Deteksi Penyakit Tanaman Cabai Menggunakan Metode Convolutional Neural Network

Athallah Tsany Rakha Dzaky[1] Wikky Fawwaz Al Maki[2]

[12]Fakultas Informatika, Universitas Telkom, Bandung
[1]rakhadzaky@students.telkomuniversity.ac.id,
[2]wikkyfawwaz@telkomuniversity.ac.id

Abstrak

Penyakit dari suatu tanaman akan sangat mempengaruhi hasil panen dari tanaman tersebut. Jika penyakitnya tidak segera ditangani maka penyakit tersebut dapat merusak tanaman dan mengakibatkan gagal panen yang akan berpengaruh ekonomi. Maka dari itu deteksi penyakit tanaman menjadi hal yang sangat penting dalam proses perawatan tanaman. Tanaman cabai merupakan salah satu bahan makanan yang paling sering digunakan dalam berbagai masakan di Indonesia. Banyaknya permintaan maka akan diperlukan adanya penambahan lahan cabai. Semakin luas lahannya maka akan semakin besar usaha yang diperlukan untuk merawat dan mengawasi tanaman. Dengan perkembangan teknologi saat ini, memungkinkan dilakukannya pengawasan terhadap tanaman secara otomatis menggunakan sistem komputer. Dengan menggunakan *image processing* penyakit yang dapat dilihat dan direkam oleh kamera akan dapat dianalisis dan diidentifikasikan oleh komputer. Dengan begitu pengawasan tanaman akan menjadi lebih mudah dan efisien. Beberapa penyakit tanaman cabai yang terlihat dan cukup sering ditemui adalah virus kuning, keriting mosaik, dan layu. Ciri penyakit tersebut bisa dilihat dari bentuk dan warna daun. Pada penelitian ini dilakukan pengenalan penyakit dengan menggunakan algoritma CNN. CNN digunakan agar model dapat mengekstrak fitur dan melakukan klasifikasi citra secara otomatis. Data citra yang digunakan diambil langsung dari perkebunan cabai di Jawa Tengah. Data diambil sekitar pukul 10.00 – 12.00 siang, sehingga citra memiliki warna yang jelas. Model ini akan bisa mengklasifikasikan 4 jenis kondisi daun dengan 3 penyakit dan 1 kondisi normal. Model CNN ini bisa menghasilkan akurasi diatas 90% menggunakan arsitektur AlexNet.

Kata kunci : Tanaman Cabai, Penyakit Tanaman, CNN, Neural Network, AlexNet.

Abstract

Disease from a plant will greatly affect the harvest of that plant. If the disease is not treated immediately, it can damage the crops and resulting in crop failure which will affect the economy. Therefore detection of plant diseases is very important. Chili plants are one of the most commonly used food ingredients in various dishes in Indonesia. With a large number of requests, it will require additional chili fields. The more extensive the land, the greater the effort required to care for and maintain the plants. With current technological developments, it is possible to observe plants disease automatically using a computer system. By using image processing, disease that can be seen and recorded by the camera can be analyzed and identified by a computer. That way, crop disease identification will become easier and more efficient. Some of the diseases that are seen and often encountered are yellow virus, curl mosaic, and wilting. The characteristics of the disease can be seen from the shape and color of the leaves. In this study, disease recognition was carried out using the CNN algorithm. CNN is used so that the model can extract features and perform image classification automatically. The image data used were taken directly from chili plantations in Central Java. Data is taken around 10:00 - 12:00 AM, so the image has a clear color. This model will be able to classify 4 types of leaf conditions with 3 diseases and 1 normal condition. This CNN model can produce an accuracy over 90% using AlexNet architecture.

Keyword : Chili Plant, Plant Disease, CNN, Neural Network, AlexNet.

1. Pendahuluan

1.1. Latar Belakang

Deteksi penyakit tanaman adalah salah satu hal terpenting dalam pemeliharaan dan perawatan tanaman. Penyakit yang tidak terdeteksi dan dibiarkan berkembang akan mengakibatkan kerusakan pada tanaman. Hal ini

akan menjadi lebih penting jika tanaman yang sedang dirawat adalah tanaman pangan. Kerusakan tanaman akan mengakibatkan penurunan kualitas atau kuantitas hasil panen. Penurunan hasil panen bisa berdampak pada ekonomi. Beberapa penelitian telah dilakukan untuk mendapatkan pengetahuan mengenai berbagai penyakit tanaman[10]. Berkat penelitian tersebut, perawatan tanaman menjadi lebih mudah dengan mengetahui ciri dari setiap penyakit, sumber penyakit, cara pengobatan hingga cara untuk mencegah penyakit tersebut muncul pada tanaman.

Berkembangnya ilmu pengetahuan saat ini telah mendorong ditemukannya cara untuk mendeteksi penyakit pada tanaman secara otomatis menggunakan komputer. Deteksi penyakit menggunakan komputer cukup direkomendasikan dikarenakan deteksi yang dihasilkan komputer dirasa cukup akurat[3,6]. Beberapa penelitian mengenai deteksi penyakit tanaman juga sudah banyak dilakukan [1-8]. Beberapa diantaranya menggunakan metode *Deep Learning*[2,11-16].

Deep Learning merupakan salah satu cabang *Machine Learning* yang menggunakan metode *Nerural Network* untuk menyelesaikan suatu masalah atau kasus yang diberikan. CNN *(Convolutional Neural Network)* merupakan salah satu algoritma *deep learning* yang cukup sering digunakan untuk mengatasi masalah klasifikasi citra[13]. CNN dapat mendapatkan tingkat akurasi klasifikasi yang cukup tinggi walaupun dengan minimal *preprocessing* maupun segmentasi [2,16]. Hasil akurasi dari algoritma CNN juga tidak kalah dari algoritma lainnya[14,15]. Maka dari itu dilakukanlah penelitian ini, yang merupakan penelitian tentang deteksi penyakit dari tanaman menggunakan metode CNN (*Convolutional Neural Network*).

1.2. Topik dan Batasannya

Pada penelitian ini, akan dilakukan deteksi penyakit tanaman cabai. Metode yang digunakan adalah *Deep Learning* menggunakan CNN (*Convolutional Neural Network*). Data masukan yang digunakan adalah citra dari daun tanaman yang memiliki penyakit maupun yang tidak, citra tesebut kemudian akan diklasifikasikan untuk mengetahui ada atau tidaknya penyakit pada daun tanaman cabai tersebut.

Pada penelitian ini, tanaman yang digunakan adalah tanaman cabai dengan 3 jenis penyakit yaitu *yellow virus*, keriting mosaik, dan layu, ditambah dengan 1 daun normal, sehingga total menjadi 4 kelas. Klasifikasi yang dilakukan hanya untuk mendeteksi satu penyakit dari satu citra daun. Data citra yang diambil memilik pencahayaan yang cukup jelas.

1.3. Tujuan

Penelitian ini memiliki tujuan untuk menciptakan model deteksi penyakit pada daun tanaman cabai yang dapat digunakan untuk mengklasifikasi citra yang diambil langsung dari kebun.

2. Studi Terkait

2.1. Penelitian Terdahulu

Di Jepang dilakukan uji model pendeteksian penyakit pada daun mentimun dengan judul *Basic Study of Automated Diagnosis of Viral Plant Diseases Using Convolutional Neural Network* yang dilakukan oleh Yusuke Kawasaki dan Hitoshi Iyatomi yang berasal dari *School of Science and Engineering Hosei University* beserta Hiroyuki Uga dari *Saitama Argicultural Technology Research Center* dan Satosthi Kagiwada dari *Faculty of Bioscience and Applied Chemistry Hosei University*. Mereka melakukan uji deteksi penyakit *Melon Yellow Spot Virus*, dan *Zucchini Yellow Mosaic Virus* menggunakan CNN *(Convolutional Neural Network)* yang dibantu dengan proses rotasi citra pada kelipatan 10 derajat sebanyak 36 kali, hal ini dilakukan agar CNN mendapatkan berbagai macam *Local Features* dan akan meningkatkan peforma klasifikasi. Hasil akhir penelitian memiliki akurasi 92.5% [2].

Pada tahun 2012 di Malaysia dilakukan percobaan deteksi penyakit Kuning pada daun cabai dengan judul *Feasibility Study on Plant Chili Disease Detection Using Image Processing Techniques* oleh Zulkifli Bin Husin, Abdul Hallis Bin Abdul Aziz, dan Rohani Binti S Mohamed Farook yang berasal dari *School of Computer and Communication Engineering* beserta Ali Yeon Bin Md Shakaff dari *School of Mechatronic Engineering*. Mereka melakukan penelitian untuk mendeteksi penyakit Kuning pada tanaman cabai. Mereka merekomendasikan sistem deteksi penyakit pada tanaman melalui daun yang mereka buat terutama untuk membantu pekerjaan para petani [3].

Pada tahun 2017 di Belanda terdapat penelitian mengenai *Data Augmentation* terhadap proses klasifikasi tanaman. Penelitian ini dilakukan oleh Pornntiwa Pawara, Emmanuel Okafor, Lambert Schomaker, dan Macro Wiering dari *Institute of Artificial Intelligence and Cognitive Engineering* (ALICE), University of Groningen. Mereka melakukan penelitian untuk mengetahui proses *data augmentation* mana yang akan meningkatkan performa klasifikasi paling tinggi. Arsitektur yang digunakan dalam penelitian mereka adalah AlexNet dan GoogleNet. Penelitian mereka menyimpulkan bahwa rotasi dan perubahan kontras merupakan kombinasi yang memiliki kemampuan meningkatkan performa model paling tinggi [16].

Sudah ada penelitian tentang pendeteksian penyakit pada tanaman cabai oleh Abdul Hafiz Bin Abdul Wahab, Rahimi Zahari, dan Tiong Hoo Lim yang berasal dari Univeristi Teknologi Brunei. Mereka melakukan penelitian menggunkan berbagai metode SVM *(Support Vector Machine)*, diantaranya adalah *Linear SVM, Quadratic SVM, Cubic SVM, Fine Gaussian SVM, Medium Gaussian SVM, Coarse Gaussian SVM.* Mereka membuat model dengan metode SVM untuk menentukan apakah suatu tanaman cabai terkena penyakit atau tidak dengan menganaslisa dari warna daun tanaman tersebut. Hasil dari masing-masing metode memiliki akuras yang berbeda dengan rata-rata akurasi 85% [1].

2.2. Penyakit Daun Tanaman Cabai

2.2.1. Virus Kuning

Merupakan penyakit yang menyerang bagian daun tanaman cabai yang disebabkan oleh virus Gemini. Gejalanya dimulai dengan pemutihan pada tulang daun dan kemudian warna daun akan berubah menjadi warna kuning, bagian tulang daun akan menebal, dan daun akan mengeriting ke atas[8], contoh citra dapat dilihat pada Gambar 1. Penyakit ini ditularkan oleh kutu kebul yang sebelumnya berada pada tanaman yang terkena penyakit, sehingga populasi kutu kebul yang tinggi akan mempengaruhi perkembangan penyakit ini. Selain itu bisa juga disebabkan oleh bibit yang memang sudah terserang penyakit[8].

Gambar 1 Virus Kuning

2.2.2. Keriting Mosaic

Penyakit yang disebabkan oleh *Cucumber Mosaic Virus* (CMV). Gejala yang ditimbulkan adalah daun menjadi berwarna belang antara hijau tua dan hijau muda. Terkadang disertai dengan perubahan bentuk daun seperti menjadi cekung, keriting, atau memanjang, contoh daun dapat dilihat pada Gambar 2. Penyakit ini ditularkan oleh kutu daun. Salah satu cara pencegahannya adalah dengan menggunakan insektisida untuk mengendalikan populasi kutu daun[8].

Gambar 2 Keriting Mosaik

2.2.3. Layu

Penyakit yang disebabkan oleh patogen *Fusarium Oxysporum* ataupun patogen *Ralstonia Solanacearum*. Ditandai dengan menguning dan layunya daun dimulai dari bagian atas, conoth citra dapat dilihat pada Gambar 3. Secara bertahap gejala akan menyebar keseluruh tanaman dan menyebabkan kelayuan permanen dengan daun yang masih menempel.

Gambar 3 Layu

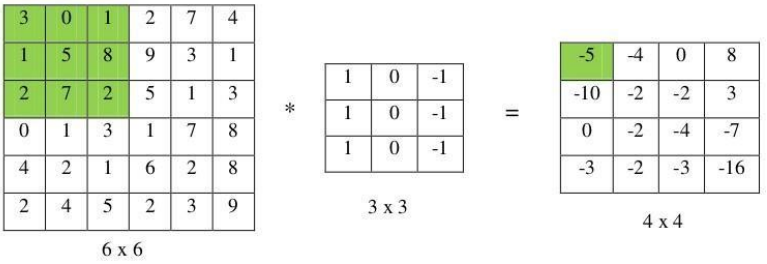
2.3. Convolutional Neural Network

CNN (*Convolutional Neural Networks*) adalah salah satu metode pembelajaran mesin yang menggunakan

neural network dan biasa digunakan untuk mengolah data citra[9]. Data citra yang digunakan sebagai data masukan, biasa memiliki ukuran Panjang × Lebar × *channel* warna. Panjang dan lebar menunjukan ukuran citra dalam satuan *pixel*. Sementara untuk *channel* warna akan menunjukan citra memiliki jenis warna apa. Sebagai contoh citra yang memiliki hanya 1 *channel* merupakan citra *Grayscale*, dimana citra hanya menampilkan 1 jenis warna dan biasa disebut dengan monokrom. Citra yang memiliki warna normal biasanya terdiri dari 3 *channel*. Setiap *channel* mewakili warna dari jenis warna dasar yaitu RGB (*Red Green Blue*). Sama seperti *neural network* lainnya, CNN juga memiliki bobot, bias, dan fungsi aktivasi yang membuat *neural network* bisa mengeluarkan sebuah keluaran yang sesuai dengan data latih yang diberikan.

2.3.1. Lapisan Konvolusi

Lapisan Konvolusi (*Convolution Layer*) merupakan lapisan yang akan dijumpai pertama kali dalam CNN. Lapisan ini akan melakukan konvolusi citra masukan dengan *filter* yang sudah didefinisikan tanpa merusak struktur dari citra awal. Lapisan ini memiliki fungsi untuk mengambil fitur pada citra yang akan digunakan untuk melatih model. Conoth proses dari lapisan ini dapat dilihat pada Gambar 4.



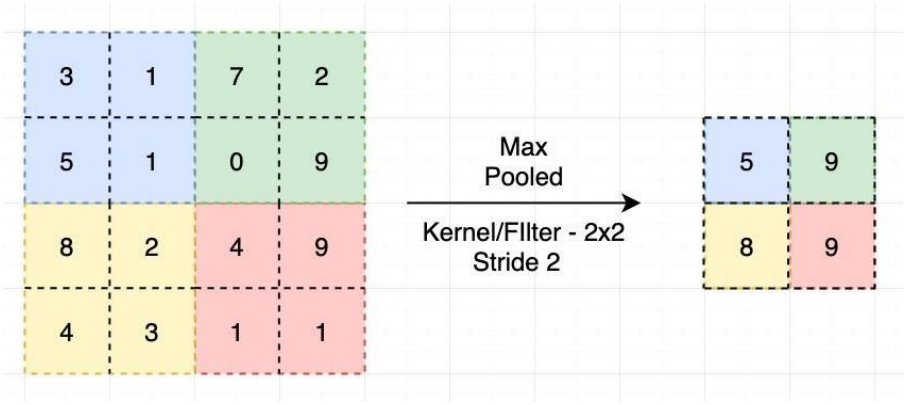
Gambar 4 Contoh Proses Konvolusi (sumber : https://rizkynovyantika.com/)

2.3.2. Lapisan *Batch Normalization*

Normalisasi merupakan sebuah proses *standardization* nilai pada suatu data. Seluruh nilai pada data akan dimasukan kedalam persamaan *batch normalization* yang akan menghasilkan nilai baru. Proses ini akan membuat seluruh data berada dalam skala standar tertentu. Lapisan ini berguna untuk mencegah adanya perbedaan nilai yang besar.

2.3.3. Lapisan *Pooling*

Lapisan ini digunakan untuk mengurangi dimensi data yang akan digunakan. Lapisan ini sangat berguna jika ukuran dari suatu citra sangat besar. Ada beberapa jenis *pooling* yang bisa digunakan. Setiap jenis *pooling* memiliki karakteristik yang berbeda karena informasi yang disimpan juga berbeda. *Max Pooling*, akan menyimpan nilai yang tebesar dari suatu kelompok data untuk dilanjutkan ketahap selanjutnya. Jenis ini akan menghilangkan nilai lain yang lebih rendah dari nilai maksimal tersebut. *Average Pooling*, jenis *pooling* ini akan mencari rata-rata dari seluruh nilai pada suatu kelompok data. Kedua jenis *pooling* tersebut merupakan yang paling sering digunakan. Contoh proses yang terjadi pada *max pooling* dapat dilihat pada Gambar 5.

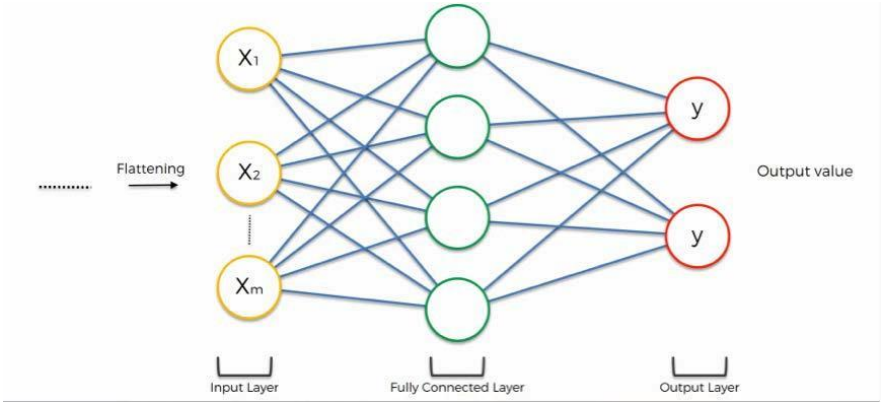


Gambar 5 Contoh Proses Pooling pada *Max Pooling* (sumber : https://medium.com/ai-in-plain-english)

2.3.4. Lapisan *Fully Connected*

Lapisan ini merupakan awal dari *Neural Network* dimana dimensi untuk semua data akan diubah menjadi 1 dimensi. Perubahan ukuran dimensi ini dinamakan *flatten*. Sama seperti *neural network* pada umumnya, lapisan ini juga memiliki *node* yang saling terhubung dan memiliki bobot serta fungsi aktivasi. Akhir dari layer

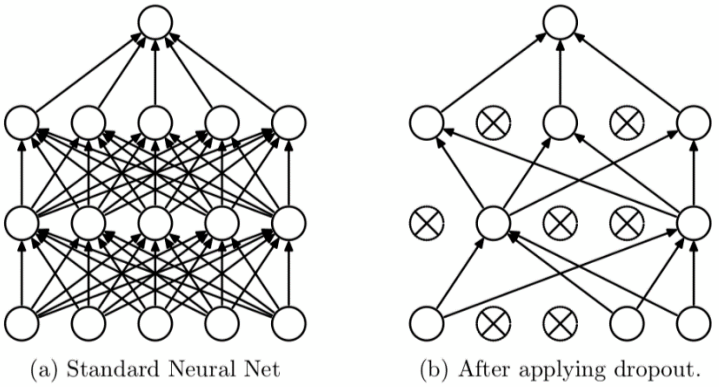
ini merupakan keluaran berbentuk sebuah prediksi berdasarkan data yang sebelumnya kita masukan. Gambaran struktur lapisan ini dapat dilihat pada Gambar 6.



Gambar 6 Contoh Proses *Fully Connected* (sumber : https://www.superdatascience.com/)

2.3.5. Lapisan *Dropout*

Pada lapisan ini terjadi proses pemilihan *neuron* secara acak untuk dinonaktifkan. Dengan berubahnya status suatu *neuron* menjadi nonaktif, maka pengaruh *neuron* tersebut akan dihilangkan pada proses *back-propagation*. Proses ini dilakukan untuk mengurangi kemungkinan *overfitting* serta mempercepat proses *learning*. Perbedaan proses penggunaan *dropout* dan tidak dapat dilihat pada Gambar 7.

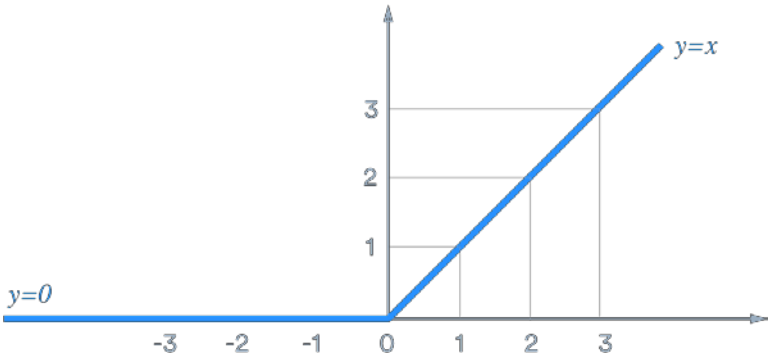


Gambar 7 Contoh Proses *Dropout* (sumber : https://medium.com/@amarbudhiraja/)

2.3.6. Fungsi Aktivasi ReLU (Rectified Linear Activation Function)

Fungsi aktivasi memiliki tugas untuk mengubah nilai bobot yang masuk dari setiap *node* yang akan menjadi sebuah acuan aktif atau tidaknya *node* tersebut. Fungsi ReLU sendiri merupakan salah satu fungsi aktivasi yang cukup populer dan sering digunakan pada proses *neural network*. Proses yang terjadi pada fungsi ini cukup sederhana. Fungsi ini akan mengeluarkan nilai sesuai dengan nilai masukan jika nilai tersebut positif, dan akan mengeluarkan nilai 0 jika nilai masukan berupa bilangan negatif.

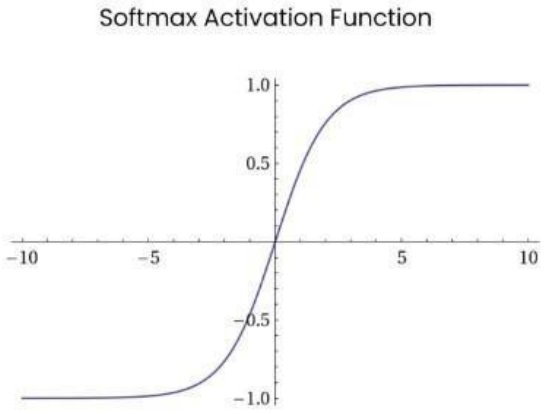
ReLU awalnya diciptakan untuk menghindari tahap jenuh dari pembelajaran yang merupakan salah satu kekurangan dalam fungsi *Sigmoid* dan *Tanh*. Pada fungsi *sigmoid*, keluaran yang diberikan merupakan sebuah bilangan yang berada pada rentang 0 hingga 1. Jika nilai dari masukan berada dibawah 0, maka fungsi ini akan mengeluarkan 0. Begitu juga jika nilai lebih tinggi dari 1, maka fungsi ini akan mengeluarkan nilai 1. Selain itu, jika nilai berada diantara kedua bilangan tersebut, maka fungsi akan mengeluarkan nilai sesuai dengan nilai masukan. Disisi lain, *Tanh* juga memilki fungsi yang mirip dengan *sigmoid*. *Tanh* memiliki batas yang berbeda dengan *sigmoid*, yaitu rentang *Tanh* ada pada -1 hingga 1. Kedua fungsi ini membatasi keluaran yang mereka berikan. Ternyata hal tersebut berdampak pada kejenuhan dari proses pembelajaran mesin. Kejenuhan dalam pembelajaran mesin akan menyebabkan model kesulitan untuk beradaptasi pada bobot untuk meningkatkan performanya. Jika fungsi ReLU dibuat grafik maka akan terlihat seperti Gambar 8.



Gambar 8 Grafik Fungsi ReLU (sumber : https://medium.com/@sonish.sivarajkumar)

2.3.7. Fungsi Aktivasi *Softmax*

Softmax merupakan fungsi aktivasi yang mengubah nilai angka menjadi nilai *probability* dengan skala tertentu yang proporsional. Fungsi ini sering digunakan pada *neural network* yang memilki kategori output yang banyak (*multi-class*). Hal ini dikarenakan fungsi ini mengubah nilai perhitungan menjadi nilai *probability* sehingga nilai hasil perhitunngan bisa saling dibandingkan. Dengan cara ini, maka akan bisa dilihat kelas mana yang memiliki nilai kemungkinan yang paling besar. Kelas yang memiliki nilai kemungikinan paling besar akan menjadi kelas terpilih, dan data masukan akan terklasfikasi sebagai kelas tersebut. Grafik dari lapisan ini dapat dilihat pada Gambar 9.



Gambar 9 Grafik Fungsi Aktifasi Softmax (sumber : https://krisbolton.com/a-quick-introduction-to-artificial-neural-networks-part-2)

2.3.8. Loss Function Sparse Categorical Cross-entropy

Loss adalah salah satu nilai dari prediksi tingkat *error* pada *neural network* saat melakukan pelatihan dalam pembuatan model. Fungsi untuk mencari nilai tersebut dinamakan sebagai *Loss Function*. Fungsi ini memiliki tugas untuk mencari nilai gradien atau nilai yang menunjukan seberapa cepat nilai suatu *variable* berubah. Nilai tersebut nantinya akan digunakan untuk memperbarui nilai bobot.

Categorical Cross-entropy (CC), meruapakan salah satu *loss function* yang biasa digunakan dalam klasifikasi yang memiliki kategori *multi-class*. Sedangkan untuk *Sparse Categorical Cross-entropy* (SCC) memiliki fungsi yang sama seperti *Categorical Cross-entropy*, hanya saja pada fungsi SCC data yang digunakan tidak dilakukan *Hot-Encoding* terlebih dahulu.

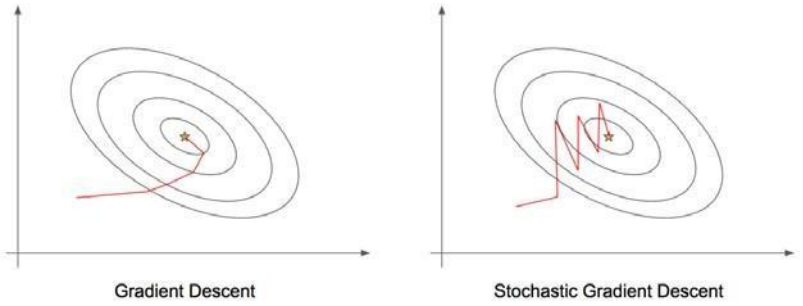
$$\text{[illegible]}(\text{[illegible]}, \text{[illegible]}) = -\sum_{\text{[illegible]}=1} \text{[illegible]} \log(\text{[illegible]})$$

(1)

2.3.9. Optimizer Stochastic Gradient Descent (SGD)

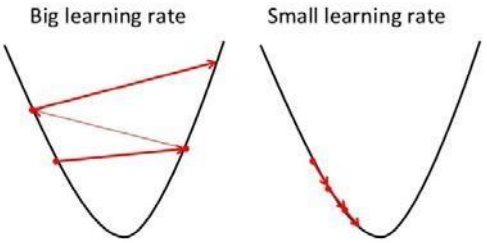
Gradient Descent merupakan sebuah algoritma berulang yang diawali dengan memunculkan titik random pada suatu fungsi. Kemudian algoritma akan menuruni kemiringan yang ada pada titik tersebut hingga

mencapai titik paling rendah dari fungsi tersebut. Algoritma ini akan mendeteksi tingkat kemiringan dengan mencari gradien pada titik tersebut. Berkaitan dengan gradien, maka jika titik sudah berada dalam posisi yang lumayan landai, perubahan nilai (*step size*) akan semakin pelan. Sementara *Stochastic Gradient Descent* merupakan versi peningkatan dari *Gradient Descent*. Perbedaan SGD dan GD terletak pada proses pemilihan suatu titik data pada setiap iterasi. GD akan menghitung keseluruhan data, sementara SGD memanfaatkan fungsi random untuk memilih suatu titik data secara acak. Dengan proses ini, SGD dapat melakukan perhitungan lebih cepat. Perbandingan SGD dan GD dapat dilihat pada Gambar 10.



Gambar 10 Grafik GD dan SGD (sumber : https://www.kaggle.com/twistcodes/)

Learning rate, merupakan *parameter* dari *optimizer* yang memepengaruhi kecepatan model dalam menemukan solusi dari data yang diberikan. *Learning rate* akan mengontrol seberapa banyak bobot akan diubah dalap algoritma optimasi. Semakin kecil nilai *learning rate* maka proses pembelajaran akan semakin lambat, namun teliti. Sementara jika nilai *learning rate* tinggi, maka proses pembelajaran akan semakin cepat namun rendah ketelitian. Gambaran dari perbedaan *learning rate* besar dan kecil dapat dilihat pada Gambar 11.



Gambar 11 Grafik *Learning Rate* (sumber : https://www.educative.io/edpresso/learning-rate-in-machine-learning)

2.3.10. Epoch

Epoch merupakan nilai yang menunjukan banyaknya iterasi yang diperlukan oleh model dalam melakukan pembelajaran. Semakin banyak *epoch* maka pengulangan dalam pembelajaran akan semakin banyak juga. Dengan kata lain, berbagai pola yang berbeda akan bisa didapatkan. Namun semakin banyak pengulangan artinya pembelajaran juga akan semakin lama.

2.3.11. Arsitektur AlexNet

Arsitektur yang digunakan dalam penelitian kali ini adalah AlexNet[16]. AlexNet merupakan arsitektur CNN yang didesain oleh Alex Krizhevsky Bersama dengan rekannya Ilya Sutskever dan Geoffrey Hinton. AlexNet terdiri dari 11 layer, dengan 5 layer *Convolutional*, 3 layer *Max Pooling*, dan 3 layer *Fully Connected*. Secara keseluruhan layer yang ada pada AlexNet dapat dilihat pada Tabel 1. Fungsi aktivasi yang digunakan pada arsitektur ini merupakan fungsi ReLU.

Tabel 1 Arsitektur CNN AlexNet

Layer
Convolutional
Max Pooling
Convolutional
Max Pooling

Convolutional
Convolutional
Convolutional
Max Pooling
Fully Connected
Fully Connected
Fully Connected

2.4. GrabCut

Merupakan salah satu teknik untuk memisahkan *foreground* dan *background*. GrabCut dibuat oleh Vladmir Kolmogorov. Dalam algoritma ini, pertama-tama diperlukan sebuah kotak yang menunjukan suatu objek *foreground* yang diinginkan. Kotak ini selanjutnya akan disebut *Region of Interest* (ROI). Seluruh pixel yang berada diluar ROI akan dianggap sebagai *background*, sementara yang berada di dalam ROI masih belum diketahui (*Unknown*). Pixel yang berada didalam ROI selanjutnya akan diberi label "mungkin *foreground*" atau "mungkin *background*" menggunakan *Gaussian Mixture Model* (GMM). Kemudian akan dibentuk Graph berdasarkan hasil label yang ditetapkan sebelumnya Graph memiliki node berupa *pixel*, ditambah dengan 2 node tambahan yang dinamai *source node* dan *sink node*. Setiap *pixel foreground* akan terhubung dengan *source node*. Sementara *pixel background* akan terhubung dengan *sink node*. Bobot hubungan antara *node* dengan *source node* atau *sink node* tergantung dengan label yang didapatkan dari GMM. Sementara bobot antara pixel, ditentukan oleh kesamaan *pixel*. Jika *pixel* tersebut memiliki banyak perbedaan maka pixel tersebut akan dianggap *edge*. Tahap selanjutnya adalah tahap yang akan memisahkan *foreground* dan *background*. Pemotongan ini dilakukan dengan memperhatikan node yang terhubung dengan *source node* dan *sink node*, yang dibantu dengan minimum *cost function*. *Cost function* didapatkan dari penjumlahan bobot dari pixel yang dianggap *edge*. Seluruh proses ini akan terus berulang hingga jumlah perulangan yang ditentukan [17]. Conoth penggunaan *GrabCut* dan hasilnya dapat dilihat pada Gambar 12.

Gambar 12 Contoh Hasil GrabCut pada daun tanaman

2.5. Confusion Matrix

2.5.1. Akurasi (Acc)

Merupakan presentase perbandingan antara jumlah data yang diklasifikasikan benar dengan jumlah seluruh data. Menggambarkan seberapa akurat model yang dibuat dapat memprediksi jawaban dengan benar. Persamaan matematika dari akurasi dapat dilihat pada Persamaan 3.

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (2)$$

2.5.2. Recall

Merupakan presentase perbandingan antara jumlah data yang diklasifikasikan benar dengan jumlah total data yang diklasifikasikan. Bisa dikatakan, *recall* merupakan keberhasilan model dalam menemukan Kembali informasi. Persamaan matematika dari *recall* dapat dilihat pada Persamaan 4.

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

2.5.3. Precission

Merupakan presentase perbandingan antara jumlah data yang diklasifikasikan benar positif dengan jumlah total data yang diprediksi positif. Menggambarkan tingkat keakuratan model dalam memprediksi terhadap data yang diminta. Persamaan dari *precission* dapat dilihat pada Persamaan 5.

$$= \frac{}{+}$$

(4)

2.5.4. F1 Score

Merupakan presentase perbandingan antara rata-rata *precision* dan *recall.* Menunjukan tingkat keselarasan antara *Recall* dan *Precision*. Dengan tingginya F1, artinya kedua *Recall* dan *Precission* bisa dikatakan memiliki tingkat yang tinggi juga. Persamaan matematika dari F1 dapat dilihat pada Persamaan 6.

$$1 = 2 \times \frac{\times}{+}$$

(5)

3. Sistem yang Dibangun

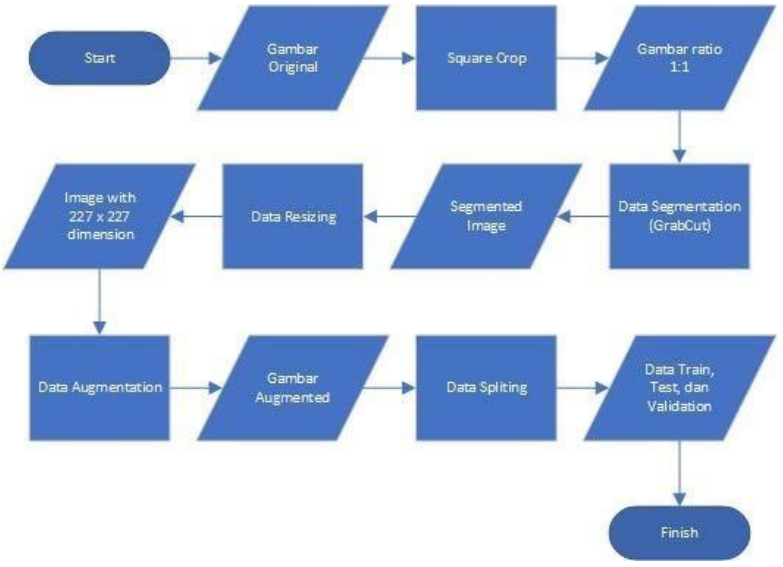
3.1. Image Acquisition

Citra diambil sekitar jam 10.00 hinga 12.00 siang dikebun. Citra memiliki *background* berupa kebun, dan terfokus pada 1 daun tanaman. Data memiliki 4 kelas yang berbeda, 3 kelas untuk daun dengan penyakit *yellow virus*, keriting mosaik, dan layu. Sementara 1 kelas yang tersisa digunakan untuk mengetahui apakah daun tersebut normal atau tidak. Citra diambil menggunakan kamera *smartphone* dengan ukuran 6 MP memiliki hasil seperti pada Gambar 13.

Gambar 13 Contoh dataset

3.2. Preprocessing

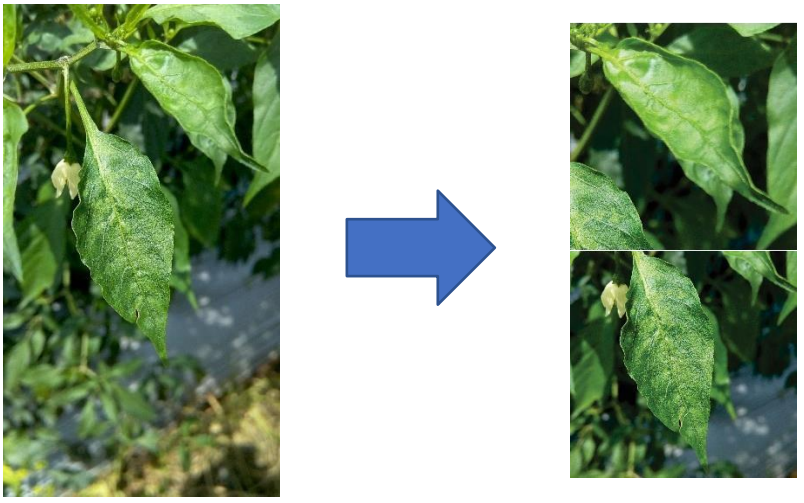
Proses *preprocessing* merupakan proses yang dilakukan untuk mempersiapkan data sebelum memasuki proses utama agar model yang dihasilkan bisa maksimal. Tahap preprocessing yang digunakan dalam penelitian ini secara keseluruhan dapat dilihat pada *flowchart* Gambar 14.



Gambar 14 Flowchart *Preprocessing*

Square Cropping

Square Cropping adalah proses pemotongan citra dengan *ratio* 1:1. Pemotongan dilakukan agar citra menjadi lebih fokus kepada bagian daunnya. Pada proses ini, ukuran dari hasil pemotongan tidak terlalu diperhatikan, dikarenakan ukuran dari seluruh citra pada saat dimasukan kedalam model CNN akan dikonversi dulu menjadi 227 px × 227 px. Pada Gambar 15 terlihat gambaran proses dari tahap ini.



Gambar 15 Gambaran Proses *Square Cropping*

Selain memfokuskan citra pada bagian daun, proses ini juga memiliki tujuan untuk memperbanyak *dataset*. Terlihat pada Gambar 15 bahwa ada beberapa citra yang memiliki citra daun lebih dari 1 dan terlihat jelas. Dengan melakukan pemotongan pada setiap citra daun yang jelas, maka akan diperoleh data baru yang bisa dimasukan kedalam data latih. Jumlah total data yang dihasilkan dari proses ini adalah 461 citra dengan detail yang dapat dilihat pada Tabel 2.

Tabel 2 Total jumlah dataset setelah proses *square cropping*

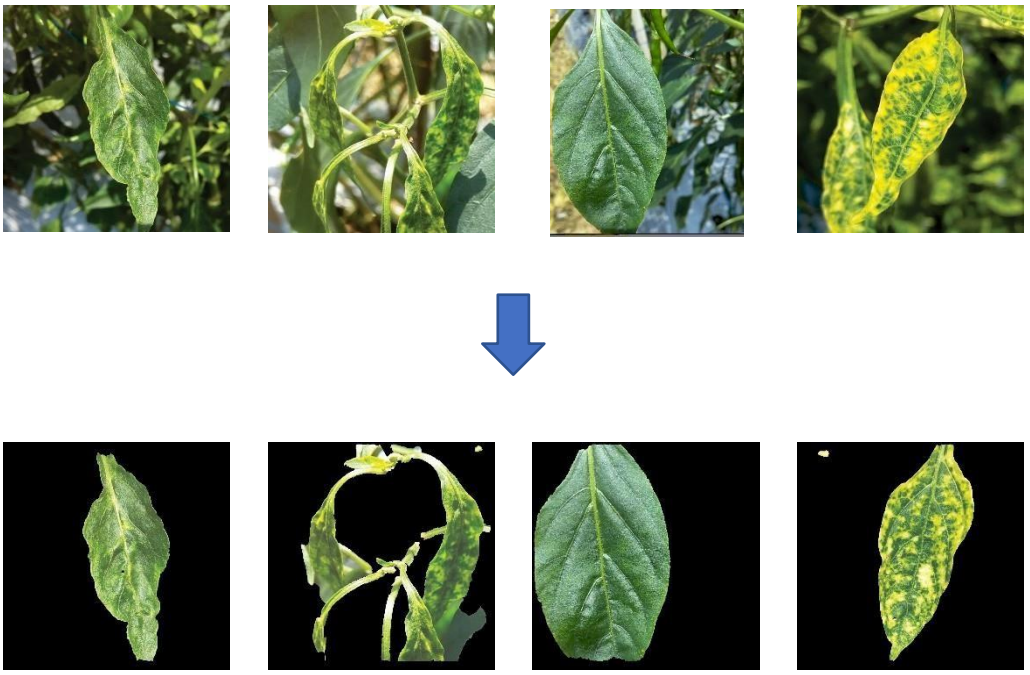
Kelas	Jumlah
Keriting	118
Layu	116
Virus Kuning	115
Normal	112
Total	461

Data Segmentation

Proses segmentasi dilakukan menggunakan algoritma GrabCut[17] untuk memisahkan *foreground* yang dalam kasus ini merupakan citra daun dan *background*-nya. ROI dibuat mulai dari pixel [10,10] dengan lebar merupakan hasil dari pengurangan 30 px terhadap lebar asli, dan tinggi merupakan hasil pengurangan 50 px terhadap tinggi asli.

$$h_{\square} = h - 30$$
$$h_{\square} = h - 50 \quad (6)$$

Pengaturan panjang dan lebar dibuat cukup luas, hal ini dikarenakan pada tahap sebelumnya, *Square Cropping*, telah dilakukan pemotongan yang menyebabkan citra sudah terfokus pada daunnya saja. Segmentasi GrabCut dilakukan dengan 5 kali iterasi tanpa bantuan *mask* buatan manual. Hasil citra yang diperoleh bisa dilihat pada Gambar 16.



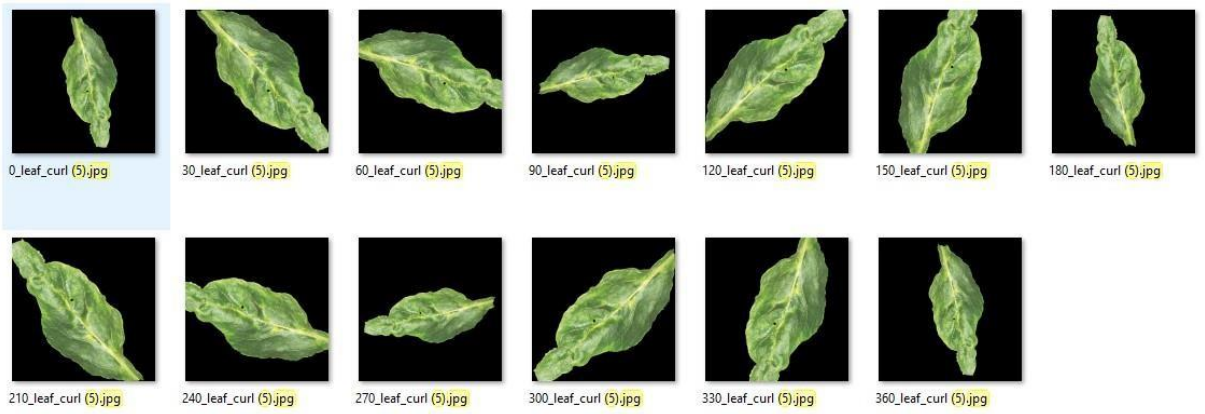
Gambar 16 Gambaran Proses *Segementasi GrabCut*

Data Resizing

Data *resizing* adalah proses mengubah ukuran dimensi citra. Hal ini dilakukan karena pada arsitektur AlexNet, input data yang diizinkan memiliki ukuran 227 px × 227 px. Jadi pada tahap ini seluruh ukuran citra akan disama-ratakan menjadi 227 px × 227 px. Perubahan ukuran dilakukan sebelum *Data Augmentation*, hal ini dilakukan untuk mengurangi beban iterasi yang diperlukan pada saat mengubah ukuran.

Data Augmentation

Data Augmentation, merupakan salah satu proses yang bisa membantu meningkatkan peforma model dengan cara menambahkan dataset secara kuantitas. Proses penambahan data menggunakan teknik rotate dan crop. Setiap citra akan di rotate dengan interval 30° sehingga akan menghasilkan 13 jenis citra yaitu pada derajat 0, 30, 60, 90, 120, 150, 180, 210, 240, 270, 300, 330, dan 360. Gambaran dari hasil yang didapat akan terlihat seperti Gambar 17.



Gambar 17 Hasil Data Augmentation Pada Daun Penyakit Curl

Penambahan data ini membuat total data citra menjadi 5.980 dengan detail seperti pada Tabel 3.

Tabel 3 Total Number of Dataset Augmented

Kelas	Jumlah
Keriting	1.521
Layu	1.508
Virus Kuning	1.456
Normal	1.495
Total	5.980

Data Splitting

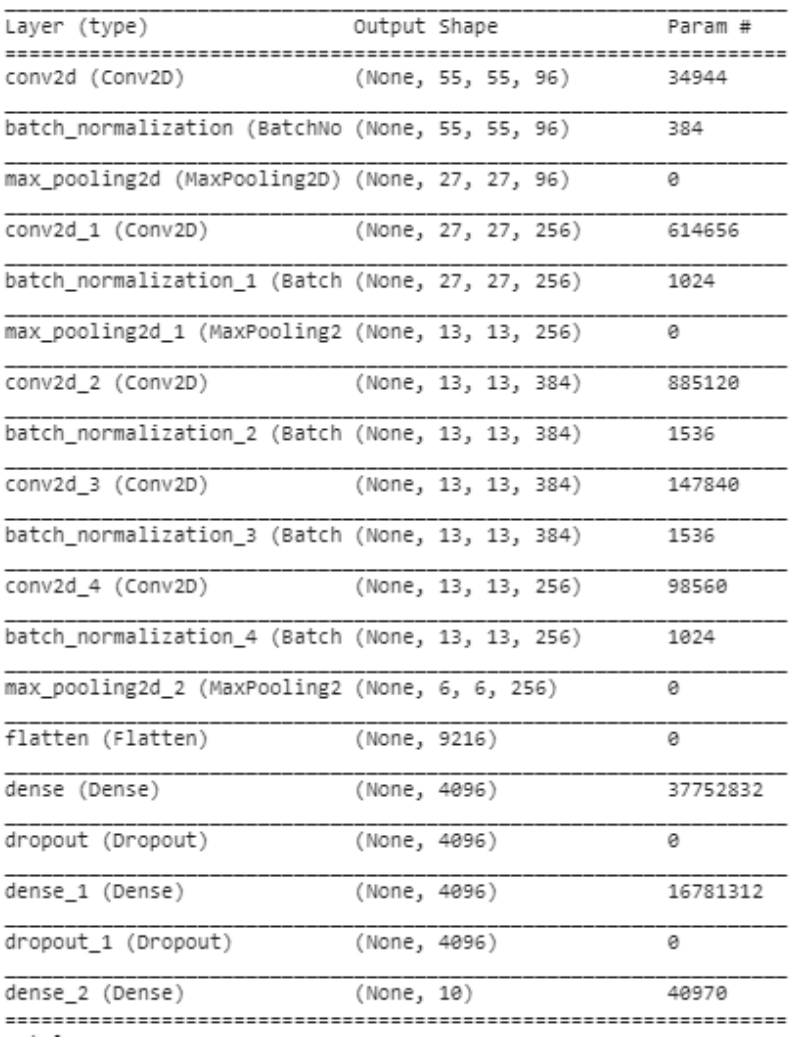
Data citra yang didapatkan dibagi menjadi 3 bagian. Data untuk *training*, data untuk *validation*, dan data untuk *testing*. Data *training* dan *validation* akan digunakan untuk melatih dan membuat model. Sementara untuk data *testing* akan digunakan untuk melihat performa yang diberikan oleh model pada *epoch* tertentu. Secara detail pembagian data dapat dilihat pada Tabel 4.

Tabel 4 Division of Training, Testing, and Validation Data

Kelompok Data	Jumlah
Training	3.506
Validation	1.974
Testing	500
Total	5.980

3.3. Classification

Proses klasifikasi dilakukan menggunakan arsitektur CNN AlexNet dan menggunakan *library* dari Keras Tensorflow. Serta proses *training* dilakukan pada *Google Colab*. Pengaturan lapisan seperti yang bisa dilihat pada *Gambar 18*.



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 55, 55, 96)	34944
batch_normalization (BatchNo	(None, 55, 55, 96)	384
max_pooling2d (MaxPooling2D)	(None, 27, 27, 96)	0
conv2d_1 (Conv2D)	(None, 27, 27, 256)	614656
batch_normalization_1 (Batch	(None, 27, 27, 256)	1024
max_pooling2d_1 (MaxPooling2	(None, 13, 13, 256)	0
conv2d_2 (Conv2D)	(None, 13, 13, 384)	885120
batch_normalization_2 (Batch	(None, 13, 13, 384)	1536
conv2d_3 (Conv2D)	(None, 13, 13, 384)	147840
batch_normalization_3 (Batch	(None, 13, 13, 384)	1536
conv2d_4 (Conv2D)	(None, 13, 13, 256)	98560
batch_normalization_4 (Batch	(None, 13, 13, 256)	1024
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 256)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 4096)	37752832
dropout (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16781312
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 10)	40970

Gambar 18 Pengaturan Layer Pada Libary CNN

Pada *Gambar 18* terlihat ada *layer* bernama *Conv2d* yang merupakan lapisan konvolusi yang berjumlah 5. Kemudian terdapat 3 lapisan *Pooling* setelah lapisan konvolusi pertama, kedua dan terakhir. *Pooling* yang digunakan merupakan jenis *Max Pooling* yang akan menyimpan nilai terbesar demi menjaga fitur yang ada pada citra. Lapisan terkahir adalah lapisan *Fully Connected* yang dimulai dari lapisan dengan nama *flatten* untuk mengubah dimensi data masukan menjadi 1 dimensi. Susunan lapisan tersebut diatur sedemikian rupa menyesuaikan dengan lapisan dari arsitektur CNN AlexNet. Selain lapisan yang disesuaikan dengan arsitektur AlexNet, juga ada lapisan *batch normalization* yang berguna untuk menstabilkan gradien dengan cara mengubah nilai kedalam skala tertentu[18]. Pada seluruh lapisan konvolusional diberlakukan fungsi aktivasi ReLU. Fungsi aktivasi ReLU juga digunakan pada 2 lapisan *fully connected* pertama, dan lapisan *fully connected* yang terakhir menggunakan fungsi *softmax*. *Softmax* akan berfungsi mengubah nilai dari setiap *node* yang sebelumnya telah dihitung menjadi nilai tingkat *probability* yang bisa menunjukan hasil klasifikasi dari citra yang diinputkan.

Parameter *loss function Sparse Categorical Crossentropy* digunakan pada percobaan ini untuk mencari nilai *error* pada proses pelatihan model. SCC diguanakan karena sifat dari klasifikasi yang sedang dibuat memiliki kelas *output* yang banyak atau bisa dikategorikan sebagai *multi-class*. Hal tersebut juga mempengaruhi lapisan akhir dari *fully connected* yang menggunakan *softmax*. Parameter selanjutnya adalah parameter dari *learning rate* yang akan mempengaruhi kecepatan adaptasi dari model. Pada penelitian ini, dilakukan 6 kali percobaan dengan *learning rate* yang berbeda, yaitu 0.1, 0.01, dan 0.001. *Epoch* yang digunakan adalah 10 dan 40. Nilai *epoch* menunjukan banyak iterasi yang dilakukan model dalam melakukan pelatihan. Percobaan juga dilakukan dengan mengkombinasikan *learning rate* dan *epoch*. Percobaan ini dilakukan sekaligus untuk melihat parameter yang optimal yang bisa digunakan untuk membuat model klasifikasi menggunakan arsitektur AlexNet untuk data tanaman cabai yang digunakan. Data pengaturan parameter yang digunakan pada penelitian ini dapat dilihat pada Tabel 5.

Tabel 5 Data Parameter Percobaan

	Learning Rate	Total Data	Batch	Epoch
Percobaan-1	0.1	5.980	32	10
Percobaan-2	0.01	5.980	32	10
Percobaan-3	0.001	5.980	32	10
Percobaan-4	0.1	5.980	32	40
Percobaan-5	0.01	5.980	32	40
Percobaan-6	0.001	5.980	32	40

4. Evaluasi

4.1. Hasil Pengujian

Hasil percobaan terbaik akan dilihat dari tingkat akurasi dan *loss*-nya. Kedua nilai tersebut kemudian akan dianalisis untuk mengetahui *learning rate* mana yang optimal. Hasil yang didapat cukup baik, untuk percobaan yang menggunakan *learning rate* 0,01 dan 0,001 mendapatkan akurasi yang cukup bagus hingga mencapai sekitar angka 80% untuk *epoch* 10 dan 90% untuk *epoch* 40. Keseluruhan hasil dapat dilihat pada Tabel 6.

Tabel 6 Hasil Percobaan

	Acc (%)	Val-Acc (%)	Loss	F1 (%)
Percobaan-1	24	24	-	-
Percobaan-2	89,7	83,1	0,2614	83
Percobaan-3	79,8	82,9	0,5335	82
Percobaan-4	24,5	24,7	-	-
Percobaan-5	99,2	88,5	0.0213	88
Percobaan-6	95,8	84,7	0.1262	84

Ket.

- Acc : Akurasi dari data test
- Val-Acc : Akurasi dari data validation
- Loss : *Loss Function Sparse Categorical Cross-entropy*
- F1 : *F1 score function micro*

4.2. Analisis Hasil Pengujian

Pertama-tama, pada penelitian ini percobaan 5 dinyatakan menjadi model terbaik diikuti dengan percobaan 6, percobaan 2, percobaan 3, percobaan 4 dan terakhir percobaan 1. Berdasarkan data pada Tabel 6, terdapat 2 alasan yang menyebabkan model pada percobaan 5 menjadi model terbaik. Selain model dapat mendapatkan akurasi yang paling tinggi, model tersebut juga memiliki tingkat *loss* yang paling rendah, yaitu mencapai 0.0213. Dengan semakin kecil angka *loss* yang ditampilkan, artinya model tersebut memiliki tingkat *error* yang cukup rendah. Jika dilihat pada Tabel 6, Percobaan 1 dan 4 tidak memiliki *loss, recall,* dan F1. Hal tersebut terjadi dikarenakan model yang dibuat menyimpang dan tidak bisa melakukan proses *learning* sebagai mana mestinya. Ada beberapa hal yang bisa menyebabkan model mengalami penyimpangan, namun pada kasus ini diduga model tersebut menyimpang dikarenakan *learning rate* yang terlalu besar. Bisa diamati dari parameter yang digunakan, karena pada penelitian ini seluruh parameter sama, terkecuali untuk ukuran *learning rate* yang juga digunakan sebagai salah satu bahan penelitian.

Jika dilihat pada setiap kelompok *epoch*, *learning rate* 0.01 menjadi *learning rate* yang paling optimal dibandingkan dengan 2 lainnya. Pada kelompok *epoch* 10, dapat dilihat bahwa model yang dibuat pada percobaan 2 menjadi model terbaik diantara mode percobaan 1 dan 3. Sedangkan untuk *epoch* 40, model yang dibuat pada percobaan 5 juga menjadi model terbaik diantara percobaan 4 dan 6. Percobaan 3 dan 6 dengan *learning rate* 0.001 juga memiliki akurasi yang bagus. Namun dilihat dari waktu yang digunakan, dengan *learning rate* 0.01 model dapat mencari titik optimal lebih cepat dari model dengan *learning rate* 0.001.

Selanjutnya untuk menguji model, dilakukan pengujian *Confusion Matrix*. *Confusion matrix* digunakan untuk memberi nilai kepada model agar bisa diketahui seberapa bagus model tersebut bisa bekerja. F1 *Score* termasuk kedalam salah satu *confusion matrix*. Pada Tabel 6 terlihat untuk percobaan 2, percobaan 3, percobaan 5 dan percobaan 6, nilai F1 memiliki angka yang cukup tinggi. Tingginya nilai F1, artinya kedua *Recall* dan *Precission* memiliki angka yang tinggi juga. Dengan kata lain, model dapat mengenali penyakit daun yang

sebelumnya telah dipelajari. Selanjutnya model juga memiliki tingkat akurasi untuk menebak setiap penyakit dengan tingkat presentase yang cukup tinggi.

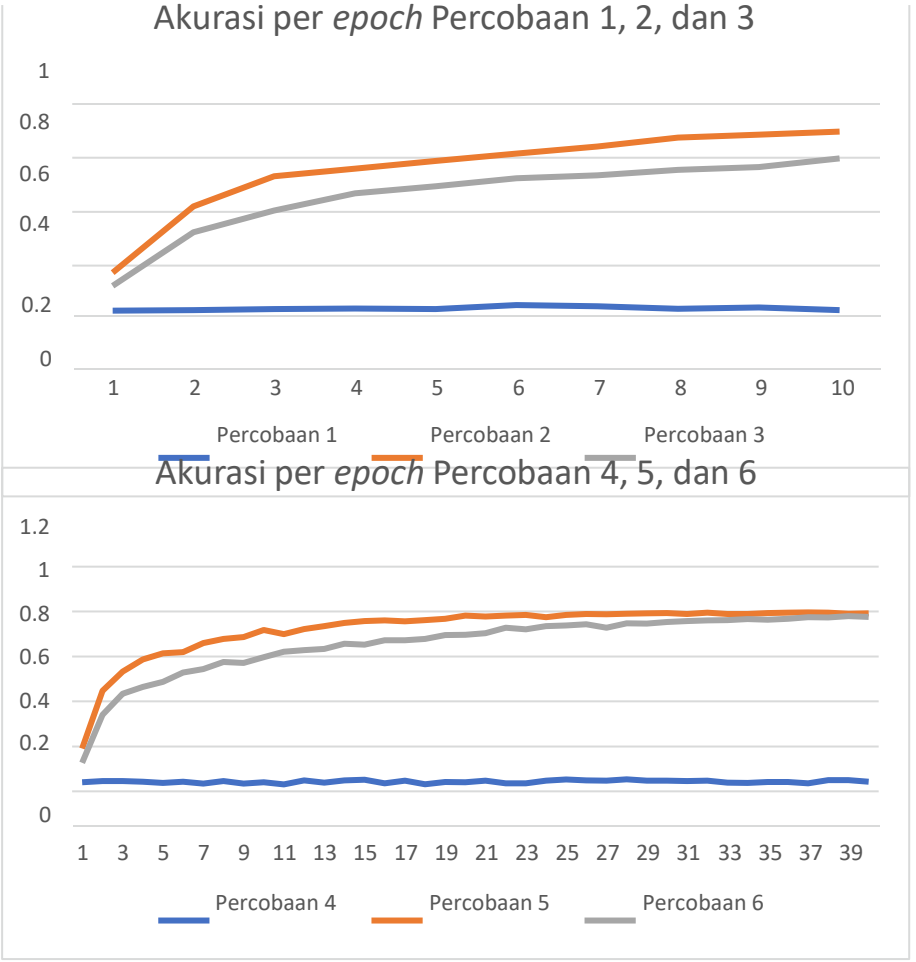
Selengkapnya mengenai *confusion matrix* untuk percobaan 5 dapat dilihat pada Tabel 7. Pada tabel tersebut terlihat bahwa setiap kelas sudah mengklasifikasikan benar sebesar lebih dari 80%. Namun tetap ada kekeliruan dari hasil klasifikasi yang disebabkan oleh beberapa hal. Pertama terapat beberapa citra yang gagal untuk disegmentasi sehingga menghasilkan citra yang tidak jelas. Beberapa dari hasil segmentasi citra ada yang tidak memiliki daun, dengan kata lain daun yang seharusnya menjadi *foreground* terdeteksi sebagai *background*. Selain itu, juga terdapat citra yang salah segmentasi. Proses segmentasi berakhir dengan mendeteksi tangkai daun sebagai *foreground*. Beberapa contoh hasil citra yang gagal disegmentasi dapat dilihat pada Gambar 19. Salah segmentasi ini diduga diakibatkan karena terdapat piksel dari *background* yang mirip dengan piksel pada *foreground* sehingga pada proses segmentasi piksel tersebut dianggap sebagai *background*.

Tabel 7 *Confusion Matrix* Percobaan 5 (*learning rate* : 0.01, *epoch* : 40)

Sebenarnya × Prediksi	Keriting Mosaik	Layu	Normal	Virus Kuning	Total
Keriting Mosaik	99	14	9	0	122
Layu	5	129	0	1	135
Normal	13	1	93	0	107
Virus Kuning	11	2	0	123	136
Total	128	146	102	124	500

Gambar 19 Perbandingan citra asli dan citra yang gagal disegmentasi

Pada Gambar 20 terlihat grafik akurasi setiap *epoch* untuk setiap percobaan. Dari grafik ini, dapat dilihat kecepatan model dalam beradaptasi dalam mencari solusi yang optimal. Dalam grafik ini juga, *learning rate* yang optimal akan terlihat. Gambar grafik dikelompokan menjadi 2 sesuai dengan jumlah *epoch* yang diberikan. Dalam grafik tersebut, terlihat akurasi untuk percobaan 1 dan 4 tidak memiliki kenaikan. Kedua percobaan tersebut dinyatakan terjebak dalam solusi lokal dari awal pembelajaran dan tidak bisa menemukan jalan untuk mencapai solusi yang lebih optimal. Sementara untuk percobaan 2, percobaan 3, percobaan 5, dan 6 memiliki kenaikan yang signifikan. Ke empat model tersebut terus naik hingga mencapai *epoch* terakhir. Namun jika diperhatikan pada awal *epoch* untuk setiap gambar grafik, model dengan *learning rate* 0,01 yaitu model percobaan 2 dan 5, selalu memiliki kenaikan lebih tinggi dari pada model lainnya. Dengan kata lain model dari percobaan 2 dan 5 memiliki kecepatan adaptasi yang tinggi pada awal *epoch*. Berdasarkan hal tersebut, dalam penelitian ini dirasa *learning rate* 0,01 menjadi *learning rate* paling optimal dalam kasus klasifikasi penyakit pada tanaman cabai ini.



Gambar 20 Grafik akurasi per *epoch* seluruh percobaan

5. Kesimpulan

Metode *Deep Learning* CNN (*Convolutional Neural Network*) dapat memberikan akurasi yang cukup tinggi dalam hal klasifikasi. CNN dapat menemukan fitur yang ada pada citra yang diberikan secara otomatis. Penggunaakn arsitektur AlexNet pada penelitian kali ini juga menghasilkan akurasi yang bagus. Akurasi yang diberikan CNN juga bisa ditingkatkan kembali dengan menggunakan teknik *data augmentation* yang berfungsi untuk menambahkan jumlah data dan juga dengan cara mencari *learning rate* yang paling optimal agar proses pembelajaran menjadi lebih cepat dalam beradaptasi. Dengan pengaturan *learning rate* yang sesuai, maka model akan lebih cepat memahami citra dan memberikan akurasi yang cukup baik.

Referensi

[1] A. Hafiz, R. Zahari, and T. H. Lim, "Detecting Diseases in Chilli Plants Using K-Means Segmented Support Vector Machine" in *3rd International Conference on Imaging, Signal Processing and Communication*, 2019.

[2] Y. Kawasaki, H. Uga, S. Kagiwada, and H. Iyatomi, "Basic Study of Automated Diagnosis of Viral Plant Diseases Using Convolutional Neural Networks" in *Springer International Publishing Switzerland 2015.*, DOI: 10.1007/978-3-319-27863-6_59.

[3] Zulkifli, A. Hallis, A. Yeon, and Rohani, "Feasibility Study on Plant Chili Disease Detection Using Image Processing Techniques" in *Third Conference on Intelligent System Modelling and Simulation*, 2012.

[4] J. W. Orillo, J. D. Cruz, L. Agapito, P. J. Satimbre, I. Valenzuela, "Identification of Diseases in Rice Plant (Oryza Sativa) using Back Propagation Artificial Neural Network" in *7th IEEEInternational Conference Humanoid, Nanotechnology, Information Technology Communication and Control, Environment and Management (HNICEM)*, 2013.

[5] D. Al Bashish, M. Braik, and S. Bani-Ahmad, "A Framework for Detection and Classification of Plant Leaf and Stem Diseases" in *International Conference on Signal and Image Processing*, 2010.

[6] S. Kumar, and B. K. Raghavendra, "Diseases Detection of Various Plant Leaf Using Image Processing Techniques: A Review"

[7] S. Mirchandani, M. Pendse, P. Rane, and A Vedula, "Plant Disease Detection and Classification Using Image Processing and Artificial Neural Network" in *International Research Journal of Engineering and Technology (IRJET)*.
[8] A. S. Duriat, N. Gunaeni, and A. W. Wulandari, "Penyakit Penting Tanaman Cabai dan Pengendaliannya" in *Pusat Penelitian dan Pengembangan Hortikultura Badan Penelitian dan Pengembangan Pertanian*, 2007.
[9] N. Sofia, "Convolutional Neural Network", Medium, 9 June 2018. [Online]. Available: https://medium.com/@nadhifasofia/1-convolutional-neural-network-convolutional-neural-network-merupakan-salah-satu-metode-machine-28189e17335b. [Accessed 31 March 2020].
[10] Wikipedia Ensiklopedia Bebas, "K-Means Clustering", Wikimedia Commons, 29 March 2020. [Online]. Available: https://en.wikipedia.org/wiki/K-means_clustering. [Accessed 1 April 2020]
[11] H. Q. Cap, K. Suwa, E. Fujita, S. Kagiwada, H. Uga, and H. Iyatomi, "A Deep Learning Approach for on-site Plant Leaf Detection" in 14th International Colloquium in Signal Processing & its Applications (CSPA), 2018.
[12] R. S. Zimmermann, and J. N. Siems, "Faster training of Mask R-CNN by focusing on instance boundaries" in Computer Vision and Image Understanding 188, 2019.
[13] M. Sardogan, A. Tuncer, and Y. Ozen, "Plant Leaf Disease Detection and Classification Based on CNN with LVQ Algorithm" in 3rd International Conference on Computer Science and Engineering, 2018.
[14] H. Yalcin, S. Razavi, "Plant Classification using Convolutional Neural Network" in 5th International Conference on Agro-Geoinformatics, 2016.
[15] S. H. Lee, C. S. Chan, P. Wilkin, and P. Remagnino, "Deep-Plant: Plant Identification with Convolutional Neural Networks" in IEEE International Conference on Image Processing (ICIP), 2015.
[16] P. Pawara, E. Okafor, L. Schomaker, and M. Wiering, "Data Augmentation for Plant Classification" in Springer International Publishing AG, 2017.
[17] C. Rother, V. Kolmogorov and A. Blake, "'GrabCut': interactive foreground extraction using iterated graph cuts", *Proc. ACM SIGGRAPH*, vol. 23, no. 3, pp. 309-314, Aug. 2004.
[18] A. Verma, "AlexNet", Medium, 28 July 2020. [Online]. Available: https://towardsdatascience.com/alexnet-8b05c5eb88d4. [Accessed 15 February 2021].

Lampiran