Detection of Lane Markings with Machine Learning

1st Muen Zhang
Dept. of Mechanical and Mechatronics Engineering
University of Waterloo
Waterloo, Canada
mezhang@uwaterloo.ca

***Abstract*—This document explores the process and results of detecting lane markings with Machine learning using the Holistically-Attracted Wireframe Parsing[1] framework used on the CULane[2] dataset and the llamas[3] dataset. It is found that the performance on the CULane dataset which includes overwhelmingly straight roads with one straight line connecting each end of the road performed extremely well with a F1 score of 60.7 with a 10 pixel threshold. The Llamas dataset had much more markings (sometimes redundant) as well as including curved roads performed much worse. A possible improvement on the llamas dataset could be to implement the projection matrix which may correct any mislabeled markings as well as alter the weightings of the losses to further encourage the detection of more junctions and edges.**

I. Introduction

The methods used in this document uses the Holistically-Attracted Wireframe Parsing[1] framework as well as the CULane[2] dataset and the llamas[3] dataset to produce the results and conclusions described. Throughout the paper, the main focus was getting the data in each of the datasets to match the HAWP format for the annotations. That is to say for each dataset, a train.json, val.json and test.json were created to train validate and test the dataset.

The train.json and val.json needed to have an array of elements each with the elements of "filename", "height", "width", "junctions", "edges_negative" and "edges_positive". The junctions contains all of the endpoints in the image and "edges_positive" and "edges_negative" is a list of lists with 2 elements (index of start and index of end).

The test.json needed to have an array of elements each with the elements of "filename", "height", "width", "junc" and "lines".

II. Methods Used

The Methods used to detect lane markings used the Holistically Attracted Wireframe Parsing[1] framework. The models were trained to 30 epochs with approximately 40000 images on each dataset.

A. CULane Dataset

The CULane[2] driver_23_30_frame dataset was used and out of the 62500 images, 37500 was used as training, 12500 was used as validation and 12500 was used as a test dataset. In order to get the annotations in the format that HAWP requires, an algorithm to parse each of the annotation files and compiling it into a single json file was used.

The annotation files consisted of up to 4 lines of points separated by a which each describe a line on the image. The edges would be made of the first and last point in each line. The algorithm to parse the annotations for the CULane[2] dataset takes advantage of the fact that most of the images are of straight roads. Therefore, when a line's midpoint was more than 10 pixels away from the line created from the two endpoints, the line would be discarded. The HAWP notations also requires the negative edges which could be determined by iterating through the combinations of the junctions appending it to the negative edges if it does not exist in the positive edges.

The model was warm started using the HAWP model as a starting point since it has a similar functionality.

B. Llamas Dataset

The Llamas[3] dataset color images were used for the train, validation and test sets. Out of the 58273 images in the data set, 38848 was used for training, and 9712 was used for validation and 9712 was used for testing. In order to get the annotations in the HAWP format, an algorithm was used to parse through each of the annotations and compile a single json file for training, validation and testing.

The annotation files consisted of several arrays of lines in its "lanes" component of the json. In the Llamas[3] dataset, the images have a great amount of images with curves, thus the same method used in CULane[2] cannot be used here, and instead, the curves were taken into account to observe how the framework handles several connected lines as part of the same lane.

The method used to compile the json annotation in the HAWP format was using each of the elements of the "markers" element of each lane using the element "pixel_start" and "pixel_end" as the start and end points.

However, since the dataset was automatically labelled, the data was quite noisy and contained several overlapping lines, thus an algorithm to check for each lane and discard any overlapping lines was used. From observation, the "markers" element of each lane, the markers were roughly ordered

Fig. 1. (a) Original annotated llamas image (b) Cleaned annotated llamas image

from the beginning of the lane to the end. The algorithm checked whether markers within a 2 marker range was the same direction (slope) within 5 degrees and within a certain distance from midpoints of 30 pixels and kept the longest of the similar lines for each marker. Below in Figure 1(a) and Figure 1(b), the results of the algorithm can be seen.

The llamas[3] models were warm started from model 21 of the CULane[2] models since they have very similar functionalities.

III. Results

A. CULane Dataset

After 30 epochs of training of 37500 images, the results can be seen below. It should be noted that the first 12 epochs were overwritten the second time the training was run through the checkpoint, and thus only epochs 13 - 30 are shown. Figure 2 shows the training and validation losses for the models at the end of each epoch.

From the above figures, it could be seen that the model trained until about epoch 23 without overfitting and after the 24th epoch, it seems to have found a local minimum which is very specific to the training set which is why the validation loss increases steadily afterwards.

From observations of all the losses, it can be determined that the 21st model is the best model because it has the lowest total loss and is situated prior to the overfitting at model 24.

Using this model, Figure 3 were predicted and the test and eval was run which resulted in a F1 score of **60.7** with a threshold of 10 pixels tolerance.

B. Llamas Dataset

The model was trained on the training dataset for 30 epochs in which it seemed to converge according to the training loss found in Figure 4, however, when testing on sample images, the performance was extremely poor as seen in Figure 5. Through multiple sample images, all of them detect lines in the sky instead of on the road.

IV. Conclusions

The performance of the HAWP framework when using the CULane[2] dataset resulted in a much better predictions and a very high F1 score. This could be due to the simplicity of the CUlane[2] dataset which only includes a maximum of 4

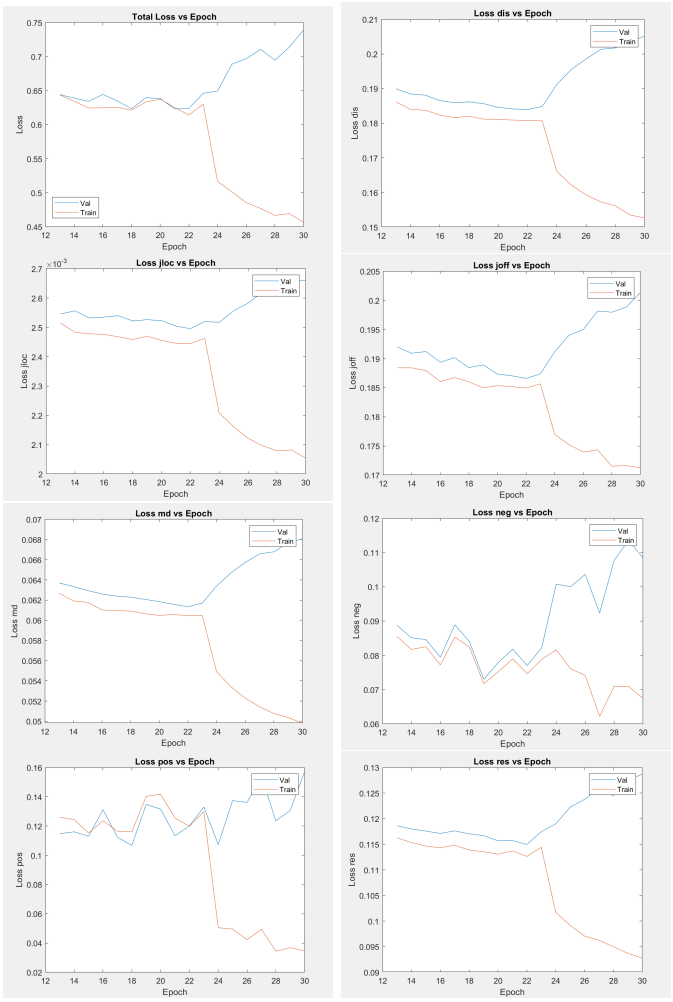

Fig. 2. (a) Total Loss (b) Loss Dis (c) Loss Junction location (d) Loss joff (e) Loss Angle (f) Loss Edge Negative (g) Loss Edge Positive (h) Loss Residual

Fig. 3. (a) Detecting lines without traffic (b) Detecting lines with traffic

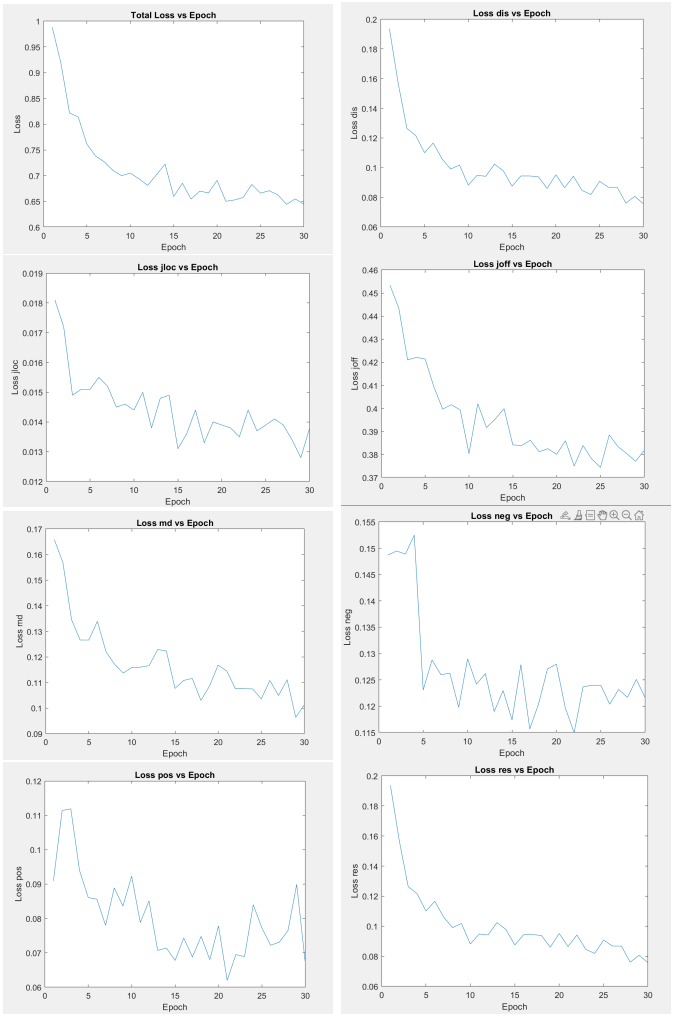

Fig. 4. (a) Total Loss (b) Loss Dis (c) Loss Junction location (d) Loss joff (e) Loss Angle (f) Loss Edge Negative (g) Loss Edge Positive (h) Loss Residual

Fig. 5. (a) Detecting lines in the sky with straight road (b) Detecting lines in the sky with curved road

lines in each image with very little noisy images. Thus the model can accurately predict the lane markings of straight roads very well.

However, on the Llamas[3] dataset, the data was not nearly as clean and had many overlapping lines. This may have resulted in the low performance of this dataset along with some other things. Another possible reason it did not perform as well was the projection matrix. In the llamas[3] annotations there was a projection matrix included which may need to be used to correct any misprojections on the annotations. This could explain why most of the predictions of lane markings were above the actual lane markings. Another thing that seemed to occur is that the model would predict curved lines on most images even with straight roads. This could be due to the fact that the curved roads have much more markings and thus have a higher weighting on the end model.

V. Next Steps

The next steps to be taken to further enhance the model to detect curved and dashed lines could be to further clean the data on the llamas[3] dataset as well as apply the projection matrix to ensure that the markings are on the lines on the image.

Another next step could be to alter the weightings of the loss on the configuration file which could punish the junctions, edges positive and edges negative much more than it punishes the angle and residual detection.

VI. References

References

[1] N. Xue, T. Wu, S. Bai, F. Wang, G.-S. Xia, L. Zhang, and P. H. Torr, "Holistically-Attracted Wireframe Parsing," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[2] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial As Deep: Spatial CNN for Traffic Scene Understanding," AAAI Conference on Artificial Intelligence (AAAI), 2018.

[3] K. Behrendt and R. Soussan, "Unsupervised Labeled Lane Markers Using Maps," 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 2019.