IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, June 2022
Impact Factor: 6.252

Road Lane Detection using Convolutional Neural Network

Farjana Farvin S[1], Sowndarya S V[2]
Assistant Professor, Computer Science and Engineering[1]
Student, Computer Science and Engineering[2]
Anjalai Ammal Mahalingam Engineering College, Thiruvarur, India

Abstract: *Road Lane detection is necessary in autonomous driverless vehicle that is used to automatically detect lane lines in a road. With the increase in the modern day's population, which has in turn increased the number of vehicles in the road, people are more prone to accidents. Increase in the number of vehicles, human errors towards traffic rules and the difficulty to oversee situational dangers by drivers are contributing to the majority of accidents on the road. Therefore, in this paper we have designed a methodology to detect road lane lines which, helps our drivers from causing life damage and contributes to safe driving. Here we have used deep leaning algorithm which has shown a quiet good accuracy rate.*

Keywords: Deep Learning, Image Pre-processing, Convolution Neural Network, Max Pooling, Filtering, Flattening, Feature extraction, Hidden Layer, etc.

I. INTRODUCTION

In the fast-growing tech world, many things have become easy. People now prefer to live a more luxurious and sophisticated life. Though we have inventions to make most of the human activities simpler, we lack technologies that has to be brought into immediate actions to avoid losing human lives. One such technology is auto detecting road lanes while somebody travels in a road. In country like India which has a high human population introducing this system could prevent people from major tragedies.

In this paper, we have developed a neural network model to detect road lane lines where we have downloaded dataset from http://web4.cs.ucl.ac.uk/staff/g.brostow/MotionSegRecData/files/701_StillsRaw_full.zip. We will also have another dataset to perform feature extraction, in which dataset is created, manually by picking up the images from internet such that the pictures contain features that has to be extracted manually. This dataset is used to achieve 20% of the feature extraction that has to manually fetched. The remaining 80% of the features will be automatically fetched.

The first step begins with the pre-processing of the images of the downloaded and collected datasets. A directory is created in google drive to store the pre-processed images of the main dataset and the dataset designed for feature extraction. The process will then proceed by creating a convolutional neural network model. Here we will have convolutional layer and pooling layer in which the input images are broken down into a matrix of specified size. As many as the convolutional and pooling layers are created that much level of accuracy is gained. Neural networks aim in mimicking the human brain. So, the CNN model tries learning from the datasets as right as possible, leading to a high efficiency of the model.

This paper structured as follows: Segment 2 presents the approaches that are currently in existence. Segment 3 describes our methodology and its several modules. Segment 4 talks about the implementation of the modules and the result obtained. Segment 5 concludes the paper and presents a few outlooks.

II. RELATED WORK

In this section we have many literatures related to road lane detection.

In [1], authors have designed a Convolutional neural network to detect road lanes. They have also described about the basic networks like ZFNet network.

In [2], authors proposed a machine vision technique to detect lanes in road. They have provided a wide knowledge on how to use cv2 libraries and how to pre-process an image dataset.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, June 2022
Impact Factor: 6.252

In [3], the authors have a clear approach on instance segmentation in edge-cloud computing. Here they have given a clear description on how edge-computing go in-hand with the deep learning algorithms, such as CNN and RNN and etc.
In [4], the authors have represented a clear view on how regression-based lane detection model works the paper has also described in detail about lane type classification stages like resnet101-based lane type classification model. This has also depicted a high knowledge on how the calculations are done in the background in a CNN model.
In [5], the authors have shown in detail about how the neural networks work in the background and has also been depicted a clear picture of RANSAC algorithm. Authors have depicted how CNN will take over when RANSAC algorithm fails.
Authors In [6], have given information about how road lane detection can increase vehicle guidance, when it is a highly populated vehicles in a region or road. It has shown an overview on how intelligent vehicles work and hough transforms guidelines. Here we can also learn about how one can perform noise reduction from an image dataset.
Authors In [7], presents many experiments that have been undertaken to turn road lane detection into a virtual assistant to increase driving safety and prevent car accidents, particularly with autonomous vehicles.
The authors in [8], depicted road lane detection using dashed and solid road lanes using visible light camera sensor.
In [9], the authors have designed a methodology to detect road lane detection and have used it under various circumstances and have performed the experiment in various regions like urban, rural, narrow and broad roads and the result has come up with high accuracy rate.
The advantage of all these methodologies discussed in all these above-mentioned reference papers are quite obvious to predict the image correctly as close as possible and discussing about various methods at each level has been a huge information for once to start with deep learning neural networks.
The disadvantage is that it doesn't clearly mention how each process is performed in the background. The experiment with this technology has been performed in a very little region and still there are regions where this has to be tested like hill stations and mountain regions. In the methodology that we have designed, it will be tested under all the circumstances possible.

III. METHODOLOGY

Our main is to detect road lane lines when a person is travelling by road. Here we have put up the system architecture in a simple and in a clear way. The system architecture depicted here is clearer to explain what is done in this whole methodology. The system architecture of the proposed system is in the figure1.

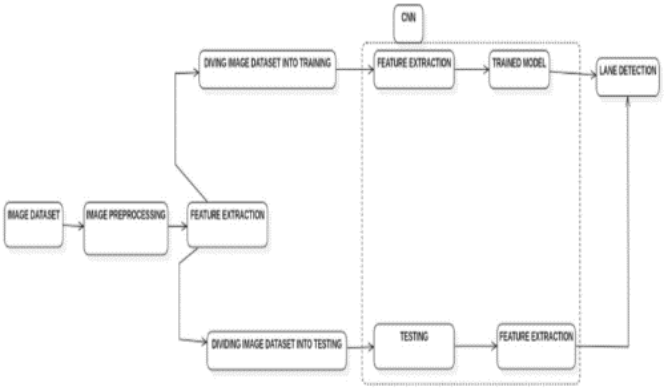

Figure 1: System Architecture

A. IMAGE PRE-PROCESSING

Image pre-processing is a technique where images are removed from noises. Noises here refers to unwanted information and colour patterns in the images like RGB colours. Image pre-processing includes converting RGB or BGR colours to grayscale images, just to make easy for the model to process the images. We use libraries like sklearn, keras library to pre-process the dataset. Deep learning algorithms when applied on improperly pre-processed dataset, can give irrelevant results with a poor accuracy rate. So, image pre-processing is the most important step in this entire methodology. Once the image is pre-processed then they have to be stored in a separate folder in the drive.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, June 2022
Impact Factor: 6.252

ALGORITHM

1. Mount google drive into google colab.
2. Create an empty list, images=[].
3. Fetch the address of each location and save it in img_loc list.
4. Define a function img_read_and_show(img_loc):
   - 4.1 for img<---- img_loc[0] to img_loc[-1]:
     - 4.1.1 read each img and append it to images list.
   - 4.2 return images.
5. Call img_read_and_show(img_loc) function.
6. read_images= images.
7. Create an empty list images=[]
8. Define function image_preprocessing(img_loc):
   - 8.1 for image<---- img_loc[0] to img_loc[-1]:
     - 8.1.1 image<----read each image.
     - 8.1.2 img<----resize the image to a proper size.
     - 8.1.3 img<----convert to gray.
     - 8.1.4 Append img to images.
9. Create a directory and store the preprocessed images in the images list into it.

B. FEATURE EXTRACTION

Collect Feature extraction the image dataset and then we convert its height and width to the required size. We then convert the resized images into grayscale images. Split train and test data in the feature extraction images. We then train and split them. In CNN 20% of the feature extraction, i.e., the main features are given by the user and the 80% of the features are extracted by the CNN model itself. So, this step is to achieve the above said 20% feature extraction process. The feature extracted images are then stored into separate directory which is used to train and test our main dataset, according to the features extracted.

ALGORITHM

1. Upload feature extraction images into the colab notebook.
2. Import sklearn.model_selection.
3. Create an empty list data=[]
4. Create train=[]
5. Create test=[]
6. Call the function image_preprocessing and pass the image location.
7. Store the preprocessed feature extraction images into a separate folder.
8. Define a variable called categories=[Trees,Vehicle].
9. For category <-----categories:
   - 9.1 label= index number of category in categories list.
   - 9.2 for img<----- in category folder:
     - 9.2.1 Read each img.
     - 9.2.2 Append [img,label] into data list.
   - 9.3 length= len(data)
   - 9.4 if label ==0:
     - 9.4.1 Append train=data[0:(len/2)]
     - 9.4.2 Append test=data[((len/2)+1):]
   - 9.5 Store length in pdl.
   - 9.6 if label ==1:
     - 9.4.1 Append train=data[pdl:((length-pdl)/2)]
     - 9.4.2 Append test=data[((length-pdl)/2)+1):len]
10. Shuffle train and test lists.
11. Now train and test lists contains images and labels now split the labels and images in train into two separate list. train_img=images in train lists. train_label= labels in train list.
12. test _img = images in test list. test_label=label in test list.
13. Now feed train_img(input), train_label(output) inside fit method to train the model.
14. Now feed the test_img(input) into predict method. And predict the output.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, June 2022
Impact Factor: 6.252

C. TRAIN-TEST SPLIT

Here we could use various libraries to split data. But the one that we have actually used is sklearn library. Now train the main dataset images and according to the features that the system classifies them, they are stored into different folders. We first need to import sklearn library by simply including the below statement from sklearn.model_selection import train_test_split.

ALGORITHM

1. Store the location of main preprocessed images into img_loc.
2. Create an empty list images=[].
3. for img<---img_loc:
   3.1 image<----Read each img.
   3.2 Append image to images list.
4. Create an empty list list_0=[]
5. Create an empty list list_1=[]
6. Create an empty list list_others=[]
7. for img<----images[0] to len(images)-1:
   7.1 predict(img).
   7.2 if predict(img)==0:
      7.2.1 Append img into list_0.
   7.3 elif predict(img)==1:
      7.3.1 Append img into list_1.
   7.4 else:
      7.4.1 Append img into list_others.
8. Create an empty list train_set=[].
9. Create an empty list test_set=[].
10. len_0=len(list_0)
11. len_1=len(list_1)
12. len_oth=len(list_others)
13. Append list_0[0:(len_0)/2], list_1[0:(len_1)/2], list_others[0:(len_oth)/2] into train_set list.
14. Append list_0[(len_0)/2)+1:], list_1[(len_1)/2+1:], list_others[(len_oth)/2)+1:] into test_set list.

D. CLASSIFICATION

Here, we will create the convolutional neural network model, where we will first import the Sequential class from tensorflow.keras.models library. In CNN, we have 2 layers one is convolutional layer and pooling layer. To create convolutional layer and pooling layer, we should first import Conv2D and MaxPooling modules. We will use add method to add convolutional and Max Pooling layer. Similarly, we could have as many convolutional and pooling layer as required. Once we have created the required number of convolutional and pooling layers, we will then apply flattening, which is then passed on through the hidden layers.

As the number of hidden layers and convolutional layer increases, we could expect our model to achieve a better accuracy rate. But we should also be careful that the model should not overfit.

ALGORITHM

1. Create CNN model.
2. Pass the train_set into CNN model and train them.
3. Analyse the model.
4. Pass the test_set into the CNN model.
5. Predict the features.
6. Remove the features and highlight the lane.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, June 2022
Impact Factor: 6.252

E. PREDICTION AND DETECTION

Finally, once we are done with neural network design, training and testing our image datasets, we will then pass on some completely new images to the neural network to see how far our model could perform well on a new data about which it has absolutely zero knowledge. We have also used feedback algorithm which will get the input from the user to make sure if it has predicted accurately on the new image dataset or not. According to the user's response our model will try enhancing itself. If the user's input says the prediction is accurate, then the model will work at the same level. If not, the model will make the neural network more efficient and then try applying the methodology in it.

ALGORITHM

1. Feed a new image.
2. Perform preprocessing.
3. Feed it into CNN model.
4. Detect the features.
5. Remove the detected features.
6. Highlight the lane.
7. Display the final output image.

IV. IMPLEMENTATION AND RESULTS

The image dataset that we have taken for the road lane detection has been predicted and detected with a good amount of accuracy rate. Here we have shown the output of scalers for the train and validation dataset that has been separated from the original main dataset. This scaler shows how the validation and train set perform well on each model individually. So, this can take into account to evaluate our model.

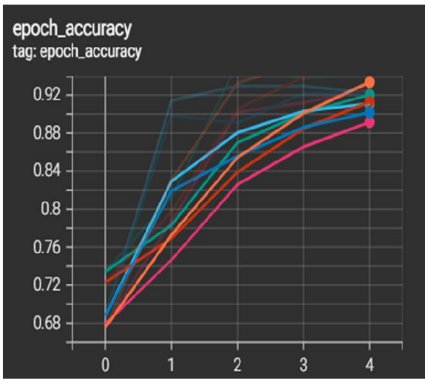

Figure 2: epoche accuracy showing train and validation set's accuracy.

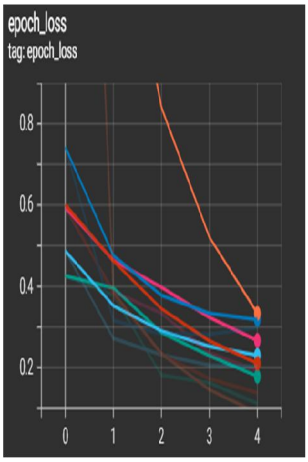

Figure 3: epoche loss showing train and validation set's loss

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, June 2022
Impact Factor: 6.252

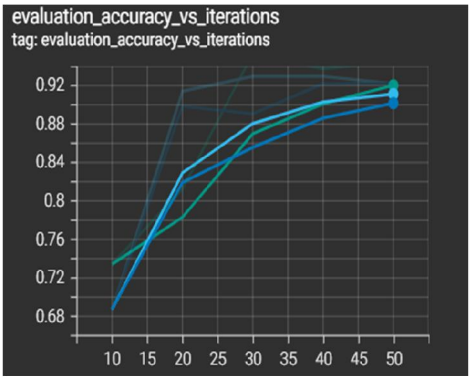

Figure 4: evaluation accuracy showing train and validation set's accuracy

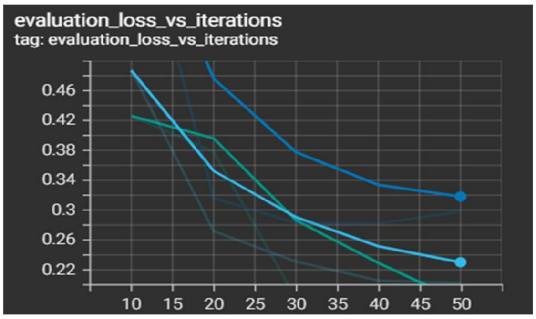

Figure 5: evaluation accuracy showing train and validation set's loss

```
def image_preprocessing(read_images_for_preprocessing):
  images=[]
  for image in read_images_for_preprocessing:
    img=cv2.imread(image)
    img=cv2.resize(img,(IMG_SIZE,IMG_SIZE))
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    images.append(gray)
  return images
```

Figure 6: Figure showing image pre-processing code

```
def store_preprocessed_images(preprocessed_images,path):
  i=1
  for pre_img in preprocessed_images:
    img=str(i)+".jpg"
    i=i+1
    filename = path+"/"+img;
    cv2.imwrite(filename,pre_img)
```

Figure 7: Figure showing code for storing pre-processed images

```
logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)
```

Figure 8: Figure showing code for creating logs for each model

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, June 2022
Impact Factor: 6.252

```
model=Sequential()

model.add(Conv2D(64,(2,2),activation = "relu"))
model.add(MaxPooling2D((2,2)))

model.add(Conv2D(64,(2,2),activation = "relu"))
model.add(MaxPooling2D((2,2)))


model.add(Conv2D(64,(2,2),activation = "relu"))
model.add(MaxPooling2D((2,2)))
```

Figure 9: Figure showing code for creating CNN model

```
model.add(Flatten())
model.add(Dense(128, input_shape=x.shape[1:],activation="relu"))
model.add(Dense(128,activation="relu"))
model.add(Dense(2,activation="Softmax"))
```

Figure10: Figure showing code for creating hidden layers and output neurons

```
model.fit(x,y,epochs=5,validation_split=0.3,batch_size=32,callbacks=[tensorboard])
```

Figure11: Figure showing code for Training the model

V. CONCLUTION

To handle the problem of lane detection in difficult conditions such as shadows and obstructions, the lane detection methods based on classical image processing and the lane detection approach based on deep learning are analysed and contrasted. A two-branch training network and a tailored training network based on semantic segmentation are proposed to improve the efficiency of central computing, and the combination of vehicle edge computation and actual engineering can improve the effect of lane line detection networks. When the ground moves in the inverse perspective transformation, the usage of a fixed transformation matrix causes mistakes, causing the vanishing point projected to infinity to move up or down.

The model's stickiness is enhanced by the fact that the prediction is based on the input image, which allows the network to modify the projection parameters when the ground plane changes. The results of the final testing suggest that the model presented in our research performs better in conditions such as poor lighting and lane line erosion.

REFERENCE

[1] Junfeng Li, Dehai Zhang, Yu Ma and Qing Liu. Lane Image Detection Based on Convolution Neural Network Multi-Task Learning, by the authors, by the authors. by the authors. Licensee MDBI, Basel, Switzerland, Published: 27 September 2021.

[2] Xining Yang, Dezhi Gao, Jianmin Duan, and Lei Yang. Research on Lane Detection Based on Machine Vision. Intelligent Measure & Control Laboratory, Beijing University of Technology, Beijing, China.

[3] Wei Wang1, Hui Lin2 and Junshu Wang3,4. CNN based lane detection with instance segmentation in edge-cloud computing. https://doi.org/10.1186/s13677-020-00172-z.

[4] Nima Khairdoost, Steven S. Beauchemin and Michael A. Bauer. Road Lane Detection and Classification in Urban and Suburban Areas based on CNN. Department of Computer Science, The University of Western Ontario,London, ON, N64-5B7, Canada. 10.5220/0010241004500457.

[5] Jihun Kim and Minho Lee. Robust Lane Detection Based on Convolutional Neural Network and Random Sample Consensus. School of Electronics Engineering, Kyungpook National University. 1370 Sankyuk-Dong, Puk-Gu, Taegu 702-701, South Korea.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 1, June 2022
Impact Factor: 6.252

[6] Othman Omran Khalifa, Sheroz Khan, Md. Rafiqul Islam. Vision Based Road Lane Detection System for Vehicles Guidance. Article in Australian Journal of Basic and Applied Sciences · May 2011. https://www.researchgate.net/publication/228958566.

[7] Jiyoung Jung 1 ID and Sung-Ho Bae 2. Real-Time Road Lane Detection in Urban Areas Using LiDAR Data. 6 September 2018; Accepted: 24 October 2018; Published: 26 October 2018.

[8] Toan Minh Hoang, Hyung Gil Hong, Husan Vokhidov and Kang Ryoung Park, Road Lane Detection by Discriminating Dashed and Solid Road Lanes Using a Visible Light Camera Sensor. Received: 26 April 2016; Accepted: 15 August 2016; Published: 18 August 2016.

[9] Zhou, S.; Jiang, Y.; Xi, J.; Gong, J.; Xiong, G.; Chen, H. A Novel Lane Detection Based on Geometrical Model and Gabor Filter. In Proceedings of the IEEE Intelligent Vehicles Symposium, San Diego, CA, USA, 21–24 June 2010; pp. 59–64.

[10] Shin, J.; Lee, E.; Kwon, K.; Lee, S. Lane Detection Algorithm Based on Top-View Image Using Random Sample Consensus Algorithm and Curve Road Model. In Proceedings of the 6th International Conference on Ubiquitous and Future Networks, Shanghai, China, 8–11 July 2014; pp. 1–2. Shin, J.; Lee, E.; Kwon, K.; Lee, S. Lane Detection Algorithm Based on Top-View Image Using Random Sample Consensus Algorithm and Curve Road Model. In Proceedings of the 6th International Conference on Ubiquitous and Future Networks, Shanghai, China, 8–11 July 2014; pp. 1–2.

[11] Benligiray, B.; Topal, C.; Akinlar, C. Video-Based Lane Detection Using a Fast-Vanishing Point Estimation Method. In Proceedings of the IEEE International Symposium on Multimedia, Irvine, CA, USA, 10–12 December 2012; pp. 348–351.

[12] Li, Z.; Cai, Z.-X.; Xie, J.; Ren, X.-P. Road Markings Extraction Based on Threshold Segmentation. In Proceedings of the International Conference on Fuzzy Systems and Knowledge Discovery, Chongqing, China, 29–31 May 2012; pp. 1924–1928.

[13] Jordan, B.; Rose, C.; Bevly, D. A Comparative Study of Lidar and Camera-based Lane Departure Warning Systems. In Proceedings of the ION GNSS 2011, Portland, OR, USA, 20–23 September 2011.

[14] Jordan, B.; Rose, C.; Bevly, D. A Comparative Study of Lidar and Camera-based Lane Departure Warning Systems. In Proceedings of the ION GNSS 2011, Portland, OR, USA, 20–23 September 2011.

[15] Qi, H.; Moore, J. Direct Kalman filtering approach for GPS/INS integration. IEEE Trans. Aerosp. Electron. Syst. 2002, 38, 687–693.

[16] Zhang Z, Schwing AG, Fidler S, Urtasun R (2015) Monocular object instance segmentation and depth ordering with cnns. In: Proceedings of the IEEE International Conference on Computer Vision. pp 2614–2622. https://doi.org/10.1109/iccv.2015.300

[17] Yang S, Wu J, Shan Y, Yu Y, Zhang S (2019) A novel vision-based framework for real-time lane detection and tracking. Technical report. SAE Technical Paper. https://doi.org/10.4271/2019-01-0690

[18] Liu, T., Chen, Z., Yang, Y., Wu, Z., and Li, H. (2020). Lane detection in low-light conditions using an efficient data enhancement: Light conditions style transfer. arXiv preprint arXiv:2002.01177.

[19] John V, Liu Z, Guo C, Mita S, Kidono K (2015) Real-time Lane estimation using deep features and extra trees regression. In: Image Video Technol. Springer. pp 721–733. https://doi.org/10.1007/978-3-319-29451-3_57

[20] Qi L, Zhang X, Dou W, Ni Q (2017) A distributed locality-sensitive hashing-based approach for cloud service recommendation from multi-source data. IEEE J Sel Areas Commun 35(11):2616–2624

[21] Ganin AA, Mersky AC, Jin AS, Kitsak M, Keisler JM, Linkov I (2019) Resilience in intelligent transportation systems (its). Transp Res Part C Emerg Technol 100:318–329.