IMPLEMENTASI *DEEP LEARNING* MENGGUNAKAN ALGORITMA *CONVOLUTIONAL NEURAL NETWORK* (CNN) UNTUK *IMAGE CLASSIFICATION* VARIETAS TANAMAN TEMBAKAU

SKRIPSI

Oleh:

Jefryka Dwi Radana

NIM 181710201057

JURUSAN TEKNIK PERTANIAN

FAKULTAS TEKNOLOGI PERTANIAN

UNIVERSITAS JEMBER

2022

IMPLEMENTASI *DEEP LEARNING* MENGGUNAKAN ALGORITMA *CONVOLUTIONAL NEURAL NETWORK* (CNN) UNTUK *IMAGE CLASSIFICATION* VARIETAS TANAMAN TEMBAKAU

SKRIPSI

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

Oleh:

Jefryka Dwi Radana

NIM 181710201057

JURUSAN TEKNIK PERTANIAN

FAKULTAS TEKNOLOGI PERTANIAN

UNIVERSITAS JEMBER

2022

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Kedua orang tua, Bapak Heru Sugito dan Ibu Suratemi yang telah mendidik, membimbing, mendoakan, memberikan dukungan, dan pengorbanan yang tidak terhingga;
2. Guru-guru sejak taman kanak-kanak hingga perguruan tinggi;
3. Almamater Fakultas Teknologi Pertanian Universitas Jember;

MOTO

Hai orang-orang yang beriman, mintalah pertolongan kepada Allah SWT dengan sabar dan shalat. Sesungguhnya Allah SWT beserta orang-orang yang sabar. (terjemahan QS. Al Baqarah : 153)[1]

Atau

Keep your eyes on the stars and your feet on the ground.

(Theodore Roosevelt)

[1] Departemen Agama Republik Indonesia. 1998. Al Qur'an dan Terjemahannya. Semarang: PT Kumudasmoro Grafindo.

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Jefryka Dwi Radana

NIM : 181710201057

menyatakan dengan sesungguhnya bahwa karya tulis ilmiah yang berjudul "Implementasi *Deep Learning* Menggunakan Algoritma *Convolutional Neural Network* (CNN) Untuk *Image Classification* Varietas Tanaman Tembakau" 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, 10 September 2022
Yang menyatakan,

Jefryka Dwi Radana
NIM. 181710201057

SKRIPSI

IMPLEMENTASI *DEEP LEARNING* MENGGUNAKAN ALGORITMA *CONVOLUTIONAL NEURAL NETWORK* (CNN) UNTUK *IMAGE CLASSIFICATION* VARIETAS TANAMAN TEMBAKAU

Oleh

Jefryka Dwi Radana

NIM. 181710201057

Pembimbing

Dosen Pembimbing Utama : Bayu Taruna Widjaja Putra., S.TP., M.Eng., Ph.D.

PENGESAHAN

Skripsi berjudul "Implementasi *Deep Learning* Menggunakan Algoritma *Convolutional Neural Network* (CNN) Untuk *Image Classification* Varietas Tanaman Tembakau" karya Jefryka Dwi Radana telah diuji dan disahkan pada:

hari, tanggal : Senin, 10 September 2022

tempat : Fakultas Teknologi Pertanian, Universitas Jember

Menyetujui,

Dosen Pembimbing Skripsi

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

NIP. 198410082008121002

Tim Penguji

Ketua,	Anggota
Dr. Ir. Bambang Marhaenanto, M.Eng., IPM.	Prof., Dr. Indarto, S.TP., DEA., IPU.
NIP. 196312121990031002	NIP. 197001011995121001

Mengesahkan

Dekan,

Dr. Ir. Bambang Marhaenanto, M.Eng., IPM.

NIP. 196312121990031002

RINGKASAN

Implementasi *Deep Learning* Menggunakan Algoritma *Convolutional Neural Network* (CNN) Untuk *Image Classification* Varietas Tanaman Tembakau; Jefryka Dwi Radana;181710201057;2022; 68 halaman; Jurusan Teknik Pertanian Fakultas Teknologi Pertanian Universitas Jember.

Tanaman tembakau (*Nicotiana tobacum L.*) merupakan salah satu komoditas perkebunan potensial di Indonesia. Tembakau memberikan dampak yang signifikan terhadap pendapatan nasional dan daerah. Teknologi *Deep Learning* merupakan suatu cabang ilmu *machine learning* berbasis Jaringan Saraf Tiruan (JST) atau bisa dikatakan sebagai perkembangan dari Jaringan Syaraf Tiruan. Teknologi yang digunakan dalam penelitian ini adalah teknologi *deep learning* untuk *image classification* terutama *convolutional neural networks* (CNN)

Tujuan pada penelitian ini adalah mengetahui bagaimana arsitektur dan hasil modifikasi algoritma *Convolutional Neural Networks* (CNN) untuk *image classification* varietas tembakau *Na-Oogst* dan *Voor-Oogst.* Metode yang digunakan adalah modifikasi CNN dengan perbedaan nilai *epoch* pada setiap data *training.* Modifikasi model CNN pada penelitian ini menggunakan *input shape* berukuran 128x128. Tahap *future learning* terdiri dari *layers* konvolusi sebanyak tiga kali, ukuran filter 3x3, *layers pooling* sebanyak tiga kali, ukuran filter 2x2. Tahap *classification* terdiri dari *flatten, full connected* dan aktivasi *sigmoid*. Data *training* 700 dan data testing 300. parameter yang dihasilkan sebesar 2.153.153 neuron pada model *training*.

Hasil didapatkan *accuracy training* yang cukup tinggi yakni mencapai 100 % dan *accuracy* validation mencapai 95%. Jika dilihat dari gambar dapat disimpulkan bahwa semakin menuju nilai 75 *epoch* yang digunkan maka akurasi dari hasil testing semakin tinggi. Tetapi ketika ditambahkan *epoch* hingga 100 nilai *accuracy validation* akan mengalami penurunan. Ini dapat disebabkkan oleh jumlah *epoch* yang terlalu banyak bisa juga dipengaruhi oleh banyaknya dataset. Selain itu, tidak ada penelitian yang mampu mengklaim rentang *epoch* terbaik pada proses pembelajaran.

SUMMARY

Implementation of Deep Learning Using Convolutional Neural Network (CNN) Algorithm for Image Classification of Tobacco Plant Varieties; Jefryka Dwi Radana;181710201057; 2022; 68 pages; Department of Agricultural Engineering, Faculty of Agricultural Technology, University of Jember.

Tobacco (*Nicotiana tobacum L.*) is one of the potential plantation commodities in Indonesia. Tobacco has a significant impact on national and local income. Deep Learning Technology is a branch of machine learning based on Artificial Neural Networks (ANN) or it can be said as the development of Artificial Neural Networks. The technology used in this research is deep learning technology for image classification, especially convolutional neural networks (CNN).

The purpose of this research is to find out how the architecture and the modified Convolutional Neural Networks (CNN) algorithm for image classification of tobacco varieties Na-Oogst and Voor-Oogst. The method used is a modified CNN with different epoch values for each training data. The CNN model modification in this study uses an input shape measuring 128x128. The future learning stage consists of three convolution layers, 3x3 filter size, three pooling layers, 2x2 filter size. The classification stage consists of flatten, fully connected and sigmoid activation. The training data is 700 and the testing data is 300. The resulting parameter is 2,153,153 neurons in the training model.

The results obtained that the accuracy of training is quite high, reaching 100% and accuracy validation reaching 95%. If seen from the picture, it can be concluded that the closer to the 75 epoch value used, the higher the accuracy of the testing results. But when you add up to 100 epochs, the accuracy validation value will decrease. This can be caused by the number of epochs that are too many can also be influenced by the number of datasets. In addition, there is no research that is able to claim the best epoch range in the learning process.

PRAKATA

Puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Implementasi *Deep Learning* Menggunakan Algoritma *Convolutional Neural Network* (CNN) Untuk *Image Classification* Varietas Tanaman Tembakau". 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. Bayu Taruna Widjaja Putra., S.TP., M.Eng., Ph.D. selaku Dosen Pembimbing Skripsi yang telah meluangkan waktu, pikiran dan perhatian untuk membimbing penulis demi terselesaikannya skripsi ini;
2. Dr. Elida Novita, S.TP., M.T. selaku Dosen Pembimbing Akademik yang telah membimbing dan memberikan semangat selama penulis menjadi mahasiswa;
3. Dr. Ir. Bambang Marhaenanto, M.Eng., IPM. selaku Dosen Penguji Utama dan Prof., Dr. Indarto, S.TP., DEA., IPU. selaku Dosen Penguji Anggota yang yang telah membimbing penulis selama penyusunan naskah skripsi;
4. Dr. Eng. Idah Andriyani. S.TP., M.T., IPM. selaku Kepala Prodi Jurusan Teknik Pertanian;
5. Dr. Ir. Bambang Marhaenanto, M.Eng., IPM. selaku Dekan Fakultas Teknologi Pertanian;
6. Segenap dosen pengampu matakuliah yang telah memberikan ilmu dan pengalaman selama penulis menjadi mahasiswa;
7. Seluruh staf dan karyawan di lingkungan Fakultas Teknologi Pertanian Universitas Jember, terima kasih atas bantuan dalam mengurus administrasi dan lainnya;
8. Teman-teman TEP C 2018 yang telah menemani penulis menjalani suka duka masa perkuliahan;
9. Keluarga besar BPM FTP UNEJ yang telah memberi pengalaman dan pelajaran pelengkap yang tidak akan pernah penulis dapatkan di perkuliahan;

10. Tim penelitian, Linggar Dwi Susulo, Deviana Yuli Tri Puspa Dewi, Eustolia Dyah Palupi, M Faris Setiawan, Adam Dika Lazuardi, Iqbal Doni Pratama, Muhammad, dan Apriza Fathur Roziqin yang telah menjadi partner berjuang dalam penyusunan skripsi ini;
11. Semua teman-teman Kos Putra Syar'i Halmahera 3 No.8, khususnya (Aceh) yang selalu memberikan semangat dan motivasi dalam penyelesaian skripsi ini;
12. Semua teman-teman grup "KWN 38" yang selalu memberikan semangat dan motivasi dalam penyelesaian skripsi ini;
13. Semua pihak yang tidak dapat penulis sebutkan satu persatu yang telah memberikan dukungan dan membantu penulis selama di Fakultas Teknologi Pertanian Universitas Jember.

Penulis menerima segala kritik dan saran dari semua pihak atas kekurangan baik secara teknis penulisan maupun pengetahuan demi kesempurnaan skripsi ini. Akhirnya penulis berharap, semoga skripsi ini dapat bermanfaat.

Jember, 10 September 2022

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1. PENDAHULUAN

1.1.Latar Belakang

Tanaman tembakau (*Nicotiana tobacum* L.) merupakan salah satu komoditas perkebunan potensial di Indonesia. Tembakau memberikan dampak yang signifikan terhadap pendapatan nasional dan daerah. Peran tembakau membantu meningkatkan perekonomian Indonesia ditunjukkan dari besarnya cukai yang disumbangkan kepada negara dan tenaga kerja lokal yang terserap baik saat proses penanaman hingga pengolahan tembakau yang akan diekspor ataupun dijadikan rokok (Santoso, 2013). Tembakau dapat dijadikan sebagai tanaman perkebunan komersial yang memiliki nilai jual dan pangsa pasar dengan keuntungan tinggi. Salah satu Kabupaten yang diakui sebagai pusat produksi tembakau yaitu Kabupaten Jember satu daerah di Provinsi Jawa Timur.

Tembakau di Kabupaten Jember terdapat dua jenis varietas yang ditanam, yaitu tembakau *Na-Oogst* dan tembakau *Voor-Oogst*. Tembakau *Na-Oogst* merupakan tembakau yang digunakan sebagai lapisan pembungkus dalam rokok cerutu (amblad) dan lapisan pembungkus luar cerutu (dekblad). Pada varietas tembakau *Voor-Oogst* dapat digunakan sebagai isian rokok kretek atau filter (Qoriah dan Meliczek, 2006). Dengan perbedaan kegunaan dari kedua varietas tersebut maka diperlukan teknologi untuk mengklasifikasikan tanaman tembakau guna mempermudah proses produksi. Untuk mempermudah penerapan teknologi maka perlu dilakukan pengukuran pengambilan citra yang digunakan untuk *image classification* varietas tembakau *Na-Oogst* dan *Voor-Oogst*.

Berdasarkan karakterisik kedua varietas tembakau memiliki bentuk fisik yang berbeda. Pada tembakau *Na-Oogst* memiliki bentuk tipis dan lembut, sedangkan pada tembakau *Voor-Ougst* memiliki bentuk tebal dan kasar. Berdasarkan perbedaan karakteristik fisik tersebut dapat dijadikan parameter untuk melakukan klasfikasi sesuai algoritma antara varietas tembakau *Na-Oogst* dan *Voor-Oogst.* Pada proses pengklasifikasian varietas tembakau menggunakan teknologi *deep learning* dikarenakan sesuai kapabilitas dalam *image classification*.

Teknologi *Deep Learning* merupakan suatu cabang ilmu *machine learning* berbasis Jaringan Saraf Tiruan (JST) atau bisa dikatakan sebagai perkembangan dari Jaringan Syaraf Tiruan. Dalam *deep learning*, sebuah komputer belajar mengklasifikasi secara langsung dari gambar atau suara (Swedia *et al*, 2018). Dalam *deep learning* sendiri banyak terdapat algoritma yang banyak digunakan untuk *image classification.* Beberapa algoritma yang sering digunakan antara lain VGG, LeNet, ResNet, K-Nearest Neighbor. Dari beberapa algoritma *deep learning* yang saat ini memiliki hasil paling signifikan dalam pengenalan citra adalah *Convolutional Neural Network* (CNN). Hal tersebut dikarenakan CNN berusaha meniru sistem pengenalan citra pada *visual cortex* manusia sehingga memiliki kemampuan mengolah informasi citra (Nugroho *et al*., 2020). Dengan menggunakan CNN, dataset citra daun tembakau yang mempunyai 2 varietas dan masing-masing varietas memiliki 1000 citra daun dapat diidentifikasi lebih mudah. Kemudahan ini karena CNN dapat mengenali objek dalam citra pada berbagai macam posisi yang mungkin (*translation invariance*)

Algoritma *Convolutional Neural Network*s (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang termasuk dalam *neural network* bertipe *feed forward* (bukan berulang). *Convolutional Neural Network* adalah *neural network* yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra (Devikar, 2016).. CNN dirancang khusus untuk pengenalan dan klasifikasi gambar. CNN memiliki beberapa lapisan (*layer*) yang mengekstrak informasi dari gambar dan menentukan klasifikasi dari gambar berupa skor klasifikasi (Nugroho *et al*., 2020).

Pada penelitian ini algoritma *Convolutional Neural Network* (CNN) modifikasi digunakan untuk mengklasifikasikan varietas tanaman tembakau *Na-Oogst* dan *Voor-Oogst* berdasarkan karakteristik daun. Kemampuan *Convolutional Neural Network* (CNN) dimodifikasi secara otomatis untuk membedakan varietas tanaman tembakau.

1.2. Rumusan Masalah

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

1. Bagaimana arsitektur modifikasi algoritma *Convolutional Neural Networks* (CNN) untuk *image classification* varietas tembakau *Na-Oogst* dan *Voor-Oogst*?
2. Bagaimana hasil akurasi modifikasi algoritma *Convolutional Neural Networks* (CNN) untuk *image classification* varietas tembakau *Na-Oogst* dan *Voor-Oogst*?

1.2. Batasan Masalah

Batas masalah yang diambil dalam penelitian ini adalah sebagai berikut.

1. Data yang digunakan adalah gambar daun tembakau *Na-Oogst* dan *Voor-Oogst*. Tembakau *Na-Oogst* diambil di PT Nusantara X Kebun Karang Anyar Kecamatan Balung Kabupaten Jember. Sedangkan tanaman *Voor-Oogst* diambil di persawahan di Desa Biting Kecamatan Arjasa Kabupaten Jember.
2. Gambar daun tembakau *Na-Oogst* dan *Voor-Oogst* diambil menggunakan kamera smartphone.
3. Metode yang digunakan untuk *image classification* varietas tembakau adalah dengan menggunakan algoritma CNN.
4. Citra input algoritma CNN memiliki ukuran pixels 128x128.
5. Aplikasi yang dirancang hanya akan mengklasifikasi dua jenis tanaman tembakau yaitu *Na-Oogst* dan *Voor-Oogst*.

1.4. Tujuan

Tujuan yang ingin dicapai dari penelitian ini adalah sebagai berikut

1. Membuat arsitektur algoritma untuk *image classification* varietas tembakau dengan algoritma *Convolutional Neural Network* (CNN).
2. Menghitung akurasi algoritma pemodelan *Convolutional Neural Network* (CNN) dari rancangan algoritma yang telah dimodifikasi.

1.5. Manfaat

Berikut merupakan manfaat dilaksanakannya penelitian mengenai Implementasi *Deep Learning* Menggunakan Algoritma *Convolutional Neural Network* (CNN) Untuk *Image Classification* Varietas Tanaman Tembakau.

1. Bagi ilmu pengetahuan dan teknologi (IPTEK) :

   Dapat dijadikan sebagai bahan rujukan, literatur dan sarana dalam menambah wawasan dan pengetahuan.

2. Bagi pemerintah:

   Dapat menerapkan klasifikasi jenis tanaman tembakau dengan menggunakan metode *deep learning*.untuk membantu produsen yang bergerak dibidang tembakau.

3. Bagi masyarakat :

   Hasil penelitian ini diharapkan menjadi acuan dan rekomendasi bagi masyarakat yang bergerak dibidang produksi tembakau untuk pengklasifikasian jenis tembakau.

BAB 2. TINJAUAN PUSTAKA

2.1. Daun Tembakau

Menurut Budiman (2010), klasifikasi tanaman tembakau adalah sebagai berikut :

Kingdom : *Plantae*
Sub Kingdom : *Viridiplantae*
Infra Kingdom: *Streptophyta*
Super Divisi : *Embryophyta*
Divisi : *Tracheophyta*
Sub Divisi : *Spermatophytina*
Kelas : *Magnoliopsida*
Super Ordo : *Asteranae*
Ordo : *Solanales*
Famili : *Solanaceae*
Genus : *Nicotiana* L
Spesies : *Nicotiana Tobacum* L

Menurut Murhawi (2014), tanaman tembakau merupakan komoditi tanaman perkebunan yang sangat strategis dan mempunyai dampak sosial yang luas, komoditi ini dapat menciptakan lapangan kerja dan usaha serta menjadi sumber penghasilan bagi masyarakat maupun pemerintah. Tembakau memberikan sumbangan pendapatan negara dalam bentuk cukai. Oleh dalam meningkatkan produksi perlu adanya budidaya tembakau yang baik.

Terdapat 2 varietas tembakau yang mempunyai arti ekonomi cukup tinggi. Kedua species secara umum di Indonesia menurut musim tanam ada dua yaitu tembakau *Voor-Oogt* dan tembakau *Na-Oogst*. Tembakau *Voor-Oogst* ini biasanya dinamakan tembakau musim kemarau. Artinya tembakau ini ditanam dimusim penghujan dan dipanen pada waktu musim kemarau. Tembakau *Na- Oogst* adalah jenis tembakau yang ditanam dimusim kemarau, kemudian dipanen atau dipetik pada musim penghujan. Berikut merupakan gambar tembakau Na-Oogst dan Voor-Oogst.

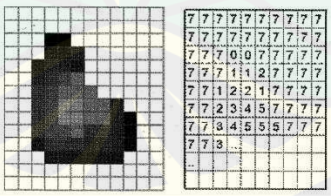
(a) (b)

Gambar 2.1 *(a) Varietas* Na-Oogst*, (b) Varietas* Voor-Oogst

2.2. Citra Digital

Citra digital adalah teknik mengolah citra yang bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin komputer yang dapat berupa foto maupun gambar bergerak (Effendi *et al*., 2017). Citra digital dapat didefinisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial dan amplitudo f di titik koordinat (x,y) merupakan intensitas atau tingkat keabuan citra pada titik tersebut. Nilai f(x,y) merupakan hasil kali dari jumlah cahaya yang mengenai objek (*illumination*) dan derajat kemampuan objek tersebut memantulkan cahaya (*reflection*) (Triwijoyo, 2019).

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$



Gambar 2.2 Representasi citra digital (Sumber: Effendi *et al*., 2017)

2.3. Deep Learning

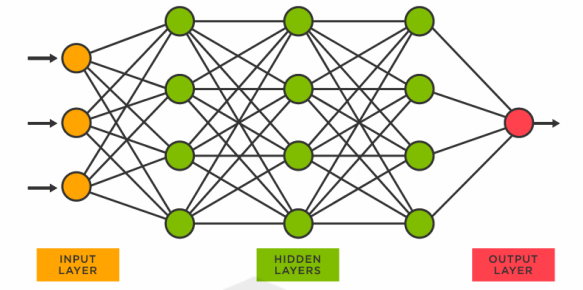
Deep Learning (DL) yang merupakan sebuah teknik berbasis jaringan saraf tiruan telah banyak digunakan dalam beberapa tahun terakhir sebagai salah satu metode implementasi *Machine Learning* (ML)(Diponegoro *et al*., 2021). Model

deep learning dapat mempelajari komputasinya sendiri dengan menggunakan otaknya sendiri. *Deep learning* dirancang untuk terus menganalisa data seperti pada otak manusia dalam mengambil keputusan (Peryanto *et al*., 2020). Pada *deep learning* lapisan pembelajaran dipresentasikan melalui model yang disebut *neural networks*, dalam harfiah lapisan ini ditumpuk di atas satu sama lain.

Deep learning memiliki suatu pandangan baru tentang representasi pembelajaran dari data yang menekankan pada lapisan pembelajaran (*layers*), semakin berturut-turut lapisan yang ada maka hasilnya akan lebih merepresentasikan objek yang ada (Chollet, 2018). Menurut LeCun *et al* (2015) *Deep learning* menemukan struktur rumit dalam kumpulan data besar dengan menggunakan algoritma *backpropagation* untuk menunjukkan bagaimana mesin harus mengubah parameter internal yang digunakan untuk menghitung representasi di setiap lapisan dari representasi di lapisan sebelumnya.

2.3.1. Neural Network

Istilah neural networks pertama kali digunakan oleh McCulloch & Pitts (1990) dalam percobaan untuk menemukan representasi matematis dari pemrosesan informasi dalam sistem biologis. Jaringan saraf (*neural networks*) merupakan jaringan dari *node* (simpul), yang meniru struktur neuron otak dari makhluk hidup. *Node* menghitung jumlah nilai bobot dari masukan dan memprosesnya pada lapisan tersembunyi, lalu mengeluarkan hasil dari fungsi pengaktifan dengan nilai bobot Neural networks telah dikembangkan dari arsitektur sederhana menjadi struktur yang semakin kompleks. Awalnya, pelopor *neural networks* memiliki arsitektur yang sangat sederhana dengan hanya lapisan input dan output, yang disebut jaringan saraf *single-layer*. Ketika lapisan tersembunyi (*hidden layer*) ditambahkan ke jaringan saraf *single-layer*, maka akan menghasilkan jaringan saraf *multi-layer*. Oleh karena itu, jaringan saraf multi-layer terdiri atas lapisan input, lapisan tersembunyi, dan lapisan output seperti pada Gambar 2.2 (Kim, 2017).

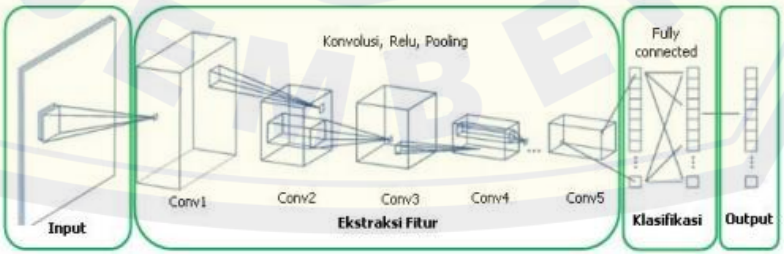


Gambar 2.3 Struktur dari neural networks (Sumber: Kim, 2017)

2.4. Convolution Neural Networks (CNN)

Convolution Neural Networks (CNN) merupakan kombinasi dari jaringan syaraf tiruan dan metode *deep learning* (Xiaofeng Han, 2015). CNN terdiri dari satu atau lebih lapisan konvoluonal, seringnya dengan suatu lapisan subsampling yang diiku oleh satu atau lebih lapisan yang terhubung penuh sebagai standar jaringan syaraf. Secara umum lapisan pada CNN terdiri dari (1) lapisan *convoluonal* (2) lapisan *sub sampling*, dan lapisan *output*. Dalam lapisan konvolusional, setiap bidang terhubung ke satu atau lebih peta fitur dari lapisan sebelumnya. Koneksi dihubungkan dengan topeng konvolusi, yang merupakan matriks 2-D dari entri yang dapat disesuaikan yang disebut bobot (*weights*) (You *et al*., 2017).

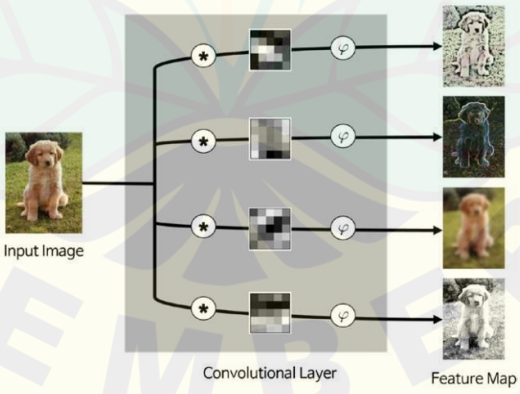
Lapisan pada CNN juga diperkuat dengan pernyataan Alom *et al* (2018) bahwa *Convolutional Neural Network* (CNN) terdiri atas tiga lapis (*layer*) yaitu lapis masukan (*input layer*), lapis keluaran (*output layer*), dan beberapa lapis tersembunyi (*hidden layers*). Lapis tersembunyi (*hidden layer*) umumnya berisi *convolutional layers, pooling layers, normalization layers*, ReLU *layer, fully connected layers*, dan serta *loss layer*.



Gambar 2.4 Arsitektur *Convolutional Neural Networks* (Sumber :Krizhevsky *et al*., 2012)

2.4.1. Convolutional Layer

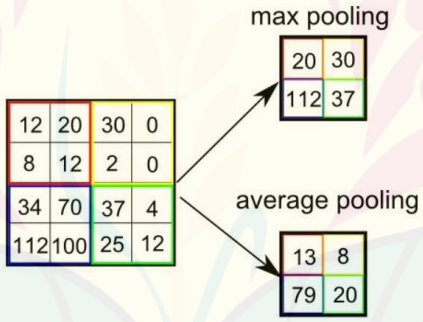
Lapisan konvulusional (*convolutional layer*) merupakan lapisan inti CNN, pada lapisan ini sebagian besar proses komputasi dilakukan. Tujuan utama konvolusi dalam kaitannya dengan ConvNet adalah untuk mengekstraksi fitur dari gambar yang dimasukkan (Karim, 2018). Lapisan konvolusi terdiri atas struktur dengan sejumlah filter dengan ukuran tetap yang memungkinkan fungsi kompleks diterapkan pada gambar yang telah dimasukkan. Proses ini dilakukan dengan cara menggeser filter di atas gambar. Setiap filter memiliki bobot dan nilai bias yang sama di seluruh gambar selama proses ini. Proses ini disebut mekanisme pembagian nilai berat dan mekanisme ini memberikan kemampuan untuk mewakili fitur yang sama pada seluruh gambar (Sarigül *et al*., 2019). Lapisan konvolusi menghasilkan gambar baru yang disebut peta fitur. Peta fitur menonjolkan fitur unik dari gambar asli. Lapisan konvolusi beroperasi dengan cara yang sangat berbeda dibandingkan dengan lapisan jaringan saraf lainnya. Pada Gambar 2.6 menunjukkan proses dari lapisan konvolusi, di mana tanda * menunjukkan operasi konvolusi, dan tanda φ adalah fungsi aktivasi. Ikon dengan skala abu-abu (*greyscale*) di antara operator ini menunjukkan filter konvolusi. Lapisan konvolusi menghasilkan jumlah peta fitur yang sama dengan filter konvolusi. Karena itu, misalnya, jika lapisan konvolusi berisi empat filter, itu akan menghasilkan empat peta fitur (Kim, 2017).



Gambar 2.5 Proses pada lapisan konvolusi (Sumber: Kim, 2017)

2.4.2. Pooling Layer

Pooling layer berfungsi menjaga ukuran data ketika *convolution* dilakukan, yaitu dengan cara melakukan *downsampling* (preduksi sampel), dengan adanya *pooling layer* data dapat direpresentasikan menjadi lebih kecil, mudah dikelola, dan mudah mengontrol *overvitting*. *Pooling layer* mengambil *layer convolutional* sebagai input. Proses pada *pooling layer* diterapkan ke *feature maps* yang telah melewati fungsi konvolusi dan aktivasi. Pada proses ini menghasilkan *feature map* yang lebih kecil, yang merupakan ringkasan dari *feature map* yang dimasukkan. Pooling dilakukan dengan cara menggeser filter pada gambar untuk menerapkan operasi yang dipilih, seperti yang ditunjukkan pada Gambar 2.7. Operasi pooling yang biasa digunakan adalah *max pooling, average pooling* dan L2-norm pooling (Sarigül *et al*., 2019). Keuntungan terbesar yang diberikan oleh operasi pooling adalah pengurangan ukuran gambar dan ekstraksi fitur visual secara independen pada gambar (Nielsen, 2015).



Gambar 2.6 Proses pada lapisan konvolusi dan lapisan *pooling* (Sumber: Shukla, 2018)

2.4.3. Normalization Layer

Lapisan normalisasi (*Normalization Layer*) dibuat untuk mengatasi adanya perbedaan rentang nilai yang signifikan pada citra yang dimasukkan. *Normalization layer* tidak banyak digunakan secara praktis karena dampaknya yang relatif kecil (Suyanto, 2018).

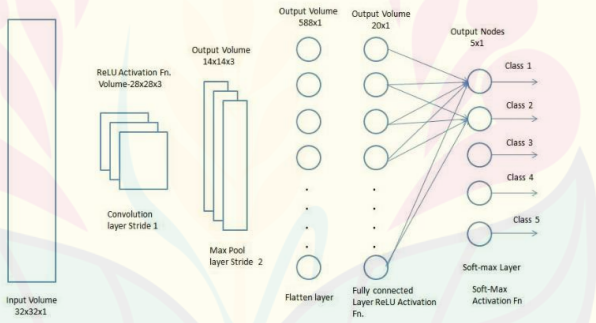
2.4.4. Fungsi Aktivasi ReLU (Rectrified Linear Unit) Layer

Rectified Linear Units (ReLU) layer berfungsi untuk meningkatkan sifat nonlinearitas fungsi keputusan dan jaringan secara keseluruhan tanpa mempengaruhi bidang-bidang reseptif pada *convolution layer* (Suyanto, 2018).

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 output dari neuron bisa dinyatakan sebagai 0 jika inputnya adalah negatif. Jika nilai input adalah positif, maka output dari neuron adalah nilai input aktivasi itu sendiri (Kim *et al*., 2016)

2.4.5. Fully Connected Layer

Fully connected layer merupakan lapisan setiap *neurons* memiliki koneksi penuh ke semua aktivasi dalam lapisan sebelumnya. Setelah proses konvolusi dan *pooling*, data ditransformasikan menjadi vektor satu dimensi. Vektor ini menjadi input dari jaringan yang sepenuhnya terhubung. Struktur yang terhubung sepenuhnya dapat berisi satu atau lebih lapisan tersembunyi. Setiap *neuron* mengalikan bobot koneksi dengan data dari lapisan sebelumnya dan menambahkan nilai bias. Nilai yang dihitung melewati fungsi aktivasi sebelum dikirim ke lapisan berikutnya (Sarigül *et al*., 2019).



Gambar 2.7 Proses *Fully Connected Layer* (Sumber: Sarigül *et al*., 2019)

2.4.6. 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), yang ditunjukan oleh.

$$Yijk = \frac{e^{xijk}}{\sum_{t=1}^{D} e^{xijt}} \quad \text{.....(2.1)}$$

Keterangan:

yijk = vektor yang berisi nilai antara 0 dan 1.

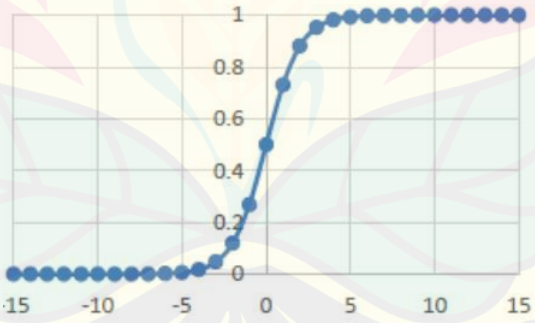
x = vektor yang berisi nilai yang didapatkan dari lapisan fully-connected terakhir.

2.4.7. Fungsi Aktivasi Sigmoid

Fungsi *sigmoid* mentransformasi range nilai dari input x menjadi antara 0 dan 1 dengan bentuk distribusi fungsi. Sehingga fungsi sigmoid memiliki bentuk sebagai berikut:

$$\sigma(x) = \frac{1}{(1 + e^{-x})} \quad \text{.....(2.2)}$$

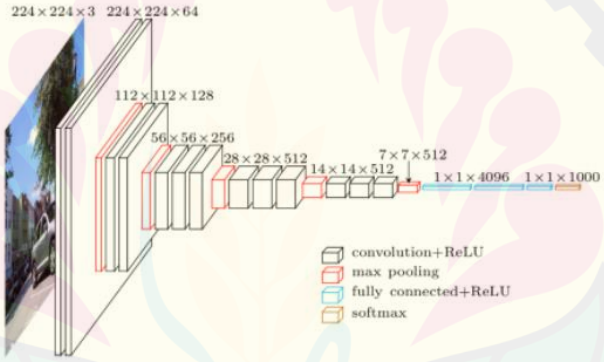
Fungsi *sigmoid* terjadinya proses backpropagation yang tidak ideal, selain itu bobot pada JST tidak terdistribusi rata antara nilai positif dan negatif serta nilai bobot akan banyak mendekati ekstrim 0 atau 1. Dikarenakan komputasi nilai propagasi menggunakan perkalian, maka nilai ekstrim tersebut akan menyebabkan efek saturating gradients dimana jika nilai bobot cukup kecil, maka lama kelamaan nilai bobot akan mendekati salah satu ekstrim sehingga memiliki gradien yang mendekati nol. Jika hal tersebut terjadi, maka neuron tersebut tidak akan dapat mengalami update yang signifikan dan akan nonaktif (Suartika *et al.,* 2016).



Gambar 2.8 Fungsi Aktivasi *Sigmoid* (Sumber: Suartika *et al.,* 2016)

2.5. Algoritma VGG

Model VGG merupakan model arsitektur yang diperkenalkan pada tahun 2014 olehdua orang peneliti yaitu Simonyan dan Zisserman dengan judul paper yaitu *Very Deep Convolutional Networks for Large Scale Image Recognition*. model arsitektur VGG memiliki ciri khas yaitu menggunakan 3x3 convolutional stack layer. Ciri khas ini yang menjadi alasan peneliti menggunakan model VGG 16 pada penelitian ini. Terdapat 2 jenis model VGG yang terkenal yaitu model VGG 16 dan model VGG 19. Setiap model VGG memiliki keunikan masing masing tetapi keunikan yang paling mencolok adalah jumlah layer yang digunakan. VGG 16 menggunakan 16 layer sedangkan VGG 19 menggunakan sebanyak 19 layer. Pada penelitian ini peneliti menggunakan model arsitektur VGG 19 yang dapat dilihat seperti pada gambar di bawah ini (You *et al*., 2017).



Gambar 2.9Arsitektur Algoritma VGG19 (Sumber: You *et al*., 2017)

Gambar ini menjelaskan bagaimana proses pada setiap *layer* di model VGG 19 ketika mengambil input dari dataset yang berupa citra digital. Perhatikan juga terdapat 19 layer yang mewakili konsep model VGG 19. Tumpukan layer sangat berpengaruh pada permodelan algoritma *Convolutional Neural Networks* tidak hanya pada model VGG 19 tetapi pada model arsitektur lainnya

2.6. Confusion Matrix

Confusion matrix adalah salah satu metode yang digunakan untuk mengevaluasi metodemetode klasifikasi. Tabel 2.1 merupakan gambaran sedehana untuk mempermudah pemahaman tentang istilah *confusion matrix* dalam keluaran klasifikasi.

Tabel 2.1 Model *Confusion Matrix* (Sumber: Sokolova & Lapalme, 2009)

		Kelas Prediksi (*Observation*)	
		Positif	Negatif
Kelas Aktual (*Expextation*)	Positif	TP	FN
	Negatif	FP	TN

Nilai *True Negative* (TN) adalah data yang di klasifikasi dengan tepat sebagai keluaran negatif atau salah. *True Positive* (TP) adalah data yang diklasifikasi dengan tepat sebagai keluaran positif atau benar. *False Positive* (FP) adalah data yang diklasifikasi dengan kurang tepat apabila keluaran berupa positif atau benar. *False Negative* (FN) adalah data yang diklasifikasi dengan kurang tepat (Rohim *et al.,* 2019).

2.6.1. Accuracy

Akurasi digunakan sebagai parameter sebagaimana akurat suatu model melakukan klasifikasi. Sementara untuk menghitung tingkat akurasi prediksi kejadian dapat digunakan persamaan sebagai berikut.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad \text{.....(2.3)}$$

Akurasi digunakan menghitung seberapa akurat suatu model untuk klasifikasi. Hasnain et al., (2020)

2.6.2. Precision

Presisi menggambarkan seberapa tepat suatu model memprediksi kejadian positif dalam serangkaian kegiatan prediksi. Perhitungan nilai presisi sebagai berikut.

$$Precision = \frac{TP}{TP + FP} \quad \text{.....(2.4)}$$

2.6.3. Recall

Sensitifitas (*recall*) digunakan untuk melihat detail kinerja suatu sistem atau suatu kelas. Perhitungan nilai *recall* sebagai berikut.

$$Recall = \frac{TP}{TP + FN} \quad \text{.....(2.5)}$$

2.6.4. F1 Score

F1 *Score* merupakan perhitungan rata-rata nilai Fscore yang merupakan nilai kombinasi dari perhitungan recall dan presisi. Berikut merupakan perhitungan F1 *Score.*

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad \text{.....(2.6)}$$

2.7. Anaconda

Anaconda merupakan salah satu aplikasi yang berfungsi sebagai distribusi bahasa pemrograman Python dan R yang memiliki sifat *open source*. Python banyak dimanfaatkan untuk berbagai perhitungan ilmiah, yang di dalamnya berupa *machine learning*, pengolahan data dengan ukuran besar, analisis prediksi, dan lain sebagainya. Anaconda memiliki tujuan untuk dapat menyederhanakan berbagai proses manajemen *package* ataupun *deployment*. Anaconda memiliki jumlah distribusi lebih dari 1500 package yang popular dan dapat diakses oleh berbagai platform sistem operasi seperti halnya *Windows*, *Linux*, dan *Mac*OS (Efanntyo, 2021).

Gambar 2.10 Logo Anaconda

2.8. Python

Python merupakan bahasa pemrograman bersifat interpreter yang mendukung paradigma pemrograman prosedural, fungsional maupun *object oriented programing* (OOP) dan berjalan dihampir semua platform sistem operasi. Python merupakan bahasa untuk pemrograman *scripting* dan *rapid application development* karena telah disediakan penggunaan modul-modul yang siap pakai dan struktur data tingkat tinggi yang efisien (Akbar *et al*, 2016).

Python merupakan bahasa pemrograman tingkat tinggi yang bertipe *interpreted language* yang banyak digunakan oleh non-programmer dan ilmuwan. Secara desain, kode Python dapat bekerja pada sebagian besar platform modern (Bingol & Krishnamurthy, 2019).

Gambar 2.11 Logo python

2.9. Jupyter Notebook

Jupyter Notebook dirancang untuk mendukung alur kerja komputasi ilmiah, mulai dari eksplorasi secara interaktif hingga penerbitan catatan. Kode dalam Jupyter Notebook disusun menjadi *cells*, potongan yang dapat dimodifikasi dan dijalankan secara individual. Jupyter bertujuan untuk membawa notebook ke khalayak yang lebih luas. Jupyter Notebook merupakan proyek *open source*, yang dapat bekerja dengan kode dalam berbagai bahasa pemrograman. Backend Bahasa yang berbeda, yang disebut kernel, berkomunikasi dengan Jupyter menggunakan protokol umum. Lebih dari 50 backend tersebut telah tersedia, untuk bahasa mulai dari C ++ hingga Bash. Jupyter Notebook tumbuh dari proyek IPython, yang awalnya menyediakan antarmuka ini hanya untuk bahasa Python. Jupyter Notebook dapat diakses melalui browser web. File di simpan dalam format JSON, dengan ekstensi .ipynb (Kluyver et al., 2016).

2.10. TensorFlow

Tensorflow merupakan salah satu *framework deep learning* dan juga salah satu *library* untuk *data science* yang bersifat *free open source* yang dikembangkan oleh para peneliti dari tim *Google*. Tensorflow dapat digunakan dalam berbagai bidang. Dalam bidang *object detection* terdapat *framework* tensorflow *object*

detection API yang merupakan suatu alat yang dapat digunakan untuk mempermudah proses *constructing*, *training* dan *deployment* pada suatu model *object detection*. Framework tensorflow *object detection* API menyediakan *pretrained object detection* model bagi user, namun memungkinkan jika user ingin menggunakan pretrained object detection model yang lain, seperti Faster R-CNN, SSD, Retinanet, Resnet50 dan masih banyak lagi (Manajang *et al*., 2020).

Tensorflow memiliki fitur untuk melatih model menggunakan *Graphics Processing Unit* (GPU). *Library* lain yang dipakai *deep learning* yaitu *Keras*. *Keras* menggunakan beberapa fitur dari Tensorflow untuk membuat jaringan syaraf tiruan karena *Keras* sudah menyediakan beberapa model dasar CNN yang telah di optimasi untuk mempermudah *Deep Learning* (Arsal dkk., 2020).

BAB 3. METODE PENELITIAN

3.1. Waktu dan Tempat Penelitian

Pelaksanaan penelitian dimulai 25 Januari 2022 sampai 31 Juni 2022. Wilayah yang dijadikan sebagai lokasi pengambilan data tembakau *Na-Oogst* yaitu di PT Nusantara X Kebun Karang Anyar Kecamatan Balung Kabupaten Jember dengan koordinat -8.252107 LS 113.525561BT untuk tembakau varietas *Voor-Oogst* di Desa Biting Kecamatan Arjasa Kabupaten Jember dengan koordinat -8.1133456 LS 113.7459203 BT. Pengolahan 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 perangkat keras dan perangkat lunak.

a. Perangkat Keras
   1. Laptop digunakan untuk mengolah data.
   2. Smartphone digunakan untuk mengambil citra daun tembakau.

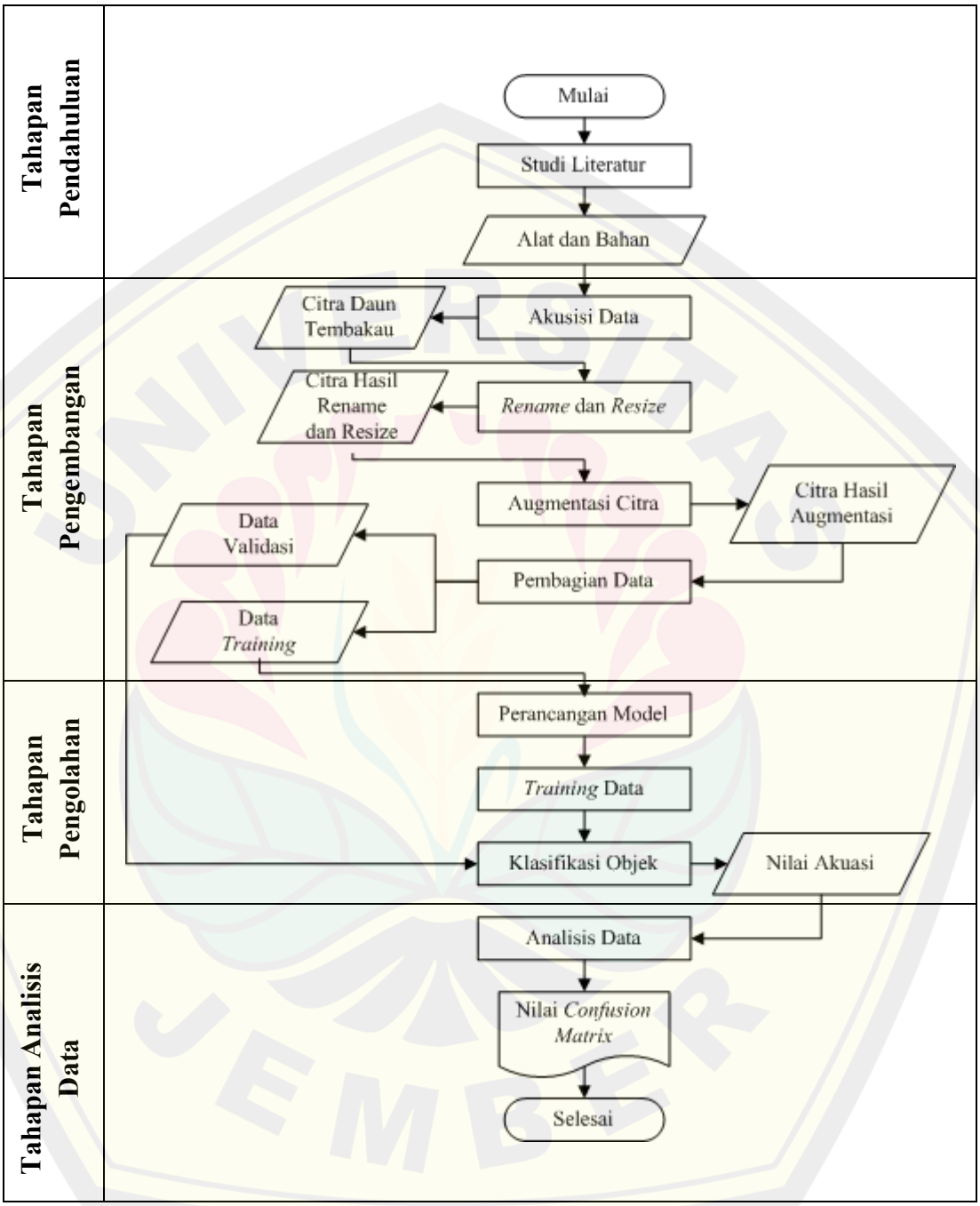
b. Perangkat Lunak
   1. *FastStone Photo Resizer* 4.4 digunakan untuk *resize* data gambar.
   2. *Bulk Rename Utility* 3.4.3 digumakan untuk *rename* data gambar.
   3. Anaconda versi 4.9.2 digunakan untuk eksekusi program yang telah dibuat.
   4. Jupyter Notebook 5.0 digunakan untuk menjalankan library pada klasifikasi objek.

3.2.2 Bahan Penelitian

Bahan yang digunakan dalam penelitian ini adalah citra daun tembakau *Na-Oogst* dan *Voor-Oogst* sebanyak 2000 gambar (masing-masing 1000 setiap jenisnya), berekstensi .JPEG dengan ukuran 3120 x 4160 pixel.

3.3. Prosesur Penelitian

Langkah atau tahapan yang dilakukan pada penelitian ini digambarkan melalui Gambar 3.1 berikut



Gambar 3 1 Diagram alir kegiatan penelitian

3.3.1. Studi Literatur

Studi literatur untuk menentukan objek penelitian dan sesuai dengan tema penelitian yang menjadi fokus peneliti. Dengan melakukan studi pendahuluan diharapkan dapat memperoleh informasi-informasi tentang permasalahan yang ada. Dalam proses ini peneliti melakukan observasi mengenai permasalahan yang berkaitan dengan *deep learning*. Merumuskan masalah dilakukan untuk menjadi alasan mengapa penelitian dilakukan dan menjadi pedoman yang dilakukan oleh peneliti dalam menyelesaikan karya tulis.

Kajian pustaka mengenai *deep learning*, tanaman tembakau, *convolutional neural networks*, *image classification* dan beberapa hal lainya yang terkait dengan penelitian ini. Pemodelan dilakukan untuk membentuk model dari *convolutional neural networks*, tujuan model pada penelitian ini adalah untuk melakukan klasifikasi terhadap varietas tanaman tembakau.

3.3.2. Akuisisi Data

Akusisi data dilakukan dengan menggunakan kamera *smartphone* dengan jumlah data sebanyak 2000 citra, 1000 citra tanaman tembakau *Na-Oogst* dan 1000 citra pada *Voor-Oogst*. Proses pengambilan citra diambil pada daun setiap varietas dengan memilih daun dengan kualitas yang baik. Metode pengambilan citra dengan melibatkan seluruh bagian daun tanaman tembakau. Jarak pengambilan gambar menyesuaikan dengan ukuran daun dengan tujuan dapat terinput seluruh bagian daun.

3.3.3. *Rename* dan *resize*

Proses pemberian label pada data berfungsi untuk memberikan nama terhadap data untuk dapat dikenali. Peneliti membuat dua folder utama yaitu folder train dan folder test / *validation*. Folder train berfungsi untuk menaruh data untuk diproses pada proses pembelajaran, sedangkan folder *test* / *validation* berfungsi untuk memvalidasi data pada proses *training*. Pada setiap subfolder diberi nama *Na-Oogst* dan *Voor-Oogst*. Untuk tahap *resize* dilakukan dengan menggunakan aplikasi FastStone Photo Resizer agar diperoleh ukuran gambar yang tidak terlalu besar. Gambar input berukuran 3120 x 3120 *pixels*, kemudian setelah dilakukan *resize* menghasilkan output gambar berukuran 224 x 224 pixels

3.3.4. Augmentasi

Proses agumentasi terhadap data citra daun untuk proses *training* dilakukan untuk mencegah terjadinyai *overfitting* (memiliki kinerja baik selama pelatihan, tetapi buruk pada data baru). Bahasa pemrograman yang digunakan ialah *Python* dan dijalankan melalui *packages* Jupyter Notebook. Proses augmentasi data citra menggunakan kode sebagai berikut:

```
train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=10,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        vertical_flip=False
)
```

1. Rescale = 1./255. Berfungsi berfungsi untuk mengubah ukuran data piksel RGB gambar (0-255) menjadi rentang angka (0-1) untuk memudahkan proses *training* data.
2. Rotation_range = 10. Berfungsi untuk mengubah rotasi gambar secara acak, angka 10 menunjukkan besaran derajat rotasi terhadap citra gambar.
3. width_shift_range = 0.2. Berfungsi untuk mengatur posisi gambar pada lebar gambar, angka 0.2 menunjukkan bahwa citra dapat secara acak berada maksimal 20% dari samping atau lebar awal citra gambar.
4. height_shift_range = 0.2. Berfungsi untuk mengatur posisi gambar pada tinggi gambar, angka 0.2 menunjukkan bahwa citra dapat secara acak berada maksimal 20% dari atas bawah atau tinggi awal citra gambar.

5. shear_range = 0.2. Merupakan sudut geser dalam arah berlawanan arah jarum jam dalam derajat.
6. zoom_range = 0.2. Berfungsi untuk membesarkan citra gambar secara acak, 0.2 menunjukkan intensitas pembesaran pada citra gambar.
7. horizontal_flip = True. Berfungsi untuk membalik secara horizontal citra gambar dengan acak, di atur dengan nilai benar.
8. vertical_flip = False. Berfungsi untuk membalik secara vertikal citra gambar dengan acak, di atur dengan nilai benar.

3.3.5. Pembagian Data

Dataset ini memiliki total data 2000 gambar. Dari jumlah tersebut, diambil data sebesar 70% dari jumlah data digunakan untuk proses pembelajaran (*training*) model dan 30% atau digunakan sebagai data untuk melakukan validasi (*validation*) model.

3.3.6. Perancangan Model Klasifikasi

Setelah dilakukan pembuatan data, langkah selanjutnya adalah melakukan pelatihan model CNN. Umumnya dalam CNN memiliki 2 tahapan, yaitu tahap *feature learning* dan *classification*. Input gambar pada model CNN menggunakan citra yang berukuran 128x128x3. Angka tiga yang dimaksud adalah sebuah citra yang memiliki 3 channel yaitu *Red, Green, dan Blue* (RGB) Citra masukan kemudian akan diproses terlebih dahulu melalui proses konvolusi dan proses *pooling* pada tahapan *feature learning*. Jumlah proses konvolusi pada rancangan ini memiliki tiga lapisan konvolusi. Setiap konvolusi memiliki jumlah filter dan ukuran kernel yang berbeda. Kemudian dilakukan proses *flatten* atau *proses* mengubah *feature map* hasil *pooling layer* kedalam bentuk *vector fully Connected layer* dan menggunakan aktifasi *sigmoid*. Bahasa pemrograman yang digunakan ialah *Python* dan dijalankan melalui *packages* Jupyter Notebook. Berikut adalah rancangan dari arsitektur CNN pada penelitian ini :

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation = 'relu', input_shape=(128,128,3), padding ='same', name='Conv_1'))
model.add(MaxPooling2D((2,2)))
```

```
model.add(Conv2D(64, (3, 3), activation = 'relu', padding ='same', name='Conv_2'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3, 3), activation = 'relu', padding ='same', name='Conv_3'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dropout(0.3))
model.add(Dense(128, kernel_initializer='normal', activation='relu', name='Dense_1'))
model.add(Dense(1, kernel_initializer='normal', activation='sigmoid', name='Dense_2'))
```

Berdasakan rancangan diatas dijelaskan terdapat dua tahap dalam arsitektur CNN, yaitu *Feature Learning* dan *classification*. *Feature learning* adalah teknik yang memungkinkan sebuah sistem berjalan secara otomatis untuk menentukan representasi dari sebuah *image* menjadi *features* yang berupa angka-angka yang merepresentasikan *image* tersebut. Tahap *Classification* adalah sebuah tahap dimana hasil dari *feature learning* akan digunakan untuk proses klasifikasi berdasarkan *subclass* yang sudah ditentukan. Pada konvolusi pertama menggunakan jumlah filter sebanyak 32 dan kernel dengan matriks 3x3. Kemudian dilakukan proses *pooling* menggunakan ukuran pooling 2x2 dengan pergeseran mask sebanyak dua langkah. Kemudian pada tahapan konvolusi kedua dengan menggunakan jumlah filter sebanyak 64 dan kernel dengan matriks 3x3 sebanyak 2 kali dengan ukuran matriks dan *pooling* yang sama. Kemudian di lanjutkan dengan *flatten* yaitu merubah output dari proses konvolusi yang berupa matriks menjadi sebuah vektor yang selanjutnya akan diteruskan pada proses klasifikasi dengan menggunakan MLP (*Multi Layer Perceptron*) dengan jumlah neuron pada lapisan tersembunyi yang telah ditentukan. Kelas dari citra kemudian diklasifikasikan berdasarkan nilai dari *neuron* pada lapisan tersembunyi dengan menggunakan fungsi aktivasi *Sigmoid*.

Hasil dari modifikasi model *Convolutional Neural Networks* (CNN) mepengaruhi data awal, semula berukuran 224x224 setelah proses konvolusi menjadi 128x128. Pada tahap selanjutnya akan dilakukan modifikasi parameter matrik yang akan dilakukan. Beberapa parameter metrik pada proses *learning* seperti *epoch, Accuracy, recall, precission, dan F1 Score*.

3.3.7. *Training* Data

Tahap training adalah tahap dimana model CNN diuji dengan data latih yang sudah disediakan. Jumlah dataset yang disediakan sebanyak 2000 data gambar, dengan jumlah gambar perkelas sebanyak 1000 gambar. Dataset dibagi kembali menjadi dua yaitu training dan validasi, yaitu sebanyak 700 training dan 300 validasi. Proses training menggunakan packages Tensorflow pada Jupyter Notebook. Tenosflow merupakan salah satu modul yang dibuat oleh Google untuk mempermudah dalam research mengenai neural network. *Hyperparameter* yang digunakan yaitu perbedaan *epoch* dengan nilai 25, 50, 75, 100.

3.3.8. Klasifikasi Objek

Algoritma *Convolutional Neural Network* membutuhkan proses *training* dan *validation*. Proses *training* ini bertujuan untuk melatih algoritma CNN dalam mengenali datasetnya dan membentuk sebuah model berdasarkan pelatihan tersebut. Proses *validation* bertujuan menguji sebuah model yang dibentuk pada saat proses *training*.

3.3.9. Analisis Data

Hasil dari prediksi ini termuat dalam sebuah matriks atau tabel kontigensi. Jika dijasikan kedalam table maka akan seperti berikut :

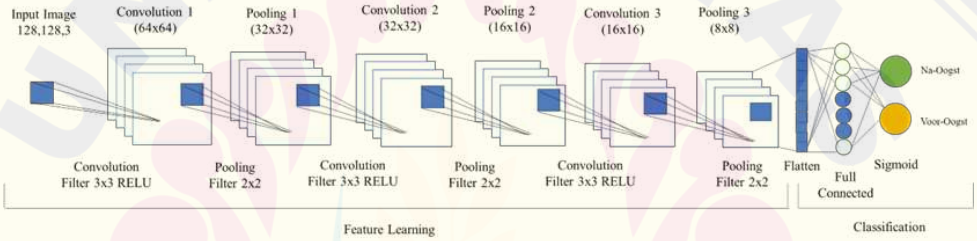
Tabel 3.1 Matriks prediksi

Matriks		Predict Class	
		Na-Oogst	Voor-Oogst
Actual Class	*Na-Oogst*	True Positive	
	Voor-Oogst		True Positive

BAB 4. HASIL DAN PEMBAHASAN

4.1. Arsitektur Jaringan

Arsitektur jaringan pada penelitian ini menggunakan struktur dari Alexnet. Alexnet dikembangkan oleh (Krizhevsky *et al*., 2012) yang merupakan basis dari arsitektur CNN modern yang cukup sukses untuk *visual recognition*. Untuk mengatasi kekurangan pada proses *training* data citra resolusi tinggi, perlu pengaturan pada beberapa lapisan dan mengekplorasi parameter untuk *training* data agar model CNN dapat menampilkan performa yang baik dan mencegah gradien menjadi tidak stabil, khususnya pada jaringan yang dalam. Arsitektur jaringan model CNN disajikan pada gambar berikut.



Gambar 4.1 Rancangan arsitektur CNN

Berdasarkan gambar 4.1 diatas dijelaskan terdapat dua tahap dalam arsitektur CNN, yaitu *Feature Learning* dan *classification*. *Feature learning* adalah teknik yang memungkinkan sebuah sistem berjalan secara otomatis untuk menentukan representasi dari sebuah *image* menjadi *features* yang berupa angka-angka yang merepresentasikan *image* tersebut. Tahap *Classification* adalah sebuah tahap dimana hasil dari *feature learning* akan digunakan untuk proses klasifikasi berdasarkan subclass yang sudah ditentukan.

Dataset yang digunakan pada proses klasifikasi terdiri dari 2 variabel yaitu *Na-Oogst* dan *Voor-Oogst.* Kedua dataset tersebut memiliki ukuran yang sama, yaitu 128x128x3. Arsitektur tersebut melibatkan proses konvolusi*, pooling, flatten atau fully connected,* dan aktifasi *sigmoid.* Pada konvolusi pertama menggunakan jumlah filter sebanyak 16 dan kernel dengan matriks 3x3. Kemudian dilakukan proses *pooling* menggunakan ukuran pooling 2x2 dengan pergeseran mask

sebanyak dua langkah. Kemudian pada tahapan konvolusi kedua dengan menggunakan jumlah filter sebanyak 32 dan kernel dengan matriks 2x2. proses *pooling* menggunakan ukuran pooling 2x2 dengan pergeseran mask sebanyak dua langkah. Proses konvolusi dilakukan dengan pengulangan 3 kali dengan jumlah filter sebesar 64 dan kernel matriks 3x3. Setelah proses konvolusi dilakukan pengulangan *pooling* dengan ukuran 2x2. Kemudian di lanjutkan dengan flatten yaitu merubah output dari proses konvolusi yang berupa matriks menjadi sebuah vector yang selanjutnya akan diteruskan pada proses klasifikasi dengan menggunakan MLP (Multi Layer Perceptron) dengan jumlah neuron pada lapisan tersembunyi yang telah ditentukan. Kelas dari citra kemudian diklasifikasikan berdasarkan nilai dari neuron pada lapisan tersembunyi dengan menggunakan fungsi aktivasi softmax. Berikut merupakan tabel struktur model yang terbentuk dari proses *training*.

Tabel 4.1 Hasil Keluaran Model CNN

Layer (type)	Output Shape	Parameter
Conv_1 (Conv2D)	(128, 128, 32)	320
max_pooling2d (MaxPooling2D)	(64, 64, 32)	0
Conv_2 (Conv2D)	(64, 64, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(32, 32, 64)	0
Conv_3 (Conv2D)	(32, 32, 64)	36928
max_pooling2d_2 (MaxPooling 2D)	(16, 16, 64)	0
flatten (Flatten)	16384	0
dropout (Dropout)	16384	0
Dense_1 (Dense)	128	2097280
Dense_2 (Dense)	1	129
Total	2,153,153	

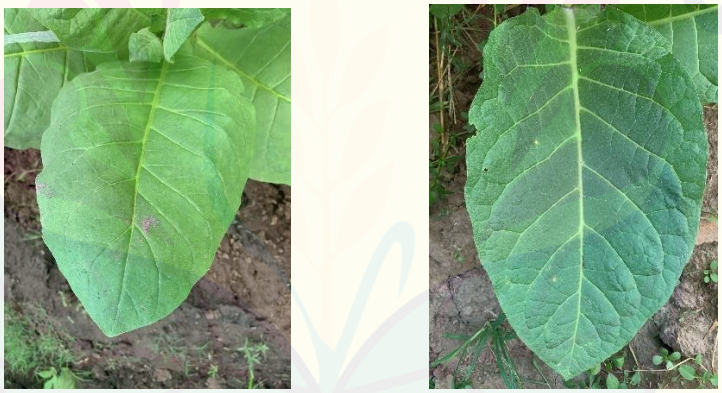
Berdasarkan Tabel 4.1 diatas, total parameter yang terbentuk dari model adalah sebanyak 2.153.153 neuron.

4.2. Tahap *Preproccesing* Data

Sebelum dilakukan proses *training* dan validasi pada citra diperlukan proses pra-pengolahan data. Hal itu dilakukan untuk memudahkan proses pengolahan data. Tahap *preprocessing* terhadap data citra dilakukan dengan pengambilan data citra, *resize & rename,* dan melakukan augmentasi pada citra.

4.2.1. Akusisi Data Citra

Data citra pertama adalah tembakau varietas *Na-Oogst* yang terdiri dari 1000 citra. Lokasi pengambilan data citra tembakau varietas *Na-Oogst* yaitu di PT Nusantara X Kebun Karang Anyar Kecamatan Balung Kabupaten Jember. Pada data citra kedua adalah tembakau varietas *Voor-Oogst* yang berjumlah 1000 citra. Lokasi pengambilan tembakau *Voor-Oogst* yaitu di Desa Biting Kecamatan Arjasa Kabupaten Jember. Pengambilan data citra dilakukan dengan menggunakan kamera *handphone*. Citra varietas tembakau *Na-Oogst* dan *Voor-Oogst* dapat dilihat pada gambar berikut.



(a) (b)

Gambar 4.2 (a) Varietas *Na-Oogst*, (b) Varietas *Voor-Oogst*

4.2.2. *Resize & Rename*

Resize adalah perlakuan mengubah ukuran data citra. Teknik ini digunakan untuk menyesuaikan citra supaya dapat dilakukan *training* atau testing (Jakaria *et al.,* 2021). Pada tahap *resize* dilakuakan reduksi citra dengan pengurangan *pixels* yang semula berukuran 3024 x 4032 menjadi 224 x224. Tujuan dilakukan proses *resize* juga untuk mengurangi beban perangkat pada proses *training* data. Karena ukuran gambar yang besar membutuhkan kapasitas prosesor laptop yang tinggi.

Setelah dilakukan proses *resize,* maka citra perlu dilakukan *rename.* Tujuan dari *rename* adalah mempermudah pengelompokan pada proses *training.* Riel *et al* (2012) juga menjelaskan bahwa, perubahan nama sangat penting untuk membedakan gambar satu dengan gambar yang lain. Berikut merupakan hasil setelah dilakukan *resize* dan *rename* pada citra.

(a) (b)

Gambar 4.3 Hasil *resize* dan *rename* (a) Na_Oogst_2, (b) Voor-Oogst_272

4.2.3. Augmentasi Data

Augmentasi merupakan proses mengolah data citra dengan memodifikasi data citra. Pada sistem ini tahap augmentasi yang dilakukan yaitu Rotasi, *Horizontal flip, Vertikal flip*, *width shift, zoom* dan *shear* pada *image* (Ibrahim *et al*., 2022). Pengaturan augmentasi data citra secara otomatis menggunakan kode sebagai berikut:

```
train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=10,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        vertical_flip=False
)
```

Augmentasi merupakan salah satu solusi dari masalah overfitting pada pembelajaran deep learning akibat data yang terbatas (Shorten & Khoshgoftaar, 2019).

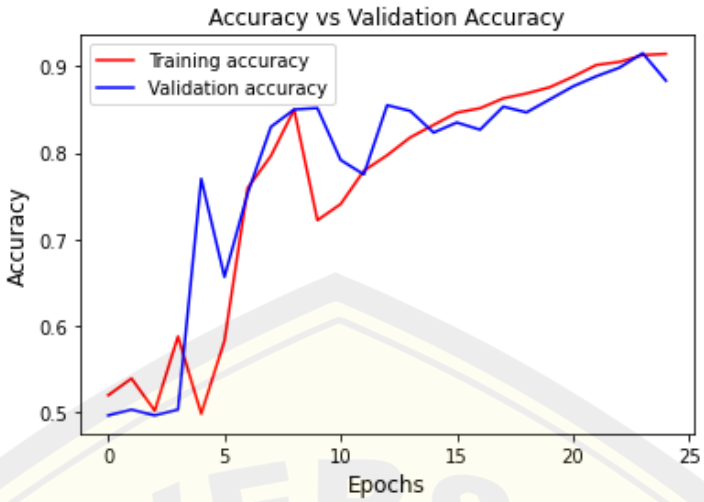
4.3. Hasil *Training* dan *Validation* Data

Pengujian pada sistem yang telah dirancang menggunakan metode CNN yang sudah dimodifikasi untuk mengetahui akurasi perbedaan varietas tembakau *Na-Oogst* dan *Voor-Oogst*. Sistem pengujian dibentuk dengan memanfaatkan perubahan *hyperparameter* pada banyaknya iterasi pelatihan (*epoch*). *Epoch* adalah ketika seluruh dataset sudah melalui proses *training* pada *Neural Network* sampai dikembalikan ke awal untuk sekali putaran, karena satu *epoch* terlalu besar untuk dimasukan (*feeding*) kedalam komputer, maka dari itu perlu membaginya kedalam satuan kecil (Digmi, 2018).

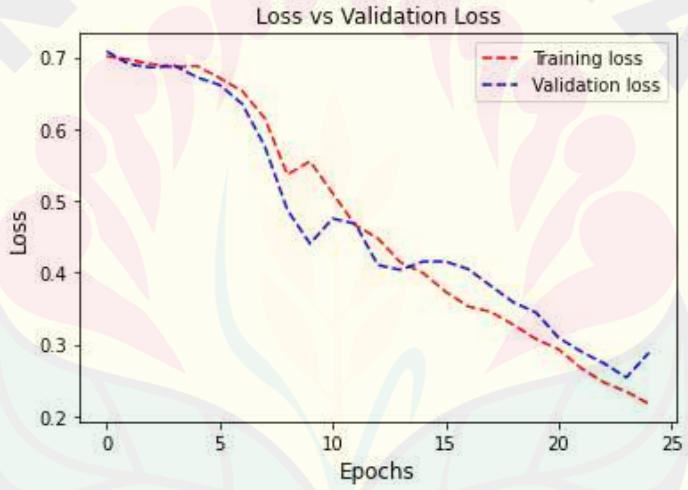
Nilai *epoch* yang akan dibandingkan pada percobaan ini yaitu *epoch* 25, 50, 75 dan 100. Berdasarkan hasil *training* diperoleh hasil perbandingan nilai *accuracy* dan nilai *loss*.

4.3.1. *Training* dan *Validation* data

Dalam bidang klasifikasi, ukuran akurasi dari suatu model klasifikasi sangat diperhatikan. Nilai akurasi dapat menggambarkan bagus tidaknya suatu model klasifikasi yang nantinya akan digunakan untuk menebak objek/citra baru (Yudianto, 2020). Berikut merupakan grafik *training accuracy*, *training loss, validation accuracy dan validation loss* pada modek CNN dengan perbedaan nilai *epoch* 25, 50, 75 dan 100.



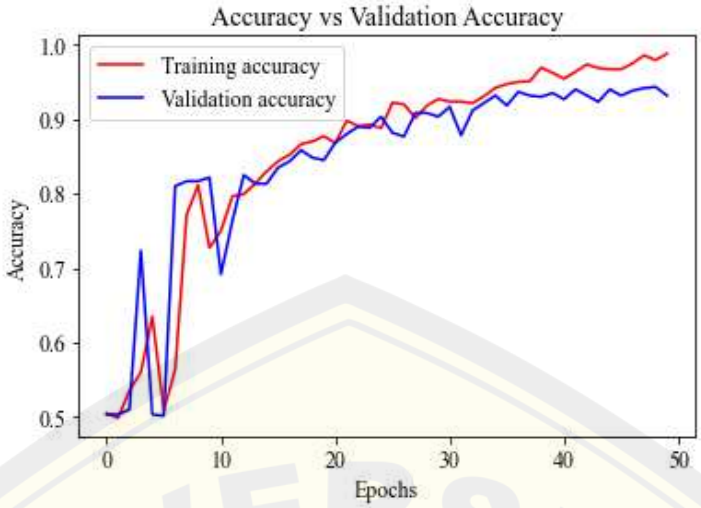
(a)



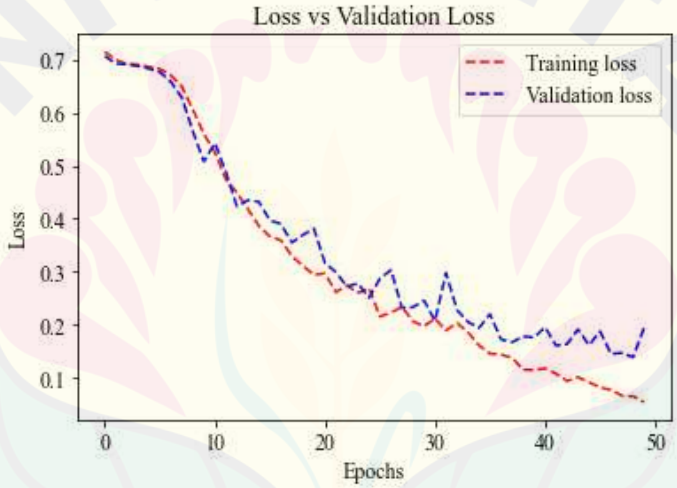
(b)

Gambar 4.4 *Training graph epoch*=25,(a) *training accuracy*, (b) *training loss*

Berdasarkan gambar 4.4 *accuracy* dari *training* model CNN mencapai 91 % dengan nilai *loss* sebesar 23%. Proses *training* disini menggunakan *batch size* sebesar 500 dengan input gambar sebesar 128 x 128 piksel. Waktu pelatihan yang dibutuhkan untuk 25 *epoch* dalam menjalankan *training* model ini yaitu 91 menit. Kemudian *accuracy* dari data validation mencapai 88 % dengan nilai loss sebesar 28%.



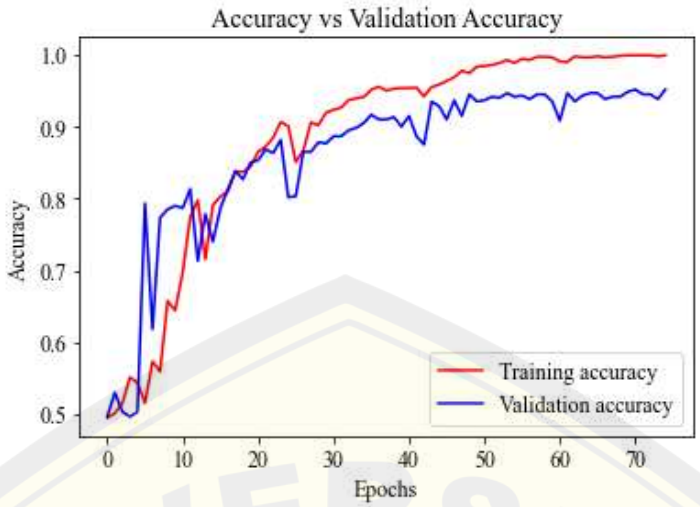
(a)



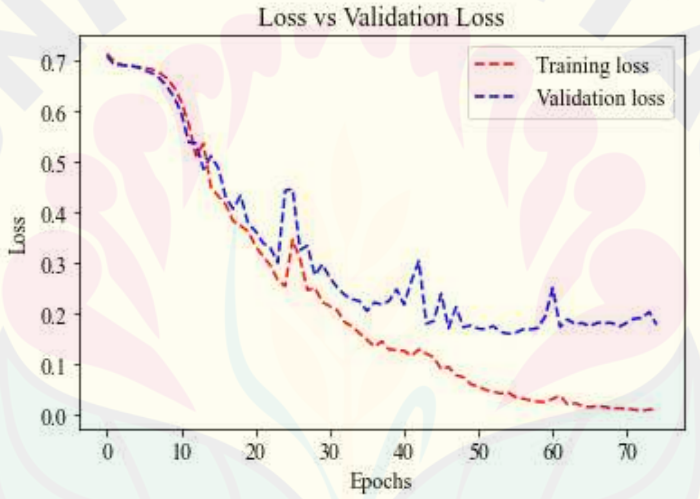
(b)

Gambar 4.5 *Training graph epoch*=50, (a) *training accuracy*, (b) *training loss*

Berdasarkan gambar 4.5 *accuracy* dari *training* model CNN mencapai 97 % dengan nilai *loss* sebesar 8%. Proses *training* disini menggunakan *batch size* sebesar 500 dengan input gambar sebesar 128 x 128 piksel. Waktu pelatihan yang dibutuhkan untuk 50 *epoch* dalam menjalankan *training* model ini yaitu 183 menit. Semakin Banyak *epoch* maka semakin lama juga waktu yang dibutuhkan untuk *training* model. Kemudian *accuracy* dari data validation mencapai 93 % dengan nilai loss sebesar 19%.



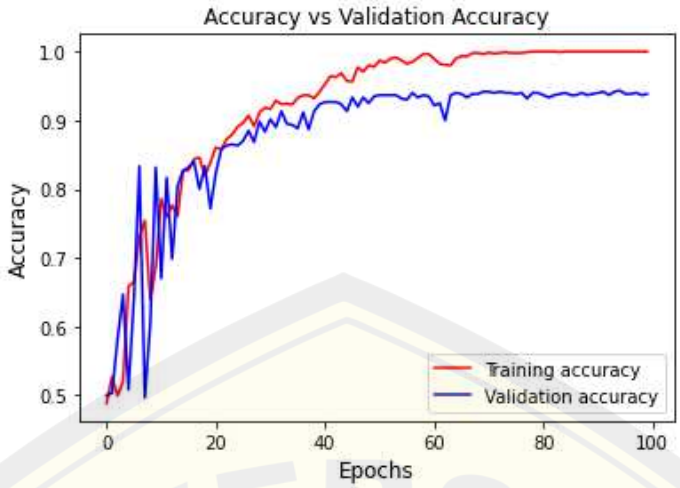
(a)



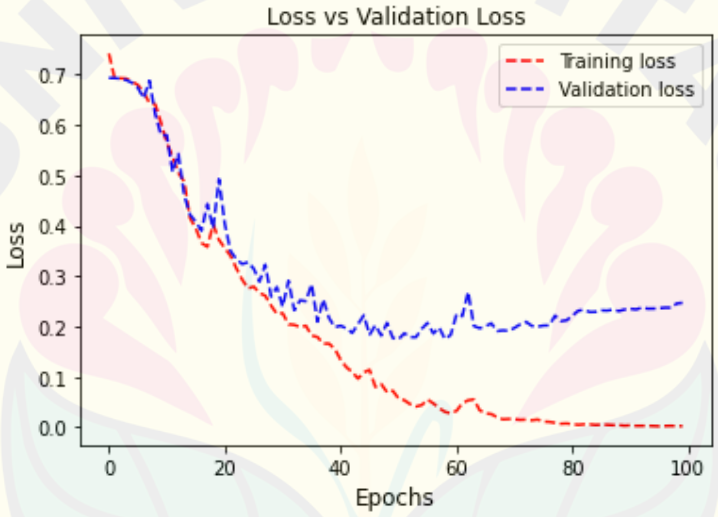
(b)

Gambar 4.6 *Training graph epoch*=75, (a) *training accuracy*, (b) *training loss*

Berdasarkan gambar 4.6 *accuracy* dari *training* model CNN mencapai 100 % dengan nilai *loss* sebesar 1%. Proses *training* disini menggunakan *batch size* sebesar 500 dengan input gambar sebesar 128 x 128 piksel. Waktu pelatihan yang dibutuhkan untuk 75 *epoch* dalam menjalankan *training* model ini yaitu 275 menit. Semakin Banyak *epoch* maka semakin lama juga waktu yang dibutuhkan untuk *training* model. Kemudian *accuracy* dari data validation mencapai 95 % dengan nilai loss sebesar 17%.



(a)



(b)

Gambar 4.7 *Training graph epoch*=100, (a) *training accuracy*, (b) *training loss*

Berdasarkan gambar 4.7 *accuracy* dari *training* model CNN mencapai 100 % dengan nilai *loss* sebesar 1%. Proses *training* disini menggunakan *batch size* sebesar 500 dengan input gambar sebesar 128 x 128 piksel. Waktu pelatihan yang dibutuhkan untuk 75 *epoch* dalam menjalankan *training* model ini yaitu 275 menit. Semakin Banyak *epoch* maka semakin lama juga waktu yang dibutuhkan untuk *training* model. Kemudian *accuracy* dari data *validation* mencapai 94 % dengan nilai loss sebesar 25%.

Berdasarkan Gambar 4.4, 4.5, 4.6 dan 4.7 diatas didapatkan *accuracy training* yang cukup tinggi yakni mencapai 100 % dan *accuracy validation* mencapai 95%. Jika dilihat dari gambar dapat disimpulkan bahwa semakin menuju nilai 75 *epoch* yang digunkan maka akurasi dari hasil testing semakin tinggi. Tetapi ketika ditambahkan *epoch* hingga 100 nilai *accuracy validation* akan mengalami penurunan. Ini dapat disebabkkan oleh jumlah *epoch* yang terlalu banyak bisa juga dipengaruhi oleh banyaknya dataset. Selain itu, tidak ada penelitian yang mampu mengklaim rentang *epoch* terbaik pada proses pembelajaran (Afaq & Rao, 2020).

4.4. Hasil Klasifikasi

Dalam mengukur kinerja-kinerja model dalam penelitian ini menggunakan *Accuracy*, *Precision*, *Recall* & f1-*Score* secara rata-rata. Sebelum dilakukan perhitungan klasifikasi maka diperlukan hasil *test* dalam *confusion matrix*. Berdasarkan perbedaan nilai *epoch* berikut merupakan hasil perhitungan *Confusion Matrix* dan *score* klasifikasi.

4.4.1. Confusion Matrix

Confusion Matrix Merupakan salah satu metode yang digunakan untuk mengevaluasi kinerja dari algoritma klasifikasi (Islam, 2017). Setelah model klasifikasi berhasil dibuat, langkah berikutnya adalah melakukan pengujian atau evaluasi dari model hasil proses pembelajaran. Proses evaluasi menggunakan metode *confusion matrix* dengan data testing yang telah disediakan. Parameter penilaian dari evaluasi ini adalah aspek *accuray*, *precission*, *recall* dan *F1-Score*. *Confusion matrix* ditampilkan pada tabel berikut.

Tabel 4.2 *Confusion Matrix*

Epoch	*Na-Oogst*		*Voor-Oogst*	
	Positif	Negatif	Positif	Negatif
25	240	62	290	8
50	266	36	293	5
75	289	13	282	16
100	282	20	281	17

Tabel 4.2 di atas menunjukkan beberapa *variable* yang berperan untuk proses evaluasi model klasifikasi. Variabel *True Negative* (TN) merupakan data yang tepat diklasifikasi oleh sistem sebagai nilai negatif atau salah, kemudian variabel *True Positive* (TP) merupakan data yang tepat diklasifikasi sebagai nilai positif atau benar. Lalu untuk variabel *False Positive* (FP) merupakan data yang diklasifikasikan tidak tepat apabila keluaran berupa positif atau benar kemudian untuk variabel *False Negatif* (FN) merupakan data yang diklasifikasikan dengan kurang tepat (Yudianto *et al.,* 2020).

4.4.2. Score Klasifikasi

Metode *confusion matrix* ini memiliki beberapa parameter penilaian terhadap kinerja dari model klasifikasi yang dihasilkan adalah sebagai berikut.

Tabel 4. 3 Perbandingan *score* klasifikasi CNN

Epoch	*Accuracy* (%)	*Precision* (%)	*Recall* (%)	F1-*Score* (%)
25	83	97	79	87
50	93	98	88	93
75	95	95	96	95
100	93	94	93	94

Tabel 4.3 menunjukkan hasil *classification report* pada pengujian model CNN menggunakan parameter perbedaan *epoch* 25, 50, 75 dan 100. Apabila melihat dari hasil tabel, *epoch* 75 memperoleh nilai akurasi tertinggi yaitu 95%, *recall* 96% dan *F1-Score* 95% dibandingkan dengan nilai pada *epoch* yang lain.

4.5. Perbandingan Performa Akurasi

Pada penelitian ini juga dilakukan klasifikasi pada model algoritma VGG19 untuk membandingkan model dari hasil yang telah dibuat. Model algoritma VGG19 merupakan salah satu model *confolusion neural networks* yang menggunakan 19 *layers* dalam pegolahan input data. Hasil *classification report* pada algoritma VGG19 sebagai berikut.

Tabel 4.4 *classification report* pada algoritma VGG19

Epoch	*Accuracy* (%)	*Precision* (%)	*Recall* (%)	F1-*Score* (%)
25	52	52	54	53
50	56	57	57	57
75	52	53	51	52
100	51	51	54	52

Tabel 4.4 menunjukkan hasil *classification report* pada pengujian algotima VGG19 menggunakan parameter perbedaan *epoch* 25, 50, 75 dan 100. Algoritma VGG19 pada penelitian ini menunjukan nilai yang lebih rendah dibandingkan model CNN modifikasi yang telah dibuat. Menurut Yudianto (2020) Banyak factor yang menyebabkan perbedaan nilai akurasi pada setiap algoritma. Perlakuan *augmentation*, pengubahan channel citra ke *grayscale* memiliki pengaruh yang signifikan terhadap nilai akurasi yang dihasilkan. Jumlah data dan pembagian data *training* dan test juga mempengaruhi *classification score.*

BAB 5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

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

1. Modifikasi model CNN pada penelitian ini menggunakan *input shape* berukuran 128x128. Terbagi dalam dua tahap yaitu *future learning* dan *classification*. Tahap *future learning* terdiri dari *layers* konvolusi sebanyak tiga kali, ukuran filter 3x3, *layers pooling* sebanyak tiga kali, ukuran filter 2x2. Tahap *classification* terdiri dari *flatten, full connected* dan aktivasi *sigmoid.* Data *training* 700 dan data *testing* 300. parameter yang dihasilkan arsitektur jaringan yang telah dibuat sebesar 2.153.153 neuron pada model *training*.
2. Modifikasi model CNN menghasilkan tingkat akurasi *training* dan *testing* tertinggi dengan nilai *epoch* 75 sebesar 100 % *training* dan 95 % *testing*. Untuk *classification report* hasil nilai *precision* tertinggi pada *epoch* 50 sebesar 98%, *recall* tertinggi pada *epoch* 75 sebesar 96% dan *F1-score* tertinggi pada *epoch* 75 sebesar 95%.

5.2. Saran

Saran yang diberikan pada penelitian ini sebagai berikut :

1. Penelitian selanjutnya diharapkan dapat menambah jumlah kelas klasifikasi varietas tembakau.
2. Penelitian ini dapat di kembangkan kedalam sebuah aplikasi yang digabungkan dengan *smartphone*.
3. Kelemahan dari penelitian ini salah satunya tidak dapat menentukan pemilihan parameter secara optimum. Penentuan parameter harus dilakukan metode *trial* and *error* untuk mendapatkan tingkat akurasi yang tinggi

DAFTAR PUSTAKA

Afaq, S., & Rao, S. 2020. Significance Of *Epoch*s On *Training* A Neural Network. International Journal of Scientific and Technology Research, 19(6), 485–488.

Akbar, M. A., Ilhamsyah., Ruslianto, I. 2016. Sistem Penjadwalan Laboratorium dan Monitoring Penggunaan Komputer Menggunakan Rfid Berbasis TCP/IP. Jurnal Coding, Sistem Komputer Untan Volume 04, No.2 (2016), hal. 23-34.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Van Esesn, B. C., *et al*. (2018). The history began from AlexNet: a comprehensive survey on deep learning approaches. arXiv preprint arXiv:1803.01164, 1–39.

Arsal, M., B. Wardijono, Agus, dan D. Anggraini. 2020. Face recognition untuk akses pegawai bank menggunakan deep learning dengan metode cnn. Jurnal Nasional Teknologi Dan Sistem Informasi. 6(1):55–63.

Bingol, O. R., & Krishnamurthy, A. (2019). NURBS-Python: An open-source object-oriented NURBS modeling framework in Python. SoftwareX, 9, 85–94.

Borji, A. (2018). Negative results in computer vision: A perspective. Image and Vision Computing, 69, 1–8.

Budiman, H. 2011. Budidaya Tanaman Tembakau. Pustaka Baru Press, Yogyakarta.

Chollet, F. (2018). Deep Learning with Phyton. In Manning.

Diponegoro, Sri Suning Kusumawardani, dan Indriana Hidayah. 2021. Tinjauan pustaka sistematis: implementasi metode deep learning pada prediksi kinerja murid. Jurnal Nasional Teknik Elektro Dan Teknologi Informasi. 10(2):131–138.

Efanntyo, 2021. Perancangan Aplikasi Sistem Pengenalan Wajah Dengan Metode Convolutional Neural Network (CNN) Untuk Pencatatan Kehadiran Karyawan. Jurnal Instrumentasi dan Teknologi Informatika (JITI) p-ISSN : 2746-7635 Vol. 3 No. 1

Effendi, M., Fitriyah., Effendi, U. 2017. Identifikasi Jenis Dan Mutu Teh Menggunakan Pengolahan Citra Digital Dengan Metode Jaringan Syaraf Tiruan. Jurnal Teknotan Vol. 11 No. 2.

Heaton, J. 2015. Artificial Intelligence for Humans: Deep learning and neural networks of Artificial Intelligence for Humans Series. Createspace Independent Publishing Platform.

Islam, K. T., Raj, R. G. and Al-Murad, A. 2017. Performance of SVM, CNN, and ANN with BoW, HOG, and Image Pixels in Face Recognition. 2nd International Conferenceon Electrical and Electronic Engineering. ICEEE 2017. IEEE

Jakaria, A., Mu'minah, S., Riana, D., Hadianti, S. 2021. Klasifikasi Varietas Buah Kiwi dengan Metode Convolutional Neural Networks Menggunakan Keras. Jurnal Media Informatika Budidarma Volume 5, Nomor 4, Oktober 2021, Page 1309-1315

Karim, M. R. 2018. Practical Convolutional Neural Networks : Implement advanced deep learning models using Python. Birmingham: Packt Publishing.

Kim, J., Sangjun, O., Kim, Y., & Lee, M. 2016. Convolutional Neural Network with Biologically Inspired Retinal Structure. Procedia Computer Science, 88, 145–154.

Kim, P. 2017. MATLAB deep learning : with machine learning, neural networks and artificial intelligence. New York, NY: Apress.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K. 2016. Jupyter Notebooks-a publishing format for reproducible computational workflows. ELPUB (pp. 87–90).

Krizhevsky, A., Sutskever, I., and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. Proceedings of the Twenty-Sixth Annual Conference on Neural Information Processing Systems. Lake Tahoe, NY, USA, 3–8 December 2012, 1097–1105.

LeCun, Y., Bengio, Y., & Hinton, G. 2015. Deep learning. Nature, 521, 436. Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.

Manajang, D. J. P., Sompie, S. R. U. A., Jacobus, A. 2020. Implementasi Framework Tensorflow Object Detection Dalam Mengklasifikasi Jenis Kendaraan Bermotor. Jurnal Teknik Informatika vol.15 no.3.

McCulloch, W., & Pitts, W. 1990. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biology, 52(1–2), 99–115.

Murhawi. 2014. Teknis Budidaya Tembakau (Nicotania Tabacum L.) Balai Besar Pembenihan dan Proteksi Tanaman Perkebunan Surabaya.

Nielsen, M. A. 2015. Neural networks and deep learning (Vol. 25). Determination press San Francisco, CA, USA.

Nugroho, P. A., I. Fenriana, dan R. Arijanto. 2020. Implementasi deep learning menggunakan convolutional neural network (cnn) pada ekspresi manusia. Jurnal Algor. 2(1):12–21.

Peryanto, A., A. Yudhana, dan R. Umar. 2020. Rancang bangun klasifikasi citra dengan teknologi deep learning berbasis metode convolutional neural network. Format : Jurnal Ilmiah Teknik Informatika. 8(2):138.

Qoriah, C. G., & Meliczek, H. 2006. Supply Response and Competitiveness of Na-Oogst Tobacco Production Analysis in Jember Regency-Indonesia. In Tropentag "Prosperity and Poverty in a Globalised World—Challenges for Agricultural Research" (p. 356). University of Bonn. Retrieved from

Riel, Van., Cees., Charles J Fombrun. 2007. Essentials of Corporate Communications: Implementing Practice for Effective Reputation Management. USA: Routledge - Taylor & Francis e-Library.

Santoso, K. 2013. Tembakau : dibutuhkan dan dimusuhi. Jember: Jember University Press.Retrieved from

Sarigül, M., Ozyildirim, B. M., & Avci, M. 2019. Differential convolutional neural network. Neural Networks, 116, 279–287.

Shorten, C., & Khoshgoftaar, T. M. 2019. A survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 1–48.

Shukla, N. 2018. Machine learning with TensorFlow. Shelter Island, NY: Manning Publications.

Suartika, E. P., Wijaya, A.Y., Soelaiman, R. 2016. Klasifikasi Citra Menggunakan Convolutional Neural Network (Cnn) pada Caltech 101. JURNAL TEKNIK ITS Vol. 5, No. 1

Suyanto. 2018. Machine Learning Tingkat Dasar dan Lanjut. Bandung: Informatika Bandung.

Swedia, E. R., Mutiara, A. B., & Subali, M. 2018. Deep learning long-short term memory (LSTM) for Indonesian speech digit recognition using LPC and MFCC Feature. In 2018 Third International Conference on Informatics and Computing (ICIC) (pp. 1-5). IEEE.

Triwijoyo, B. K. 2019. Model Fast Tansfer Learning pada Jaringan Syaraf Tiruan Konvolusional untuk Klasifikasi Gender Berdasarkan Citra Wajah. MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer, 18(2), 211-221.

Vedaldi, A., & Lenc, K. 2015. MatConvNet: Convolutional Neural Networks for MATLAB. In Proceedings of the 23rd ACM International Conference on Multimedia (hal. 689–692).

Xiaofeng Hang, Yan Li. 2015. The Application of Convolutional Neural Networks in Handwritten Numeral Recognition. International Journal of Database Theory and Application, Vol.9, No, 3.

You, W., Shen, C., Guo, X., Jiang, X., Shi, J., & Zhu, Z. 2017. A hybrid technique based on convolutional neural network and support vector regression for intelligent diagnosis of rotating machinery. Advances in Mechanical Engineering, 9(6), 1687814017704146. SAGE Publications Sage UK: London, England.

LAMPIRAN

Lampiran 1 Dokumentasi Penelitian

Gambar 1. Pengambilan Gambar

Gambar 2. Hasil Pengambilan gambar

```
Epoch 1/25
3/3 [==============================] - 21s 7s/step - loss: 0.7180 - accuracy: 0.4950 - val_loss: 0.6981 - val_accuracy: 0.4967
Epoch 2/25
3/3 [==============================] - 25s 8s/step - loss: 0.6935 - accuracy: 0.5014 - val_loss: 0.6902 - val_accuracy: 0.6033
Epoch 3/25
3/3 [==============================] - 23s 8s/step - loss: 0.6884 - accuracy: 0.5871 - val_loss: 0.6879 - val_accuracy: 0.5100
Epoch 4/25
3/3 [==============================] - 20s 7s/step - loss: 0.6856 - accuracy: 0.6021 - val_loss: 0.6823 - val_accuracy: 0.5000
Epoch 5/25
3/3 [==============================] - 20s 7s/step - loss: 0.6759 - accuracy: 0.6257 - val_loss: 0.6618 - val_accuracy: 0.8333
Epoch 6/25
3/3 [==============================] - 21s 7s/step - loss: 0.6509 - accuracy: 0.7171 - val_loss: 0.6220 - val_accuracy: 0.8533
Epoch 7/25
3/3 [==============================] - 22s 7s/step - loss: 0.5993 - accuracy: 0.8271 - val_loss: 0.5560 - val_accuracy: 0.8183
Epoch 8/25
3/3 [==============================] - 19s 6s/step - loss: 0.5652 - accuracy: 0.6957 - val_loss: 0.5544 - val_accuracy: 0.6567
Epoch 9/25
3/3 [==============================] - 20s 7s/step - loss: 0.5322 - accuracy: 0.7121 - val_loss: 0.5040 - val_accuracy: 0.7650
Epoch 10/25
3/3 [==============================] - 613s 204s/step - loss: 0.4823 - accuracy: 0.7786 - val_loss: 0.5043 - val_accuracy: 0.75
83
Epoch 11/25
3/3 [==============================] - 27s 9s/step - loss: 0.4396 - accuracy: 0.8071 - val_loss: 0.4485 - val_accuracy: 0.7950
Epoch 12/25
3/3 [==============================] - 20s 7s/step - loss: 0.4123 - accuracy: 0.8207 - val_loss: 0.4018 - val_accuracy: 0.8400
Epoch 13/25
3/3 [==============================] - 21s 7s/step - loss: 0.3819 - accuracy: 0.8379 - val_loss: 0.5494 - val_accuracy: 0.7517
Epoch 14/25
3/3 [==============================] - 21s 7s/step - loss: 0.4284 - accuracy: 0.8007 - val_loss: 0.4262 - val_accuracy: 0.7983
Epoch 15/25
3/3 [==============================] - 21s 7s/step - loss: 0.3777 - accuracy: 0.8343 - val_loss: 0.3728 - val_accuracy: 0.8633
```

Gambar 3. Proses *Training*

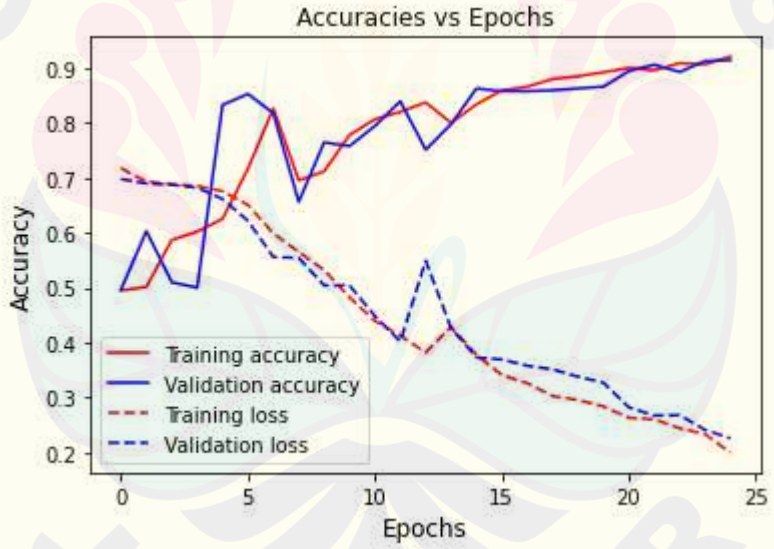
```
0.915
[[273  29]
 [ 22 276]]
              precision    recall  f1-score   support

           0       0.93      0.90      0.91       302
           1       0.90      0.93      0.92       298

    accuracy                           0.92       600
   macro avg       0.92      0.92      0.91       600
weighted avg       0.92      0.92      0.91       600

Confusion Matrix :
 [[273  29]
 [ 22 276]]
Accuracy :  0.915
Sensitivity :  0.9039735099337748
Specificity :  0.9261744966442953
False PR :  0.09602649006622517
False NR :  0.0738255033557047
```

Gambar 4. Uji Akurasi Data Testing Dengan Confusion Matrix



Gambar 5. Grafik *Training Accuracy* & *Training Loss*

Lampiran 2 ***List versi library*** **dalam** ***image classification***

Jupyter Notebook==1.0.0

Keras==2.4.3

tensorflow==1.15.0

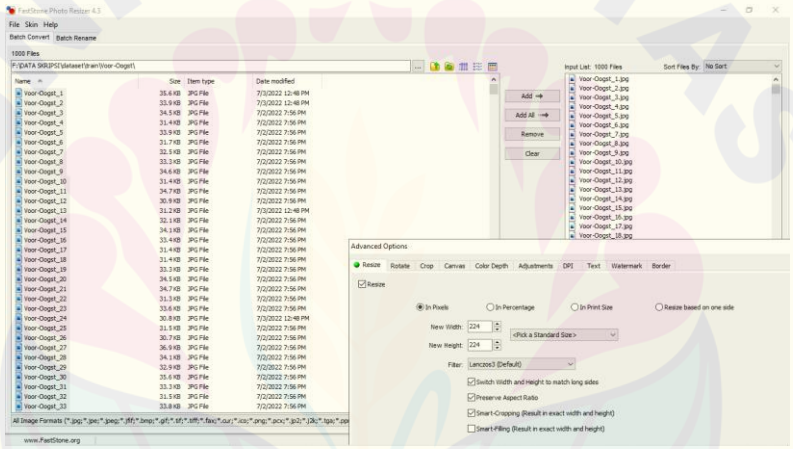
Numpy==1.23.1

Pandas==1.4.3

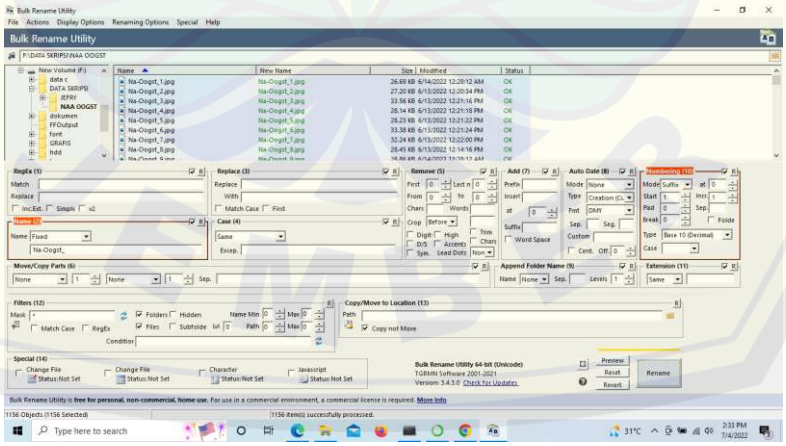
Sklearn==0.0

Tqdm==4.64.0

Lampiran 3 ***Resize & Rename***



Gambar 6. *Resize* Citra



Gambar 6. *Rename* Citra

Lampiran 4 Listing Program Model CNN

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
from os import listdir
from os.path import isfile, join
from keras.layers.convolutional import Conv2D, MaxPooling2D
from sklearn.model_selection import train_test_split
from keras.layers import Flatten, Dense, Dropout,
BatchNormalization
from keras.models import Sequential, Model
from keras import optimizers
from keras.models import load_model
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix,
classification_report
from sklearn import metrics
from sklearn.decomposition import PCA

from random import shuffle
from tqdm import tqdm
dir1 = 'C:/work/' #path to your folder
os.chdir(dir1)
train_dir = 'C://work//data//'

#Read images from train directory
onlyfiles = [f for f in listdir(train_dir) if
isfile(join(train_dir, f))]

images = []
labels = []
size = 224, 224

for file in tqdm(onlyfiles):
    if('Na-Oogst' in file):
        labels.append(0)
    else:
        labels.append(1)
    im = cv2.imread(train_dir + file,0)
    im = cv2.resize(im, (128,128))
    im = rescale=1./255,
    im = rotation_range=10,
    im = width_shift_range=0.2,
    im = height_shift_range=0.2,
    im = shear_range=0.2,
    im = zoom_range=0.2,
    im = horizontal_flip=True,
    im = vertical_flip=False
    images.append(np.array(im))

images = np.asarray(images)
```

```
labels = np.asarray(labels)
images = images/255.0

images.shape

labels

labels.shape

images_=images.reshape(2000, 128, 128,3)

from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(images_,
labels, test_size=0.3, random_state=42)


print('X Train Shape : ', train_x.shape)
print('Y Train Shape : ', train_y.shape)
print('X Test Shape : ', test_x.shape)
print('Y Test Shape : ', test_y.shape)

from sklearn.model_selection import KFold

from keras.optimizers import SGD
from sklearn.model_selection import KFold
model = Sequential()
model.add(Conv2D(32, (3, 3), activation = 'relu',
input_shape=(128,128,1), padding ='same', name='Conv_1'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3, 3), activation = 'relu', padding ='same',
name='Conv_2'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3, 3), activation = 'relu', padding ='same',
name='Conv_3'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dropout(0.3))
model.add(Dense(128, kernel_initializer='normal',
activation='relu', name='Dense_1'))
model.add(Dense(1, kernel_initializer='normal',
activation='sigmoid', name='Dense_2'))
model.compile(loss='binary_crossentropy', optimizer = 'adam',
metrics=['accuracy'])
model.summary()

history =model.fit(train_x, train_y, validation_data=(test_x,
test_y), epochs=25, batch_size=500, verbose=1)

loss, accuracy = model.evaluate(test_x, test_y)
print("Test: accuracy = %f  ;  loss = %f " % (accuracy, loss))
loss, accuracy = model.evaluate(train_x, train_y)
print("Train: accuracy = %f  ;  loss = %f " % (accuracy, loss))

plt.title('Accuracy vs Validation Accuracy')
plt.plot(history.history['accuracy'], color='red',label='Training
accuracy' )
```

```
plt.plot(history.history['val_accuracy'], color='blue',
linestyle='solid',label='Validation accuracy')
plt.ylabel('Accuracy',fontsize=12)
plt.xlabel('Epochs',fontsize=12)
plt.legend()

plt.figure()

plt.title('Loss vs Validation Loss')
plt.plot(history.history['loss'],color='red',linestyle='dashed',la
bel='Training loss ')
plt.plot(history.history['val_loss'],color='blue',linestyle='dashe
d',label='Validation loss')
plt.ylabel('Loss',fontsize=12)
plt.xlabel('Epochs',fontsize=12)
plt.legend()

plt.show()

y_train_pred=(model.predict(train_x) > 0.5)*1
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score,
classification_report,confusion_matrix, roc_auc_score, f1_score,
roc_curve
print(metrics.accuracy_score(train_y, y_train_pred))
print(confusion_matrix(train_y,y_train_pred))
print(classification_report(train_y,y_train_pred))

from sklearn.metrics import confusion_matrix

cm1 = confusion_matrix(train_y, y_train_pred)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))
accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)


FPR = cm1[0,1]/(cm1[0,1]+cm1[0,0])
print('False PR : ', FPR)

FNR= cm1[1,0]/(cm1[1,1]+cm1[1,0])
print('False NR : ', FNR)


y_test_pred = (model.predict(test_x) > 0.5)*1
```

Lampiran 5Listing Program VGG19

```
import os

base_dir = r'C:\image classification'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
train_NaOogst_dir = os.path.join(train_dir, 'Na-Oogst')
train_VoorOogst_dir = os.path.join(train_dir, 'Voor-Oogst')
Testing_NaOogst_dir = os.path.join(validation_dir, 'Na-Oogst')
Testing_VoorOogst_dir = os.path.join(validation_dir, 'Voor-
Oogst')

print('total training NaOogst images:',
len(os.listdir(train_NaOogst_dir)))
print('total training VoorOogst images:',
len(os.listdir(train_VoorOogst_dir)))
print('total Testing NaOogst images:',
len(os.listdir(Testing_NaOogst_dir)))
print('total Testing VoorOogst images:',
len(os.listdir(Testing_VoorOogst_dir)))

train_path = r'C:\image classification\train'
valid_path = r'C:\image classification\validation'

IMAGE_SIZE = [128, 128, 3]

from tensorflow.keras.applications.vgg19 import VGG19
mobilnet = VGG19(input_shape=IMAGE_SIZE, weights='imagenet',
include_top=False)

for layer in mobilnet.layers:
    layer.trainable = False

from keras.layers import Flatten
from keras.layers import Dense
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.applications.resnet50 import
preprocess_input

x = Flatten()(mobilnet.output)

from glob import glob
folders = glob(r'C:\image classification\validation\*')
print(folders)

from tensorflow.keras.models import Model
prediction = Dense(len(folders), activation='softmax')(x)

model = Model(inputs=mobilnet.input, outputs=prediction)
```

```
model.summary()

from tensorflow.keras.preprocessing.image import
ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory(train_path,
                                   target_size = (128, 128),
                                   batch_size = 500,
                                   class_mode = 'categorical')

test_set = test_datagen.flow_from_directory(valid_path,
                                   target_size = (128, 128),
                                   batch_size = 500,
                                   class_mode = 'categorical')

model.compile(
  loss='categorical_crossentropy',
  metrics=['accuracy']
)

history = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=100,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)

import matplotlib.pyplot as plt

plt.title('Accuracy vs Validation Accuracy')
plt.plot(history.history['accuracy'],
color='red',label='Training accuracy' )
plt.plot(history.history['val_accuracy'], color='blue',
linestyle='solid',label='Validation accuracy')
plt.ylabel('Accuracy',fontsize=12)
plt.xlabel('Epochs',fontsize=12)
plt.legend()

plt.figure()

plt.title('Loss vs Validation Loss')
plt.plot(history.history['loss'],color='red',linestyle='dashed
',label='Training loss ')
plt.plot(history.history['val_loss'],color='blue',linestyle='d
ashed',label='Validation loss')
plt.ylabel('Loss',fontsize=12)
```

```
plt.xlabel('Epochs',fontsize=12)
plt.legend()

plt.show()

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score,
classification_report,confusion_matrix, roc_auc_score,
f1_score, roc_curve
import sklearn.metrics as metrics
import numpy as np

y_pred_one = model.predict_generator(test_set)
y_pred = np.argmax(y_pred_one, axis=1)

y_true_labels = test_set.classes
class_labels = list(test_set.class_indices.keys())

print(metrics.accuracy_score(y_true_labels , y_pred))
print(confusion_matrix(y_true_labels , y_pred))
print(classification_report(y_true_labels , y_pred))

from sklearn.metrics import confusion_matrix

cm1 = confusion_matrix(y_true_labels , y_pred)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))
accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)


FPR = cm1[0,1]/(cm1[0,1]+cm1[0,0])
print('False PR : ', FPR)

FNR= cm1[1,0]/(cm1[1,1]+cm1[1,0])
print('False NR : ', FNR)

from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score,
classification_report,confusion_matrix, roc_auc_score,
f1_score, roc_curve
import sklearn.metrics as metrics
import numpy as np

y_pred_one = model.predict_generator(training_set)
y_pred = np.argmax(y_pred_one, axis=1)

y_true_labels = training_set.classes
class_labels = list(training_set.class_indices.keys())

print(metrics.accuracy_score(y_true_labels , y_pred))
print(confusion_matrix(y_true_labels , y_pred))
print(classification_report(y_true_labels , y_pred))

from sklearn.metrics import confusion_matrix

cm1 = confusion_matrix(y_true_labels , y_pred)
print('Confusion Matrix : \n', cm1)

total1=sum(sum(cm1))
accuracy1=(cm1[0,0]+cm1[1,1])/total1
print ('Accuracy : ', accuracy1)

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])
print('Specificity : ', specificity1)


FPR = cm1[0,1]/(cm1[0,1]+cm1[0,0])
print('False PR : ', FPR)

FNR= cm1[1,0]/(cm1[1,1]+cm1[1,0])
print('False NR : ', FNR)
```