Implementasi *Convolutional Neural Network* untuk Mengetahui Buah Tomat yang Matang pada Pohon Tomat Menggunakan Perangkat Android

Timothy Christian Yunanto[1], Kartika Gunadi[2], Anita Nathania Purbowo[3]
Program Studi Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Surabaya, Indonesia
+6282233363253[1], +628123528487[2], +62818571336[3]
tim.chris035@gmail.com[1], kgunadi@petra.ac.id[2], anitaforpetra@gmail.com[3]

ABSTRAK

Perkembangan jaman yang serba instan sekarang ini membuat manusia ingin sesuatu yang cepat dan efisien. Seperti yang kita ketahui memetik buah tomat yang matang pada pohonnya memerlukan waktu yang lama apabila dilakukan oleh manusia. Untuk menyelesaikan permasalahan tersebut, digunakan robot otomatis yang dapat mengganti peran manusia. Untuk mendapatkan robot otomatis yang sukses diperlukan pembuatan fungsi algoritma (program) yang efisien. Program yang akan dibuat dapat dijalankan pada perangkat android.

Metode yang digunakan adalah *Blob Detection* pada *Computer Vision*, hasilnya akan diolah oleh metode Convolutional Neural Network untuk mengetahui apakah objek tersebut buah yang matang atau benda lainnya. Blob Detection digunakan untuk mendeteksi objek tomat berdasarkan mask yang diperoleh sebelumnya. Sebelum melakukan training, diperlukan pembuatan model yang berisikan layer *convolutional*, layer *max polling*, layer *flatten*, layer *dropout*, dan layer *dense*. Pengujian dilakukan dengan studi skenario dan beberapa kasus misalnya buah tomat bergerombol, buah tomat yang terpencar-pencar, buah tomat yang mask nya tidak berbentuk lonjong, dan sebagainya.

Hasil pengujian menunjukkan bahwa hasil dari CNN sangat bergantung pada hasil dari *Blob Detection* dikarenakan input dari CNN merupakan hasil dari *Blob Detection*. Apabila *Blob Detection* gagal mendapatkan objek tomat, maka CNN tidak akan berjalan. Hasil pengujian menunjukkan bahwa *Blob Detection* akan gagal mendeteksi objek tomat apabila tomat terhalang oleh benda lain yang mengakibatkan bentuk mask dari objek tersebut menjadi kacau. Hasil pengujian dari CNN juga menunjukkan nilai akurasi dari training sebesar 96% dan akurasi testing sebesar 93%.

Kata Kunci: *Convolutional Neural Network, Blob Detection*, Kematangan Tomat, *Tensorflow, Keras*.

ABSTRACT

The development of instant era makes people want something that fast and efficient. As we know, picking ripe tomatoes on the tree requires a long time if done by humans. To solve these problems, automatic robots are used that can replace the role of humans. To get a successful automated robot requires the creation of efficient algorithm function (program). The Program can be run on an Android Device.

We use Blob Detection method on Computer Vision, and the result will be processed by the Convolutional Neural Network method. CNN method requires to determine whether the object is ripe tomatoes or other objects. Blob Detection is used to detect tomato objects based on previously obtained masks. Before doing the training, it is necessary to make a model that contains convolutional layer, max polling layer, flatten layer, dropout layer, and dense layer. The test is carried out with a scenario study and several cases such as bunched tomatoes, scattered tomatoes, tomatoes whose masks are not oval, and so on.

The results show that the results of CNN are very dependent on the results of Blob Detection because the input from CNN is from the result of Blob Detection. If Blob Detection fails to get the tomato object, CNN will not run properly. The results show that Blob Detection will fail to detect the tomato object if the tomato is blocked by another object which causes the mask shape of the object to be chaotic. The test results from CNN also showed an accuracy value of training of 96% and testing accuracy of 93%.

Keywords: *Convolutional Neural Network, Blob Detection, Mature Tomato, Ripe Tomato, Tensorflow, Keras*.

1. PENDAHULUAN

Perkembangan jaman yang serba instan sekarang ini membuat manusia ingin sesuatu yang cepat dan efisien. Kita sebagai manusia sering kali merasa tidak puas dengan hasil yang telah diperoleh. Semisal dalam memetik buah tomat yang matang pada pohonnya, seperti yang kita ketahuni memetik buah tomat yang matang pada pohonnya memerlukan waktu yang lama apabila dilakukan oleh tenaga kerja manusia [5]. Untuk menyelesaikan permasalahan tersebut, digunakan robot otomatis yang dapat mengganti peran kerja manusia.

Robot otomatis merupakan mesin pintar yang dapat melakukan tugas tanpa campur tangan yang intensif dari manusia. Untuk mendapatkan robot otomatis yang sukses, diperlukan sistem yang dapat meningkatkan efisiensi. Tahapan yang penting dalam membuat robot pemanen buah tomat secara otomatis adalah pembuatan fungsi algoritma pendeteksian yang berhasil [4]. Algoritma tersebut dapat dirancang dengan memanfaatkan fitur yang spesifik dari objek tersebut seperti bentuk, warna, ukuran, kebulatan, dan lain-lain [4].

Aplikasi yang akan dibuat akan menjawab permasalahan yang sudah disebutkan diatas yakni pembuatan algoritma aplikasi yang efisien dalam menentukan buah tomat yang matang pada pohonnya. Dengan memanfaatkan Blob Detection pada Computer Vision (CV) akan didapatkan data pada buah tersebut yang kemudian akan diolah oleh metode *Convolutional Neural Network* untuk mengetahui kematangannya. Algoritma Blob digunakan

untuk menemukan dan menghitung objek, dan untuk membuat pengukuran dasar karakteristik [2]. *Convolutional Neural Network* (CNN) merupakan salah satu tipe dari *Artificial Intelligence* yang menggunakan algoritma *Deep Learning* yang dapat mengambil *input* gambar dan dapat mengklasifikasi suatu objek [6]. Tugas utama CNN dalam *Image Classification* adalah menerima *input* gambar dan diikuti dengan pemaknaan gambar tersebut (memberi arti dan definisi) [7]. Sebelum melakukan uji coba, aplikasi akan di training terlebih dahulu dengan beberapa sample buah tomat yang sudah matang dan buah tomat yang masih mentah serta komonen lainnya yang bukan merupakan tomat matang. Setelah di training, barulah aplikasi siap untuk diuji coba. Aplikasi yang akan dibuat untuk skripsi ini akan menggunakan perangkat Android.

Android merupakan sistem operasi mobile dengan jumlah pengguna terbanyak. Android memiliki market share sebesar 39,16% meninggalkan Windows Desktop yang hanya 38.04% [8]. Meningkatnya populasi pengguna Android didorong oleh ekosistem aplikasi pada Android yang mudah dimana developer dapat dengan mudah memasukkan aplikasinya ke dalam Google PlayStore. Selain itu di era serba cepat dan instan saat ini membuat kebanyakan orang lebih sering menggunakan mobile daripada desktop dengan alasan lebih praktis untuk dibawa-bawa. Maka dari itu dengan membuat aplikasi pada Android dapat digunakan oleh lebih banyak pengguna.

2. TINJAUAN STUDI

Pada percobaan pada implementasi metode *Improved* HSV berhasil menghilangkan *background* yang kompleks meskipun terdapat banyak *noise*. Sedangkan metode *Watershed Algorithm* berhasil memisahkan tomat yang bersinggungan dan mendeteksi buah tomat mana yang matang. Hasil dari penggabungan 2 metode tersebut menghasilkan angka akurasi yang baik (sekitar 81.66%) [5].

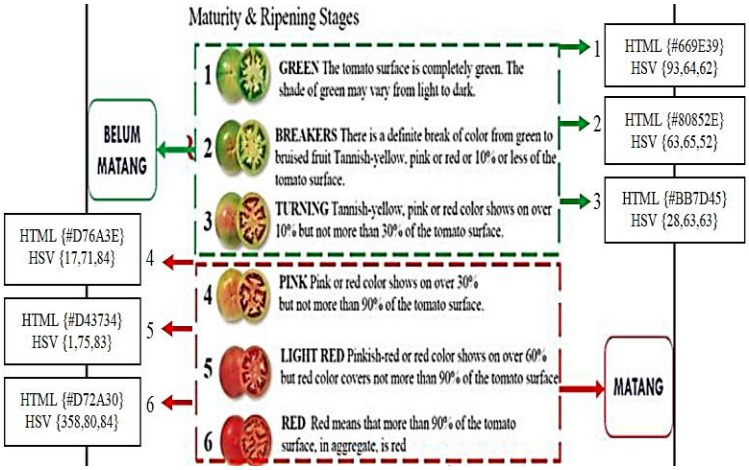
Percobaan lain menunjukkan metode *Faster R-Convolutional Neural Network* dengan implementasi oleh ZFNet dapat memperoleh hasil akurasi yang baik dalam mendeteksi buah kiwi (sekitar 92.3%) [3]. Percobaan lain juga menunjukkan menggunakan metode *Deep Convolutional Neural Network* menghasilkan nilai akurasi yang baik (sekitar 82.61%) dalam mendeteksi tingkat kematangan buah *strawberry* [4].

3. DESAIN SISTEM

Proses dalam pembuatan sistem meliputi *pre-processing*, mengidentifikasi objek tomat dengan *blob detector*, pembuatan model untuk *Convolutional Neural Network*, *training* dan *testing* data dari model yang telah dibuat, dan desain *user interface*. Terdapat *back end* dan *front end* dalam pembuatan sistem, *back end* bertugas untuk memproses gambar yang diupload dari *front end.* Sedangkan *front end* bertugas untuk mengambil gambar, mengirimkannya kedalam server, dan menampilkan gambar yang telah diproses dari server.

3.1 *Pre-processing*

Gambar yang telah di upload dari *front end*, akan di resize terlebih dahulu, setelah itu gambar akan dikonversi dari BGR *color space* menjadi HSV *color space*. Kemudian diikuti dengan menentukan *range* HSV untuk *dark red, red, orange, light orange*, dan *purple*. Penentuan *range* HSV tersebut sebagai bahan dasar untuk menjalankan fungsi inRange pada cv2 dalam menangkap *mask* dari tiap warna yang telah ditentukan sebelumnya. *Mask-mask* dari tiap warna akan digabungkan menjadi 1. Setelah itu, *mask* yang sudah diperoleh akan di *median filter* yang berguna untuk mengurangi *noise* yang tidak diperlukan. Gambar 1 merupakan tingkat kematangan buah tomat dalam warna.



Gambar 1. Tingkat Kematangan Buah Tomat [1]

3.2 *Blob Detector*

Pada proses ini diawali dengan menkonfigurasi *parameter* SimpleBlobDetector, dan dilanjutkan dengan menetapkan *minimal* serta maksimal *threshold*. Setelah itu proses dilanjutkan dengan mengatur agar *filter by circularity* menjadi aktif dan mengatur *filter by color* menjadi aktif. *Filter by color* diaktifkan dengan tujuan untuk mendeteksi titik hitam yang timbul pada objek putih. Objek putih merupakan mask tomat yang telah didaaptkan sedangkan titik hitam timbul karena adanya sinar matahari pada objek tomat. Setelah itu, proses dilanjutkan dengan membuat *detector* yang berfungsi untuk mendeteksi titik hitam dan menyimpan lokasi dari titik hitam. Lokasi dari titik hitam tersebut tersimpan dalam bentuk keypoint. Keypoint tersebut akan diproses untuk mendapatkan koordinat titik pusat (X,Y). Selain titik pusat, keypoint akan di proses juga untuk mendapatkan size (diameter) dari titik hitam. Setelah kedua hal itu didapatkan, dilanjutkan dengan proses menambal titik hitam yang ada dengan cara menggambar lingkaran putih solid pada titik pusat dan diameter dari titik hitam.

Setelah itu proses dilanjutkan dengan menkonfigurasi ulang *parameter* SimpleBlobDetector yang sebelumnya dengan menonaktifkan *filter by color*. Hal ini diperlukan untuk mendeteksi objek yang berwarna putih (yaitu objek tomat) yang telah di *mask* pada proses sebelumnya. Setelah *parameter* dikonfigurasi ulang, dilanjutkan dengan membuat *detector* yang baru sesuai dengan konfigurasi yang baru. Setelah itu *detector* berfungsi untuk mendeteksi objek tomat pada gambar dan menyimpan lokasi dari objek tomat. Lokasi dari objek tomat tersimpan dalam bentuk keypoint. Keypoint yang didapatkan kemudian diproses agar mendapatkan titik pusat (X,Y) dan diameter dari objek tomat. Kemudian dilakukan pemotongan objek tomat sesuai dengan keypoint yang telah di proses.

3.3 Pembuatan Model CNN

Pembuatan model diawali dengan melakukan import library keras pada tensorflow. Kemudian dilanjutkan dengan membuat model secara sequential. Setelah itu barulah dibuat layer-layer yang dibutuhkan yaitu layer Convolutional2D, layer Max Polling, layer Convolutional2D, layer Max Polling, layer Convolutional2D, layer Max Polling, layer Convolutional 2D, layer Dropout, layer Flatten, layer Dropout, dan diakhiri dengan layer Dense. Setelah itu barulah model tersebut dapat di compile.

3.4 *Training* dan *Testing*

Proses *training* diawali dengan *import library keras* pada *tensorflow*. Kemudian dilanjutkan dengan *load* semua gambar dengan kategori "matang" beserta resize menjadi ukuran 100x100px dan *convert* dari BGR *color space* menjadi RGB *color space*. Tujuan dari *resize* adalah agar semua gambar untuk *training* dan *testing* memiliki ukuran yang sama sebagai syarat utama untuk CNN yaini memiliki ukuran yang sama antara data training dengan data testing. Setelah itu RGB *value* pada masing-masing gambar akan dinormalisasi dari *range* 0-255 menjadi 0-1 dengan tujuan agar CNN dapat memproses gambar tersebut dengan mudah. Proses *load* gambar diikuti dengan *load label*. Ketika semua gambar kategori "matang" telah di *load*, gambar di *shuffle* terlebih dahulu sebanyak 3x dengan tujuan agar urutan gambar menjadi *random* barulah gambar displit antara untuk *training* dan untuk *testing* (75:25). Setelah di-*split*, proses dilanjutkan dengan *load* semua gambar dengan kategori "other" dan diikuti dengan proses yang sama seperti kategori "matang".

Setelah semua data gambar telah di *load*, dilanjutkan dengan mengubah array *train* dan array *test* menjadi *numpy array images*. Setelah itu dilakukan proses verifikasi untuk memastikan data gambar beserta label yang di *load* telah benar. Apabila sudah benar, proses *load* data gambar beserta label telah selesai dan dapat dilanjutkan untuk proses *training* dan *testing*.

Setelah verifikasi dan data sudah benar, proses *training* dapat dilakukan. Diawali dengan menambah dimensi dari label agar array yang semula 1 dimensi berubah menjadi 2 dimensi. Hal ini diperlukan agar proses *fitting* untuk *train* dapat berjalan dengan baik. Setelah itu, proses *fitting* pada model untuk training data dapat dilakukan dan diikuti dengan evaluasi *testing* data untuk mengetahui tingkat akurasi dari model yang telah di *train*. Setelah itu, model dapat di simpan pada *hard disk* untuk digunakan pada saat akan melakukan klasifikasi kematangan buah tomat.

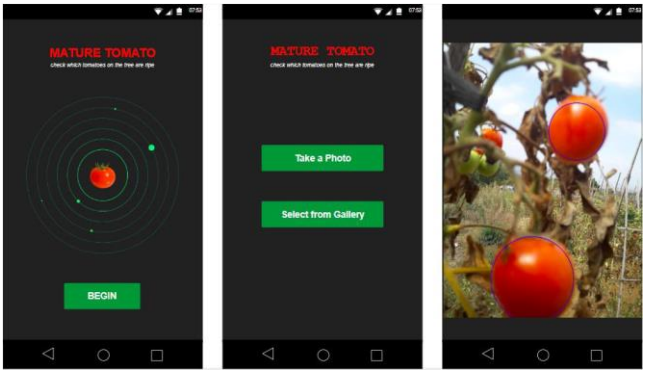
Model yang telah di simpan akan di *load* terlebih dahulu. Setelah itu dilanjutkan dengan *load* semua gambar tomat dan lainnya yang telah dipotong pada proses sebelumnya kedalam array. Dilanjutkan dengan menkonversi gambar dari BGR *color space* menjadi RGB *color space*. Setelah di konversi dilanjutkan dengan me-*resize* ukuran dari gambar menjadi 100x100px. *Resize* ini bertujuan agar ukuran gambar yang akan di tes dengan gambar yang telah di train didalam model memiliki ukuran yang sama. Setelah itu dilanjutkan dengan melakukan normalisasi pada RGB *value* dari 0-255 menjadi 0-1. Setelah itu barulah dapat diperoleh prediksi dari tiap gambar berdasarkan model *train* yang telah di *load*. Apabila hasil dari prediksi menyatakan bahwa objek tersebut "matang" maka keypoint dari objek tersebut akan tetap di *keep*, tetapi apabila hasil prediksi menyatakan bahwa objek tersebut "other" maka keypoint dari objek tersebut akan di-*delete*. Setelah itu dilanjutkan dengan menandai pada gambar yang asli hasil dari keypoint yang telah di sortir. Proses penanda dilakukan dengan menggambar lingkaran pada gambar berdasarkan keypoint yang telah di sortir.

3.5 *User Interface*

Gambar 2 menunjukkan *interface* pada aplikasi Mature Tomato. Halaman awal yang akan dibuat adalah halaman home yang berisikan judul program, logo, beserta button untuk memulai. Setelah itu, *button* diarahkan ke halaman berikutnya yang berisikan *form input file*.

Pada halaman kedua, berisikan form untuk input file foto/gambar. Form tersebut dapat membuka *browse file* pada Android dan juga mengakses *camera* pada android. Setelah itu, sebuah button disiapkan untuk interaksi user bahwa gambar siap untuk di proses dan diupload ke dalam server. Di server, data gambar akan diolah dengan menjalankan *script* python yang ada di dalam file php. Hasil dari olahan gambar tersebut akan disimpan dan bersiap untuk halaman berikutnya.

Setelah halaman *section* selesai, dapat dilanjutkan dengan halaman *result*. Pada halaman *result* akan menampilkan gambar yang telah di proses dari server. Gambar yang disimpan pada server akan di download dan ditampilkan pada layar *Handphone*.



Gambar 2. Desain User Interface

4. PENGUJIAN SISTEM

Pengujian ini akan dilakukan dengan melakukan skenario dan kasus pada beberapa contoh. Pengujian ini bertujuan untuk melihat sejauh mana akurasi dari metode *Blob Detection*, metode *Convolutional Neural Network*, dan gabungan kedua metode tersebut. Hasil dari akurasi tersebut dapat menjadi acuan dari metode *Blob Detection* sampai sejauh mana, dari metode *Convolutional Neural Network* sampai sejauh mana, dan gabungan dari kedua metode tersebut sampai sejauh mana. Hasil untuk akurasi ini didapatkan dari rumus (1).

$$Akurasi\ 1 = \frac{Jumlah\ tomat\ matang\ yang\ berhasil\ terdeteksi}{Total\ jumlah\ tomat\ matang} \times 100$$

$$Akurasi\ 2 = \frac{Jumlah\ objek\ tomat\ matang\ yang\ terdeteksi}{Total\ jumlah\ objek\ yang\ terdeteksi} \times 100$$

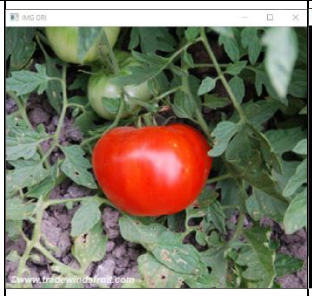
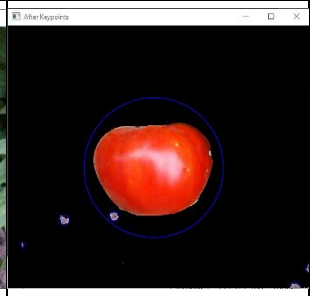
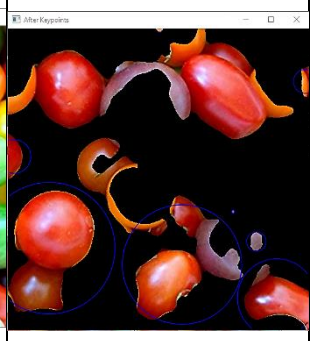
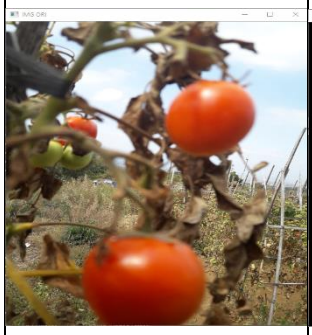
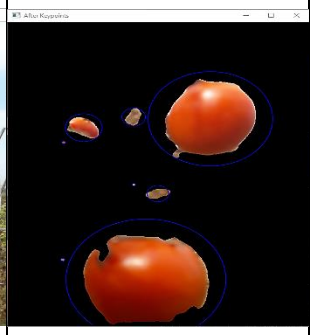
$$Akurasi\ rata-rata = \frac{Akurasi\ 1 + Akurasi\ 2}{2} \quad \ldots\ldots\ldots\ldots (1)$$

4.1 Akurasi *Blob Detection*

Tabel 1. Hasil *Blob Detection*

No	Percobaan	Hasil
1		

Tabel 1. Hasil *Blob Detection* (lanjutan)

No	Percobaan	Hasil
2		
3		
4		
5		

Tabel 2. Akurasi *Blob Detection*

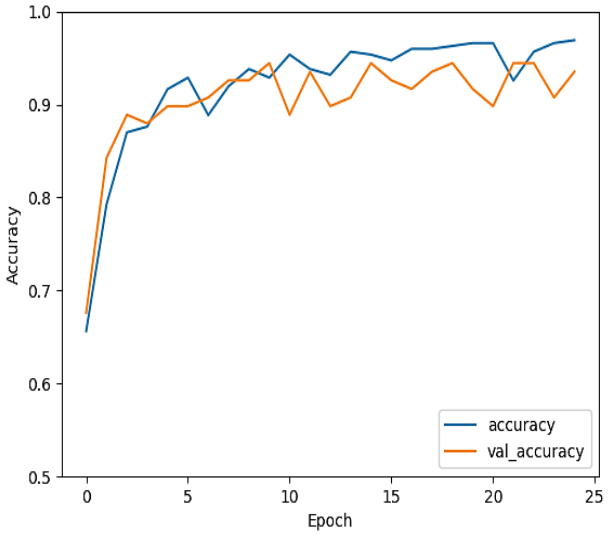
Percobaan	Akurasi 1	Akurasi 2	Rata-rata
1	(12/13) × 100 = 92%	(5/5) × 100 = 100%	(92+100) / 2= 96%

Tabel 2. Akurasi *Blob Detection* (lanjutan)

Percobaan	Akurasi 1	Akurasi 2	Rata-rata
2	(13/13) × 100 = 100%	(5/5) * 100 = 100%	(100+100) / 2 = 100%
3	(1/1) * 100 = 100%	(1/6) * 100 = 17%	(100+17) / 2 = 59%
4	(5/11) * 100 = 45%	(4/8) * 100 = 50%	(100+38) / 2 = 48%
5	(3 / 3) * 100 = 100%	(3/8) * 100 = 38%	(100+38) / 2 = 69%

4.2 Akurasi *Convolutional Neural Network*

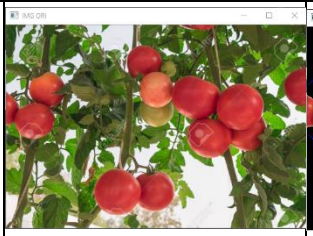
Pengambilan nilai test akurasi dan loss didapatkan dari fungsi model.evaluate yang terdapat dalam *Library* Keras. Pada Gambar 3 merupakan hasil akurasi dari model yang dibuat dan disimpan untuk digunakan dalam mengenali tomat matang atau lainnya pada CNN. Garis biru merupakan hasil akurasi dari train data sedangkan garis orange merupakan hasil akurasi dari test data. Pada epoch yang pertama menunjukkan akurasi yang masih rendah (di bawah 70%) dan bergerak naik menjadi lebih baik. Pada epoch yang terakhir yaitu 25, train data mempunyai nilai akurasi sebesar 0,9690 (96%) sedangkan test data mempunyai nilai akurasi sebesar 0,9352 (93%) dengan loss sebesar 0, 2101.



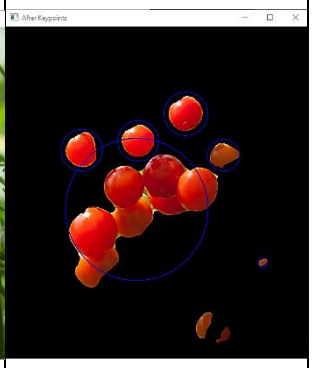
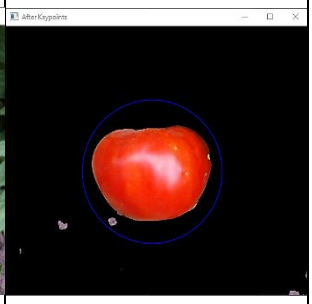
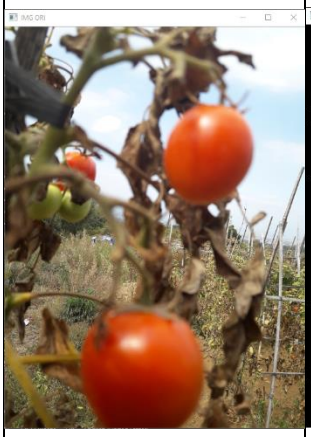
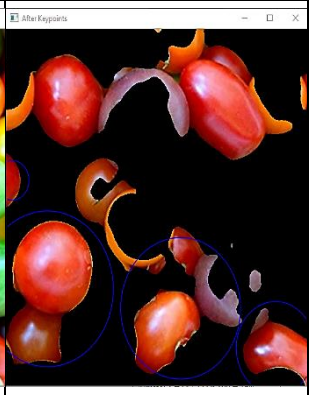
Gambar 3. Hasil Akurasi *Train* dan *Test* pada CNN

4.3 Akurasi *Blob Detection* + CNN

Tabel 3. Hasil *Blob Detection* + CNN

No	Percobaan	Hasil
1		

Tabel 3. Hasil *Blob Detection* + CNN (lanjutan)

No	Percobaan	Hasil
2		
3		
4		
5		

Tabel 4. Akurasi *Blob Detection* + CNN

Percobaan	Akurasi 1	Akurasi 2	Rata-rata
1	(12 / 13) * 100 = 92%	(5/5) * 100 = 100%	(92+100) / 2 = 96%
2	(11 / 13) * 100 = 85%	(5/5) * 100 = 100%	(85+100) / 2 = 93%
3	(1 / 1) * 100 = 100%	(1/1) * 100 = 100%	(100+100) / 2 = 100%
4	(3 / 3) * 100 = 100%	(3/3) * 100 = 100%	(100 + 100) / 2 = 100%
5	(5 / 11) * 100 = 45%	(4/4) * 100 = 100%	(45 + 100) / 2 = 73%

5. KESIMPULAN DAN SARAN

Berdasarkan hasil pengamatan dan pengujian pada sistem, dapat disimpulkan beberapa hal sebagai berikut:

- Berdasarkan hasil pengujian pada sistem dapat disimpulkan bahwa CNN sangat bergantung pada hasil dari *blob detector* dalam mengenali objek tomat karena input dari CNN berasal dari hasil *blob detector*.
- Berdasarkan hasil pengujian dapat disimpulkan bahwa apabila menggunakan foto buah-buahan yang menyerupai warna tomat, dapat membuat hasil akurasi menurun dikarenakan mask yang dihasilkan kacau.
- Berdasarkan hasil pengujian dapat disimpulkan bahwa mask yang menghasilkan titik hitam, setelah dilakukan penambalan membuat blob detection dapat mengidentifikasi objek tersebut sehingga CNN dapat mengetahui objek tersebut adalah tomat matang.
- Berdasarkan hasil pengujian dapat disimpulkan bahwa posisi buah tomat yang tersebar tidak mempengaruhi kinerja *blob detection* dan juga CNN.
- Berdasarkan hasil pengujian dapat disimpulkan bahwa *blob detection* dapat gagal dalam mendeteksi buah tomat apabila hasil *mask* kacau.
- Berdasarkan hasil pengujian dapat disimpulkan bahwa nilai akurasi dengan penggabungan metode menghasilkan akurasi yang lebih baik dari pada *Blob Detection* saja.

Dengan adanya kesimpulan, ada beberapa hal yang dapat dijadikan sebagai saran dalam pengembangan untuk selanjutnya antara lain:

- Pengambilan gambar pohon tomat tidak terlalu jauh agar informasi penting dari pohon tomat dapat diperoleh.
- Untuk menghindari blob detection gagal mendeteksi tomat yang terhalang benda lain, lebih baik pohon tomat difoto dari berbagai sudut.

6. REFERENCES

[1] Anggriawan, M.A., Ichwan, M., & Utami, D.B. 2017. Pengenalan Tingkat Kematangan Tomat Berdasarkan Citra Warna pada Studi Kasus Pembangunan Sistem Pemilihan Otomatis. Jurnal Teknik Informatika dan Sistem Informasi, December 2017 (pp. 550-564). No. 3. Vol (3). Jakarta, Indonesia.

[2] *Blob Detection.* 2016. Retrieved from http://www.discoversdk.com/blog/blob-detection

[3] Fu, L., Feng, Y., Majeed, Y., Zhang, X., Zhang, J., Karkee, M., & Zhang, Q. 2018. *Kiwifruit detection in field images using faster R-CNN with ZFNet: Proceedings of the Sixth International Federation of Automatic Control Conference on Bio-Robotics BIOROBOTICS*, 13-15 July 2018 (pp. 45-50), Beijing, China: PapersOnLine

[4] Habaragamuwa, H., Ogawa, Y., Suzuki, T., Shiigi, T., Ono, M., & Kondo, N. 2018. Engineering in Agriculture, Environment and Food. *Detecting greenhouse strawberries (mature and immature), using deep convolutional neural network*,11(3), 127-138. doi:10.1016/j.eaef.2018.03.001

[5] Malik, M., Zhang, T., Li, H., Zhang, M., Shabbir, S., & Saeed, A. 2018. *Mature tomato fruit detection algorithm based on improved HSV and watershed algorithm: Proceedings of the Sixth International Federation of Automatic Control Conference on Bio-Robotics BIOROBOTICS*, 13-15 July 2018 (pp. 431-436), Beijing, China: PapersOnLine

[6] *Medium: Can a simple CNN work as well as Facial Recognition for differentiating Redheads?*. 2018. Retrieved from https://towardsdatascience.com/can-a-simple-cnn-work-as-well-as-facial-recognition-for-differentiating-redheads-18596b05fdec

[7] *SearchEnterpriseAI: Convolutional Neural Network.* 2018. Retrieved from https://searchenterpriseai.techtarget.com/definition/convolutional-neural-network.

[8] *StatCounter GlobalStats: Desktop & Mobile Operating System Market Share Worldwide May 2018 - May 2019*. 2019. Retrieved from http://gs.statcounter.com/os-market-share/desktop-mobile/worldwide/#month ly-201805-201905-bar.