PERANCANGAN APLIKASI PERHITUNGAN JUMLAH POHON MANGGA BERBASIS *DEEP LEARNING* MEMANFAATKAN *LOW-ALTITUDE REMOTE SENSING*

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan Program Studi Teknik Pertanian (S1)
dan mencapai gelar Sarjana Teknik

Oleh
Maheqsa Alamsyah Sutikno Putra
NIM 161710201098

JURUSAN TEKNIK PERTANIAN
FAKULTAS TEKNOLOGI PERTANIAN
UNIVERSITAS JEMBER
2020

PERANCANGAN APLIKASI PERHITUNGAN JUMLAH POHON MANGGA BERBASIS *DEEP LEARNING* MEMANFAATKAN *LOW-ALTITUDE REMOTE SENSING*

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan Program Studi Teknik Pertanian (S1)
dan mencapai gelar Sarjana Teknik

Oleh
Maheqsa Alamsyah Sutikno Putra
NIM 161710201098

JURUSAN TEKNIK PERTANIAN
FAKULTAS TEKNOLOGI PERTANIAN
UNIVERSITAS JEMBER
2020

PERSEMBAHAN

Skripsi ini saya persembahkan sebagai rasa terima kasih saya yang tidak terkira kepada:

1. Kedua orang tua, Bapak Buyung Sutikno dan Ibu Umi Sri Wulan yang telah mendidik, membimbing, mendoakan, memberikan dukungan, dan pengorbanan yang tidak terhingga.
2. Kakak saya, Dea Mayella Putri Sutikno, Nonny Rulisty Putri Sutikno, dan Rian Bagus Danang Permadi yang telah mendukung dan mendoakan selama penyusunan skripsi ini.
3. Almamater tercinta Fakultas Teknologi Pertanian Universitas Jember.

MOTO

"Maka sesungguhnya bersama kesulitan ada kemudahan. Sesungguhnya bersama kesulitan ada kemudahan. Maka apabila engkau telah selesai (dari suatu urusan), tetaplah bekerja keras (untuk urusan yang lain). Dan hanya kepada Tuhanmulah engkau berharap."
(QS. Al Insyirah: 6-8) *)

"Siapa yang menghendaki kehidupan dunia, maka harus disertai dengan ilmu. Dan siapa yang menghendaki kehidupan akhirat, juga harus dengan ilmu."
(Imam Syafi'i) **)

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

nama : Maheqsa Alamsyah Sutikno Putra

NIM : 161710201098

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "*Perancangan Aplikasi Perhitungan Jumlah Pohon Mangga Berbasis Deep Learning Memanfaatkan Low-Altitude Remote Sensing*" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi manapun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak mana pun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, 04 Oktober 2020

Yang menyatakan,

Maheqsa Alamsyah S. P.

NIM 161710201098

SKRIPSI

PERANCANGAN APLIKASI PERHITUNGAN JUMLAH POHON MANGGA BERBASIS *DEEP LEARNING* MEMANFAATKAN *LOW-ALTITUDE* REMOTE SENSING

Oleh

Maheqsa Alamsyah Sutikno Putra
NIM 161710201098

Dosen Pembimbing

Bayu Taruna Widjaja Putra, S.TP., M.Eng., Ph.D.

PENGESAHAN

Skripsi berjudul "Perancangan Aplikasi Perhitungan Jumlah Pohon Mangga Berbasis *Deep Learning* Memanfaatkan *Low-Altitude Remote Sensing*" karya Maheqsa Alamsyah Sutikno Putra telah diuji dan disahkan pada:

hari, tanggal : 04 Oktober 2020

tempat : Fakultas Teknologi Pertanian Universitas Jember

Menyetujui,

Dosen Pembimbing Utama

Bayu Taruna Widjaja Putra, S.T.P., M.Eng., Ph.D.
NIP. 198410082008121002

Tim Penguji,

Ketua	Anggota
Prof. Dr. Indarto, S.T.P., DEA.	Dr. Siswoyo Soekarno, S.T.P.M.Eng.
NIP 197001011995121001	NIP 196809231994031009

Mengesahkan,

Dekan Fakultas Teknologi Pertanian

Universitas Jember

Dr. Siswoyo Soerkarno, S.TP., M.Eng.
NIP 196809231994031009

RINGKASAN

Perancangan Aplikasi Perhitungan Jumlah Pohon Mangga Berbasis Deep Learning Memanfaatkan Low-Altitude Remote Sensing; Maheqsa Alamsyah Sutikno Putra, 161710201098; 2020; 97 halaman; Jurusan Teknik Pertanian Fakultas Teknologi Pertanian Universitas Jember.

Tanaman mangga *(Magnifera indica L.)* adalah tanaman buah tahunan berupa pohon yang biasa tumbuh baik didaerah beriklim kering. Menurut Badan Pusat Statistik (BPS), pada tahun 2017, produksi mangga di Kabupaten Situbondo sendiri tercatat memiliki area tanam seluas 601 Ha mencapai 16.530,6 ton. Melihat data tersebut diperlukan teknologi informasi untuk pemantauan lahan perkebunan mangga untuk menjaga produksivitas mangga semaksimal mungkin. Salah satunya dengan cara merancang aplikasi perhitungan jumlah pohon mangga secara otomatis dengan memanfaatkan gambar dari ketinggian tertentu. *Remote sensing* dapat digunakan untuk mendapatkan gambaran kondisi lahan secara cepat dan akurat. Data *remote sensing* dapat diperoleh dari *drone* karena memiliki resolusi spasial yang tinggi sehingga kesalahan dalam identifikasi suatu objek di lapangan dapat diminimalisir. Hasil data yang diperoleh dari *drone* dapat digunakan untuk perhitungan jumlah pohon secara otomatis dengan memanfaatkan metode *Deep Learning* dengan *neural network* yang dinamakan *Faster-RCNN*.

Faster-RCNN merupakan salah satu algoritma *deep learning* yang banyak dikembangkan untuk *computer vision* karena tingginya akurasi yang didapatkan. Arsitektur *Faster R-CNN* terdiri dari *RPN, dan Faster R-CNN Detector*. Penelitian ini menggunakan *Faster-RCNN* sebagai algoritma untuk mendeteksi dan menghitung jumlah pohon mangga dengan memanfaatkan dua model yaitu *Inception V2* dan *ResNet-50*. Kedua model tersebut digunakan untuk proses *training* data. namun sebelum dilakukannya proses *training* data, input data yaitu yang berupa gambar perlu dilakukannya proses *pre-processing* image yang meliputi indeks vegetasi, *crop*, *resize*, augmentasi, dan anotasi. Data dibandingkan berdasarkan data tanpa indeks vegetasi yaitu RGB dan data dengan menggunakan indeks vegetasi yaitu VARI, GRVI, dan GNDVI.

Hasil *training* data dapat dilihat berdasarkan beberapa aspek yaitu *learning rate*, *global* step, dan *total loss*. Setelah dilakukannya proses *training* data, kemudian data dianalisis menggunakan uji *confusion matrix* dengan menggunakan parameter *precision*, *recall*, *accuracy*, dan *error*. Berdasarkan hasil yang didapat menggunakan uji *confusion matrix*. Data dengan nilai tertinggi didapatkan oleh data RGB dengan nilai *precision* sebesar 97,62 %, *recall* sebesar 74,55 %, *accuracy* 73,21 %, dan *error* 26,79 % pada model *Inception V2*. Sedangkan pada model *ResNet-50* didapatkan nilai *precision* sebesar 95,45 %, *recall* sebesar 95,45 %, *accuracy* 91,30 %, dan *error* 8,70 % pada model *ResNet-50*. Hasil model terbaik didapatkan oleh model *ResNet-50* dengan rata-rata nilai *precision* sebesar 96,84 %, *recall* sebesar 65,71 %, *accuracy* 65,04 %, dan *error* 34,96 %.

SUMMARY

Design Of Mango Tree Counting Application Based On Deep Learning With Low-Altitude Remote Sensing; Maheqsa Alamsyah Sutikno Putra, 161710201098; 2020; 97 pages; Department of Agricultural Engineering, Faculty of Agricultural Technology, University of Jember.

Mango plants (Mangifera indica L.) are annual fruit plants consisting of trees that normally grow well in dry climates. According to Badan Pusat Statistik (BPS), in 2018, mango production in Situbondo Regency itself recorded reaching 16.530,6 tons while Situbondo Regency has a planting area of 601 hectares. According to this data, information technology is needed to monitor mango plantation for maintaining the maximum production of mangoes. One of the ways is by designing the application to count the number of mango trees automatically using images from a certain height. Remote sensing can be used to get a quick and accurate picture of land conditions. Remote sensing data can be obtained from drones because it has a higher spatial resolution so the problems in field objects can be minimized. The results data from the drones can be used to count the number of trees automatically by using the Deep Learning method with a neural network called Faster-RCNN.

Faster-RCNN is one of the deep learning algorithms which is widely developed for computer vision because obtained high accuracy. The R-CNN Faster Architecture consists of RPN dan Faster R-CNN Detector. This research used Faster-RCNN as an algorithm to detect and count the number of mango trees using two models called Inception V2 and ResNet-50. Those second models were used to process *training* data. However, before processing the *training* data, input the data which was in the form of images that needed a pre-processing image process that includes a vegetation index, cropped, resized, augmented, and annotated. Data compared based on data without vegetation index, namely RGB and data using vegetation index, called VARI, GRVI, and GNDVI.

The results of the *training* data can be seen with several aspects which are the learning rate, global steps, and total *loss*. After trained the data and get the result, the data will analyzed using the confusion matrix test by some parameters which is precision, recall, accuracy, and error. The results of data analyzed is data with the highest result obtained by RGB data with the precision value of 97,62 %, recall in the amount of 74,55 %, accuracy was 73,21 %, and error was 26,79 % in the Inception V2 model. Whereas in the ResNet-50 model the precision value was 95,45 %, recall was 95,45%, accuracy was 91,30 %, and error was 8,70 %. The best model results obtained by the ResNet-50 model with an average precision value of 96,84 %, recall in the amount of 66,71%, accuracy was 65,04 %, and error was 34,96 %.

PRAKATA

Puji syukur ke hadirat Allah SWT. atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "*Perancangan Aplikasi Perhitungan Jumlah Pohon Mangga Berbasis Deep Learning Memanfaatkan Low-Altitude Remote Sensing*". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan strata satu (S1) pada Jurusan Teknik Pertanian Fakultas Teknologi Pertanian Universitas Jember.

Penyusunan skripsi ini, tidak lepas dari bantuan dari berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Kedua orang tua penulis, yang telah memberi semangat, motivasi, dan doa untuk penulis;
2. Bayu Taruna Widjaja Putra, S.TP., M.Eng., Ph.D. selaku Dosen Pembimbing Utama yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi ini;
3. Dr. Elida Novita, S.TP., M.T. selaku Dosen Pembimbing Akademik yang telah membimbing selama penulis menjadi mahasiswa;
4. Rufiani Nadzirah, S.TP., M.Sc., selaku Komisi Bimbingan yang telah memberikan arahan dan dorongan dalam penyelesaian skripsi ini;
5. Seluruh dosen pengampu mata kuliah yang telah membimbing penulis selama menimba ilmu di Fakultas Teknologi Pertanian Universitas Jember;
6. Seluruh staf dan karyawan di lingkungan Fakultas Teknologi Pertanian Universitas Jember, terima kasih atas bantuan dalam mengurus administrasi dan lainnya;
7. Sahabat-sahabat seperjuangan, Bacrul Ulum, dan Yahya Sultoni, yang telah memberikan motivasi dan dukungan kepada penulis;
8. Sahabat-sahabat dari "Geng Persegi", Qoniatul Habibah, Ruli Nur Fajri, Viandra Yashinta, yang senantiasa memberikan dukungan dan menemani saya mengerjakan skripsi ini hingga selesai;
9. Keluarga besar Teknik Pertanian 2016 khususnya TEP-B yang telah berbagi manis dan pahit bersama serta memberikan dukungan kepada penulis;

10. Pihak-pihak lain yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan. Oleh karena itu, kritik dan saran yang bersifat membangun sangat penulis harapkan demi kesempurnaan skripsi ini. Akhirnya penulis berharap agar skripsi ini dapat bermanfaat bagi semua.

Jember, 04 Oktober 2020
Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Tanaman mangga *(Magnifera indica* L.*)* adalah tanaman buah tahunan berupa pohon yang biasa tumbuh baik di daerah beriklim kering seperti di Indonesia. Menurut Badan Pusat Statistik (BPS), pada tahun 2017 produksi mangga di Indonesia mencapai 2.203.793 ton dengan luas area tanam 201.080 ha, sedangkan produksi mangga di Kabupaten Situbondo sendiri tercatat memiliki area tanam seluas 601 ha mencapai 16.530,6 ton. Data tersebut menunjukkan bahwa Kabupaten Situbondo merupakan salah satu penghasil buah mangga yang berkontribusi besar dalam menunjang perekonomian nasional Indonesia.

Besarnya tingkat produktivitas mangga di Kabupaten Situbondo perlu didukung dengan pemeliharaan atau perawatan tanaman untuk meningkatkan produktivitas mangga seperti pemberian pupuk, pengendalian OPT (organisme pengganggu tanaman), pemangkasan tajuk, dan penjarangan buah. Namun dengan lahan yang luas dan jumlah pohon mangga yang banyak membuat para petani perlu untuk melakukan survei lapang untuk mengumpulkan data berupa jumlah pohon yang membutuhkan pemeliharaan. Hal tersebut dapat memperlambat proses kegiatan pemeliharaan pohon mangga sehingga berpotensi untuk menyebabkan perencanaan pengelolaan pertanian tidak dapat dilakukan tepat sasaran dan bijaksana. Inovasi lewat teknologi dapat menjadi kunci dalam pengolahan data perkebunan mangga sehingga mempermudah pemilik kebun untuk mengoptimalkan pengolahan perkebunan dengan efisien.

Teknologi pengindraan jauh *(remote sensing)* menggunakan pesawat tanpa awak (*drone*) merupakan salah satu teknologi alternatif untuk mendapatkan data spasial digital lebih detil, *real time*, cepat, akurat dan lebih murah. Selain itu dalam beberapa tahun terakhir kecerdasan buatan berbasis *deep learning* semakin berkembang dan banyak diterapkan dalam survei lahan pertanian karena memiliki kemampuan pengolahan gambar dan analisis data yang bagus sehingga akan sangat membantu dalam ekstraksi informasi data kondisi lahan (Arrofiqoh dan Harintaka, 2018). Salah satu manfaat yang bisa diambil pada kedua hal tersebut, yaitu dengan

menghitung jumlah pohon secara otomatis. Perhitungan jumlah pohon secara otomatis dapat mempermudah pemilik kebijakan lahan untuk menentukan jumlah kebutuhan yang digunakan untuk pemeliharaan pohon mangga sehingga dapat membantu pengambilan informasi data yang selama ini masih dilakukan secara manual. Perhitungan jumlah pohon secara manual dinilai tidak efisien dan praktis karena perlu menghitung satu persatu pohon pada kebun sehingga memerlukan banyak tenaga dan biaya. Permasalahan tersebut menjadi salah satu penghambat dalam usaha peningkatan produktivitas mangga di Kabupaten Situbondo dan di Indonesia. Oleh karena itu, perlu adanya suatu aplikasi yang dapat memberikan informasi mengenai perhitungan jumlah pohon di kebun mangga secara otomatis.

Penelitian ini dilakukan dengan memanfaatkan citra *low-altitude remote sensing* menggunakan *platform drone* untuk mendapatkan data berupa gambaran kondisi lahan perkebunan mangga dan juga *deep learning* untuk merancang sebuah aplikasi yang bisa menghitung jumlah pohon secara otomatis. Perancangan aplikasi menggunakan metode sistem deteksi objek bernama *Faster R-CNN*. *Faster R-CNN* memiliki dibagi menjadi beberapa macam berdasarkan model pada layer *Convolutional Neural Network* (CNN). Penelitian ini menggunakan dua model yaitu *Inception V2* dan *ResNet-50*. Kedua model tersebut akan digunakan untuk melakukan perhitungan jumlah pohon dengan cara mendeteksi setiap objek yang ada pada gambar lahan perkebunan.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka rumusan masalah dari penelitian ini adalah sebagai berikut.

1. Bagaimana proses pada perhitungan jumlah pohon secara otomatis menggunakan metode *Faster R-CNN*?
2. Bagaimana hasil perhitungan jumlah pohon secara otomatis menggunakan metode *Faster R-CNN*?

1.3 Batasan Masalah

Batas masalah yang diambil dalam penelitian ini adalah sebagai berikut.

1. Data yang digunakan adalah perkebunan mangga di Desa Sopet, Kecamatan Jangkar, Kabupaten Situbondo merupakan hasil dari akuisisi menggunakan *platform drone*.
2. Metode yang digunakan untuk perhitungan jumlah pohon secara otomatis menggunakan metode *Faster R-CNN*.
3. Aplikasi yang dirancang hanya akan mendeteksi dan menghitung vegetasi pohon mangga.

1.4 Tujuan

Tujuan yang ingin dicapai dari penelitian ini adalah sebagai berikut.

1. Merancang aplikasi untuk menghitung jumlah pohon mangga secara otomatis dengan metode *Faster R-CNN*.
2. Menghitung performa aplikasi dari hasil dari *training* data menggunakan metode *Faster R-CNN*.

1.5 Manfaat

Manfaat yang diharapkan dari penelitian ini adalah sebagai berikut.

1. Bagi IPTEK, dapat menambah wawasan dan mengetahui perancangan aplikasi perhitungan jumlah pohon secara otomatis.
2. Bagi masyarakat, dapat dijadikan referensi terhadap perancangan aplikasi perhitungan jumlah pohon secara otomatis.
3. Bagi pemerintah, dapat dipergunakan sebagai rekomendasi untuk melakukan kegiatan perhitungan jumlah pohon secara otomatis.

BAB 2. TINJAUAN PUSTAKA

2.1 Tanaman Mangga

Tanaman mangga *(Mangifera indica,* L*)* merupakan tanaman buah tahunan (parennial plants) berupa pohon berbatang keras berasal dari negara India. Menurut Safitri (2012), dalam taksonomi tanaman mangga diklasifikasikan sebagai berikut,

Kingdom : *Plantae;*
Diviso : *Spermatophyta;*
Kelas : *Dicotyledoneae;*
Ordo : *Sapindales;*
Famili : *Anacardiaceae;*
Genus : *Mangifera;*
Spesies : *Mangifera indica* L.

Varietas tanaman mangga yang ditanam di daerah kajian penelitian ini merupakan varietas mangga gadung. Varietas ini dapat ditanam pada ketinggian 600 mdpl dengan suhu antara 28-36° (Iswanto, 2002). Menurut Suparman (2007), tanaman mangga varietas gadung memiliki karakteristik sebagai berikut:

a. Batang pohonnya berdiameter antara 50 cm – 80 cm.
b. Tinggi pohonnya bisa mencapai 7 m – 10m.
c. Cabang dan rantingnya banyak sehingga tampak rimbun.
d. Daun berbentuk meruncing agak pendek dan memiliki panjang 20 cm- 25 cm dengan lebar 5 cm – 8 cm.
e. Bunga berwarna hijau muda agak kekuningan.

Karakteristik pada setiap tanaman mangga memiliki perbedaan di antara lebar kanopi, tinggi tanaman, lingkar batang, dan percabangan. Perbedaan tersebut lebih dipengaruhi oleh faktor umur tanaman dan kondisi lingkungan. Semakin tua umur tanaman, tinggi tanaman semakin tinggi dan kanopi tanaman cenderung tumbuh melebar. Permukaan batangnya pun semakin kasar. Dan semakin tinggi tanaman kanopi, tanaman cenderung tumbuh melebar sehingga produksi tanaman juga semakin tinggi (Oktavianto *et al.*, 2015).

2.2 *Remote Sensing*

Remote sensing atau pengindraan jauh merupakan definisi yang berasal dari dua kata, yaitu '*remote*' yang berarti dari jauh dan *sensing* yang berarti 'mengukur' (Indarto, 2014). *Remote sensing* berarti mengukur objek dari jauh tanpa menyentuh objek yang akan diukur. *Remote sensing* digunakan untuk memperoleh informasi atau data tentang suatu objek suatu daerah melalui analisis data yang diperoleh dengan suatu alat tanpa kontak langsung dengan objek, daerah, atau fenomena yang dikaji. Pengumpulan data pengindraan jauh dilakukan dengan menggunakan alat pengindra atau alat pengumpul data yang disebut sensor. Data pengindraan jauh dapat berupa gambar, grafik, dan data numerik (Purwadhi 2001). *Remote sensing* memiliki banyak teknik berdasarkan *platform*, ketinggian, dan cara pengambilan datanya. Salah satu teknik *remote sensing*, yaitu LARS.

LARS (*low-altitude remote sensing*) adalah sebuah teknik yang digunakan untuk memperoleh gambar permukaan bumi dari ketinggian yang rendah. sistem ini sebagian besar gambar diperoleh dari bawah tutupan awan karena *platform* yang digunakan bukanlah satelit, namun UAV (*Unmanned Aerial Vehicle*) yang memudahkan penggunanya mendapatkan data lebih cepat dan praktis. Data tersebut berupa format gambar yang kemudian diinterpretasikan untuk dapat diambil informasinya mengenai objek yang diamati (Swain, 2010). Ketinggian dari LARS berkisar antara 150-200 m dengan jalur terbang yang masih di dalam cakupan penglihatan pilot UAV itu sendiri. Teknik LARS banyak dimanfaatkan oleh kegiatan pertanian presisi untuk *monitoring* area perkebunan, sawah, dan juga hutan (Everaets, 2008). Berdasarkan penelitian Salim et al. (2018) gambar hasil akuisisi menggunakan *drone* dapat memberikan hasil yang sangat detail karena terbang pada ketinggian yang rendah. Sehingga ekstraksi informasi vegetasi pada pemetaan lahan akan sangat akurat.

2.3 *Deep Learning*

Deep Learning merupakan sebuah metode pembelajaran terhadap data yang bertujuan untuk membuat representasi (abstraksi) data secara bertingkat menggunakan sejumlah *layer* pengolahan data. *Deep learning* mampu membuat

representasi data secara bertingkat tanpa diprogram secara eksplisit melainkan dengan proses optimasi yang dilakukan menggunakan algoritma pembelajaran (LeCun *et al.*, 2018). Arsitektur model yang dipergunakan pada *deep learning* untuk membuat representasi data adalah *neural network* dengan jumlah layer yang banyak. Kata ”deep atau depth” pada *deep learning* menunjukkan bahwa jumlah *layer* dari model yang digunakan sangat banyak. Banyaknya jumlah *layer* tersebut membuat model semakin dalam sehingga disebut pembelajaran mendalam atau *deep learning* (Francois, 2018).

Proses pembelajaran model *deep learning* memungkinkan implementasi secara paralel sehingga bisa memanfaatkan kapasitas GPU (*Graphic Processing Unit*) dan TPU (*Tensor Processing Unit*). Selain itu pembelajaran yang bersifat iteratif memungkinkan model dapat menggunakan dataset dengan skala besar (Heryadi dan Irwansyah). Data yang digunakan sebagai input proses pembelajaran tidak membutuhkan rekayasa fitur sehingga *deep learning* dikenal sebagai algoritma pembelajaran yang sederhana (Heryadi dan Irwansyah 2020).

2.4 *Computer Vision*

Computer vision adalah salah satu pembelajaran mesin untuk menganalisis gambar dan video untuk memperoleh hasil sebagaimana yang bisa dilakukan manusia. Pada hakikatnya, *computer vision* mencoba meniru cara kerja sistem visual manusia (*Human Vision*). Manusia melihat obyek dengan indra penglihatan (mata), lalu citra obyek diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti obyek apa yang tampak dalam pandangan matanya. Hasil interpretasi ini mungkin digunakan untuk pengambilan keputusan (Umam dan Negara, 2016). Menurut *Computer vision* adalah kombinasi antara:

a. Pengolahan Citra

   Merupakan bidang yang berhubungan dengan proses transformasi citra atau gambar yang bertujuan untuk mendapatkan kualitas citra yang lebih baik.

b. Pengenalan Pola

   Merupakan bidang yang berhubungan dengan proses identifikasi obyek pada citra atau interpretasi citra untuk mengekstrak informasi dari citra tersebut.

Tahapan awal yang dilakukan oleh komputer dalam mengenali objek adalah tahapan segmentasi. Segmentasi adalah tahapan pemisahan citra menjadi bagian – bagian yang diharapkan merupakan objek – objek tersendiri dengan cara membagi suatu wilayah pada citra berdasarkan kriteria kemiripan tertentu antara derajat keabuan *pixel– pixel* terdekat (Darmawan, 2009). Hasil dari *computer vision* ini merupakan informasi sehingga *computer* tidak hanya melihat sebuah gambar namun juga dapat belajar dari apa yang dilihat (Szelski, 2010).

2.5 Citra Digital

Citra merupakan gambaran bagian permukaan bumi sebagaimana terlihat dari ruang angkasa (satelit) atau dari udara (pesawat terbang) (Prahasta, 2008). Citra yang dimaksud adalah gambar diam (foto) maupun citra bergerak (yang berasal dari sensor), sedangkan digital di sini mempunyai maksud bahwa pengolahan gambar atau gambar dilakukan secara digital menggunakan komputer. Citra digital diperoleh secara otomatis dari sistem penangkapan gambar digital dan membentuk suatu matriks yang menyatakan intensitas cahaya pada suatu himpunan diskrit dari suatu titik atau gambar masukan diperoleh melalui suatu kamera yang di dalamnya terdapat suatu alat digitasi yang mengubah gambar masukan berbentuk analog menjadi gambar digital (Suhandy, 2003).

Citra digital dilihat atau dibaca berdasarkan dari gabungan dari beberapa titik-titik kecil yang merupakan elemen inti dari suatu gambar inilah yang disebut *pixel*, singkatan dari *picture elements*. Setiap *pixel* tersebut direpesentasikan dalam bentuk nilai *pixel* yaitu 1 – 255 bit tergantung dari warnanya. Setiap *pixel* tersebut bergabung dengan *pixel–pixel* lainnya sehingga membentuk suatu pola dan menghasilkan gambar (Prabowo dan Ahmad, 2018). Gabungan-gabungan antar *pixel* tersebut dapat diwakili oleh sebuah matriks dua dimensi f(x,y) yang terdiri dari M kolom dan N baris, di mana perpotongan antara kolom dan baris disebut pixel (*pixel = picture element*) atau elemen terkecil dari sebuah gambar (Kusumanto dan Tompunu, 2011). Citra digital direpresentasikan dalam matriks yang ditunjukkan pada persamaan 2.1.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) \dots & f(0,M) \\ \dots . & \dots & \dots \\ f(N-1.0) & f(N-1.1) \dots & f(N-1.M) \end{bmatrix} \dots\dots\dots\dots\dots\dots\dots\dots\dots 2.1$$

2.6 Indeks Vegetasi

Indeks vegetasi (*vegetation index*), merupakan sebuah gambar yang dianalisis berdasarkan nilai-nilai kecerahan digital. Indeks vegetasi terbentuk dari kombinasi dari beberapa nilai spektral dengan menambahkan, dibagi atau dikalikan dengan cara yang dirancang untuk menghasilkan nilai tunggal yang menunjukkan jumlah atau kekuatan vegetasi dalam *pixel* (Cambell *et al.*, 2011). Indeks vegetasi dilihat berdasarkan besaran nilai kehijauan vegetasi yang diperoleh dari pengolahan sinyal digital data nilai kecerahan (*brightness*) beberapa kanal data sensor satelit. Pemantauan dilakukan dengan proses perbandingan antara tingkat kecerahan kanal cahaya merah (*red*), hujau (*green*), dan biru (*blue*) vegetasi serta kanal cahaya inframerah dekat (*near infrared*) (Sudiana, 2008).

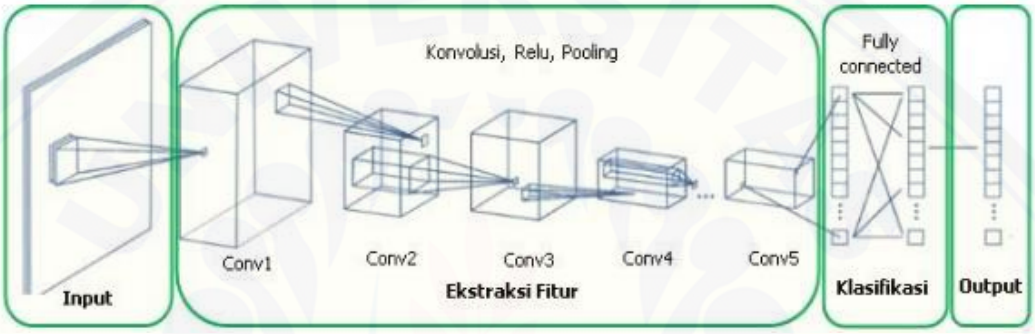
Algoritma pemrosesan sinyal digital untuk menghitung indeks vegetasi diperoleh dengan memanfaatkan fenomena penyerapan cahaya merah oleh klorofil dan pemantulan cahaya inframerah dekat oleh jaringan mesofil yang terdapat pada daun membuat nilai kecerahan yang diterima sensor satelit pada kanal tersebut jauh berbeda. Daratan non-vegetasi, termasuk di antaranya wilayah perairan, pemukiman penduduk, tanah kosong terbuka, dan wilayah dengan kondisi vegetasi yang rusak, tidak menunjukkan nilai rasio yang tinggi (minimum). Sebaliknya wilayah bervegetasi sangat rapat dengan kondisi sehat, perbandingan kedua kanal tersebut akan sangat tinggi (Sudiana, 2008).

2.7 *Convolutional Neural network*

Convolutional Neural Network (CNN atau ConvNet) adalah salah satu algoritma dari *deep learning* yang merupakan pengembangan dari *Multilayer Percepton* (MLP) yang dirancang untuk mengolah data dalam bentuk dua dimensi, misalnya gambar atau suara. CNN digunakan untuk mengklasifikasi data yang terlabel dengan menggunakan metode *supervised learning*. Cara kerja dari *supervised learning* adalah terdapat data yang dilatih dan terdapat variabel yang

ditargetkan sehingga tujuan dari metode ini adalah mengelompokkan suatu data ke data yang sudah ada (Ilahiyah dan Nilogi, 2018). Pemberian nama *Convolutional Neural network* mengindikasikan bahwa jaringan tersebut menggunakan operasi matematika yang disebut konvolusi. Konvolusi sendiri adalah jumlah total dari hasil kali antara setiap elemen yang bersesuaian (memiliki posisi koordinat yang sama) dalam dua matriks atau dua vektor (Madenda, 2015).

Arsitektur CNN terdiri dari proses ekstraksi fitur dan proses klasifikasi seperti yang ditampilkan pada Gambar 2.1.



Gambar 2.1 Arsitektur CNN (Sumber: Krizhevsky et al., 2012)

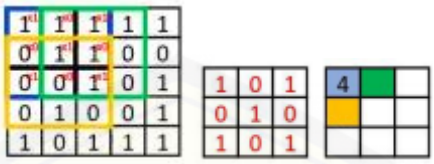
1. Ekstraksi Fitur

   Ekstraksi fitur merupakan tahap di mana gambar akan melalui proses konvolusi dan yang nantinya akan dilanjutkan ke tahap klasifikasi.

a. Lapisan Konvolusi

   Lapisan konvolusi menggunakan *Filter* untuk mengekstraksi objek dari gambar input. *Filter* ini berisi bobot yang digunakan untuk mendeteksi karakter dari objek seperti tepi, kurva, atau warna. Konvolusi akan menghasilkan transformasi linear dari gambar input yang sesuai dengan informasi spasial pada data. *Filter* diaplikasikan secara berulang sehingga menghasilkan serangkaian bidang *receptive*. Terdapat parameter yang dapat diubah untuk memodifikasi sifat tiap lapisan, yaitu ukuran filter, *stride* dan *padding*. *Stride* mengontrol bagaimana *Filter* diterapkan pada data input dengan bergerak sepanjang ukuran piksel yang telah ditentukan. *Padding* adalah penambahan ukuran piksel dengan nilai tertentu di sekitar data input agar hasil dari bidang *receptive* tidak terlalu kecil sehingga tidak banyak informasi yang hilang. Nilai ini biasanya nol sehingga

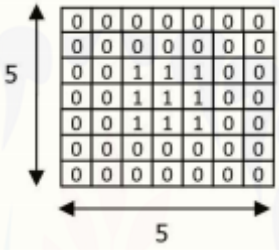
disebut dengan zero *padding*. Hasil dari bidang *receptive* berupa data tunggal. Input dari proses konvolusi ini dijadikan sebagai input untuk lapisan konvolusi selanjutnya (Castelluccio *et al*., 2015).



(a) (b) (c)

(b) *Filter* 3x3 (c) Bidang *Receptive* 3x3.

Gambar 2.2 Operasi konvolusi dengan *stride* 1 (a) input data 5x5



Gambar 2.3 Operasi zero *padding* 2 pada data 3x3

b. Fungsi Aktivasi (Relu)

ReLU (Rectification Linear Unit) merupakan operasi untuk mengenalkan nonlinearitas dan meningkatkan representasi dari model. Fungsi aktivasi *ReLU* adalah f(x) = max(0,x) (Heaton, 2015). Nilai input dari neuron bisa dinyatakan sebagai 0 jika input adalah negatif. Jika nilai input adalah positif, maka input dari neuron adalah nilai input aktivasi itu sendiri (Kim *et al*., 2016).

c. *Pooling*

Pooling atau *subsampling* adalah pengurangan ukuran matriks. Terdapat dua macam *pooling* yang sering digunakan yaitu *average pooling* dan *max pooling* (Bejiga *et al*., 2017). Nilai yang diambil pada *average pooling* adalah nila rata-rata sedangkan pada *max pooling* adalah nilai maksimal (Zhi *et al*., 2016).

2. Klasifikasi

Pada tahap ini gambar akan dikelompokkan berdasarkan jumlah fitur yang terdapat pada gambar. pada tahap klasifikasi akan dilakukan dua proses yaitu fully connected dan fungsi aktivasi.

a. *Fully Connected*

Lapisan *fully connected layer* merupakan kumpulan dari proses konvolusi (Hijazi *et al.*, 2015). Lapisan ini mendapatkan input dari proses sebelumnya untuk menentukan fitur mana yang paling berkorelasi dengan kelas tertentu. Fungsi dari lapisan ini adalah untuk menyatukan semua *node* menjadi satu dimensi (Albelwi & Mahmood, 2017).

b. Fungsi Aktivasi *(Softmax)*

Fungsi aktivasi *softmax* digunakan untuk mendapatkan hasil klasifikasi. Fungsi aktivasi menghasilkan nilai yang diinterpretasi sebagai probabilitas yang belum dinormalisasi untuk tiap kelas. Nilai kelas dihitung dengan menggunakan fungsi *softmax* (Vedaldi & Lenc, 2015). Persamaan fungsi aktivasi *(softmax)* ditunjukkan pada persamaan di bawah ini.

$$y_{ijk} = \frac{e^{xijk}}{\sum_{t=1}^{D} e^{xijt}} \quad \text{.....(2,2)}$$

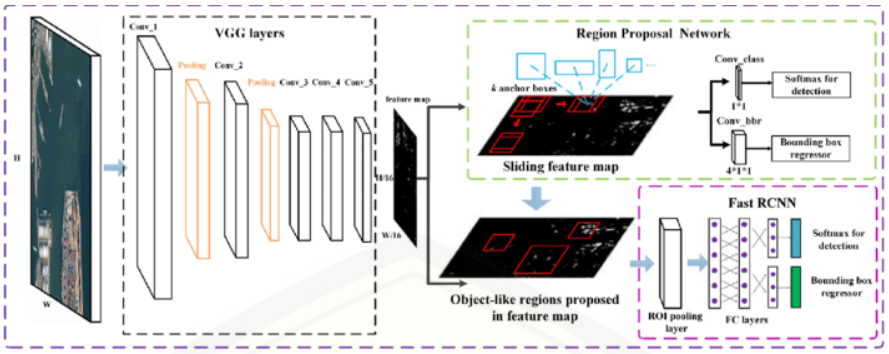
Keterangan:

y_{ijk} = vektor yang berisi nilai antara 0 dan 1.

X = vektor yang berisi nilai yang didapatkan dari lapisan *Fully-Connected* terakhir.

2.8 *Faster R-CNN*

Faster R-CNN merupakan model arsitektur yang digunakan sebagai sistem deteksi objek. *Faster RCNN* akan mendeteksi bagian-bagian yang dianggap sebagai pohon pada gambar. Arsitektur *Faster R-CNN* terdiri dari dua bagian yaitu RPN (Region Proposal Network) dan *Fast R-CNN detector*. Berikut merupakan arsitektur dari *Faster R-CNN*.



Gambar 2.4 Arsitektur *Faster-RCNN* (Sumber:Deng, 2018)

RPN dimulai dengan layer CNN sebagai *backend* yang disebut dengan *backbone* untuk menghasilkan *feature map*. *Feature map* yang dihasilkan dari *backbone* digunakan untuk mencari dan menentukan letak pohon mangga pada koordinat *pixel* tertentu di dalam gambar. RPN memprediksi tingkat keberadaan pohon mangga berdasarkan *bounding box ground truth* yang dibuat pada saat anotasi data. *Ground truth* tersebut akan definisikan sebagai *foreground* oleh sistem. Keluaran RPN ini adalah sejumlah *region proposal* yang disebut dengan *anchor*. *Anchor* yang berisi pohon mangga disebut *foreground* dan sebaliknya disebut *background*. *Anchor* akan didefinisikan sebagai *foreground* jika memiliki nilai overlap (IoU) yang lebih tinggi dari nilai *threshold* yaitu 0,7 dan jika nilai rendah maka akan didefinisikan sebagai *background* (Karlita *et al.*, 2019).

Fast R-CNN *Detector* akan menerima beberapa *region of interest* (RoI) sebagai input. Feature vector diekstrak oleh ROI *pooling* layer dari *convolutional* layer. Setiap *feature vector* akan dimasukkan ke dalam *fully connected* (FC) layer. Hasil keluaran dari *detector* akan diarahkan menuju *softmax layer* dan *bounding box regressor* layer antara lain: 1) probabilitas yang menentukan *K* kelas objek ditambah kelas "latar belakang" dan (2) nilai *bounding box* (bbox). Pada penelitian ini, nilai *K* atau kelas yang digunakan adalah 1, yaitu kelas objek hanya berisi satu objek "pohon mangga" sebagai *foreground* dan kelas "latar belakang" (Xu, 2017).

BAB 3. METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

Pelaksanaan penelitian dimulai 17 Oktober 2019 sampai 20 Mei 2020. Wilayah yang dijadikan sebagai lokasi pengambilan data yaitu di perkebunan mangga di Desa Sopet, Kecamatan Jangkar, Kabupaten Situbondo. Pengolahan data dilakukan di Laboratorium N Computing Jurusan Teknik Pertanian, Fakultas Teknologi Pertanian, Universitas Jember.

3.2 Alat dan Bahan

3.2.1 Alat Penelitian

Alat yang digunakan dalam penelitian ini, yaitu sebagai berikut.

a. *Personal Computer* digunakan untuk mengolah.
b. *Drone* digunakan untuk mengambil data gambar pohon mangga.
c. *MapInr* digunakan untuk mengetahui letak dan jumlah pohon mangga.
d. *Pix4D* digunakan untuk ortmosaic.
e. *QGIS* digunakan untuk pengolahan indeks vegetasi.
f. *Photoshop* digunakan untuk *crop* dan augementasi gambar.
g. LabelImg untuk proses anotasi data.
h. *Idle Python* untuk *resize* gambar dan menulis kode program
i. *Anaconda Prompt* digunakan untuk eksekusi program yang telah dibuat.
j. RGB Color Code digunakan untuk mendapatkan nilai intensitas warna.

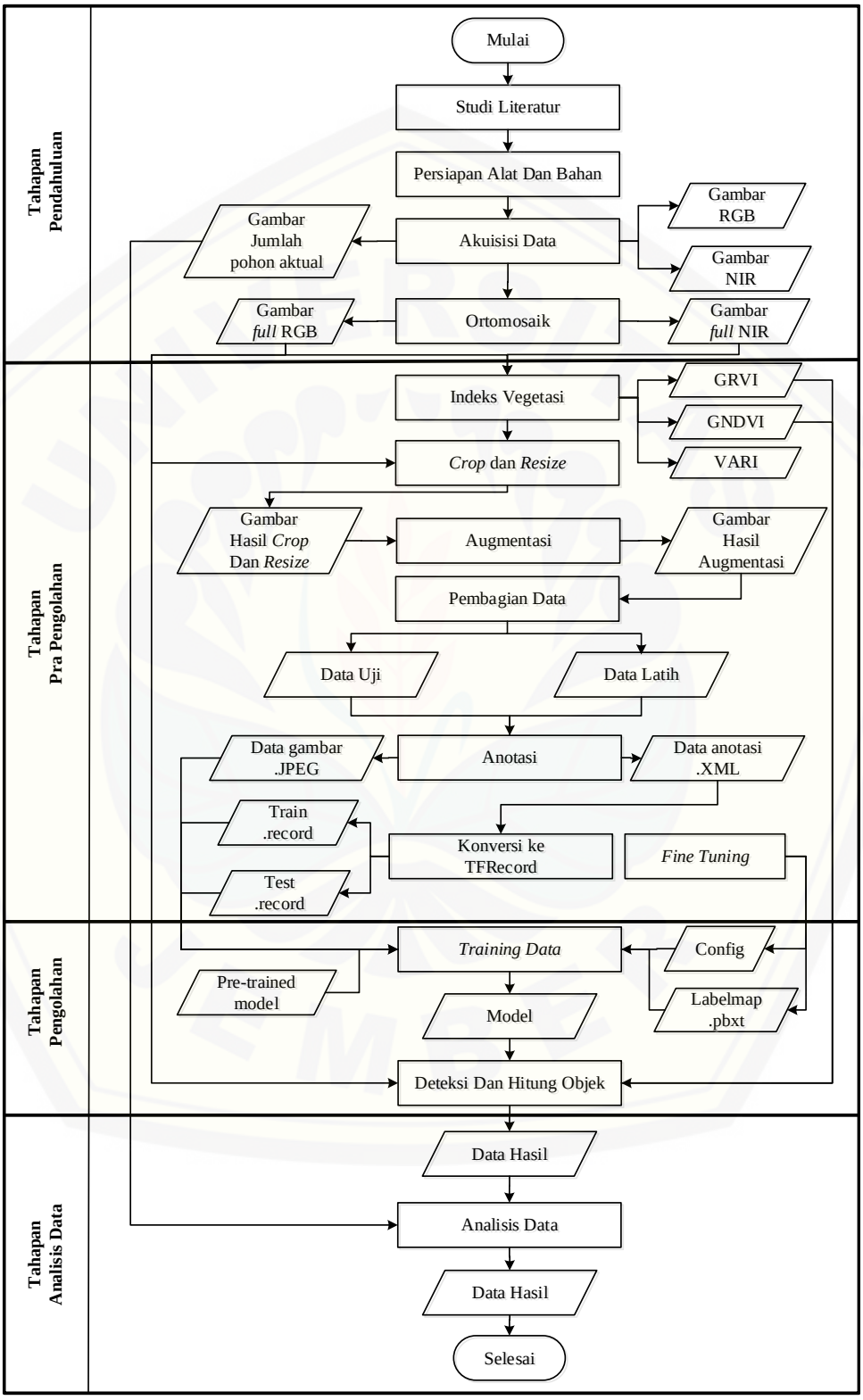
3.2.2 Bahan Penelitian

Bahan yang digunakan dalam penelitian ini, yaitu sebagai berikut.

a. Gambar RGB dan NIR lahan perkebunan mangga sebanyak 87 gambar berekstensi .JPEG dengan resolusi 72 dpi dan ukuran 4000 x 3000 pixel.
b. Gambar letak pohon hasil perhitungan pohon mangga manual.
c. *Pre-Trained* model Inception V2 dan ResNet-50 yang sebelumnya telah dilatih dengan dataset COCO (*Common Objects In Context*).
d. *Source code* Tensorflow Object Detection API.

3.3 Prosedur Penelitian

Prosedur penelitian disajikan dalam diagram alir pada Gambar 3.1 berikut.



Gambar 3.1 Diagram alir kegiatan penelitian

3.3.1 Studi Literatur

Tahap awal penelitian, yaitu **s**tudi literatur yang dilakukan dengan cara mencari dan mempelajari referensi teori-teori dari berbagai sumber khususnya buku dan penelitian terkait. Hal-hal uang dipelajari seperti *image acquisition* menggunakan *drone*, *image processing*, *deep learning*, bahasa pemrograman *python* dan algoritma deteksi objek. Studi literatur dilakukan untuk penyusunan yang sistematis agar memudahkan langkah-langkah yang akan diambil.

3.3.2 Akuisisi Data

Akuisisi data pada penelitian ini dibagi menjadi dua yaitu data berupa gambar dengan drone dan data jumlah pohon manual.

a. Akuisisi data gambar drone

Akuisisi data gambar dilakukan menggunakan *drone* yang diambil dari ketinggian tegak lurus 50 m dengan luas area 70 m^2. Berikut merupakan prosedur pengambilan data menggunakan drone.

Gambar 3.2 Diagram alir akusisi data

Hasil data (gambar) RGB (Red Green Blue) dan NIR (Near Infrared) tergantung oleh jenis filter lensa pada kamera atau sensor yang dipasang pada *drone*. Setelah itu gambar hasil drone yang masih terbagi-bagi menjadi beberapa bagian akan dilakukan proses ortomozaik untuk menyatukan gambar menjadi satu. Gambar RGB dan NIR ditampilkan pada Lampiran 1.

b. Akuisisi data jumlah pohon manual

Akuisisi data gambar dilakukan menggunakan aplikasi GPS berbasis *android* yaitu *MapInr*. Setiap pohon mangga akan ditandai *point* atau titik lalu pohon yang bukan pohon mangga akan diabaikan. Setelah itu hasil perhitungan akan ditampilkan di *attribute layer*. Jumlah hasil perhitungan pohon mangga secara manual yang didapatkan yaitu sebanyak 110 pohon mangga. Hasil dari perhitungan secara manual pada penelitian ini akan dijadikan nilai aktual dan akan dibandingkan dengan nilai prediksi hasil *training* data. Hasil pengambilan data perhitungan secara manual ditampilkan pada Lampiran 4.2.

3.3.3 Indeks Vegetasi

Indeks Vegetasi dilakukan dengan cara memisahkan *band* warna pada gambar RGB sehingga menghasilkan gambar *red, green dan blue*. Setelah itu akan dilakukan proses pengabungan gambar menggunakan persamaan tergantung dari jenis indeks vegetasi yang digunakann. Penggabungan gambar dilakukan menggunakan fitur *raster calculator*. Berikut merupakan dari persamaan indeks vegetasi VARI (Gitelson et al., 2002) yang ditunjukkan pada persamaan 3.1, GRVI (Sripada *et al.*, 2006) yang ditunjukkan pada persamaan 3.2, dan GNDVI (Gitelson et al., 1998) yang ditunjukkan pada persamaan 3.3.

$$\text{VARI} = \frac{Green - Red}{Green + Red - Blue} \quad \text{(3.1)}$$

$$\text{GRVI} = \frac{NIR}{Green} \quad \text{(3.2)}$$

$$\text{GNDVI} = \frac{(NIR - Green)}{(NIR + Green)} \quad \text{(3.3)}$$

3.3.4 *Crop* dan *Resize*

Setelah didapatkan gambar indeks vegetasi, setiap gambar yaitu RGB, VARI, GRVI, dan GNDVI akan di *crop* untuk memotong gambar menjadi bentuk persegi yang sebelumnya masih berbentuk berantakan. Selain itu *crop* gambar juga berguna untuk mendapatkan lebih banyak data dengan cara memotong bagian tertentu (Endrianti *et al.*, 2018). Gambar yang dipotong masing-masing menghasilkan 30 buah gambar. Gambar hasil *crop* menghasilkan gambar yang berukuran kecil yaitu 300 x 300 *pixel* sehingga perlu dilakukan penyesuaian ukuran dengan cara *resize* gambar. *Resize* gambar dilakukan dengan cara menambah resolusi gambar menjadi ukuran 800 x 800 *pixel.*

3.3.5 Augmentasi Data

Data yang dihasilkan dari proses *crop* masih dapat dikatakan terbatas sehingga model yang akan dihasilkan tidak akan optimal. Oleh karena itu perlu dilakukan perbanyakan data dengan cara augmentasi data. Augmentasi data dilakukan dengan cara mengubah atau memodifikasi gambar sedemikian rupa sehingga komputer akan mendeteksi bahwa gambar yang diubah adalah gambar yang berbeda. Dari hasil transformasi tersebut akan menghasilkan data yang lebih banyak sehingga komputer bisa lebih banyak belajar (Zufar dan Budiyono, 2016). Jenis augmentasi yang digunakan pada penelitian ini adalah *rotate right, rotate left, flip vertical, flip horizontal* dan *skew* yang menghasilkan. Gambar hasil proses augmentasi ini menghasilkan 360 buah gambar pada setiap data gambar.

3.3.6 Pembagian Data

Gambar yang telah diaugmentasi selanjutnya akan dibagi 30 % untuk data uji 70 % untuk data latih. Jumlah data latih lebih banyak daripada data uji dipilih karena untuk mendapatkan hasil model yang bagus. Selain itu agar hasil dari *training* tidak hanya dapat diterapkan pada dataset yang lama tapi juga dataset yang baru (Zainudin, 2019). Berikut merupakan pembagian dari data yang digunakan.

1. Data latih

   Data latih digunakan untuk digunakan proses *training* data. Jumlah data yang digunakan pada data latih berjumlah 224 gambar dari total 360 data gambar.

2. Data uji

   Data uji digunakan untuk menguji hasil dari model yang telah dibuat pada proses *training*. Jumlah data yang digunakan pada data latih berjumlah 136 gambar.

3.3.7 Anotasi Data

Anotasi gambar dilakukan dengan cara melabeli area pohon mangga. Pelabelan dilakukan dengan cara *tagging* secara manual pada semua gambar, baik pada data *train* maupun data *test*. Hasil dari anotasi ini berupa informasi nama kelas, jumlah kelas, dan posisi *bounding box*. *Bounding box* akan berbentuk persegi atau persegi panjang pada setiap objek (pohon mangga). *Bounding box* hasil anotasi ini didefisikan sebagai *ground truth box* yang berisi informasi berupa koordinat pojok kiri atas *bounding box* beserta dimensi panjang dan lebar, nama kelas objek. *File* gambar hasil anotasi disimpan dalam *file* berekstensi .xml. dan dalam format PASCAL VOC (Karlita, 2019).

3.3.8 Konversi ke *TFRecord*

TFRecord atau *Tensorflow Record* merupakan *file* format untuk menyimpan data urutan berupa catatan biner. *TFRecord* memiliki beberapa kelebihan salah satunya dapat membaca dan menulis (data input) dengan lebih cepat sehingga dapat mengatasi lamanya durasi *training* karena jumlah data yang banyak. Pada penelitian ini data rekaman yang digunakan adalah urutan yang akan diproses saat *training* antara lain. (1) *file* gambar data latih dan data uji dengan ekstensi .JPEG. (2) *file* anotasi data latih dan data uji pada ekstensi .XML. Kedua *file* tersebut akan di-*encode* atau dikonversi menjadi representasi *integer* untuk mendapatkan hasil catatan biner tersebut. Hasil dari konversi menghasilkan dua *file* dengan ekstensi .TFRecord yaitu untuk data latih dan data uji (Nelson, 2020).

3.3.9 *Fine Tuning*

Fine tuning merupakan proses konfigurasi parameter yang dilakukan untuk mendapatkan hasil model yang optimal dan terhindar dari *overfitting* dan *underfitting*. Pemilihan parameter yang digunakan untuk *training* data akan berpengaruh terhadap model yang dihasilkan. Oleh karena itu *fine tuning* perlu dilakukan agar dapat menghasilkan model yang fit dan terhnidar dari model yang *overfitting* (Li et al., 2020). Proses *fine tuning* dilakukan dengan cara melakukan *training* berkali-kali sampai menemukan parameter yang cocok dengan dataset dan mendapatkan hasil yang optimal. Parameter yang digunakan pada masing-masing pre-trained model akan menggunakan pamater yang sama, sebagai berikut.

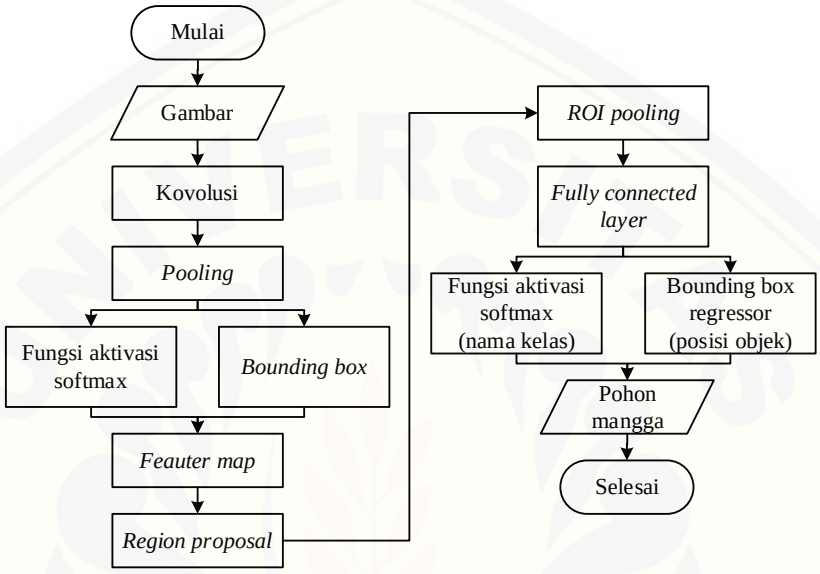
Epoch	:	menentukan jumlah pengulangan saat *training*. Jumlah yang digunakan adalah 10.000
Learning rate	:	Menentukan seberapa lama waktu yang dibutuhkan untuk mencapai hasil yang optimal. Jumlah yang digunakan adalah 0,02
Strides	:	menentukan seberapa banyak *kernel* melakukan *sliding windows*. Jumlah yang digunakan adalah8
Batch	:	menentukan seberapa banyak data yang akan diproses untuk *training*. Jumlah yang digunakan adalah 1
Dropout	:	menentukan seberapa banyak *nodes* yang akan dibuang pada *fully connected layer*. Jumlah yang digunakan adalah 0,5

Hasil dari fine tuning ini merupakan *file* dengan ekstensi .config yang digunakan sebagai konfigurasi parameter untuk optimasi model dan *file* dengan ekstensi .pbxt yang digunakan sebagai nama kelas. Kedua *file* tersebut merupakan *file* yang akan di input saat *training* data.

3.3.10 *Training* Data

Training data merupakan proses pelatihan atau pembelajaran neural *network* terhadap suatu data yang bertujuan untuk memahami informasi yang terkandung dalam data tersebut yang diatur menurut parameter yang digunakan. Target dalam proses *training* ini adalah nilai output yang sesuai dengan nilai input.

Pada penelitian ini target yang ingin dicapai adalah model dapat memberikan *bounding box* prediksi yang tepat pada pohon mangga sesuai dengan *bounding box* aktual (*ground truth*) yang merupakan hasil dari proses anotasi. *Training* data akan dilakukan menggunakan *Faster R-CNN*. Berikut merupakan proses yang terjadi selama *training* data menggunakan *Faster R-CNN*.



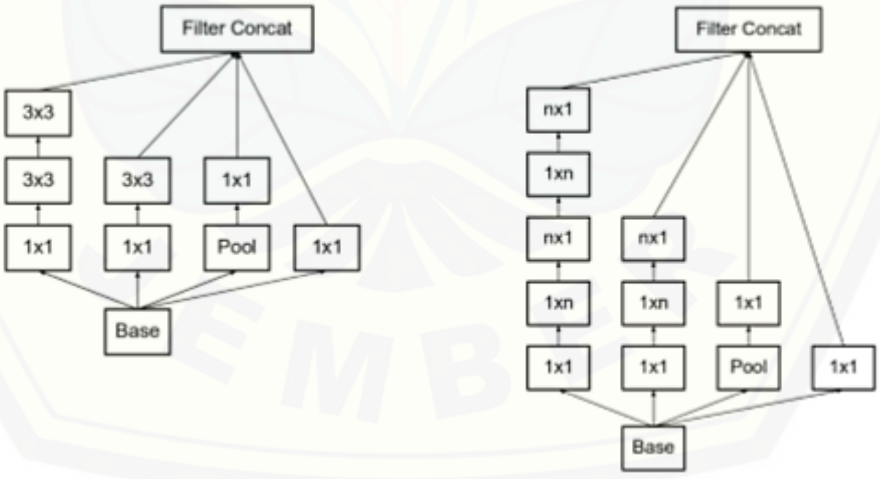
Gambar 3.3 Diagram alir *training* data menggunakan *Faster-RCNN*

Pada tahap input gambar sampai tahap *fully connected layer* data yang digunakan adalah data latih sedangkan setelah tahap *fully connected layer* yaitu pada tahap fungsi aktivasi *softmax* dan *bounding box regressor* sampai output gambar data yang digunakan adalah data uji. Langkah-langkah pada Gambar 3.3 tersebut akan dilakukan berkali-kali setiap melakukan *epoch* atau pengulangan. Jadi, jika *epoch* diatur sebanyak 10.000 maka model neural *network* tersebut akan melakukan pengulangan sebanyak 10.000 kali. Setiap *epoch* yang berjalan, *Faster R-CNN* akan menghitung *loss fuction* yang didapat. *Loss fuction* merupakan fungsi yang digunakan untuk menghitung kinerja dari suatu model yaitu dengan menghitung *error* yang dihasilkan dari model tersebut (Pangestu *et al.*, 2020). Berikut merupakan persamaan dari *loss function* dari *Faster R-CNN*.

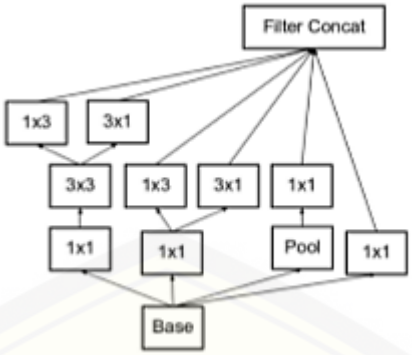
$$(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i * L_{reg}(t_i, p_i^*) \quad \text{.....(3.4)}$$

i merupakan indeks *anchor* dan p_i merupakan prediksi i sebagai objek. Label *ground truth* p_i^* adalah 1 jika positif dan 0 jika negatif. *Multitask loss* ada dua bagian yaitu, klasifikasi L_{cls} dan regresi L_{reg}. t_i merupakan vektor yang merepresentasikan 4 *parameter* koordinat *bounding box* prediksi dan t_i^* merupakan vektor yang *ground truth* terhubung dengan *anchor* positif. t_i dan t_i^* akan dinormalisasikan oleh N_{cls} dan N_{reg} dan akan diseimbangkan dengan λ. Regresi akan bertugas untuk mencari *ground truth* box dari *anchor box* terdekat. Berikut merupakan persamaan 4 koordinat dari setiap *anchor* (Ren, 2015).

Faster-RCNN membutuhkan suatu layer CNN yang disebut dengan *backbone*. *Backbone* tersebut berguna untuk mengekstraksi fitur pada objek untuk menghasilkan *feature map* (Ren, 2015). *Backbone* yang digunakan pada penelitian ini adalah *Inception V2* dan *ResNet-50*. Berikut merupakan arsitektur *Inceptic* (2) dan *ResNet-50* yang disajikan dalam gambar 3.4 dan 3.5.

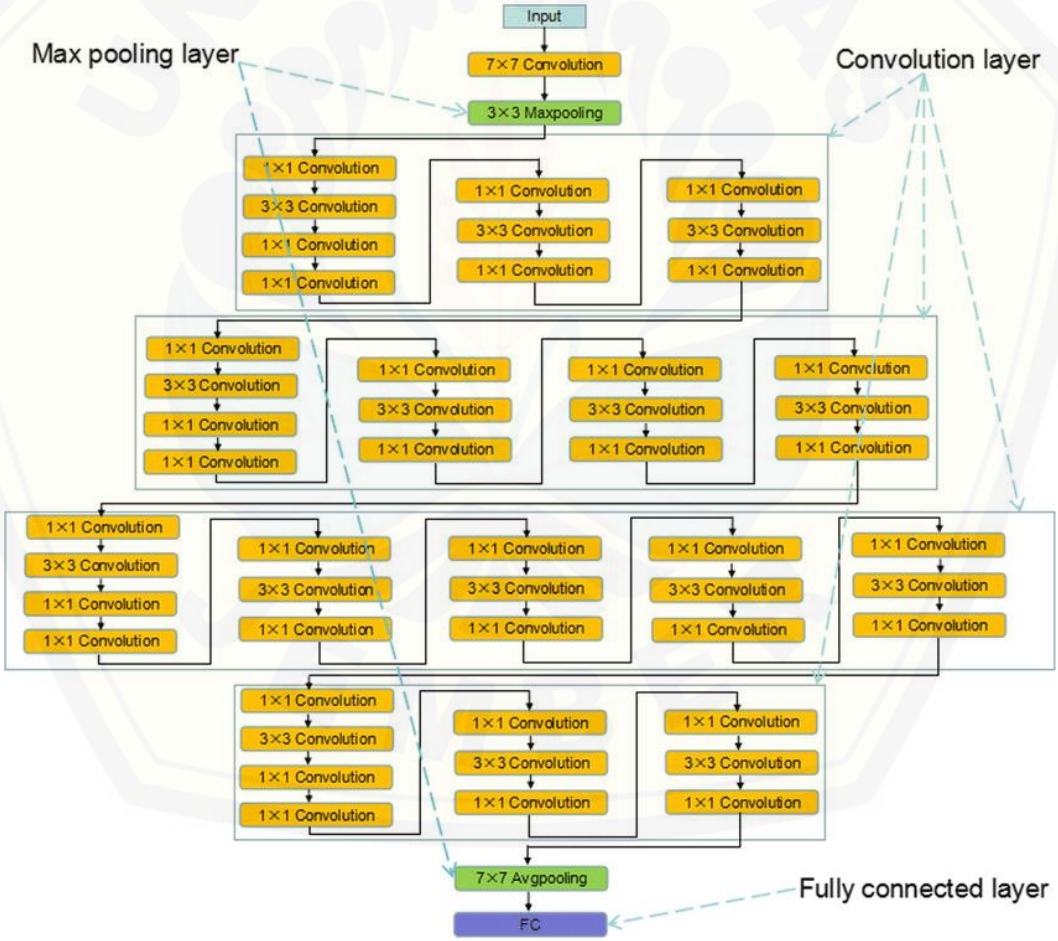


(a) (b)



(c)

Gambar 3.4 Arsitektur *Inception V2* (a) modul 1, (b) modul 2, (c) modul 3 (Sumber: Alamsyah dan Pratama, 2019)



Gambar 3.5 Arsitektur ResNet-50. (Sumber: Peng et al., 2019)

3.3.11 Deteksi dan Hitung Objek

Model yang didapatkan dari hasil *training* data akan diuji dengan cara mendeteksi objek yang ada pada gambar. Gambar yang digunakan saat pengujian merupakan gambar *full* yang terdiri dari banyak pohon mangga. Pada tahap ini program akan mendeteksi letak objek (pohon mangga) pada gambar. Hasil deteksi objek akan menghasilkan gambar baru yang memunculkan sebuah *bounding box* pada objek (pohon) yang terdiri dari nama kelas objek dan nilai akurasi deteksi objek. *Bounding box* hasil deteksi objek ini merupakan *bounding box* prediksi. Jumlah *bounding box* yang muncul akan dihitung oleh program secara otomatis. Gambar hasil pengujian ditunjukkan pada Lampiran 4.3.

3.3.12 Analisis Data

Analisis data dilakukan untuk melihat kinerja aplikasi yang didapat dari model yang telah dilakukan proses *training* data sebelumnya. Analisis data yang digunakan pada penelitian ini adalah uji *confusion matrix*. Menurut Han dan Kamber (2011) *confusion matrix* dapat diartikan sebagai suatu alat yang memiliki fungsi untuk melakukan analisis apakah aplikasi tersebut baik dalam mengenali *tuple* dari kelas yang berbeda. Metode ini disusun dalam tabel yang terdiri dari baris data uji yang diprediksi benar dan tidak benar berdasarkan hasil deteksi sistem (model). Menurut Azzakirot (2018) terdapat empat nilai pengukuran yang digunakan untuk mengukur kinerja model yang diuji, berikut penjelasannya.

1. TP (*True Positive*) merupakan nilai saat program akan mendeteksi pohon mangga yang memang merupakan pohon mangga. Maka TP merupakan nilai di saat *bounding box* tepat berada di objek pohon mangga.
2. TN (*True Negative*) merupakan nilai saat program akan mendeteksi yang bukan merupakan pohon mangga dan memang bukan pohon mangga. TN akan selalu bernilai 0 (nol) karena penelitian ini hanya menggunakan satu kelas yaitu pohon mangga.
3. FP (*False Positive*) merupakan nilai saat program akan mendeteksi yang bukan merupakan pohon mangga dan merupakan pohon mangga. Maka FP merupakan

nilai di saat letak *bounding box* tidak tepat pada pohon mangga atau salah deteksi.

4. FN (*False Negative*) merupakan nilai saat aplikasi tidak akan mendeteksi pohon mangga sebagai pohon mangga. Maka FN merupakan dilai di saat *bounding box* tidak muncul atau tidak bisa mendeteksi pohon mangga.

Hasil yang didapat dari nilai TP, TN, FP, dan FN tersebut merupakan hasil nilai yang didapat setelah menghitung selisih antara perhitungan jumlah pohon secara manual (aktual) dan secara otomatis (prediksi). Berikut merupakan tabel dari *confussion matrix* yang disajikan dalam tabel 3.1.

Tabel 3.1 Contoh tabel *confusion matrix*

		Predicted Class	
		Positif (P)	Negatif (N)
Actual Class	Positif (P)	TP	FP
	Negatif (N)	FN	TN

Setelah dilakukan perhitungan dengan menggunakan uji *confusion matrix*, nilai yang didapat akan dilakukan perhitungan lanjutan untuk mendapatkan hasil analisis performa aplikasi yang dibuat. Berikut persamaan yang digunakan untuk menghitung performa aplikasi. Berikut merupakan persamaan dari *confusion matrix* menurut penelitian dari Azzakirot (2018).

1. *Precision*

   Precision merupakan perbandingan jumlah data yang relevan yang terambil oleh sistem dengan keseluruhan data yang terambil oleh sistem. Berikut merupakan persamaan dari *precision.*

   $$Precision = \frac{TP}{TP + FP}$$

   $$Persentase = N\ x\ 100\ \% \quad \text{.....(3,5)}$$

2. *Recall*

   Recall merupakan perbandingan jumlah data yang relevan yang terambil oleh sistem dengan keseluruhan data relevan yang ada dalam sistem. Berikut merupakan persamaan dari *recall.*

   $$Recall = \frac{TP}{TP + FN} \quad \text{.....(3,6)}$$

$Persentase = N\ x\ 100\ \%$

3. *Accuracy*

   Accuracy merupakan kedekatan hasil pengukuran (prediksi) dengan nilai yang sebenarnya (aktual). Berikut merupakan persamaan dari *accuraccy*.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{.....(3,7)}$$

$$Persentase = N\ x\ 100\ \%$$

4. *Error*

   Error digunakan untuk menghitung persentase tidak berhasilnya aplikasi dalam mendeteksi pohon. Berikut merupakan persamaan dari *error rate*.

$$Error = \frac{TP}{TP + FP} \quad \text{.....(3,8)}$$

$$Persentase = N\ x\ 100\ \%$$

BAB 5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan dari penelitian ini, dapat disimpulkan sebagai berikut.

1. Proses perancangan aplikasi meliputi *preprocessing* seperti *resize*, *crop*, augmentasi, anotasi, dan *training* data. adapun parameter-parameter hasil *traning* data yaitu *learning rate*, *global step*, dan *total loss*.
2. Model terbaik yang digunakan untuk aplikasi perhitungan jumlah pohon adalah model *ResNet-50* menggunakan data RGB dengan nilai *precision* 95,45 %, *recall* sebesar 95,45 %, *accuracy* 91,30 %, dan *error* 8,70 %. Nilai tersebut lebih tinggi jika dibandingkan dengan data yang menggunakan indeks vegetasi.

5.2 Saran

Saran yang diberikan untuk mengembangkan penelitian ini, yaitu sebagai berikut.

1. Menggunakan lebih banyak lagi dataset yang digunakan untuk mendapatkan hasil model yang lebih baik lagi.
2. Menambahkan jumlah *epoch* pada saat *training* data sehingga menghasilkan hasil akurasi yang lebih tinggi lagi.
3. Menggunakan spesifikasi perangkat yang lebih tinggi seperti *Tensor Processing Unit* (TPU) *dan Graphics Processing Unit* (GPU) untuk mempercepat proses *training* data.

DAFTAR PUSTAKA

Aditya Santoso, Gunawan Ariyanto. 2018. Implementasi Deep Learning Berbasis Keras Untuk Pengenalan Wajah. *Jurnal Emitor*. 18(1):15-21.

Ahmad, Usman. 2005. *Pengolahan Gambar Digital*. Yogyakarta: Graha Ilmu.

Akhmad Rohim, Yuita Arum Sari, Tibyan. 2019. Convolution Neural Network (CNN) Untuk Pengklasifikasian Citra Makanan Tradisional. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer. 3(7): 7037-7042.

Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural networks. *Journal of Neural Information Processing Systems*. 25(10):1145.

Anatoly Gitelson, Yoram Kaufman, Mark N, Merzlyak. 1996. Use of a green channel in remote sensing of global vegetation from EOS-MODIS. *Remote Sensing of Environment*. 58(1):289-298.

Anatoly Gitelson, Yoram Kaufman, Robert Stark, Donald Rundquist. 2002. Novel Algorithms for Remote Estimation of Vegetation Fraction. *Remote Sensing of Environment*. 80(1):76-87.

Ari Setiani, Yudo Prasetyo, Sawitri Subiyanto. 2016. Optimalisasi Parameter Segmentasi Berbasis Algoritma Multiresolusi Untuk Identifikasi Kawasan Industri Antara Citra Satelit Landsat Dan Alos Palsar (Studi Kasus: Kecamatan Tugu Dan Genuk, Kota Semarang). *Jurnal Geodasi UNDIP*. 5(4):2337.

Budi Santosa, Ardian Umam. 2018. Data *Mining dan Big* Data *Analytics: Teori dan Implementasi Menggunakan Python & Apache Spark*. Yogyakarta: Media Pustaka.

Campbell, J.B., Wynne, R.H. 2011. *Introduction Remote Sensing. Fifth Edition*. New York: The Guildford Press.

Chen Zhang, Di Wu2 , Jiayu Sun, Guangyu Sun, Guojie Luo, and Jason Cong. 2016. *Energy-Efficient CNN Implementation on a Deeply Pipelined FPGA Cluster*.https://www.microsoft.com/en-us/research/publication/energy-efficient-cnn-implementation-on-a-deeply-pipelined-fpga-cluster/. [Diakses pada 18 Mei 2020].

Dega Dino Sanjaya Dimulya. 2016. Segmentasi Daerah Pemukiman Berbasis Orientasi Objek Menggunkan Citra Resolusi Tinggi (Worldview-2).

Deng, Zhipeng & Sun, Hao & Zhou, Shilin & Zhao, Juanping & Lei, Lin & Zou, Huanxin. 2018. Multi-Scale Object Detection In Remote Sensing Imagery With Convolutional Neural Networks. *ISPRS Journal of Photogrammetry and Remote Sensing*. 10(1016):15.

Derry Alamsyah, Dicky Pratama. 2019. Deteksi Ujung Jari menggunakan Faster RCNN dengan Arsitektur Inception V2 pada Citra Derau. Jurnal Sistem & Teknologi Informasi Komunikasi. 2(1):3.

Djoko Budiyanto Setyohadi, Felix Ade Kristiawan, Ernawati. 2017. Perbaikan Performansi Klasifikasi Dengan Preprocessing Iterative Partitioning Filter Algorithm. *Jurnal TELEMATIKA*. 14(2):12-20.

Erlyna Nour Arrofiqoh dan Harintaka. 2018. Implementasi Metode Convolutional Neural Network Untuk Klasifikasi Tanaman Pada Citra Resolusi Tinggi. *Jurnal Geomatika*. 24(2): 61-68.

Dodi Sudiana dan Elfa Diasmara.2008. Analisis Indeks Vegetasi Menggunakan Data Satelit NOAA/AVHRR dan TERRA/AQUA-MODIS. *Seminar on Intelligent Technology and Its Applications*.

Dony Satria, Mushthofa Mushthofa. 2013. *Perbandingan Metode Ekstraksi Ciri Histogram dan PCA untuk Mendeteksi Stoma pada Citra Penampang Daun Freycinet. Jurnal Ilmu Komputer dan AgroInformatika*. 2(1):20.

Fenti Endrianti, Wawan Setiawan, Yaya Wihardi. 2018. encatatan Kehadiran Otomatis di Ruang Kelas Berbasis Pengenalan Wajah MenggunakanMetode Convolutional Neural Network (CNN). *Jurnal JATIKOM*. 1(1): 40-44.

Enie Yuliani, Afifah Nur Aini, Chan Uswatun Khasanah. 2019. Perbandingan Jumlah *Epoch* Dan Steps Per *Epoch* Pada Convolutional Neural Network Untuk Meningkatkan Akurasi Dalam Klasifikasi Gambar. *Jurnal INFORMA Politeknik Indonusa Surakarta*. 5(3): 2442-7942.

Everaets, Jurgen. 2008. The Use of Unmaned Aerial Vehicles (UAVS) for Remote Sensing Mapping. *The Intenational Archives of the Photogrametry, Remote Sensing and Spatial Information Sciences*. 37(1):1182.

François Duval. 2018. *Deep Learning for Beginners: Concepts and Algorithms*. Tanpa Kota: CreateSpace Independent Publishing Platform

Jiawei Han Micheline Kamber Jian Pei. 2011. Data Mining Concepts and Techniques Third Edition. Waltham: Elsevier Inc.

Hadiwijaya Lesmana Salim, Restu Nur Afi Ati Dan Terry Louise Kepel. 2018. Pemetaan Dinamika Hutan Mangrove Menggunakan Drone Dan

Penginderaan Jauh Di P. Rambut, Kepulauan Seribu. *Jurnal Kelautan Nasional.* 13(2):89-97.

Hao Li , Pratik Chaudhari , Hao Yang, Michael Lam, Avinash Ravichandran, Rahul Bhotika, Stefano Soatto. 2020. Rethinking The Hyperparameters For Fine-Tuning. conference paper at ICLR 2020.

Hendro E. Prabowo, Tohari Ahmad. 2018. Peningkatan Kualitas Citra Stego Pada Adaptive Pixel Block Grouping Reduction Error Expansion Dengan Variasi Model Scanning Pada Pembentukan Kelompok Piksel. Jurnal Teknologi Informasi dan Ilmu Komputer. 5(2): 185-196.

Hermawan, A. 2006. Jaringan Syaraf Tiruan Teori dan Aplikasi. Yogyakarta: ANDI.

Indarto. 2014. *Teori dan Praktik Penginderaan Jauh (AV)*. Yogyakarta: ANDI

Nhat-Duy Nguyen, Tien Do , Thanh Duc Ngo, and Duy-Dinh Le. 2020. An Evaluation of Deep Learning Methods for Small Object Detection. *Journal of Electrical and Computer Engineering.* 2020(1)

Irma Amelia Dewi, Lisa Kristiana, Arsyad Ramadhan Darlis, Reza Fadilah Dwiputra. 2019. Deep Learning RetinaNet based Car Detection for Smart Transportation Network. *Jurnal Elkomika.* 7(3): 570-584.

Jeff Heaton. 2015. *Artificial Intelligence For Humans: Deep Learning And Neural networks Of Artificial Intelligence For Humans Series*. Chesterfield: Heaton Research Inc.

Jonghong Kim, Sangjun, Yoonnyun Kim2 and Minho Lee. 2018. Convolutional Neural network with Biologically Inspired Retinal Structure. *Journal of Procedia Computer Science.* 88(13):148.

Joseph Nelson. 2020. How to Create to a TFRecord *File* for Computer Vision and Object Detection. https://blog.roboflow.com/create-tfrecord/ [Diakses pada 20 Mei 2020].

K. C. Swain, S. J. Thomson, H. P. W. Jayasuriya. 2014. Adoption Of An Unmanned Helicopter For Low-Altitude Remote Sensing To Estimate Yield And Total Biomass Of A Rice Crop. Transactions of the ASABE. 53(1): 21-27.

Khairul Umam , Benny Sukma Negara. 2016. Deteksi Obyek Manusia Pada Basis Data Video Menggunakan Metode *Background* Subtraction Dan Operasi Morfologi. *Jurnal CoreIT.* 02(02):32.

Kusumanto, R., & Tompunu, A. N. (2011). Pengolahan Citra Digital Untuk Mendeteksi Obyek

Menggunakan Pengolahan Warna Model Normalisasi RGB. *Seminar Nasional Teknologi Informasi & Komunikasi Terapan*.

LeCun, Y., Bengio, Y., & Hinton, G. 2015. Deep Learning. *Nature Journal*, 521 (7533): 436.

Jan Wira Gotama Putra. 2019. *Pengenalan Konsep Pembelajaran Mesin dan Deep Learning*. Tokyo: Tokyo Institute of Technology.

M. Wirman Darmawan. 2009. Identifikasi Mutu Buah Mangga Arum Manis Berdasarkan Warna Menggunakan Image Processing dan JST. *Skripsi.* Yogyakarta: Fakultas Teknik, Universitas Gadjah Mada.

Marco Castelluccio, Giovanni Poggi, Carlo Sansone, Luisa Verdoliva. 2015. Land Use Classification in Remote Sensing Images by Convolutional Neural networks. https://arxiv.org/abs/1508.00092. [Diakses pada 18 Mei 2020].

Mesay Belete Bejiga, Abdallah Zeggada, and Farid Melgani. 2017. Convolutional Neural networks For Near Real-Time Object Detection From Uav Imagery In Avalanche Search And Rescue Operations. *Journal of Remote Sensing.* 9(2):693.

Muhammad Zufar dan Budi Setiyono. 2016. Convolutional Neural Networks untuk Pengenalan Wajah Secara Real-Time. *Jurnal Sains Dan Seni ITS*. 5(2): 2337-3520.

Novita Kurnia Ningrum, Defri Kurniawan, Novi Hendiyanto. 2018. Penerapan Ekstraksi Ciri Orde Satu Untuk Klasifikasi Tekstur Motif Batik Pesisir Dengan Algoritma Backpropagasi. *Jurnal SIMETRIS*. 8(2): 2252-4983.

Ilahiyah, S., & Nilogiri, A. 2018. Implementasi Deep Learning Pada IdentifikasiJenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network. *Jurnal Sistem & Teknologi Informasi Indonesia*. 3(2): 49–56.

Iswanto, H. 2002. *Membuat Mangga Tiga Rasa* Jakarta: PTAgromediia Pustaka

Prahasta, E. 2002. *Pemmrograman Web Mencakup: HTML, CSS, Java Script & PHP*. Yogyakarta: Andi Offset.

Purwadhi, Sri H. 2001. *Interpretasi Gambar Digital*. Jakarta: Grasindo.

Ravi Sripada, Ronnie Heiniger, Jeff White, Alan Meijer. 2006. Aerial Color Infrared Photography for Determining Early In-Season Nitrogen Requirements in Corn. *Agronomy Journal* . 98(4):1.

Richard Szelski. 2010. *Computer Vision: Algorithms and Applications*. Washington: Springer.

Ridho Aji Pangestu, Basuki Rahmat, Fetty Tri Anggraeny. 2020. Implementasi Algoritma Cnn Untuk Klasifikasi Citra Lahan Dan Perhitungan Luas. *Jurnal Informatika dan Sistem Informasi*. 1(1)

Royani Nurfita, Gunawan Ariyanto. 2018. Implementasi Deep Learning Berbasis Tensorflow Untuk Pengenalan Sidik Jari. *Emitor: Jurnal Teknik Elektro*. 18(22):27.

Saban Öztürka, Bayram Akdemir. 2018. Effects of Histopathological Image Preprocessing on Convolutional Neural Networks. *Journal of Elsevier*. 132(10):1016

Safitri, A. A. 2012. Studi Pembuatan fruit leather mangga-rosela. *Skripsi*. Makasar: Fakultas Pertanian, Universitas Hasanudin.

Saleh Albelwi, Ausif Mahmood. 2017. A Framework for Designing the Architectures of Deep Convolutional Neural networks. *Journal of Entropy*. 19(6):4.

Samer Hijazi, Rishi Kumar, and Chris Rowen. 2015. Using Convolutional Neural networks for Image Recognition. https://ip.cadence.com/uploads/901/CNNwp. [Diakses pada 27 Januari 2020].

Saripuddin Madenda. 2015. *Pengolahan Citra dan Video Digital*. Jakarta Penerbit: Erlangga.

Shaoqing Ren, Kaiming He Ross, Girshick Jian Sun. 2015 Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.

Siwi Prihatiningsih, Nadhiranisa Shafiy M, Feni Andriani, Nurma Nugraha. Analisa Performa Pengenalan Tulisan Tangan Angka Berdasarkan Jumlah *Epoch* Menggunakan Metode Convolutional Neural Network. *Jurnal Ilmiah Teknologi dan Rekayasa*. 24(1):1934.

Suhandy, Diding, U. Ahmad. 2003. *Pengembangan Image Processing untuk Menduga kemasakan Buah Manggis Segar*. *Buletin Keteknikan Pertanian*.17(02):10.

Suparman. 2007. *Bercocok Tanam Mangga*. Jakarta: Azka Press

Tita Karlita, I Made Gede Sunarya, Joko Priambodo, Rika Rokhana, Eko Mulyanto Yuniarno, I Ketut Eddy Purnama, Mauridhi Hery Purnomo. 2019. Deteksi Region of Interest Tulang pada Citra B-mode secara Otomatis Menggunakan Region Proposal Networks. *Jurnal JNTETI*. 8(2): 2301-4156.

Peng, Jie & Kang, Shuai & Ning, Zhengyuan & Deng, Hangxia & Shen, Jingxian & Xu, Yikai & Zhang, Jing & Zhao, Wei & Li, Xinling & Gong, Wuxing & Huang, Jinhua & Liu, Li. 2019. Residual Convolutional Neural network for predicting response of transarterial. *Journal of European Radiology*. 30(1):4

Vedaldi, A., Lenc, K. 2015. *MatConvNet: Convolutional Neural networks for MATLAB*. New York: ACM.

Wahyudi, N., Suhartono, V., & Pramunendar, R. A. 2018. *Background* Subtraction Berbasis Self Organizing Map Untuk Deteksi Objek Bergerak. Information. *System and Informatics Journal*.1(1): 42–51.

Wayan Suartika Eka Putra, Arya Yudhi Wijaya, dan Rully Soelaiman. 2015. Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) Pada Caltech 101. *Jurnal Teknik ITS*. 5(1): 2337-3539.

Yasirun Azzakirot. 2018. Perhitungan Pohon Kelapa Sawit Dengan Mengindentifikasi Pohon Menggunakan Algortitma Haar-Cascade Clasifier. *Skripsi*. Medan: Fakultas Ilmu Komputer Dan Teknologi Informasi, Universitas Sumatera Utara.

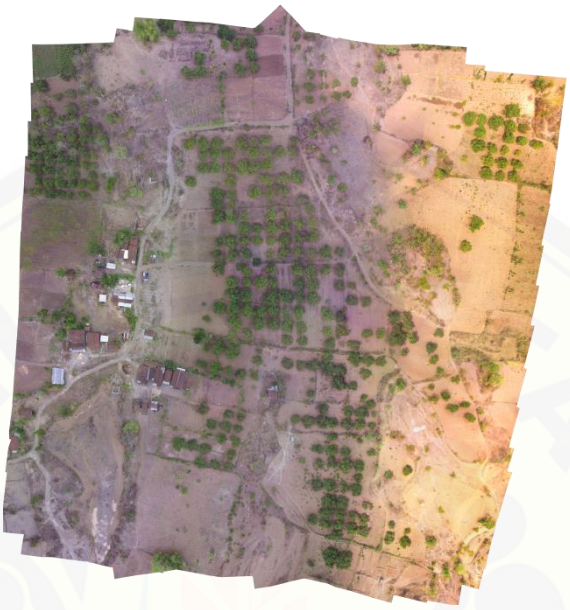
Yaya Heryadi, Edy Irwansyah. 2020. *Deep Learning: Aplikasinya di Bidang Geospasial*. Depok : AWI Technology Press

Yoga Oktavianto , Sunaryo, dan Agus Suryanto. 2015. Karakterisasi Tanaman Mangga (Mangifera Indica L.) Cantek, Ireng, Empok, Jempol Di Desa Tiron, Kecamatan Banyakan Kabupaten Kediri. *Jurnal Produksi Tanaman*. 3(2):91-97.

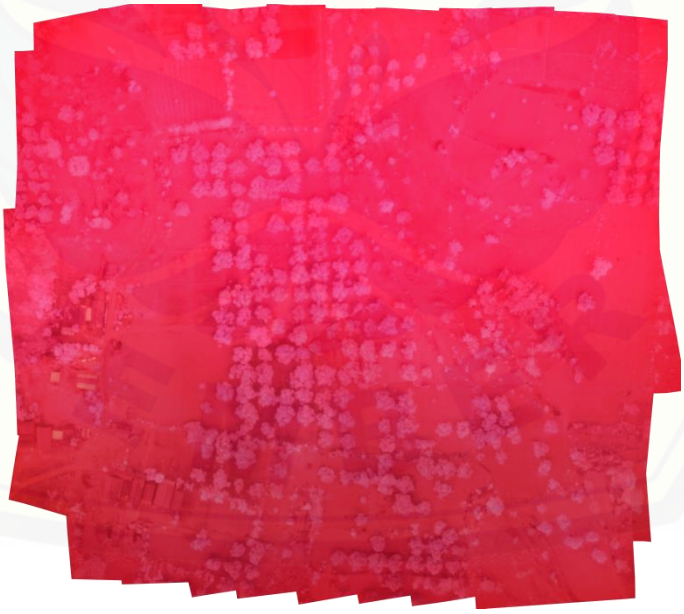
Zainudin, Z & Shamsuddin, S & Hasan, S. 2019. Deep Learning Layer Convolutional Neural Network (CNN) Scheme for Cancer Image. IOP Conference Series: Materials Science and Engineering. 551 (2019): 012039.

LAMPIRAN

Lampiran 4.1 Data Hasil Akuisisi



Data RGB

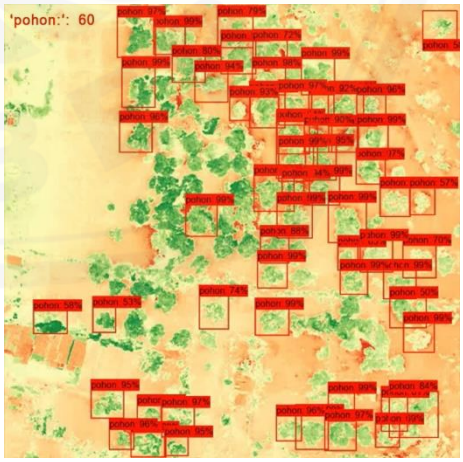


Data NIR

Lampiran 4.2 Data Perhitungan Pohon Mangga Aktual

Lampiran 4.3 Data Perhitungan Pohon Mangga Prediksi Model Inception V2

(a)

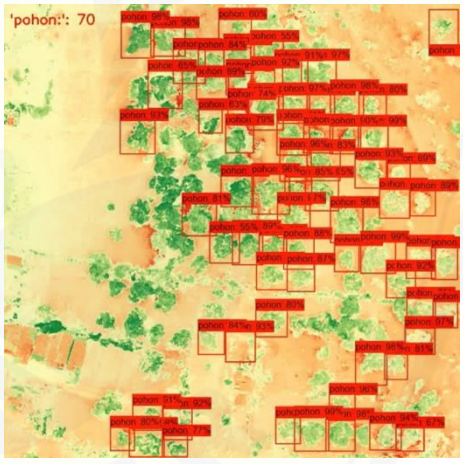


(b)

(c) (d)

Hasil perhitungan secara prediksi model *Inception V2* (a) RGB; (b) S2; VARI; (c) GRVI; dan (d) GNDVI

Lampiran 4.4 Data Perhitungan Pohon Mangga Prediksi Model *ResNet-50*

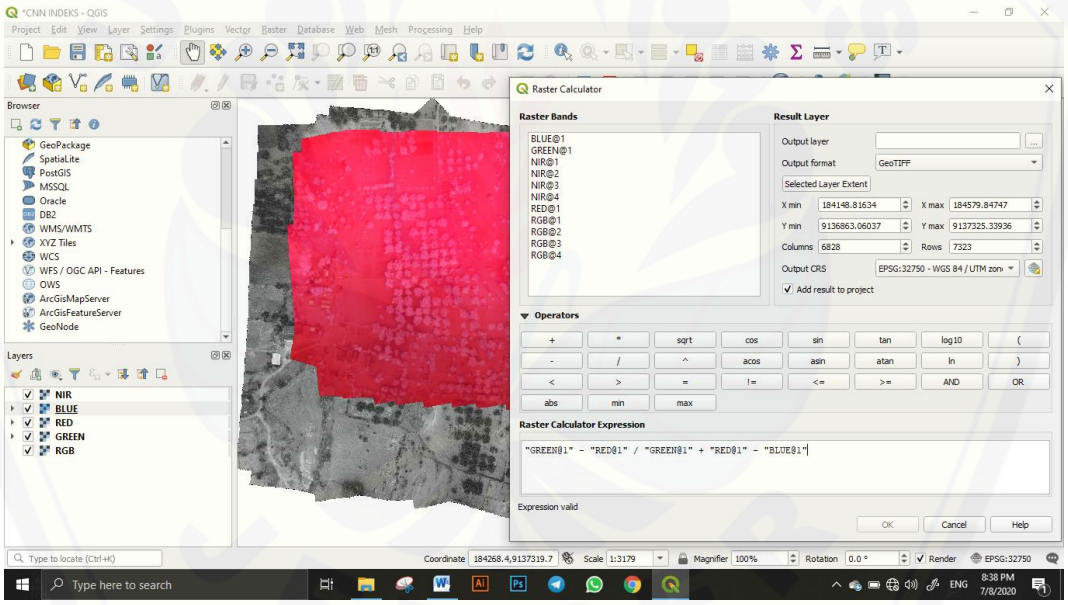


(a) (b)

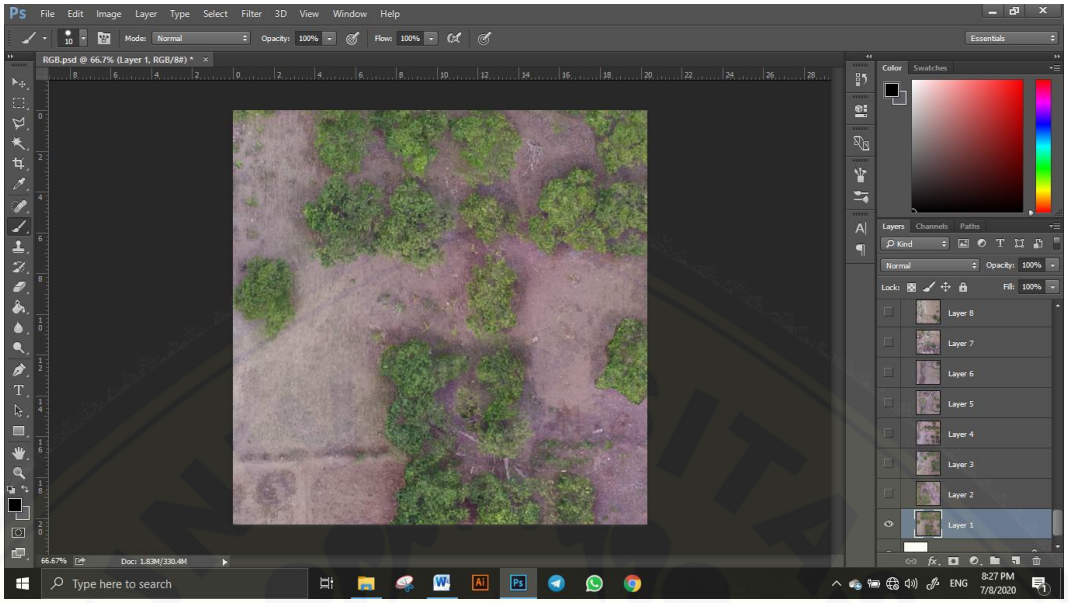
(c)

(d)

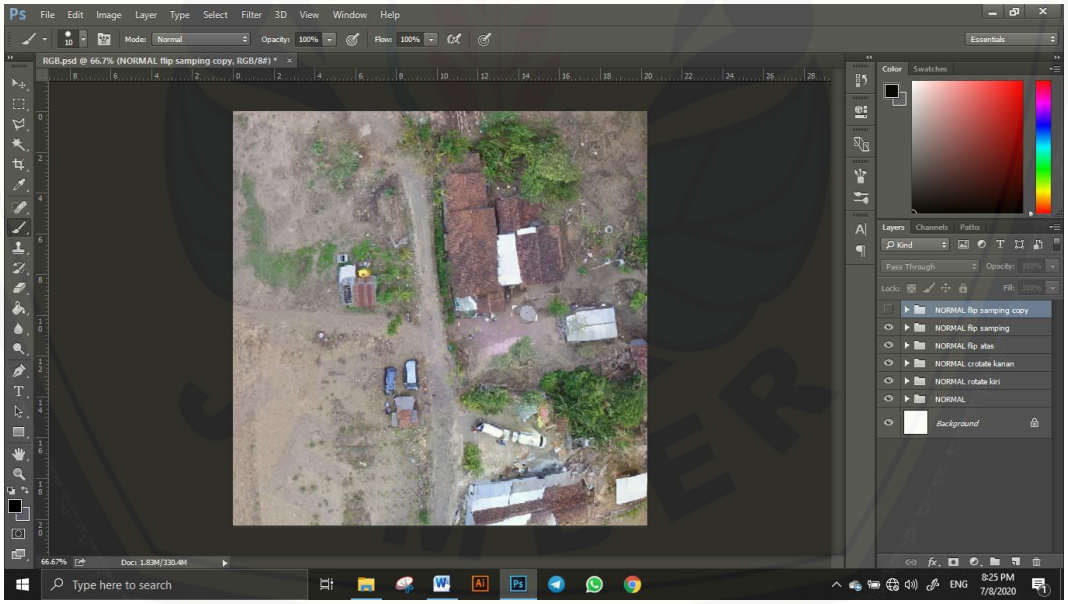
Lampiran 4.5 Proses indeks vegetasi



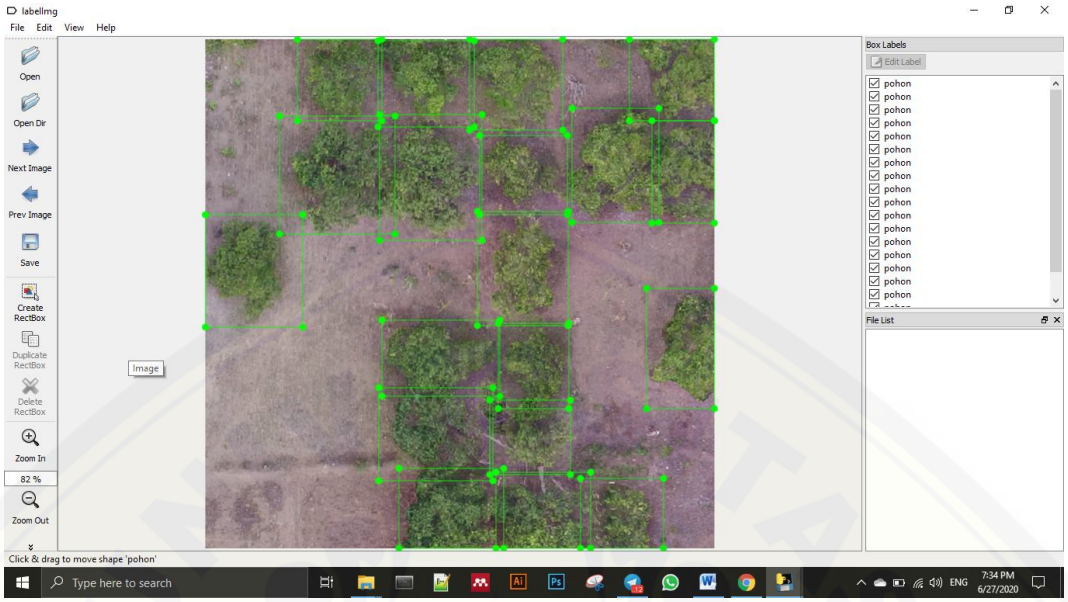
Lampiran 4.6 Proses *crop*



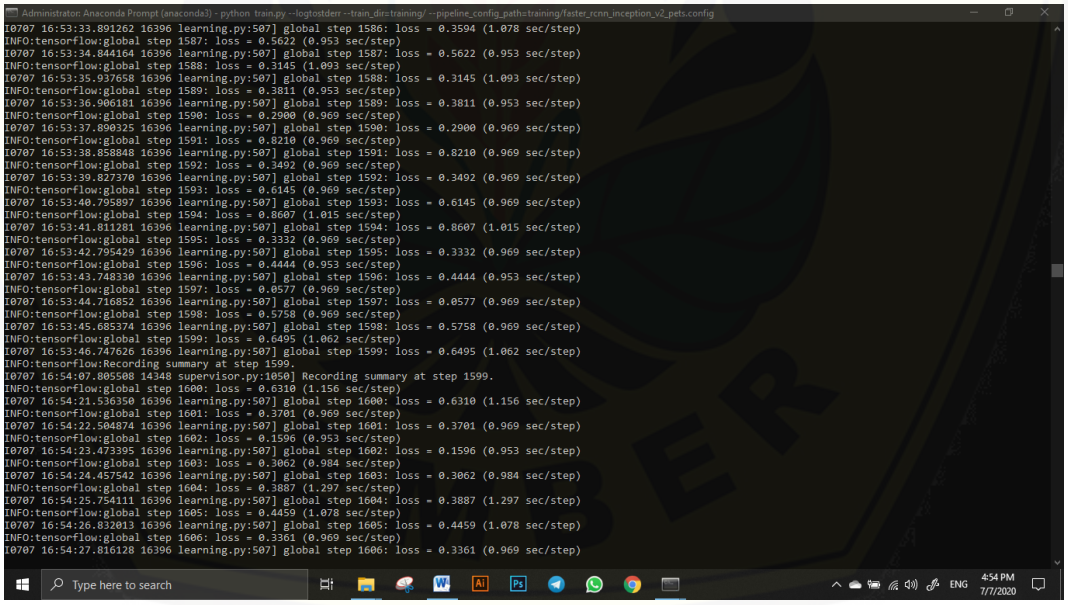
Lampiran 4.7 Proses augementasi data



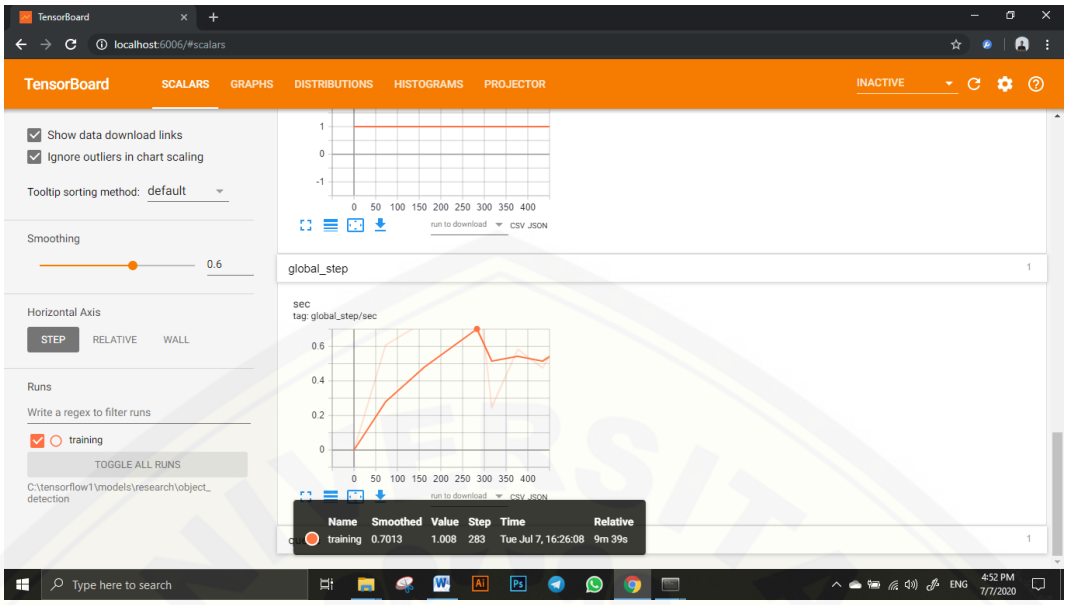
Lampiran 4.8 Proses anotasi data



Lampiran 4.9 Proses *Training* Data



Proses *training* data di Anaconda Prompt



Proses *training* data di Tensorboard

Lampiran 4.10 *Script resize*

```
import cv2

img = cv2.imread('RGB_0001.jpg', cv2.IMREAD_UNCHANGED)
print('Original Dimensions : ',img.shape)

scale_percent = 80
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)

print('Resized Dimensions : ',resized.shape)
input = 'RGB_0001.jpg'
cv2.imwrite(input, resized)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Lampiran 4.11 *Script resize*

```
from __future__ import division
from __future__ import print_function
from __future__ import absolute_import

import os
import io
import pandas as pd
import tensorflow as tf

from PIL import Image
from object_detection.utils import dataset_util
from collections import namedtuple, OrderedDict

flags = tf.app.flags
flags.DEFINE_string('csv_input', '', 'Path to the CSV input')
flags.DEFINE_string('image_dir', '', 'Path to the image directory')
flags.DEFINE_string('output_path', '', 'Path to output TFRecord')
FLAGS = flags.FLAGS


# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'pohon':
        return 1
    else:
        return 0


def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
```

```
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]


def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path, '{}'.format(group.filename)), 'rb') as fid:
        encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size

    filename = group.filename.encode('utf8')
    image_format = b'jpg'
    xmins = []
    xmaxs = []
    ymins = []
    ymaxs = []
    classes_text = []
    classes = []

    for index, row in group.object.iterrows():
        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)
        classes_text.append(row['class'].encode('utf8'))
        classes.append(class_text_to_int(row['class']))

    tf_example = tf.train.Example(features=tf.train.Features(feature={
        'image/height': dataset_util.int64_feature(height),
```

```
        'image/width': dataset_util.int64_feature(width),
        'image/filename': dataset_util.bytes_feature(filename),
        'image/source_id': dataset_util.bytes_feature(filename),
        'image/encoded': dataset_util.bytes_feature(encoded_jpg),
        'image/format': dataset_util.bytes_feature(image_format),
        'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
        'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
        'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
        'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
        'image/object/class/text': dataset_util.bytes_list_feature(classes_text),
        'image/object/class/label': dataset_util.int64_list_feature(classes),
    }))
    return tf_example


def main(_):
    writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
    path = os.path.join(os.getcwd(), FLAGS.image_dir)
    examples = pd.read_csv(FLAGS.csv_input)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())

    writer.close()
    output_path = os.path.join(os.getcwd(), FLAGS.output_path)
    print('Successfully created the TFRecords: {}'.format(output_path))


if __name__ == '__main__':
    tf.app.run()
```

Lampiran 4.12 *Script fine tuning*

```
model {
  faster_rCNN {
    num_classes: 1
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rCNN_resnet50'
      first_stage_features_stride : 16
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride : 16
        width_stride : 16
      }
    }
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        truncated_normal_initializer {
```

```
      stddev: 0.01
    }
   }
}
first_stage_nms_score_threshold: 0.0
first_stage_nms_iou_threshold: 0.7
first_stage_max_proposals: 300
first_stage_localization_loss_weight: 2.0
first_stage_objectness_loss_weight: 1.0
initial_crop_size: 14
maxpool_kernel_size: 4
maxpool_stride: 4
second_stage_box_predictor {
  mask_rCNN_box_predictor {
    use_dropout: true
    dropout_keep_probability: 0.05
    fc_hyperparams {
      op: FC
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        variance_scaling_initializer {
          factor: 1.0
          uniform: true
          mode: FAN_AVG
        }
      }
    }
```

```
    }
   }
   second_stage_post_processing {
    batch_non_max_suppression {
     score_threshold: 0.0
     iou_threshold: 0.6
     max_detections_per_class: 100
     max_total_detections: 300
    }
    score_converter: SOFTMAX
   }
   second_stage_localization_loss_weight: 2.0
   second_stage_classification_loss_weight: 1.0
  }
}

train_config: {
 batch_size: 1
 optimizer {
  momentum_optimizer: {
   learning_rate: {
    manual_step_learning_rate {
     initial_learning_rate: 0.0003
     schedule {
      step: 900000
      learning_rate: .00003
     }
     schedule {
      step: 1200000
      learning_rate: .000003
     }
```

```
      }
     }
     momentum_optimizer_value: 0.9
    }
    use_moving_average: false
   }
   gradient_clipping_by_norm: 10.0
   fine_tune_checkpoint:
"C:/Tensorflow1/models/research/object_detection/faster_rCNN_resnet50_coco_2
018_01_28/model.ckpt"
   from_detection_checkpoint: true
   load_all_detection_checkpoint_vars: true
   num_steps: 10000
   data_augmentation_options {
    random_horizontal_flip {
    }
   }
   max_number_of_boxes: 50
}

train_input_reader: {
  tf_record_input_reader {
    input_path: "C:/Tensorflow1/models/research/object_detection/train.record"
  }
  label_map_path:
"C:/Tensorflow1/models/research/object_detection/training/labelmap.pbtxt"
}

eval_config: {
  metrics_set: "coco_detection_metrics"
  num_examples: 1101
```

```
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "C:/Tensorflow1/models/research/object_detection/test.record"
  }
  label_map_path:
"C:/Tensorflow1/models/research/object_detection/training/labelmap.pbtxt"
  shuffle: false
  num_readers: 1
}
```

Lampiran 4.13 *Script train data*

```
import functools
import json
import os
import tensorflow as tf
from tensorflow.contrib import framework as contrib_framework

from object_detection.builders import dataset_builder
from object_detection.builders import graph_rewriter_builder
from object_detection.builders import model_builder
from object_detection.legacy import trainer
from object_detection.utils import config_util

tf.logging.set_verbosity(tf.logging.INFO)

flags = tf.app.flags
flags.DEFINE_string('master', '', 'Name of the TensorFlow master to use.')
flags.DEFINE_integer('task', 0, 'task id')
flags.DEFINE_integer('num_clones', 1, 'Number of clones to deploy per worker.')
```

```
flags.DEFINE_boolean('clone_on_cpu', False,
                     'Force clones to be deployed on CPU.  Note that even if '
                     'set to False (allowing ops to run on gpu), some ops may '
                     'still be run on the CPU if they have no GPU kernel.')
flags.DEFINE_integer('worker_replicas', 1, 'Number of worker+trainer '
                     'replicas.')
flags.DEFINE_integer('ps_tasks', 0,
                     'Number of parameter server tasks. If None, does not use '
                     'a parameter server.')
flags.DEFINE_string('train_dir', '',
                    'Directory to save the checkpoints and training summaries.')

flags.DEFINE_string('pipeline_config_path', '',
                    'Path to a pipeline_pb2.TrainEvalPipelineConfig config '
                    'file. If provided, other configs are ignored')

flags.DEFINE_string('train_config_path', '',
                    'Path to a train_pb2.TrainConfig config file.')
flags.DEFINE_string('input_config_path', '',
                    'Path to an input_reader_pb2.InputReader config file.')
flags.DEFINE_string('model_config_path', '',
                    'Path to a model_pb2.DetectionModel config file.')

FLAGS = flags.FLAGS

@contrib_framework.deprecated(None, 'Use object_detection/model_main.py.')
def main(_):
  assert FLAGS.train_dir, '`train_dir` is missing.'
  if FLAGS.task == 0: tf.gfile.MakeDirs(FLAGS.train_dir)
  if FLAGS.pipeline_config_path:
    configs = config_util.get_configs_from_pipeline_file(
```

```
    FLAGS.pipeline_config_path)
 if FLAGS.task == 0:
  tf.gfile.Copy(FLAGS.pipeline_config_path,
                os.path.join(FLAGS.train_dir, 'pipeline.config'),
                overwrite=True)
else:
 configs = config_util.get_configs_from_multiple_files(
    model_config_path=FLAGS.model_config_path,
    train_config_path=FLAGS.train_config_path,
    train_input_config_path=FLAGS.input_config_path)
 if FLAGS.task == 0:
  for name, config in [('model.config', FLAGS.model_config_path),
                       ('train.config', FLAGS.train_config_path),
                       ('input.config', FLAGS.input_config_path)]:
    tf.gfile.Copy(config, os.path.join(FLAGS.train_dir, name),
                  overwrite=True)

model_config = configs['model']
train_config = configs['train_config']
input_config = configs['train_input_config']

model_fn = functools.partial(
  model_builder.build,
  model_config=model_config,
  is_training=True)

def get_next(config):
 return dataset_builder.make_initializable_iterator(
    dataset_builder.build(config)).get_next()

create_input_dict_fn = functools.partial(get_next, input_config)
```

```
env = json.loads(os.environ.get('TF_CONFIG', '{}'))
cluster_data = env.get('cluster', None)
cluster = tf.train.ClusterSpec(cluster_data) if cluster_data else None
task_data = env.get('task', None) or {'type': 'master', 'index': 0}
task_info = type('TaskSpec', (object,), task_data)

# Parameters for a single worker.
ps_tasks = 0
worker_replicas = 1
worker_job_name = 'lonely_worker'
task = 0
is_chief = True
master = ''

if cluster_data and 'worker' in cluster_data:
  # Number of total worker replicas include "worker"s and the "master".
  worker_replicas = len(cluster_data['worker']) + 1
if cluster_data and 'ps' in cluster_data:
  ps_tasks = len(cluster_data['ps'])

if worker_replicas > 1 and ps_tasks < 1:
  raise ValueError('At least 1 ps task is needed for distributed training.')

if worker_replicas >= 1 and ps_tasks > 0:
  # Set up distributed training.
  server = tf.train.Server(tf.train.ClusterSpec(cluster), protocol='grpc',
                           job_name=task_info.type,
                           task_index=task_info.index)
  if task_info.type == 'ps':
    server.join()
```

```
    return

  worker_job_name = '%s/task:%d' % (task_info.type, task_info.index)
  task = task_info.index
  is_chief = (task_info.type == 'master')
  master = server.target

graph_rewriter_fn = None
if 'graph_rewriter_config' in configs:
  graph_rewriter_fn = graph_rewriter_builder.build(
      configs['graph_rewriter_config'], is_training=True)

trainer.train(
    create_input_dict_fn,
    model_fn,
    train_config,
    master,
    task,
    FLAGS.num_clones,
    worker_replicas,
    FLAGS.clone_on_cpu,
    ps_tasks,
    worker_job_name,
    is_chief,
    FLAGS.train_dir,
    graph_hook_fn=graph_rewriter_fn)


if __name__ == '__main__':
  tf.app.run()
```

Lampiran 4.14 *Script* deteksi dan hitung jumlah objek

```
import os
import cv2
import numpy as np
import Tensorflow as tf
import sys

sys.path.append("..")
from utils import label_map_util
from utils import visualization_utils as vis_util

MODEL_NAME = 'inference_graph'
IMAGE_NAME = 'test1.jpg'

CWD_PATH = os.getcwd()
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

PATH_TO_LABELS = os.path.join(CWD_PATH,'training','labelmap.pbtxt')
PATH_TO_IMAGE = os.path.join(CWD_PATH,IMAGE_NAME)
NUM_CLASSES = 6

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
```

```
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

    sess = tf.Session(graph=detection_graph)

image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')

num_detections = detection_graph.get_tensor_by_name('num_detections:0')

image = cv2.imread(PATH_TO_IMAGE)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image_expanded = np.expand_dims(image_rgb, axis=0)

 (boxes, scores, classes, num) = sess.run(
    [detection_boxes, detection_scores, detection_classes, num_detections],
    feed_dict={image_tensor: image_expanded})

counting_mode = vis_util.visualize_boxes_and_labels_on_image_array(
    image,
    np.squeeze(boxes),
    np.squeeze(classes).astype(np.int32),
    np.squeeze(scores),
    category_index,
    use_normalized_coordinates=True,
    line_thickness=8,
    min_score_thresh=0.60)
if(len(counting_mode) == 0):
```

```
            print (counting_mode)
            cv2.putText(input_frame, "...", (10, 35), font, 0.8,
(0,0,255),2,cv2.FONT_HERSHEY_SIMPLEX)
        else:
            print (counting_mode)
            cv2.putText(input_frame, counting_mode, (10, 35), font, 0.8,
(0,0,255),2,cv2.FONT_HERSHEY_SIMPLEX)

        input_movie.write(input_frame)
        print ("writing frame")

cv2.imshow('Object detector', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Lampiran 4.15 Data *learning rate*

No.	*Epoch*	*ResNet-50*				*Inception V2*			
1	0	0.0594	0.0939	0.0925	0.0985	0.0706	0.0866	0.0993	0.0919
2	100	0.0471	0.0940	0.6106	2.4889	0.1974	0.3585	1.5048	1.2302
3	200	0.0471	0.0946	0.3261	8.3631	0.0498	3.2306	0.5388	0.5911
4	300	0.0476	0.9521	0.2035	1.9621	0.0180	0.0824	0.2898	0.9866
5	400	0.0476	0.0955	0.2617	1.4655	0.1241	0.2247	0.5762	0.3283
6	500	0.0602	0.0957	0.0914	0.0988	0.0731	0.0876	0.0959	0.0852
7	600	0.0610	0.0961	0.1982	0.8956	0.0177	0.1018	0.5641	0.5553
8	700	0.0618	0.0969	0.2552	0.9199	0.0010	0.0989	0.0970	0.1856
9	800	0.0615	0.0975	0.1782	0.8019	0.0478	0.1573	0.4296	0.3436
10	900	0.0612	0.0979	0.1654	0.9448	0.0337	0.1788	0.1822	0.0334
11	1000	0.0614	0.0989	0.0905	0.0989	0.0337	0.0891	0.0905	0.0788
12	1100	0.0598	0.0323	0.5543	1.0187	0.0174	0.1604	0.2806	0.2044
13	1200	0.0580	0.0306	0.1078	0.5642	0.0016	0.0287	0.1053	0.1779
14	1300	0.0575	0.0342	0.1565	0.5370	0.0349	0.1311	0.4224	0.1814
15	1400	0.0567	0.0984	0.2972	0.6642	0.0174	0.0662	0.2756	0.3060
16	1500	0.0558	0.0856	0.0895	0.0989	0.0778	0.0850	0.0952	0.0786
17	1600	0.0534	0.0243	0.1325	0.5842	0.0116	0.0243	0.2378	0.2697
18	1700	0.0548	0.0415	0.1364	0.6628	0.0076	0.0563	0.3715	0.2305
19	1800	0.0545	0.0494	0.1411	0.4708	0.0280	0.0194	0.1371	0.1556

20	1900	0.0544	0.0573	0.2109	0.8089	0.0164	0.1499	0.0967	0.3106
21	2000	0.0518	0.0808	0.0887	0.0999	0.0795	0.0808	0.1007	0.0801
22	2100	0.0530	0.0502	0.3648	0.7014	0.0066	0.0696	0.1543	0.1589
23	2200	0.0511	0.0349	0.1588	0.2225	0.0210	0.0070	0.1692	0.2250
24	2300	0.0502	0.0534	0.1801	0.3158	0.0034	0.0194	0.0502	0.1847
25	2400	0.0498	0.0282	0.1402	0.3752	0.0066	0.0344	0.1090	0.2269
26	2500	0.0482	0.0829	0.0855	0.0951	0.0704	0.0822	0.0956	0.0825
27	2600	0.0482	0.0262	0.1234	0.4638	0.0033	0.0827	0.1133	0.1963
28	2700	0.0480	0.0356	0.1226	0.3356	0.0063	0.0226	0.0867	0.2261
29	2800	0.0479	0.0293	0.1301	0.3627	0.0035	0.0388	0.1190	0.2332
30	2900	0.0478	0.0224	0.1011	0.2874	0.0063	0.0204	0.1560	0.2473
31	3000	0.0481	0.0840	0.0836	0.0904	0.0614	0.0831	0.0906	0.0860
32	3100	0.0474	0.0555	0.1366	0.4296	0.0148	0.0560	0.2177	0.1845
33	3200	0.0476	0.0532	0.2213	0.6871	0.0023	0.0497	0.0999	0.0587
34	3300	0.0476	0.0431	0.2309	1.4462	0.0081	0.6508	0.1397	0.1317
35	3400	0.0480	0.0301	0.1071	0.3985	0.0081	0.0157	0.1746	0.1469
36	3500	0.0482	0.0847	0.0830	0.0859	0.0579	0.0845	0.0877	0.0847
37	3600	0.0483	0.0368	0.1371	0.2651	0.0041	0.0230	0.1478	0.1955
38	3700	0.0486	0.0221	0.0798	0.6094	0.0148	0.0826	0.1321	0.1301
39	3800	0.0490	0.0504	0.2332	0.4331	0.0139	0.0838	0.0885	0.0583
40	3900	0.0491	0.0380	0.2200	1.5599	0.0017	0.6502	0.0916	0.2783
41	4000	0.0492	0.0859	0.0808	0.0834	0.0546	0.0869	0.0842	0.0832
42	4100	0.0493	0.0611	0.2409	0.2432	0.0006	0.0062	0.0799	0.2049
43	4200	0.0496	0.0411	0.1287	0.4737	0.0102	0.0269	0.1562	0.1483
44	4300	0.0498	0.0333	0.1306	0.2357	0.0005	0.0158	0.0307	0.2637
45	4400	0.0499	0.0359	0.1285	0.3602	0.0309	0.0457	0.2481	0.1223
46	4500	0.0513	0.0744	0.0783	0.0854	0.0508	0.0824	0.0807	0.0824
47	4600	0.0514	0.0769	0.0312	0.6184	0.0093	0.0245	0.1216	0.1001
48	4700	0.0519	0.0213	0.0983	1.4457	0.0031	0.6551	0.1687	0.0476
49	4800	0.0522	0.0151	0.0663	0.5192	0.0052	0.0963	0.0916	0.1982
50	4900	0.0532	0.0547	0.2643	0.1193	0.0058	0.0198	0.1788	0.1578
51	5000	0.0533	0.0650	0.0756	0.0835	0.0530	0.0780	0.0797	0.0815
52	5100	0.0530	0.0212	0.1078	0.2642	0.0000	0.0367	0.0101	0.0305
53	5200	0.0530	0.0283	0.1616	0.7617	0.0003	0.0547	0.0162	0.1397
54	5300	0.0540	0.0175	0.1197	0.5696	0.0076	0.0643	0.0257	0.1132
55	5400	0.0524	0.3200	0.0936	0.1070	0.0046	0.0060	0.1224	0.5579
56	5500	0.0514	0.0682	0.0675	0.0805	0.0554	0.0796	0.0791	0.0808
57	5600	0.0522	0.0530	0.1959	0.5015	0.0075	0.0913	0.0984	0.1566
58	5700	0.0520	0.0153	0.0849	0.3296	0.0181	0.0357	0.3142	0.1059
59	5800	0.0520	0.0228	0.0987	0.2091	0.0074	0.0266	0.1423	0.0492
60	5900	0.0510	0.0287	0.1289	0.2562	0.0048	0.0526	0.0743	0.3895
61	6000	0.0479	0.0709	0.0700	0.0783	0.0582	0.0805	0.0790	0.0792

62	6100	0.0508	0.0344	0.1785	0.0755	0.0028	0.0076	0.1366	0.1712
63	6200	0.0498	0.0119	0.0859	0.0614	0.0038	0.0018	0.1440	0.1738
64	6300	0.0482	0.3326	0.2121	0.3008	0.0123	0.0130	0.1722	0.0157
65	6400	0.0470	0.1556	0.0367	0.1968	0.0029	0.0020	0.1213	0.0220
66	6500	0.0436	0.0686	0.0736	0.0765	0.0607	0.0820	0.0782	0.0784
67	6600	0.0437	0.0160	0.0410	0.3035	0.0009	0.0472	0.0288	0.1149
68	6700	0.0425	0.0359	0.1476	0.1162	0.0020	0.0074	0.0284	0.1252
69	6800	0.0404	0.0589	0.0293	0.1684	0.0018	0.0192	0.1086	0.0064
70	6900	0.0398	0.0247	0.1090	0.1846	0.0021	0.0061	0.1148	0.1055
71	7000	0.0386	0.0661	0.0750	0.0767	0.0554	0.0761	0.0727	0.0780
72	7100	0.0380	0.0242	0.1017	0.1756	0.0052	0.0091	0.0748	0.1638
73	7200	0.0378	0.0328	0.1550	0.1383	0.0044	0.0017	0.1629	0.3927
74	7300	0.0375	0.0337	0.1924	0.2504	0.0014	0.0606	0.0888	0.0891
75	7400	0.0373	0.0329	0.1458	0.3895	0.0031	0.0459	0.0972	0.1582
76	7500	0.0364	0.0636	0.0786	0.0788	0.0495	0.0702	0.0658	0.0769
77	7600	0.0366	0.0291	0.1185	1.5203	0.1277	0.5299	0.3213	0.0510
78	7700	0.0368	0.0307	0.0922	0.1930	0.0006	0.0079	0.0149	0.0921
79	7800	0.0370	0.0401	0.2277	0.2675	0.0015	0.0282	0.0480	0.1131
80	7900	0.0371	0.0479	0.0205	0.0724	0.0056	0.0030	0.1722	0.1379
81	8000	0.0371	0.0620	0.0805	0.0809	0.0437	0.0658	0.0608	0.0764
82	8100	0.0389	0.0325	0.1721	0.1383	0.0026	0.0113	0.0542	0.0243
83	8200	0.0390	0.0399	0.3028	0.2927	0.0014	0.0323	0.0367	0.0607
84	8300	0.0394	0.0280	0.0672	1.1013	0.0004	0.3624	0.0228	0.0682
85	8400	0.0395	0.0208	0.0803	0.2463	0.0041	0.0104	0.1206	0.0742
86	8500	0.0385	0.0602	0.0739	0.0820	0.0375	0.0630	0.0625	0.0759
87	8600	0.0410	0.0582	0.0119	0.2868	0.0056	0.0093	0.0274	0.0151
88	8700	0.0417	0.2379	0.1102	0.1012	0.0032	0.0043	0.0569	0.0256
89	8800	0.0421	0.0302	0.0090	0.1325	0.0003	0.0101	0.0204	0.0314
90	8900	0.0428	0.0946	0.0689	0.2503	0.0016	0.0104	0.0899	0.0110
91	9000	0.0405	0.0376	0.0606	0.0839	0.0309	0.0605	0.0650	0.0753
92	9100	0.0419	0.0106	0.0410	0.0999	0.0001	0.0019	0.0126	0.0787
93	9200	0.0390	0.0299	0.1045	0.1605	0.0015	0.0116	0.0631	0.0466
94	9300	0.0374	0.0248	0.1351	0.0768	0.0018	0.0125	0.0180	0.0734
95	9400	0.0371	0.0118	0.0393	0.1323	0.0163	0.0017	0.1584	0.0859
96	9500	0.0343	0.0396	0.0646	0.0819	0.0320	0.0565	0.0622	0.0749
97	9600	0.0337	0.0144	0.0796	0.0710	0.0035	0.0011	0.0771	0.0579
98	9700	0.0310	0.0181	0.0887	0.5735	0.0033	0.0254	0.0566	0.0705
99	9800	0.0283	0.0683	0.0454	1.1754	0.0017	0.4305	0.0347	0.0472
100	9900	0.0263	0.0342	0.1751	0.1694	0.0009	0.0078	0.0508	0.0447
101	10000	0.0288	0.0404	0.0673	0.0786	0.0325	0.0540	0.0587	0.0744

Lampiran 4.16 Data *global step*

No.	*Epoch*	*ResNet-50*				*Inception V2*			
1	0	0	0	0	0	0	0	0	0
2	100	0.8171	0.7833	0.4786	0.4657	0.3169	0.3357	0.3167	0.3003
3	200	0.9999	1.0030	0.9916	0.4917	0.3837	0.5157	0.4215	0.5324
4	300	1.0166	0.9816	0.9834	0.2917	0.5083	0.3832	0.5315	0.5175
5	400	0.6501	0.8166	1.0067	0.8081	0.3833	0.4670	0.5141	0.5313
6	500	0.9833	0.9751	0.1612	0.6169	0.5235	0.5257	0.4252	0.2593
7	600	0.9916	1.0250	0.7432	0.9560	0.4257	0.5001	0.4997	0.5242
8	700	1.0167	0.9830	0.7164	0.9508	0.5257	0.5152	0.2314	0.5176
9	800	1.0167	0.9831	0.9253	0.9752	0.5166	0.3755	0.3870	0.5233
10	900	0.7917	0.7087	0.9412	0.9252	0.5164	0.5169	0.5070	0.5173
11	1000	0.9914	0.9659	0.6838	0.7007	0.5246	0.5072	0.3007	0.4173
12	1100	0.9586	1.0092	0.9583	1.0001	0.3666	0.5265	0.5079	0.5038
13	1200	0.9666	0.9999	0.9495	0.9833	0.5258	0.5160	0.5172	0.5189
14	1300	0.9582	0.9917	0.9838	0.9750	0.5020	0.3339	0.4906	0.5191
15	1400	0.7167	0.7406	0.9416	0.9999	0.5125	0.5165	0.4924	0.5165
16	1500	0.9500	0.9930	0.7515	0.4667	0.4852	0.5160	0.3836	0.3832
17	1600	0.9453	1.0251	0.9588	0.9250	0.4250	0.5009	0.5078	0.5238
18	1700	0.9547	1.0309	0.9499	1.0083	0.5171	0.4935	0.4660	0.5183
19	1800	0.9751	1.0024	0.9745	0.9916	0.5333	0.4476	0.5264	0.5165
20	1900	0.6666	0.8583	0.9755	0.9735	0.5323	0.5029	0.4992	0.4999
21	2000	0.9667	0.9917	0.8501	0.8512	0.5311	0.4607	0.4169	0.4498
22	2100	0.9666	1.0250	1.0084	0.9751	0.4683	0.5018	0.5057	0.5252
23	2200	0.9584	1.0166	0.9750	0.9917	0.5347	0.5244	0.5114	0.4503
24	2300	0.9750	0.9916	0.9583	0.9382	0.5333	0.4087	0.5165	0.5328
25	2400	0.7917	0.8906	0.9663	0.9281	0.5333	0.5244	0.4903	0.5252
26	2500	0.9667	1.0012	0.7753	0.7906	0.5248	0.5173	0.4263	0.3086
27	2600	0.9584	1.0084	1.0082	0.8762	0.4002	0.5256	0.4721	0.5249
28	2700	0.9916	0.9835	1.0001	0.9255	0.5333	0.5075	0.5276	0.3833
29	2800	0.9834	1.0083	1.0083	0.9333	0.5247	0.2504	0.4923	0.0517
30	2900	0.7416	0.9333	1.0000	0.9332	0.5337	0.5082	0.4970	0.4501
31	3000	0.9250	1.0084	0.7584	0.7834	0.5324	0.5240	0.3521	0.5253
32	3100	1.0001	0.9744	0.9916	0.9905	0.4673	0.5344	0.5078	0.5325
33	3200	1.0000	0.7575	1.0083	0.9762	0.5252	0.5167	0.5172	0.0821
34	3300	1.0000	1.0083	0.9834	1.0083	0.5334	0.4160	0.5249	0.3057
35	3400	0.8333	0.9916	0.9996	1.0333	0.5333	0.5260	0.5243	0.4931
36	3500	1.0250	0.9501	0.8336	0.7417	0.5328	0.5247	0.4673	0.4479
37	3600	1.0250	1.0077	1.0083	1.0083	0.2669	0.5083	0.5164	0.4750
38	3700	0.9916	0.7755	1.0084	1.0167	0.5167	0.5164	0.5334	0.4795
39	3800	0.9917	0.9833	1.0000	1.0001	0.4998	0.4077	0.5254	0.5291
40	3900	0.6750	1.0084	1.0000	0.9749	0.5002	0.5177	0.5333	0.5147

41	4000	0.9751	1.0166	0.8333	0.7745	0.5158	0.5086	0.4412	0.4504
42	4100	0.9999	1.0251	0.9999	0.9590	0.4089	0.5167	0.5256	0.5263
43	4200	1.0083	0.9830	0.9917	0.9999	0.5082	0.5074	0.5249	0.5170
44	4300	1.0001	1.0169	0.9896	1.0001	0.5335	0.3838	0.5243	0.5250
45	4400	0.7750	0.8499	0.9854	1.0163	0.5317	0.5169	0.5248	0.5250
46	4500	1.0000	1.0335	0.8667	0.6585	0.5346	0.5238	0.2922	0.4249
47	4600	1.0166	1.0250	0.9915	0.9495	0.4496	0.5076	0.5165	0.5334
48	4700	1.0167	1.0333	1.0049	0.9588	0.5174	0.5265	0.5227	0.5251
49	4800	1.0083	1.0330	0.9867	1.0083	0.5252	0.3837	0.5104	0.5333
50	4900	0.8416	0.8753	1.0001	1.0168	0.5333	0.5242	0.5168	0.5323
51	5000	0.9749	1.0166	0.7493	0.7832	0.5333	0.5080	0.4249	0.4592
52	5100	0.9667	1.0331	0.9675	1.0168	0.4248	0.5095	0.4921	0.5249
53	5200	1.0001	1.0085	1.0000	0.9916	0.5333	0.5248	0.4991	0.5242
54	5300	0.9575	0.9576	0.9912	1.0249	0.5337	0.4000	0.5083	0.5260
55	5400	0.8091	0.8574	1.0005	1.0084	0.5325	0.5249	0.5091	0.5250
56	5500	1.0166	1.0268	0.6833	0.8384	0.5233	0.5249	0.4163	0.4500
57	5600	1.0083	1.0250	0.9834	0.9700	0.4437	0.5004	0.5173	0.5246
58	5700	0.9751	1.0083	0.9917	1.0004	0.5167	0.5248	0.5250	0.5253
59	5800	1.0073	1.0417	1.0084	0.9249	0.5166	0.4669	0.5166	0.5334
60	5900	0.8508	0.9082	1.0079	0.9668	0.5334	0.5083	0.5206	0.5249
61	6000	1.0250	1.0335	0.8587	0.5666	0.5311	0.5250	0.4452	0.4501
62	6100	1.0250	0.9666	1.0000	0.9916	0.4435	0.5249	0.5159	0.5332
63	6200	1.0083	1.0166	0.9832	0.9500	0.5332	0.5250	0.5250	0.5252
64	6300	1.0000	1.0373	1.0001	1.0250	0.5252	0.3740	0.5261	0.5231
65	6400	0.8250	0.7946	0.9917	0.9834	0.5332	0.5183	0.5228	0.5250
66	6500	1.0083	1.0256	0.7486	0.8249	0.5334	0.5250	0.4422	0.3428
67	6600	1.0001	1.0083	0.9656	1.0167	0.4410	0.4832	0.5244	0.5251
68	6700	0.9417	1.0167	0.9362	1.0167	0.5258	0.5075	0.5255	0.5076
69	6800	1.0083	1.0166	1.0000	1.0168	0.5251	0.3840	0.5183	0.5174
70	6900	0.7917	0.7916	1.0083	1.0163	0.5250	0.5083	0.5241	0.5168
71	7000	0.9500	0.9751	0.6500	0.7669	0.5332	0.5248	0.4174	0.4500
72	7100	0.9499	1.0342	0.9876	0.9999	0.4667	0.5165	0.5165	0.5166
73	7200	1.0083	1.0073	0.9708	1.0001	0.5251	0.4834	0.5158	0.4496
74	7300	1.0083	1.0249	0.9999	0.9667	0.5332	0.2883	0.5175	0.4921
75	7400	0.7917	0.8918	0.9417	0.9893	0.5334	0.4382	0.5335	0.5001
76	7500	1.0166	1.0084	0.8749	0.7010	0.5333	0.4901	0.4409	0.4397
77	7600	0.9833	1.0250	0.9745	0.9925	0.4165	0.4940	0.5000	0.5022
78	7700	0.9917	1.0249	0.9755	0.9750	0.5336	0.5250	0.5163	0.5239
79	7800	1.0000	0.9918	0.9833	0.9833	0.5248	0.4485	0.5095	0.5259
80	7900	0.8830	0.8417	1.0068	1.0084	0.5333	0.5099	0.4834	0.5319
81	8000	1.0003	1.0083	0.8597	0.5750	0.5247	0.5083	0.4162	0.4346
82	8100	1.0166	1.0166	0.9999	0.9750	0.4671	0.5072	0.5243	0.5334

83	8200	1.0250	1.0083	0.9712	1.0250	0.5251	0.5156	0.5179	0.5249
84	8300	1.0237	1.0333	1.0039	1.0000	0.5332	0.4182	0.5247	0.5329
85	8400	0.5007	0.8251	0.9915	0.9750	0.5334	0.5248	0.5244	0.5256
86	8500	0.9167	1.0332	0.8334	0.7584	0.5302	0.5332	0.4007	0.3248
87	8600	0.9916	1.0166	1.0083	0.9915	0.4346	0.5256	0.5334	0.5162
88	8700	0.9748	1.0251	1.0001	1.0251	0.5180	0.5334	0.5167	0.5090
89	8800	0.9809	1.0330	0.9833	0.9917	0.5166	0.3996	0.5167	0.5000
90	8900	1.0001	0.4751	1.0083	1.0249	0.5170	0.4673	0.5165	0.5166
91	9000	0.9575	0.9987	0.4667	0.5333	0.5409	0.4748	0.4251	0.4084
92	9100	0.8091	1.0096	0.9668	0.9500	0.4173	0.4835	0.5081	0.5331
93	9200	1.0166	1.0249	0.9583	1.0124	0.5333	0.4916	0.5167	0.4418
94	9300	0.9748	0.8417	0.9500	1.0019	0.5165	0.2060	0.4996	0.4834
95	9400	0.9751	1.0153	0.8113	0.8686	0.5332	0.4917	0.5255	0.5156
96	9500	1.0073	0.9929	0.8473	0.9833	0.5331	0.4826	0.4234	0.1165
97	9600	0.8508	1.0167	0.9833	0.9917	0.4166	0.5083	0.5092	0.3580
98	9700	0.9575	1.0083	0.8939	1.0249	0.5257	0.5086	0.5177	0.4771
99	9800	1.0250	0.7417	0.9636	1.0084	0.5249	0.4082	0.5078	0.4495
100	9900	1.0083	1.0000	0.7592	0.9500	0.5244	0.5260	0.5323	0.4916
101	10000	1.0000	1.0084	0.9836	0.9084	0.5338	0.5249	0.4095	0.4498

Lampiran 4.17 Data *total loss*

No.	*Epoch*	*ResNet-50*				*Inception V2*			
1	0	2.4208	2.8405	3.6406	3.3949	2.5568	2.5293	2.4208	4.325
2	100	1.9617	2.6850	3.6248	3.9231	2.4487	2.1883	1.9617	3.1441
3	200	1.2488	1.8657	3.0934	1.7532	0.9999	3.5350	1.2488	1.4282
4	300	0.8060	0.7098	1.1370	0.4720	0.9381	0.9821	0.8060	0.5345
5	400	0.8793	0.9479	0.6914	0.4580	0.2670	0.7389	0.8793	0.8041
6	500	0.2997	0.8588	0.4819	0.8732	0.4095	0.7477	0.2997	0.7498
7	600	0.5481	0.4389	0.4448	0.7653	0.8643	0.4146	0.5481	0.7415
8	700	0.7327	0.5260	0.7157	0.7262	2.2162	0.7470	0.7327	0.3397
9	800	0.5109	0.3231	0.5634	0.5259	0.5640	0.8075	0.5109	1.0662
10	900	0.3339	0.7551	0.8260	0.7668	0.3660	0.6590	0.3339	0.3800
11	1000	0.4333	0.5438	0.4772	0.5691	1.8962	0.2128	0.4333	1.1282
12	1100	1.3227	0.6519	0.2916	1.1505	0.2973	0.3964	1.3227	0.8723
13	1200	0.3063	0.7090	0.6129	0.3696	0.4005	0.2188	0.3063	0.7138
14	1300	0.3417	0.4063	1.0688	0.5288	0.3004	0.5380	0.3417	0.7263
15	1400	0.9839	0.6445	0.3935	0.3842	0.1765	0.2051	0.9839	0.1897
16	1500	0.5526	0.2942	0.4150	0.7122	0.5445	0.6707	0.5526	0.5502
17	1600	0.2434	0.1407	0.3593	0.3705	0.3714	0.5716	0.2434	0.4116
18	1700	0.4145	0.4644	0.2416	0.1901	0.5068	0.7099	0.4145	0.6273
19	1800	0.4945	0.3926	0.3991	0.5777	0.3707	0.5255	0.4945	0.4285
20	1900	0.5726	0.5616	0.1297	0.3182	0.7366	0.6625	0.5726	0.5741

21	2000	0.2298	0.4851	0.6236	0.4304	0.4154	0.9263	0.2298	0.4828
22	2100	1.5022	0.3488	0.5196	0.7335	2.0666	0.6774	1.5022	0.5013
23	2200	0.3492	0.3615	0.3223	0.1173	0.2177	0.4993	0.3492	0.6560
24	2300	0.5335	0.4424	0.5521	0.2801	0.4967	0.3058	0.5335	0.2994
25	2400	0.2817	0.5269	0.4738	0.5826	0.5148	0.7523	0.2817	2.2919
26	2500	0.3519	0.5520	0.4601	0.3792	0.2133	0.3434	0.3519	0.5777
27	2600	0.2621	0.8814	0.3605	0.7367	0.3057	0.2259	0.2621	0.4191
28	2700	0.3561	0.8118	0.6163	0.4662	0.8721	0.6435	0.3561	0.4901
29	2800	0.2927	0.3670	0.6364	0.1879	0.3589	0.2016	0.2927	0.2814
30	2900	0.2238	0.3964	0.1210	0.2781	0.2319	0.5345	0.2238	0.5946
31	3000	0.1500	0.2653	0.9434	0.3231	0.4756	0.4035	0.1500	0.2965
32	3100	0.5545	0.1912	0.3936	0.2722	0.3870	0.2727	0.5545	0.4489
33	3200	0.5323	0.2801	0.4834	0.4297	0.2762	0.6180	0.5323	0.1293
34	3300	0.4308	0.3002	0.4770	0.5109	0.3674	0.7169	0.4308	0.3508
35	3400	0.3006	0.1104	0.5195	0.4320	0.1980	0.1124	0.3006	1.0595
36	3500	0.8787	0.0664	0.3097	0.2854	0.7501	0.4950	1.8787	0.3965
37	3600	0.3679	0.1236	0.2399	0.5508	0.4418	0.2759	0.3679	0.3267
38	3700	0.2207	0.3433	0.4002	0.5446	0.2368	0.1061	0.2207	0.4773
39	3800	0.5044	0.3403	0.3843	0.3116	0.3789	0.3496	0.5044	0.2593
40	3900	0.3800	0.2430	0.3975	0.2285	0.2751	0.0293	0.3800	0.4745
41	4000	0.3849	1.2517	0.7901	0.5794	0.3469	0.1442	0.3849	0.0218
42	4100	0.6106	0.3990	0.4124	1.6622	0.1105	0.3758	0.6106	0.5681
43	4200	0.4113	0.2797	0.2787	0.4526	0.2304	0.5959	0.4113	0.4459
44	4300	0.3335	0.1708	0.3183	0.1172	0.2917	0.3805	0.3335	0.4430
45	4400	0.3593	0.3928	0.2250	0.3521	0.2963	0.5320	0.3593	0.7533
46	4500	0.1194	0.2450	0.2613	0.1369	0.3496	0.3107	0.1194	0.1131
47	4600	0.0769	0.0879	0.5527	0.2361	0.3535	0.3743	0.0769	0.2898
48	4700	0.2135	0.4636	0.2037	0.2904	0.1885	0.4004	0.2135	0.2109
49	4800	0.1513	0.2352	0.0893	0.3079	0.0540	0.0596	0.1513	0.5397
50	4900	0.5467	0.2707	0.1750	0.1278	0.1224	0.3752	0.5467	0.3782
51	5000	0.2803	0.0894	0.2164	0.2473	1.3421	0.1111	0.2803	0.4477
52	5100	0.2121	0.1932	0.0365	0.2252	0.0808	0.3468	0.2121	0.5180
53	5200	0.2830	0.3020	0.4339	0.2050	0.1962	0.2860	0.2830	0.3760
54	5300	0.1746	0.3196	0.3986	0.2978	0.2453	0.6174	0.1746	0.2594
55	5400	0.3200	0.2012	0.1888	0.3261	0.1466	0.1215	0.3200	0.1725
56	5500	0.1682	0.2793	0.1775	0.0460	0.0435	0.2342	0.1682	0.7077
57	5600	0.5296	0.2010	0.2549	0.0589	0.2608	0.1126	0.5296	0.5279
58	5700	0.1533	0.3808	0.3517	0.2969	0.1641	0.2993	0.1533	0.1069
59	5800	0.2276	0.3085	0.1136	0.3330	0.2762	0.4413	0.2276	0.5299
60	5900	0.2874	0.2386	0.0651	0.3388	0.3233	0.1425	0.2874	0.2816
61	6000	0.2747	0.2852	0.0588	0.3172	0.1897	0.5549	1.2747	0.4063
62	6100	0.3441	0.1464	0.4250	0.2679	0.1768	0.3200	0.3441	0.3432

63	6200	0.1193	1.8978	0.1591	0.1482	0.2112	0.2908	0.1193	0.6578
64	6300	0.3326	0.2276	0.2581	0.3080	0.0744	0.1451	0.3326	1.1031
65	6400	0.1556	0.0507	0.2876	0.3017	0.2316	0.4556	0.1556	0.3502
66	6500	0.2145	0.8269	0.2074	0.1432	0.1384	0.1732	0.2145	0.3610
67	6600	0.1597	0.2961	0.1375	0.2407	0.1294	0.3304	0.1597	0.1131
68	6700	0.3591	0.0965	0.1317	0.0442	0.0467	0.1616	0.3591	0.3183
69	6800	0.0589	0.1283	1.4001	1.3031	0.1514	0.3690	0.0589	0.0649
70	6900	0.2474	0.9796	0.2871	0.3037	0.1212	0.3031	0.2474	0.1704
71	7000	0.1921	0.2364	0.2535	0.1571	0.1859	0.0855	0.1921	0.0952
72	7100	0.2415	0.1531	0.3746	0.0459	0.2978	0.0505	0.2415	0.2394
73	7200	0.3284	0.1942	0.3225	0.2485	0.2717	0.2729	0.3284	0.2545
74	7300	0.3370	0.2242	0.2539	0.0888	0.2386	0.1789	0.3370	0.0837
75	7400	0.3289	0.3136	1.1873	0.0433	0.2077	0.3161	0.3289	0.0563
76	7500	0.1960	0.7976	0.3234	0.0548	0.0537	0.1338	0.1960	0.2969
77	7600	0.2907	0.1594	0.1586	0.1044	0.0492	0.2298	0.2907	0.3711
78	7700	0.3074	0.2132	0.1474	0.1900	0.0799	1.3799	0.3074	0.3703
79	7800	0.4014	0.0478	0.3165	0.1238	0.0322	0.1138	0.4014	0.2933
80	7900	0.0479	0.3291	0.1239	0.0656	0.0632	0.2044	0.0479	0.3171
81	8000	0.2405	0.1677	0.1852	0.1747	0.0742	0.2159	0.2405	0.1615
82	8100	0.3250	0.1936	0.1348	0.3923	1.3226	0.1625	0.3250	0.2209
83	8200	0.3995	0.1867	0.2725	0.2572	0.0336	0.0827	0.3995	0.3348
84	8300	0.2800	0.0551	0.2265	0.1828	0.2071	1.1187	0.2800	0.4117
85	8400	0.2082	0.1775	0.1573	0.3135	0.1435	0.1647	0.2082	0.2238
86	8500	0.1517	0.1279	0.2154	0.3004	0.2794	0.2725	0.1517	0.2495
87	8600	0.0582	0.2498	0.2459	0.3160	0.2077	0.2265	0.0582	0.4785
88	8700	0.2379	0.2515	0.2203	0.2676	0.1229	0.1573	0.2379	0.1568
89	8800	0.0302	0.8649	0.2250	0.0903	0.1245	0.2154	0.0302	0.1569
90	8900	0.0946	0.3125	0.2517	0.0951	0.2728	0.2459	0.0946	0.2087
91	9000	0.1105	0.1785	0.2091	0.2450	0.1781	0.2203	0.1105	0.3535
92	9100	0.1062	0.0762	0.9668	0.1130	0.0331	0.2250	0.1062	0.3554
93	9200	0.2988	1.0249	0.2177	0.2197	0.5333	0.2517	0.2988	0.1178
94	9300	0.1080	0.1775	0.1604	0.2589	0.1435	0.1080	0.2015	0.1452
95	9400	0.0741	0.1279	0.1922	0.8686	0.2794	0.0741	0.1569	0.1952
96	9500	0.1260	0.2498	0.1416	0.9833	0.2077	0.1260	0.2087	0.4826
97	9600	0.1718	0.2515	0.9184	0.9917	0.1229	0.1718	0.3535	0.5083
98	9700	0.1388	0.8649	0.1535	0.3037	0.1245	0.1388	0.3554	0.5086
99	9800	0.1895	0.3125	0.1618	0.1571	0.2728	0.1895	0.1178	0.4082
100	9900	0.1044	0.1785	0.1979	0.0459	0.1781	0.1044	0.2015	0.5260
101	10000	0.0785	0.0762	0.2091	0.2485	0.0331	0.0785	0.2430	0.5249