# ECE 697 Project Mileage Extraction from Odometer Pictures

Elizabeth Murphy

August 7, 2021

***Abstract* - Object Detection is a technology that aims to detect semantic objects in pictures or video frames. There are two main ways to perform object detection, either using neural networks or non-neural ways. In this project, two MobleNet models were used to extract mileage information from an odometer picture. This paper details the entire process of extracting this mileage information. The first step of this process was to complete the annotation of the images. Most of this work was already performed by American Family Insurance but there was approximately 1% of the data that was mislabeled and needed to be relabeled. After this, the images went through pre-processing where both the resolution and the sharpness of the images were increased. Next, the images went through two MobleNet models. The first model was used for screen localization. Using the MobleNetV2 the screen localization has an accuracy of 96%. After the screen was localized the images of the screen were put through another MobelNet model to detect the characters. The last step of the process was post-processing where unnecessary detection boxes were deleted and the final mileage was collected. This system has an accuracy of 37.26% for the MobleNetV2. The accuracy goes up to 44.8% if you count the results that are within 1k of the real mileage. The code for this project can be found at: https://github.com/emurphy7/Odometer_project**

## 1 Introduction

American Family Insurance has been researching a simple and effective way for customers to provide important information when reporting a claim or requesting a quote. One important piece of information that is needed to process these requests is the mileage of the vehicle. To get this information, they have decided to request an image of an odometer from their customers and use computer vision and machine learning methods in order to extract the mileage information. For my project, I focused on re-creating the work in the paper "Mileage Extraction From Odometer Pictures for Automating Auto Insurance Processes." [1] published by American Family Insurance with a different object detection model as well as adding image pre-processing steps.

## 2 Background / Previous Work

American Family compared two different models when creating their mileage extractor. These two models are the Faster-RCNN and the SSD ( single shot multiBox detector ). Below are a summary of the models as well as the results achieved by American Family Insurance.

## 2.1 Faster-RCNN

The Faster-RCNN consists of three different stages.

1. The first stage is the convolutional layer. This layer is composed of convolution layers and pooling layers. The output of this stage is the feature map which is used as the input into the second stage.

2. The second stage is a region proposal network.

   The region proposal network uses anchors - which can be thought of as boxes. For each position in the image, there are 9 anchors.

   The region proposal network predicts the probabilities of each of the different anchors being 'foreground' or 'background' where foreground is the object we are trying to detect and background is everything else. After the region proposal network ROI, pooling is used to reduce the feature map into the same size.

3. The last stage of this network uses fully connected layers. The regressor refines the bounding box and the classifier labels the object.

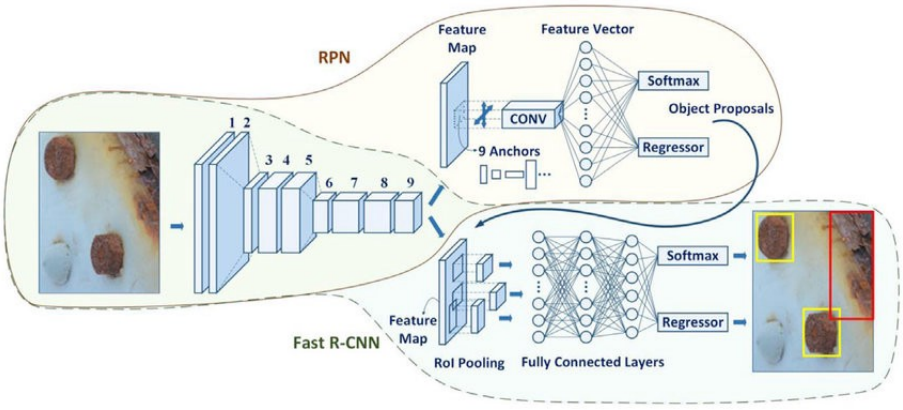



Figure 1: https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4 [5]

## 2.2 Single Shot multiBox Detector

The SSD (single shot multiBox detector) predicts category scores and box offsets for a fixed set of bounding boxes. The architecture of this model consists of a base VGG-16 network without the fully connected layers. Instead of these layers, convolutional layers were added to extract features at multiple scales and decrease the size of the input at each layer.

A MultiBox's loss function is used for this model. This loss function consists of confidence loss ( categorical cross-entropy loss) and a location loss which is the L2 norm of the predicted location with the true location.

The reason this network was picked by American Family is because it is suitable for real-time applications.

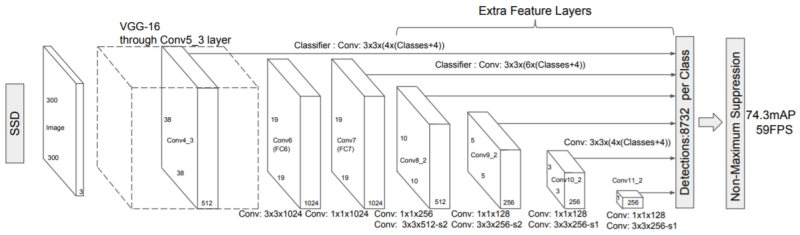



Figure 2: https://towardsdatascience.com/ssd-single-shot-detector-for-object-detection-using-multibox-1818603644ca [6]

### 2.2.1 American Family Results

Below are the Mean Average Precision results achieved from American Family.

| Network | SSD | Faster RCNN |
|---|---|---|
| Odometer Localization | 0.79 | 0.82 |
| Character Recognition | 0.89 | 0.93 |

The final results achieved by American Family using the Faster-RCNN model are **85.4%** for true accuracy and **88.8%** for results within 1000 miles.

# 3 Data

American Family Insurance is provided a filtered data set that contains 6209 images of odometers. These images contain odometers that are both analog and digital. Even after filtering the dataset still has many images of poor quality. The issues with these images include low resolution, poor lighting, and incorrect orientation. An analysis of the quality of the images was performed by American Family and the results are in the table below.

Table 1: Image Quality

| Image Quality | Percentage |
|---|---|
| Extremely Poor | 8.1 |
| Poor | 12.8 |
| Average | 25.3 |
| Good | 33.4 |
| Excellent | 20.1 |

American Family Insurance also provided an excel sheet with annotations of bounding boxs for both the odometer screen and the individual characters on the screen. These annotations were created through crowdsourcing.

### 3.1 Further Annotation Work

Cropped images for the odometer screens were created from the annotation provided by American Family Insurance. I reviewed the cropped results and noticed that approximately 1% of the annotations for the screens were incorrect. Since the number of images provided were limited, I decided that correcting the annotations was necessary.

The images below show common annotation mistakes where the red bounding box represents the incorrect annotation and the green bounding box represents the fixed annotation.

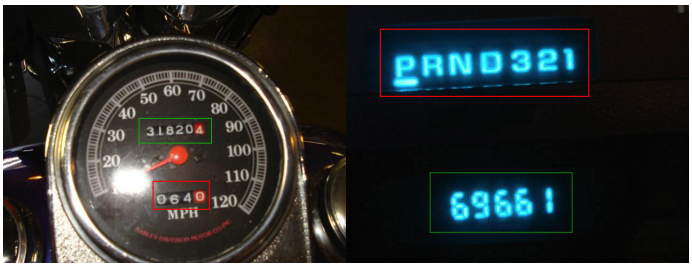



Figure 3:

Figure 4:

## 4 Image Pre-Processing

In general, object detection models perform better on higher-quality images. Since American Family Insurance assessed the quality of the data set and labeled 20.9 % as either poor or extremely poor quality pre-processing was performed as an attempt to negate sub-par performance.

### 4.1 Bicubic Interpolation vs VDSR

Both bicubic interpolation and a VDSR (Very Deep Super Resolution) network were used to increase the resolution of the images. Below is a comparison of an image that was doubled in resolution using both of the methods. As you can see, the difference between the two images is not noticeable to the human eye. Performing bicubic interpolation on all of the images using the CPU of my local desktop took approximately 8 hours while using a VDSR network took 9x as long. Because of the large computational difference, I have decided to use bicubic interpolation on the images.


Figure 5: Bicubic

Figure 6: VDSR

### 4.2 Image Sharpening

Kernel filters are a common technique in image processing for both the sharpening or blurring of images. These filters work by sliding a matrix across the image. Sharpening images can result in encapsulating more details about objects of interest in the image. I decided that this was a good approach for my project since the objects that I am concerned with detecting consist of edges. For this, I used a Gaussian lowpass filter which made the edges in my images appear more defined by darkening the darker pixels and brightening the lighter pixels.

The image sharpening function from the Image Processing Toolbox in Matlab was used for this task. This function has three parameters that can be changed; the radius, the amount, and the threshold. The radius ( which corresponds to the radius of the Gaussian filter) and the amount ( which is the strength of the sharpening effect) were both chosen by visual inspection of a group of 10 images. The threshold was set to the default of zero.

Figure 7: Original Image Figure 8: Sharpened Image

## 5 MobleNet Arcitecture

The model that was chosen for my project is the MobleNet. This MobleNet is a SSD network that was introduced by Google. There are currently three different versions of this model. I decided to compare the first and second version. The MobleNet was developed to have both the small network size and a small complexity cost. This was done so the network can be run on moble devices.

Both of the MobleNets use depthwise separable convolutions to build a light weight neural network. A depthwise separable convolution consists of two different convolution operations. First a depthwise convolution is performed followed by a pointwise convolution ( convolution with a 1x1 kernel) This type of convolution is used because it requires less computational work because there are less weights to learn.

The two different versions of the MobleNet I am comparing both use a RELU6 activation function, a weighted sigmoid classification loss, and a weighted_smooth_11 localization loss. They both also use the rms prop optimizer with a learning rate of 0.004.

Since, training them from scratch with a small dataset can lead to overfitting transfer learning was used. Transfer learning is the processes of further training a pre-trained network with new

data. I decided to use models that were previously trained on the coco dataset for this implementation.

The article "Custom Object Detection Using Tensorflow in Google Colab"[4] by Matus Tanonwong was referenced in order to set up both of the MobleNet models.

## 5.1 MobileNetV1

The MobleNet V1 consists of 30 different layers. The first layer is a convolutional layer (the only one of the network) with a stride of 2. The layers in the middle section of the network go in the pattern of a depthwise layer followed by a pointwise layer. Sometimes the depthwise layer will have a stride of two in order to reduce the size of the data as it goes through the network. Each convolutional layer is followed by a batch normalization and a RELU6 activation function. The second to last layer is an average-pooling layer to create a vector. The final layer of this network is a fully-connected layer that uses a softmax function.

## 5.2 MobileNetV2

The MobleNet V2 consists of 17 "building blocks" in a row. Each "building block" has three different layers. The first layer consists of a 1x1 convolution. The purpose of this block is to expand the number of channels in the data. After this convolution there is batch normalization followed by a RELU6 activation function. The second layer consists of a 3x3 depthwise convolution followed by batch normalization and a RELU6 activation function. The last layer is called the projection layer. It is named this because it uses a 1x1 convolution to project the data into a lower dimension. This last layer is linear since it was found that no-linearity destroyed useful information. Again the last two steps consist of an average-pooling layer followed by a fully-connected layer.

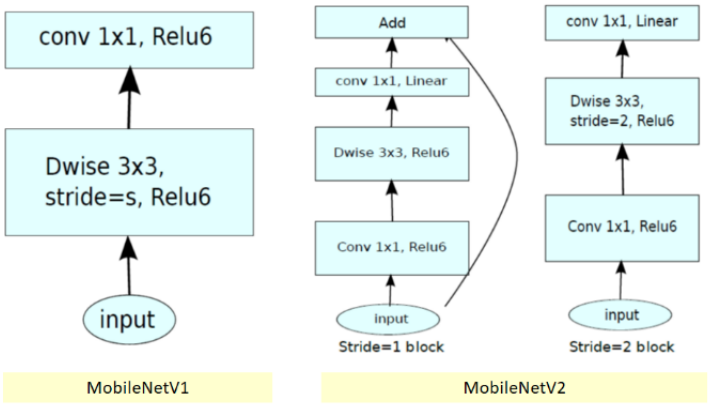



Figure 9: https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c [3]

# 6 Odometer Screen Localization

Odometer screen localization is the process of isolating the odometer display from the rest of the image. Both the MobileNet V1 and MobileNet V2 object detection models are used in order to identify the odometer display in the image. A pre-trained model for both networks were used and then further trained using the odometer images. The pre-processed images were not used in this step since the increase in resolution caused google colab to crash due to memory constraints. The data is spit 75% / 25%. Two csvs are created that contain the filename, width, height, xmin, ymin, xmax, ymax values of odometer screen. The csvs and the images are used to create two TFRecords. (one for training and one for testing) These TFRecords and a label file are the inputs of the networks. 140k addition steps were added to the training processes to adapt the model to recognize odometer screens.

The most four most likely bounding boxes were saved into a csv file but only the most likely bounding box is used for the character recognition stage.

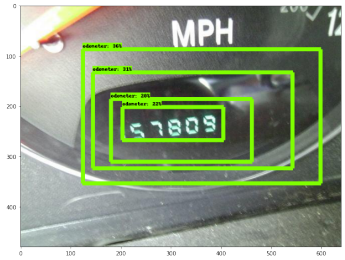



Figure 10: Odometer Localization Results 1

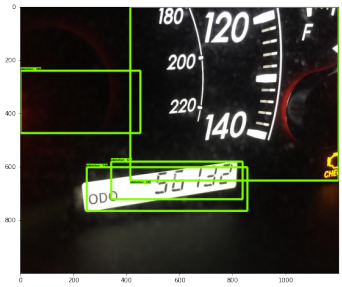



Figure 11: Odometer Localization Results 2

In order for the odometer screen to be localized correctly the mileage must be seen in the bounding box and the bounding box must not be of large size. The results for the most likely bounding box of this localization are shown below.

| Network | MobleNet V1 | MobileNet V1 w/ annot fix | MobileNet V2 w/ annot fix |
|---|---|---|---|
| Validation Accuracy | 92.07% | 94.07% | 96.00 % |

# 7 Character Recognition

The Character recognition portion of this project also uses the MobleNet as its object detection model. The input of this model in the cropped odometer screens ( Whose coordinates were provided by American Family) and the coordinates of the bounding boxes of the characters on the screen. The character bounding boxes were split 85% / 15%. The images used for this section are the ones with increased sharpness and resolution. Just like the odometer localizization step, two csvs are created that contain the filename, width, height, xmin, ymin, xmax, ymax values of the character locations. Then the csvs and the images are used to create two TFRecords. A pre-trained network for both of the models were trained an additional 180k steps. The test set of images were cropped in accordance to the output of the localization MobleNet V2 model's output csv file. These cropped

images were then input into the network. The output of both of these models are the coordinates of the individual characters bounding boxes as well as their classification and classification score.

# 8 Post-processing

The Post-processing steps will be based off of the steps outlined in "Mileage extraction from odometer pictures for automating auto insurance processes" [1] Other post-processing steps where added to aid in final classification

Post-processing algorithm:

1. Filter bounding boxes that belong to non-digit character.
2. Filter bounding boxes by the classification score (10% was used)
3. Remove boxes that were contained in other bounding boxes
4. Calculate the median of all of the remaining bounding boxes. If any box lies far outside of the median then the box is to be removed.
5. Remove all boxes that are close in xmin, xmax, ymin, and ymax values
6. Augment/extrapolate boxes in horizontal direction by a factor of the width along both sides.

   - W idth was used for digits classified as one (This was done because the bounding boxes for 'ones' are normally smaller in width than other digits)

   - W idth / 2.5 was used for all other digits.
7. Create a graph G where vertices represents bounding box and edges represent overlap between two boxes.
8. Select subgraph with largest number of vertices.
9. Sort boxes in that subgraph horizontally and extract digits in order to form a mileage number.
10. If the mileage has 7 digits or is a value greater than 300k then the last digit is discarded

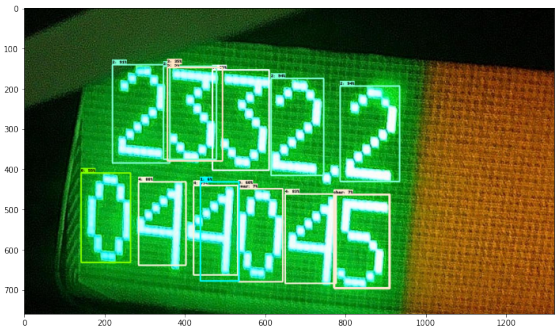


Figure 12: Before Post-Processing

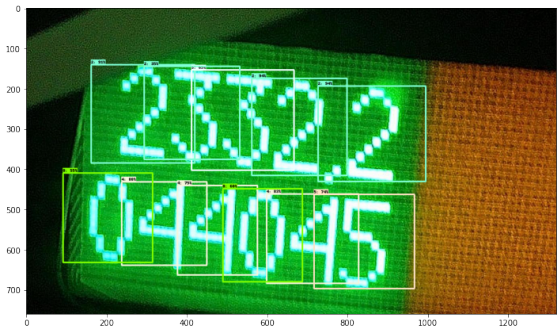


Figure 13: After Post-Processing

# 9 Results

Since the use case for this implemtation is insurace quotes or claims processing a margin of 1000 miles was deemed as acceptable. The accuracy for this and the true mileage are shown below.

| Network | MobleNet V1 | MobleNet V2 |
|---|---|---|
| Accuracy | 17.02% | 37.26% |
| Results within 1k mileage | 25.78% | 44.80% |

There were three main failure types:

1. Incorrect digit classification.
2. Incorrect odometer screen localization ($\approx 4\%$)
3. Digits not being classified or under the threshold

The results are much worse than the results achieved by American Family who had a **85.4%** for true accuracy and **88.8%** for results within 1000 miles.

## 9.1 Implementation Challenges

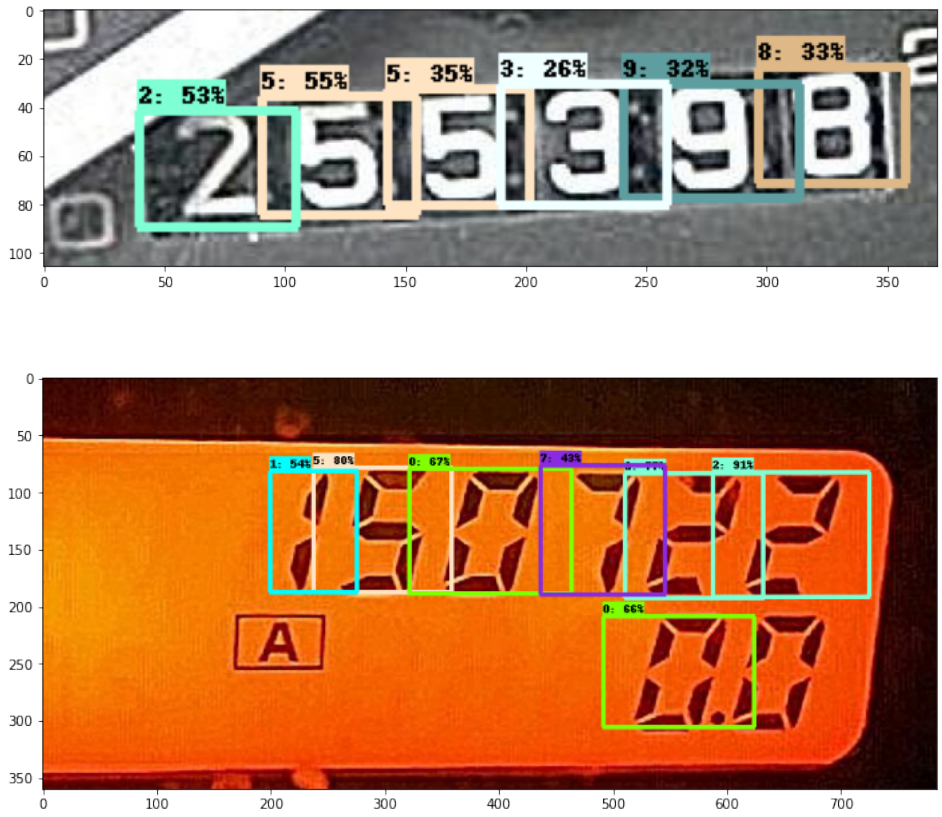
The main implementation challenge was access to computing resources. The Pro subscription of Google Colab limits the run time of notebooks to 24 hours and also has restrictions on how long a GPU can be used for. This caused many issues when training the network. To overcome this I had to sporadically save checkpoints from the training network to my Google Drive in order to not loose training progress. Memory issues were also encountered when trying to train the odometer localization network with the pre-processed images.

## 9.2 Future Work

There are many things that can be done to improve the results for this project. These include:

1. Change the configuration of the model.
2. Train the model from scratch instead of using a pre-trained model
3. Change the model to increase accuracy (MobleNet 3)
4. Increase the amount of training data
5. Improve post-processing

## 9.3 Correctly Labeled Images




## 9.4 Incorrectly Labeled Images


Figure 14: Error Caused by Screen Localization

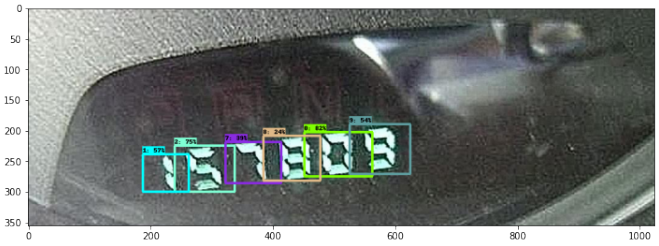


Figure 15: Error Caused by Mislabeled Number

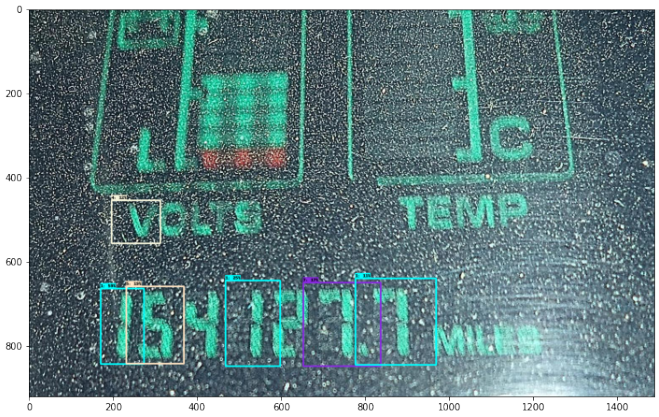



Figure 16: Error Due to Network not Identifying Numbers

# 10 Sources

[1] Acharya, Shailesh and Glenn M. Fung. "Mileage Extraction From Odometer Pictures for Automating Auto Insurance Processes." Frontiers in Applied Mathematics and Statistics 5 (2019): n. pag
[2] https://github.com/emurphy7/Odometer_project
[3]https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c
[4] https://medium.com/@matus.tanon/custom-object-detection-using-tensorflow-in-google-colab-e4d6e1a17f18
[5] https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4
[6] https://towardsdatascience.com/ssd-single-shot-detector-for-object-detection-using-multibox-1818603644ca