or deep learning works very well in good weather conditions. However, when conditions are less ideal, the camera data becomes obfuscated and object detection will perform much worse. Worsening conditions can for example be severe weather conditions, or sparsely lit areas at night. Combining this camera data with radar data solves this issue, and improves camera object detection in such conditions. The authors show that the fusion network is able to outperform a state-of-the-art image-only network for two different datasets.

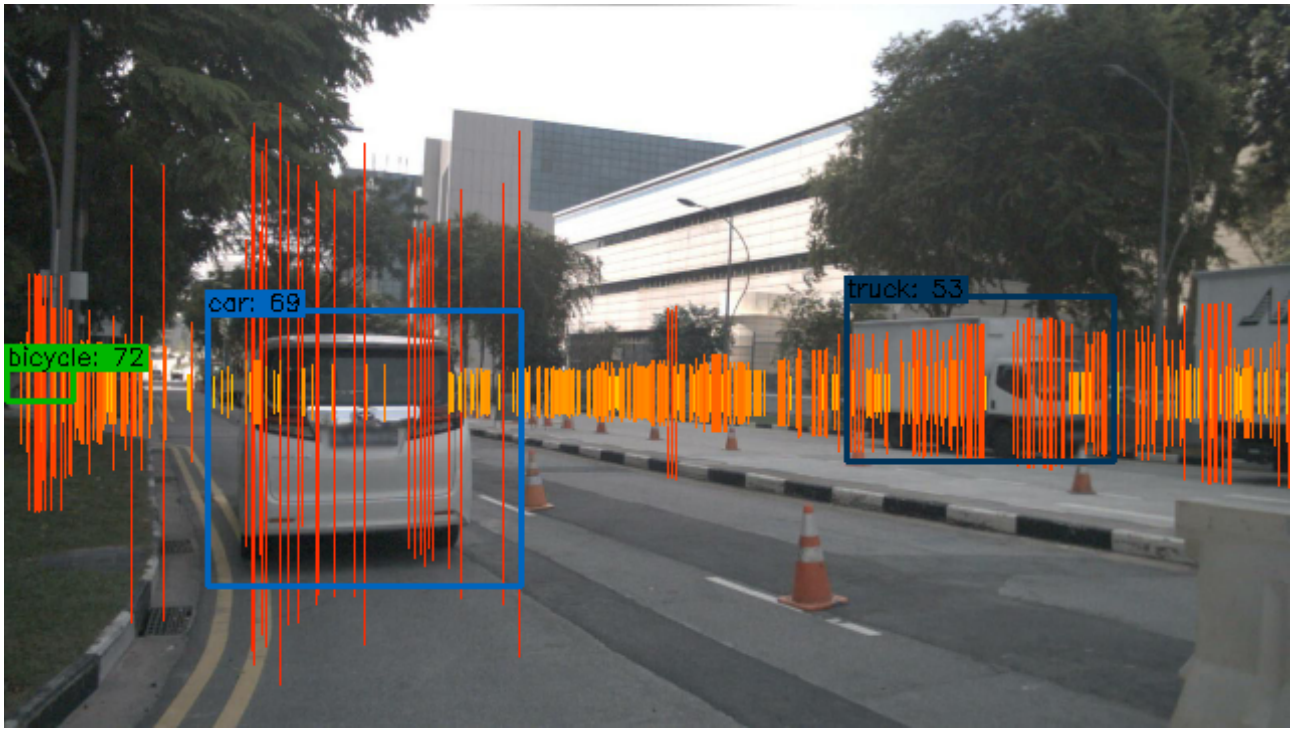

Figure 1: Visualization fusion of camera and radar data

2 Camera Radar Fusion

Their approach enhances current 2D object detection networks by fusing camera data and projected sparse radar data in the network layers. The proposed CameraRadarFusionNet (CRF-Net), as seen in 4, automatically learns at which level the fusion of the sensor data is most beneficial for the detection result. Additionally, the authors introduce BlackIn, a training strategy inspired by Dropout, which focuses the learning on a specific sensor type.

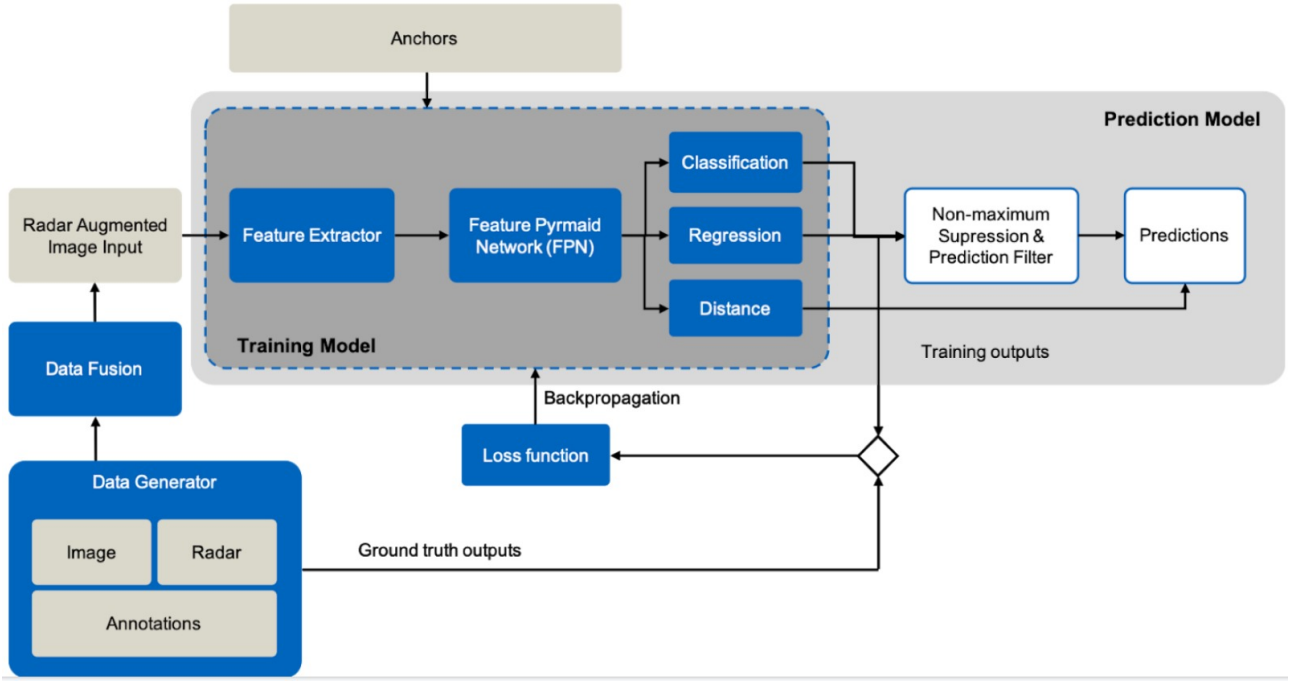

Figure 2: Overview of the network architecture

3 Original Goal

Originally, to determine the reproducability of this paper, we set out to follow the paper in their description of the network. The paper is deemed reproducible if we are able to reproduce their results using only the information in their paper. Only then we can determine if the paper is reproducible. One would wonder why we can not use the code provided by the paper if the authors made this available. As researchers need to verify the claims of the paper that are supported by their results, the results need to be verified as well. If the concepts in the paper lead to the results that the authors claim, one should be able to reproduce these results using the concepts and descriptions in the paper.

The original network is implemented in TensorFlow and we were going to re-implement the concepts of this network in Pytorch. If we were able to get the same results, the concepts are proven to contribute to the result and consequently the claims made by the authors.

4 Obstacles encountered and change of approach

Unfortunately, it quickly became clear that it would be a too difficult task to port all code to Pytorch. Because it turned out that just get the current tensorflow version running was already very hard and reproducing the original experiment would take up the entire timewindow of the project. All groups encountered numerous errors ranging from old python framework and old tensorflow version which are no longer supported to Ubuntu, as well as Docker related issues. This caused the supervisor of the reproduction project to change the original goal. The new goal was to get the original code running and reproduce the same result.

This was discussed in our weekly meeting with our Teaching Assistant and he agreed.

With the help of other groups doing the same paper and much trial and error, we were able to get the network trained using GPU. This was a huge milestone as we were now able to do training much quicker. Unfortunately, the nuScenes dataset was too large to download and too large for our budget on Google Cloud. This let us to the decision to limit the scope of the reproduction to the nuScenes mini dataset, which is a subset of the complete nuScenes dataset and contains 10 scenes. Usage of the mini dataset was approved by our Teaching Assistant.

5 Code environment

We ran the code using Windows Subsystem for Linux 2 (WSL2). This allows us to still use windows and run the code in a VM like environment, while also allowing tensorflow to utilize the GPU. In order for this GPU pass-through to work, the correct NVIDIA CUDA drivers and toolkits need to be installed on both the windows host as well as inside the WSL2 environment. Furthermore, the matching version of NVIDIA cuDNN needs to be installed inside WSL2. Configuring TensorFlow to use GPU rather than the CPU cost us a large amount of time, as we were not yet familiar with WSL2, not very familiar with linux in general nor with NVIDIA CUDA/cuDNN. An overview of the environment can be seen in image 3

After many hours of trial, error and using google, the environment was finally configured. It was now time to install Python 3.5, and all python dependencies. The authors of the paper did specify which dependencies were needed in a requirements.txt file, however, not all versions of dependencies were specified. This caused problems, because PIP tries to install the newest versions of dependencies if the version is not specified, but those new versions are not compatible with the now old python 3.5 version. These issues cost us much time to identify, but in the end we got the code up and running, using the GPU.

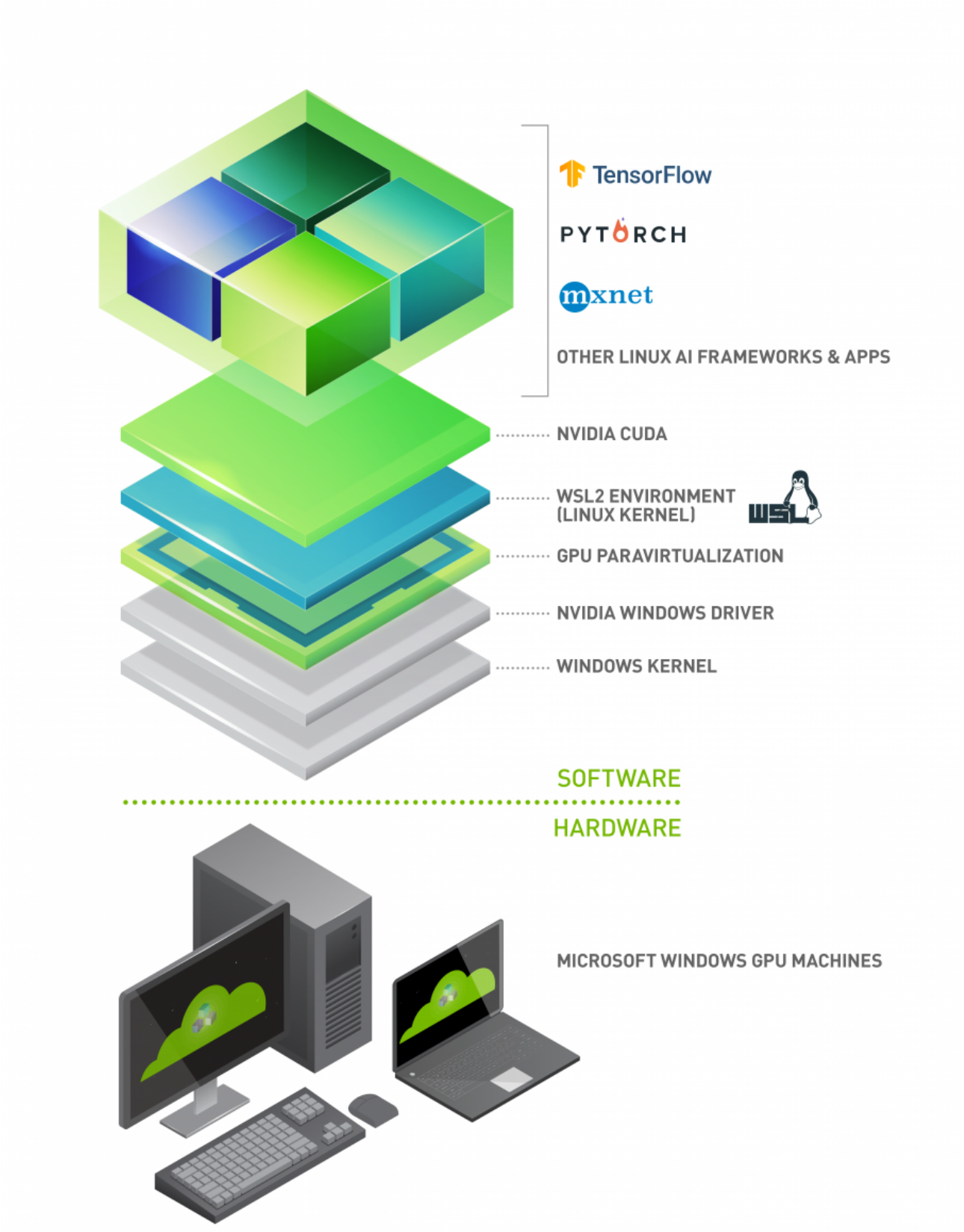

Figure 3: Overview of the WSL2 NVIDIA GPU environment

6 Reproduced Results

Using the mini nuScenes dataset, which consists of 10 scenes of the complete nuScenes dataset, we were able to achieve 56.94% accuracy. The reported accuracy in the original paper was 55.99%. This is not a full reproduction, because we used a much smaller dataset. However, from this small scale experiment we can at least conclude that the reported accuracy's are not unthinkable, as the accuracy is in the same order of magnitude. In our reproduction, a 20%,20%,60% (test, validation, training) data split is used. This is the same split as the authors used in their experiments.

Data	Network	mAP
nuScenes	Baseline image network	43.47 %
	CRF-Net w/o BlackIn	43.6 %
	CRF-Net	**43.95** %
	Baseline image network (AF)	43.03 %
	CRF-Net (AF)	44.85 %
	CRF-Net (AF, GRF)	**55.99 %**
	CRF-Net (AF, GRF, NRM)	53.23 %
TUM	Baseline image network	56.12 %
	CRF-Net	**57.50** %

Data	Network	mAP
nuScenes mini (10 scenes)	CRF-NET (AF, GRF)	56.94 %

Original results

Reproduced results

Figure 4: Comparison of the achieved results and the result of our reproduction attempt

7 Conclusion

While not fully reproduced, we did manage to get the code of the paper to run with GPU support on a subset of the data used originally. Using this smaller dataset, we achieved very similar accuracy's. We acknowledge that this fact alone is not enough to conclude the validity of the results of the authors, due to the large discrepancy in the datasets used. However, our experiment can be seen as preliminary research, and our findings as an indicator of the feasibility of the achieved results. In this case, the accuracy's we achieved suggest that the claimed accuracy's by the author are plausible.

8 Potential Improvements

To fully reproduce the results of the original paper one would need to use the same dataset, not just a subset of the data. The way for this is already paved, since we spend much time to get TensorFlow to use the GPU instead of the CPU, which makes training much quicker. This was not possible in the limited amount of time we were given for this project, but it is certainly possible to do.