Explainable Object-induced Action Decision for Autonomous Vehicles for ML Reproducibility Challenge 2020

Roy Yang
zh2yang@uwaterloo.ca

Qihang Wang
q338wang@uwaterloo.ca

Xiangyu Zhao
x324zhao@uwaterloo.ca

Jonathan Rasmussen
jrasmuss@uwaterloo.ca

Reproducibility Summary

Scope of Reproducibility

The paper explores the application of computer vision in the realm of autonomous driving. It is claimed that action inducing objects such as traffic lights and pedestrians as local features have been proven effective in action decisions. This project aims to verify the claim that the proposed architecture, which extracts both global and local features is able to achieve substantial results and outperforms the baseline model, which only takes global features into account.

Methodology

The repository provided by the authors was directly used during the reproduction study. The model architecture remains the same and very minor changes are made in the training file (pathing etc.) to accommodate our environment on Google Colab. The batch size was changed from two to four and the number of epochs is reduced from 50 to 25 due to time constraints. The total recorded training time for this experiment is 40 hours.

Results

Substantial results were produced that were surprisingly close to the claims in the paper even though the training was significantly shortened. Several categories outperformed the proposed results in the paper while others fall short. Training logs indicate that the model is still able to improve if further training is performed. Overall we believe the author's claim that this experiment covers is indeed credible.

What was easy

There was very little concerning this reproduction study that was easy. The easiest aspect of this replication was the easy access to the novel-labelled dataset the authors provided. The dataset and its accompanying JSON file was well formatted and offered good variability between the various actions and explanations.

What was difficult

The reproduction of this study had several difficult components. The backbone detector used was depreciated and therefore the author's code was broken and unusable when downloaded directly to a local machine. An alternative was to use Google Colab to run the model rather then using the local machine. This resulted in a functional model that could be trained. However, another difficulty was the long periods of time and the high amount of computational resources necessary for training the model. Using a Tesla T4 provided by Google Colab, it would take approximately 100 hours to train the model on their action-inducing dataset.

Communication with original authors

No form of communication was conducted with the original authors during this reproduction study.

1 Introduction

The concept of autonomous vehicles has grown in popularity in the last few years.(1) An autonomous vehicle recognizes a scene image, detects objects of interest, plans an actions based on objects of interest in the scene, and executes the action. Although the technology exists and is studied by many researchers, public concerns have been raised concerning the efficacy of such autonomous systems. Further studies to develop more accurate and effective models are before this technology can be widely adopted by the public. There are two approaches in the autonomous driving field that has gained recent popularity by academics.(2) One method is an end-to-end system utilizing the global features found in the entire scene to make predictions on an appropriate action. However, a large dataset is required to train an accurate model and the black box nature of this method does not offer explanations for why an action is proposed. The other method is called the pipeline approach, which perceives local features such as the objects and obstacles around the vehicle and then predicts an action based on these local objects.

Deep learning models have seen applications in autonomous driving using both types of methods. The performance of these deep learning models has had great success in autonomous driving applications and simulations due to their learning ability. Neural network models have the ability to detect objects from an image and determine an optimal output action after extensive training on a autonomous driving dataset. The original paper trained a Mask-RCNN Benchmark neural network model (3) by using last frame scene images from the BDD100K dataset. The model can output driving actions and provide explanations to these actions. The published model from the original paper was reproduced and implemented to verify the authors' results. The model perceives local surroundings such as the objects and obstacles around a vehicle and output actions and the explanation of the actions.

2 Scope of reproducibility

The report used Mask-RCNN neural network model to solve the problem of stopping or redirecting an autonomous vehicle by detecting and recognizing obstacles and objects. The model in the report also solved the problems of reading and understanding traffic light to output right actions for an autonomous vehicle. Multiple labels can be output as an optimized action combination to avoid an obstacle so that the model of a autonomous vehicle has more intelligence. Including outputting actions, the model trained also output explanations for actions output. The model trained in the paper knows what to do and why to make these actions based on scene images.

Four actions (move forward, slow/stop, turn left, and turn right) and 21 explanations are considered and can be output from the model. The result of each sample can have multi-label since more options are available (e.g. turn left or stop). Both action and explanation contribute to the loss function in this system. The baseline model is proved to successfully detect action-inducing objects and make correct actions. One thing that can be improved is the small number of explanations. Reality and decision making during the activity of driving is far more complex and requires more extensive explanations.

- Claim 1: The proposed architecture is able to achieve substantial results and outperforms the Baseline model, which only takes global features into account.

3 Methodology

3.1 Model descriptions

Fast Region-based Convolutional Network (FastRCNN) was proposed in 2015 and is an extension of the Region-based Convolutional Network (RCNN) method. RCNN typically has expensive training in space and time and the object detection is slow (4). FastRCNN fixes these drawbacks and takes the image as input and then implements several convolutional layers and max pooling layers to the image data. The feature vectors will be extracted by the regions of interest (RoI) pooling layer and each feature vector will be fed into a set of fully connect layers. There will be two output branches: one is the probability of each object producing by softmax function, the other is a set of four values defining the position of bounding box (4).

Facebook AI research extended the FastRCNN model regarding the bounding box in 2017 and named the new model Mask-RCNN (3). Each detecting object will be given a segmentation mask that defines the boundary. This is accomplished by using a top-down architecture with lateral connections, called Feature Pyramid Network (FPN) for the purpose of semantic segmentation (5).

In the selected paper, although the authors refer to FastRCNN as the backbone to extract features, they actually employ MaskRCNN as the base model in the code provided on Github. This reproduction study will use MaskRCNN to replicate the results of the authors. The dataset consists of the last image frame of many video examples of driving (BDD100K). The MaskRCNN model is pretrained on BDD100K and then the weights will be adapted regarding our dataset.

Mask-RCNN is implemented to extract the features of action-inducing objects, which have strong causality with vehicle movement. Each action-inducing object will have an explanation of action change. The extracted features then will be trained by a global module as well as a local branch. In the global branch, the spatial context such as the location of action-inducing objects and the scene context are provided using two convolutional layers. In the local branch, each action-inducing object will be evaluated as a score and the top k objects will be selected. Finally, the vehicle action and correlated explanation will be predicted based on the synthesis of these two branches.

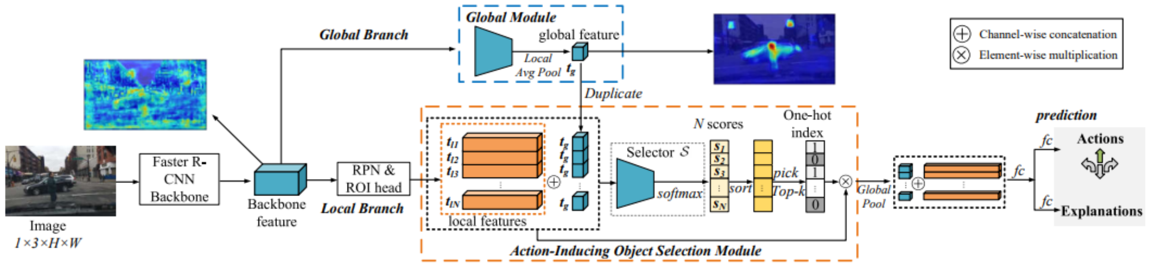


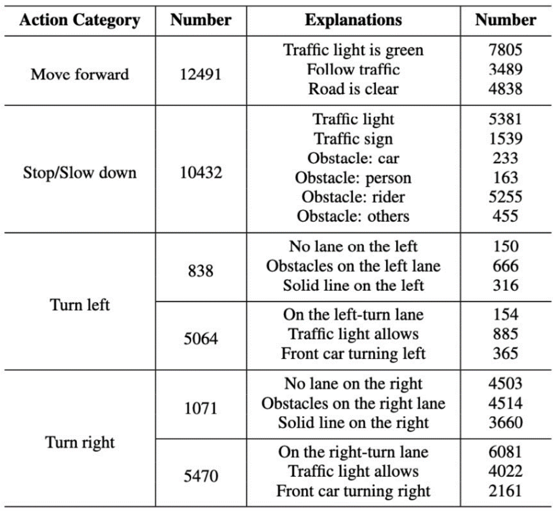
Figure 1: The proposed model architecture in the paper(2)

3.2 Datasets

The implemented dataset (BDD-OIA) is originated from BDD100K(6), which is a large and abundant driving video dataset collected in several cities for the purposes of computer version. BDD-OIA is a novel improvement to Bdd100K with the addition of explanations to the dataset annotations. Only images depicting scenes with greater than 5 pedestrians or five vehicles will be included in the training set. BDD-OIA is manually annotated by the original authors with four actions (move forward, slow/stop, turn left, and turn right) and twenty-one explanations to the reason why the action is valid. The distribution of actions and explanations are shown in Figure 2. There are 22924 five-second videos in the dataset and only the last image frame was used to train the original model. The number of training, validation, and testing samples are respectively 16082, 2270, and 4572. This gives the model a 70:10:20 training/validation/testing split of the data. The various splits of the data are categorized and annotated via JSON files that describes the image ids, filenames, action annotations, and explanation annotations. Interestingly, approximately 18000 of the first images in the JSON files contain annotations for five different actions. However, the remaining images have annotations for the four actions described in the original paper. It is believed the original authors initially wanted a model to describe five different actions, however opted for four actions half way through their labelling process. The dataset is provided by the original authors and can be easily accessed here(7).

3.3 Hyperparameters

The hyperparameters are set according to the original paper with the only exception being the batch size. In the original paper, the batch size is two. However, using a batch size of 2 will require training of approximately a week to complete. Therefore, the batch size has been increased to four during the reproduction's training. This essentially cuts the training time in half as there were only half the number of iterations per epoch. It is also worth noting that a batch size over four will cause a CUDA out of memory error on the Tesla T4 GPU from Google Colab Pro. This is because of the large image size and the fact that the pixel values are not scaled to 0-1. Instead, they are normalized with the pre-calculated image mean and standard deviation. It is believed the intention for this is to maintain the consistency with the pre-trained weights using FasterRCNN on the BDD100k dataset. Other hyperparameters are kept exactly the same as the original paper such as the initial learning rate of 0.001, weight decay of 0.0001, etc.



Action Category	Number	Explanations	Number
Move forward	12491	Traffic light is green	7805
		Follow traffic	3489
		Road is clear	4838
Stop/Slow down	10432	Traffic light	5381
		Traffic sign	1539
		Obstacle: car	233
		Obstacle: person	163
		Obstacle: rider	5255
		Obstacle: others	455
Turn left	838	No lane on the left	150
		Obstacles on the left lane	666
		Solid line on the left	316
	5064	On the left-turn lane	154
		Traffic light allows	885
		Front car turning left	365
Turn right	1071	No lane on the right	4503
		Obstacles on the right lane	4514
		Solid line on the right	3660
	5470	On the right-turn lane	6081
		Traffic light allows	4022
		Front car turning right	2161

Figure 2: label distribution(2)

3.4 Experimental setup and code

The code used was provided by the authors at this repo [https://github.com/Twizwei/bddoia_project]. Initially,it was attempted to set up Detectron2 instead of Mask-RCNN as Detectron2 is the successor of the latter. Mask-RCNN has recently been deprecated and no longer supported by Facebook AI. Though still powered by PyTorch and using the same backbone models, much have changed since the deprecation of Mask-RCNN and therefore it could not be run as intended on a local machine. One major difference is the format of the input data, which is a lot stricter and encourages the use of pre-formatted datasets such as COCO and CityScapes formatting. However, the dataset used in this project is a custom dataset and not easily adapted to the new requirements. Therefore, it was eventually decided to use Mask-RCNN in a Google Colab environment instead of on a local machine.

The setup requires a very specific version of Mask-RCNN, which can only be found in the provided repo. Since the experiments are performed on Google Colab, the repo was directly cloned and the corresponding dependencies were installed. The Mask-RCNN is also built completely using the provided version. The training and testing functions were slightly modified to accommodate for the different environment without changing the core architecture. The dataset is directly copied and unzipped from the provided Google Drive. For claim 1, a pre-trained model using Faster-RCNN on the BDD100k is used as the starting weights. Both the global and local features are extracted using the described structure in the paper and the top 10 action-inducing objects are selected. These features are then used to predict the action and explanations of the model predictions. Unfortunately, the model was only trained to the 25th epochs as opposed to the 50th epoch as described in the original paper due to the time constraints.

Once the training is finished, the inference was done using the provided testing code in the repo and evaluated using the pre-defined metrics from the authors. These include the accuracy of all 4 different actions (move forward, stop/slow down, turn left and turn right) in the test set as well as the overall and mean F1 scores for both the actions and the reasons predicted by the model.

3.5 Computational requirements

As discussed above, the computational requirement is quite heavy. All training is done on Google Colab using the Tesla T4 GPU. For a batch size of four, every 50 iterations takes 1.5 minutes, which means each epoch takes approximately 1.5 hours. The model is trained for 25 epochs, taking a total of 40 hours to complete. This is exceedingly demanding once you consider the limitations of Google COlab, where the user must be active every thirty minuted and a connection expires every 24 hours.

4 Results

As only one claim was investigated in this reproduction, the following section shows the results of the inference of the reproduced model compared to the results in the original paper.

4.1 Results reproducing original paper

Claim 1 states that the proposed architecture outperforms the baseline model, which is a pure global method that does not extract object features. Since this experiment does not include the training and inference for the baseline model, it was assumed that the said results for the baseline model from the paper are true. Here only the results with the results of the proposed architecture claimed in the paper were compared. Table 1 shows the comparison.

models	F	S	L	R	mF1	$F1_{all}$	explanation mF1	explanation $F1_{all}$
Baseline	0.755	0.607	0.098	0.108	0.392	0.601	0.180	0.331
original	0.829	0.781	0.630	0.634	0.718	0.734	0.208	0.422
ours	**0.808**	**0.799**	**0.455**	**0.564**	**0.657**	**0.695**	**0.228**	**0.467**

Table 1: Comparison of Results

In Table 1, F, S, L, R are the accuracy of the actions "move forward", "stop/slow down", "turn left" and "turn right". mF1 is the mean F1 score of the actions and $F1_{all}$ is the overall F1 score of the actions. Explanation mF1 is the mean F1 score of the reasons and explanation $F1_{all}$ is the overall F1 score of the reasons. The results of both the baseline model and the proposed architecture ("original") are taken from the paper while the last row shows the results of our experiment using the proposed architecture. As mentioned before, it was assumed that the results of the baseline model were true and will be verified in future works. Furthermore, Figure 3 and Figure 4 demonstrate the average training loss and average F1 score as training progresses. This is further discussed in the following section.

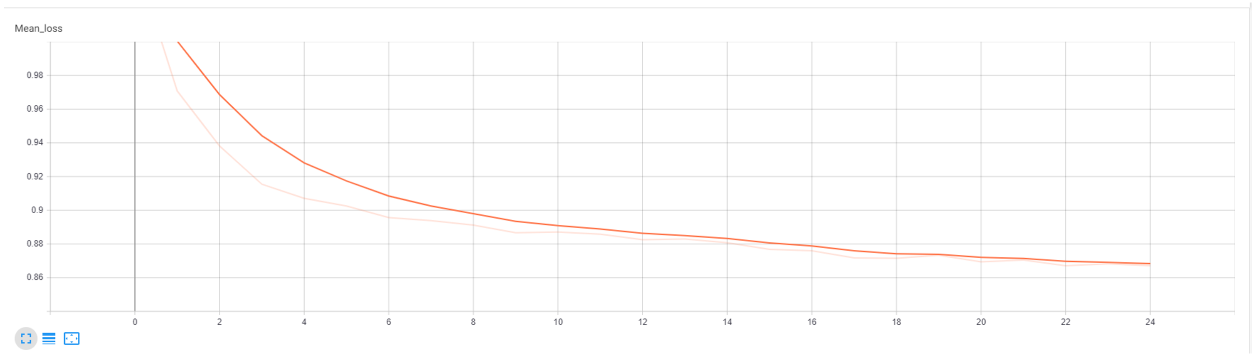


Figure 3: Mean Loss vs. Epoch

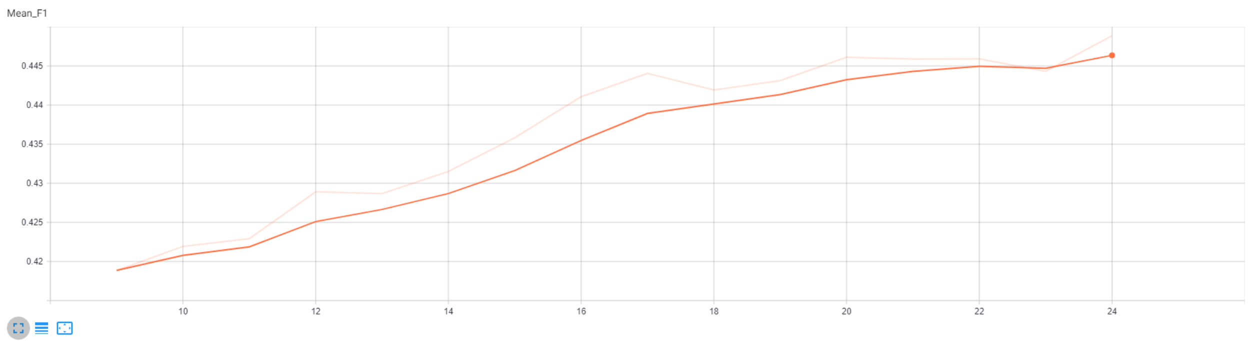


Figure 4: Mean F1 vs. Epoch

5 Discussion

5.1 Results comparison

It is clearly shown in Table 1 that the results of our experiment are quite close to the results claimed in the original paper, even though our training is a lot shorter. The accuracy of "move forward" and "stop/slow down" are extremely close to the proposed results while the other 2 action categories fall short significantly. However, as showns in Figure 3 and Figure 4, the average training loss shows a clear decreasing trend and is still decreasing at epoch 25. The average F1 score shows a clear increasing trend and is still increasing at epoch 25. This indicates that the model is not overfitting and there is still room for improvement if further training is performed. Furthermore, a surprising observation is that both the mean and overall F1 score of the explanations/reasons have outperformed the claimed results. It is believed that this will likely be re-adjusted as further training progresses considering that there is still a substantial gap between the F1 scores of the actions.

5.2 What was easy

The explanation of the purpose of the study was simple enough to understand. The model architecture itself initially seemed challenging yet manageable. However, in practice several external factors occurred that made this more difficult then anticipated. The dataset itself is a major contribution to action induced autonomous driving systems. It was very easy to access, download, and use on a local machine. For the purposes outlined in the study was effective at describing the actions and explanations of the driving scenes.

5.3 What was difficult

There were many difficulties that were faced when replicating the results proposed by the original authors.

The major component of the proposed system is the Faster-RCNN Backbone which derives backbone features that would be further separated into local and global features. The original author's code uses MaskRCNN-Benchmark, a project developed by Facebook Research to provide Faster-RCNN and MaskRCNN building blocks in PyTorch 1.0. However, since the original publication the MaskRCNN-Benchmark has depreciated and been replaced by Detectron2. This depreciation has effectively made the author's code from Github unusable on a local machine. Therefore, it was decided that reproduction could be conducted using Detectron2.

It was quickly determined however that adapting a Detectron2 architecture to the classification problem resulted in a defunct model that wouldn't accurately depict the system developed by the original authors. To start it would be necessary to rewrite the dataset's accompanying JSON files to work effectively with the Detectron2 model. The Detectron2 would also not work with the preprocessing and architecture published by the original authors. A significant amount of resources went into adapting the code to work with Detectron2 before an alternative option was determined.

It was determined that the original backbone feature extractor, MaskRCNN-Benchmark, could be utilized using a Google Colab Notebook. It was slight challenge to convert the files into a Colab notebook format. Google Colab was not ideal as it disconnects the running notebook if there is no activity for 30 minutes, or when a notebook run time has reached 24 hours. an ideal implementation would use a local machine, however there was no alternative.

The final difficult challenge was the high computational cost of training the model. The proposed architecture is complex with a large number of hyper-parameters. Coupled with the complex architecture, is the extensive dataset with a large sample of driving scenes. To save time, the batch size was changed from two to four. This change still resulted in a training time of over 100 hours using a Tesla T4 GPU. The computational expense of training the model resulted in major delays and frustrations through out the reproduction project.

5.4 Communication with original authors

No communication of any sort was initiated between the original authors or the reproduction authors.

References

[1] Kato, S., Takeuchi, E., Ishiguro, Y., Ninomiya, Y., Takeda, K., amp; Hamada, T. (2015). An Open Approach to Autonomous Vehicles. IEEE Micro, 35(6), 60-68. doi:10.1109/mm.2015.133

[2] Xu, Y., Yang, X., Gong, L., Lin, H., Wu, T., Li, Y., amp; Vasconcelos, N. (2020). Explainable Object-Induced Action Decision for Autonomous Vehicles. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr42600.2020.00954

[3] He, K., Gkioxari, G., Dollar, P., amp; Girshick, R. (2017). Mask R-CNN. 2017 IEEE International Conference on Computer Vision (ICCV). doi:10.1109/iccv.2017.322

[4] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.

[5] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 936-944, doi: 10.1109/CVPR.2017.106.

[6] https://bair.berkeley.edu/blog/2018/05/30/bdd/

[7] https://drive.google.com/drive/folders/1NzF-UKaakHRNcyghtaWDmc-Vpem7lyQ6