Laporan Hasil Pengerjaan Test Computer Vision Engineer

PT. Integrasi Inti Sinergi

Oleh: Mochammad Lanang Afkaar Arrachmandhika

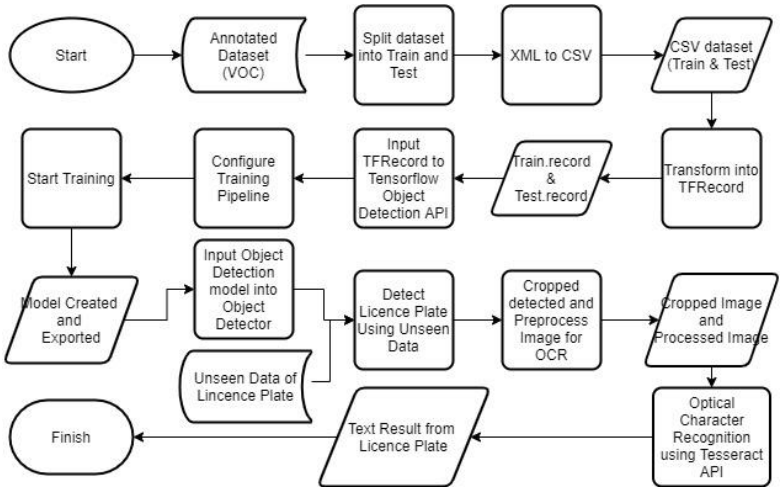
A. Latar Belakang.

Sejak pertama diciptakannya mobil oleh para penemu di eropa dan amerika hingga perkembanganya saat ini. Sudah jutaan bahkan miliaran lebih mobil yang sudah diproduksi oleh pabrik-pabrik mobil dan Sejak dari dulu juga untuk mengenali kepemilikan mobil digunakan tanda pengenal seperti nomor kendaraan dan *ID registration number*. Namun dengan jumlah mobil yang jutaan ini, kemampuan manusia yang terbatas untuk memproses satu persatu nomor kendaraan untuk mengidentifikasi kepemilikan, maka dibutuhkanlah sebuah mesin yang dapat memproses kendaran yang melanggar aturan dan menangkap nomor kendaraan yang terdeteksi.

Maka dari hal yang diatas, dokumen tugas ini bertujuan untuk mengetahui dan memberikan laporan kemampuan sebuah model mesin yang sudah dilatih dan bisa digunakan untuk mendeteksi nomor kendaraan dan membaca nomor kendaraan dari objek nomor kendaraan yang sudah di deteksi.

B. Skema Model.

Dalam model deteksi nomor kendaraan yang dibuat ini, digunakan algoritma *Machine Learning* yang bernama *Faster Region Convolutional Neural Network (Faster R-CNN)* dan *Single Shot Detector MobileNet (MobileNet SSD). Faster R-CNN* merupakan algoritma *Machine Learning* yang biasa digunakan untuk deteksi objek dengan akurasi serta perfoma tinggi namun memiliki kelemahan dimana membutuhkan daya komputasi yang tinggi untuk pelatihannya dan penggunaan modelnya. Sedangkan untuk *MobileNet SSD* sendiri merupakan algoritma deteksi objek yang populer dengan kebutuhan resource khusus untuk komputasi kecil namun dengan perfoma yang dibawah Faster R-CNN.



Gambar 1. Skema Pelatihan Model Deteksi Objek dan OCR untuk Pelat Nomor Kendaraan

Menggunakan dataset yang sudah disediakan yaitu dataset gambar mobil dengan anotasi deteksi objeknya mengarah kepada pelat nomor kendaraan. Dengan menggunakan skema pelatihan pada gambar 1, kita akan melatih modelnya menggunakan dua algoritma yang sudah disebutkan. Pelatihan menggunakan library *Tensorflow Object Detection API* dimana library ini merupakan *open source* dan bisa digunakan untuk membantu pembuatan dan pelatihan model deteksi objek yang akan dibuat. *Tensorflow* menyediakan banyak opsi dan bisa menggunakan GPU/TPU sebagai *resource* yang membantu proses pelatihan menjadi cepat.

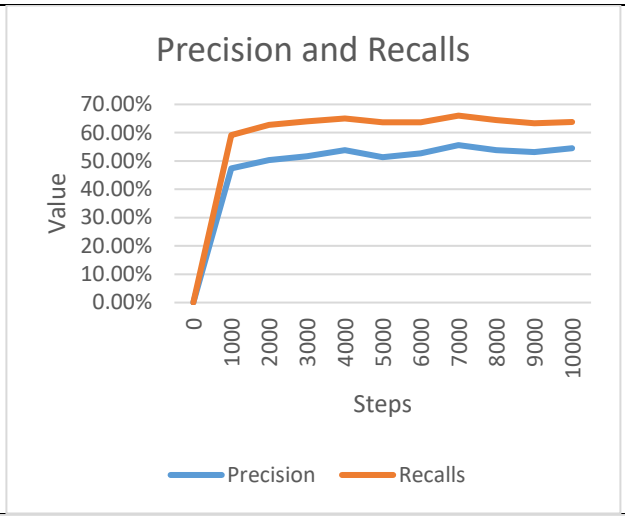
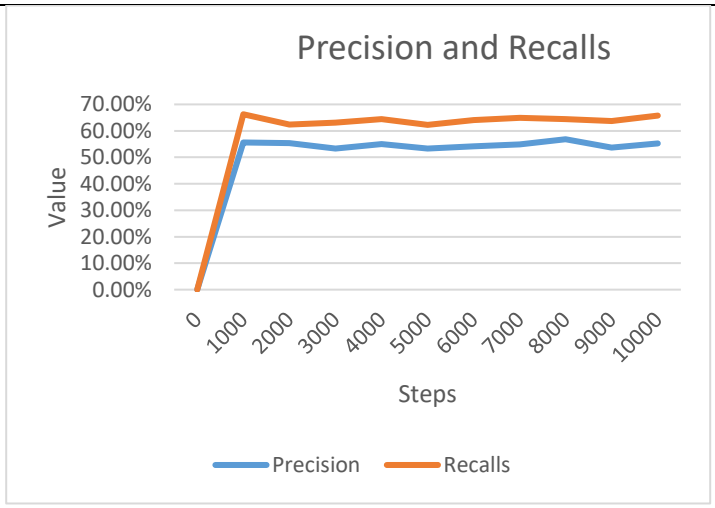
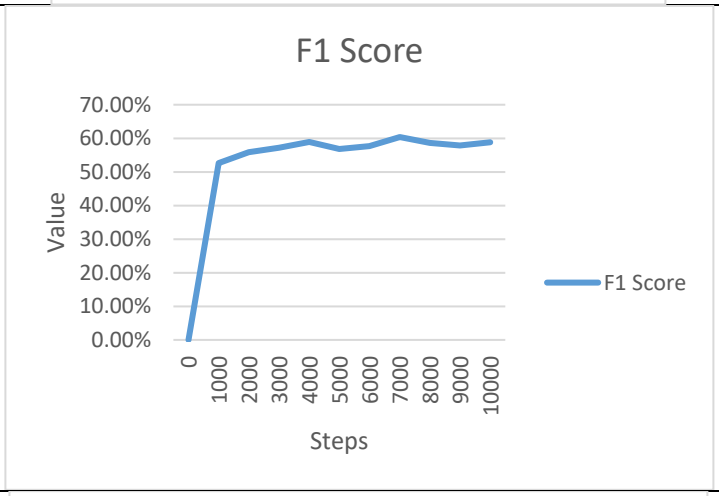
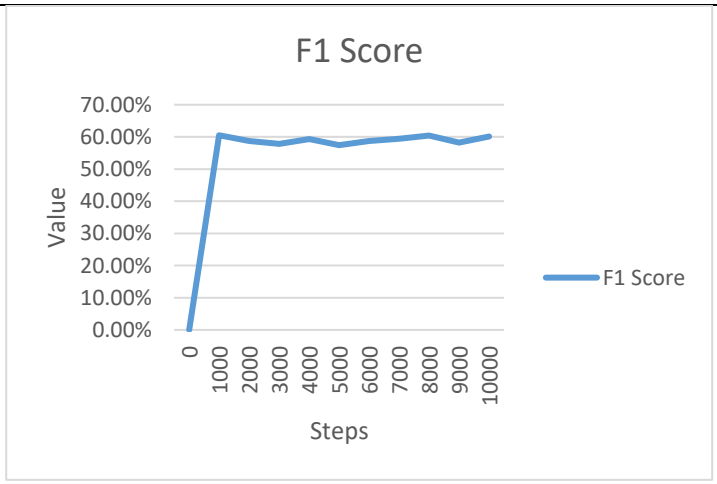
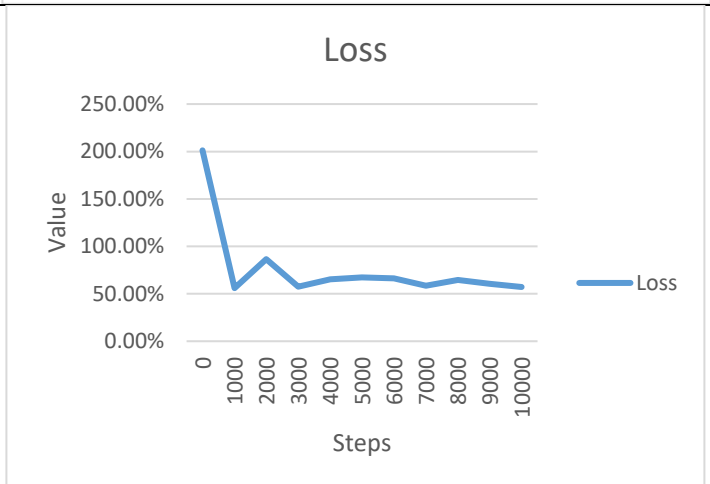
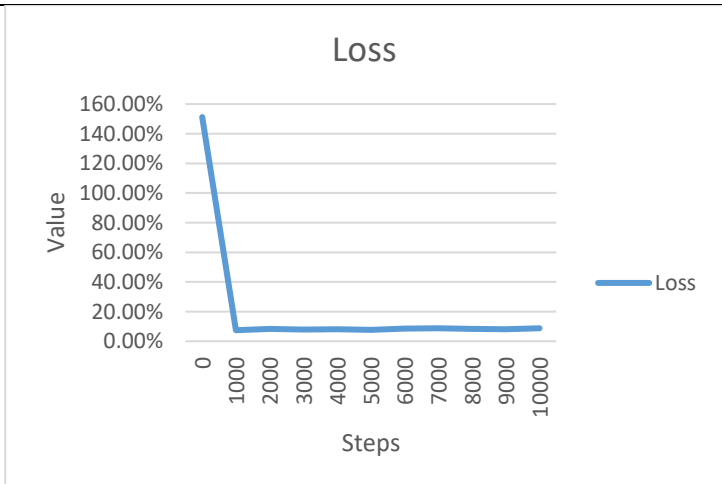
Pada model *Optical Character Recognition*, kita menggunakan Tesseract API untuk mengenali abjad karakter ABCD normal. Sebelum kita memasukkan gambar yang sudah dipotong berdasarkan hasil deteksi objek, kita akan melakukan *pre-processing* gambar tersebut menggunakan library OpenCV. *Pre-processing* yang dilakukan seperti mengubah menjadi *grayscale, Binarization, Deskew* (Perubahan rotasi) dan lain-lain. Setelah pre-processing selesai, maka gambar siap untuk di masukkan ke Tesseract API dan mengeluarkan text yang ada dalam tersebut.

Gambar 2. Deteksi Objek yang Berhasil, Sudah Dipotong, dan Dilakukan Preprocessing.

Penggunaan libray Tesseract API pada model yang akan dibuat adalah karena Tesseract API adalah sebuah *Object Character Recognition* yang memiliki ke akuratan yang lebih baik daripada librart OCR yang lainnya yaitu GOCR dan OCROPUS. Tesseract merupakan proyek open source yang sudah lama dimana awalnya dikembangkan di Hewlett-Packard Laboratories Bristol dan di Hewlett-Packard Co, Greeley Colorado antara tahun 1985 dan 1994, dengan beberapa perubahan lagi yang dilakukan pada tahun 1996 untuk port ke Windows, dan beberapa C++izing pada tahun 1998. Pada tahun 2005 Tesseract dibuka bersumber dari HP. Dari 2006 hingga November 2018 dikembangkan oleh Google.

C. Hasil Pelatihan Model

Tabel 1. Grafik Training dan Evaluasi selama 10000 *Steps*

Jenis Hasil Pelatihan	MobileNet SSD	Faster R-CNN
Precisions dan Recalls		
F1 *Score*		
Loss		

Tabel 2. *Training* dan Evaluasi *MobileNet SSD*

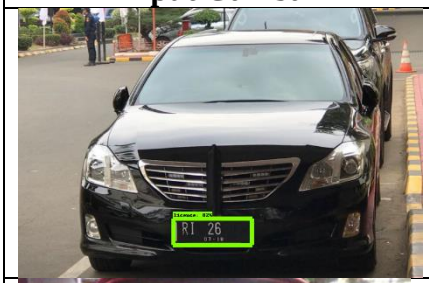
Step	Precision	Recalls	F1 Score	Loss
0	0.00%	0.00%	0.00%	201.19%
1000	47.43%	59.12%	52.63%	55.96%
2000	50.38%	62.79%	55.90%	86.67%
3000	51.69%	63.97%	57.18%	57.64%
4000	53.89%	65.00%	58.92%	65.36%
5000	51.31%	63.68%	56.83%	67.42%
6000	52.70%	63.68%	57.67%	66.16%
7000	55.61%	66.03%	60.38%	58.42%
8000	53.85%	64.41%	58.66%	64.52%
9000	53.17%	63.38%	57.83%	60.48%
10000	54.54%	63.82%	58.82%	57.24%

Tabel 3. Training dan Evaluasi *Faster R-CNN*

Step	Precision	Recalls	F1 Score	Loss
0	0.00%	0.00%	0.00%	151.22%
1000	55.67%	66.27%	60.51%	7.42%
2000	55.39%	62.39%	58.68%	8.24%
3000	53.28%	63.13%	57.79%	7.92%
4000	54.97%	64.48%	59.35%	8.15%
5000	53.36%	62.24%	57.46%	7.75%
6000	54.17%	64.03%	58.69%	8.44%
7000	54.84%	64.93%	59.46%	8.68%
8000	56.83%	64.48%	60.41%	8.23%
9000	53.63%	63.73%	58.25%	8.07%
10000	55.25%	65.82%	60.08%	8.84%

Dengan melihat grafik perbandingan pada tabel 1 dan skor tabel *Training*/Evaluasi dapat disimpulkan bahwa *Faster* R-CNN memiliki keunggulan dalam deteksi objek pelat nomor kendaraan baik dalam training atau evaluasi namun perlu di ingatkan kembali bahwa *Faster* R-*CNN* merupakan algoritma yang haus *resource* dalam pemrosesannya maka dari itu menggunakan MobileNet SSD bisa juga dipertimbangkan sebagai *cost-effective reason* untuk membangun mesin deteksi yang murah.

Pada model OCR, kita akan menggunakan Faster R-CNN untuk model deteksi objek pelat nomor kendaraan. Setelah dilakukan pemotongan dan *pre-processing* yang dipilih yaitu Grayscale dan Noise Reduction. Namun ada beberapa kelemahan untuk model OCR yang kita buat dimana kita harus membutuhkan image size yang cukup besar dan maka dari itu bisa mampu membaca karakter.

Input Gambar	Hasil Deteksi dan OCR	Keterangan
		Model OCR hanya bisa mendeteksi beberapa tulisan namun tulisannya yang diambil salah
		Model bisa mendeteksi semua karakter namun ada karakter yang salah karena huruf “S” itu tersebut mirp dengan angka “5”

D. Kesimpulan

Dengan menggunakan algoritma Faster R-CNN, model deteksi objek pelat nomor kendaraan mampu mendeteksi pelat nomor dengan cukup baik dan dengan menggunakan lokasi deteksi yang ditentukan, melakukan pemotongan gambar dan melakukan pengenalan karakter di pelat nomor kendaraan tersebut namun masih membutuhkan preprocessing yang lebih lanjut seperti *Deskew*, *Skeletonization, thinning.* Dengan begitu akan menghasilkan pengenalan karakter yang lebih baik. Serta lebih baik jika input gambar kendaraan memiliki pixel yang lebih tinggi daripada 400x400 agar pembacaan karakter lebih jelas.