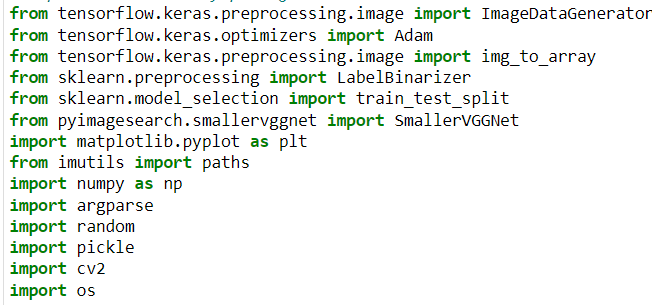


Pertama, import lalu atur backend matplotlib sehingga dapat disimpan di latar belakang

kita mengatur backend matplotlib ke "Agg" sehingga kita dapat menyimpan plot ke disk di latar belakang. Ini penting jika kita menggunakan headless server untuk melatih jaringan kita (seperti Azure, AWS, atau instans cloud lainnya).



Import Package yang diperlukan

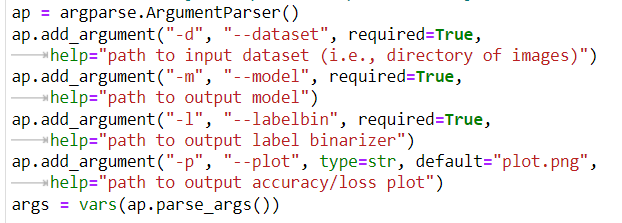
Package diatas memungkinkan kita untuk :

Muat dataset gambar kita dari disk

Pra-proses gambar

Buat Instansi Jaringan Saraf Konvolusi kita

Latih pengklasifikasi gambar kita



membangun argumen mengurai dan mengurai argument

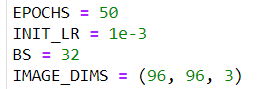
Di sini kita memiliki dua argumen baris perintah yang diperlukan, “—dataset” dan “—model”

serta jalur opsional ke diagram accuracy/loss kita yaitu “—plot” .

“—dataset” harus menunjuk ke direktori yang berisi gambar tempat kita akan melatih pengklasifikasi gambar kita (yaitu, gambar mesin kendaraan yang "Baik" dan "Tidak Baik")

sementara “—model” mengontrol tempat kita akan menyimpan pengklasifikasi gambar serial kita setelah dilatih.

Dan “—plot” untuk mengarahkan diagram hasil training ke direktori yang diinginkan, jika dibiarkan tidak ditentukan, maka akan default ke plot.png di direktori root.



EPOCHS: Jumlah total epoch yang akan kita latih untuk jaringan kita (yaitu, berapa kali jaringan kita “melihat” setiap contoh pelatihan dan mempelajari pola darinya).

INIT\_LR: Tingkat pembelajaran awal — nilai 1e-3 adalah nilai default untuk Adam Optimizer, pengoptimal yang akan kita gunakan untuk melatih jaringan. Learning rate merupakan salah satu parameter pelatihan untuk menghitung nilai koreksi bobot selama proses pelatihan. Nilai learning rate ini berada pada rentang nol (0) sampai (1). Semakin tinggi tingkat pembelajaran, semakin cepat proses pelatihan akan berjalan

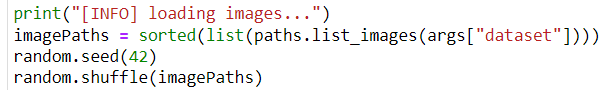
BS (Batch Size): Kita akan mengirimkan kumpulan gambar ke jaringan kita untuk pelatihan. Ada beberapa batch per Epoch. Nilai BS mengontrol ukuran batch.

IMAGE\_DIMS: Di sini kita menyediakan dimensi spasial dari gambar masukan kita. Kita akan membutuhkan gambar input kita menjadi 96 x 96 piksel dengan 3 saluran (yaitu, RGB).



Membuat daftar “data” dan “label” yang nantinya menampung data sesuai dengan namanya atau pada umumnya bisa disebut menginisialisasi data dan label

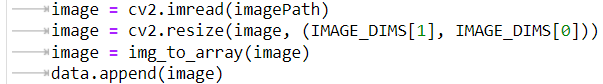
Penginisialisasian ini akan bertanggung jawab untuk menyimpan gambar yang kita muat dari disk bersama dengan label kelasnya masing-masing.



mengambil jalur direktori ke gambar input diikuti dengan mengacaknya



Loop imagePaths



dan melanjutkan untuk memuat gambar

kemudian mengubah ukurannya untuk mengakomodasi model yang kita punya

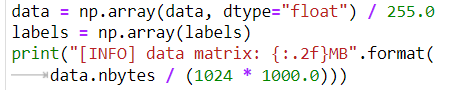
lalu dengan memakai function “img\_to\_array” untuk mengonversi gambar ke array yang kompatibel dengan Keras

dan gambar tersebut akan di tambahkan kedalam daftar “data” yang sudah di inisialisasi sebelumnya



Mengekstrak label dari file direktori

Kemudian menambahkannya juga kedalam daftar “label” yang juga suda di inisialisasi sebelumnya



Di sini pertama-tama kita mengonversi data array ke array NumPy kemudian menskalakan intensitas piksel ke kisaran [0, 1]

Dan kita juga mengonversi label dari daftar sebelumnya ke array NumPy.  
Lalu output INFO dicetak untuk menunjukkan/menginformasikan ukuran (dalam MB) dari matriks data.

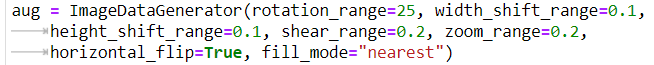


Kemudian, kita melakukan binerisasi label menggunakan LabelBinarizer dari scikit-learn



Dengan Deep Learning, atau Machine Learning apa pun dalam hal ini, praktik umum adalah membuat pemisahan training dan testing

ini ditangani pada baris diatas di mana kita membuat pemisahan data secara acak 80/20 (80 untuk pengujian dan sisanya untuk pelatihan).

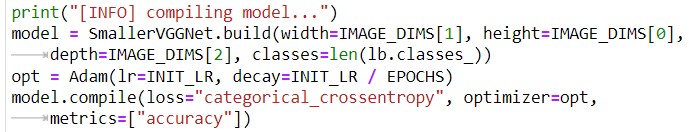


Lalu buat objek augmentasi data gambar kita

Karena kita memiliki jumlah data terbatas (<250 gambar per kelas), kita dapat menggunakan augmentasi data selama proses pelatihan untuk memberikan model lebih banyak gambar (berdasarkan gambar yang ada) untuk dilatih.

Augmentasi Data adalah alat yang harus ada di setiap Deep Learning. Kita menginisialisasi aug, dengan ImageDataGenerator seperti yang ada pada kode diatas.

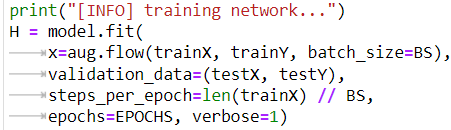
Setelah itu kita Kompilasi (Compile) Model dan Mulai Pelatihan (Training)



kita menginisialisasi model CNN Keras kita dengan dimensi spasial input 96 x 96 x 3. SmallerVGGNet dirancang untuk menerima gambar masukan 96 x 96 x 3.

Lalu kita menggunakan Adam Optimizer dengan peluruhan tingkat pembelajaran

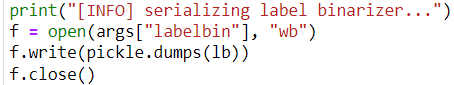
dan kemudian kompilasi model dengan categorical cross entropy karena kita memiliki > 2 kelas



Kode diatas melakukan pemanggilan *Keras* fit method untuk melatih jaringan (biasanya membutuhkan waktu selama training/pelatihan dimulai)

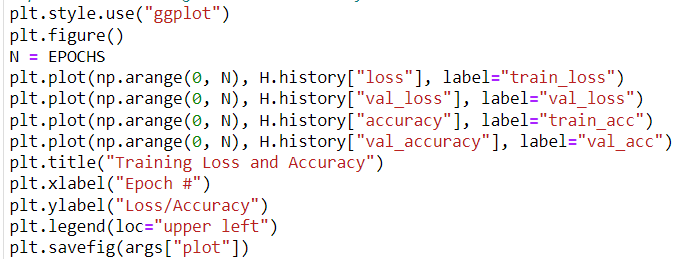


Setelah Training/Pelatihan selesai maka model akan disimpan kedalam penyimpanan computer dengan perintah diatas, sehingga model yang sudah jadi nantinya dapat digunakan.



Dan juga menyimpan binarizer label kedalam disk

File binarizer label berisi indeks kelas ke kamus label kelas yang dapat dibaca manusia. Objek ini memastikan kita tidak perlu membuat hardcode label kelas kita dalam skrip yang ingin menggunakan Keras CNN kita.



Dan terakhir, kode diatas yang akan membuat plot hasil training model yang sudah dilatih sebelumnya dan akan menghasilkan gambaran berupa grafik yang menginformasikan performa selama training/pelatihan berjalan, seperti pada gambar berikut :

