Klasifikasi Citra Penyakit Daun Cabai Rawit Dengan Menggunakan CNN Arsitektur AlexNet dan SqueezeNet

[1, 2, 3]Program Studi Informatika, Fakultas Ilmu Komputer, Universitas Pembangunan Nasional "Veteran" Jawa Timur
Jl. Raya Rungkut Madya, Gunung Anyar, Surabaya
Telp. (031) 8706369, Fax. (031) 8706372
Email: [1*]fahmiad48@gmail.com, [2*]fettyanggraeny.if@upnjatim.ac.id,
[3*]hendra.maulana.if@upnjatim.ac.id

Abstrak. Cabai rawit merupakan salah satu sektor komoditas holtikultura yang ada di Indonesia. Salah satu penyebab penurunan produksi cabai rawit adalah adanya penyakit yang terdapat pada bagian tanaman cabai rawit, salah satu bagian yang terserang adalah bagian daun. Klasifikasi jenis penyakit pada daun cabai rawit dalam skala besar diperlukan teknologi yang dapat membantu mengatasi hal tersebut dengan mengunakan teknologi pengolahan citra digital yang cepat dan akurat dengan menggunakan CNN serta penggunaan arsitektur AlexNet dan SqueezeNet sebagai salah satu contoh metode dalam teknologi tersebut. Tingkat akurasi yang didapat dari arsitektur SqueezeNet dan AlexNet memiliki hasil akurasi yang berbeda dimana SqueezeNet memiliki nilai 85% dan AlexNet memiliki nilai 90% yang dimana AlexNet memiliki nilai yang sangat baik.

Kata kunci: *Penyakit daun cabai rawit, Pengolahan Citra Digital, CNN, AlexNet, SqueezeNet.*

1 Pendahuluan

Cabai rawit atau disebut juga sebagai *Capsicum frutescens L.* merupakan salah satu sektor komoditas holtikultura yang ada di Indonesia[1]. Salah satu penyebab penurunan produksi cabai rawit adalah adanya penyakit yang terdapat pada bagian daun tanaman cabai rawit. Untuk melakukan klasifikasi jenis penyakit pada daun cabai rawit dalam skala besar diperlukan teknologi yang dapat membantu mengatasi hal tersebut dengan melakukan klasifikasi jenis penyakit pada daun cabai rawit dengan menggunakan citra secara cepat dan akurat. Teknologi tersebut merupakan jenis *Machine Learning* dengan algoritma *deep learning* yang mampu merancang data citra dua dimensi yaitu metode *Convolutional Neural Network* (CNN)[2]. Metode ini selain mampu merancang juga dapat mengklasifikasikan jenis citra.

Salah satu jenis *deep learning* yaitu *Convolutional Neural Network* (CNN), yang dikembangkan dari *Multilayer Percepton* (MLP) dan kemudian dirancang untuk memproses data dalam dua dimensi, seperti gambar atau suara[3]. CNN termasuk dalam jenis *Deep Neural Network* karena tingkat jaringan yang dalam dan banyak diimplementasikan ke dalam data citra. Definisi dari konvolusi yaitu istilah

Diterima 12 Mei 2023 Direvisi 28 Mei 2023 Diterima untuk publikasi 30 Mei 2023

matematis yang bermakna mengaplikasikan suatu fungsi pada *output* fungsi lain secara berulang dan menghasilkan *feature map*[4]. Fungsi dari konvolusi adalah untuk mengekstrak fitur citra[5]. Metode CNN terus mengalami perkembangan dari awal mulanya pada tahun 1998 dengan menggunakan arsitektur LeNet [6], kemudian disusul dengan keberhasilan dari arsitektur AlexNet pada tahun 2012[7], ResNet pada tahun 2015[8], GoogleNet[9], SqueezeNet[10], dan lain-lain di mana di masa yang akan mendatang terdapat jenis arsitektur yang lebih baik lagi.

Hasil dari penelitian ini yang diharapkan adalah untuk memberikan gambaran klasifikasi citra secara digital pada penyakit daun cabai dengan menggunakan *Convolutional Neural Network* sehingga dapat bermanfaat bagi penulis dan juga bermanfaat bagi sektor pertanian khususnya pada komoditas holtikultura.

2 Metode Penelitian

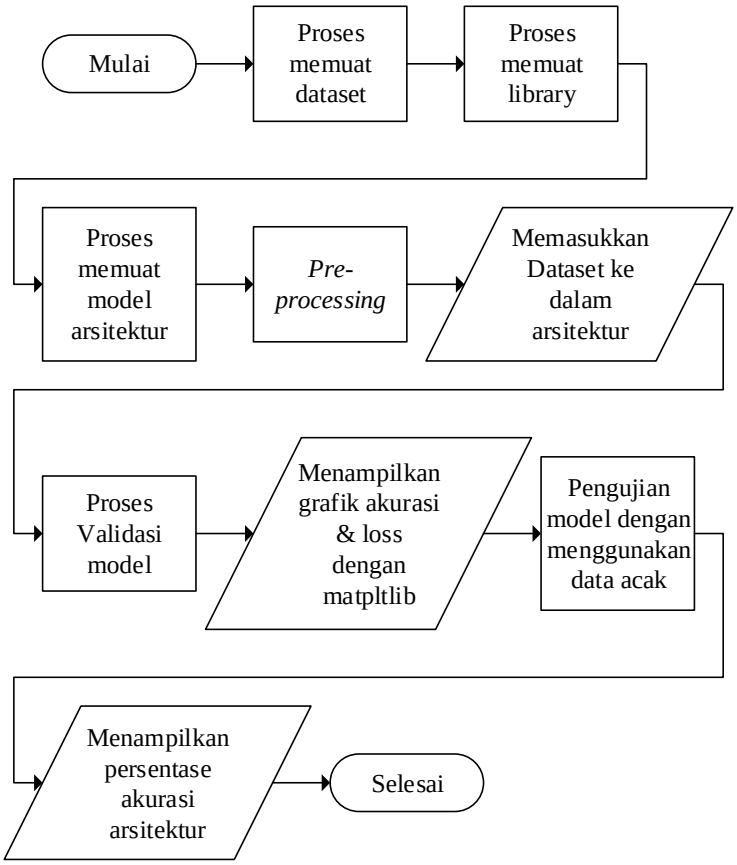
Pada tahapan ini menggunakan *dataset* citra dari setiap penyakit pada daun cabai rawit yang berjumlah 1000 data yang didapat dari internet.

Dataset tersebut akan diproses melalui tahap pelatihan dan pengujian. Tahapan awal untuk menyelesaikan penelitian yang dilakukan dengan mengumpulkan data yang diperlukan yaitu dengan melakukan tahap pelatihan, lalu tahap berikutnya dilanjut dengan memasukkan data latih, pra-proses dengan menambahkan label pada setiap data dan merubah ukuran citra, perancangan CNN, dan bobot optimal. Tahap selanjutnya yaitu tahap pengujian dengan proses memasukkan data uji, pra-proses, pengujian CNN, yang kemudian hasil klasifikasi CNN dengan menggunakan bobot optimal yang didapat dari hasil tahap pelatihan.

2.1 Pra-proses

Pra-proses merupakan langkah pertama sebelum melakukan proses klasifikasi. Pada tahap ini dilakukan perubahan ukuran citra dengan ukuran yang simetris serta dilakukan proses pelabelan data. Semakin tinggi ukuran citra, maka semakin berat dalam melakukan proses, namun data yang didapatkan pada gambar semakin besar dan sangat berpengaruh terhadap akurasi yang didapat. Untuk penelitian ini menggunakan gambar dengan ukuran 128 x 128 piksel agar dapat mengetahui apakah hasil dari gambar tersebut dapat diketahui atau tidak.

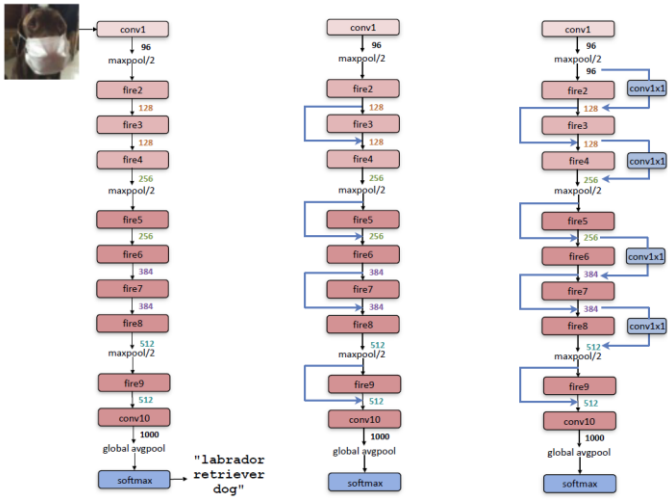
2.2 Perancangan Alur CNN



Gambar 1. Alur Perancangan CNN

Gambar 1. menjelaskan bagaimana proses tahapan dari CNN, . Langkah pertama dari tahap tersebut adalah memuat *dataset* lalu memuat *library* yang akan digunakan, kemudian berlanjut pada proses memuat model. Untuk model yang digunakan yaitu SqueezeNet dan AlexNet dengan 512 neuron dan 5 kelas. Sebelum melakukan proses *training*, diperlukan *pre-processing* atau tahap pra-proses yang kemudian dilanjutkan melakukan proses *training* model CNN setelah itu melakukan proses validasi model, kemudian menampilkan grafik akurasi & grafik *loss* dengan menggunakan matplotlib, proses selanjutnya yaitu melakukan proses pengujian model menggunakan data acak, lalu tahap akhir pada rancangan ini yaitu menampilkan persentase akurasi arsitektur dan selesai.

2.3 SqueezeNet

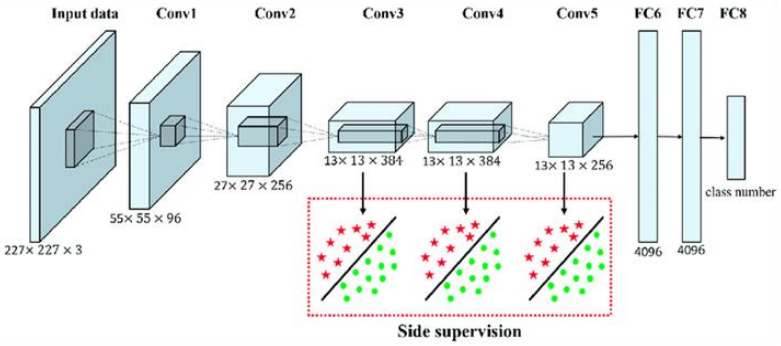


Gambar 2. Arsitektur SqueezeNet

Gambar 2. merupakan struktur arsitektur dari Squeezenet yang merupakan salah satu jenis arsitektur jaringan syaraf tiruan yang menggunakan CNN. SqueezeNet sanggup menggapai akurasi AlexNet (pemenang ImageNet classification task 2012) dengan parameter 50 kali lebih sedikit serta waktu pelatihan 2 kali lebih cepat[11].

SqueezeNet banyak mengubah susunan konvolusi 3x3 dengan 1x1 serta filter yang lebih sedikit untuk mengecilkan ukuran *activation map* (*squeeze*). Metode reduksi dimensi ini pertama kali dikenalkan dalam model *Network In Network* (NIN)[12].

2.4 AlexNet



Gambar 3. Arsitektur AlexNet

Pada Gambar 3. merupakan struktur arsitektur dari AlexNet. AlexNet adalah jaringan saraf kompleks dengan 60 juta parameter dan 650.000 neuron. AlexNet meningkatkan kapasitas pembelajaran dengan meningkatkan kedalaman jaringan dan menerapkan strategi pengoptimalan multi - parameter[13]. Secara khusus, untuk mengatasi masalah bahwa fungsi aktivasi tradisional (termasuk fungsi logistik, tanh, dan arctan) sering terjebak dalam penghilangan gradien jaringan dalam, Krizhevsky menggunakan *Rectified Linear Unit* (ReLU) sebagai fungsi aktivasi baru.

2.5 Skenario Uji Coba

Skenario uji coba dilakukan agar mendapatkan hasil yang sesuai dengan target yang akan dicapai, skenario uji coba yang dilakukan pada penelitian ini adalah :

1. Melakukan pengujian data latih, data validasi, dan data uji berupa citra penyakit daun keriting, penyakit daun kuning, dan daun yang sehat pada daun cabai rawit yang berjumlah 722 gambar pada data latih, 128 gambar pada data validasi, dan 150 gambar pada data uji.
2. Memberikan parameter dengan jumlah objek kelas sebanyak 5, ukuran dimensi citra latih sebesar 128x128 piksel dan jumlah *convolution layer* sebanyak 36. dan variabel uji coba pada CNN yang dijelaskan pada Tabel 1. untuk mendapatkan hasil yang optimal.
3. Memberikan Arsitektur SqueezeNet dan AlexNet pada CNN dengan variabel uji coba yang terdiri dari *epoch* sebesar 32, *batch size* sebesar 16, *learning rate* & *weight decay* sebesar 0.0001. Untuk persentase data latih dan validasi sebesar 85% serta data uji sebesar 15%. Untuk optimizer menggunakan ADAM serta fungsi aktivasi menggunakan Softmax
4. Mendapatkan tingkat akurasi dari hasil data latih dan validasi. Lalu menampilkan hasil data uji.

3 Hasil dan Pembahasan

3.1 Implementasi SqueezeNet

Alur kerja SqueezeNet adalah menggunakan satu *convolution layer* dengan kernel 7x7, *stride* 2, dan aktivasi ReLU dan dikombinasikan dengan *max pooling* yang lalu diikuti 8 *Fire Module* yang hanya memiliki stride 1x1. Pada tahapan akhir model ini yaitu saat klasifikasi hanya menggunakan 1 layer yang menggunakan *convolution layer* dengan aktivasi ReLU dan ditutup dengan Average Pooling (AdaptiveAvgPool2d). Untuk hasil dari model arsitektur SqueezeNet dijelaskan pada Tabel 1.

Tabel 1. Arsitektur CNN SqueezeNet

Input	Channel In Out	Kernel Size	Stride	Padding	Dilation	Aktivasi
Conv2d	3 96	7, 7	2, 2	-	-	ReLU
Max Pooling	- -	3	2	0	1	-
			Fire1			
Conv2d	96 16	1, 1	1, 1	-	-	ReLU
Conv2d	16 64	1, 1	1, 1	-	-	ReLU
Conv2d	16 64	3, 1	1, 1	1, 1	-	ReLU
			Fire2			
Conv2d	128 16	1, 1	1, 1	-	-	ReLU
Conv2d	16 64	1, 1	1, 1	-	-	ReLU
Conv2d	16 64	3, 1	1, 1	1, 1	-	ReLU
			Fire3			
Conv2d	128 32	1, 1	1, 1	-	-	ReLU
Conv2d	32 128	1, 1	1, 1	-	-	ReLU
Conv2d	32 128	3, 1	1, 1	1, 1	-	ReLU
Max Pooling	- -	3	2	0	1	-
			Fire4			
Conv2d	256 32	1, 1	1, 1	-	-	ReLU
Conv2d	32 128	1, 1	1, 1	-	-	ReLU
Conv2d	32 128	3, 1	1, 1	1, 1	-	ReLU
			Fire5			
Conv2d	256 48	1, 1	1, 1	-	-	ReLU
Conv2d	48 192	1, 1	1, 1	-	-	ReLU
Conv2d	48 192	3, 1	1, 1	1, 1	-	ReLU
			Fire6			
Conv2d	384 48	1, 1	1, 1	-	-	ReLU
Conv2d	48 192	1, 1	1, 1	-	-	ReLU

Conv2d	48 192	3, 1	1, 1	1, 1	-	ReLU
			Fire7			
Conv2d	384 64	1, 1	1, 1	-	-	ReLU
Conv2d	64 256	1, 1	1, 1	-	-	ReLU
Conv2d	64 256	3, 1	1, 1	1, 1	-	ReLU
Max Pooling	- -	3	2	0	1	-
			Fire8			
Conv2d	384 64	1, 1	1, 1	-	-	ReLU
Conv2d	64 256	1, 1	1, 1	-	-	ReLU
Conv2d	64 256	3, 1	1, 1	1, 1	-	ReLU
			Classifier			
Conv2d	512 1000	1, 1	1, 1	-	-	ReLU
		Adaptive Average Pool2d (1,1) *Trainable Parameter* = 134.7M				

3.2 Implementasi AlexNet

Alur kerja pada AlexNet adalah ekstrasi fitur dengan menggunakan *Conv2d* yang kemudian diaktivasi melalui ReLU dan dikombinasikan dengan *max pooling*. Pada bagian *classifier* bertujuan untuk mengklasifikasi dengan menggunakan *fully connected* dengan aktivasi ReLU. Setelah melewati lapisan ini, maka akan memasuki proses klasifikasi dengan menggunakan aktivasi *softmax*. Proses keseluruhan dengan menggunakan arsitektur ini menggunakan lebih dari 61 juta parameter yang digunakan untuk proses pelatihan. Untuk hasil dari model arsitektur AlexNet dijelaskan pada Tabel 2.

Tabel 2. Arsitektur CNN AlexNet

Input	Channel In Out	Kernel Size	Stride	Padding	Dilation	Aktivasi
Conv2d	3 64	11, 11	4, 4	2, 2	-	ReLU
Max Pooling	- -	3	2	0	1	-
Conv2d	64 192	5, 5	1, 1	2, 2	-	ReLU
Max Pooling	- -	3	2	0	1	-
Conv2d	192 384	3, 3	1, 1	1, 1	1, 1	ReLU

Conv2d	384 256	3, 3	1, 1	1, 1	1, 1	ReLU
Conv2d	256 256	3, 3	1, 1	1, 1	1, 1	ReLU
Max Pooling	- -	3	2	0	1	-
			Classifier			
FC	9216 4096		Dropout p = 0.5			ReLU
FC	9216 4096		Dropout p = 0.5			ReLU
Linear	4096 1000		-			ReLU
Linear	512 3		-			-
			Trainable Parameter = 61M			

3.3 Pelatihan Model

Untuk tahap pelatihan model akan menunjukkan jumlah *epoch* yang digunakan. Hasil dari pelatihan tersebut berupa grafik akurasi (*train* dan *val*) dan grafik *loss* (*train* dan *val*) dan waktu yang dibutuhkan pada setiap pemrosesan *epoch*. Untuk setiap model arsitektur memiliki waktu eksekusi *epoch* dengan durasi yang berbeda meskipun jumlah *epoch* dan *batch size* yang sama yaitu dengan jumlah *epoch* sebanyak 32 dan jumlah *batch size* sebanyak 16.

```
[12] time_str = float(sum(times))
     print('total waktu {:.3f} s'.format(time_str))
     # print(time_str)

     total waktu 1489.296 s
```

Gambar 4. Waktu tempuh iterasi arsitektur SqueezeNet

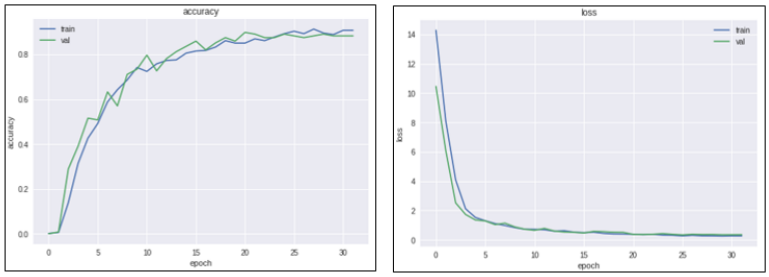
Untuk model SqueezeNet seperti pada Gambar 4. membutuhkan waktu dengan durasi rata – rata 24 menit yang ditunjukkan pada gambar 4. dengan jumlah *epoch* sebanyak 32 dan jumlah *batch size* sebanyak 16.

```
[ ] time_str = float(sum(times))
    print('total waktu {:.3f} s'.format(time_str))
    # print(time_str)

    total waktu 2283.243 s
```

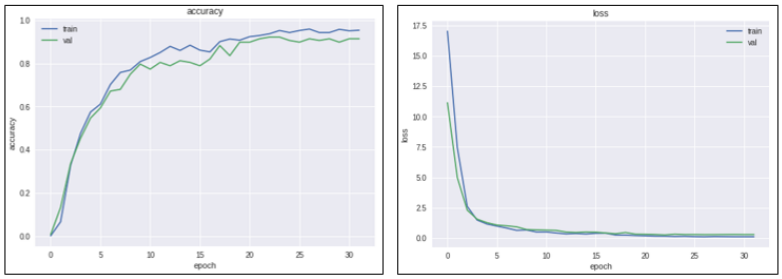
Gambar 5. Waktu tempuh iterasi arsitektur AlexNet

Untuk model AlexNet seperti pada Gambar 5.membutuhkan waktu dengan durasi rata – rata 38 menit yang ditunjukkan pada gambar 5. dengan jumlah *epoch* sebanyak 32 dan jumlah *batch size* sebanyak 16.



Gambar 6. Grafik akurasi dan *loss* pada arsitektur SqueezeNet

Pada Gambar 6. dijelaskan bahwa hasil grafik dari pelatihan model dengan menggunakan arsitektur SqueezeNet dan menggunakan *learning rate* sebesar 0.0001 yang terdiri dari grafik akurasi dan grafik *loss* menunjukkan hasil yang bagus dan stabil untuk sistem pembelajarannya.



Gambar 7. Grafik akurasi dan *loss* pada arsitektur AlexNet

Pada Gambar 7. dijelaskan bahwa hasil grafik dari pelatihan model dengan menggunakan model arsitektur AlexNet dan menggunakan *learning rate* sebesar 0.0001 yang terdiri dari grafik akurasi dan grafik *loss* menunjukkan hasil yang bagus dan stabil untuk sistem pembelajarannya.

Tahapan selanjutnya yang dilakukan setelah dilakukan tahapan pelatihan model yaitu tahapan pengujian model. Pada proses tahap ini, model menghasilkan nilai akurasi. Untuk mendapatkan nilai akurasi, maka nilai data yang benar dibagi dengan nilai total data uji lalu dikali 100%. Maka nilai akurasi dapat ditampilkan dari sebuah model CNN yang dilakukan dalam tahap pengujian.

3.4 Evaluasi Model

Pada penelitian ini, hasil implementasi dari arsitektur SqueezeNet dan AlexNet yang kemudian dapat mengklasifikasikan objek. Hasil dari klasifikasi tersebut berupa citra yang berukuran 128 x 128 piksel yang kemudian diberi label. Untuk hasil yang benar akan diberi label dengan tulisan berwana hijau dan untuk hasil yang salah akan diberi label dengan tulisan berwarna merah. Hasil tersebut dapat dilihat pada Gambar 8.

Gambar 8. Hasil klasifikasi

Untuk hasil akurasi pada model arsitektur SqueezeNet yang sesuai pada Gambar 9. tersebut berhasil mengklasifikasikan 127 gambar dari 150 gambar yang diujikan dan gagal mengklasifikasikan pada 23 gambar sehingga menghasilkan akurasi sebesar 85%. Sedangkan untuk hasil akurasi pada model arsitektur AlexNet yang sesuai pada Gambar 10. tersebut berhasil mengklasifikasikan 135 gambar dari 150 gambar yang diujikan dan gagal mengklasifikasikan pada 15 gambar sehingga menghasilkan akurasi sebesar 90%.

	precision	recall	f1-score	support
0	0.74	0.83	0.78	30
1	0.79	0.90	0.84	30
2	0.92	0.73	0.81	30
3	0.85	0.93	0.89	30
4	1.00	0.83	0.91	30
accuracy			0.85	150
macro avg	0.86	0.85	0.85	150
weighted avg	0.86	0.85	0.85	150

Gambar 9. Hasil *performance metrics* SqueezeNet

	precision	recall	f1-score	support
0	0.81	0.83	0.82	30
1	0.84	0.90	0.87	30
2	0.97	0.93	0.95	30
3	0.90	0.93	0.92	30
4	1.00	0.90	0.95	30
accuracy			0.90	150
macro avg	0.90	0.90	0.90	150
weighted avg	0.90	0.90	0.90	150

Gambar 10. Hasil *performance metrics* AlexNet

4 Kesimpulan

Kesimpulan yang berdasarkan dari percobaan dalam tahap pelatihan dan pengujian pada penelitian ini menggunakan *dataset* sejumlah 1000 gambar yang diambil dari internet yang kemudian dapat disimpulkan bahwa *Dataset* yang dilatih dan diuji dengan menggunakan arsitektur AlexNet dengan nilai *epoch* sebanyak 32, *batch size* sebanyak 16, *learning rate* & *weight decay* dengan nilai 0.0001 memiliki nilai akurasi sebesar 90% yang dimana nilai akurasi tersebut lebih baik daripada arsitektur SqueezeNet yang memiliki nilai akurasi sebesar 85% dengan nilai parameter dan variabel yang sama.

5 Referensi

[1] Mukarlina, S. K., & Rianti, R. (2010). Uji Antagonis Trichoderma harzianum Terhadap Fusarium spp. Penyebab Penyakit Layu pada Tanaman Cabai (Capsicum annum) Secara In Vitro. Jurnal Fitomedika.

[2] Ridho Aji Pangestu, Basuki Rahmat, & Fetty Tri Anggraeny. (2020). Implementasi Algoritma CNN Untuk Klasifikasi Citra Lahan Dan Perhitungan Luas. Jurnal Informatika Dan Sistem Informasi (JIFoSI) , 1(1), 166–174.

[3] Ilahiyah, S., & Nilogiri, A. (2000). Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network. 49–56.

[4] Irfansyah, D., Mustikasari, M., & Suroso, A. (2021). Arsitektur Convolutional Neural Network (CNN) Alexnet Untuk Klasifikasi Hama Pada Citra Daun Tanaman Kopi. Jurnal Informatika: Jurnal Pengembangan IT (JPIT), 6(2), 87–92. http://ejournal.poltektegal.ac.id/index.php/informatika/article/view/2802

[5] Putra, E., & Suartika, W. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101. Jurnal Teknik ITS, 5(1). https://doi.org/10.12962/j23373539.v5i1.15696

[6] Setiawan, W. (2020). Perbandingan Arsitektur Convolutional Neural Network Untuk Klasifikasi Fundus. Jurnal Simantec, 7(2), 48–53. https://doi.org/10.21107/simantec.v7i2.6551

[7] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. 1–9.

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition Kaiming. Indian Journal of Chemistry - Section B Organic and Medicinal Chemistry, 45(8), 1951–1954. https://doi.org/10.1002/chin.200650130

[9] Szegedy, C., Vanhoucke, V., & Ioffe, S. (2015). Rethinking the Inception Architecture for Computer Vision Christian.

[10] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. 1–13. http://arxiv.org/abs/1602.07360

[11] Wu, Z., Shen, C., & van den Hengel, A. (2019). Wider or Deeper: Revisiting the ResNet Model for Visual Recognition. Pattern Recognition, 90, 119–133. https://doi.org/10.1016/j.patcog.2019.01.006

[12] Amin, A., Sari, Y. A., & Adinugroho, S. (2019). Klon Perilaku Menggunakan Jaringan Saraf Tiruan Konvolusional Dalam Game SuperTuxKart. Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya, 3(1), 866–875. https://www.researchgate.net/profile/Yuita-Arum-Sari/publication/327822325_Klon_Perilaku_Menggunakan_Jaringan_Saraf_Tiruan_Konvolusional_Dalam_Game_SuperTuxKart/links/5ba6622aa6fdccd3cb6c50c1/Klon-Perilaku-Menggunakan-Jaringan-Saraf-Tiruan-Konvolusional-D

[13] Khan, A., Jamil, M., Naz, R., Humayun, A., Ullah, S., & Jelani, G. (2020). Investigation of Treatment Regimen of the Genital Warts Using Various Chemotherapeutic Agents. Biomedical Sciences, 6(1), 1. https://doi.org/10.11648/j.bs.20200601.11