BAB II
TINJAUAN PUSTAKA

2.1. Burung Hama Padi

Aktivitas burung hama pada tanaman padi dilaporkan berbeda-beda tergantung pada fase pertumbuhan tanaman. Pada fase vegetatif, misalnya, burung Pipit Peking (*Lonchura punctulata*) dan Merpati (*Columba sp.*) teramati menyerang area persemaian untuk mencari sumber makanan. Namun, serangan yang lebih intensif dan melibatkan lebih banyak jenis burung terjadi pada fase generatif, yaitu saat padi mulai memasuki tahap masak susu hingga menjelang panen. Pada fase krusial ini, jenis pipit seperti *Lonchura atricapilla* dan *Lonchura punctulata* menjadi hama utama yang dapat menurunkan produksi padi secara signifikan, karena fase ini menyediakan sumber pakan biji padi yang melimpah dan sangat disukai oleh burung (Rinny Kumar, 2023).

2.2. Artificial Intelligence

Artificial Intelligence (AI) adalah bidang studi yang berfokus pada pemahaman dan penciptaan entitas cerdas (mesin) yang dapat bertindak secara efektif dan aman dalam situasi baru. AI memiliki berbagai definisi yang dikategorikan berdasarkan dua dimensi utama—kinerja manusia versus rasional, dan pemikiran versus perilaku—yang menghasilkan empat pendekatan, yaitu bertindak seperti manusia (*Acting humanly*), Berpikir seperti manusia (*Thinking humanly*), Berpikir rasional (*Thinking rationally*), Bertindak rasional (*Acting rationally*). Pendekatan "bertindak rasional" merupakan yang paling dominan dan dianggap sebagai model standar dalam kecerdasan buatan (Stuart Russell & Peter Norvig, 2020).

2.2.1. Machine Learning

Machine Learning (ML) adalah cabang dari Kecerdasan Buatan (AI) yang berfokus pada pengembangan sistem yang mampu belajar secara mandiri dari data, tanpa perlu diprogram berulang kali oleh manusia. Tujuan utama ML adalah agar sistem komputer dapat belajar dan meningkatkan

kinerjanya seiring dengan bertambahnya data yang diolah. ML membutuhkan data yang valid sebagai bahan belajar (proses *training*) sebelum dapat digunakan untuk menghasilkan output yang optimal saat proses pengujian (*testing*). Menurut buku (Soebroto, 2019), Secara umum, terdapat tiga jenis utama machine learning:

- *Supervised Learning*

  Dalam jenis pembelajaran ini, model dilatih menggunakan data yang sudah memiliki label atau target output yang diketahui. Model belajar untuk memetakan input ke output yang sesuai. Contohnya adalah klasifikasi (memprediksi kategori) dan regresi (memprediksi nilai kontinu).

- *Unsupervised Learning*

  Berbeda dengan *supervised learning, unsupervised learning* menggunakan data yang tidak memiliki label. Model bertugas untuk menemukan pola atau struktur tersembunyi dalam data secara mandiri. Contohnya adalah *clustering* (pengelompokan data) dan reduksi dimensi.

- *Reinforcement Learning*

  Jenis pembelajaran ini berprinsip bahwa pengalaman mandiri adalah guru terbaik. Model (disebut agen) belajar melalui interaksi dengan lingkungannya. Agen akan menerima *reward* atau *punishment* berdasarkan tindakan yang diambil, dan tujuannya adalah untuk memaksimalkan total *reward* yang diterima.

2.2.2. Deep Learning

Deep Learning (DL) atau Pembelajaran Mendalam adalah salah satu cabang dari *Machine Learning* yang terdiri dari algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam. Model DL pada dasarnya dibangun berdasarkan Jaringan Saraf Tiruan (*Neural Network*) dengan banyak lapisan tersembunyi *(hidden layer*). Jika suatu jaringan memiliki lebih dari 3 lapisan, maka jaringan tersebut termasuk *Deep Network*.

2.3. Neural Network

Neural Network adalah model komputasi yang terinspirasi dari cara kerja sistem saraf biologis, khususnya otak manusia. Model ini pada dasarnya dibangun untuk memungkinkan mesin belajar dan membuat keputusan seperti manusia. Secara umum, Jaringan Saraf Tiruan terdiri dari unit-unit pemrosesan sederhana yang disebut neuron atau node, yang saling terhubung dan tersusun dalam lapisan-lapisan (layers). Model pada *Deep Learning* secara fundamental dibangun berdasarkan Jaringan Saraf Tiruan. Salah satu contoh awal yang disebutkan adalah model sel saraf tiruan yang dibuat oleh Warren McCulloch dan Walter Pitts pada tahun 1943 (Soebroto, 2019). Berdasarkan (Soebroto, 2019), *neural network* memiliki beberapa tipe antara lain:

- *Convolutional Neural Networks* (CNN)

  Jaringan ini sangat mirip dengan jaringan saraf tiruan standar yang dapat divisualisasikan sebagai kumpulan neuron yang disusun sebagai graf asiklik. CNN memiliki ciri khas yaitu terdapat lapisan tersembunyi yang hanya terhubung ke subset neuron di lapisan sebelumnya. Karena konektivitas tersebut, CNN dapat mempelajari fitur secara implisit. Arsitektur CNN menghasilkan ekstraksi fitur hirarki. Contoh penerapan CNN adalah studi kasus untuk prediksi maupun klasifikasi, seperti prediksi curah hujan.
- *Recurrent Neural Networks* (RNN)

  Ciri khas RNN adalah memiliki memori untuk mengingat semua informasi tentang apa yang telah dihitung pada keadaan sebelumnya, untuk digunakan pada proses berikutnya. RNN dikembangkan untuk mengatasi masalah gradien yang menumpuk dan saling berbenturan.
- *Deep Belief Networks* (DBN)

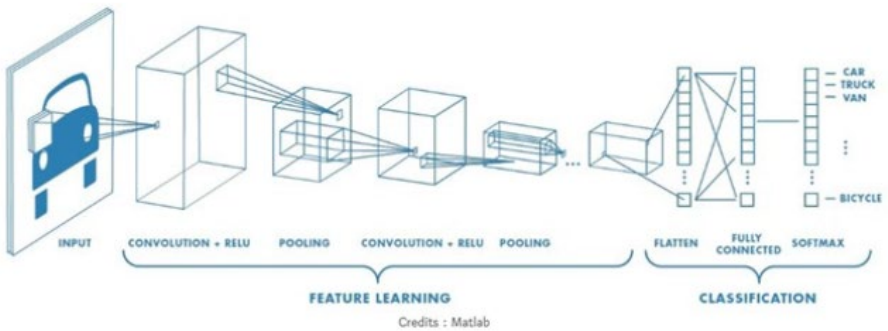
  Merupakan salah satu metode *non-convolutiona*l pertama yang berhasil menerima dan melakukan proses pelatihan sesuai standar arsitektur deep learning. DBN adalah model generatif dengan satu atau beberapa lapisan variabel laten (*hidden layer*). Variabel laten biasanya biner, sedangkan unit yang terlihat bisa biner atau non-biner. Tidak ada koneksi intralayer, dan koneksi antara dua lapisan teratas tidak diberikan arah, sedangkan koneksi

antar lapisan lainnya mengarah ke lapisan input. *Restricted Boltzmann Machine* (RBM) adalah blok bentuk visual dari model probabilistik yang tidak berarah yang mengandung lapisan variabel yang dapat diobservasi dan minimal satu hidden layer, RBM dapat ditumpuk untuk membentuk DBN.

2.4. *Convolutional Neural Network* (CNN)

Struktur jaringan *Convolutional Neural Network* (CNN) secara mendasar dapat dibagi menjadi dua tahapan utama, yaitu Pembelajaran Fitur (*Feature Learning*) dan Klasifikasi (*Classification*). Tahap pembelajaran fitur mencakup lapisan-lapisan seperti Lapisan Konvolusi (Convolutional *Layer*), unit aktivasi ReLU (*Rectified Linear Unit*), dan Lapisan Pooling (*Pooling Layer*). Di sisi lain, tahap klasifikasi bertujuan untuk menyimpulkan hasil dari pembelajaran fitur menjadi penentuan satu kelas spesifik untuk input yang diberikan. Lapisan-lapisan dalam tahap pembelajaran fitur berfungsi untuk mengubah data input menjadi sekumpulan fitur-fitur numerik (vektor) yang merepresentasikan ciri-ciri khas dari input tersebut.

Kemampuan belajar CNN dimulai dari pengenalan pola atau objek pada area-area kecil (regional) dari input, yang terhubung ke volume data masukan. Dalam proses konvolusi, nilai piksel dari citra input dikalikan dengan nilai-nilai yang terdapat pada sebuah filter (Suriani Alamgunawan & Dr. Yosi Kristian, 2020). Berdasarkan aristektur LeNet5, terdapat 4 macam *layer* utama pada sebuah CNN yaitu *convolutional layer, relu layer, subsampling layer/pooling layer,* dan *fully connected layer*. Arsitektur CNN dapat dilihat pada Gambar 3.

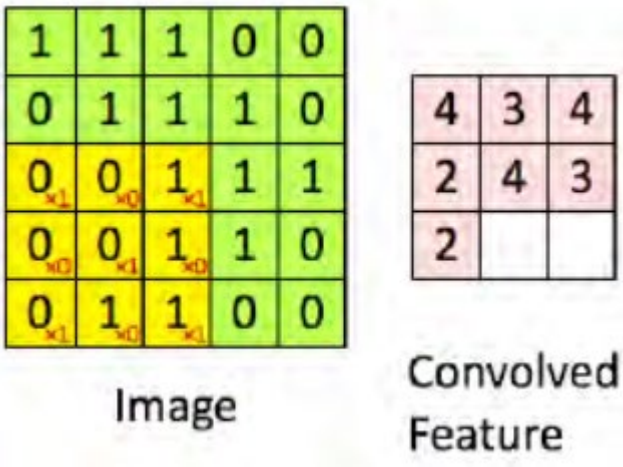


Gambar 3. *Arsitektur* CNN

Sumber: (Yuliany & Nur Rachman, 2022)

2.4.1. Konvolusi

Operasi utama dalam sebuah CNN terjadi di *Convolution Layer*, di mana proses matematis yang disebut konvolusi diterapkan pada hasil dari layer sebelumnya. Konvolusi, yang berarti mengaplikasikan satu fungsi berulang kali ke output fungsi lain, dalam konteks citra melibatkan penggunaan kernel yang digeser melintasi gambar. Tujuannya adalah untuk mengekstraksi fitur dari citra input melalui transformasi linier yang mempertimbangkan informasi spasial. Bobot pada layer ini menentukan kernel konvolusi, sehingga kernel tersebut dapat disesuaikan (dilatih) sesuai dengan input CNN (Eka Putra, 2016).



Gambar 4. Operasi Konvolusi

Sumber: (Eka Putra, 2016)

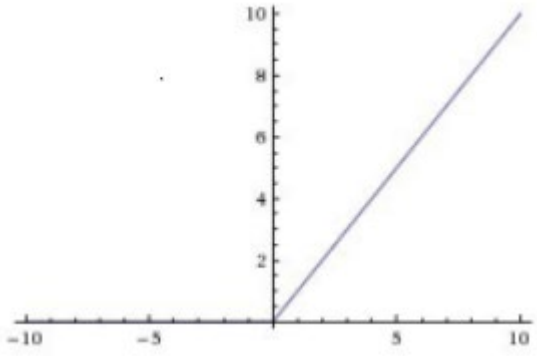
2.4.2. Pooling Layer

Lapisan pooling memiliki peran penting dalam memperkecil ukuran atau dimensi citra. Dengan melakukan *downsampling* (pengurangan ukuran), lapisan ini mengurangi jumlah parameter dalam model, yang pada gilirannya mempercepat proses komputasi. Dua metode *pooling* yang paling umum adalah *Max Pooling*, yang memilih nilai terbesar dari suatu area, dan *Average Pooling*, yang menghitung nilai rata-ratanya. *Max Pooling* lebih sering digunakan karena kemampuannya menangkap fitur yang paling menonjol (nilai tertinggi) dalam citra, sedangkan *Average Pooling* memberikan representasi rata-rata fitur pada tiap dimensi (Suriani Alamgunawan & Dr. Yosi Kristian, 2020).

2.4.3. *Rectified Linear Unit* (ReLu)

Fungsi aktivasi merupakan komponen krusial yang tidak dapat diabaikan dalam jaringan saraf tiruan (*neural network*). Fungsi ini sangat penting karena memberikan kemampuan non-linear pada jaringan. Terdapat berbagai macam fungsi aktivasi yang bisa digunakan, dan salah satu yang populer adalah ReLU (*Rectified Linear Unit*). ReLU dikenal dapat membantu mengatasi masalah vanishing gradient dengan menerapkan aktivasi secara element-wise. Artinya, fungsi ini bekerja dengan menetapkan batas pada angka nol: jika nilai input (x) kurang dari atau sama dengan nol ($x \leq 0$), maka hasilnya diubah menjadi nol (0); namun, jika nilai input lebih besar dari nol ($x > 0$), maka hasilnya tetap nilai input itu sendiri (x) (Suriani Alamgunawan & Dr. Yosi Kristian, 2020).

$$f(x) = \max(0, x) = \begin{cases} 0 & jika\ x \leq 0 \\ x & jika\ x > 0 \end{cases} \quad (1)$$



Gambar 5. *Rectified Linear Unit* (ReLU)

2.4.4. Dropout

Dropout adalah sebuah teknik yang dikembangkan untuk mencegah *overfitting* pada jaringan saraf tiruan (*neural networks*), terutama pada jaringan yang memiliki banyak parameter. *Overfitting* adalah masalah serius di mana jaringan belajar pola-pola yang sangat kompleks yang hanya ada pada data pelatihan (seringkali akibat sampling noise) tetapi tidak dapat digeneralisasikan dengan baik ke data uji yang belum pernah dilihat (Srivastava dkk., 2014).

2.4.5. Flatten

Proses *flattening* dilakukan dengan tujuan utama untuk mengubah *feature map* yang telah diperoleh dari layer sebelumnya (yaitu, setelah *Convolutional Layer* dan *Pooling Layer*) menjadi vektor satu dimensi. Pengubahan ini penting agar feature map tersebut dapat diklasifikasikan dengan *fully-connected layer* dan *softmax*. Dengan kata lain, *layer Flattening* bertindak sebagai jembatan antara feature learning yang menghasilkan representasi data dalam bentuk matriks multi-dimensi dan *fully-connected layer* yang membutuhkan input dalam bentuk vektor satu dimensi untuk proses klasifikasi (Yusuf dkk., 2019).

2.4.6. Softmax

flattening berperan sebagai penghubung antara layer yang memproses fitur spasial (seperti *convolutional* dan pooling) dengan *fully-connected layer* yang membutuhkan input linear. Sementara *Softmax* adalah fungsi di layer terakhir yang mengubah output mentah menjadi probabilitas kelas, memungkinkan model untuk membuat prediksi akhir (Yusuf dkk., 2019).

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (2)$$

di mana K adalah jumlah kelas dan softmax mengonversi logits menjadi distribusi probabilitas (Yusuf dkk., 2019).

2.4.7. Global Average Pooling

Global Average Pooling adalah operasi reduksi dimensi yang menghitung nilai rata-rata dari seluruh aktivasi spasial untuk setiap feature map. Berbeda dengan average pooling konvensional yang menggunakan window berukuran terbatas (misalnya 2×2 atau 3×3), GAP mengambil rata-rata dari keseluruhan dimensi spasial feature map. Operasi ini mentransformasi tensor tiga dimensi (tinggi × lebar × channel) menjadi vektor satu dimensi dengan panjang sama dengan jumlah channe (Lin dkk., 2014). Diberikan feature map F dengan dimensi H × W × K, dimana H adalah tinggi (height), W adalah lebar (width), dan K adalah jumlah channel atau filter, maka operasi Global Average Pooling untuk channel

ke-k didefinisikan sebagai:

$$GAP_k = \left(1/(H \times W)\right) \sum_{i=1}^{H} \sum_{j=1}^{W} F_{i,j,k} \quad (3)$$

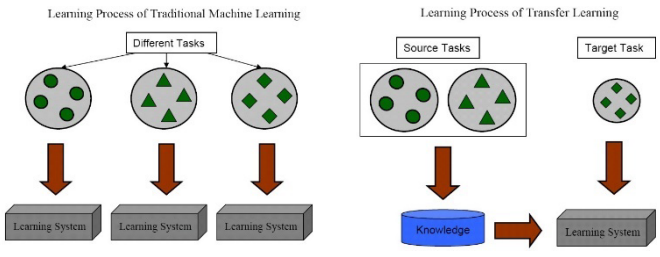
di mana:

- H adalah tinggi (height) feature map
- W adalah lebar (width) feature map
- K adalah jumlah channel/filter
- F_{i,j,k} adalah nilai pada posisi (i,j) di channel k
- k = 1, 2, ..., K

2.4.8. Fully Connected Layer

Sebelum memasuki *Fully Connected Layer*, feature map akan melalui tahap transformasi. Setiap neuron pada *convolution layer* perlu diubah menjadi data satu dimensi terlebih dahulu, sebuah proses yang melibatkan pengubahan bentuk (*reshape*) atau yang kita sebut sebagai *flattening* Proses *flattening* ini menghasilkan vektor satu dimensi yang kemudian digunakan sebagai input bagi *Fully Connected Layer. Fully Connected Layer* bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. Karena proses ini menyebabkan data kehilangan informasi spasialnya dan tidak reversibel, *Fully Connected Layer* hanya dapat diimplementasikan di akhir jaringan (Eka Putra, 2016).

2.5. Transfer Learning



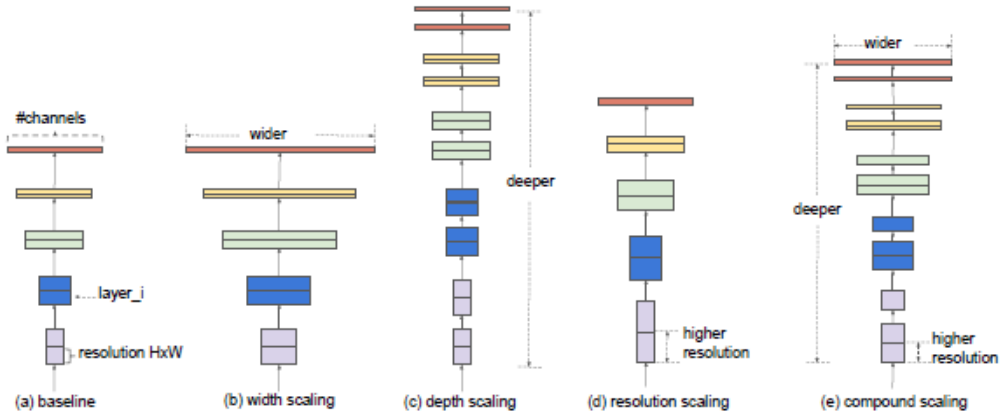
Gambar 6. Konsep Transfer Learning

Sumber: (Pan & Yang, 2010a)

Transfer learning adalah paradigma pembelajaran yang memanfaatkan pengetahuan dari satu atau lebih tugas sumber untuk meningkatkan kinerja model pada domain target, sehingga kita tidak perlu selalu melatih model dari nol sebuah pendekatan yang ampuh ketika asumsi klasik "data latih dan data uji berada pada ruang fitur serta distribusi yang sama" tidak terpenuhi dan pelabelan data baru mahal atau sulit dilakukan. Dalam praktiknya, transfer learning kerap berjalan beriringan dengan domain adaptation, yaitu proses mengurangi perbedaan distribusi antar-domain agar pengetahuan dari sumber lebih relevan untuk target. Secara umum terdapat tiga skenario utama yaitu inductive transfer learning ketika label pada domain target tersedia; transductive transfer learning ketika label hanya ada di sumber dan unsupervised transfer learning ketika keduanya tanpa label. Skenario ini juga dapat dipandang sebagai homogeneous (ruang fitur dan label sumber-target sama) versus heterogeneous (salah satu atau dua ruang berbeda). Pendekatan teknisnya meliputi instance-based, feature-based (mencari representasi bersama baik transformasi asimetris dari sumber ke target maupun simetris ke ruang laten bersama), parameter-based (berbagi bobot atau priors model), relational-based (mentransfer aturan atau relasi di data berjejaring) (Zhuang dkk., 2020b).

2.6. EfficientNet

EfficientNet merupakan keluarga arsitektur jaringan saraf konvolusional yang dikembangkan oleh Google Research pada tahun 2019 untuk mengatasi permasalahan efisiensi dalam deep learning. Arsitektur ini didesain berdasarkan prinsip *compound scaling* yang melakukan penskalaan terkoordinasi terhadap tiga dimensi utama jaringan neural, kedalaman (*depth*), lebar (*width*), dan resolusi input (*resolution*) secara bersamaan menggunakan koefisien tunggal. Berbeda dengan pendekatan konvensional yang hanya meningkatkan satu dimensi (misalnya hanya menambah lapisan atau memperbesar resolusi input), EfficientNet menerapkan strategi *compound scaling* yang mempertimbangkan ketiga faktor secara proporsional. Pendekatan ini didasarkan pada observasi bahwa peningkatan performa model tidak optimal jika hanya dilakukan pada satu dimensi saja (Tan & Le, 2020).



Gambar 7. Model Scaling

Sumber: (Tan & Le, 2020)

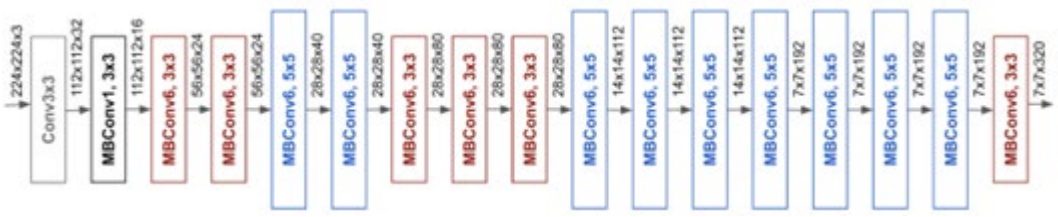
Gambar 7 menunjukkan perbandingan berbagai strategi penskalaan model:

a) Baseline model sebagai titik awal
b) Width scaling: memperlebar jaringan dengan menambah channel
c) Depth scaling: memperdalam jaringan dengan menambah layer
d) Resolution scaling: meningkatkan resolusi input
e) Compound scaling: kombinasi proporsional ketiga pendekatan

Formula compound scaling yang digunakan adalah:

- depth: $d = \alpha^{\varphi}$
- width: $w = \beta^{\varphi}$
- resolution: $r = \gamma^{\varphi}$

dimana α, β, γ adalah konstanta yang ditentukan melalui grid search, dan φ adalah koefisien yang mengontrol seberapa banyak sumber daya komputasi yang tersedia untuk penskalaan model (Tan & Le, 2020).



Gambar 8. Arsitektur EfficientNet

Sumber : (Tan & Le, 2020)

Gambar 8 menggambarkan struktur dasar EfficientNet yang terdiri dari rangkaian

blok building block utama. EfficientNet mengadopsi blok MBConv dari MobileNetV2 yang menggunakan *depthwise separable convolution* untuk efisiensi komputasi (Sandler dkk., 2018). EfficientNet tersedia dalam 8 varian (B0-B7) dengan kompleksitas dan performa yang meningkat. Model B0 berfungsi sebagai baseline yang kemudian diskalakan menggunakan compound scaling untuk menghasilkan varian B1 hingga B7.

Tabel 1. Variant model EfficientNet

Sumber: (Tan & Le, 2020)

Varian Model	Jumlah Parameter	Ukuran Input
EfficientNet-B0	5.3 M	224x224
EfficientNet-B1	7.8 M	240x240
EfficientNet-B2	9.2 M	260x260
EfficientNet-B3	12 M	300x300
EfficientNet-B4	19 M	380x380
EfficientNet-B5	30 M	456x456
EfficientNet-B6	43 M	528x528
EfficientNet-B7	66 M	600x600

Dalam transfer learning, EfficientNet yang telah dilatih pada dataset ImageNet dapat dimanfaatkan sebagai feature extractor untuk tugas klasifikasi yang lebih spesifik (Pan & Yang, 2010; Russakovsky dkk., 2015). Proses ini melibatkan:

1. Feature Extraction: Menggunakan lapisan konvolusional sebagai ekstraksi fitur
2. Head Replacement: Mengganti lapisan klasifikasi akhir sesuai jumlah kelas target

Pendekatan ini memungkinkan pemanfaatan representasi fitur yang telah dipelajari dari dataset besar (ImageNet) untuk diterapkan pada domain yang lebih spesifik dengan data yang terbatas, sehingga dapat mencapai performa yang baik dengan waktu training yang lebih efisien (Zhuang dkk., 2020). EfficientNet menawarkan beberapa keunggulan dibanding arsitektur CNN tradisional sebagi berikut:

1. Efisiensi Parameter Mencapai akurasi tinggi dengan jumlah parameter yang relatif sedikit

2. Skalabilitas Compound scaling memungkinkan penyesuaian model sesuai resource yang tersedia
3. Performa State-of-the-art accuracy pada berbagai benchmark dataset
4. Fleksibilitas Dapat diadaptasi untuk berbagai tugas computer vision

Karakteristik ini menjadikan EfficientNet sebagai pilihan yang optimal untuk aplikasi praktis yang membutuhkan keseimbangan antara akurasi dan efisiensi komputasi (Tan & Le, 2020).

2.7. Mobile Inverted Bottleneck Convolution (MBConv)

Mobile Inverted Bottleneck Convolution (MBConv) adalah komponen utama dalam arsitektur EfficientNet, khususnya pada varian EfficientNetB0, yang berperan penting dalam meningkatkan efisiensi model dan mengurangi kompleksitas komputasi. MBConv menggunakan struktur inverted residual, yang pertama-tama memperluas jumlah channel input, kemudian menerapkan konvolusi depthwise untuk mengurangi beban komputasi, dan akhirnya mengontraksi kembali jumlah channel. Desain ini memungkinkan model untuk mengekstraksi fitur dengan lebih baik, sambil tetap menjaga efisiensi komputasi.

Pada EfficientNetB0, MBConv digunakan untuk meningkatkan efisiensi dengan mengurangi jumlah parameter dan beban komputasi yang diperlukan dalam pemrosesan citra. Struktur MBConv ini memungkinkan model untuk mempertahankan kinerja tinggi meskipun dengan lebih sedikit parameter dibandingkan dengan arsitektur tradisional seperti ResNet50, menjadikannya cocok untuk diterapkan pada perangkat dengan sumber daya terbatas, seperti perangkat mobile. Selain itu, penambahan operasi squeeze-and-excitation (SE) dalam MBConv semakin meningkatkan mekanisme perhatian channel, sehingga jaringan menjadi lebih selektif dalam memfokuskan perhatian pada fitur yang relevan(Xue dkk., 2022) .

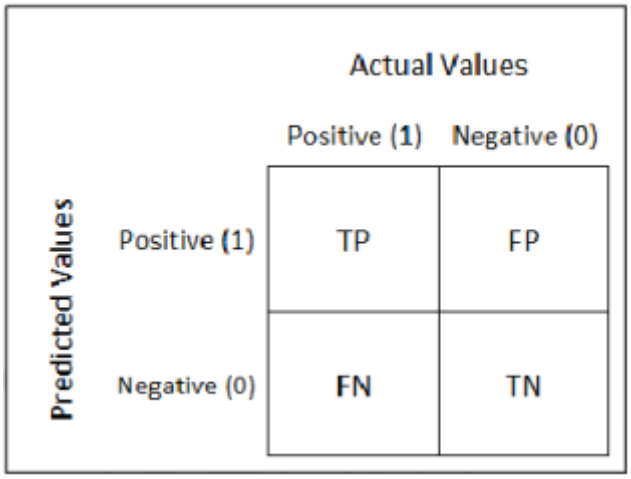
2.8. ImageNet dan Pre-trained Models

ImageNet merupakan dataset fundamental dalam computer vision yang terdiri dari lebih dari 14 juta gambar yang dikategorikan ke dalam hierarki WordNet. Dataset

ini menjadi standar de facto untuk pre-training model CNN karena keragaman dan skala datanya yang masif (Deng dkk., 2009; Krizhevsky dkk., t.t.). Penggunaan model pre-trained ImageNet telah terbukti efektif untuk berbagai tugas downstream, termasuk klasifikasi spesies burung. Hal ini dimungkinkan karena fitur-fitur visual yang dipelajari dari ImageNet bersifat hierarkis dan transferable - mulai dari deteksi tepi dan tekstur di layer awal hingga konsep objek kompleks di layer akhir (Kornblith dkk., 2019).

2.9. Confusion Matrix

Confusion matrix adalah sebuah alat atau metode yang sering dipakai untuk mengevaluasi kinerja sebuah model klasifikasi, terutama untuk mengukur tingkat akurasinya. Secara sederhana, confusion matrix disajikan dalam bentuk tabel. Tabel ini membandingkan hasil prediksi yang dibuat oleh model dengan nilai aktual (atau kelas sebenarnya) dari data uji. Tujuannya adalah untuk menunjukkan secara jelas berapa banyak data yang diklasifikasikan dengan benar dan berapa banyak yang salah oleh model. Berikut adalah tabel confusion pada gambar 5.



Gambar 9. Tabel Confusion Matrix
Sumber: (Yoga Wibowo dkk., 2023)

Tabel *confusion matrix* memiliki empat komponen utama yang merupakan kombinasi dari nilai prediksi dan nilai aktual:

1. *True Positive* (TP) Ini adalah jumlah data yang sebenarnya termasuk dalam kelas positif dan diprediksi oleh model sebagai kelas positif juga. Artinya, ini adalah prediksi positif yang benar.

2. *False Negative* (FN) Ini adalah jumlah data yang sebenarnya termasuk dalam kelas positif, tetapi diprediksi oleh model sebagai kelas negatif. Artinya, ini adalah prediksi negatif yang salah (model gagal mendeteksi kelas positif).
3. *False Positive* (FP) Ini adalah jumlah data yang sebenarnya termasuk dalam kelas negatif, tetapi diprediksi oleh model sebagai kelas positif. Artinya, ini adalah prediksi positif yang salah (model salah mengira sebagai kelas positif, sering disebut "*false alarm*").
4. *True Negative* (TN) Ini adalah jumlah data yang sebenarnya termasuk dalam kelas negatif dan diprediksi oleh model sebagai kelas negatif juga. Artinya, ini adalah prediksi negatif yang benar.

Berdasarkan nilai-nilai TP, FN, FP, dan TN ini, kita dapat menghitung beberapa metrik evaluasi penting lainnya:

- Akurasi (*Accuracy*) Mengukur seberapa sering model membuat prediksi yang benar secara keseluruhan (baik positif maupun negatif) dibandingkan dengan total data.

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

- Presisi (*Precision*) Mengukur seberapa banyak prediksi positif yang dibuat model benar-benar positif. Ini fokus pada ketepatan prediksi positif.

$$Presisi = \frac{TP}{TP + FP} \quad (5)$$

- Recall (*Sensitivity*) Mengukur seberapa banyak kasus positif aktual yang berhasil diidentifikasi dengan benar oleh model. Ini fokus pada kemampuan model menemukan semua kasus positif.

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

- *F1-Score* adalah rata-rata khusus dari precision dan recall yang memberikan hasil lebih ketat dibandingkan rata-rata biasa. Cara kerjanya adalah jika salah satu nilai (precision atau recall) rendah, maka F1-Score akan ikut

rendah juga. Ini berbeda dengan rata-rata biasa yang bisa tetap tinggi meskipun salah satu nilai rendah.

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN} \quad (7)$$

2.10. Cross X Validation

Cross-validation (CV) merupakan prosedur evaluasi statistik yang membagi data ke dalam beberapa lipatan (fold) lalu melakukan pelatihan dan pengujian secara berulang pada partisi yang saling melengkapi untuk memperoleh estimasi kinerja yang lebih tidak bias dibanding hold-out tunggal. Praktik yang umum adalah k-fold CV (sering k=5 atau k=10) dengan stratifikasi pada tugas klasifikasi agar proporsi kelas seimbang di setiap lipatan; metrik seperti akurasi, presisi, dan recall dihitung pada tiap lipatan lalu dirata-ratakan beserta simpangan bakunya. Pendekatan ini menurunkan varians estimasi, membantu deteksi overfitting, dan memberi dasar perbandingan yang adil antar-model/algoritma (Toro & Sri Lestari, t.t.)

2.11. Software Pengembangan Model

2.11.1. Python

Python banyak dipilih sebagai bahasa kerja untuk pembelajaran mesin dan visi komputer karena ekosistem pustakanya matang (NumPy, scikit-learn, TensorFlow/Keras, dll.), sintaks yang ringkas, serta antarmuka tingkat tinggi yang tetap dapat memanggil pustaka berperforma tinggi di bawahnya. Survei terkini menegaskan peran Python sebagai bahasa dominan di sains data/ML berkat kombinasi produktivitas dan performa melalui binding ke pustaka C/CUDA, sekaligus konsistensi API yang memudahkan replikasi riset (Raschka dkk., 2020).

2.11.2. TensorFlow & Keras

TensorFlow menyediakan platform pembelajaran mesin end-to-end, sedangkan Keras (sebagai API tingkat-tinggi resmi di TensorFlow) menyederhanakan perancangan dan pelatihan jaringan saraf dalam—termasuk arsitektur CNN untuk klasifikasi citra. Efektivitas keduanya sebagai “tooling”

riset telah banyak ditunjukkan pada publikasi jurnal: misalnya pengenalan wajah berbasis Keras serta pengenalan sidik jari berbasis TensorFlow, keduanya memanfaatkan CNN dan menunjukkan bahwa kombinasi TF/Keras relevan untuk penelitian vision dari sisi kemudahan prototipe sekaligus kinerja (Santoso & Ariyanto, t.t.).

2.11.3. Google Colaboratory (Colab)

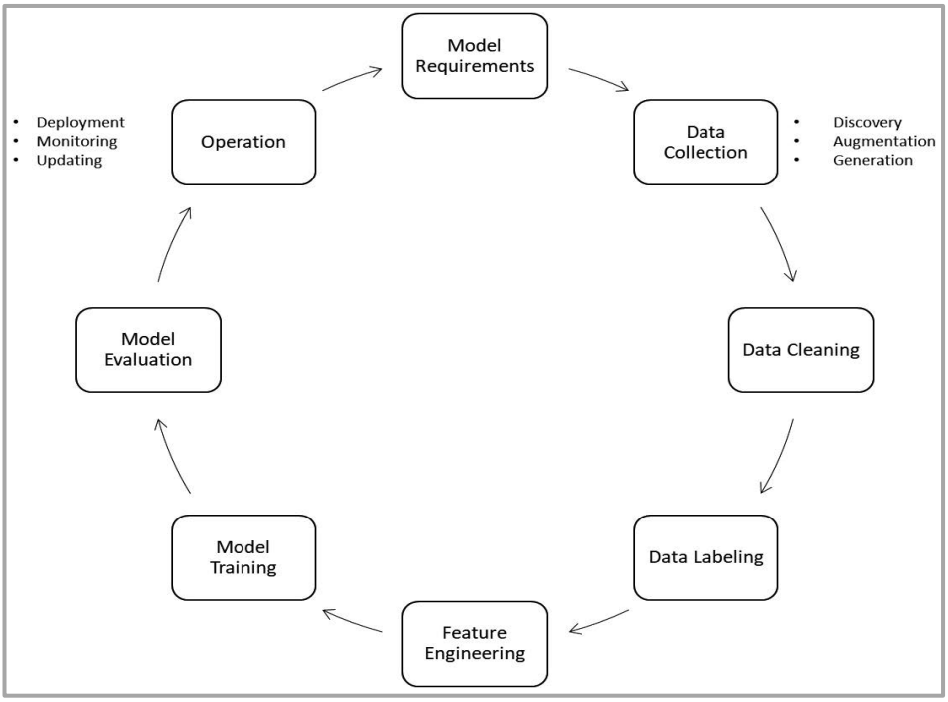
Google Colab adalah lingkungan notebook berbasis cloud yang menyediakan runtime siap pakai (termasuk GPU atau TPU) tanpa instalasi lokal, sehingga populer untuk eksperimen deep learning dan aktivitas perkuliahan atau penelitian. Analisis performa menunjukkan Colab efektif sebagai "accelerator" komputasi DL di ranah visi komputer, sementara berbagai jurnal menggunakan Colab sebagai platform utama untuk pelatihan CNN atau transfer learning (Carneiro dkk., 2018).

2.11.4. Visual Studio Code (VS Code)

VS Code berperan sebagai editor/IDE lintas-platform yang ringan namun kaya fitur (linting, debugging, kontrol versi, serta ekosistem ekstensi Python dan ML), sehingga sering dipakai pada tahap pengembangan aplikasi, integrasi, dan *model deployment*. Laporan pengalaman dan studi komparatif IDE menempatkan VS Code sebagai opsi praktis untuk pengembangan pemrograman/ML karena kombinasi ringan-fleksibel dan dukungan ekosistem, sedangkan pada riset terapan ia kerap diintegrasikan bersama Jupyter atau Streamlit untuk membangun prototipe antarmuka model (Brasoveanu dkk., 2020).

2.12. Machine Learning Life Cycle

Berikut adalah metode pengembangan perangkat lunak menggunakan *machine learning life cycle.*



Gambar 10. Machine Learning Life Cycle

Sumber: (Gärtler dkk., 2021)

Pada tahapan pengembangan model cnn menggunakan *machine laerning life cycle* yang memiliki beberapa tahapan antara lain (Gärtler dkk., 2021):

1. *Model Requirements*

   Tahap ini menetapkan tujuan fungsional sistem yakni klasifikasi lima kelas (empat spesies burung pemakan biji-bijian Lonchura leucogastroides, L. maja, L. punctulata, Passer montanus dan satu kelas unknown) beserta batasan non-fungsional seperti waktu latih/inferensi, jejak memori, dan target kinerja. Di sini pula dipilih paradigma transfer learning dengan EfficientNetB0 sebagai backbone karena rasio akurasi kompleksitasnya baik pada citra natural beresolusi 224×224×3. Hasil tahap ini menjadi kriteria penerimaan pada evaluasi berikutnya.

2. Data *Collection*

   Data citra dikimpulkan terutama dari eBird (empat spesies target) dan kelas unknown: burung non-pemakan biji yang kerap muncul pada tanaman padi. Data test di ambil secara langsung diladang padi. Penyaringan dilakukan agar

relevan secara geografis (prioritas Indonesia atau Asia) dan representatif terhadap variasi pencahayaan, latar, sudut pandang, serta perangkat. Secara praktis, dataset awal memuat ribuan citra dari kedua sumber ini, yang selanjutnya dikurasi.

3. Data *Cleaning*

Citra yang bermasalah (duplikat, korup, tidak relevan) dieliminasi. Seluruh gambar kemudian distandardisasi ke 224×224 piksel (RGB) dan di-normalisasi sesuai praproses EfficientNet. Untuk mengurangi bias kelas, dilakukan penyeimbangan (*downsampling*) sehingga setiap kelas memiliki ukuran yang sebanding mendorong pelatihan yang adil dan stabil.

4. Data *Labeling*

Pelabelan mengacu pada identitas spesies atau kategori yang telah diverifikasi saat kurasi, dan diorganisasikan pada struktur direktori per kelas (4 spesies ditambah 1 kelas unknown). Pedoman label di ambil dari studi litrature yang melakukan penelitian langsung di lapangan.

5. Feature Engineering

Walau CNN belajar fitur end-to-end, tahap ini mencakup augmentasi (rotasi kecil, pergeseran, zoom, perubahan kecerahan atau warna) khusus untuk data latih, serta penetapan skema split. Dalam penelitian ini digunakan skema 70:30 (latih:uji) dengan subset validasi dari data latih, dan metadata (kelas, asal, versi praproses) didokumentasikan agar reproduktif.

6. Model *Training*

Backbone EfficientNetB0 pralatih ImageNet digunakan sebagai feature extractor. Di atasnya ditambahkan head klasifikasi ringan (GlobalAveragePooling → Dropout → Dense sesuai jumlah kelas), lalu dilakukan feature extraction pada lapisan atas dengan learning rate kecil. label smoothing diterapkan untuk mengurangi efek ketidakseimbangan dan *over-confidence*. *Callbacks* menjaga generalisasi.

7. Model *Evaluation*

Kinerja diukur pada set uji 30% yang tidak dilihat saat latih, menggunakan Akurasi, Precision, Recall, F1-macro, serta confusion matrix per kelas. Untuk menilai kestabilan estimasi, digunakan Stratified k-fold cross-validation pada data latih atau validasi laporan hasil disajikan sebagai rata-rata simpangan baku. Praktik ini sejalan dengan anjuran evaluasi berulang yang mengurangi varians dan mendeteksi overfitting.

8. Operation

Dalam penelitian ini tidak di gunakan fase produksi seperti deployment dan pengetesan model secara langsung di lapangan.

2.13. Penelitian Terdahulu

Tabel 2. Penelitian Terdahulu

No	Penulis	Judul Penelitian	Tujuan Utama	Metodologi	Dataset	Hasil
1	Nwonye dkk., 2024	*Monitoring of Migration of Harmful Sparrow Birds in a Rice Farm Using Convolutional Neural Network (CNN) Model*	Memantau migrasi burung pipit (*sparrow*) perusak padi, membedakannya dari burung menguntungkan, & memberi tahu petani via SMS	EfficientNet-B5; pelatihan back-prop; implementasi Python (Colab) + Raspberry Pi 5	±85.000 citra (100 sawah, 84.900 internet); 224×224×3; basis data spesies (bird.csv)- Database spesies	Akurasi deteksi/klasifikasi 98%; akurasi latih/val 97%; uji 700 gambar: akurasi 97,2%, presisi 96,5%

					(bird.csv)	
2	Yuliany & Nur Rachman, 2022	Implementasi Deep Learning pada Sistem Klasifikasi Hama Tanaman Padi Menggunakan Metode Convolutional Neural Network (CNN)	Mengklasifikasikan jenis hama padi untuk penanganan tepat sasaran & mengatasi overfitting CNN	CNN custom (3 Conv + 2 FC); dropout, padding, augmentasi; Adam; uji split/parameter	1.065 citra sekunder (Google, Instagram); 5 kelas; 100×100 px	Val terbaik 83,02% (90:10, 100 epoch); Uji terbaik 77,33% (90:10, 200 epoch); overfitting; kelas "Ulat Grayak" buruk
3	Tenriola dkk., 2025	*Identifying Rice Plant Damage Due to Pest Attacks Using Convolutional Neural Networks*	Mengidentifikasi kerusakan spesifik (sundep & beluk) pada batang padi akibat penggerek batang,	Perbandingan 4 arsitektur: CNN Umum, CNN Usulan (filter 32–128, pool 2×2, dropout 50%),	Awal 700/kelas (Hama/Tidak) dari lapangan (batang dikupas) → seleksi	CNN Usulan 99,8% (14,19 s); EfficientNet-B0 99,6% (688 s); CNN Umum 96,1% (395 s); MobileNet 47,2% (6,6 s); error

			fokus pada fitur batang, & mengusulkan arsitektur CNN optimal	MobileNet, EfficientNet-B0; augmentasi rotasi; 100 epoch, batch 32- Training: 100 epoch, batch 32	500/300; augmentasi → 2000/kelas; ukuran bervariasi (216×288, 224×224, 100×100)	FP/FN dicatat
4	Raihan Maulana dkk., 2023	Implementasi Algoritma Convolutional Neural Network Dalam Mengklasifikasi Jenis Burung	Mengklasifikasikan jenis burung untuk memudahkan identifikasi oleh masyarakat umum & membangun model rujukan	- Algoritma CNN - Pelatihan 20 epoch	Subset Kaggle "Birds 525 Species"; 30 kelas; 5.050 citra (4.760 latih, 150 val, 150 uji); 224×224×3	Akurasi: latih 96,30%; val 81,33% (indikasi overfitting)

5	Dharaniya dkk., 2022	*Bird Species Identification Using Convolutional Neural Network*	Mengidentifikasi spesies burung dari gambar untuk konservasi & aplikasi potensial (pariwisata, pelacakan)	CNN standar (Conv2D, MaxPool, Flatten, Dense, Softmax); preprocessing smoothing; 50 epoch, batch 128; aplikasi web	Citra internet; 6 spesies; (jumlah/kelas tidak jelas)	Akurasi 87–92%; akurasi val ~0,9 di 50 epoch
6	(Rahmalia Syahputri dkk, 2025)	Implementasi Convolutional Neural Network Menggunakan Tensorflow untuk Mendeteksi Penyakit pada Daun Tanaman Melon	Membuat model Convolutional Neural Network (CNN) yang dapat mengklasifikasikan penyakit pada daun tanaman melon.	Menggunakan library TensorFlow, Arsitektur CNN terdiri dari 4 layer convolutional, 4 layer dropout, dan 1 layer neural network, Fungsi aktivasi yang digunakan adalah	250 citra observasi (Green House IIB Darmajaya & BPP Lampung); 3 penyakit + daun sehat + "unknown"	Akurasi 78%; fitur deteksi real-time di aplikasi Tani Cerdas berjalansecara real-time

				ReLU pada *hidden layer* dan Softmax pada *output layer*.		
7	(Song, 2024)	*Bird image classification based on improved ResNet-152*	Klasifikasi spesies burung & membandingkan performa dgn arsitektur populer	Transfer learning Improved ResNet-152; evaluasi vs ANN/CNN/ VGG-16	Kaggle "BIRDS 525 SPECIES" (multi-kelas)	Eval: 98% (cls), 97% (latih/valid), 97.2% (uji), 96.5% presisi. Implementasi: dipasang di sawah (kamera + Raspberry Pi 5) dan kirim SMS peringatan, membedakan burung perusak vs menguntungkan.
8	(Neeli dkk., 2023)	*Bird Species Detection Using CNN and Efficient Net-B0*	Membangun deteksi/klasifikasi spesies burung yang	EfficientNet-B0 (transfer learning) + CNN; fokus efisiensi parameter &	84.635 citra latih, 525 kategori burung (Kaggle	Eval: 83.02% (val), 77.33% (uji); kelas "Ulat Grayak"

			ringan & efisien	device-friendly	); setup Google Colab	lemah. Implementasi: studi eksperimental; rekomendasi mitigasi overfitting, belum ada integrasi lapangan.
9	(Rendi Pratama dkk., t.t.)	*The Application of CNN Using a Webcam to Analyze the Facial Expressions of Problematic Students in the Counseling Guidance Unit*	Deteksi ekspresi wajah siswa saat sesi BK (konseling)	CNN; 7 kelas emosi (Angry, Disgust, Fear, Happiness, Neutral, Sadness, Surprise); alur ML lifecycle; implementasi webcam	618 gambar; split 80:20 (latih:uji)	Akurasi uji ≈33%; masalah: kelas tidak seimbang & overfitting; direkomendasikan perlu tambah data/arsitektur lebih kuat & tuning.

10	(Sutedi dkk., 2023)	Brain Tumor Detection on MRI Using Deep Neural Network	Memperc epat & meningka tkan akurasi deteksi tumor otak berbasis citra MRI	Deep Neural Network (CNN layers); training-validation-testing; evaluasi (accuracy, sensitivity, specificity, precision, F1, DSC)	3000 citra MRI (Kaggle); split 70:20:1 0	Akurasi 98.3%, sensitivitas 98%, spesifisitas 98%, presisi 98%, F1 98.6%, DSC 98.6%; waktu prediksi ~0.2 s.

Penelitian-penelitian sebelumnya telah banyak memanfaatkan teknologi *computer vision* untuk berbagai aplikasi di bidang pertanian dan observasi satwa. Sebagai contoh, beberapa studi yang dirujuk oleh (Tenriola dkk., 2025) berhasil menerapkan CNN, Faster R-CNN, dan model hybrid untuk mendeteksi penyakit daun padi, mengidentifikasi tahap perkembangan malai (tandan bunga padi), atau mendiagnosis tingkat nutrisi tanaman padi. Sementara itu, dalam konteks pengenalan burung, penelitian yang dikutip oleh (Dharaniya dkk., 2022) dan (Raihan Maulana dkk., 2023) telah menggunakan berbagai pendekatan mulai dari SVM dengan fitur warna, CNN dengan arsitektur seperti LeNet, VGG-16, MobileNet, GoogleNet, ResNet, InceptionResNet-v2, hingga jaringan kolaboratif seperti CoCoNet.

Namun, banyak dari pendekatan ini memiliki keterbatasan untuk melakukan hal yang lebih spesifik yaitu klasifikasi. Seperti yang ditekankan oleh (Yuliany & Nur Rachman, 2022) dan (Tenriola dkk., 2025), fokus pada deteksi penyakit atau kerusakan umum mungkin kurang memadai karena tidak mengidentifikasi jenis hama spesifik yang menyebabkan masalah atau pola kerusakan unik pada bagian tanaman tertentu seperti batang padi. Lebih lanjut, dalam konteks pengendalian

hama burung di lahan pertanian, (Nwonye dkk., 2024) menyoroti bahwa sistem deteksi keberadaan burung saja tidak cukup; penting untuk mengklasifikasikan jenis burung tersebut karena tidak semua burung bersifat merusak—beberapa justru bermanfaat sebagai pemakan serangga. Selain itu, (Yuliany & Nur Rachman, 2022) juga mencatat bahwa beberapa implementasi CNN sebelumnya, seperti yang dilaporkan oleh (Yuliany & Nur Rachman, 2022), cenderung mengalami masalah *overfitting*, yang memerlukan perhatian pada arsitektur dan strategi pelatihan.

Penelitian (Rahmalia Syahputri dkk., t.t.) bertujuan utama untuk membangun model *Convolutional Neural Network* (CNN) yang mampu mengklasifikasikan tiga jenis penyakit utama pada daun tanaman melon (layu fusarium, ulat daun, dan kutu daun) serta daun sehat. Model ini kemudian diimplementasikan ke dalam aplikasi Tani Cerdas berbasis Android untuk membantu petani dalam mendeteksi penyakit secara dini

Oleh karena itu, melakukan klasifikasi yang lebih rinci baik itu jenis hama spesifik, tipe kerusakan batang, atau spesies burung tertentu menjadi krusial. Pendekatan ini memungkinkan pengembangan strategi pengelolaan hama atau pemantauan keanekaragaman hayati yang lebih presisi, efektif, dan sesuai dengan target spesifik, daripada hanya mengandalkan deteksi umum atau metode pengendalian yang bersifat pukul rata.