MENDETEKSI HAMA PADA SAWI HIJAU MENGGUNAKAN ALGORITMA CNN

Alamsyah Ahmad H.Hadiningrat  
Departemen Teknik Elektro, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Gunung Djati Bandung  
Bandung, Indonesia  
1207070012@student.uinsgd.ac.id

Rajasabian Ardhi Awangga  
Departemen Teknik Elektro, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Gunung Djati Bandung  
Bandung, Indonesia  
1207070096 @student.uinsgd.ac.id

Ahmad Pahrurrozi  
Departemen Teknik Elektro, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Gunung Djati Bandung  
Bandung, Indonesia  
1207070008@student.uinsgd.ac.id

*Abstract*— Penelitian ini bertujuan untuk mengembangkan metode deteksi hama pada tanaman sawi hijau menggunakan algoritma *Convolutional Neural Network* (CNN). Hama pada tanaman merupakan masalah serius dalam pertanian yang dapat menyebabkan kerugian besar. Dalam penelitian ini, dilakukan pemrosesan citra digital tanaman sawi hijau yang terinfeksi hama menggunakan teknik CNN. Pada penelitian ini dilakukan pengujian terhadap empat model yang sudah dibuat yang mana dua model dengan akurasi tertinggi diambil untuk di implementasikan pada program. Dua model tertinggi yaitu model D dengan tingkat akurasi 95.% dan model C dengan tingkat akurasi 84%. Penelitian ini diharapkan dapat memberikan kontribusi dalam pengembangan metode deteksi hama yang lebih efisien dan akurat pada tanaman sawi hijau. Penggunaan algoritma CNN dalam deteksi hama dapat memberikan solusi yang lebih cepat dan otomatis, sehingga memungkinkan petani untuk mengambil tindakan pencegahan yang tepat dalam menjaga kesehatan tanaman sawi hijau.

Keywords—Convolutional neural network, deep learning, klasifikasi

# **PENDAHULUAN**

Sawi Hijau (*Brassica rapavar. parachinensis*) yang juga dikenal dengan nama Caisin (Nazaruddin, 1999) termasuk sayuran daun dari keluarga *cruciferae* yang mempunyai nilai ekonomis tinggi. Tanaman sawi hijau berasal dari Tiongkok (Cina) dan Asia Timur. Daun sawi hijau merupakan bahan makanan sayuran yang mengandung zat gizi yang cukup lengkap sehingga mengkonsumsi daun sawi hijau dapat mempertahankan kesehatan tubuh. Menurut Direktorat Gizi Departemen Kesehatan RI (1981) bahwa sawi hijau mengandung protein, lemak, karbohidrat, Ca, P, Fe, Vitamin A, Vitamin B, dan Vitamin C. Selain itu, mengkonsumsi tanaman sawi hijau dipercaya dapat menghilangkan rasa gatal pada tenggorokan karena batuk, menghilangkan sakit kepala, dan dapat membersihkan darah (Haryanto et al., 2003).

Namun demikian, pertumbuhan sawi hijau tentunya memiliki kendala dalam membudidayakannya yaitu adanya serangan hama yang dapat mengganggu pertumbuhan dan produktifitas tanaman sawi hijau. Hama yang biasanya menyerang tanaman sawi hijau diantaranya adalah ulat Tritip/ngegat punggung berlian (*Plutella xylostella L*.), ulat Krop Kubis (*Crocidolomia binotalis Zeller, Lepidoptera: Pyralidae*) dan kumbang daun atau kumbang anjing (*Phyllotretasp, Coleoptera: Chrysomelidae*) (*Kalshoven,* 1981; *Pracaya,* 1999). Beberapa jenis hama lainnya menurut Wahyudi (2004) seperti ordo Coleoptera, Hemiptera (kepik-kepikan), Lepidoptera (pengorok daun), Nematoda, dan Orthoptera (belalang) yang merupakan hama sekunder.

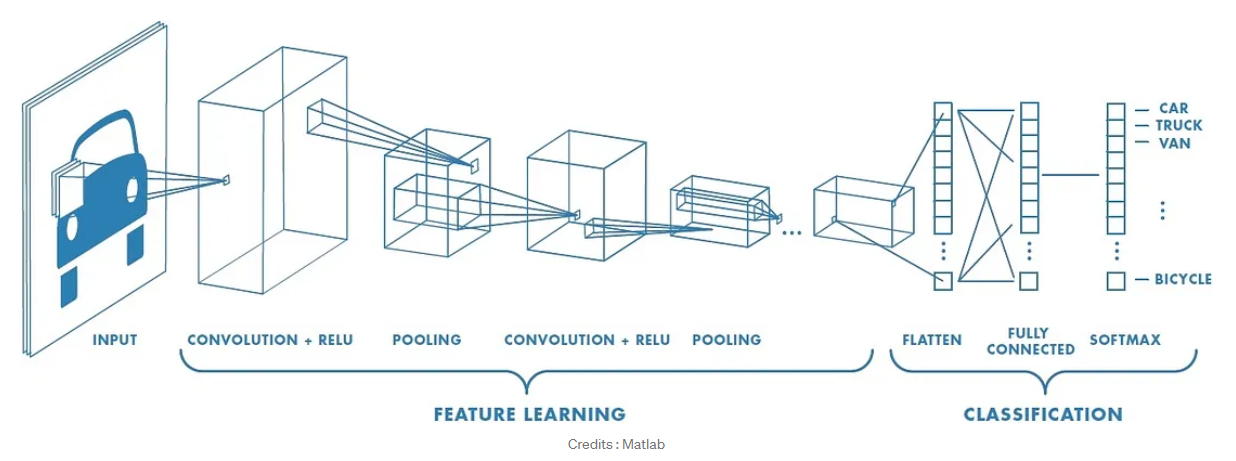
Ditinjau dari hal-hal tersebut maka diperlukannya sebuah program mendeteksi sawi guna memberikan solusi permasalahan hama yang dialami oleh para petani sawi.

Yalcin & Razavi (2016) melakukan klasifikasi tanaman menggunakan metode CNN (*Convolutional Neural Network*). Penelitian tersebut menggunakan foto objek tanaman yang dipotret dari samping. Hasil penelitian tersebut menunjukkan bahwa metode CNN dapat melakukan klasifikasi lebih baik daripada metode SVM (*Support Vector Machine*). Metode CNN merupakan salah satu metode *deep learning* yang mampu melakukan proses pembelajaran mandiri untuk pengenalan objek, ekstraksi objek dan klasifikasi serta dapat diterapkan pada citra resolusi tiggi yang memiliki model distribusi nonparametrik (Zhang et al., 2018).

Maka dari itu Pada percobaan ini dilakukan dua perbandingan arsitektur yaitu CNN from Scratch dan VGG16 karena dilansir oleh pada keras applications VGG16 memiliki akurasi sebesar 71.3%.

***Convolutional Neural Network* (CNN)**

*Convolutional Neural Network* (CNN) termasuk dalam jenis *deep learning* karena kedalaman jaringannya. Deep learning adalah cabang dari machine learning yang dapat mengajarkan komputer untuk melakukan pekerjaan selayaknya manusia, seperti komputer dapat belajar dari proses training (Deng & Yu, 2013). CNN merupakan operasi konvolusi yang menggabungkan beberapa lapisan pemrosesan, menggunakan beberapa elemen yang beroperasi secara paralel dan terinspirasi oleh sistem saraf biologis (Hu et al., 2015). Pada CNN setiap neuron dipresentasikan dalam bentuk 2 dimensi, sehingga metode ini cocok untuk pemrosesan dengan input berupa citra (Maggiori et al., 2016).



**Gambar 1**. Arsitektur CNN

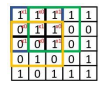
**Gambar 1** menggambarkan arsitektur jaringan menggunakan CNN. Arsitektur ini terdiri dari beberapa komponen, yaitu input, proses ekstraksi fitur, proses klasifikasi, dan output. Proses ekstraksi fitur dilakukan melalui beberapa lapisan tersembunyi atau hidden layer, termasuk lapisan konvolusi, fungsi aktivasi ReLU, dan lapisan pooling. CNN bekerja secara hierarkis, di mana output dari lapisan konvolusi pertama digunakan sebagai input untuk lapisan konvolusi berikutnya. Pada proses klasifikasi, terdapat lapisan fully-connected dan fungsi aktivasi softmax yang menghasilkan output berupa hasil klasifikasi. Metode ini telah digunakan dalam penelitian oleh Katole et al. (2015).

**Lapisan Konvolusi**

Lapisan konvolusi dalam CNN menggunakan filter untuk mengekstraksi objek dari citra input. Filter ini memiliki bobot yang digunakan untuk mendeteksi karakteristik objek, seperti tepi, kurva, atau warna. Melalui proses konvolusi, transformasi linier dari citra input dihasilkan sesuai dengan informasi spasial yang terdapat dalam data tersebut. Filter diterapkan secara berulang, menghasilkan serangkaian bidang receptive.

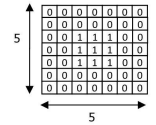
Terdapat beberapa parameter yang dapat diubah untuk memodifikasi sifat setiap lapisan konvolusi, seperti ukuran filter, stride, dan *padding*. *Stride* mengontrol bagaimana filter diterapkan pada data input dengan pergerakan sepanjang ukuran piksel yang telah ditentukan. *Padding* merupakan penambahan piksel dengan nilai tertentu di sekitar data input untuk mencegah hilangnya informasi dalam hasil bidang *receptive*. Biasanya, *padding* menggunakan nilai nol dan dikenal sebagai *zero padding*. Hasil dari bidang *receptive* adalah data tunggal yang menjadi input untuk lapisan konvolusi berikutnya.

Proses konvolusi ini memiliki ilustrasi mengenai stride dan padding yang ditampilkan dalam **Gambar 2** dan **Gambar 3.** Castelluccio et al. (2015) menjelaskan konsep ini dalam penelitiannya.

(a) (b) (c)

**Gambar 2.** Operasi konvolusi dengan stride 1 (a) Input data 5x5 (b) filter 3x3 (c) bidang receptive 3x3



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

**Fungsi Aktifasi ReLU**

ReLU (*Rectification Linear Unit*) adalah operasi yang digunakan untuk memperkenalkan nonlinearitas dan meningkatkan representasi dari sebuah model. Fungsi aktivasi ReLU didefinisikan sebagai f(x) = max(0, x) (Heaton, 2015). Artinya, nilai output dari neuron akan menjadi 0 jika inputnya adalah negatif. Jika nilai inputnya positif, maka output dari neuron akan sama dengan nilai input tersebut (Kim et al., 2016).

ReLU berfungsi sebagai fungsi aktivasi nonlinear dalam jaringan saraf, yang memungkinkan model untuk menangkap hubungan kompleks dan meningkatkan kemampuannya dalam belajar dan menggeneralisasi dari data. Dengan memperkenalkan ambang batas pada nilai nol, ReLU memungkinkan model untuk mengaktifkan neuron secara selektif berdasarkan nilai input yang positif, sementara memfilter nilai negatif. Nonlinearitas ini membantu model untuk belajar dan merepresentasikan fitur-fitur yang lebih ekspresif dan diskriminatif. Penggunaan ReLU telah dibahas dan didokumentasikan oleh para peneliti seperti Heaton (2015) dan Kim et al. (2016).

***Pooling***

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

***Fully Connected Layer***

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

**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 **Persamaan 1.**



dimana:

yijk = vektor yang berisi nilai antara 0 dan 1.

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

Fungsi kesalahan klasifikasi dihitung dengan **Persamaan 2:**



dimana:

l(x,c) = membandingkan prediksi (x) dan label (c)

x = vektor dari probabilitas akhir.

p(k) = xk, k =1.

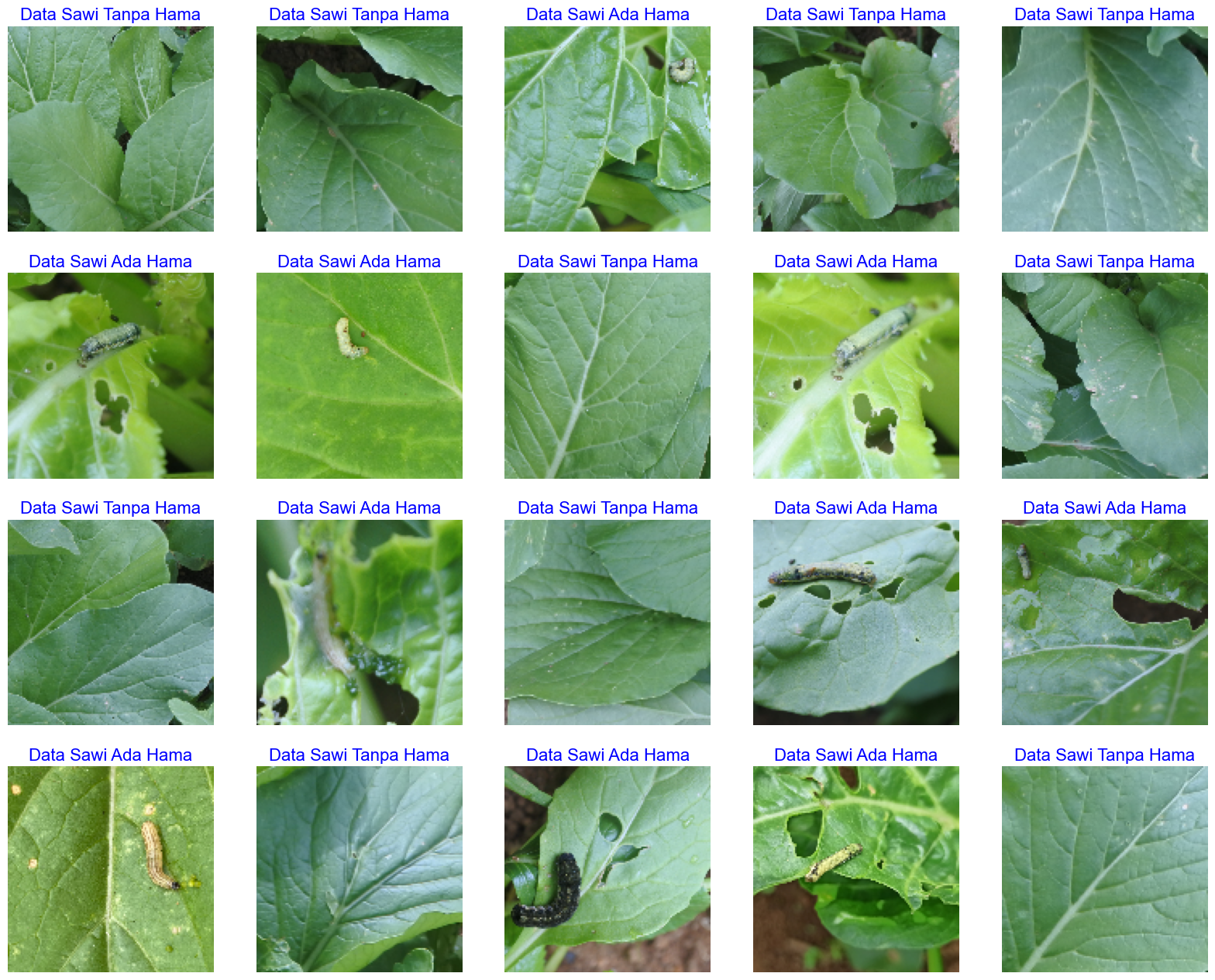
C = banyak kelas.

Untuk mengontrol overfitting, pooling layer digunakan untuk mengurangi representasi ukuran spasial dan mengurangi jumlah parameter. Lapisan dropout memberikan aturan untuk menghilangkan atau menjaga neuron dengan beberapa nilai probabilitas p yang bernilai antara 0 dan 1 (Srivastava et al., 2014). Lapisan dropout berguna untuk memudahkan penggolongan kelasnya.

# **METODOLOGI DAN IMPLEMENTASI**

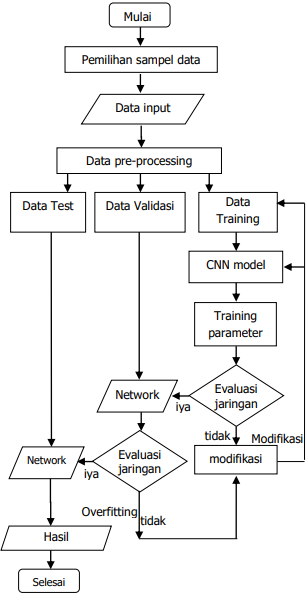
## Metodologi

Pada penelitian ini data yang digunakan menggunakan foto-foto dari sebuah *Data Science Company* yaitu Kaggle, data yang diambil berupa foto sawi dengan dan tanpa hama dapat dilihat pada **Gambar 4.** Pembuatan model CNN dilakukan dengan menggunakan Bahasa pemrogramman *python*.



**Gambar 4.** Kelas tanaman sawi dengan dan tanpa hama

**Diagram Alir**

****

Alur pengerjaan penelitian ini dimulai dari pemilihan data sampel yang digunakan sebagai input untuk data training, validasi dan pengujian/tes. Kemudian merancang jaringan dengan metode CNN untuk melakukan klasifikasi semantik pada objek tanaman. Rancangan jaringan CNN diaplikasikan dengan data training, agar komputer dapat belajar mengenali objek. Apabila pembelajaran jaringan diperoleh hasil yang baik dalam membedakan jenis objek tanaman, maka jaringan tersebut kemudian dilakukan uji coba terhadap data validasi. Apabila data validasi juga menunjukkan hasil yang baik maka jaringan dapat digunakan untuk klasifikasi pada data tesdata, atau pembagian data menjadi set pelatihan dan set pengujian.

**Pengambilan Sampel untuk Data Input**

Input data yang dilakukan pada jaringan ini berupa sampel gambar dari dua jenis klasifikasi yaitu daun sawi dengan hama dan tanpa hama. Ukuran gambar sampel untuk pemrosesan 400x300 piksel dan saat menggunakan model VGG16 yaitu 224x224 piksel. Data untuk masing-masing kelas berjumlah 500 sehingga data keseluruhan berjumlah 1000 data. Data dari du akelas tersebut dibagi menjadi tiga jenis kelompok data, yaitu data *training*, data validasi dan proses pengujian.

# **HASIL DAN PEMBAHASAN**

**Implementasi CNN**

Terdapat tiga tahap dalam mengimplementasikan CNN, yaitu training, validasi dan tes. Tahap training adalah tahap utama untuk melatih jaringan mempelajari data input. Kemudian jaringan tersebut diuji pada data validasi. Apabila memberikan hasil yang baik, maka jaringan tersebut dapat digunakan untuk melakukan klasifikasi data dengan data tes.

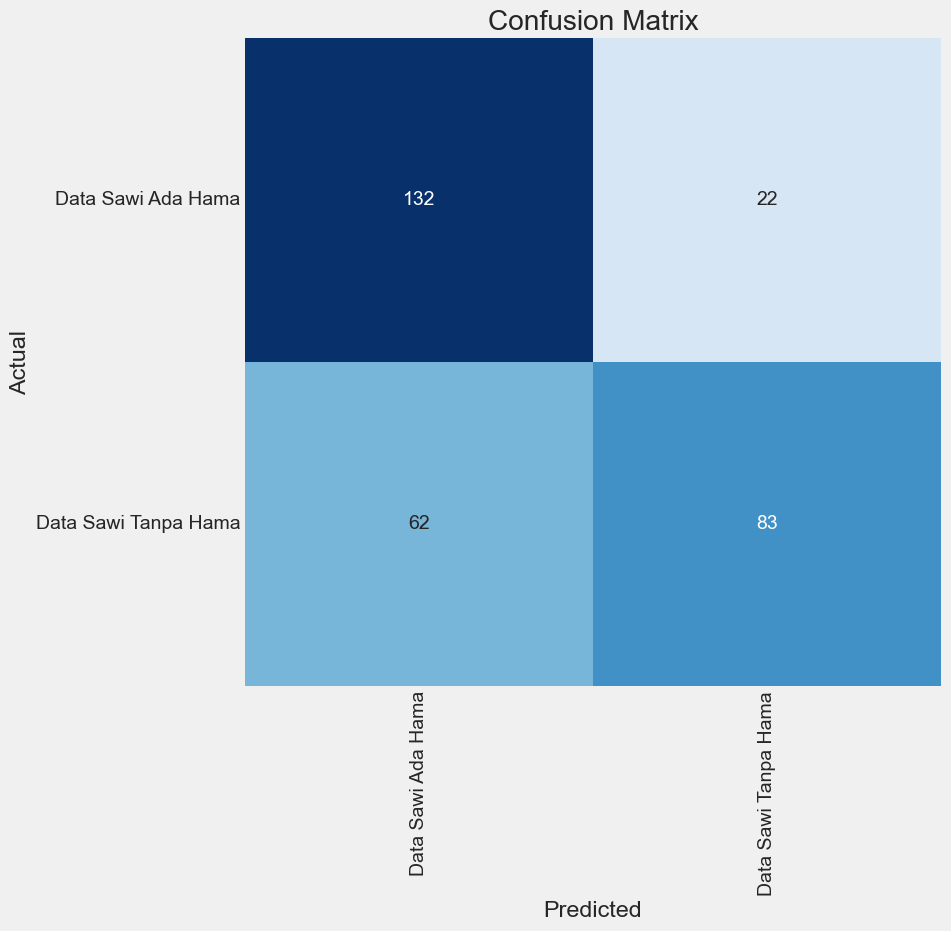
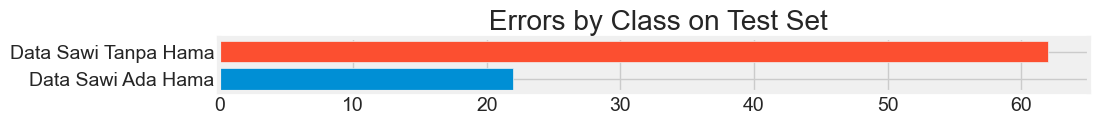
**Hasil Percobaan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Learning Rate | Epoch | Optimizer | Accuracy |
| A | 0.1 | 5 | RMSprop | 71.91% |
| B | 0.01 | 10 | Adam | 61.00% |
| C | 0.01 | 50 | Adam | 84.00% |
| D (VGG16) | 0.01 | 10 | Adam | 95.00% |

**Tabel 1.** parameter dan hasil percobaan

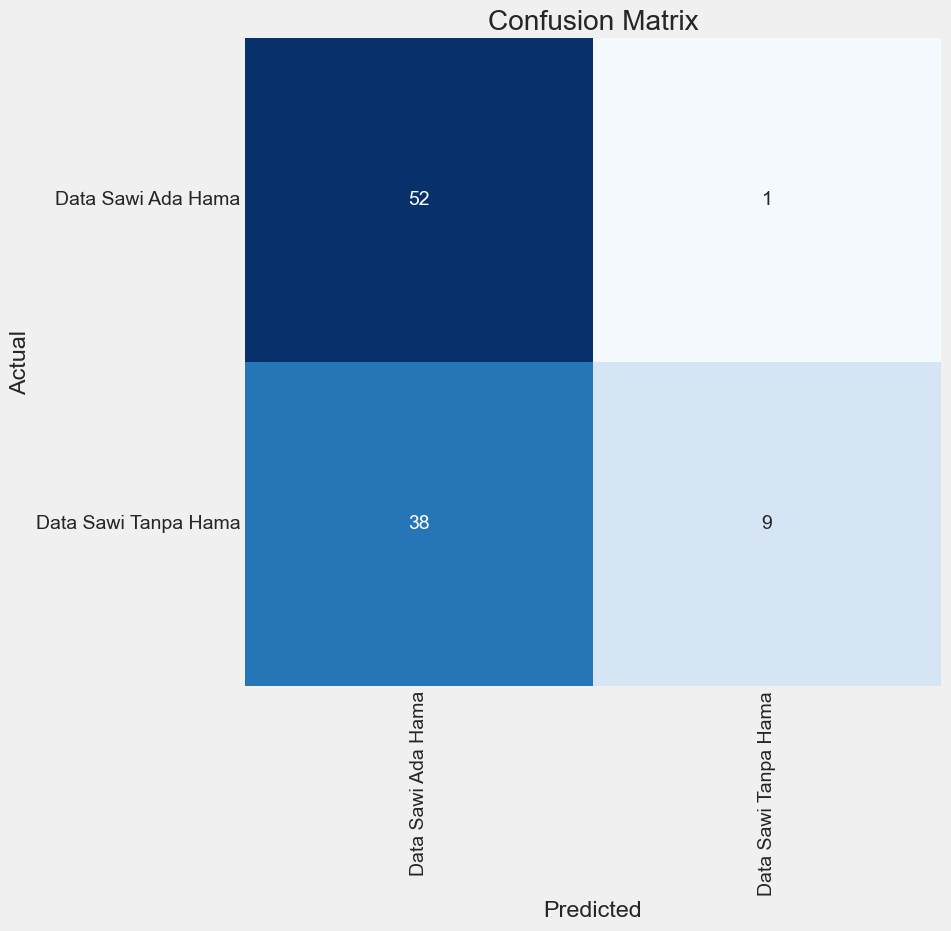
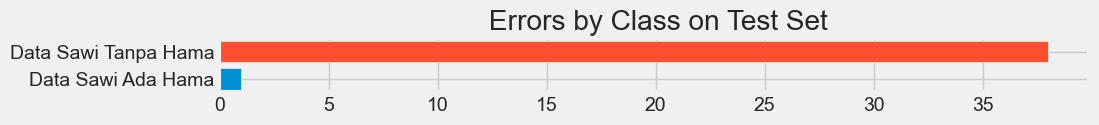
Dilihat dari hasil penelitian yang disajikan pada **Tabel 1.**

Model B dan C Epoch berpengaruh pada Accuracy, tidak hanya Epoch, learning rate pun berpengaruh, semakin kecil learning rate maka akan semakin lambat dalam belajar dalam artian semakin teliti, lalu optimizer pun berperan dalam accuracy dan terakhir yang paling berpengaruh bisa dilihat bahwa penggunaan arsitektur sangat berpengaruh yaitu mencapai acucuracy sebesar 95.00%

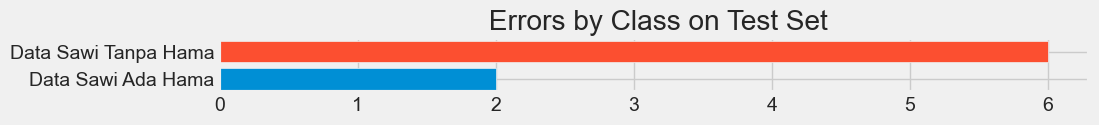


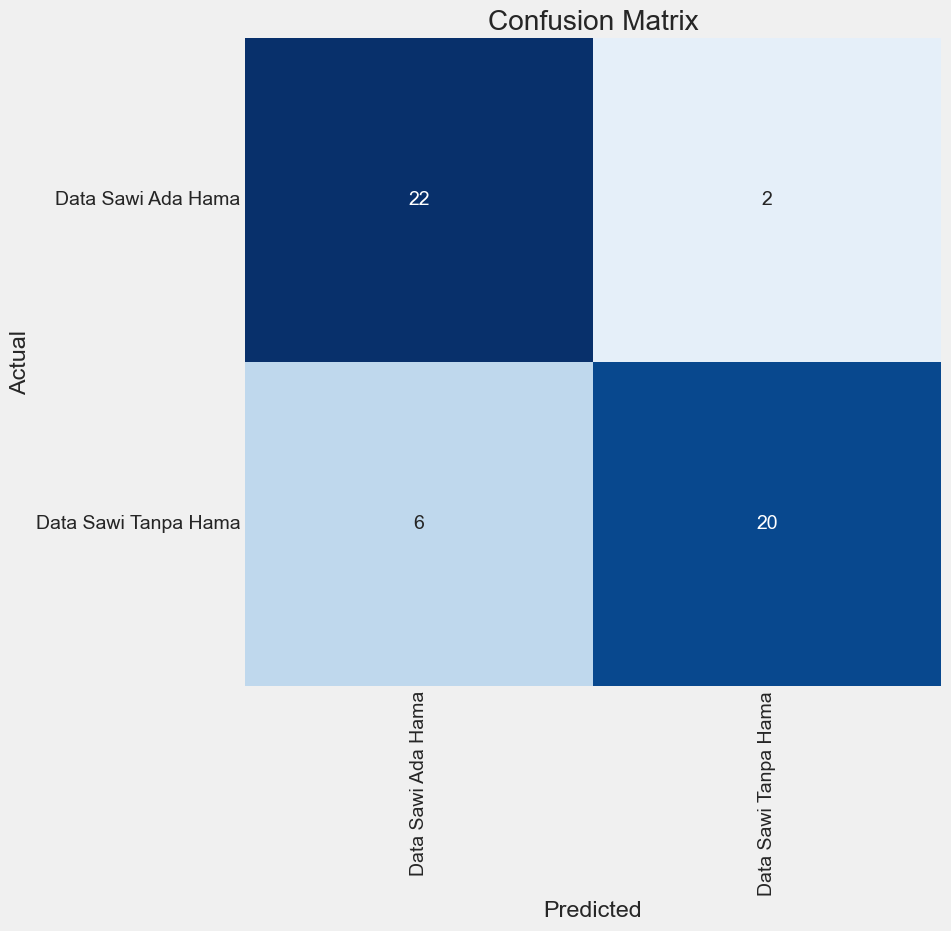
**Gambar 5.1** *Confusion matrix* Model A

Percobaan B

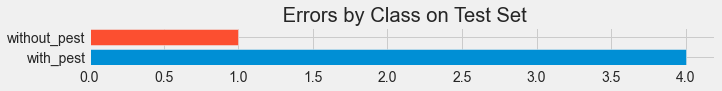
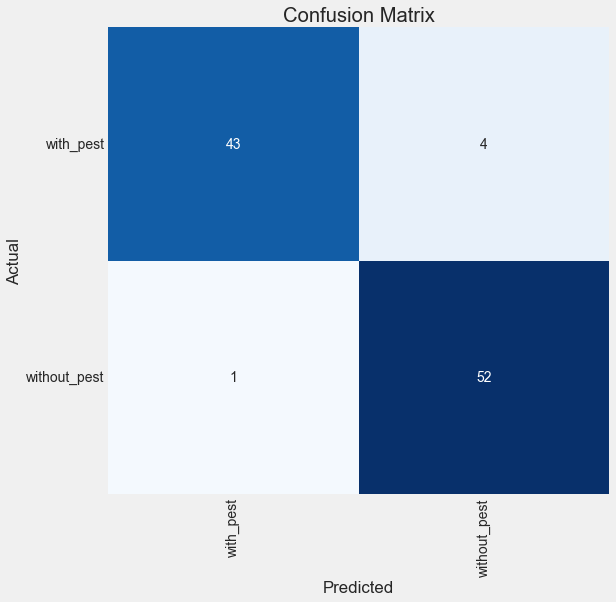


**Gambar 5.2** *Confusion matrix* Model B

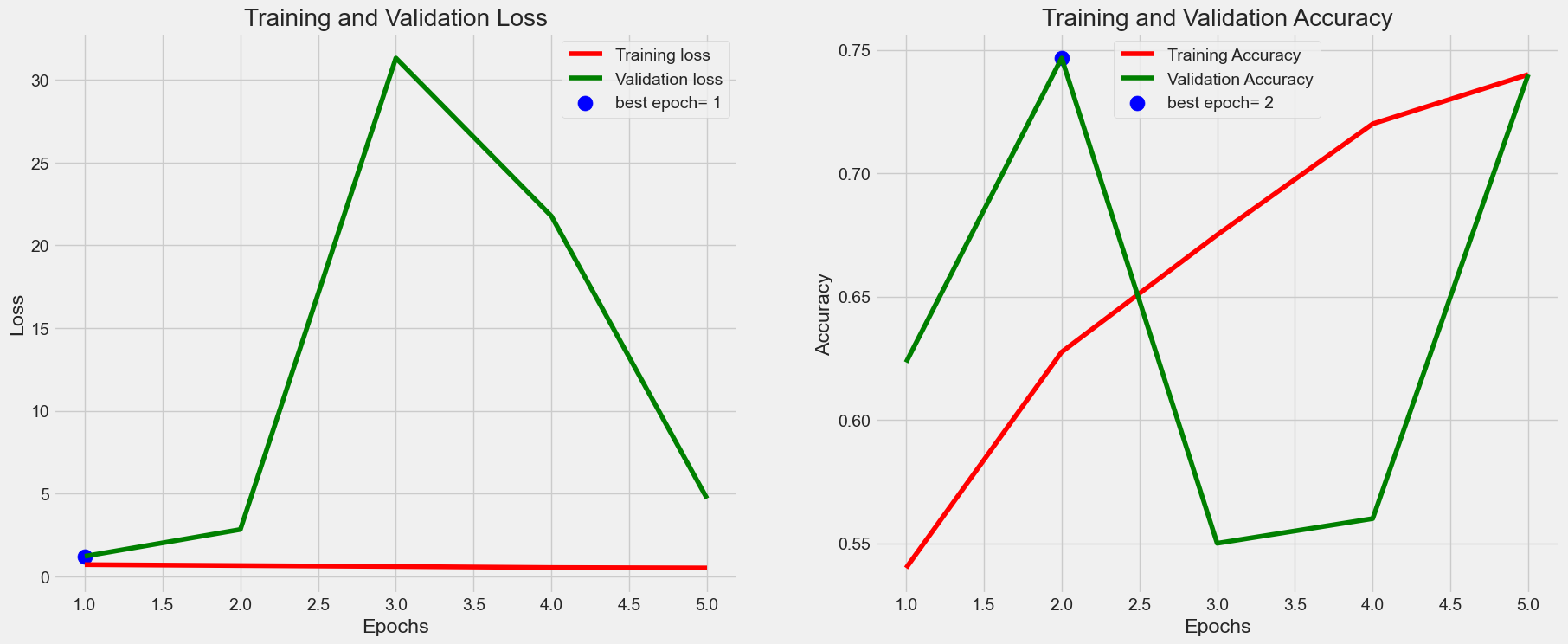




**Gambar 5.3** *Confusion matrix* Model C

****

**Gambar 5.4** *Confusion matrix* Model D



**Gambar 6.1** Grafik *loss* dan *accruation function* model A



**Gambar 6.2** Grafik *loss* dan *accruation function* model B



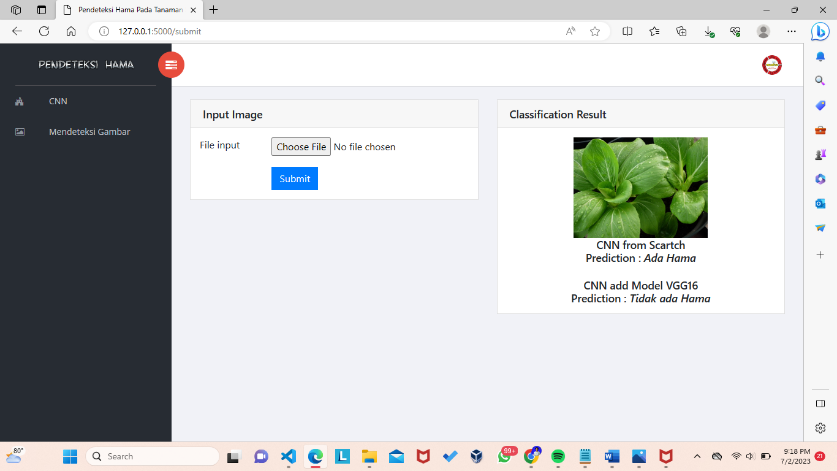
**Gambar 6.3** Grafik *loss* dan *accruation function* model C



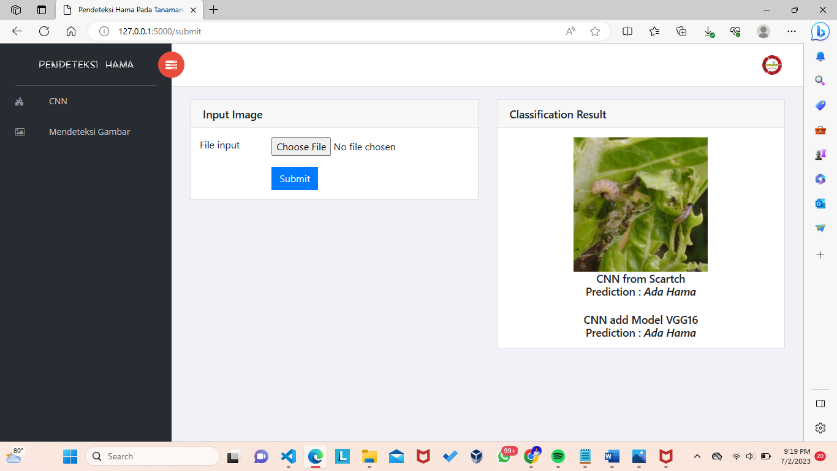
**Gambar 6.4** Grafik *loss* dan *accruation function* model D

Grafik loss function yang bagus itu perlahan turun dan accrucay function itu perlahan naik, pada model A dapat dilihat pada **Gambar 6.1** grafik *loss function* tidak *underfitting*  dan pada *accuration function* pun demikian. Lalu dicoba mengganti dengan model B pada **Gambar 6.2** pada grafik membaik namun pada *validation accuracy*  mengalami penurunan. Lalu pada grafik pada **Gambar 6.3** baik pada *loss function* maupun *accuration function* lebih baik dari pada dua grafik sebelumnya, lalu terakhir pada **Gambar 6.4** yang mana menggunakan arsitektur model VGG16 dapat dilihat grafik hampir stabil dan bagus.

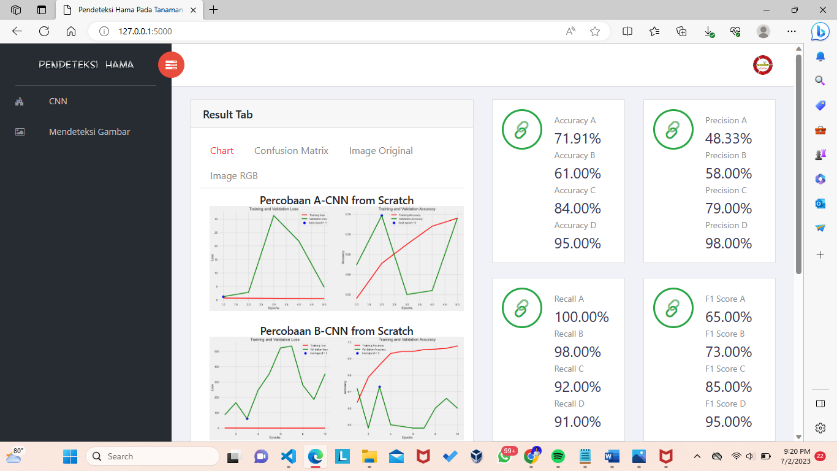
**Impelentasi Program**

****

**Gambar 7.1** Saat ada hama

****

**Gambar 7.2** Saat tidak ada hama

****

**Gambar 7.3** CNN Model

Pada implementasinya diambil dari model diambil yang terbaik yaitu model C dan model D, tingkat akurasi pada model C sudah dapat dibilang cukup baik karena mengasilkan akurasi sebesar 84.00% namun pada implementasinya yang bisa dilihat pada **Gambar 7.2** model ini mengalami overfitting, yang mana ketika diberi gambar di luar dataset maka akan gagal dalam mendektsi hasil yang benar, berbeda halnya dengan model D yang mana menggunakan arsitektur VGG16 dapat dilihat pada **Gambar 7.2** model ini berhasil mendekteksi dengan akurat.

# **KESIMPULAN**

Hasil implementasi metode CNN untuk mendeteksi hama pada daun sawi menunjukan bahwa arsitektur CNN dapat mengklasifikasikan daun sawi yang dihinggapi hama dan daun sawi yang bebas dari hama dengan tingkat akurasi 95% pada model D dan 84.00% pada model C.

Untuk faktor-faktor yang mempengaruhi tingkat akurasi ialah *hyper parameter*nya, arsitektur, optimizer, epoch, dan data augmentasi. Semakin kecil learning rate nya maka proses pembelajaran data akan semakin lambat dengan begitu maka hasil akan lebih akurat, dan semakin besar learning rate nya maka hasil pembacaan data akan semakin cepat, tidak hanya itu data augmentasi pun berperan, semakin banyak foto entah itu di flip atau horizontal dll, maka semakin cerdas program itu dalam mendeteksi.

Dalam grafik pun dapat dibilang bagus jika pada *loss function* grafik perlahan turun dan pada *validation function* grafik perlahan naik

##### **DAFTAR PUSTAKA**

[1]. Nazaruddin. 1999. Budidaya dan Pengaturan Panen Sayuran Dataran Rendah, Penerbit Swadaya: Jakarta

[2]. Direktorat Gizi Departemen Kesehatan Republik Indonesia. 1981. Daftar Komposisi Bahan Makanan, Bhatara Karya Aksara: Jakarta.

[3]. Haryanto, W., T. Suhartinidan E. Rahayu. 2003. Sawidan Selada Edisi Revisi, Penebar Swadaya: Jakarta.

[4]. Wahyudi. 2010. Petunjuk Praktis Bertanam Sayuran. Agromedia Pustaka. Jakarta.

[5]. Yalcin, H., & Razavi, S. (2016). Plant classification using convolutional neural networks. 2016 Fifth International Conference on Agro-Geoinformatics (Agro-Geoinformatics), 1-5. . https://doi.org/10.1109/AgroGeoinformatics.2016.7577698

[6]. Zhang, C., Sargent, I., Pan, X., Gardiner, A., Hare, J., & Atkinson, P. M. (2018). VPRS-Based Regional Decision Fusion of CNN and MRF Classifications for Very Fine Resolution Remotely Sensed Images. IEEE Transactions on Geoscience and Remote Sensing, 1–15. <https://doi.org/10.1109/TGRS.2018.2822783>

[7]. Deng, L., & Yu, D. (2013). Deep Learning: Methods and Applications. Foundations and Trends® in Signal Processing, 7(3–4), 197–387. <https://doi.org/10.1136/bmj.319.7209.0a>

[8]. Hu, F., Xia, G. S., Hu, J., & Zhang, L. (2015). Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. Remote Sensing, 7(11), 14680–14707.

[9]. Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2016). Convolutional Neural Networks for LargeScale Remote-Sensing Image Classification. IEEE Transactions on Geoscience and Remote Sensing, 55(2), 645–657. <https://doi.org/10.1109/TGRS.2016.2612821>

[10]. Katole, A. L., Yellapragada, K. P., Bedi, A. K., Kalra, S. S., & Siva Chaitanya, M. (2015). Hierarchical Deep Learning Architecture for 10K Objects Classification. Computer Science & Information Technology ( CS & IT ), (September), 77–93. <https://doi.org/10.5121/csit.2015.51408>

[11]. Kim, J., Sangjun, O., Kim, Y., & Lee, M. (2016). Convolutional Neural Network with Biologically Inspired Retinal Structure. Procedia Computer Science, 88, 145–154. <https://doi.org/10.1016/j.procs.2016.07.418>

[12]. Bejiga, M. B., Zeggada, A., Nouffidj, A., & Melgani, F. (2017). A convolutional neural network approach for assisting avalanche search and rescue operations with UAV imagery. Remote Sensing, 9(2). <https://doi.org/10.3390/rs9020100>

[13]. Hijazi, S., Kumar, R., & Rowen, C. (2015). Image Recognition Using Convolutional Neural Networks. Cadence Whitepaper, 1–12.

[14]. Albelwi, S., & Mahmood, A. (2017). A Framework for Designing the Architectures of Deep Convolutional Neural Networks. Entropy, 19, 242.

[15]. Vedaldi, A., & Lenc, K. (2015). MatConvNet: Convolutional Neural Networks for MATLAB. In Proceedings of the 23rd ACM International Conference on Multimedia (hal. 689–692). New York, NY, USA: ACM. <https://doi.org/10.1145/2733373.2807412>

[16]. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15, 1929– 1958. <https://doi.org/10.1214/12-AOS1000>