**PROJEK AKHIR KECERDASAN BUATAN**

**“DETEKSI NOMOR PLAT KENDARAAN**

**MENGGUNAKAN ALGORITMA**

**YOLO DAN TENSERFLOW”**



DISUSUN OLEH :

Khalilurrahman Almundzir

(09011281924051)

DOSEN PENGAMPU :

Prof. Dr. Ir. Siti Nurmaini, M.T.

JURUSAN SISTEM KOMPUTER

FAKULTAS ILMU KOMPPUTER

UNIVERSITAS SRIWIJAYA

TAHUN 2021

YOLO menerapkan jaringan syaraf tunggal pada keseluruhan gambar. Jaringan ini akan membagi gambar menjadi wilayah-wilayah kemudian memprediksi kotak pembatas dan probabilitas, untuk setiap kotak wilayah pembatas ditimbang probabilitasnya untuk mengklasifikasian sebagai objek atau bukan.

Table

Description automatically generated

YOLO adalah jaringan syaraf yang pintar untuk melakukan deteksi secara real-time. YOLO memiliki arsitektur yang sederhana yaitu jaringan saraf convolutional. Jaringan saraf ini hanya menggunakan jenis lapisan standar: konvolusi dengan kernel 3 × 3 dan max-pooling dengan 2 × 2 kernel. Lapisan konvolusional terakhir memiliki 1 × 1 kernel digunakan untuk mengecilkan data ke bentuk 13 × 13 × 125. 13×13 ini seharusnya terlihat familier: itu adalah ukuran grid yang dibagi menjadi gambar. 125 merupakan Channel untu setiap grid. 125 ini berisi data untuk kotak pembatas dan prediksi kelas. Kenpa 125? Nah, setiap sel grid memprediksi 5 kotak sekeliling dan dijelaskan oleh 25 elemen data.

OpenCV (Open Source Computer Vision Library), adalah sebuah library open source yang dikembangkan oleh intel yang fokus untuk menyederhanakan programing terkait citra digital. Di dalam OpenCV sudah mempunyai banyak fitur, antara lain : pengenalan wajah, pelacakan wajah, deteksi wajah, Kalman filtering, dan berbagai jenis metode AI (Artificial Intellegence). Dan menyediakan berbagai algoritma sederhana terkait Computer Vision untuk low level API.

TensorFlow merupakan library open source untuk menghitung komputasi numerik secara mudah dan cepat. TensorFlow dirancang untuk penelitian dan pengembangan sistem produksi.

1. PRAPENGOLAHAN

Kita ketahui bahwa cuaca atau intensitas cahaya setiap harinya sulit untuk ditebak, citra hasil pengambilan data tidak selalu memiliki level cahaya yang sama, ada yang lebih terang ada yang lebih gelap, bahkan dalam satu citra kendaraan pun terkadang terdapat bagian yang terhalang oleh bayangan sehingga apabila tidak dilakukan prapengolahan mungkin bisa mengacaukan proses selanjutnya. Proses ini digunakan untuk mempersiapkan citra agar mudah dalam mendeteksi plat.

Citra RGB diload ke sistem (OpenCV akan membacanya sebagai BGR), citra tersebut diresize karena rasanya sulit untuk mengamati hasilnya apabila ukurannya terlalu besar saat ditampilkan di layar (hampir full menutupi satu layar). Saya resize dengan mengalikan lebar dan tingginya dengan nilai 0.4 sehingga yang asalnya berukuran 1920 x 2560 sekarang menjadi 768 x 1024, lebih kecil, tidak menutupi hampir seluruh layar ketika menampilkannya.

Sekarang citra akan dilakukan normalisasi cahaya, fungsinya agar intensitas cahayanya sama, lebih mudah untuk dilanjutkan ke proses berikutnya. Normalisasi cahaya ini prosesnya dimulai dari mengubah citra RGB ke grayscale, lakukan operasi opening ke citra grayscale, lalu kurangkan citra grayscale dengan citra hasil opening, hasil pengurangan tersebut selanjutnya bisa dikonversi ke citra BW (hitam putih) dengan pengambangan Otsu.

Bisa dilihat pada gambar di bawah perbedaan citra BW hasil normalisasi dan tanpa normalisasi, bagian plat dan karakter yang dihasilkan pun lebih jelas apabila dilakukan normalisasi.

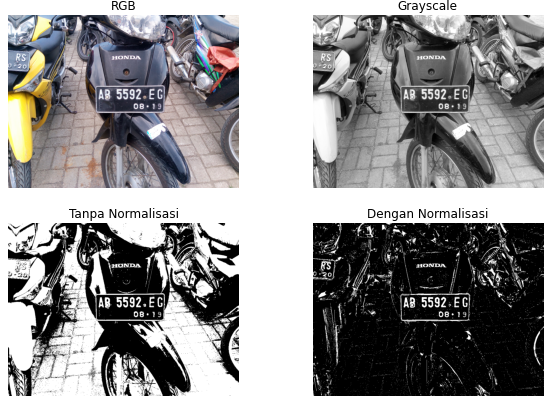


Figure 1. Proses Prapengolahan

1. DETEKSI PLAT

Dari hasil pengamatan pada hasil prapengolahan, kita melihat bahwa bagian plat nomor membentuk garis kotak dan memiliki intensitas yang sama.

Pada Projek kali ini kita akan menggunakan fungsi *contours*, karena dilihat dari hasil citra prapengolahan yang mana kontur atau ares plat bisa dilihat dengan lumayan jelas, saling berhubungan membuat suatu area.

Dari semua area kontur pada citra, untuk mendapatkan area plat nomor, saya filter berdasarkan aspect ratio dan lebarnya, jika lebarnya lebih dari atau sama dengan 200 piksel dan aspect rationya kurang dari atau sama dengan 4.

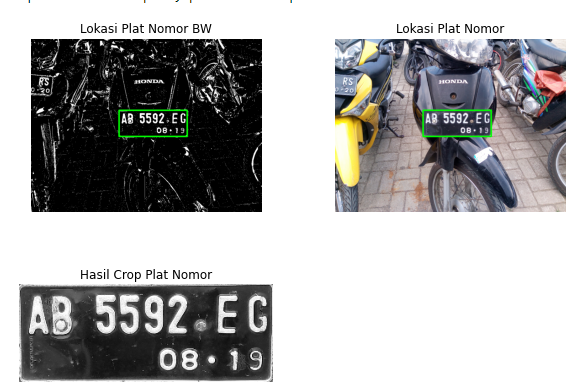


Figure 2. Deteksi plat nomor

1. SEGMENTASI KARAKTER

Contours juga saya gunakan untuk segmentasi karakter. Citra hasil crop diubah ke citra BW, lalu karena hasilnya terdapat area piksel yang tidak diinginkan maka lakukan operasi opening, operasi ini akan menghilangkan area kecil dekat karakter.

Untuk mendapatkan kandidat karakternya lakukan filter terhadap kontur dengan berdasarkan lebar dan tingginya, yaitu apabila tingginya dalam rentang 40-60 piksel dan lebarnya lebih dari 10 piksel.

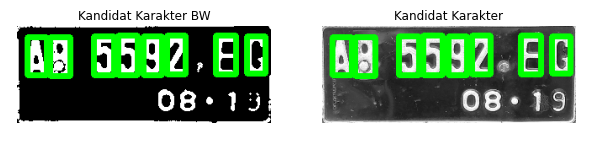


Figure 3. Kandidat Karakter

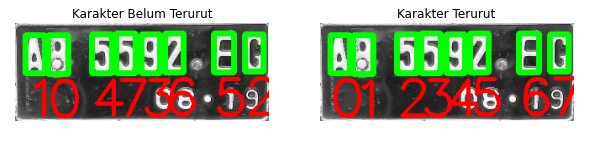
Apabila dilihat dari hasil di atas, kandidat karakter memiliki bagian yang bukan karakter. Untuk mendapatkan bagian yang benar sebuah karakter, kita bisa mengecek apakah bagian tersebut sebaris, berderet secara horizontal, atau memiliki letak sumbu y yang tidak jauh berbeda.

Kita bisa menghitung selisih letak sumbu y antara kandidat satu dengan yang lainnya, setiap kandidat yang selisihnya kurang dari 11 piksel maka mendapatkan 1 score/poin. Kandidat yang memiliki score yang sama dan tertinggi maka itu adalah karakter yang sesungguhnya.





Dari sini kita sudah memiliki bagian yang benar-benar sebuah karakter. Namun sayangnya karena kita menggunakan contours, karakter tersebut tidak berurutan dari kiri ke kanan melainkan dari atas ke bawah, oleh karena itu mari kita urutkan. Pengurutan ini penting sekali karena menentukan urutan karakter yang nantinya akan diklasifikasi oleh model, kalo urutannya ngaco meskipun klasifikasi karakternya benar ya tetap saja ngaco.



1. KLASIFIKASI KARAKTER

Untuk klasifikasi karakter, di sini saya menggunakan model dari tutorial Tensorflow ini, jadi kita bikin modelnya terlebih dahulu, lalu melatihnya dengan dataset karakter. Hasil dari model terlatih bisa digunakan untuk klasifikasi karakter.

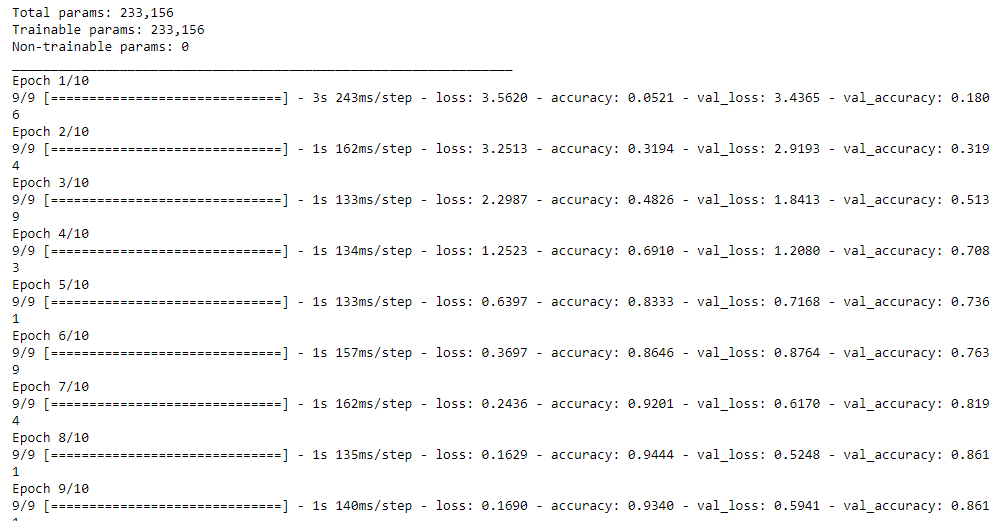
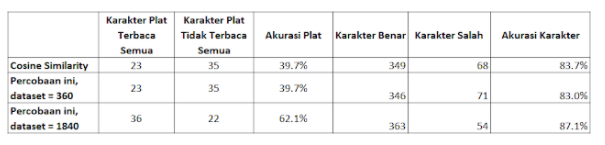


Figure 4. Pelatihan Model

Dan hasil klasifikasinya seperti ini:



Jadi memang ada banyak hal yang mempengaruhi hasilnya, kalau dari data diatas, dataset yang lebih banyak memang sangat disarankan agar klasifikasi lebih baik. Jika memakai dataset yang sedikit maka performanya tidak jauh beda dengan memakai cosine similarity yang hanya memakai 36 dataset saja.

Kebanyakan misklasifikasi adalah karakter yang mirip seperti 1 dengan I atau B dengan 8, solusinya bisa memisahkan karakter ke dalam 3 bagian, yaitu huruf awal, angka tengah, dan huruf akhir. Selain itu juga karena hasil prapengolahan atau segmentasi yang kurang bagus.

