DAFTAR SINGKATAN

API	*Application Programing interface*
CNN	*Convolutional neural network*
OPT	Organisme Penganggu Tanaman
RNN	*Recurent Neural Network*
GAN	*Generative adversial network*
Ha	Hektare

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Indonesia sebagai negara agraris dan negara yang berada pada daerah tropis sangat cocok untuk lahan pertanian dan perkebunan. Tomat (*Lycopersicum esculentum*) sebagai salah satu produk tani di Indonesia yang memiliki nilai ekonomis yang cukup tinggi, selain untuk di konsumsi secara langsung dan menjadi bumbu masakan, tomat dapat di olah sebagai baku untuk saus dan sambal. Menurut badan statistika nasional, Indonesia memproduksi setidaknya 1.084.993 ton tomat di tahun 2020 sendiri [1], dengan lahan produksi seluas 57.304 Ha yang tersebar di berbagai provinsi di Indonesia[2].

Meskipun memiliki tingkat konsumsi dan produksi yang tinggi, hal tersebut tidak berarti tanaman tomat tidak menghadapi masalah. Penyakit dan hama yang menyerang tanaman dapat mengganggu hasil panen petani, bahkan menyebabkan kegagalan panen atau penurunan kualitas yang berdampak pada penurunan nilai jual. Organisme Penganggu Tanaman (OPT) adalah organisme mikro, hewan, dan gulma yang dapat mengganggu, menghambat, atau bahkan mematikan tanaman yang di inangi-nya[3]. OPT yang menyerang pada tanaman tomat salah satu nya seperti penyakit busuk daun (Phytophthora infestans), virus mosaic, early Light (Alternaria solani).

Kasus yang sering terjadi oleh para petani, banyak kerugian yang di akibatkan keterlambatan dalam mendiagnosis dan memberi solusi pada penyakit tanaman sejak dini, sampai sudah mencapai tahap yang parah hingga menyebabkan gagal panen [4]. Sebenarnya bahkan sebelum pada tahap yang lebih parah, tanaman yang terjangkit OPT dapat menunjukkan gejala-gejala yang di derita pada tahap yang ringan dan sedikit. Namun masih ada petani yang masih mengabaikan hal ini karena ketidaktahuan dan menanggap gejala tersebut biasa terjadi pada tanaman pada masa tanam. Hingga di saat timbul gejala yang parah dan meluas, sehingga terlambat untuk di kendalikan.

Pakar dan ahli pertanian memainkan peran penting dalam memberikan analisis terhadap gejala-gejala OPT yang menyerang tanaman, serta solusi dan sosialisasi mengenai penyakit yang terjadi pada tanaman tersebut. Namun, terdapat kendala ketika pakar dan ahli tidak dapat melayani petani secara simultan, yang dapat menyebabkan kerugian material bagi petani yang terlambat mengidentifikasi penyakit pada tanaman mereka. Disisi lain beberapa studi telah menunjukan implementasi teknologi *deep learning* pada deteksi penyakit tanaman[5], [6].

Penelitian ini berfokus pada pengembangan dan evaluasi sistem berbasis Convolutional Neural Network (CNN) mengunakan perbandingan model CNN pada *DenseNet121*, *MobileNet2*, dan *NasNetMobile* untuk mendiagnosis hama dan penyakit (OPT) yang menyerang tanaman tomat, dengan menggunakan analisis gambar daun tomat yang terserang penyakit. Penelitian ini mencoba untuk mengeksplorasi dan menevaluasi metode *transfer learning* lanjutan seperti *fine tuning*, lapisan *custom*, dan agumentasi data untuk meningkatkan kinerja model CNN dalam mengklasifikasi citra penyakit pada daun tomat.

Fine Tuning dalam *machine learning* model CNN mengacu pada proses memodifikasi lapisan pada jaringan CNN untuk mengadaptasi model pre *train*ed yang sudah dilatih dengan bobot sebelumnya pada tugas atau *dataset* baru[7]. Pada CNN *Fine Tuning* dapat dilakukan pada seluruh jaringan saraf, atau hanya pada sebagian lapisannya[8]. Dalam hal ini, lapisan yang tidak *fine tune* akan dibekukan "*frozen*" (tidak diperbarui selama langkah *back propagation*). Metode ini digunakan untuk mempertahankan fitur yang telah dipelajari dari model pra-latih, sehingga mencegah *overfitting*, mempercepat proses pelatihan, dan meningkatkan efisiensi komputasi.

Pembuatan sistem CNN yang mampu menganalisis Organisme Pengganggu Tanaman (OPT) pada daun tomat dapat menjadi acuan dalam membangun implementasi sistem penanggulangan yang dapat diakses melalui aplikasi web secara *online*. Hal ini memungkinkan petani untuk mengakses sistem ini kapan saja dan di mana saja. Penelitian ini diharapkan dapat memberikan kontribusi pada ruang lingkup yang lebih luas dari penerapan *deep learning* dalam bidang pertanian, khususnya dalam diagnosis dan pengelolaan penyakit pada tanaman tomat. Selain itu, penelitian

ini juga diharapkan dapat memberikan wawasan penting mengenai aplikasi dan dampak dari *fine-tuning* dalam meningkatkan kinerja model CNN.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah disampaikan sebelumnya, rumusan masalah yang ingin diajukan oleh penelitian ini adalah:

1. Bagaimana cara merancang sistem untuk mengklasifikasi penyakit pada daun tomat dengan mengunakan *Framework*(CNN)?
2. Bagaimana cara menentukan model arsitektur dan *hyperparameter* terbaik untuk mengklasifikasikan citra penyakit pada daun tomat?
3. Apa efek dari penerapan metode *transfer learning* lanjutan seperti *fine tuning*, augmentasi data, dan lapisan *custom* terhadap kinerja model CNN?

1.3. Batasan Masalah

Pada penelitian ini digunakan asumsi dan batasan masalah sebagai berikut:

1. Penelitian ini berfokus pada perancangan sistem deteksi penyakit tanaman tomat, mengunakan pendekatan metode *fine-tune* guna meningkatkan performa model.
2. *Dataset* yang digunakan merupakan *dataset* "*Village plant*" yang merupakan second hand *dataset* yang di kurasi oleh *Mendeley* dan bersifat publik.
3. Arsitektur model CNN (*Convolutional Neural Network*) yang digunakan sebagai perbandingan adalah *DenseNet121, MobileNetV2*, dan *NasNetMobile*.
4. Penulis menggunakan algoritma CNN (*Convolutional Neural Network*) dan bahasa pemograman Python.

1.4. Tujuan dan Manfaat Penelitian

Tujuan dari riset ini, selain menjadi tugas akhir bagi penulis, adalah untuk mencapai beberapa tujuan dalam pembuatan sistem deteksi penyakit pada daun tomat berbasis *deep learning*. Tujuan tersebut meliputi:

1. Mampu mengidentifikasi penyakit pada tanaman tomat dengan menggunakan metode *deep learning* dan mencapai tingkat akurasi yang tinggi.

2. Meningkatkan tingkat *recall* dan presisi sistem di atas 85% guna menghindari kesalahan klasifikasi.
3. Mempelajari dan menganalisis efek pada penerapan metode *Fine Tuning* terhadap model model CNN.

Adapun manfaat dari penelitian ini diharapkan dapat memberikan solusi dalam masalah :

1. Memberikan wawasan mengenai implementasi *Fine Tuning* pada model *Convolutional Neural Network* (CNN), khususnya untuk diagnosis dan pengelolaan penyakit pada tanaman tomat.
2. Inisiatif digitalisasi sistem layanan terpadu bagi petani.
3. Membuat sistem yang dapat diperluas dan dikembangkan oleh lembaga terkait.

1.5. Metode Penelitian

Metode yang di gunakan untuk menyelesaikan penulisan Tugas Akhir ini adalah sebagai berikut :

1. Studi Literatur

   Pengumpulan materi dan referensi berupa jurnal, artikel, paper, slide materi perkuliahan, dan lain-lainnya yang berkaitan dengan penelitian tugas akhir ini.

2. Studi Lapangan

   Melakukan diskusi atau bimbingan bersama dosen pembimbing dan ahli dalam bidang machine learning dan penyakit dan hama pada tanaman tomat yang dapat memberikan masukan serta arahan untuk penelitian tugas akhir ini.

3. Perancangan dan Realisasi Sistem

   Merancang desain kerja aplikasi ini pada petani dan diterapkan di perkebunan.

4. Analisis Kerja Sistem

   Menganalisis hasil dari kedua metode yang di pakai untuk mengetahui hasil terbaik.

5. Kesimpulan

Menyimpulkan hasil akhir dari penelitian yang sudah dilakukan berdasarkan Aplikasi Identifikasi penyakit pada tanaman tomat menggunakan *machine learnning.*

1.6. Jadwal Pelaksanaan

Adapun jadwal pelaksanaan untuk penelitian tugas akhir ini adalah sebagai berikut :

Tabel 1. 1 Jadwal Pelaksanaan.

No.	Deskripsi	Durasi	Tanggal Selesai	Milestone
1	Studi Literatur	2 minggu		Memahami konsep yang akan digunakan
2	Desain Sistem	4 minggu		Merancang desain sistem
3	Implementasi, Konfigurasi dan integrasi model kedalam Aplikasi	2minggu		Prototype 1selesai
4	Penyusunan laporan/buku TA	3 Minggu		Buku TA selesai

BAB II
TINJAUAN PUSTAKA

2.1. PENYAKIT TANAMAN TOMAT

Tomat (*Lycopersicum esculentum*) adalah tumbuhan musiman dari keluarga Solanaceae, tumbuhan asli Amerika Tengah dan Selatan, dari Meksiko sampai Peru. Tumbuhan tomat memiliki buah berwarna hijau, kuning, dan merah yang biasa dipakai sebagai sayur dalam masakan atau dimakan secara langsung tanpa diproses.

Organisme Penganggu Tanaman (OPT) adalah organisme mikro, hewan dan gulma gulma yang dapat mengganggu, menghambat, atau bahkan mematikan tanaman yang di inangi-nya [3]. pada tanaman tomat sendiri memiliki OPT yang bervarietas, dan gejala nya masing-masing, pada penelitian ini di batasi OPT tanaman tomat yang dapat dilihat dan diklasifikasi berdasarkan gejala yang menyerang daun nya.

Adapun Organisme Pengganggu Tanaman (OPT) yang penulis coba klasifikasi di antaranya:

2.1.1. Hawar Awal ("*Early Bligth*")

Penyakit hawar awal, atau yang dikenal juga sebagai early blight, disebabkan oleh patogen bernama Alternaria solani. Akibatnya, pertumbuhan tanaman tomat terhambat dan mengakibatkan hasil buah yang kecil dan membusuk. Jika tidak ditangani dengan baik, kondisi ini dapat berujung pada kematian tanaman tomat[9].

2.1.2. Hawar Daun ("*Late Blight*")

Hawar daun di akibatkan oleh mikroorganisme jamur *miselium phytophthora infestans*[10]. Selain menyerang tanaman tomat hawar daun bisa menyerang tanaman kentang dan terung. Pada tanaman tomat sendiri hawar daun bisa di identifikasi dengan tumbuhnya bercak hitam atau ke-unguan yang timbul pada anak daun, tangkai, atau batang. Pada keadaan kelembaban yang tinggi bercak tersebut akan menyebar lebih cepat meluas dan menyebar ke tanaman di sekitarnya, mengakibatkan kematian tanamankan secara masih yang dapat mengakibatkan gagal panen.

2.1.3. Leaf Mold

Penyakit bercak daun atau *Leaf Mold* di sebabkan oleh mikroorganisme jamur *Cladosporium fulfum*, yang banyak di derita oleh tanaman tomat yang di budaya di rumah kaca dan daerah yang lembab[11]. Penyakit bercak daun pada umumnya hanya menyarang daun tanaman tomat dengan menimbulkan bercak abu-abu keunguan, dan menyerang mulai dari daun tomat yang tua berlanjut ke daun yang muda, menimbulkan bunga tanaman tomat mati sebelum menjadi buah.

2.1.4. Baterial Spot

Bacterial spot pada tanaman tomat di bisa akibatkan oleh bakteri *Xanthomonas vesicatoria*, *Xanthomonas euvesicatoria*, *Xanthomonas gardneri*, dan *Xanthomonas perforans*, patogen bakteri ini bisa menyerang tanaman tomat dikarenakan bibit atau cangkokan yang terkontaminasi bakteri ini [12]. *Bacterial spot* bisa menyerang daun, batang, dan buah tanaman tomat, adapun gejala yang terlihat dari infeksi *Bacterial spot* bahwa daun tanamanya akan tumbuh bercak lingkaran yang berwarna kuning kehijauan, setelah lama kemudian akan mengelap.

2.1.5. Septoria Leaf Spot

Septoria leaf spot di akibatkan oleh infeksi jamur *Septoria lycopersici*, umum menyerang tanaman tomat pada cuaca lembab dan musim peng-hujan. Septoria leaf spot bisa di identifikasi dengan melihat gejala bercak kecil berukuran 0.2 cm sampai 0.5 cm pada daun yang umumnya berwarna kecokelatan, *Septoria leaf spot* termasuk susah di identifikasi karena umumnya gejala yang muncul menjelang tanaman mulai berbuah [9].

2.1.6. Tomato Mossaic Virus

Virus mosaik tomat di akibatkan dari satu virus atau gabungan virus di antara *Cucumber Mosaic Virus* (CMV*)*, *Potato Virus Y* (PVY), *Tobaco Mosaic Virus* (TMV)[13]. Tanaman tomat dapat tertular virus mosaic dari inang tanaman sayuran di dekatnya, gejala tanaman tomat yang terjangkit virus mosaik bervariasi tergantung

jenis virus yang menjangkitnya, pada virus mosaik CMV daun pada tanaman tomat akan menguning dan tumbuh dengan ukuran kerdil.

2.1.7. Yellow Leaf Curl

Yellow leaf curl diakibatkan oleh virus TYLCV atau *Tomato Yellow Leaf Curl Virus*, gejala yang di alami tanaman tomat yang terjangkit virus ini antara lain tanaman tumbuh kerdil, dengan arah cabang dan tangkai daun yang cenderung tegak, daun muda tanamannya tumbuh kecil dan mengerut serta cekung, ditambah pinggiran daun dengan atau tanpa warna kuning [14].

2.2. CITRA DIGITAL

Citra digital merupakan gambar 2 dimensi yang terdiri dari piksel yang di susun dalam pola M baris dan N kolom pada peta *grid* [15]. piksel adalah representasi sebuah titik terkecil dalam sebuah gambar, yang dimana setiap piksel memiliki nilai warna pada citra warna, atau tingkat ke-abuan pada citra abu-abu (*grayscale*). Dengan disusun dalam peta grid, sebuah piksel memiliki dua parameter, yaitu koordinat dalam *grid* dan nilai warna atau intensitas. Oleh karena itu, citra dapat dituliskan ke dalam sebuah matriks :

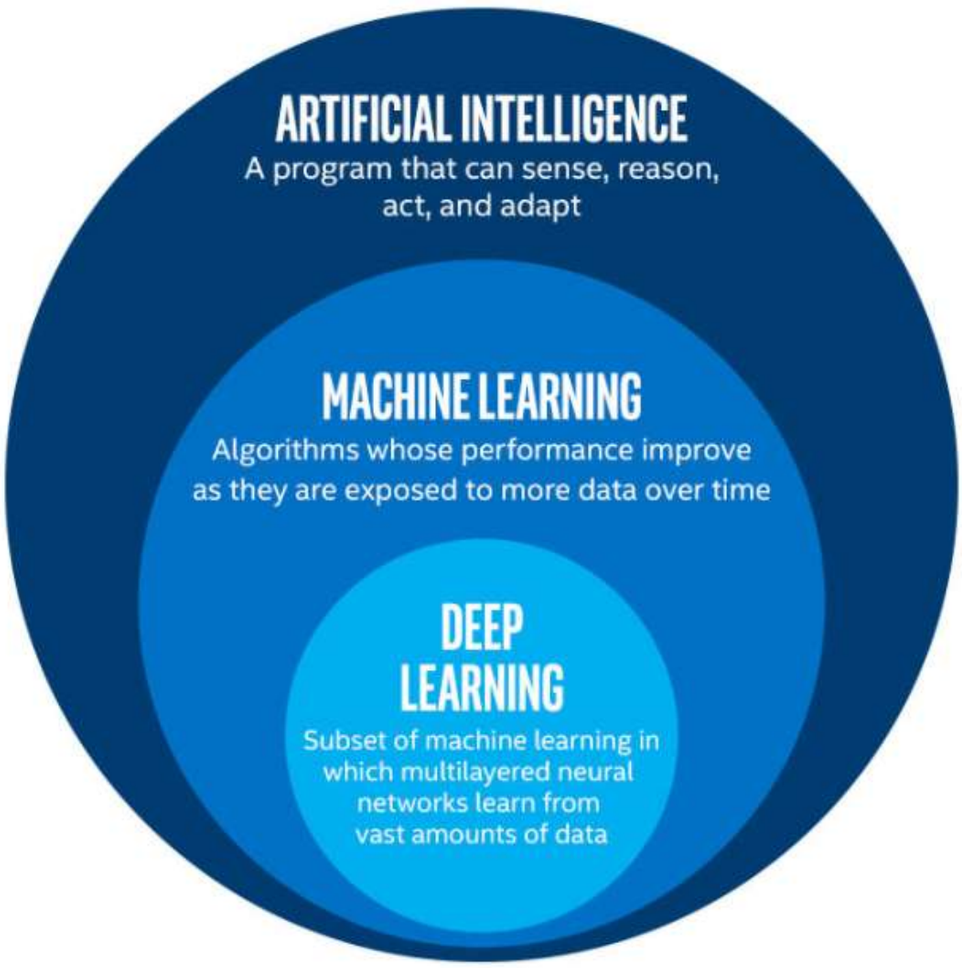
$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \cdots & f(1,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix} \quad 2.4.1.$$

Nilai (x,y) adalah koordinat, dan nilai f dalam f(x,y) adalah nilai warna atau tingkat ke-abuan dari piksel di titik tersebut.

2.3. MACHINE LEARNING

Pada Gambar 2. 1, menunjukan *Machine learning* yang merupakan sub cabang ilmu kecerdasan buatan atau *artificial intelligence* (AI) yang memungkinkan komputer memiliki keterampilan untuk dapat belajar tanpa perlu di program lebih lanjut. Sebuah model *machine learning* dapat dibangun dengan algoritma sehingga memungkinkan model *machine learning* tersebut dapat terus belajar dan bertugas sendiri tanpa ada

instruksi lanjutan dari penggunanya [16]. Algoritma *machine learning* bekerja dengan cara memproses *input* atau masukan data ke dalam sebuah model pembelajaran, yang nantinya menghasilkan suatu prediksi atau pengambilan keputusan dari model yang di pelajari.



Gambar 2. 1 Diagram sederhana lingkup studi AI

Dalam pembelajaran *machine learning*, terdapat tiga skenario utama yaitu:

a. *Supervised learning*, Pada kategori ini, data yang dimiliki dilengkapi dengan label/kelas yang menunjukkan klasifikasi atau kelompok data tersebut berada [15].

b. *Unsupervised learning*, Pada kategori ini, data pembelajaran tidak dimiliki label/kelas sehingga harus mencari struktur dari data yang ada, kemudian melakukan pengelompokan berdasarkan informasi yang dimiliki [15].

c. *Reinforcement learning*, atau pembelajaran penguatan yaitu pembelajaran dilakukan untuk mendapatkan sesuatu yang maksimal dan meminimalkan resiko [15].

2.3.1. DEEP LEARNING

Didalam bidang *machine learning, deep learning* merupakan sub-bidang yang berfokus pada representatif algoritma sebagai sebuah lapisan jaringan saraf layaknya otak manusia. Jaringan saraf dalam *deep neural network* terdiri dari tiga atau lebih lapisan, yang bekerja untuk memahami dan mengekstrak fitur-fitur dari data input. Hal ini memungkinkan jaringan untuk membuat prediksi atau klasifikasi dengan tingkat akurasi yang tinggi[17].

Kata "*deep*" atau dalam sendiri mengacu pada jumlah lapisan pada jaringan saraf, sebagaimana semangkin banyak jumlah lapisan pada jaringan saraf, semangkin dalam (*deep*) jaringan tersebut. *Deep neural network* terdiri dari banyak lapisan neuron yang memungkinkan mereka mempelajari representasi hierarkis data. Setiap lapisan akan mengekstraksi fitur yang semakin kompleks dari data *input*, dimulai dengan fitur sederhana pada lapisan pertama, dan membangun konsep yang lebih abstrak dan komplek pada lapisan-lapisan berikutnya. Dengan menggunakan fitur-fitur ini, *deep learning* telah mencapai kesuksesan besar dalam berbagai aplikasi tugas seperti pengenalan citra, pengenalan ucapan, dan pemrosesan bahasa alami [18].

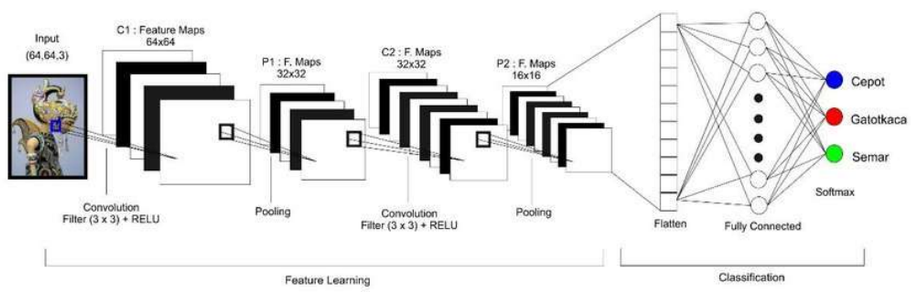
Ada berbagai pendekatan dalam mengunakan *deep learning* untuk menyelesaikan tugas, masing-masing dengan kekuatan dan kelemahannya tersendiri. Beberapa pendekatan yang paling umum termasuk:

1. *Recurrent neural network* (RNN): RNN biasa digunakan untuk mengeleola tugas pemprosesan bahasa alami atau NLP. RNN dirancang untuk menangani data berurutan dengan mempertahankan keadaan internal yang memungkinkan RNN dapat mengingat informasi dari *input* sebelumnya melalui fungsi *back propagation* [18].

2. *Generative adversarial network* (GAN): GAN merupakan salah satu tipe dari *deep neural network* yang terdiri dari dua bagian: sebuah generator dan *discriminator*, Generator belajar membuat data baru yang mirip dengan data pelatihan, sedangkan *diskriminator* belajar membedakan antara data asli dan palsu. Generator dan *discriminator* tersebut dilatih bersama, dengan generator mencoba mengelabui *diskriminator* dan *diskriminator* mencoba mengklasifikasikan data dengan benar [18].
3. *Convolutional neural network* (CNN): CNN biasa di gunakan untuk tugas pengenalan citra. CNN terdiri dari beberapa lapisan filter konvolusi yang belajar untuk mengekstrasi fitur dari gambar masukan[18].

2.4. *CONVULITIONAL NEURAL NETWORK* (CNN)

Convolutional Neural Network (CNN) merupakan jenis algoritma *neural network* dalam mempelajari hierarki fitur kompleks dengan *convolution layer, activation layer* dan *pooling layer*. *Convolutional Neural Network* merupakan *special case* dari *aritifical neural network* (ANN) yang saat ini kenal sebagai model terbaik untuk memecahkan masalah *object recoginition* dan *detection*. Kelebihannya yaitu bahwa *input* jaringan akan menjadi jelas ketika *input* dan gambar multidimensi dapat langsung masuk ke dalam jaringan, sehingga dapat menghindari penggalian fitur yang kompleks dan rekonstruksi data.



Gambar 2. 2 Arsitektur CNN

CNN terdiri dari lapisan masukan, lapisan keluaran dan sejumlah lapisan tersembunyi. Pada Gambar 2. 2, dapat dilihat arsitektur CNN yang terbagi menjadi dua bagian utama, yaitu bagian pembelajaran fitur (feature learning) dan klasifikasi

(classification). Secara sederhana, fungsi dari setiap jenis lapisan dalam arsitektur CNN tersebut dapat dijelaskan sebagai berikut:

2.4.1. Lapisan Konvolusi

Lapisan konvolusi menentukan *output* neuron yang terhubung dengan daerah lokal *input* melalui perkalian skalar antara weights dan daerah yang terhubung dengan volume *input* [19]. Pemberian nama *convolutional neural network* mengindikasikan bahwa jaringan tersebut menggunakan operasi matematika yang disebut konvolusi. Di sisi lain, konvolusi merupakan operasi dua buah fungsi yang khusus untuk memproses data berbentuk matriks.

Operasi konvolusi melibatkan filter (juga dikenal sebagai *kernel*) yang digeser melintasi data input (seperti gambar) dan melakukan operasi dot untuk menghasilkan peta fitur. Operasi ini membantu dalam mendeteksi pola, seperti tepi, sudut, atau tekstur, dalam data masukan [20]. Sebagai contoh, diberikan sebuah matriks input X dengan ukuran (W, H, D). Di mana W dan H adalah lebar dan tinggi, dan D adalah kedalaman saluran atau *Channel*, serta filter berukuran (F,F,D), operasi konvolusi dapat direpresentasikan sebagai:

$$Output[i,j,k] = \sum_{m=0}^{F-1} \sum_{n=0}^{F-1} \sum_{p=0}^{D-1} X[i+m, j+n, p] \times Filter[m,n,p] \qquad 2.4.1.$$

Berikut ini penjelasan singkat mengenai notasi yang digunakan:

$Output[i,j,k]$ = nilai pada posisi (i,j) pada ke-k dari peta fitur output.

$X[i+m, j+n, p]$ = Nilai pada posisi $(i\ +\ m, j\ +\ n, p)$ dalam matriks input.

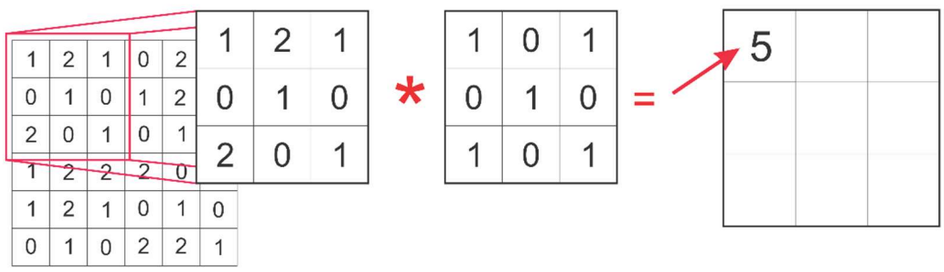
$Filter[m,n,p]$ = Nilai pada posisi (m,n,p) dalam filter.

F = Ukuran spasial filter (diasumsikan berbentuk persegi).

D = Kedalaman input, sesuai dengan jumlah saluran (misalnya, 3 untuk gambar RGB).

Representasi operasi konvolusi dapat dilihat pada Gambar 2. 3, di mana filter dengan ukuran 3 x 3 bergeser sambil melakukan operasi dot pada data input dengan ukuran 6 x 6 untuk menghasilkan peta fitur. Operasi dot dilakukan dengan melakukan perkalian

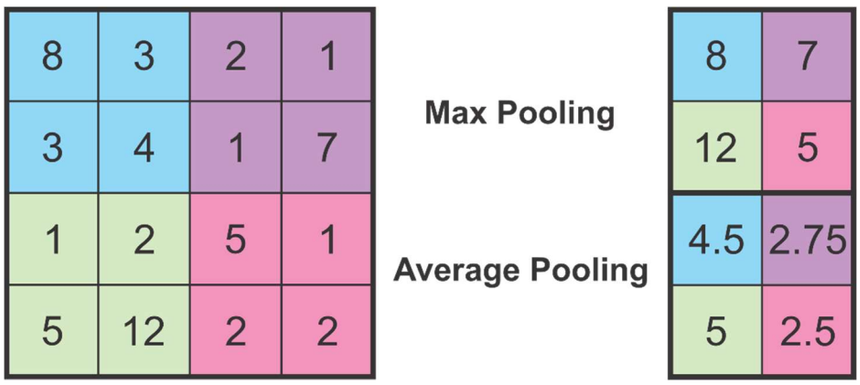
matriks pada baris dan kolom yang sama pada data input dan filter, lalu menjumlahkan hasilnya untuk menjadi satu nilai representatif pada peta fitur. Setelah operasi dot dilakukan, maka operasi berlanjut dengan bergeser sebesar 1 stride untuk melakukan operasi dot hingga seluruh matriks 6x6 selesai diolah.



Gambar 2. 3 *Proses operasi konvolusi pada Convolutional Layer*

2.4.2. Lapisan *Pooling*

Pooling layer melakukan *downsampling* terhadap dimensi spasial dari input peta fitur yang diberikan, kemudian mengurangi jumlah parameter dalam aktivasi tersebut sambil tetap mempertahankan informasi yang paling penting. Hal ini membantu dalam membuat deteksi fitur tidak berubah-ubah terhadap perubahan skala dan orientasi. Selain itu, metode ini mengurangi kompleksitas komputasi dengan menghilangkan redundansi dari peta fitur [19].

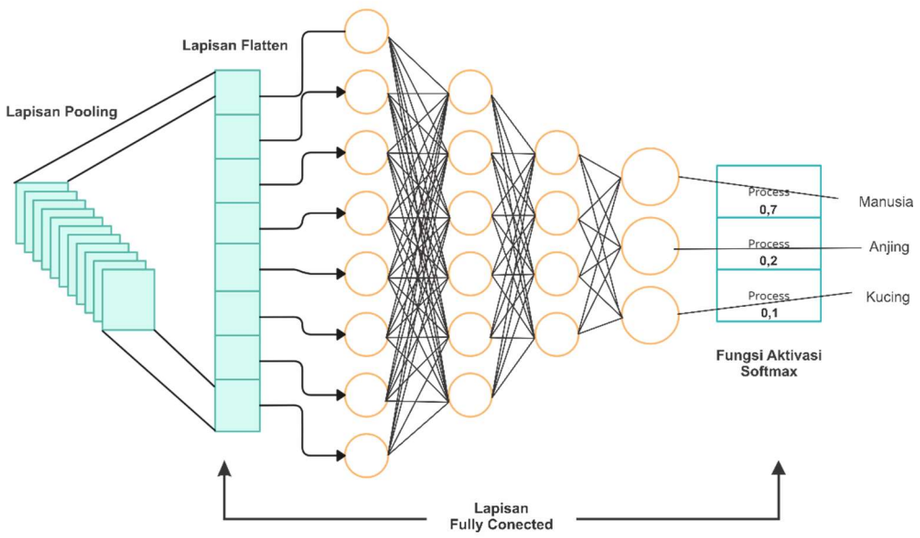


Gambar 2. 4 Ilustrasi Operasi Max Pooling dan Average Pooling

Terdapat dua jenis lapisan *pooling* yang sering digunakan, yaitu *max pooling* dan *average pooling*. Jenis *pooling* yang paling umum adalah *max pooling*. Pada *max pooling*, nilai maksimum diambil dari sekumpulan nilai di dalam filter atau *kernel* berukuran 2x2, dengan stride 2. Sedangkan pada *average pooling*, nilai rata-rata diambil dari sekumpulan nilai di dalam filter atau *kernel* tersebut [21]. Gambar 2. 4 merupakan ilustrasi perbandingan operasi *max pooling* dan *average pooling.*

2.4.3. Lapisan *Fully-connected*

Lapisan *neural fully connected*, juga dikenal sebagai lapisan *dense*, merupakan bagian terakhir dari struktur *convolutional neural network*. Lapisan ini beroperasi setelah menerima vektor hasil dari lapisan *pooling* yang telah diratakan dimensinya menggunakan teknik *flattening* atau *global average pooling*. Hasil ini kemudian dimasukkan ke dalam satu atau beberapa lapisan yang terhubung sepenuhnya (*fully connected*).



Gambar 2. 5 Ilustrasi jaringan lapisan fully conected

Setiap neuron pada lapisan *fully connected* menerima input dari setiap neuron pada lapisan sebelumnya. Koneksi ini memungkinkan lapisan yang terhubung sepenuhnya untuk mempelajari hubungan yang kompleks antara fitur-fitur dan membuat keputusan pada tingkat tinggi berdasarkan fitur-fitur yang telah diekstraksi.

Hal ini menghasilkan keluaran berupa distribusi probabilitas yang merepresentasikan kelas keluaran.

Fungsi aktivasi merupakan elemen integral dalam lapisan *neural fully connected*. Fungsi ini memainkan peran penting dalam membentuk kemampuan jaringan untuk belajar dan merepresentasikan hubungan yang kompleks dalam data. Sebagai komponen intrinsik dari setiap neuron, fungsi aktivasi memperkenalkan non-linearitas pada perhitungan jaringan. Non-linearitas ini sangat penting untuk menangkap pola rumit yang melampaui transformasi linear, memungkinkan jaringan untuk memodelkan dan memahami aspek nonlinear dari data.

Dalam konteks lapisan *fully connected*, fungsi aktivasi umumnya digunakan untuk meningkatkan kinerja klasifikasi dengan bertindak sebagai pengambil keputusan. Keluaran dari setiap neuron ditentukan tidak hanya oleh jumlah bobot masukan, tetapi juga oleh fungsi aktivasi yang diterapkan pada jumlah tersebut. Fungsi aktivasi memperkenalkan batasan, yang memungkinkan neuron untuk merespons pola input secara selektif. Responsivitas selektif ini memungkinkan jaringan untuk menekankan fitur-fitur yang signifikan, mengurangi noise yang tidak relevan, dan membuat keputusan yang kompleks berdasarkan representasi yang telah dipelajari [22]. Berikut ini adalah beberapa fungsi aktivasi yang syaraf umum digunakan dalam jaringan neural:

a. ReLU (*Rectified Linear Unit*):

   ReLU adalah fungsi aktivasi yang sederhana dan banyak digunakan yang menghasilkan input jika positif dan nol jika tidak [23]. ReLU memperkenalkan sparsitas dalam jaringan dengan meniadakan nilai negatif, membuat pelatihan lebih cepat dan membantu mengatasi masalah gradien yang hilang. Representasi matematis dari relu adalah:

$$f(x)_{ReLU} = \max(0, x) \qquad 2.4.2.$$

   Dimana nilai x merupakan input, dan menghasilkan output berupa nilai positif bila nilai input tersebut bernilai positif, dan bernilai 0 bila input bernilai negatif.

b. Softmax

Softmax mengubah vektor skor mentah menjadi distribusi probabilitas atas beberapa kelas. Fungsi sofmax sering digunakan di lapisan output untuk tugas klasifikasi untuk menghasilkan probabilitas kelas.

$$\sigma(\vec{Z})_i = \frac{e^{Z_i}}{\sum_{j=1}^{K} e^{Z_j}} \qquad 2.4.3.$$

Dimana:

σ = Keluaran Softmax

$\vec{Z} =$ Input vektor

$K =$ Jumlah kelas dalam pengklasifikasi multi-kelas

$e^{Z_i} =$ Fungsi eksponensial standar untuk vektor input

$e^{Z_j} =$ Fungsi eksponensial standar untuk vektor output.

2.5. ARSITEKTUR JARINGAN SARAF

Arsitektur atau *framework* jaringan saraf merujuk pada tata letak dan konstruksi komprehensif dari jaringan saraf yang digunakan dalam klasifikasi gambar. Ini mencakup susunan dan jumlah lapisan, jumlah neuron di setiap lapisan, jenis fungsi aktivasi yang digunakan, dan metode pelatihan jaringan. Semua elemen ini bersama-sama membentuk arsitektur jaringan saraf, memainkan peran krusial dalam kemampuan jaringan untuk melakukan klasifikasi dengan akurasi yang diharapkan.

Convolutional Neural Network (CNN) adalah arsitektur jaringan saraf yang dominan digunakan untuk tugas klasifikasi gambar yang juga merupakan jenis jaringan saraf *feedforward*, dimana informasi mengalir hanya dalam satu arah, dari lapisan masukan melalui satu atau lebih lapisan tersembunyi ke lapisan keluaran. Arsitektur tersebut dirancang untuk secara efektif memproses dan mengkategorikan gambar dengan menggunakan lapisan konvolusional khusus yang dapat mengambil fitur dari gambar, yang diikuti oleh lapisan terhubung penuh untuk klasifikasi. Memahami potensi dan kekurangan berbagai struktur jaringan saraf dapat membantu peneliti dalam memilih struktur yang sesuai untuk tugas klasifikasi gambar tertentu.

Adapun arsitektur jaringan saraf yang penulis latih pada tugas akhir ini di antaranya:

2.4.1. *DenseNet*

Dense Convolutional Network atau di singkat *DenseNet*, merupakan salah satu model arsitektur jaringan pada bidang visi kompututer yang di perkenal oleh Huang et al pada publikasi ilmiah tahun 2017[24]. *DenseNet* memanfaatkan koneksi rapat antar lapisan. Dalam *DenseNet*, semua lapisan dengan ukuran peta fitur yang sesuai terhubung langsung satu sama lain, dengan setiap lapisan menerima *input* tambahan dari semua lapisan sebelumnya dan meneruskan peta fiturnya ke semua lapisan berikutnya.

DenseNet memiliki keunggulan utama dalam menyebarkan fitur secara efisien. Setiap lapisan secara langsung menggunakan gradien dari fungsi *loss* dan sinyal *input* awal, yang menghasilkan bentuk pengawasan mendalam yang terintegrasi. Pendekatan ini juga mendorong penggunaan ulang fitur, yang mengarah ke jaringan yang lebih efisien dan lebih sedikit parameter. Selain itu, *DenseNet* mengatasi masalah *vanishing-gradient*, memperkuat penyebaran fitur, mendorong penggunaan kembali fitur, dan secara signifikan mengurangi jumlah parameter. Hal ini membuat jaringan lebih efisien dan lebih mudah dilatih.

2.4.2. MobileNetV2

MobileNetV2 adalah arsitektur jaringan saraf konvolusional yang dirancang untuk klasifikasi gambar yang efisien pada perangkat seluler dan sistem *microcontroller* [25]. *MobileNetV2* merupakan peningkatan dari arsitektur *MobileNet* asli, yang juga bertujuan untuk mengurangi kompleksitas komputasi jaringan saraf untuk pengunaan perangkat seluler. Untuk mengurangi jumlah parameter dan meningkatkan efisiensi jaringan, *MobileNetV2* menggunakan teknik yang disebut "*bottleneck linier*". *Bottleneck linier* menggantikan fungsi aktivasi *non-linier* pada lapisan bottleneck dengan fungsi linier, yang mengurangi biaya komputasi lapisan tersebut.

Framework dari *MobileNetV2* terdiri dari urutan lapisan konvolusi *depthwise separable* dan blok *residual* terbalik. Konvolusi *depthwise separable* digunakan untuk mengurangi kompleksitas komputasi jaringan dengan memfaktorkan operasi konvolusi menjadi konvolusi deepwise diikuti oleh konvolusi *pointwise*. Blok *residual* terbalik digunakan untuk meningkatkan kapasitas jaringan sambil tetap menjaga jumlah parameter dan kompleksitas komputasi rendah.

2.4.3. NasNet

NasNet mengambil inspirasi dari metode *Neural Architecture Search* untuk memecahkan masalah optimalisasi konfigurasi arsitektur. Model *NasNet* terdiri dari susunan yang disebut 'sel', di mana setiap 'sel' merupakan jaringan kecil. 'Sel' ini ditemukan melalui proses *Neural Architecture Search* (NAS) dan dikategorikan menjadi dua tipe: sel normal dan sel reduksi, yang merupakan sel konvolusi yang mengembalikan peta fitur dengan dimensi yang sama dan peta fitur yang tinggi dan lebarnya masing-masing dikurangi dengan faktor dua[26].

Masing masing sel pada arsitektur *NasNet* memiliki tugas sebagai berikut;

- Sel Normal: Sel ini mempertahankan ukuran peta fitur yang sama. Sel ini mengambil dua peta fitur sebelumnya sebagai *input* dan menghasilkan satu peta fitur dengan ukuran yang sama. Struktur sel normal diulang beberapa kali (ditumpuk) dalam satu tahap sebelum ukuran peta fitur berubah[27].
- Sel Reduksi: Sel ini digunakan untuk mengurangi ukuran setengah dari peta fitur. Sel reduksi digunakan apabila ukuran peta fitur perlu diubah. Sel reduksi akan mengurangi 2 piksel sebelum melakukan operasi konvolusi, yang mengurangi dimensi spasial peta fitur. Seperti sel normal, sel reduksi juga mengambil dua peta fitur sebelumnya sebagai *input*, tetapi menghasilkan satu peta fitur dengan ukuran yang diperkecil[27].

Setiap sel terdiri dari beberapa blok, dan rediksi dari pengontrol untuk setiap sel dikelompokkan ke dalam B blok, di mana setiap blok memiliki 5 langkah prediksi yang dibuat oleh 5 pengklasifikasi *softmax* yang berbeda yang sesuai dengan pilihan diskrit dari elemen-elemen blok. Setiap jalur menerapkan serangkaian operasi pada

*input*nya. Operasi ini mencakup berbagai jenis konvolusi (misalnya, 3x3 atau 5x5), berbagai ukuran max pooling, dan lain-lain.

2.6. *TRANSFER LEARNING*

Transfer learning merupakan metode dalam *machine learning* di mana model yang telah dilatih sebelumnya pada *dataset* besar disesuaikan atau dilatih ulang pada *dataset* dengan tugas yang sama yang lebih spesifik dan kecil untuk melakukan tugas baru yang terkait[7]. Metode ini memanfaatkan pengetahuan dan bobot yang telah diperoleh dari tugas sebelumnya untuk mempercepat proses pelatihan model dan meningkatkan performa pada tugas yang baru. Dengan menggunakan informasi yang sudah ada, model dapat lebih cepat beradaptasi dengan tugas baru dan menghasilkan hasil yang lebih baik.

Pendekatan *transfer learning* yang digunakan dalam tugas akhir ini berfokus pada menjaga bobot jaringan pada model CNN pra-latih dan memanfaatkan lapisan baru yang ditambahkan setelah lapisan asli dari model pra-latih untuk menginterpretasikan keluaran. Pendekatan ini memperlakukan *transfer learning* sebagai skema ekstraksi fitur, di mana lapisan yang sudah ada mengekstraksi fitur yang relevan sementara lapisan baru belajar untuk memahaminya.

Berikut adalah beberapa metode lanjutan *transfer learning* yang digunakan dalam tugas akhir ini:

2.6.1. AUGMENTASI DATA

Pada pipeline *transfer learning*, teknik augmentasi data mengacu pada modifikasi atau duplikasi data asli menggunakan berbagai teknik augmentasi. Augmentasi data adalah proses meningkatkan jumlah *dataset* dengan menciptakan data baru melalui penerapan transformasi yang berbeda pada data yang ada[28]. Augmentasi data dapat membantu meningkatkan kinerja model, terutama ketika *dataset* asli memiliki ukuran kecil atau tidak seimbang.

Berikut ini adalah beberapa teknik augmentasi data yang paling umum digunakan untuk gambar dalam visualisasi komputer: