Deteksi dan Klasifikasi Objek pada Citra Sel Darah menggunakan Metode Berbasis *Deep Learning*

Samatha Marhaendra Putra
Departemen Teknik Elektro dan Teknologi Informasi
Fakultas Teknik, Universitas Gadjah Mada
Yogyakarta, Indonesia
sam.marhaendrap@mail.ugm.ac.id

***Abstrak*—Sel darah merupakan komponen yang ada pada darah. Terdapat beberapa jenis sel darah, dua di antaranya yakni sel darah merah (*red blood cells*) dan sel darah putih (*white blood cells*). Dalam dunia medis, pendeteksian dan penghitungan jumlah sel darah menjadi suatu hal yang penting guna mengevaluasi kondisi kesehatan seseorang secara keseluruhan. Pendeteksian dan penghitungan jumlah sel darah secara manual menggunakan haemocytometer memakan waktu yang banyak dan melibatkan pekerjaan yang membosankan. Permasalahan yang digarap dan tujuan yang akan dicapai yakni terkait dengan deteksi dan penghitungan dua tipe sel darah dalam waktu yang singkat dengan tingkat akurasi yang tinggi. Metode yang dipakai menggunakan pendekatan *Deep Learning*. Algoritma yang dipakai yaitu algoritma deteksi objek dan klasifikasi 'You Only Look Once' (YOLO). Luaran yang dicapai dari pengerjaan proyek ini yaitu citra medis sel darah yang telah terdapat hasil deteksi objek berupa *bounding box* beserta hasil klasifikasi dan tingkat keyakinan terhadap suatu klasifikasi tersebut dalam waktu yang sangat cepat. *Metric* yang digunakan yakni *mean Average Precision* (mAP).**

Kata kunci— Blood Cell, Classification, Object Detection, Object Counting, YOLO Architecture

I. LATAR BELAKANG

Sel darah merupakan salah satu komponen utama yang terdapat pada darah. Kebutuhan terkait dengan pendeteksian dan penghitungan sel darah menjadi hal yang penting di dunia medis guna mengevaluasi kondisi kesehatan seseorang [1]. Beberapa tipe sel darah yang utama di antaranya yakni sel darah merah (*red blood cells* atau RBCs) dan sel darah putih (*white blood cells* atau WBCs). RBCs merupakan tipe sel darah yang paling umum, di mana tipe sel darah ini memiliki proporsi sekitar 40-45% dari keseluruhan sel darah. Tipe sel darah ini bertugas membawa oksigen menuju sel-sel pada tubuh. Jumlah sel darah dengan tipe ini memengaruhi seberapa banyak oksigen yang dapat dibawa menuju sel-sel pada tubuh. WBCs merupakan tipe sel darah yang proporsinya cukup kecil, hanya sekitar 1% dari keseluruhan sel darah. Tipe sel darah ini bertugas untuk melawan infeksi. Metode pendeteksian dan penghitungan sel darah yang dilakukan secara manual menggunakan haemocytometer sangat memakan waktu dan tingkat akurasinya sangat bergantung pada analis laboratorium, yang mana karena sangat bergantung pada manusia, metode ini memiliki risiko terjadinya kesalahan yang besar [2]. Di sini terlihat bahwa dengan adanya suatu sistem yang mengotomasi proses pendeteksian dan penghitungan sel darah dari citra medis akan sangat membantu proses penghitungan sel darah secara keseluruhan.

Secara umum, permasalahan ini dapat diselesaikan menggunakan dua macam pendekatan. Pendekatan pertama menggunakan pendekaan *image processing*. Pendekatan kedua menggunakan pendekatan *machine learning*.

Salah satu penelitian yang menggunakan pendekatan *image processing* pada permasalahan seperti ini yaitu penelitian Acharya dan Kumar [3] yang mengusulkan tentang penghitungan RBCs menggunakan teknik *image processing*. Penelitian ini melakukan pemrosesan citra medis sel darah untuk kemudian dilakukan penghitungan RBCs bersama dengan pengidentifikasian sel yang normal dan tidak normal. Algoritma yang dipakai yakni K-medoids yang bertugas untuk mengekstrak WBCs dari citra medis dan melakukan analisis *granulometric* untuk memisahkan RBCs dari WBCs kemudian dilanjutkan dengan penghitungan jumlah sel menggunakan algoritma pelabelan dan *circular Hough Transform* (CHT).

Penelitian lain yang menerapkan pendekatan *machine learning* pada permasalahan serupa yaitu penelitian Zhao *et al*. [4] yang mengusulkan tentang sistem identifikasi dan klasifikasi yang otomatis untuk WBCs menggunakan *convolutional neural network* (CNN). Tahap pertama di sini yakni diawali dengan melakukan pendeteksian WBCs dari citra mikroskopis, yang kemudian CNN dipakai untuk mendeteksi jenis WBCs.

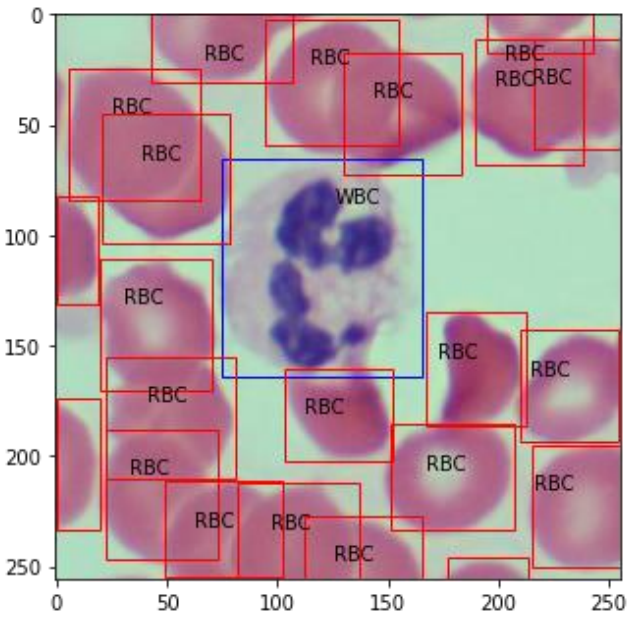
Penulis mengusulkan metode penyelesaian permasalahan seperti ini menggunakan YOLO yang mampu mendeteksi berbagai macam tipe sel darah secara simultan. Metode ini tidak memerlukan pemrosesan citra seperti pengonversian *grayscale* dan segmentasi biner. Pemrosesan menggunakan metode yang diusulkan ini terotomasi, cepat, dan menghasilkan akurasi yang tinggi.

II. DATA DAN METODOLOGI

A. Data

Data yang dipakai yaitu dataset citra medis sel darah yang telah disediakan. Dataset ini terdiri dari 100 citra medis sel darah yang dilengkapi dengan sebuah berkas berisi anotasi citra yang menjadi *ground truth* dari dataset yang dipakai. Jumlah label yang terdapat pada dataset yaitu 2237 RBC dan 103 WBC.

Data citra yang diberikan diberikan dalam mode RGB dengan ukuran 256 x 256 piksel.



Gambar 1. Citra Medis Sel Darah dengan *Ground Truth*-nya

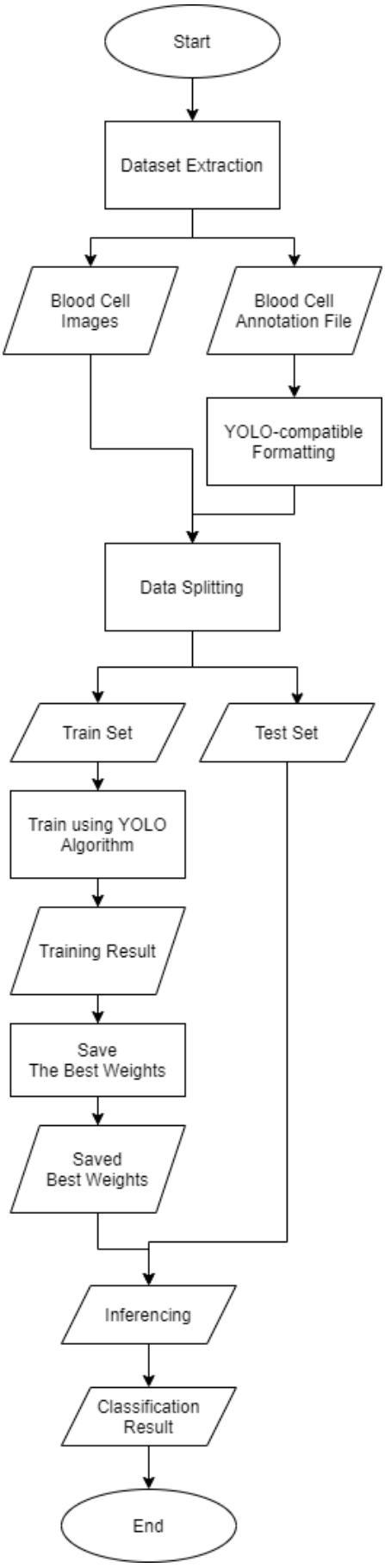
	image	xmin	ymin	xmax	ymax	label
0	image-100.png	0.000000	0.000000	25.190198	40.816803	rbc
1	image-100.png	15.010502	0.000000	68.337223	23.527421	rbc
2	image-100.png	25.017503	16.021004	78.374562	73.735123	rbc
3	image-100.png	75.565928	1.061844	140.248541	45.591599	rbc
4	image-100.png	77.483081	23.290548	131.936989	74.806301	rbc

Gambar 2. Data Anotasi Citra Medis Sel Darah

Gambar 1 merupakan contoh salah satu citra medis sel darah yang dilengkapi dengan *ground truth* berupa *bounding boxes* beserta labelnya. Gambar 2 merupakan data anotasi dari sekumpulan citra medis sel darah yang terdapat pada dataset yang disimpan dalam suatu berkas dengan format berkas CSV.

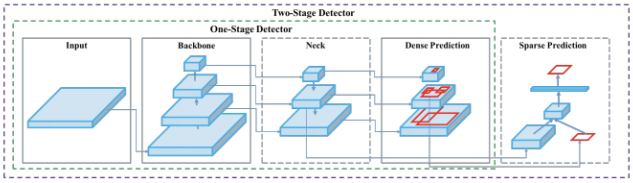
B. Proposed Method

Proses yang dilakukan pada proyek ini yaitu pemformatan data, pembagian data menjadi *train set* dan *test set*, pelatihan model, dan diakhiri dengan melakukan prediksi/*inference*.



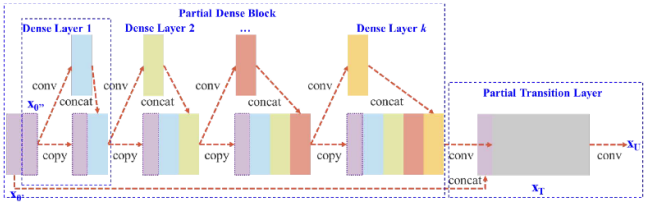
Gambar 3. Alur Pengerjaan Proyek

Gambar 3 menjelaskan alur pengerjaan proyek. Pada tahap awal, dilakukan ekstraksi data dari sumber yang disediakan. Kemudian, dilakukan pemformatan data anotasi agar sesuai dengan format yang *compatible* dengan algoritma YOLO. Setelah itu, proses dilanjutkan dengan melakukan *data splitting* menjadi *train set* dan *validation set*. Ini bertujuan untuk melihat seberapa baik algoritma yang dipakai dalam mengeneralisasi permasalahan yang dihadapi. *Weights* yang menghasilkan hasil terbaik sebagai hasil dari pelatihan model kemudian disimpan untuk kemudian di-*load* pada saat melakukan *inference* terhadap *validation set*.

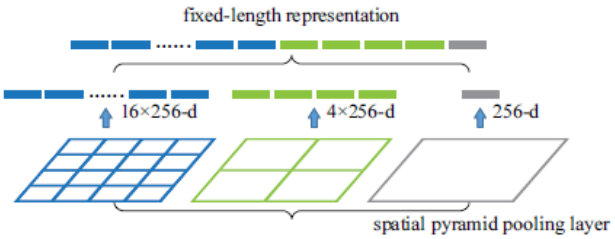


Gambar 4. *Overview* Proses *Object Detection* [6]

Berdasarkan Gambar 4, arsitektur YOLO memiliki tiga bagian utama, yakni *Backbone*, *Neck*, dan *Head*. Pada bagian pertama, yakni *Backbone*, merupakan suatu CNN yang menggabungkan dan membentuk fitur dari gambar dalam tingkat *granularity* yang berbeda [5]. Berdasarkan [7], *Backbone* yang dipakai di sini yakni CSPDarknet53 seperti ditunjukkan pada Gambar 5, dengan *Spatial Pyramid Pooling* (SPP) seperti ditunjukkan pada Gambar 6 berfungsi untuk meningkatkan *receptive field*, yang mana memisahkan fitur-fitur yang signifikan, dan tidak ada pengurangan dari kecepatan operasional *network*.

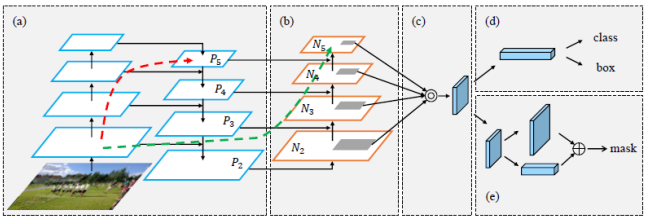


Gambar 5. *Cross Stage Partial (CSP) DenseNet* [7]



Gambar 6. *Spatial Pyramid Pooling (SPP) Layer* [8]

Bagian kedua dari arsitektur YOLOv5, yaitu *Neck*, merupakan serangkaian *layer* yang berfungsi untuk mencampur dan menggabungkan fitur dari gambar untuk kemudian diteruskan menuju tahap prediksi [5]. *Neck* yang dipakai di sini yaitu *Path Aggregation Network* (PAN) seperti ditunjukkan pada Gambar 7.



Gambar 7. Arsitektur *Path Aggregation Network* (PAN) [9]

Bagian ketiga dari arsitektur YOLOv5 yakni *Head*. Bagian ini mengonsumsi fitur yang merupakan luaran dari bagian *Neck* untuk kemudian dapat membuat *bounding box* beserta kelas prediksinya [5].

Parameter yang dipakai pada pengerjaan proyek ini yaitu *mean average precision* (mAP) untuk mengetahui seberapa akurat model yang dipakai dalam melakukan tugas deteksi objek sel darah.

III. HASIL DAN ANALISIS

Pada bagian ini dipaparkan terkait langkah-langkah yang diimplementasikan dalam kode program untuk menyelesaikan tugas terkait deteksi objek sel darah. Penjelasan pada bagian ini diawali dengan tahap pemformatan data agar sesuai dengan format yang diminta YOLO. Selanjutnya yakni tahap *data splitting*, di mana data akan dibagi menjadi 2 *sets*, yaitu *train set* dan *validation set*. Setelah itu, tahap selanjutnya yaitu pelatihan model menggunakan *train set* dengan mengatur beberapa hal, seperti *configuration file* dan *arguments* yang diperlukan. Berikutnya yaitu berkaitan dengan hasil latih, di mana ditampilkan *metric* yang merepresentasikan hasil latih model. Tahap terakhir yaitu berkaitan dengan proses inferensi/prediksi pada *validation set*.

A. Pemformatan Data

Pada bagian pertama, data anotasi yang tersimpan dalam format CSV dilakukan penambahan kolom-kolom yang diperlukan oleh YOLO. Gambar 8 menunjukkan data awal yang berisi anotasi objek.

	image	xmin	ymin	xmax	ymax	label
0	image-100.png	0.000000	0.000000	25.190198	40.816803	rbc
1	image-100.png	15.010502	0.000000	68.337223	23.527421	rbc
2	image-100.png	25.017503	16.021004	78.374562	73.735123	rbc
3	image-100.png	75.565928	1.061844	140.248541	45.591599	rbc
4	image-100.png	77.483081	23.290548	131.936989	74.806301	rbc

Gambar 8. Sampel Data Anotasi Awal

YOLO memerlukan data terkait lebar, tinggi, dan posisi titik pusat suatu objek yang telah dinormalisasi. Data terkait lebar dan tinggi dapat dicapai dengan secara berturut-turut menghitung selisih antara *xmax* dan *xmin*, juga *ymax* dan *ymin*. Data terkait posisi titik pusat objek dapat dicapai dengan menambahkan nilai *xmin* dan *ymin* secara berturut-turut dengan setengah dari lebar dan tinggi. Kemudian, untuk mendapatkan nilai yang telah dinormalisasi, dilakukan pembagian antara data yang telah telah didapatkan sebelumnya dengan ukuran lebar dan tinggi dari gambar, di mana pada proyek ini menggunakan gambar dengan ukuran yang sama, yakni 256 x 256. Gambar 9 menunjukkan data

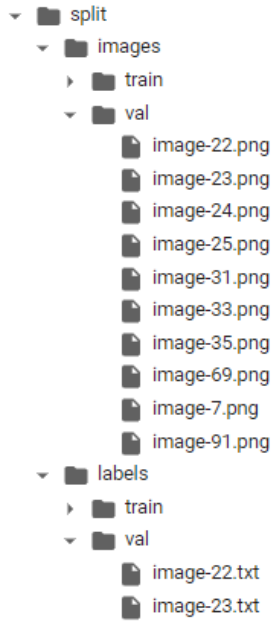
anotasi yang telah mengandung data baru yang didapatkan dari hasil perhitungan pada tahap ini.

	image	width	height	x_center	y_center	x_center_norm	width_norm	y_center_norm	height_norm
0	image-100.png	25.190198	40.816803	12.595099	20.408401	0.049200	0.098399	0.079720	0.159441
1	image-100.png	53.326721	23.527421	41.673862	11.763711	0.162789	0.208308	0.045952	0.091904
2	image-100.png	53.357060	57.714119	51.696033	44.878063	0.201938	0.208426	0.175305	0.225446
3	image-100.png	64.682614	44.529755	107.907235	23.326721	0.421513	0.252666	0.091120	0.173944
4	image-100.png	54.453909	51.515753	104.710035	49.048425	0.409024	0.212711	0.191595	0.201233

Gambar 9. Sampel Data Anotasi setelah Pemformatan Data

B. *Data Splitting*

Pada tahap ini, dilakukan agregasi data menjadi dua bagian, yakni *train set* dan *validation set* dengan proporsi 90% dan 10% dari keseluruhan data. Gambar 10 menunjukkan struktur *subfolders* yang menjadi tempat hasil agregasi data, di mana untuk setiap *set* terdapat dua buah *folder*, yakni *folder* yang menyimpan gambar dan *folder* yang menyimpan berkas dalam format TXT yang berisi anotasi untuk setiap gambar.



Gambar 10. Struktur *Subfolders* Hasil *Data Splitting*

C. Pelatihan Model dan Hasil

Setelah data telah disusun dengan rapi, tahap berikutnya yaitu melakukan pelatihan model YOLO yang dipakai dengan memasukkan *training set* yang telah tersedia. Pada tahap awal di bagian ini, diperlukan kustomisasi *configuration file* dikarenakan YOLO di sini dipakai pada *custom data*. Konfigurasi yang dilakukan yakni terkait dengan penentuan jumlah kelas, nama kelas, dan lokasi di mana *training set* dan *validation set* berada. Gambar 11 menunjukkan proses kustomisasi *configuration files* yang disimpan dalam format YAML.

```
# creating .yaml file for specifying about the paths, number of classes, and class names
!echo -e "train: /content/split/images/train\nval: /content/split/images/val\n\nnc: 2\nnames: ['rbc', 'wbc']" >> rbcdet.yaml
!cat 'rbcdet.yaml'

train: /content/split/images/train
val: /content/split/images/val

nc: 2
names: ['rbc', 'wbc']
```

Gambar 11. Kustomisasi *configuration files*

Dapat dilihat dari Gambar 11 bahwa jumlah kelas pada *configuration file* diatur menjadi 2 dengan dua buah kelas, yakni *rbc* dan *wbc*, yang mana ini didasarkan pada *dataset* yang dipakai.

Berikutnya, dilakukan pelatihan model dengan memberikan argumen seperti ukuran gambar, *batch size*, *epochs*, *configuration file* yang dipakai dan data yang dipakai. Ukuran gambar diatur pada angka 256 sesuai dengan ukuran gambar pada *dataset*. *Batch size* merupakan ukuran yang menspesifikasi berapa banyak data yang terlibat untuk satu kali *epoch*. *Epochs* merupakan jumlah iterasi pelatihan model yang diinginkan. Pada pelatihan model untuk *custom dataset* ini, diberikan nilai *batch size* sebesar 64 dengan 300 *epochs*.

D. Hasil Latih

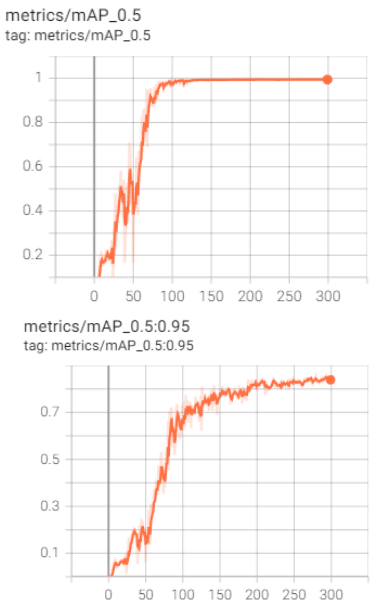
Gambar 12 menunjukkan hasil pelatihan model.

```
300 epochs completed in 0.125 hours.
Optimizer stripped from yolov5/runs/train/rbcdet_model/weights/last.pt, 14.3MB
Optimizer stripped from yolov5/runs/train/rbcdet_model/weights/best.pt, 14.3MB

Validating yolov5/runs/train/rbcdet_model/weights/best.pt...
Fusing layers...
YOLOv5s summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00,  3.54it/s]
                 all         10        241      0.979      0.983      0.994      0.855
                 rbc         10        231      0.976      0.965      0.993      0.808
                 wbc         10         10      0.982          1      0.995      0.902
Results saved to yolov5/runs/train/rbcdet_model
CPU times: user 4.61 s, sys: 524 ms, total: 5.13 s
Wall time: 8min 10s
```

Gambar 12. Hasil Pelatihan Model

Dari Gambar 12, diketahui bahwa proses pelatihan model memakan waktu 8 menit 10 detik secara keseluruhan. Didapatkan pula nilai mAP_0.5 yang mencapai 99% untuk setiap kelas dan nilai mAP_0.5:0.95 yang secara keseluruhan mencapai 85%. Ini menunjukkan bahwa model mampu menghasilkan performa yang sangat baik untuk memprediksi semua kemungkinan kelas. Hal ini dikarenakan semakin besar nilai mAP, maka hasil prediksi yang berupa *bounding box* beserta kelasnya semakin menyerupai *ground truth*. Gambar 13 merupakan grafik yang ditampilkan menggunakan Tensorboard yang menunjukkan peningkatan nilai mAP_0.5 dan mAP_0.5:0.95 seiring meningkatkan jumlah iterasi saat pelatihan model.



Gambar 13. Grafik Pergerakan Nilai mAP di setiap Epoch

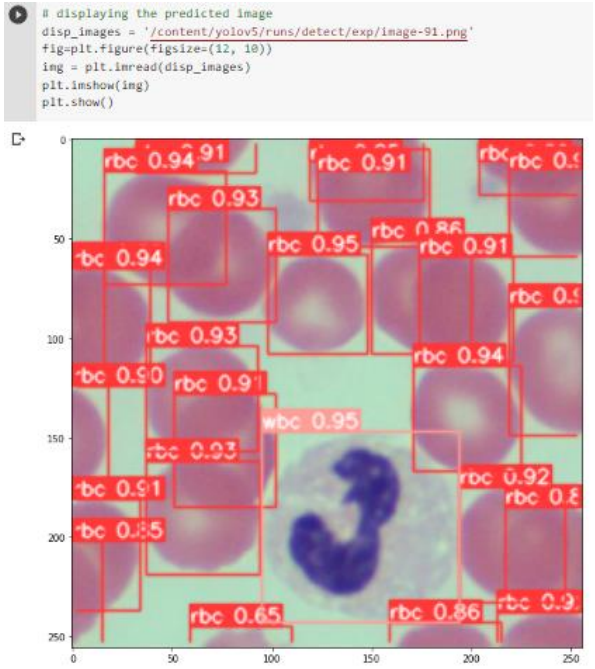
Dari Gambar 13 dapat dilihat bahwa pelatihan model mencapai hasil optimum ketika berada di kisaran iterasi pelatihan ke-100. Bobot terbaik pasti didapatkan pada saat

kisaran iterasi tersebut. Bobot itulah yang dipakai pada tahap terakhir, yakni tahap inferensi.

E. Inferensi

Setelah dilakukan proses pelatihan model, tahap terakhir yakni berkaitan dengan proses inferensi/prediksi terhadap *validation set* untuk menguji apakah model mampu mengeneralisasi permasalahan dengan baik atau tidak. Sama seperti pada tahap pelatihan model, dilakukan penentuan nilai argumen yang dipakai untuk proses inferensi. Argumen yang diatur yakni terkait dengan ukuran gambar, lokasi di mana bobot terbaik tersimpan, dan ketebalan *bounding box*. Ketebalan *bounding box* ini diatur pada nilai 1 agar hasil *bounding box* tidak saling menimpa satu sama lain.

Gambar 14. *Summary* Hasil Inferensi *Single Image*



Gambar 15. Hasil Inferensi *Single Image*

Dari Gambar 14 dan Gambar 15, dapat dilihat bahwa model mampu mendeteksi objek pada sebuah gambar dengan baik. Hal ini dapat dilihat dari *confidence score* untuk setiap *bounding box* hasil prediksi, di mana nilai untuk setiap *box* mayoritas berada di kisaran di atas 80%. Model ini bahkan mampu mendeteksi objek yang berada di pinggir gambar dan terpotong, yang mana objek-objek ini ada yang tidak terdapat pada *ground truth* yang optimal pun didapatkan ketika model memprediksi objek pada 9 gambar lain yang terdapat pada *validation set*.

IV. Kesimpulan

Pengerjaan proyek deteksi objek sel darah putih (WBCs) dan sel darah merah (RBCs) menggunakan algoritma YOLO mampu menghasilkan prediksi lokalisasi objek berupa *bounding box* beserta dengan kelas prediksinya dan *confidence score* yang sesuai harapan. *Metric* mAP yang menjadi parameter untuk melihat performa model juga dapat ditampilkan.

Kelemahan dari proyek yang dikerjakan penulis di sini yang dapat menjadi pengembangan ke depannya yaitu terkait dengan *metric* dan pemakaian bobot. Belum terdapat *metric* seperti *confusion matrix* yang dapat melihat perbandingan antara jumlah prediksi setiap kelas dengan jumlah sebenarnya. Pada bagian pemakaian bobot, akan lebih baik apabila pada proses pelatihan model menggunakan beberapa variasi nilai argumen, seperti variasi nilai *batch size* dan *epochs*.

Daftar Pustaka

[1] Habibzadeh M., Krzy˙zak A., Fevens T., “White Blood Cell Differential Counts Using Convolutional Neural Networks for Low Resolution Images,” Artificial Intelligence and Soft Computing, pp. 263–274, Aug. 2013. Accessed: Jun. 1, 2022.

[2] Acharjee S., Chakrabartty S., Alam M.I., ET AL.: "A semiautomated approach using GUI for the detection of red blood cells". Proc. Int. Conf. on Electrical, Electronics, and Optimization Techniques, 2016, pp. 525–529. Accessed: Jun. 1, 2022.

[3] Acharya V., Kumar P.: "Identification and red blood cell automated counting from blood smear images using computer-aided system", Med. Biol. Eng. Comput., 2018, **56**, (3), pp. 483–489. Accessed: May 23, 2022.

[4] J. Zhao, M. Zhang, Z. Zhou, J. Chu, and F. Cao, “Automatic detection and classification of leukocytes using convolutional neural networks,” Medical & Biological Engineering & Computing, vol. 55, no. 8, pp. 1287–1301, Nov. 2016, doi: 10.1007/s11517-016-1590-x. Accessed: May 24, 2022.

[5] J. Solawetz, “YOLOv5 New Version Explained [May 2022],” *Roboflow Blog*, Jun. 29, 2020. https://blog.roboflow.com/yolov5-improvements-and-evaluation/. Accessed May 24, 2022.

[6] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” *arXiv.org*, 2020, doi: 10.48550/arXiv.2004.10934.

[7] C.-Y. Wang, H.-Y. M. Liao, I-Hau. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, “CSPNet: A New Backbone that can Enhance Learning Capability of CNN,” *arXiv.org*, 2019, doi: 10.48550/arXiv.1911.11929.

[8] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” *Computer Vision – ECCV 2014*, pp. 346–361, 2014, doi: 10.1007/978-3-319-10578-9_23.

[9] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path Aggregation Network for Instance Segmentation,” *arXiv.org*, 2018, doi: 10.48550/arXiv.1803.01534.

[10] Kumar B., “Yolo-v5 Object-Detection Blood Cell Count and Detection,” https://github.com/bala-codes/Yolo-V5_Object_Detection_Blood_Cell_Count_and_Detection. Accessed May 24, 2022.