VEHICLE INTERIOR DETECTION WITH DEEP LEARNING & COMPUTER VISION

Stephanie Rogers
Data Science & Business Analytics
Wayne State University
gp5880@wayne.edu

Ranjitha Vidyashankar
Data Science & Business Analytics
Wayne State University
gm8135@wayne.edu

Dante Burch
Data Science & Business Analytics
Wayne State University
gl0706@wayne.edu

Jatin Gongiwala
Data Science & Business Analytics
Wayne State University
go7984@wayne.edu

Table of Contents

Acknowledgments

Thank you to our faculty advisors and company sponsor who championed this work. Our experience this summer was extremely challenging and without regular touchpoints with our professors, Dr. Ratna Babu Chinnam and Dr. Ming Dong, as well as constant course related updates from our program manager, Nikki Taylor-Vargo, and the goals of the project presented to us by our sponsor, Krishna Murthy, we would not have been able to prepare this report.

Executive Summary

In the summer of 2019, a team of graduate students from the MS Data Science Program at Wayne State University undertook the task of using state of the art deep learning techniques to identify the make of a vehicle by using only images of its interior features and to detect key components of a vehicle's interior. Identifying the make of a vehicle or, what is generally known as image classification, relies on a deep learning algorithm known as a Convolutional Neural Network (CNN).

CNNs are considered state of the art for classification tasks because they differentiate images by assigning weighting or bias to certain elements within an image. CNNs are generally developed at a fixed resource budget, then scaled up for better accuracy (resource permitting) via CNN architectures.

The CNN architectures used for image classification are EfficientNet for binary classification and MobileNet for multi-class classification. Detecting key components of a vehicle falls under what is known as Object Detection within deep learning and it also relies on CNN. The CNN architecture used for object detection throughout are project is SSD Inception. TensorFlow, a python-based library that allows us to deploy these models in a python environment, was used for both classification and object detection throughout this project.

Deliverables included in this report

- Github Project Page
- 3 Commented Notebooks for Classification & Detection
- Instructions on TensorFlow JS - a way to deploy models in browser

Introduction

Due to numerous technological improvements in recent years, deep learning is rapidly advancing and impacting industries all around the world, including automotive manufacturing (autonomous vehicles — example image below), security, and surveillance. The value of artificial intelligence in automotive manufacturing and cloud services will exceed $10.73 billion by 2024 [1]. In addition to improvements in the deep learning technological architecture, consumer demand for personalized content has reached an all-time high [2].

Our team evaluated and applied state of the art deep learning technologies to a variety of vehicle interior images to investigate what can be seen and understood. Not knowing what the future designs will look like, we sought to identify what could be seen with current technologies and determine what the best course of action is moving forward to maximize the business opportunity.

Objectives & Deliverables

The goal of this project was to develop and evaluate machine learning models that can help identify the content of a vehicle and its features using visual cues from its interior.
The primary goal of this project was achieved by fulfilling the Practicum Learning Objectives and developing the following models:

1. Image-Based Classification
2. Object Localization & Detection

Assumptions A few key assumptions were made at the onset of this project:

- This project would be exploratory in nature as we did not have a specific use case.
- We would only use state of the art techniques to ensure our deliverables could be leveraged for future applications.
- Our classification tasks would be supervised, requiring pre-labeled images to learn the features from curves, edges, and combination of features.
- Our object detection model would prioritize speed over accuracy as we wanted to ensure that the end-user experience was applicable to modern-day applications.

Scope Exterior images were out of scope for this project. 500 vehicle interior images per class were selected for classification training purposes while 100 per vehicle class were selected for object detection training. Images were restricted to only those that contain roughly 50% of a dashboard. Furthermore, transfer learning provided us an opportunity to detect object outside of scope such as people, cell phones, and traffic lights. Models varied from capturing patterns by region of interest and pixels to only focusing on points of interest indicated in shallower methods.

Literature Review

Vehicle Interior AI Use Case While images of vehicle interiors have yet to be used for vehicle classification or object detection tasks, interior images of driving vehicles have been used to create models that can determine whether or not a driver is distracted. Similar models have already had a large impact on the automotive industry — for example, Subaru's DriverFocus feature available in the 2020 Subaru Forester monitors a driver's eyes and head position to determine if he or she is drowsy [3]. Furthermore, models have been created specifically to determine cell phone usage and to compete in State Farm's Distracted Driver competition that was launched a few years ago [4].

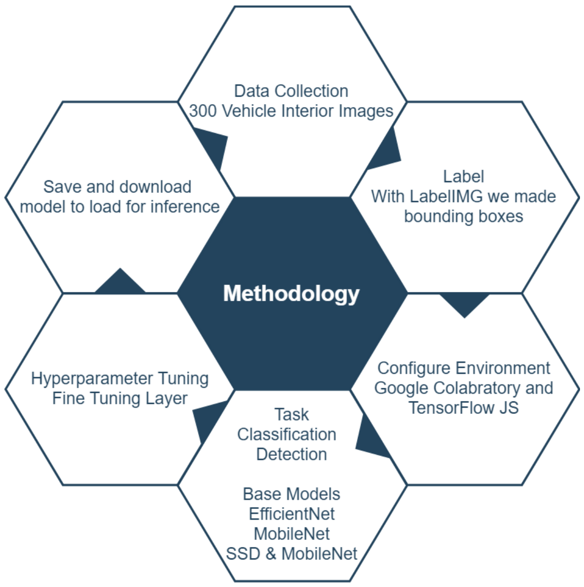
According to the World Health Organization (WHO), 1.25 million people worldwide die every year as a result of traffic accidents and according to the National Highway Traffic Safety Administration (NHTSA), one-fifth of accidents in the US are the result of distracted drivers [5]. As one can imagine, the primary benefit of detecting whether a driver is distracted is potentially discovering ways we can reduce collisions and deaths from car accidents.

Deep Learning Trends Vehicle classification & detection using computer vision has been used extensively for different applications. Previous car model research has been based on a relatively small number of car models and only using exterior vehicle images. There has been no previous attempt to accomplish the car model classification task by using only interior images. Raul Humberto and Marco Aurelio's 2014 paper "Computer Vision Based Real-Time Tracking & Classification System", talks about the vision-based system to classify, detect & track the moving vehicles [6]. We studied their image classification technique and implemented in our car dashboard classification. A separate paper titled, "Real-Time Detection and Classification of Cars in Video Sequences", uses Convolutional Neural Network to classify and detect cars [7]. This proved to be extremely valuable in providing us with the background knowledge needed to work on a project of this scope.

As the powerful algorithms emerged a new generation of explainability techniques appeared, suitable to any machine learning model — not only tree-based. The first explainability method of this type was published in the paper: "Why Should I Trust You?: Explaining the Predictions of Any Classifier". It explains the predictions of any classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction [8]. A method based on this framework is available out of the box in the LIME package (Local Interpretable Model-Agnostic Explanations) [9].

Project Methodology

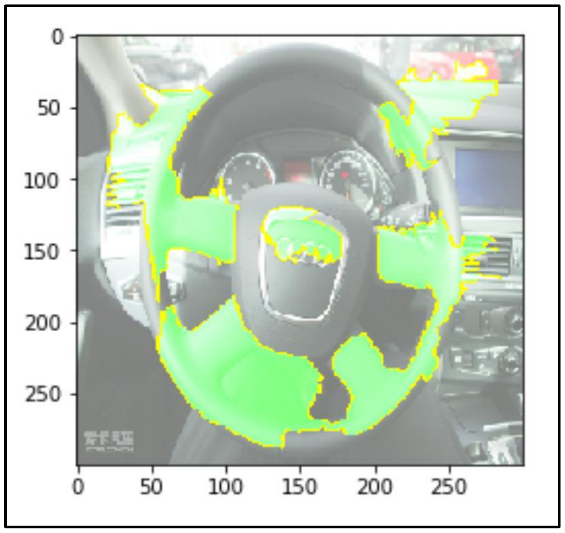
This project followed the standard computer vision project life cycle. Best practices were implemented and set as the default when experimenting with different configurations.



With regards to our model, we wanted to create an all-encompassing model framework that could receive images of the car's dashboard and output what was in the picture. To do this, we needed two structures: an object detector and an object classifier. Getting these two structures to work together took up the majority of our project time. Ultimately, we landed on using VGG 16 & Efficient Net for the object detector since it uses the weights and feature maps of our car dashboard specific CNN classifier. This helps the detector better find objects related to a car's dashboard in the image and deal with identifying them. Due to AI's growing influence on more aspects of life, recent years have shown a growing need for "Explainable AI" which essentially is used to allow us to trust our results. We'll go into further detail about this concept in the next section.

Vehicle Interior Example

Explainable AI tools and techniques are used to get a better picture of where the model is looking and what features it is using to determine class labels. As stated in the prior section, these techniques can visually validate our model by showing us what does and doesn't matter in either a classification or object detection task. Explainable AI is becoming more popular and tools are being developed rapidly from the need for better interpretations of neural networks. One such tool is called Local Interpretable Model-Agnostic Explanations or what is commonly referred to as LIME. LIME is an algorithm that can explain the predictions of any classifier in a faithful way, by approximating it locally with an interpretable model. Below is the result of a standard InceptionV3 pre-trained model being used on one of our Audi interior images.

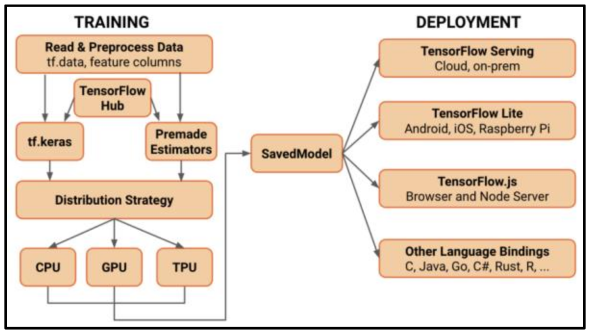


Explainable AI Output Example

The results of the image above reveal that the shape of the steering wheel and more importantly, the 4-circled logo in the middle of the steering wheel, reveals that this image belongs to an Audi. Object detection is also a form of explainable AI as it requires anchoring to the image and finding the location to produce bounding boxes. These bounding boxes visualize the model outputs in a way that allows better interpretation of what the model knows.

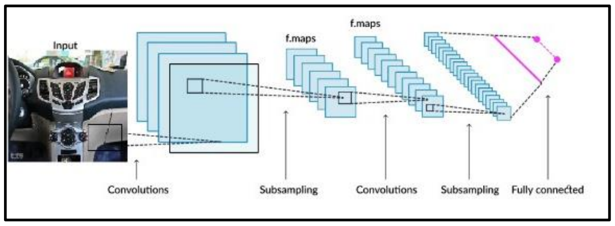
Models

This project focused on two main computer vision tasks -- image classification and object detection. Below, the TensorFlow Model pipeline visualizes the process.



TensorFlow Model Pipeline

These tasks have slightly different configurations and dependencies. Our research indicates configuring projects to maximize future use would mean to prepare projects for mask and segmentation tasks. Segmentation or masking tasks have been briefly mentioned but will be elaborated more in future work.



Fully Connected CNN

Transfer learning is applying the knowledge from one domain to another in the form of weights [10]. Transfer learning is the start of any modern computer vision project thanks to the numerous benefits and cost savings. Weights can be brought in from a pre-trained model with retraining of the last layer. Another option is to retrain the entire architecture, losing all previous knowledge and requiring many more images per class. and requires many more images. A better option, with a limited number of vehicle interior images, is to fine-tune different layers and focus more on hyperparameters.

Image Classification

For the computer to be able to see an image of a Ford and know it is a Ford, many images from all classes need to be used to train the model. The model learns best with more images that show the class differences, but the differences even between models within the same make, make this difficult.

Image classification groups an image into a predefined category such as the make of a vehicle or a part type. How does the computer know if the steering wheel or odometer is from a specific make? To ensure learning was not from something obvious like a logo, some images had logos masked out and explainable AI was used where fit.

Binary Image Classification determines if an image is Ford or Not Ford. This model is trained on a dataset that is half Ford Images and half images of other make interiors. While basic, implementing a binary classification is useful for many reasons. A binary classification model of Ford vs. Not Ford allows experimentation of different architectures quickly. A potential use case for a binary image classification model could be used in manufacturing or maintenance activities as an application to match parts or quality by rendering MATCH or NO MATCH, or for the purposes of this project: Ford or Not Ford. During the practicum, we implemented a cutting edge Efficient Net architecture that was designed by Google and released in May 2019 [11].

Multi-Class Classification Similar to the binary classification, the multi-class image classification model's output can include the top k results that will output all of the classes detected in the image. Using Ford, Audi, and BMW dashboards as the three classes, the resulting model for inference is able to correctly identify the vehicle interiors when tested on a new image from one of those classes. The primary advantage of using a multi-class classification model over a binary one is that it can be scaled up to include as many vehicles makes as one wishes to include. A potential use case for multi-class classification can be the identification of small design differences between classes that can be used to assist design teams. When trained and tested on part images instead of full dashboard images, the performance of the model declined despite having more training part images.

Mobile With the clients expressed interest deploying to mobile in the future, the MobileNet V2, 2018 architecture was chosen for the efficient on-device image classification. Mobilenets come in various sizes controlled by a multiplier for the depth (number of features) in the convolutional layers. They can also be trained for various sizes of input images to control inference speed. The TF-Hub module uses the TF-Slim implementation of mobilenet_v2 with a depth multiplier of 1.0 and an input size of 224x224 pixels. [12] From a total of 2,260,546 parameters, we retrain 2,562 with our class labels while freezing the rest of the parameters.

Object Detection

This project used Deep Learning Object Detection for object detection per the project requirements. Other techniques include Feature-Based, Viola-Jones and SVM Classification with HOG Features [13].

Deep Learning Object Detection localizes an object in an image and displays its class label and a bounding box. The speed of inference in any application will vary based on architecture design choices. Faster R-CNN replaced selective search from Fast R-CNN with a very small convolutional network called Region Proposal Network to generate regions of interests [14]. To avoid delay or latency, experiments were completed on both Faster R-CNN and an SSD, (Single Shot Detector) [15]. SSD only takes a single shot to detect multiple objects within the image in contrast to the regional proposal network (RPN). Faster R-CNN needs two shots to generate region proposals and detect the object of each proposal. This is worth doing for real-time object detection (i.e. moving objects) but not if the image and features will stay the same as in-vehicle interiors, for the most part, therefore, SSD is much faster compared with two-shot RPN-based approaches. There is an accuracy improvement advantage to using Faster R-CNN, especially when using Faster R-CNN in conjunction with an extension known as Mask R-CNN that we'll briefly touch on later.

Results

The performance of a model for an image classification task is evaluated using accuracy, precision, recall [16].

TP = True positive
TN = True negative
FP = False positive
FN = False negative

$$Precision = \frac{TP}{TP + FP}$$

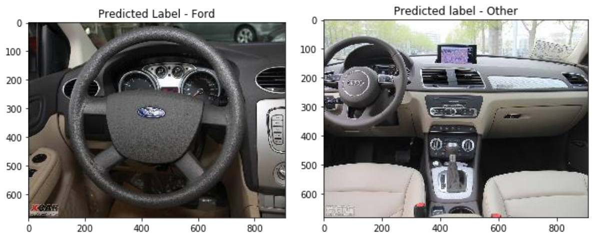
$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

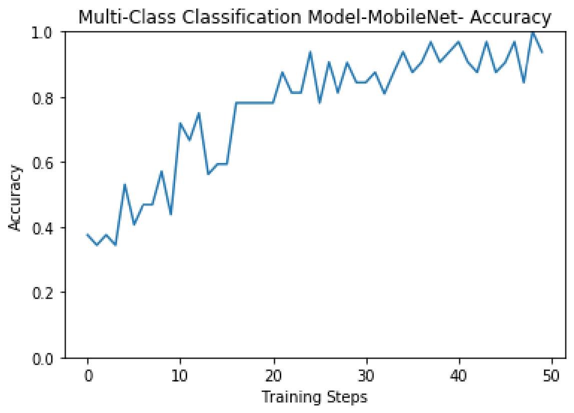
$$AP = \frac{1}{11} \sum_{r \in \{0.0, \ldots, 1.0\}} AP_r = \frac{1}{11} \sum_{r \in \{0.0, \ldots, 1.0\}} p_{interp}(r)$$

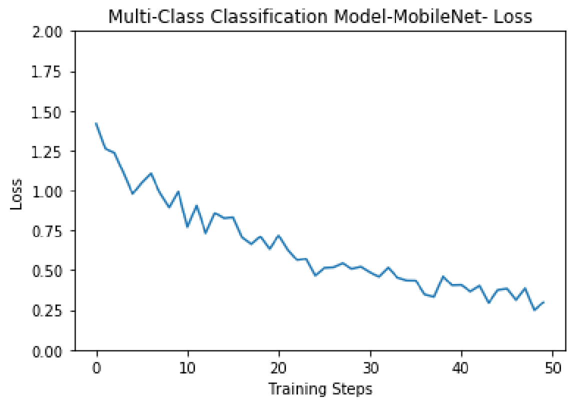
$$p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})$$

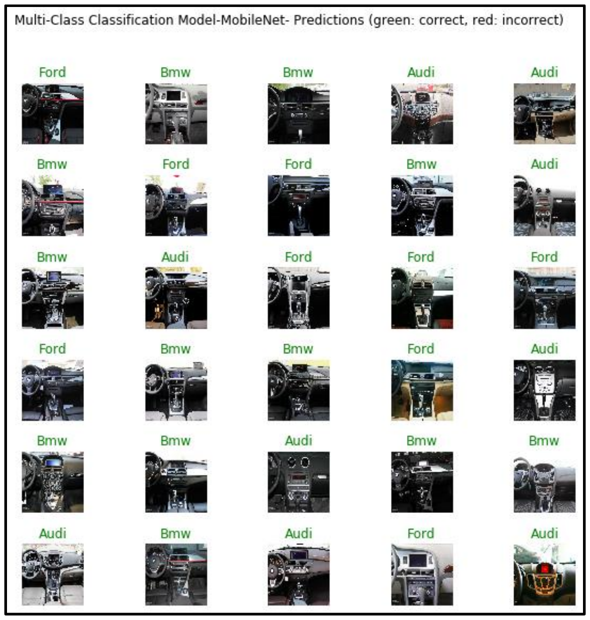
Binary Classification with EfficientNet achieved 98% accuracy when trained on 500 images in each class.



Multi-Class Classification with MobileNet achieved 93% accuracy when trained on 75 images in each class.





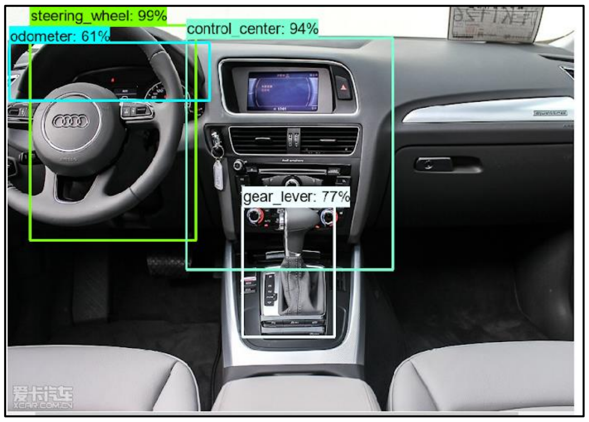


Object Detection with SSD outperformed other architectures with limited training data. This rendered significantly faster than the Faster RCNN even when tuning the number of boxes. SSD makes predictions based on feature maps taken at different stages whereas Faster RCNN belongs to the Regional Proposal Network.

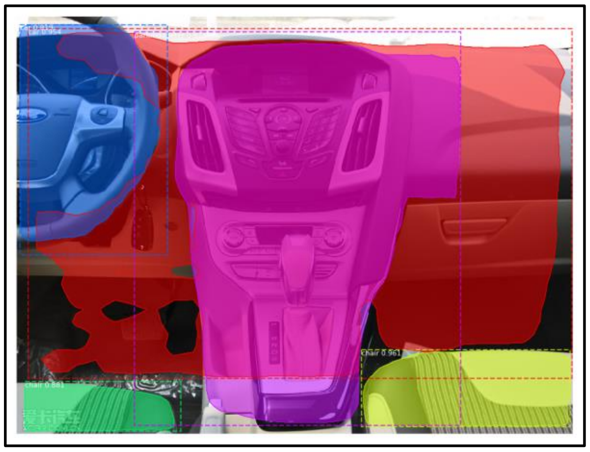
IOU A perfect bounding box prediction will have an IoU of 1. Predicting bounding boxes and determining if the prediction is good or bad based on how well the predicted and actual bounding boxes overlap. This can be calculated by dividing the area of overlap by the total area of both bounding boxes, or the intersection divided by the union, referred to as “intersection over union,” or IoU.

mAP the performance of a model for an object recognition task is evaluated using the mean absolute precision, or mAP.

Recall refers to the percentage of the correctly predicted bounding boxes (IoU > 0.5) out of all bounding boxes predicted. The recall is the percentage of the correctly predicted bounding boxes (IoU > 0.5) out of all objects in the photo. As we make more predictions, the recall percentage will increase, but precision will drop or become erratic as we start making false-positive predictions. The recall (x) can be plotted against the precision (y) for each number of predictions to create a curve or line. We can maximize the value of each point on this line and calculate the average value of the precision or AP for each value of recall. The average or mean of the average precision (AP) across all of the images in a dataset is called the mean average precision, or mAP



Mask R-CNN while Faster R-CNN predicts the bounding boxes, Mask R-CNN essentially adds an additional branch to the model that better detect objects [17]. While the image above returned bounding boxes for objects detected, the image below instead returns a more accurate visual representation of the objects being identified.



Conclusion

Since there was no use case, our conclusions are more or less observations from the project. they are as follows:

1. Faster R-CNN models are better suited for cases where high accuracy is desired and latency is of lower priority. Conversely, if the processing time is the most important factor, SSD models are recommended.
2. Creating a scalable model is extremely important due to the number of technological breakthroughs that have taken place during the summer of 2019 alone. Fortunately, modern deep learning techniques provide a platform that makes this easily attainable.
3. Leverage existing resources as much as possible. Our initial plan to limit the number of possible classes in our object detection model proved to not make much sense in the end since our model architecture could easily identify objects beyond the scope of the project.

References

[1] 5 Ways Artificial Intelligence is Impacting the Automotive Industry. (2019, February 08). Retrieved from https://igniteoutsourcing.com/automotive/artificial-intelligence-in-automotive-industry/

[2] Consumer Demand for Personalized Content Reaches All-Time High. (n.d.). Retrieved from https://www.cmo.com/features/articles/2018/1/31/adobe-2018-consumer-content-survey.html#gs.x7945l

[3] Phelan, M. (2019, August 03). 2020 Subaru models will greet you, help you keep your eyes on the road. Retrieved from https://www.freep.com/story/money/cars/mark-phelan/2019/08/03/subaru-driverfocus-outback-forester-legacy/1903279001/

[4] @viritaromero. (2019, March 06). AI can detect distracted drivers on the road. Retrieved from https://medium.com/datadriveninvestor/ai-can-detect-distracted-drivers-on-the-road-6cebeecb0ead

[5] Eraqi, H., Abouelnaga, Y., Saad, M., Moustafa, M. (2019). *Driver Distraction Identification with an Ensemble of Convolutional Neural Networks* [PDF file]. Retrieved from https://arxiv.org/pdf/1901.09097.pdf

[6] Huberto, R. & Aurelio, M. 2014. *Computer Vision Based Real Time Vehicle Tracking and Classification System* [PDF File]. Retrieved from https://ieeexplore.ieee.org/document/6908506/citations#citations

[7] Gepperth, A., Edelbrunner, J., Bucher, T. 2005. *Real-Time Detection and Classification of Cars in Video Sequences* [PDF File]. Retrieved from https://ieeexplore.ieee.org/abstract/document/1505173

[8] Ribeiro, M., Singh, S., Guestrin, C. 2016. *"Why Should I Trust You?" Explaining the Predictions of Any Classifier* [PDF File]. Retrieved from https://arxiv.org/pdf/1602.04938.pdf

[9] Marcotcr. (2019, July 23). Marcotcr/lime. Retrieved from https://github.com/marcotcr/lime

[10] @pranoyradhakrishnan. (2017, November 23). What is Transfer Learning? Retrieved from https://towardsdatascience.com/what-is-transfer-learning-8b1a0fa42b4

[11] Tan, M. & Le, Q. (2019, June 10). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.* Retrieved from https://arxiv.org/pdf/1905.11946.pdf

[12] Google Inc. (2018, January 16). *Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection, and Segmentation.* Retrieved from https://pdfs.semanticscholar.org/462e/4d0b35bf571bfc35dcd8e9bd589dca07a464.pdf?_ga=2.132936138.533789064.1566158891-760149531.1566158891

[13] Zhao, Z., Zheng, P., Xu, S. and Wu, X. (2019, April 16). *Object Detection with Deep Learning: A Review.* Retrieved from https://arxiv.org/pdf/1807.05511.pdf

[14] Ren, S., He, K., Girshick, R., & Sun, J. (2016, January 6). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [PDF File]. Retrieved from https://arxiv.org/pdf/1506.01497.pdf

[15] @franky07724_57962. (2019, February 23). Exploring OpenCV's Deep Learning Object Detection Library. Retrieved from https://medium.com/@franky07724_57962/exploring-opencvs-deep-learning-object-detection-library-e51fe7c82246

[16] @koolanalytics. (2018, June 08). Accuracy, Precision, Recall or F1? Retrieved from https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

[17] @gabogarza. (2019, January 08). Mask R-CNN for Ship Detection & Segmentation. Retrieved from https://towardsdatascience.com/mask-r-cnn-for-ship-detection-segmentation-a1108b5a083

Appendix

Technology

Github Page

https://stephanierogers-ml.github.io/practicum/

The project page is hosted as a Github page to keep the materials organized. To run the demo locally, run the notebook file and save the files locally and run tf.js in the browser.

We used a CNN to recognize the steering wheel, odometer, gear lever, and control center by fine-tuning. We trained the model on GPU for free on Google Colab using Keras then ran it on the browser directly using TensorFlow.js(tfjs).

This notebook uses Tensorflow 1 with the Object Detection API. This critical package has not been ported for use in Tensorflow 2 but should be this year.

EfficientNet Binary Classification with Lime

The Ford vs. Not Ford using the efficientNet architecture and using LIME for explainable AI demonstrates the performance gains of an optimized network, even on a small data set.

MobileNet Multi Classification with Hub

The classification notebook file is executed with Tensorflow 2 and Tensorflow Hub using a MobileNet architecture.

SSD Object Detection with API ‘ssd_mobilenet_v2_coco_2018_03_29

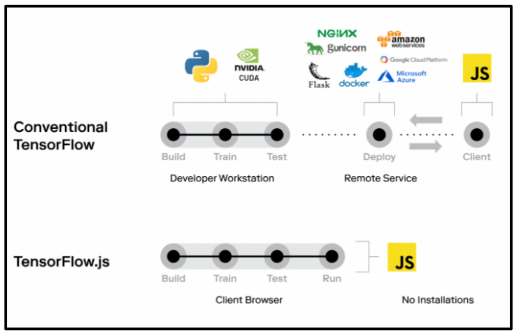
The object detection notebook is executed with the Object Detection API in TensorFlow 1. Tensorflow Hub models cannot be fine-tuned for object detection, so they have to be used for inference only and for retraining the entire model.

Virtual Machine The use of Wayne State University’s Virtual Machine gave access to 40GB VM Ram and 16 vCPUs and GPUs.

IDE Spyder and Jupyter Notebook were chosen as the integrated development environment (IDE). Notebooks are preferred ways to share and document code for a wide audience. The downside of Jupyter Notebook is that the order of execution is critical.

GOOGLE COLABORATORY These three notebook files and can be run with Google Colaboratory, making experimenting and sharing code easier. Users are given access to 12 GPUs daily. This was also ideal to take advantage of running Tensorflow 1.x and 2.x in the project without environment dependency issues. With the speed of development, Colaboratory is popular in the industry. Its format is also compatible with a simple Jupyter Notebook, the format needed to use the school’s resources

TENSORFLOW Currently, the most famous deep learning library in the world is Google's TensorFlow.


Conventional TensorFlow
Build
Train
Test
Deploy
Client
Developer Workstation
Remote Service
TensorFlow.js
Build
Train
Test
Run
Client Browser
No Installations