**Machine learning auto analysis of optical coherence tomography images**

|  |  |  |
| --- | --- | --- |
| Start/end dates | Project start = March 2020 | Project end (estimate) = May 2020 |
| Background and description of project objective | The Level Crossing Removal Project (LXRP) is a program by the Victorian Government, to remove 75 level crossings in Melbourne. As there are disruptions of trains during the cross removal work, buses will replace trains on these disruptions sites. The buses are ordered via 3rd party operators. There is a belief that the number of buses that arrive on-site is less than the actual number contracted.  The project's objective is to automate the process of collecting the replacement bus's identification details (i.e. Bus Run, Plate No) from image captures via mobile devices, and this will speed up the process of data collection by onsite staff and support further analysis and reconciliation. | |
| Technical uncertainties & unknowns prior to development | Before this implementation, an out-of-the-box Google vision API was used to attempt to extract the data. The model was able to detect the license plate number but unable to identify the bus id content.  We also used pytesseract on the full image. It was able to extract some of the text visible in the bus image including the bus id after parameter fine-tuning and adjusting multiple properties. However, it also captured other unrelated data from other sections found in the image that is outside the area of interest without clear segregation to differentiate the text context. Also, the ability to fetch the text was inconsistent with different angles, lighting and others.  Part of the uncertainties as well as the number of buses expected in the image capture. Will there be one or multiple buses to be identified in one image? The latter will require another layer of correlation to determine the busid & plate number that relates to each bus separately. 🡺 Our assumption at this stage there will only be one bus per image.  The distribution of the training data is also unknown at the time of development. The buses, for example, are from different contractors, so each contractor can likely have different information/stickers put in the front window and it is unknown if they would have the same/similar shape of the bus run label. Such elements may impact the accuracy of the model is not trained with sample images of such cases to differentiate. | |
| Description of experimentation & prototyping  Include results, data, metrics & specific variables, parameters  [3000 char] | There are two steps & models involved: 1) Object Detection, 2) Text Recognition   1. **Object Detection:**   With this experiment, used Tensorflow Object Detection API to do transfer learning on an SSD model pre-trained with COCO dataset [Model name: " ssd\_mobilenet\_v2\_quantized\_300x300\_coco\_2019\_01\_03"1, 2, 3, can be downloaded from https://github.com/tensorflow/models/blob/master/research/objectdetection/g3doc/detectionmodel\_zoo.md#coco-trained-models].  *Notes:*  1 A quantized model was selected to be able to convert it to tflite and run on a mobile device.  2 SSD MobileNet model was selected instead of Yolo, Faster R-CNN for its relatively high accuracy with lower complexity which is suitable for real-time processing.  3 Following Models were also tried. However, the MobileNetV2 gave the best relative to performance considering the number of steps used in training:   * ssd\_mobilenet\_v3\_large\_320x320\_coco 🡺 Needed 15K training steps to get results equivalent to ssdmobilenetv2 with 5K training steps * ssdlite\_mobilenet\_v3\_small\_320x320\_coco ( Didn't give good results * ssdlite\_mobilenet\_v2\_coco\_2018\_05\_09 🡺 The False Positives for plate numbers were almost there in every image   For the training following steps was done (This was done outside HighLighter):   1. Took around 260 photos of hatchback cars with Bus Run labels printed on the back window (shot at different times with different lighting). 2. Run a script to scale images dimensions to 800x600 and divide images to 80% train, 10% test, 10% validation 3. Use labelImg for annotating "plate no" and "bus run" objects on each of the train & test images. The generated annotations are in PASCAL VOC 2007 format 4. Using Object Detection API scripts to convert the annotations to tfrecord files 5. Update the corresponding training hyperparameters in the pipeline config file that is available with the base model "ssd\_mobilenet\_v2\_quantized\_300x300\_coco\_2019\_01\_03.config" with the following: -    1. batch-size: 12    2. steps: 10,000 (Initially it was done with 5K steps)    3. Data augmentation options (With initial experiments, the first two options were used, but the model was unable to detect successfully using the original image without rescaling, adding other augmentation options improved the model performance):       1. random crop       2. random horizontal flip       3. random rotations       4. colour distort       5. brightness adjustment 6. Run Training and exporting the trained Inference Graph 7. Convert the model to tflite (Although this is not used at the moment as the tests are not run on mobile)   The outcome of this model provides only classes, bounding boxes, and confidence score. We set confidence minimum threshold to 0.5. The bounding boxes are used to crop the images to be fed in the next step for Text recognition.   1. **Text Recognition:**   We initially attempted to do the text recognition using pytesseract. However, it didn't lead to acceptable results. Hence for this experiment proceeded with out of the box Google Vision 'TEXT\_DETECTION' API.  The output of the Google Vision API is processed with regex depending on the class type (i.e. Plate No or Bus ID). This adds as another layer to filter only valid sequences per expected object type.  The script execution was done using Google colab  **Test Results:**   * Used 26 test images that includes different combinations from 10 Unique Plate Numbers and 10 Unique Bus IDs: * Object Detection: * 24/26 Correctly identify bounding box for BusID (2 undetected) * 25/26 Correctly identify bounding box for PlateNo (1 partially identified) * Text Detection: * 24/26 Correctly recognized BusID (2 are due to the undetected bounding box) * 19/26 Correctly recognized PlateNo (1 due to partial box, others are from 2 unique misrecognized plate numbers out of 9 unique plate numbers used) | |
| Conclusions from prototyping  How do the outcomes bridge the knowledge gap prior to work? | With cropping the area of interest, the out-of-the-box text recognition model was able to identify the required information. It is still worth checking other text recognition alternatives than using Google Vision.  It is also worth noting that since these experiments were done using a limited dataset collected manually, the model may not perform well on real-life data of busses. It needs to be retrained with the actual bus images when it is available. | |
| Supporting research | **Papers**  Robust Chinese traffic sign detection and recognition with deep convolutional neural network  <https://www.researchgate.net/publication/304290449_Robust_chinese_traffic_sign_detection_and_recognition_with_deep_convolutional_neural_network>  An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition  <https://arxiv.org/pdf/1507.05717.pdf>  Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks  <https://arxiv.org/abs/1312.6082>  Speed/accuracy trade-offs for modern convolutional object detectors  <https://arxiv.org/pdf/1611.10012.pdf>  **Annotation tool**  <https://github.com/tzutalin/labelImg>  Tensorflow's Object Detection API provides implementations of object detection pipelines, including Faster R-CNN, MobileNet, SSDLite and others with pre-trained models:  <https://ai.googleblog.com/2018/07/accelerated-training-and-inference-with.html>  <https://github.com/tensorflow/models/tree/master/research/object_detection>  Similar to above, Detectron2 was released by Facebook AI Research (FAIR) as an open-source project for object detection and segmentation, but didn't find a model that can run on mobile so not considered:  <https://github.com/facebookresearch/detectron2>  **Blog Posts and Articles:**  <https://www.hosstechnology.com/post/how-to-quickly-build-a-tensorflow-training-pipeline>  <https://www.analyticsvidhya.com/blog/2018/03/comprehensive-collection-deep-learning-datasets/>  <https://blog.cambridgespark.com/50-free-machine-learning-datasets-image-datasets-241852b03b49>  **Object Detection Related:**  <https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852>  <https://mc.ai/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo/>  <https://medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06>  <https://missinglink.ai/guides/tensorflow/building-faster-r-cnn-on-tensorflow-introduction-and-examples/>  <https://towardsdatascience.com/fast-r-cnn-for-object-detection-a-technical-summary-a0ff94faa022>  <https://towardsdatascience.com/faster-r-cnn-object-detection-implemented-by-keras-for-custom-data-from-googles-open-images-125f62b9141a>  https://medium.com/@jonathan\_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359  <https://www.tensorflow.org/lite/models/object_detection/overview>  <https://firebase.google.com/docs/ml-kit/object-detection>  <https://medium.com/tensorflow/training-and-serving-a-realtime-mobile-object-detector-in-30-minutes-with-cloud-tpus-b78971cf1193>  <https://www.dlology.com/blog/how-to-train-an-object-detection-model-easy-for-free/>  <https://towardsdatascience.com/deeppicar-part-6-963334b2abe0>  **Digit and Text Recognition Related:**  <https://nanonets.com/blog/ocr-with-tesseract/>  <https://nanonets.com/blog/attention-ocr-for-text-recogntion/> | |