International Journal For Technological Research In Engineering
Volume 8, Issue 11, July-2021 *ISSN (Online): 2347 - 4718*

AUTOMATED DENT DETECTION USING CNN

[1]Himanshu Bhardwaj, [2]Adarsh Lunthi, [3]Hitesh Bhat, [4]Karan Singh Rawat, [5]Jogendra Kaushik
[1,2,3,4]student, [5]Assistant Professor
Department of Information Technology
ADGITM, Delhi, India

Abstract: - This project is all about to reduce the time taken by companies for creating the insurance policy and completing the insurance claim. We make this project to revolutionize the process of applying insurance policies to vehicles; we try to speed up the process of insurance to the various vehicles by making the whole process automated. Our project helps the insurance company to make the insurance-related decision faster and remove the middleman between the customer and the insurance companies, Users can by themselves login to our platform for an insurance purpose and it will get their insurance done in a few minutes. We make an automated system with the help of ML that will identify the condition of vehicles and generate the amount of policy of insurance that will be required by the customer. This all will be done with a website or a simple user dashboard in which the user has to login to our website and then they will provide the required information and then will upload the pictures of the vehicle from required sides and in required and appropriate parameters. After all this procedure an amount will be projected from our side based on the condition of the vehicle and will be verified by the insurance issuing company whether they are satisfied by the condition and pass the claim further. The decision of the company will be projected on the dashboard and the user will have the choice whether they want to go for insurance or not depending on the claim pricing. If they are satisfied with the insurance amount and conditions they can apply to get their insurance by paying online and get the insurance paper through our website. So we make the whole insurance process online and faster by removing the middle-man and manual time taking processes.

1. INTRODUCTION

Today, in the car insurance industry, a lot of money is wasted due to claims leakage. Claims leakage / Underwriting leakage is defined as the difference between the actual claim payment made and the amount that should have been paid if all industry leading practices were applied. Visual inspection and validation have been used to reduce such effects. However, they introduce delays in the claim processing. There has been efforts by to few start-ups to mitigate claim processing time An automated system for the car insurance claim processing is a need of an hour. In this paper, we employ Convolutional Neural Network (CNN) based methods for classification of car damage types. Specifically, we consider common damage types such as bumper dent, door dent, glass shatter, head lamp broken, tail lamp broken, scratch and smash. To best of our knowledge, there is no publicly available dataset for car damage classification. Therefore, we created our own dataset by collecting images from web and manually annotating them. The classification task is challenging due to factors such as large inter-class similarity, barely visible damages. We experimented with many techniques such as directly training a CNN, pre-training a CNN using auto-encoder followed by fine-tuning, using transfer learning from large CNNs trained on Imagenet and building an ensemble classifier on top of the set of pre-trained classifiers. We observe that transfer learning combined with ensemble learning works the best. We also device a method to localize a particular damage type. Experimental results validate the effectiveness of our proposed solution. Object Detection is the challenging artificial intelligence problem of detecting any desired object from any image containing multiple objects in a single image. It requires both image understanding from the domain of computer vision and a semantic segmentation analysis of an image. It is important to consider and test multiple ways to frame a given predictive modeling problem and there are indeed many ways to frame the problem of detecting object in an image.

2. DATASET DESCRIPTION

As far as we know there is no publicly available dataset for car damage classification, without a large and diverse dataset it becomes rather difficult to apply standard computer vision techniques, therefore we created our own dataset by collecting images using web crawling as done by We manually filtered and categorized images into 7 commonly observed damage types We also collected images belonging to No Damage class. Different damage types; bumper dent, scratches, door dent, glass shatter, head-lamp broken, tail-lamp broken and smashed respectively. For the purpose of detection, we manually annotated different types of damaged regions.

3. CONVOLUTION NEURAL NETWORK

The human brain processes a huge amount of information the second we see an image. Each neuron works in its own receptive field and is connected to other neurons in a way that they cover the entire visual field. Just as each neuron responds to stimuli only in the restricted region of the visual field called the receptive field in the biological vision system, each neuron in a CNN processes data only in its receptive field as well. The layers are arranged in such a way so that they detect simpler patterns first (lines, curves, etc.) and more complex patterns (faces, objects, etc.) further along. By using a CNN, one can enable computer eyesight. The convolution layer is the core building block of the CNN. It carries the main portion of the network's computational load. This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels. During the forward pass, the kernel slides across the height and width of the image-producing the image representation of that receptive region. This produces a two-dimensional representation of the image known as an activation map that gives the response of the kernel at each spatial position of the image. The sliding size of the kernel is called a stride. If we have an input of size W x W x D and Dout number of kernels with a spatial size of F with stride S and amount of padding P, then the size of output volume can be determined by the following formula:

Wout = W-F+2P/S+1

4. TRAINING A CNN

1. Download the tensorflow models zip file from the link given below and extract the content from it. 'https://github.com/tensorflow/models/archive/7c0e458.zip'

2. Download and extract its content to models\research\object_detection folder. http://download.tensorflow.org/models/object_detection/faster_rcnn_inception_v2_coco_ 2018_01_28.tar.gz'

3.Create environment conda create -n objdet python=3.6.5 anaconda

4.Activate the environment

5.activate objdet

6.Run this to install all the required packages
pip install -r models\research\object_detection\requirements.txt

7.Set required paths set PYTHONPATH=C:\Users\yt\Documents\tf1\models;C:\Users\yt\Documents\tf1\mo dels\research;C:\Users\yt\Documents\tf1\models\research\slim set PATH=%PATH%;PYTHONPATH

8.Compile Protobufs protoc object_detection/protos/*.proto --python_out=.

9.Run Setup
cd Documents/tf1/models/research

python setup.py build

python setup.py install

10.Add all training images to images/train_unaugmented and testing images to images/test.

11. Label images using LabelImg.

12. Make labels from xml files for
images/train_unaugmented and images/test. which can be found as
'images/train_unaugmented_labels.csv' & 'images/test_labels.csv'. python xml_to_csv.py

13. To augment images. Images inside images\training will be augmented and for each image 4 new images will be generated.

14. python ImageAugmentor_imgaug.py

15. To add the class to detect in Images. Open generate_tfrecord.py file and change the label map starting at line
31 with your own label map.

16.Then generate tfrecords for training purpose. This will generate train.record and a test.record file in \object_detection

17. python generate_tfrecord.py --csv_input=images\train_labels.csv

--image_dir=images\train --output_path=train.record python generate_tfrecord.py -- csv_input=images\test_labels.csv --image_dir=images\test

--output_path=test.record

18. Create a label map inside object_detection\training folder with .pbtxt extension. Sample label map is given below.

```
item {
id: 1

name: 'class_name'

    }
```

19. Configure training: Navigate to \object_detection\samples\configs and copy the faster_rcnn_inception_v2_pets.config file into the \object_detection\training directory. Then, open the file there are several changes to make to the .config file, mainly changing the number of classes and examples, and adding the file paths to the training data.

20 .Line 9. Change num_classes to the number of different objects you want the classifier to detect.

1. Line 106. Change fine_tune_checkpoint to: fine_tune_checkpoint:"C:/tensorflow1/models/research/object_detection/faster_rcnn_inception_v2_coco_2018_01 _28/model.ckpt"

2. Lines 123 and 125. In the train_input_reader section, change input_path and label_map_path to: input_path : "C:/tensorflow1/models/research/object_detection/train.record" label_map_path: "C:/tensorflow1/models/research/object_detection/training/labelmap.pbtxt"

3. Line 130. Change num_examples to the number of images you have in the \images\test directory.

4. Lines 135 and 137. In the eval_input_reader section, change input_path and label_map_path to: input_path : "C:/tensorflow1/models/research/object_detection/test.record "label_map_path:"C:/tensorflow1/models/research/object_detection/training/labelmap.pbtxt"

21. To Train

```
python train.py --logtostderr --train_dir=training/
--
pipeline_config_path=training/faster_rcnn_inception_v2_pets.config
```

1. FOR RESUMING TRAINING: make sure to update the 'fine_tune_checkpoint' parameter inside 'training/faster_rcnn_inception_v2_pets.config' file with the latest name of latest model which you have after training inside 'training' folder and then run the below command.

```
python train.py --logtostderr --train_dir=../training/ --pipeline_config_path=../training/faster_rcnn_inception_v2_pets.config
```

2. To Test

```
python eval.py --logtostderr --eval_dir=../eval/

--
pipeline_config_path=../training/faster_rcnn_inception_v2_pets.config

--checkpoint_dir=../training
```

3. For Tensorboard (Open command prompt as admin and then change the path to objecte_detection folder in it)

4. Training Tensorboard

```
tensorboard --logdir=training
```

5. Testing Tensorboard
6. tensorboard --logdir=eval

7. To Generate inference graph; First move old 'saved_model' folder and 'frozen_inference_graph.pb' to a safe place(if those exists). Make sure to update the value of xxxxx in the below code to match the model number in your training folder. python export_inference_graph.py --input_type image_tensor --pipeline_config_path. training/faster_rcnn_inception_v2_pets.config-- trained_checkpoint_prefix training/model.ckpt-xxxxx --output_directory inference_graph

8. To try it on custom images; Put all the images in images/extras folder then run script and see results in images/processed folder

9. python Object_detection_image.py

5. TRANSFER LEARNING

So as we have problems of over-fitting on small datasets to avoid this we plan to use the Transfer learning instead of training the CNN model from scratch this approach results in the significant improvement on classification problems when the available dataset to us is limited or scarce. So we decided to carry forward our CNN training on the already available large dataset. Also training our car dataset on large dataset it is better to learn features on a more different or diverse datasets. After training on Image net dataset, we retrain the classifier on top of the CNN on our dataset. We also fine-tune all the layers of the CNN while keeping in mind that the earlier layers learn more generic features that are common in all classification tasks .

6. VGG19

Convolutional Neural Networks (CNN) have been used for several image classification tasks. They require a lot of data and time to train. However, sometimes the data set may be limited and not enough to train a CNN from scratch. In such a scenario it is helpful to use a pre-trained CNN, which has been trained on a large data set. We will use VGG-19 pre-trained CNN, which is a 19-layer network trained on Imagenet .VGG-19 is a convolutional neural network that is 19 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database.The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224. For more pretrained networks in MATLAB.

7. CONCLUSION

In this paper, we proposed a deep learning based solution for car damage classification. Since there was no publicly available dataset, we created a new dataset by collecting images from web and manually annotating them. We experimented with multiple deep learning based techniques such as training CNNs from random initialization, Convolution Autoencoder based pre-training followed by supervised fine tuning and transfer learning. We observed that the transfer learning performed the best. We also note that only car specific features may not be effective for damage classification. It thus underlines the superiority of feature representation learned from the large training set.

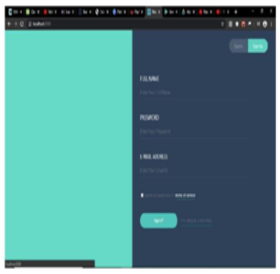
Fig 1 Sign Up Page

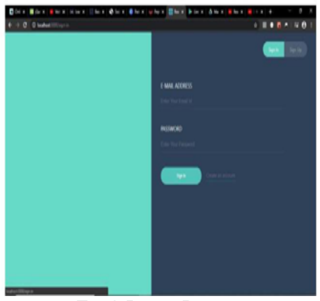
Fig 2 Login Page

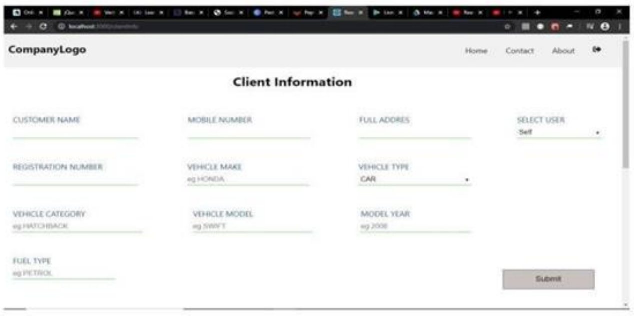

Fig 3 Client Information Page

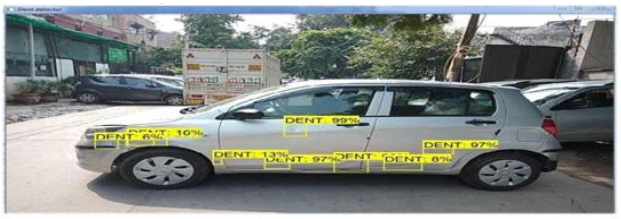

Fig 4 Dent Detection without score

We build an application which revolves around solving the time issues that were occurring when a user was trying to apply for an insurance policy for their vehicle because it was a very cumbersome experience involving many stages. We tried to reduce one stage by automating the dent detection part through which the amount of the policy is decided. Our project can be used by various company agents and companies to fasten the process. The prediction of the amount would be done by analyzing the following parameters:

1. Condition of the car
2. Amount of dents
3. Amount of scratches

REFERENCES

[1] http://www.ey.com/publication/vwluassets/ey-doesyour-firm-need-a-claims-leakage-study/ey-does-yourfirmneed-a-claim -leakage-study.pdf

[2] https://tractable.ai/

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014. [4] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M.Smeulders, Selective search for object recognition," International Journal of Computer Vision, vol. 104, pp. 154{171, Sep 2013.

[4] “http://www.tractable.io/,”.

[5] Bengio Y. Lecun Y., Bottou L. and Haffner P., “Gradient-based learning applied to document recognition,”Proceedings of IEEE, vol. 86, no. 11, 1998.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., pp. 1097–1105. Curran Associates, Inc., 2012.

[7] Michael Giering Mark R. Gurvich Soumalya Sarkar, Kishore K. Reddy, “Deep learning for structural health monitoring: A damage characterization application,” in Annual Conference of the Prognostics and Health Management Society, 2016.

[8] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio,“Why does unsupervised pre-training help deep learning?,” Journal of Machine Learning Research, vol. 11, no.Feb, pp. 625–660, 2010.

[9] Jonathan Masci, Ueli Meier, Dan Cires¸an, and Jurgen ¨ Schmidhuber, “Stacked convolutional auto-encoders for hierarchical feature extraction,” in International Conference on Artificial Neural Networks. Springer, 2011, pp. 52–59.