

**SKRIPSI**

**SISTEM KLASIFIKASI PENYAKIT TANAMAN CABAI  
MENGUNAKAN METODE *DEEP LEARNING* DENGAN  
*LIBRARY TENSORFLOW LITE***

**Disusun dan diajukan oleh:**

**AHMAD KURNIAWAN SYARIF**

**D421 14 518**



**DEPARTEMEN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS HASANUDDIN**

**MAKASSAR**

**2021**

**LEMBAR PENGESAHAN SKRIPSI**

**SISTEM KLASIFIKASI PENYAKIT TANAMAN CABAI MENGGUNAKAN  
METODE DEEP LEARNING DENGAN LIBRARY TENSORFLOW LITE**

Disusun dan diajukan oleh

**AHMAD KURNIAWAN SYARIF**

**D421 14 518**

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka  
Penyelesaian Studi Program Sarjana Program Studi Teknik Informatika

Fakultas Teknik Universitas Hasanuddin


Pada tanggal 14 Oktober 2021

dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,


Pembimbing Pendamping,

  
Dr. Indrabayu, S.T., M.T., M.Bus, Sys  
Nip. 19750716 200212 1 004

  
Dr. Eng. Intan Sari Areni, ST., M.T.  
NIP. 19750203 200012 2 002

Ketua Program Studi Teknik Informatika



  
Dr. Amil Ahmad Ilham, S.T., M.IT  
Nip. 19731010 199802 1 001

## HALAMAN PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama : Ahmad Kurniawan Syarif

NIM : D42114518

Program Studi : Teknik Informatika

Menyatakan dengan sebenar-benarnya bahwa skripsi yang berjudul :

### **SISTEM KLASIFIKASI PENYAKIT TANAMAN CABAI MENGGUNAKAN DEEP LEARNING DENGAN LIBRARY TENSORFLOW LITE**

Adalah karya ilmiah saya sendiri dan sepanjang pengetahuan saya di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan/ditulis/diterbitkan sebelumnya. Kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila dikemudian hari ternyata di dalam naskah ini terdapat unsur-unsur plagiasi, saya bersedia menerima sanksi atas perbuatan tersebut dan diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2000, pasal 25 ayat 2 dan pasal 70)

Makassar, 15 Oktober 2021

Yang Membuat Pernyataan



AHMAD KURNIAWAN SYARIF

## ABSTRAK

Indonesia menempati urutan keempat sebagai penghasil cabai terbesar di dunia menurut *Food and Agriculture Organization* (FAO) dengan hasil produksi sebanyak 1,8 juta ton pada tahun 2017. Petani cabai di Indonesia sering mengalami peristiwa gagal panen di berbagai daerah. Beberapa peristiwa gagal panen disebabkan oleh cuaca ekstrim serta hama dan penyakit. Salah satu kasus serangan hama dan penyakit terjadi di Bandar Lampung pada tahun 2010 yang menyebabkan lahan seluas 77 hektar menjadi rusak dan gagal panen. Maret 2019 di Desa Ngatru, Kecamatan Ngantang, Kabupaten Malang lahan dengan luas 600 hektar sekitar 70 persen terserang virus gemini dan menyebabkan gagal panen. Maret 2021 terjadi lagi kasus gagal panen yang disebabkan oleh serangan hama dan virus kuning dan penyakit keriting daun di Dusun Karangwidoro Kecamatan Dau Kabupaten Malang. Dalam penelitian ini dibuat sistem klasifikasi penyakit tanaman cabai yaitu Virus Gemini dan Keriting Mozaik dengan menggunakan metode Deep Learning *Convolution Neural Network* (CNN) dengan *Library Tensorflow Lite*. Sistem dibangun menggunakan *google.colab* untuk melatih model klasifikasi. Arsitektur model yang digunakan adalah *EfficientNet* yang dapat diintegrasikan pada perangkat mobile. Total data digunakan sebanyak 300 citra yang terbagi dalam 3 label kelas daun cabai yaitu sehat, keriting mozaik dan virus gemini. Hasil training dan validasi model dengan menggunakan 240 citra daun cabai diperoleh akurasi sebesar 96,67%. Hasil testing dengan menggunakan 60 citra daun cabai diperoleh akurasi sebesar 96,67%.

**Kata Kunci:** *Deep Learning, Convolutional Neural Network, Tensorflow Lite, Klasifikasi Citra Penyakit Cabai, Virus Gemini, Keriting Mozaik, EfficientNet;*

## KATA PENGANTAR

*Bismillahirrahmanirrahim.*

*Assalamu 'alaikum Warahmatullahi Wabarakatuh.*

Alhamdulillah segala puji dan syukur kami panjatkan hanya kepada Allah *subhanahu wata'ala*. Tiada sesembahan yang berhak disembah selain-Nya, yang menguasai langit dan bumi. Atas segala limpahan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir ini dengan judul “**Sistem Klasifikasi Penyakit Cabai Menggunakan Metode Deep Learning dengan Library TensorFlow Lite**” ini dapat diselesaikan sebagai salah satu syarat dalam menyelesaikan jenjang Strata-1 pada Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

Dalam penyusunan penelitian ini disajikan hasil penelitian terkait judul yang telah diangkat dan telah melalui proses pencarian dari berbagai sumber baik jurnal penelitian, prosiding pada seminar-seminar nasional/internasional, buku maupun dari situs-situs di internet.

Penulis menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, mulai dari masa perkuliahan sampai dengan masa penyusunan tugas akhir, sangatlah sulit untuk menyelesaikan tugas akhir ini. Oleh karena itu, pada kesempatan ini penulis menyampaikan rasa syukur dan ucapan terima kasih sedalam-dalamnya kepada:

- 1) Allah *subhanahu wata'ala* Tuhan Yang Maha Esa atas semua hidayah dan karunia-Nya yang tiada batas sehingga dalam menjalani kehidupan dunia penulis mampu menyelesaikan penulisan laporan skripsi ini, Segala puji hanya

untuk Allah yang tak pernah mengabaikan hamba-Nya, disaat tak ada satu pun yang mampu menolong hamba-Nya.

- 2) Kedua orangtua penulis, ayah dan ibunda tercinta yang telah mendoakan kebaikan untuk penulis, berjuang mendidik dan membina dengan penuh rasa kasih sayang yang tak kan mampu terbalaskan. Serta seluruh keluarga dan kerabat yang senantiasa memberi dukungan moral dan materi.
- 3) Bapak Dr. Indrabayu ST., MT., M.Bus.Sys., kepala laboratorium lab kecerdasran buatan dan selaku pembimbing I dan Dr.Eng. Intan Sari Areni, ST.,M.T selaku pembimbing II yang selalu menyediakan waktu, tenaga, pikiran dan memberikan bimbingan dalam penyusunan skripsi ini.
- 4) Ibu Dr. Ingrid Nurtanio, M.T dan Ibu Elly Warni, ST.M.T selaku penguji yang telah membantu penulis dengan memberi masukan perbaikan serta motivasi untuk menjadi karakter yang lebih baik lagi.
- 5) Bapak Dr. Amil Ahmad Ilham, S.T., M.IT., selaku ketua Prodi Teknik Informatika Fakultas Teknik Universitas Hasanuddin atas ilmu, bimbingan dan bantuan yang diberikan selama masa perkuliahan penulis.
- 6) Bapak Dr.Ing. Wahyu Haryadi Piarah MS.,ME. selaku mantan Dekan Fakultas Teknik Universitas Hasanuddin dan segenap alumni IATEL yang telah memberi solusi dan bantuan dana pendidikan kepada penulis untuk mampu melanjutkan proses perkuliahan.
- 7) Bapak Robert, Bapak Zainuddin serta segenap staf Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah membantu kelancaran administrasi akademik penulis.

- 8) Seluruh rekan Informatika 2014, terkhusus kepada Muh. Nuralamsyah Ibrahim, Fadel Pratama, Muh. Kahfi Fajri Kuddus, Hermawan Safrin, Syarif Hidayatullah, Yakip, Abdillah Satari Rahim, Rahmat Firman, Ahmad Setiadi yang telah memberikan dukungan, dan bantuan di masa perkuliahan sampai dengan penyelesaian tugas akhir ini.
- 9) Saudara-saudari Rectifier 2014 yang telah memberikan doa, nasihat, bantuan dan semangat selama proses penyelesaian tugas akhir ini. Serta seluruh pihak yang tidak sempat disebutkan satu persatu yang telah banyak meluangkan tenaga, waktu dan pikiran selama penyusunan laporan tugas akhir ini.
- 10) *Last but not least*, seluruh kerabat AIMP Research Group yang selalu ada dan saling mendukung setiap saat, saling melengkapi kebutuhan dalam “Rumah” yang kita sebut Lab Kecerdasan Buatan. Terus semangat belajar dan berkarya, *Keep on fighting till the end*.

Akhir kata, penulis berharap semoga Tuhan Yang Maha Esa berkenan membalas segala kebaikan dari semua pihak yang telah banyak membantu. Semoga tugas akhir ini dapat memberikan manfaat bagi pengembangan ilmu selanjutnya.  
Amin.

*Wassalamualaikum Warahmatullahi Wabarakatuh.*

Makassar, 21 Juli 2021

Penulis

## DAFTAR ISI

HALAMAN JUDUL .....	i
HALAMAN LEMBAR PENGESAHAN .....	ii
HALAMAN PERNYATAAN KEASLIAN .....	iii
ABSTRAK .....	iv
KATA PENGANTAR .....	v
DAFTAR ISI .....	viii
DAFTAR GAMBAR .....	xi
DAFTAR TABEL .....	xiii
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	3
1.3 Tujuan Penelitian .....	3
1.4 Manfaat Penelitian .....	3
1.5 Batasan Masalah .....	4
1.6 Sistematika Penulisan .....	4
BAB II TINJAUAN PUSTAKA.....	6
2.1 Tanaman Hortikultura .....	6
2.2 Cabai .....	6



2.3	Penyakit Cabai .....	7
2.3.1	Penyakit Kuning/Virus Gemini.....	7
2.3.2	Penyakit Keriting <i>Mozaik</i> .....	9
2.4	<i>Deep Learning</i> .....	10
2.5	<i>Confusion Matrix</i> .....	31
2.6	Tensorflow .....	33
2.6.1	Komponen <i>TensorFlow</i> .....	35
2.6.2	Kelebihan <i>TensorFlow</i> .....	37
2.6.3	Cara Kerja <i>TensorFlow</i> .....	38
2.7	Penelitian Terkait.....	41
2.7.1	Penelitian 1 .....	42
2.7.2	Penelitian 2.....	42
2.7.3	Penelitian 3.....	44
2.7.4	Penelitian 4.....	45
2.7.5	Penelitian 5.....	45
BAB III METODOLOGI PENELITIAN.....		46
3.1	Tahapan Penelitian .....	46
3.2	Lokasi dan Waktu Penelitian.....	47
3.3	Instrumen Penelitian .....	48
3.4	Teknik Pengambilan Data .....	48
3.4.1	<i>Preprocessing Data</i> .....	49

3.5	Perancangan dan Implementasi Sistem .....	51
3.5.1	Tahap Training .....	64
3.5.2	Tahap Testing .....	64
3.5.3	Implementasi Program .....	60
BAB IV HASIL DAN PEMBAHASAN .....		68
4.1	Hasil Penelitian.....	68
4.1.1	Hasil training model .....	67
4.2	Pembahasan .....	72
4.2.1	Parameter Jumlah Epochs Saat Training.....	73
4.2.2	Parameter Learning Rate.....	74
4.2.3	Parameter Dropout Rate .....	75
BAB V PENUTUP.....		77
5.1	Kesimpulan.....	77
5.2	Saran .....	77
DAFTAR PUSTAKA .....		78

## DAFTAR GAMBAR

<b>Gambar 2.1</b> Virus gemini .....	8
<b>Gambar 2.2</b> Pola Gejala Serangan Virus Gemini .....	9
<b>Gambar 2.3</b> Keriting Mozaik.....	9
<b>Gambar 2.4</b> Perbedaan Lapisan Layer JST dengan Deep Learning .....	13
<b>Gambar 2.5</b> Fungsi Linear .....	14
<b>Gambar 2.6</b> Fungis Sigmoid.....	14
<b>Gambar 2.7</b> Fungsi ReLU.....	15
<b>Gambar 2.8</b> Struktur CNN.....	17
<b>Gambar 2.9</b> <i>Convolutional Layer</i> .....	21
<b>Gambar 2.10</b> <i>Max Pooling</i> dan <i>Average Pooling</i> .....	22
<b>Gambar 2.11</b> <i>Global Average Pooling</i> .....	24
<b>Gambar 2.12</b> <i>Processing of Fully-Connected Layer</i> .....	25
<b>Gambar 2.13</b> <i>Processing of Dropout</i> .....	26
<b>Gambar 2.14</b> Layer dan Activation Functions dalam Model <i>EfficientNet</i> .....	30
<b>Gambar 2.15</b> Sturktur Model <i>EfficientNet</i> .....	30
<b>Gambar 2.16</b> Deskripsi Layer Arsitektur Model <i>EfficientNet</i> .....	31
<b>Gambar 2.17</b> Arsitektur Model <i>EfficientNet</i> .....	31
<b>Gambar 2.18</b> Parameter <i>default Efficient Net</i> dalam <i>TensorFlow</i> .....	31
<b>Gambar 2.19</b> <i>TensorFlow Toolkit Hierarchy</i> .....	34
<b>Gambar 2.20</b> Perbandingan performa algoritma <i>EfficientNet</i> .....	41
<b>Gambar 3.1</b> Diagram tahap penelitian .....	46
<b>Gambar 3.2</b> Ilustrasi teknis pengambilan data .....	49

<b>Gambar 3.3</b> Sampel data penelitian dari setiap kelas .....	49
<b>Gambar 3.4</b> Contoh proses <i>cropping</i> .....	51
<b>Gambar 3.5</b> Gambaran Umm Proses Kerja Sistem .....	52
<b>Gambar 3.6</b> <i>Flowchart</i> .....	52
<b>Gambar 3.7</b> Import Library Python dan TensorFlow .....	53
<b>Gambar 3.8</b> Load Dataset .....	54
<b>Gambar 3.9</b> Pembagian Dataset.....	54
<b>Gambar 3.10</b> Membuat model dan menentukan parameter .....	56
<b>Gambar 3.11</b> Arsitektur Model .....	57
<b>Gambar 3.12</b> Layer Arsitektur Model .....	57
<b>Gambar 3.13</b> Model Summary .....	58
<b>Gambar 3.14</b> Proses Konvolusi .....	58
<b>Gambar 3.15</b> Perhitungan Konvolusi .....	64
<b>Gambar 3.16</b> Posisi Perhitungan .....	64
<b>Gambar 3.18</b> Evaluasi Model .....	64
<b>Gambar 3.19</b> Fungsi Prediksi data Testing.....	64
<b>Gambar 3.20</b> Fungsi <i>Export</i> Model .....	65
<b>Gambar 3.21</b> <i>Interface</i> Sistem .....	67
<b>Gambar 4.1</b> Grafik hasil <i>training</i> dan <i>validasi</i> .....	56
<b>Gambar 4.2</b> Tampilan Interface Hasil Prediksi Citra .....	59

## DAFTAR TABEL

<b>Tabel 2.1</b> <i>Confusion Matrix</i> .....	31
<b>Tabel 2.2</b> Hirarki <i>Toolkit TensorFlow</i> .....	33
<b>Tabel 3.1</b> Rincian Pembagian Dataset Berdasarkan Kelas .....	47
<b>Tabel 3.2</b> Rincian Pembagian Dataset (Latih, Validasi, dan Tes).....	51
<b>Tabel 3.3</b> <i>Confusion Matrix</i> .....	54
<b>Tabel 4.1</b> Hasil Kesalahan Klasifikasi .....	56
<b>Tabel 4.2</b> Hasil Klasifikasi yang benar .....	57
<b>Tabel 4.3</b> Hasil perhitungan <i>Confusion Matrix</i> .....	58
<b>Tabel 4.4</b> Perbandingan Hasil <i>Training</i> berdasarkan parameter <i>epoch</i> .....	59
<b>Tabel 4.5</b> Perbandingan Hasil <i>Training</i> berdasarkan parameter <i>learning rate</i> .	59
<b>Tabel 4.6</b> Perbandingan Hasil <i>Training</i> berdasarkan parameter <i>dropout rate</i> ..	59

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Indonesia menempati urutan keempat sebagai penghasil cabai terbesar di dunia menurut *Food and Agriculture Organization* (FAO) dengan hasil produksi sebanyak 1,8 juta ton pada tahun 2017 (Maulina, 2017).

Petani cabai di Indonesia sering mengalami peristiwa gagal panen di berbagai daerah. Salah satu kasus serangan hama dan penyakit virus kuning terjadi di Bandar Lampung pada tahun 2010 yang menyebabkan lahan seluas 77 hektar menjadi rusak dan gagal panen. Ini menyebabkan harga cabai saat itu melonjak berkisar Rp.45.000,- hingga Rp.50.000 per kilogram (Wadrianto, 2010). Pada Maret 2019 di Desa Ngatru, Kecamatan Ngantang, Kabupaten Malang lahan dengan luas 600 hektar sekitar 70 persen terserang virus gemini dan menyebabkan gagal panen(Aminuddin,2019). Pada Maret 2021 terjadi lagi kasus gagal panen yang disebabkan oleh serangan hama dan virus kuning dan penyakit keriting daun di Dusun Karangwidoro Kecamatan Dau Kabupaten Malang(KompasTV, 2021).

Beberapa peristiwa gagal panen disebabkan oleh cuaca ekstrim serta hama dan penyakit. Hama yang menjadi penyebab gagal panen yaitu Kutu Daun (*Myzus persicae*), Thrips (*Thrips parvispinus*), dan Tungau (*Polyphagotarsonemus latus*). Hama ini menyerang tanaman cabai pada fase *vegetative* hingga fase *generative* dan hama ini merupakan vektor dari penyebaran virus kuning dan mozaik Pada umumnya hama-hama ini merusak

tanaman dengan cara menghisap cairan daun terutama pada pucuk tanaman sehingga daun tanaman tersebut mengeriting dan sulit untuk terbuka serta berubah menjadi kuning lalu mati. Ketiga hama tersebut juga merupakan vektor virus gemini atau virus kuning dan keriting mozaik (Banu, 2017).

Permasalahan yang masih dihadapi oleh petani saat ini adalah cara untuk mengenal berbagai jenis hama dan penyakit yang menyerang tanaman cabai mereka. Gejala penyakit tanaman cabai tidak separah yang ditimbulkan oleh serangan hama, namun dampak yang dihasilkan sama-sama memberikan kerugian. Meskipun penyakit pada tanaman cabai ini musiman tapi dapat mengakibatkan tanaman rusak sehingga mengakibatkan jumlah produksi berkurang.

Penyakit pada tanaman cabai dapat menghambat pertumbuhan tanaman dan mengurangi kualitas produksi, bahkan dapat menyebabkan kematian pada tanaman. Tanaman cabai yang terinfeksi penyakit akan menampilkan gejala perubahan warna daun dan munculnya bercak yang memiliki pola tertentu pada bagian daun, batang dan buah cabai. Daun adalah bagian yang paling mudah untuk mengidentifikasi adanya serangan hama atau penyakit pada tanaman. Sebab daun adalah tempat berkumpul dan berkembangnya vektor pembawa virus dan penyakit.

*Efficient-Net* merupakan salah satu arsitektur model *Convolution Neural Network* (CNN) yang dapat memprediksi dan mengklasifikasi objek secara akurat dan dapat diaplikasikan ke peralatan *mobile* yang memiliki komputasi terbatas (Duang et al., 2020). Maka melalui penelitian ini, peneliti

ingin membuat sistem pengklasifikasian penyakit pada tanaman cabai dengan mendeteksi gejala pada daun tanaman cabai. Oleh karena itu, tugas akhir yang akan dilakukan berjudul “**Sistem Klasifikasi Penyakit Tanaman Cabai menggunakan *Deep Learning* dengan Library Tensorflow Lite**”.

## **1.2 Rumusan Masalah**

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam penelitian tugas akhir ini adalah:

1. Bagaimana cara mengklasifikasi penyakit pada tanaman cabai dengan menggunakan metode *deep learning*?
2. Bagaimana kinerja sistem dan akurasi dalam klasifikasi penyakit pada tanaman cabai menggunakan metode *deep learning*?

## **1.3 Tujuan Penelitian**

Merujuk pada rumusan masalah yang ada, maka tujuan yang ingin dicapai dalam penelitian ini adalah:

1. Untuk mengimplementasikan metode *deep learning* pada proses klasifikasi penyakit tanaman cabai.
2. Untuk mengukur akurasi dalam proses klasifikasi penyakit pada tanaman cabai.

## **1.4 Manfaat Penelitian**

Ada pun manfaat dalam penelitian ini adalah:

1. Manfaat bagi masyarakat, penelitian ini dapat digunakan sebagai alat bantu untuk mengklasifikasikan penyakit tanaman cabai agar dapat



mengurangi kesalahan dalam mengenali penyakit sehingga dapat memberi penanganan yang tepat.

2. Manfaat bagi peneliti, penelitian ini dapat dijadikan sebagai referensi dalam menggunakan metode *deep learning* untuk mengembangkan sistem yang lebih kompleks dengan berbagai objek yang serupa.

### **1.5 Batasan Masalah**

Mengingat banyaknya perkembangan yang bisa ditemukan dalam permasalahan ini, maka perlu adanya batasan masalah yang jelas mengenai apa yang dibuat dan diselesaikan dalam program ini. Adapun batasan-batasan masalah pada penelitian ini sebagai berikut:

1. Objek penelitian difokuskan pada daun tanaman cabai.
2. Penyakit yang akan diteliti yaitu penyakit Keriting Mozaik dan Virus Gemini.
3. Sistem klasifikasi dibuat menggunakan arsitektur *EfficientNet*.
4. Menggunakan *library Deep Learning TensorFlow Lite*
5. Pengambilan data diambil menggunakan kamera *smartphone*

### **1.6 Sistematika Penulisan**

Untuk memberikan gambaran singkat mengenai isi tulisan secara keseluruhan, maka akan diuraikan beberapa tahapan dari penulisan secara sistematis, yaitu:

## **BAB I PENDAHULUAN**

Bab ini membahas latar belakang penelitian, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan sistematika penulisan.

## **BAB II TINJAUAN PUSTAKA**

Pada bab ini akan dijelaskan teori-teori yang mendasari dan berhubungan dengan penelitian, termasuk di dalamnya literasi tentang penyakit tanaman cabai, metode *deep learning*, dan library *TensorFlow Lite*.

## **BAB III METODOLOGI PENELITIAN**

Pada bab ini berisi tentang perancangan sistem, pembuatan sistem, skenario pelatihan dan komputasi data, serta skenario pengujian sistem untuk melihat performa dari sistem yang dibuat.

## **BAB IV HASIL DAN PEMBAHASAN**

Dalam bab ini menjelaskan tentang penyajian data hasil performa sistem, proses-proses yang dilakukan terhadap data-data yang telah diperoleh, dan pembahasan atau evaluasi hasil implementasi sistem secara keseluruhan.

## **BAB V PENUTUP**

Bab ini berisi kesimpulan dan penelitian yang telah dibuat, serta saran-saran untuk dapat meningkatkan dan mengembangkan sistem pada masa yang akan datang.

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **2.1 Tanaman Hortikultura**

Tanaman hortikultura merupakan komponen penting dalam pembangunan pertanian yang terus bertumbuh dan berkembang dari waktu ke waktu. Sub sector hortikultura menempati posisi strategis dalam pembangunan pertanian terus meningkat seperti tercermin dalam beberapa indikator pertumbuhan ekonomi. Potensi pasar komoditas hortikultura baik untuk pasar domestik maupun internasional masih sangat tinggi. Perubahan iklim merupakan salah satu tantangan yang perlu segera dihadapi supaya produk hortikultura Indonesia tetap bertumbuh dan berkembang. Solusi untuk permasalahan itu adalah masyarakat harus mampu menghasilkan varietas yang mempunyai daya saing dan teknologi yang mampu mempertahankan atau bahkan meningkatkan kualitas produksi (M. Jawal, 2015).

#### **2.2 Cabai**

Cabai (*Capsicum annum L*) merupakan tanaman hortikultura yang banyak dibudidayakan oleh petani karena memiliki nilai ekonomi yang tinggi dan banyak digunakan untuk konsumsi rumah tangga maupun industri (Husain dan Abdul, 2021).

Cabai adalah tanaman perdu dengan rasa buah pedas yang disebabkan oleh kandungan *capsaicin*. Cabai yang beredar di pasaran sebenarnya ada beberapa jenis, tetapi sistem penjualan yang umum dilakukan adalah

mencampur semua jenis cabai tersebut menjadi satu. Sehingga orang hanya mengenal satu macam cabai saja, yaitu cabai merah. Kulitnya terlihat mengkilap dan mulus. (Ananto, 2015).

## **2.3 Penyakit Cabai**

Penyakit pada tanaman cabai dapat menurunkan kualitas produk yang dihasilkan bahkan menyebabkan kematian pada tanaman. Tanaman yang terinfeksi penyakit, akan menampilkan gejala berupa bercak yang memiliki pola dan warna tertentu pada beberapa bagian tubuh tanamana cabai.

Berikut merupakan beberapa jenis penyakit pada tanaman cabai yang akan diteliti:

### **2.3.1 Penyakit Kuning/Virus Gemini**

Virus Gemini adalah virus yang diklasifikasikan dalam famili *Geminiviridae* yang terbagi dalam *Begomovirus*. Genus *Begomovirus* mempunyai genom berukuran 2.5-2.9 kb yang menyerang tanaman dikotil (Faizah, 2010). Penyakit yang disebabkan virus gemini di Indonesia dikenal dengan berbagai nama antara lain: Penyakit brekele (Sumatera Barat dan Bengkulu), penyakit golkar (Jawa Tengah dan Jawa Timur), penyakit bulai (Jawa Timur), dan penyakit kuning (di berbagai tempat).



**Gambar 2.1** Virus Gemini

Pertikel virus berukuran kecil (20 nm) berbentuk isometrik dan materi genetiknya berupa *deoxyribonucleic acid* (DNA) utas tunggal. Partikel ini muncul secara berpasangan atau kembar sebagai akibat fusi parsial dua partikel isometrik.

### **Gejala Serangan**

Gejala yang ditimbulkan oleh virus gemini berbeda-beda, tergantung pada genus dan spesies tanaman yang terinfeksi. Gejala pada cabai merah pertama kali muncul pada daun muda atau pucuk berupa bercak kuning di sekitar tulang daun, kemudian berkembang menjadi urat daun berwarna kuning (*vein clearing*), cekung dan mengkerut dengan warna mosaik ringan atau kuning seperti pada Gambar 2.1. Gejala berlanjut hingga hampir seluruh daun muda atau pucuk berwarna kuning cerah, dan ada pula yang berwarna kuning bercampur dengan hijau, dan cekung dan mengkerut berukuran lebih kecil dan lebih tebal.



**Gambar 2.2** Pola gejala serangan virus gemini

Virus ditemukan di dataran rendah dari 100 mdpl (diatas permukaan laut) hingga dataran tinggi di atas 1000 mdpl. Virus dapat menyerang berbagai umur tanaman. Virus menyerang berbagai varietas cabai dan berpotensi menyebabkan kehilangan hasil produksi tanaman 20-100% (Gunaeni *et al.* 2008).

### 2.3.2 Penyakit Keriting *Mozaik*

Pada umumnya petani khawatir jika pada lahan ada tanaman cabai yang terinfeksi virus ini karena akibat dari serangannya dapat menyebabkan kehilangan hasil mencapai 100% alias gagal panen dan penyakit ini dapat menyebar dengan cepat jika disertai dengan keberadaan vektornya (serangga penyebar virus). *Myzus persicae* (*aphids*), *Bemisia tabaci* (lalat putih), *Thrips tabaci* adalah vektor dari virus ini. (babel.litbang.pertanian.go.id).



**Gambar 2.3** Pola gejala serangan virus keriting mozaik

Gejalanya bervariasi tergantung pada varietas yang terinfeksi dan kondisi lingkungan. Dalam beberapa kasus, virus mungkin ada tetapi gejalanya tersembunyi atau tertutup. Pada varietas yang rentan, bercak kekuningan atau bintik hijau muda dan kuning dapat terlihat pada daun dan buah. Pada beberapa varietas, pola bercak cincin yang jelas atau garis nekrotik kuning dapat terlihat. Daun muda tampak berkerut dan kurus, dan daun-daun yang berwarna hijau muda nampak kusam dengan penampilan kasar. Seluruh pertumbuhan tanaman sangat terhambat dan mengalami cacat, terlihat rimbun namun tidak produktif. Jika berkembang, buah memiliki banyak luka melingkar berwarna coklat, kadang-kadang dilingkupi lingkaran cahaya kuning.

(<https://plantix.net/>)

## **2.4 Deep Learning**

*Deep Learning* adalah salah satu cabang dari ilmu pembelajaran mesin (*Machine Learning*) yang terdiri algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linier yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam *deep learning* dapat digunakan baik untuk kebutuhan pembelajaran terarah (*supervised learning*), pembelajaran tak terarah (*unsupervised learning*) dan semi-terarah (*semi supervised learning*) dalam berbagai aplikasi seperti pengenalan citra, pengenalan suara, klasifikasi teks, dan sebagainya. *Deep Learning* disebut sebagai *Deep* (dalam) karena struktur dan jumlah jaringan saraf pada algoritmanya sangat banyak bisa mencapai hingga ratusan lapisan.

*Deep Learning* adalah salah satu jenis algoritma jaringan saraf tiruan yang menggunakan metadata sebagai input dan mengolahnya menggunakan sejumlah lapisan tersembunyi (*hidden layer*) transformasi non linier dari data masukan untuk menghitung nilai output. Algoritma pada *Deep Learning* memiliki fitur yang unik yaitu sebuah fitur yang mampu mengekstraksi secara otomatis. Hal ini berarti algoritma yang dimilikinya secara otomatis dapat menangkap fitur yang relevan sebagai keperluan dalam pemecahan suatu masalah. Algoritma semacam ini sangat penting dalam sebuah kecerdasan buatan karena mampu mengurangi beban pemrograman dalam memilih fitur yang eksplisit. Algoritma ini dapat digunakan untuk memecahkan permasalahan yang perlu pengawasan (*supervised*), tanpa pengawasan (*unsupervised*), dan semi terawasi (*semi supervised*).

Jaringan saraf yang dimiliki oleh *Deep Learning* terbentuk dari hirarki sederhana dengan beberapa lapisan hingga tingkat tinggi atau banyak lapisan (*multi layer*). Berdasarkan hal itulah *Deep Learning* dapat digunakan untuk memecahkan masalah kompleks yang lebih rumit dan terdiri dari sejumlah besar lapisan transformasi non-linier. *Deep Learning* bekerja berdasarkan pada arsitektur jaringan dan prosedural optimal yang digunakan pada arsitektur.

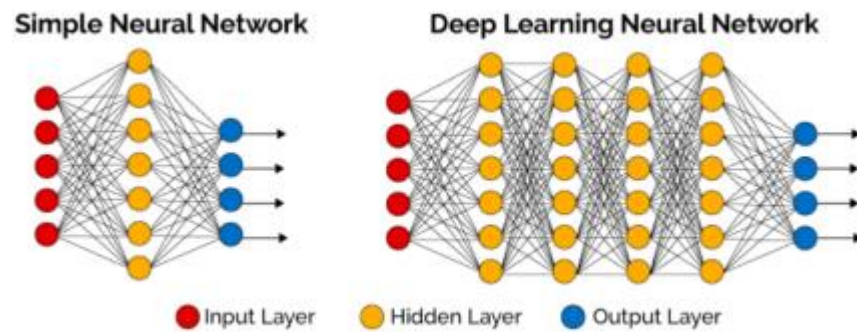
Setiap *output* dari lapisan per lapisan yang tersembunyi dapat dipantau dengan menggunakan grafik khusus yang dirancang untuk setiap *output* neuron. Kombinasi dan rekombinasi dari setiap neuron yang saling terhubung dari semua unit lapisan tersembunyi dilakukan menggunakan gabungan dari



fungsi aktivasi. Prosedur-prosedur tersebut dikenal sebagai Transformasi Non-Linier yang digunakan untuk prosedur optimal untuk menghasilkan bobot optimal pada setiap unit lapisan guna mendapatkan nilai target yang dibutuhkan.

Dalam proses perancangan, apabila jumlah saraf yang ditambahkan sangat banyak, hal tersebut tidak akan pernah cocok untuk menyelesaikan setiap masalah. Persoalan terpenting dalam *Deep Learning* adalah jaringan sarafnya dilatih dengan cara penurunan gradien secara sederhana. Pada saat kita menambahkan lapisan jaringan yang semakin banyak, maka sebaliknya penurunan dari gradien semakin berkurang sehingga dapat mempengaruhi nilai outpunya. Jaringan Saraf Tiruan adalah jaringan saraf yang biasanya menggunakan jaringan seperti umpan maju (*feed forward*) atau *recurrent network* yang hanya memiliki 1 atau 2 lapisan tersembunyi. Tetapi, jika lapisan jaringan sarafnya lebih dari 2 *layer* atau bahkan mencapai ratusan lapisan itulah yang disebut sebagai *Deep Learning*. Pada Jaringan Saraf Tiruan, arsitektur jaringan yang dimilikinya kurang kompleks dan membutuhkan lebih banyak informasi tentang data input sehingga dapat menentukan algoritma mana yang dapat digunakan yang dapat dilihat pada Gambar 6. Dalam Jaringan Saraf Tiruan terdiri dari beberapa algoritma yaitu *Model Hebb*, *Perceptron*, *Adaline*, *Forward Propagation*, dll. Sedangkan pada algoritma jaringan saraf *Deep Learning* tidak memerlukan informasi apapun terhadap data yang akan dipelajarinya, dan algoritmanya dapat secara

mandiri melakukan tuning (penyetelan) dan pemilihan model yang paling optimal (Wayan Dadang, 2018).



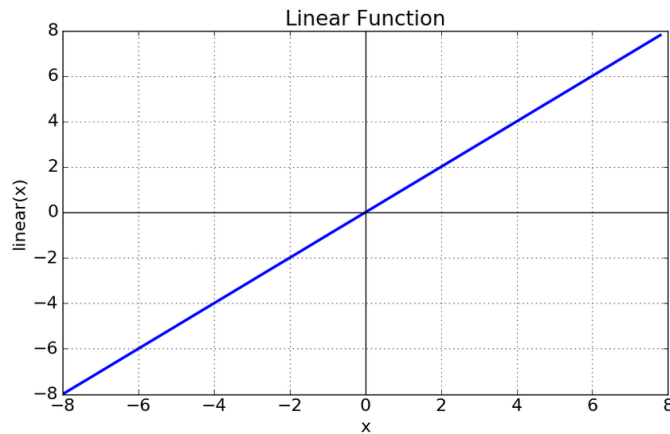
**Gambar 2.4** Perbedaan lapisan layer antara Jaringan Saraf Tiruan dengan Jaringan Deep Learning (CS231n)

### 2.4.1 *Activation Function*

*Activation Function* merupakan fungsi yang menggambarkan hubungan antara tingkat aktivitas internal (*summation function*) yang mungkin berbentuk linear ataupun non-linier. Fungsi ini bertujuan untuk menentukan apakah neuron diaktifkan atau tidak. Ada beberapa fungsi aktivasi yang sering digunakan dalam *Neural Network*, yaitu sebagai berikut (Sena, 2018):

#### 1. Fungsi Aktivasi Linear

Fungsi aktivasi linear merupakan fungsi yang memiliki nilai *output* yang sama dengan nilai inputnya. Hal ini berkaitan dengan, jika sebuah neuron menggunakan *linear activation*, maka keluaran dari neuron tersebut adalah *weighted sum* dari input + bias. Grafik fungsi linear ditunjukkan oleh gambar berikut.

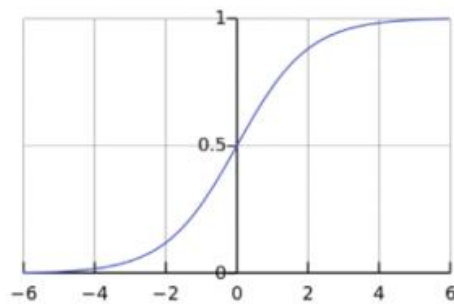


**Gambar 2.5** Fungsi Linear

## 2. Fungsi Aktivasi *Sigmoid*

Fungsi aktivasi *sigmoid* merupakan fungsi non-linear. Masukan untuk fungsi aktivasi ini berupa bilangan riil dan *output* dari fungsi tersebut memiliki nilai antara 0 sampai 1. Berikut ini grafik fungsi aktivasi *sigmoid*:

$$A = \frac{1}{1+e^{-x}}$$



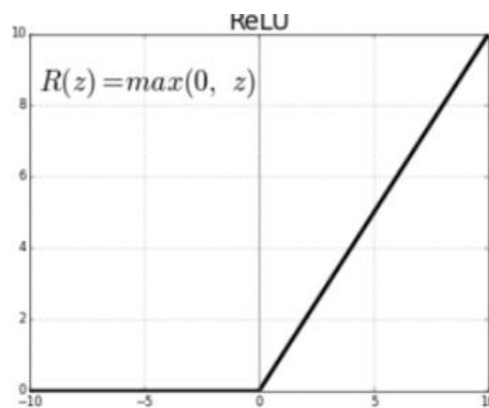
**Gambar 2.6** Fungsi Sigmoid

Jika input dari suatu node pada *neural network* bernilai negatif maka keluaran yang didapatkan adalah 0, sedangkan jika masukannya bernilai positif maka keluaran nilainya adalah 1. Kelemahan dari fungsi ini adalah kemungkinan besar *sigmoid* mematikan *gradient* (*gradient* bernilai 0). Hal ini dapat terjadi ketika

neuron mengeluarkan nilai antara 0 dan 1. Selain itu, *output* dari *sigmoid* tidak berpusat pada 0 (*zero-centered*).

### 3. Activation ReLU

Pada dasarnya fungsi ReLU (*Rectified Linear Unit*) melakukan *threshold* dari 0 hingga *infinity*. Fungsi ini menjadi salah satu fungsi yang populer saat ini. Berikut ini grafik fungsi aktivasi ReLU.



**Gambar 2.7** Fungsi ReLU

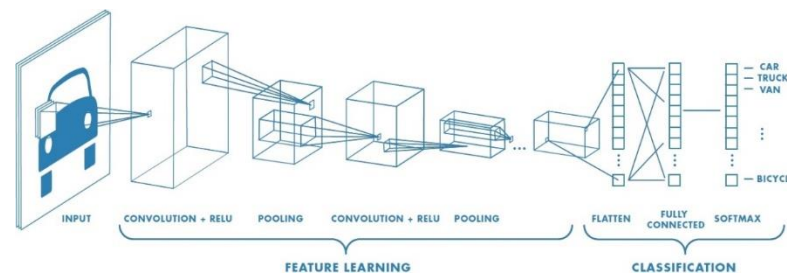
Pada fungsi ini, jika masukan dari neuron-neuron berupa bilangan negatif, maka fungsi ini akan menerjemahkan nilai tersebut kedalam nilai 0, jika masukan bernilai positif maka *output* dari neuron adalah nilai aktivasi itu sendiri. Fungsi aktivasi ini memiliki kelebihan yaitu dapat mempercepat proses konfigurasi yang dilakukan dengan *Stochastic Gradient Descent* (SGD) jika dibandingkan dengan fungsi *sigmoid* dan *tan h*. Namun aktivasi ini juga memiliki kelemahan yaitu aktivasi ini bisa menjadi rapuh pada proses *training* dan bisa membuat unit tersebut mati.

## 2.4.2 Convolutional Neural Network

*Convolutional Neural Network* (CNN) merupakan salah satu algoritma dari *deep learning* yang merupakan hasil pengembangan dari *Multi Layer Perceptron* (MLP) yang dirancang untuk melakukan olah data menjadi bentuk dua dimensi, misalnya gambar atau suara. CNN digunakan untuk melakukan klasifikasi data yang berlabel dengan menggunakan metode *supervised learning* yang dimana terdapat data yang dilatih dan terdapat variabel yang ditargetkan sehingga tujuan dari metode ini yaitu mengelompokkan suatu data ke data yang sudah ada. CNN sering digunakan untuk mengenali benda atas pemandangan dan melakukan deteksi dan melakukan segmentasi objek. CNN belajar langsung melalui data citra, sehingga dapat menghilangkan ekstraksi ciri dengan cara manual. Penelitian awal yang menjadi dasar penemuan ini yaitu pertama kali dilakukan oleh Hubel dan Wiesel yang melakukan penelitian *visual cortex* pada indera penglihatan kucing. *Visual cortex* pada hewan sangat *powerful* kemampuannya dalam sistem pemrosesan visual yang pernah ada. Sehingga, banyak penelitian yang terinspirasi oleh cara kerjanya dan menghasilkan banyak model-model baru yang beberapa diantaranya yaitu, *Neocognitron*, *HMAX*, *LeNet-5*, dan *AlexNet* (Hubel & Wiesel, 1968).

CNN juga merupakan saraf yang dikhususkan untuk memproses data yang memiliki struktur kotak (*grid*). Sebagai contoh yaitu berupa citra dua dimensi. Nama konvolusi merupakan operasi dari aljabar

linear yang mengalikan matriks dari filter pada citra yang akan diproses. Proses ini disebut dengan lapisan konvolusi dan merupakan salah satu jenis dari banyak lapisan yang bisa dimiliki dalam suatu jaringan. Meskipun begitu, lapisan konvolusi ini merupakan lapisan utama yang paling penting digunakan. Jenis lapisan yang lain yang biasa digunakan adalah *Pooling Layer*, yakni lapisan yang digunakan untuk mengambil suatu nilai maksimum atau nilai rata-rata dari bagian-bagian lapisan piksel pada citra. Berikut merupakan gambaran umum arsitektur CNN:



**Gambar 2.8** Struktur CNN (Patel & Pingel, 2017)

Pada Gambar 10 yaitu di setiap lapisan input yang dimasukkan mempunyai susunan neuron 3 dimensi, yaitu lebar, tinggi, dan kedalaman. Lebar dan tinggi yaitu ukuran lapisan, sedangkan untuk ke dalam yaitu mengacu pada jumlah lapisan. Setiap besaran yang didapat tergantung dari hasil filtrasi dari lapisan sebelumnya dan banyaknya filter yang digunakan. Model jaringan seperti ini sudah terbukti efektif dalam menangani permasalahan klasifikasi citra. Sebuah CNN mampu memiliki puluhan hingga ratusan lapisan yang masing-masing lapisan mempelajari deteksi berbagai gambar. Pengolahan citra diterapkan pada setiap citra latih pada resolusi yang berbeda, dan *output* dari masing-masing data gambar yang diolah dan digunakan sebagai input

ke lapisan berikutnya. Pengolahan citra dapat dimulai sebagai fitur yang sederhana, seperti ukuran kecerahan dan tepi atau meningkatkan kekompleksan pada fitur secara unik untuk menentukan objek sesuai ketebalan lapisan. Secara umum tipe lapisan CNN dibagi menjadi dua bagian, yaitu:

1. *Feature extraction layer*

Lapisan pada jenis pertama yaitu adalah *convolutional layer* dan lapisan kedua adalah *pooling layer*. Pada setiap lapisan diberlakukan fungsi aktivasi dengan posisinya yang berselangseling antara jenis pertama dan jenis kedua. Lapisan ini menerima input gambar secara langsung dan memprosesnya sampai menghasilkan *output* berupa vektor untuk diolah pada lapisan berikutnya.

- ***Convolutional Layer***

Tahap ini melakukan operasi konvolusi pada *output* dari *layer* sebelumnya. *Layer* tersebut adalah proses utama yang mendasari jaringan arsitektur CNN. Konvolusi adalah istilah matematis dimana pengaplikasian sebuah fungsi pada *output* fungsi lain dilakukan secara berulang. Operasi konvolusi merupakan operasi pada dua fungsi argumen bertipe *riil*. Operasi ini menerapkan fungsi *output* sebagai *feature map* dari input citra. Input dan *output* ini dapat dilihat sebagai dua argumen bernilai *riil*. Operasi konvolusi dapat dituliskan sebagai berikut:

$$s(t) = (x * t)(t) = \sum_{\alpha} x(\alpha) * w(t - \alpha) \dots (2)$$

Keterangan:

$s(t)$  = Fungsi hasil operasi konvolusi

$x$  = Input

$w$  = bobot (kernel)

Fungsi  $s(t)$  memberikan *output* tunggal berupa *feature map*.

Argumen pertama adalah input yang merupakan  $x$  dan argument kedua  $w$  sebagai kernel atau filter. Apabila input dilihat sebagai citra dua dimensi, maka  $t$  dapat dianggap sebagai piksel dan nilainya dapat dikonversi ke dalam nilai  $i$  dan  $j$ . Jadi sebuah operasi konvolusi dengan input yang mempunyai dimensi lebih besar dari satu dapat ditulis sebagai berikut.

$$s(i,j) = (K*I) (i,j) = \sum_{\infty} \sum_{\infty} I(i - m, j - n)K(m, n) \dots (3)$$

$$s(i,j) = (K*I) (i,j) = \sum_{\infty} \sum_{\infty} I(i + m, j + n)K(m, n) \dots (4)$$

Kedua persamaan di atas merupakan perhitungan dasar dalam operasi konvolusi, dengan  $i$  dan  $j$  adalah sebuah piksel dari citra. Perhitungan tersebut bersifat kumulatif dan muncul saat  $K$  sebagai kernel, kemudian  $I$  sebagai input dan kernel yang dapat dibalik relatif terhadap input. Sebagai alternatif, operasi konvolusi dapat dilihat sebagai operasi perkalian matriks antara citra masukan dan kernel dimana keluarannya dihitung dengan operasi dot. Selain itu, penentuan volume *output* juga dapat ditentukan dari masing-masing lapisan dengan *hyperparameters*.

*Hyperparameter* yang digunakan pada persamaan di bawah ini



digunakan untuk menghitung banyaknya neuron aktivasi dalam sekali *output*. Perhatikan Persamaan 5 berikut.

$$\frac{W - F + 2P}{S + 1} \dots (5)$$

Keterangan:

$W$  = Ukuran volume gambar

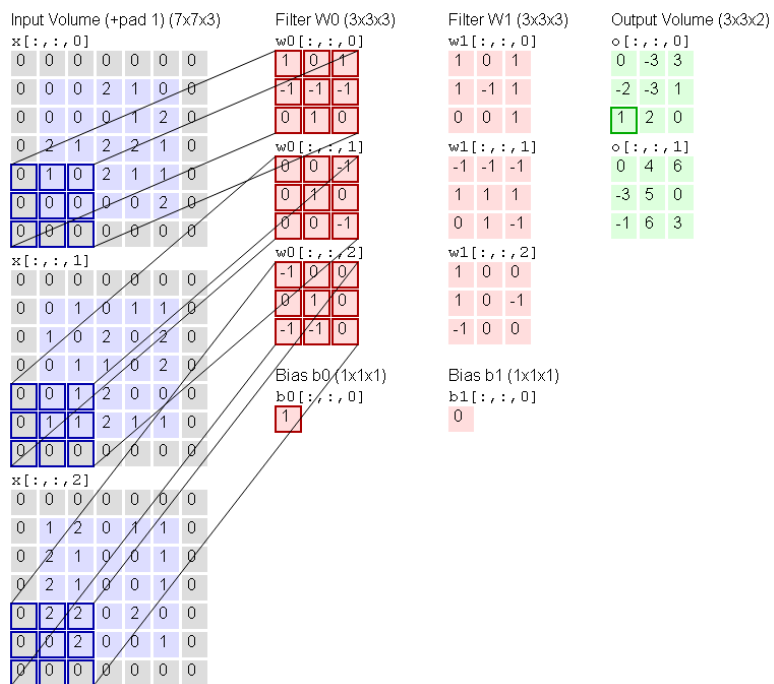
$F$  = Ukuran Filter

$P$  = Nilai *Padding* yang digunakan

$S$  = Ukuran Pergeseran (*Stride*)

Berdasarkan persamaan di atas, dapat dihitung ukuran spasial dari volume *output* dimana *hyperparameter* yang dipakai adalah ukuran volume ( $W$ ), filter ( $F$ ), *Stride* yang diterapkan ( $S$ ) dan jumlah padding nol yang digunakan ( $P$ ). *Stride* merupakan nilai yang digunakan untuk menggeser filter melalui input citra dan *Zero Padding* adalah nilai untuk mendapatkan angka nol di sekitar border citra. Operasi *Convolutional Layer* terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (piksel). Sebagai contoh, *layer* pertama pada *feature extraction layer* biasanya adalah *conv layer* dengan ukuran 5x5x3. Panjang 5 piksel, tinggi 5 piksel dan tebal/jumlah 3 buah sesuai dengan *channel* dari image tersebut. Ketiga filter ini akan digeser ke seluruh bagian dari gambar. Setiap pergeseran akan dilakukan operasi “dot”

antara input dan nilai dari filter tersebut sehingga menghasilkan sebuah *output* atau biasa disebut sebagai *activation map* atau *feature map*. Perhatikan Gambar 11.

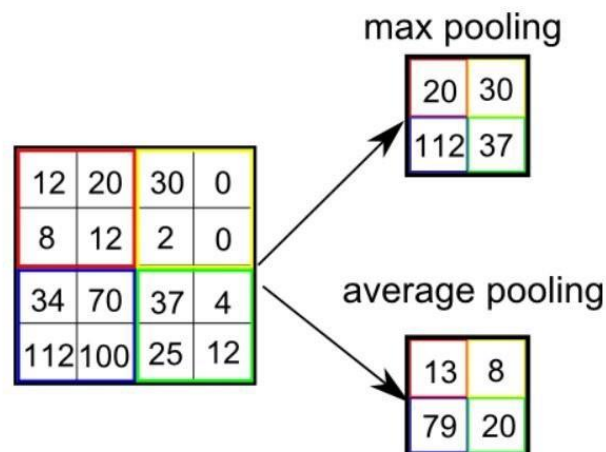


**Gambar 2.9** Convolutional Layer (CS231n)

- **Pooling Layer**

*Pooling* merupakan pengurangan ukuran matriks dengan menggunakan operasi *pooling*. *Pooling Layer* biasanya berada setelah *conv layer*. Pada dasarnya *pooling layer* terdiri dari sebuah filter dengan ukuran dan *stride* tertentu yang akan secara bergantian bergeser pada seluruh area *feature map*. Dalam *pooling layer* terdapat dua macam *pooling* yang biasa digunakan yaitu *average pooling* dan *max-pooling*. Nilai yang diambil pada *average pooling* adalah nilai rata-rata, sedangkan pada *maxpooling* adalah nilai maksimal. Lapisan *pooling* yang

dimasukkan di antara lapisan konvolusi secara berturut-turut dalam arsitektur CNN dapat secara progresif mengurangi ukuran volume *output* pada *feature map*, sehingga mengurangi jumlah parameter pada jaringan untuk mengendalikan *overfitting*. Lapisan *pooling* bekerja pada setiap tumpukan *feature map* dan melakukan pengurangan pada ukurannya. Bentuk lapisan *pooling* umumnya dengan menggunakan filter dengan ukuran 2x2 yang diaplikasikan dengan langkah (*stride*) sebanyak dua dan beroperasi pada setiap irisan dari inputnya. Berikut ini adalah contoh gambar operasi *max-pooling*:



**Gambar 2.10** *Max Pooling* dan *Average Pooling*

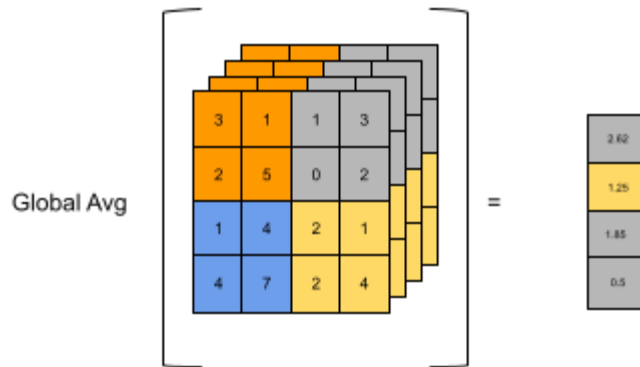
Gambar 12 di atas menunjukkan nilai keluaran *max-pooling* dan *average pooling* dari proses *pooling* adalah sebuah matriks dengan dimensi yang lebih kecil dibandingkan dengan citra awal. Lapisan *pooling* akan beroperasi pada setiap irisan kedalaman volume input secara bergantian. Operasi *max-pooling* pada gambar di atas menggunakan ukuran filter 2x2. Masukan pada

proses tersebut berukuran 4x4. Dari masing-masing 4 angka pada input operasi tersebut diambil nilai maksimalnya kemudian dilanjutkan membuat ukuran *output* baru menjadi ukuran 2x2.

#### **- Global Average Pooling (GA Pooling)**

*Feature map* dari lapisan konvolusi terakhir divektorkan dan dimasukkan ke dalam lapisan yang Fully Connected (FC) diikuti oleh lapisan regresi logistik *softmax*. Struktur ini menjembatani struktur konvolusi dengan jaringan saraf tradisional. Ini menjadikan lapisan konvolusi sebagai ekstraktor fitur, dan fitur yang dihasilkan diklasifikasikan dengan cara tradisional.

GA Pooling adalah proses konvolusi yang dapat menggantikan *Flatten* sebagai pengekstrak fitur. GA Pooling umumnya dipadukan dengan *dropout* agar dapat menghindari terjadinya *overfitting*. Flatten Layer akan mengambil tensor dalam bentuk apa pun dan mengubahnya menjadi tensor satu dimensi tetapi menyimpan semua nilai dalam tensor. Misalnya tensor (sampel, 10, 10, 32) akan diratakan menjadi (sampel, 10 \* 10 \* 32). Global Average Pooling melakukan sesuatu yang berbeda. Ini menerapkan penyatuan rata-rata pada dimensi spasial sampai setiap dimensi spasial adalah satu, dan membiarkan dimensi lain tidak berubah. Misalnya, tensor (sampel, 10, 10, 32) akan ditampilkan sebagai (sampel, 1, 1, 32).



**Gambar 2.11** *Global Average Pooling*

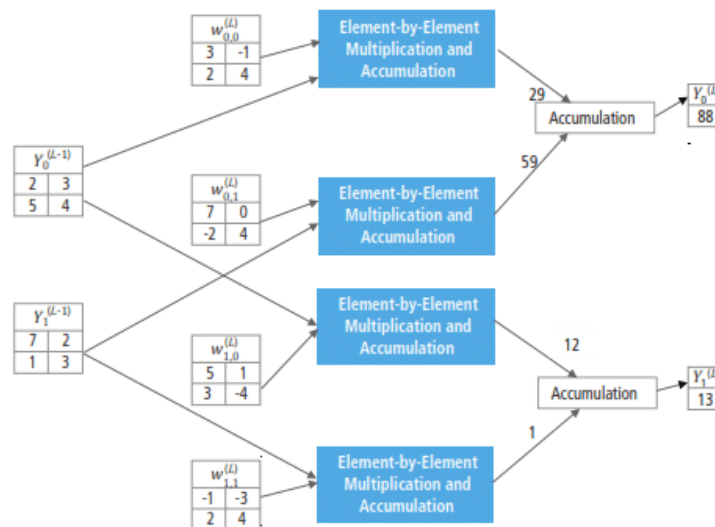
## 2. *Classification Layer*

*Layer* ini tersusun atas beberapa lapisan yang di setiap lapisan tersusun atas neuron yang terkoneksi secara penuh (*fully connected*) dengan lapisan yang lainnya. *Layer* ini menerima input dari hasil *output layer* ekstraksi fitur gambar berupa vektor yang kemudian ditransformasikan seperti pada *Multi Neural Network* dengan tambahan beberapa *hidden layer*. Hasil *output* berupa akurasi kelas untuk klasifikasi. Dengan ini, CNN merupakan metode untuk melakukan transformasi gambar asli lapisan per lapisan dari nilai piksel gambar ke dalam nilai skoring kelas untuk klasifikasi. Setiap lapisan ada yang memiliki *hyperparameter* dan ada yang tidak memiliki parameter (bobot dan bias pada neuron).

### - *Fully-Connected Layer*

*Fully-Connected Layer* adalah sebuah lapisan dimana semua neuron aktivasi dari lapisan sebelumnya terhubung semua dengan neuron di lapisan selanjutnya sama seperti halnya dengan *neural network* biasa. Pada dasarnya lapisan ini biasanya digunakan pada

MLP (*Multi Layer Perceptron*) yang mempunyai tujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. Perbedaan antara lapisan *Fully-Connected* dan lapisan konvolusi biasa adalah neuron di lapisan konvolusi terhubung hanya ke daerah tertentu pada input, sementara lapisan *fully-connected* memiliki neuron yang secara keseluruhan terhubung. Namun, kedua lapisan tersebut masih menggunakan operasi dot, sehingga fungsinya tidak begitu berbeda. Berikut ini adalah proses *fully-connected*:

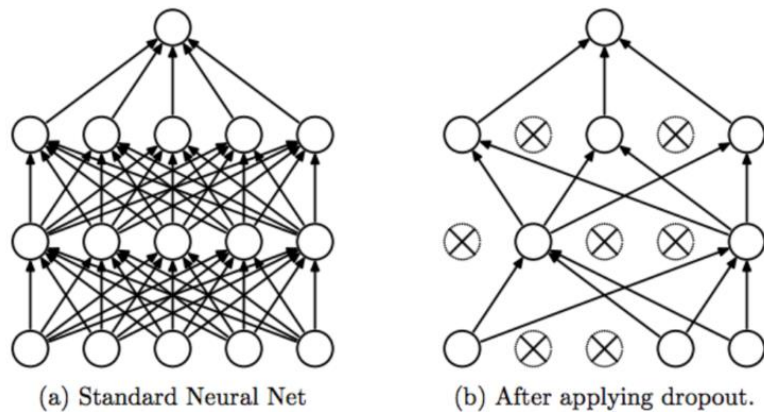


**Gambar 2.12** *Processing of Fully-Connected Layer*

- **Dropout Regulation**

*Dropout* merupakan sebuah teknik regulasi jaringan saraf dengan tujuan memilih beberapa neuron secara acak dan tidak akan dipakai selama proses pelatihan, dengan kata lain neuron-neuron tersebut dibuang secara acak. Hal ini berarti bahwa kontribusi neuron yang dibuang akan diberhentikan sementara jaringan dan

bobot baru juga tidak diterapkan pada neuron pada saat melakukan *backpropagation*. Berikut adalah gambar proses *dropout* (Srivastava dkk, 2014):



**Gambar 2.13** *Processing of Dropout* (Deepak Battini, 2018)

Gambar 14(a) di atas merupakan jaringan saraf biasa yang memiliki dua *hidden layer*. Sedangkan pada bagian 14(b) merupakan jaringan saraf dengan menggunakan *dropout*. Dari gambar tersebut terlihat terdapat beberapa neuron aktivasi yang tidak dipakai lagi. Penggunaan teknik ini akan berdampak pada performa model dalam melatih serta mengurangi *overfitting*. Pada jaringan saraf tiruan biasa, jika  $y^l$  adalah nilai keluaran dari suatu lapisan  $l$  dan  $z^l$  adalah nilai masukan pada layer  $l$  dengan  $W_i$  dan  $b_i$  adalah bobot dan bias dari lapisan  $l$  pada unit ke  $i$ , maka perhitingan proses *feedforward* menggunakan fungsi aktivasi  $f$  dapat dilakukan dengan Persamaan 6 berikut.

$$\begin{aligned}
z_i^{l+1} &= W_i^{(l+1)} y^l + b_i^{(l+1)} \\
y_i^{l+1} &= f(z_i^{(l+1)}) \quad \dots (6)
\end{aligned}$$

Sementara pada jaringan yang mengimplementasikan teknik *DropOut*, variable  $r^l$  melambangkan vektor sepanjang  $j$  yang menyimpan nilai yang diperoleh dari distribusi *Bernoulli*. Proses *feedforward* dilakukan dengan Persamaan 7.

$$\begin{aligned}
\tilde{y}^l &= r_j^l * y^l \\
z_i^{l+1} &= W_i^{(l+1)} \tilde{y}^l + b_i^{(l+1)} \\
y_i^{l+1} &= f(z_i^{(l+1)}) \quad \dots (7)
\end{aligned}$$

- ***Softmax Classifier***

*Softmax Classifier* adalah generalisasi dari fungsi logistik. *Output* dari *softmax* ini dapat digunakan untuk mewakili distribusi sebuah kategori. *Softmax function* digunakan dalam berbagai macam metode klasifikasi, contohnya *multinomial logistic regression*, *multiclass linear discriminant analysis*, *naive bayes classifier*, dan *neural network*. Secara spesifiknya fungsi ini biasa digunakan pada metode klasifikasi *multinomial logistic regression* dan *multiclass linear discriminant analysis*. Berikut adalah fungsi yang diberikan:

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}} \quad \dots (8)$$



Notasi  $f_j$  menunjukkan hasil fungsi untuk setiap elemen ke- $j$  pada vektor keluaran kelas. Argumen  $z$  adalah hipotesis yang diberikan oleh model pelatihan agar dapat diklasifikasi oleh fungsi *Softmax*. *Softmax* juga memberikan hasil yang lebih intuitif dan juga memiliki interpretasi probabilistik yang lebih baik dibanding algoritma klasifikasi lainnya. *Softmax* memungkinkan kita untuk menghitung probabilitas untuk semua label. Dari label yang ada akan diambil sebuah vektor nilai bernilai riil dan merubahnya menjadi vektor dengan nilai antara nol dan satu yang bila semua dijumlah akan bernilai satu.

### **2.4.3 Cross Entropy Loss Function**

*Loss Function* merupakan salah satu komponen penting dalam *neural network*. *Loss* menggambarkan kemungkinan kesalahan yang dihasilkan oleh model. *Loss Function* yang baik adalah fungsi yang diharapkan menghasilkan *error* yang paling rendah. Ketika suatu model memiliki kelas yang cukup banyak, perlu adanya cara untuk mengukur perbedaan antara probabilitas hasil hipotesis dan probabilitas kebenaran yang asli. *Categorical Cross Entropy* merupakan salah satu pilihan terbaik untuk menghitung nilai *loss* pada permasalahan *multi-class classification*. *Categorical Cross Entropy* (CE) biasa juga disebut *Softmax Loss* yang merupakan gabungan dari *softmax activation* dan *cross-entropy loss*. Berikut merupakan rumus *Categorical Cross Entropy* (Gomez, 2018).

$$CE = -\log \left( \frac{e^{s_p}}{\sum_j^c e^{s_j}} \right) \dots (9)$$

#### 2.4.4 Transfer Learning

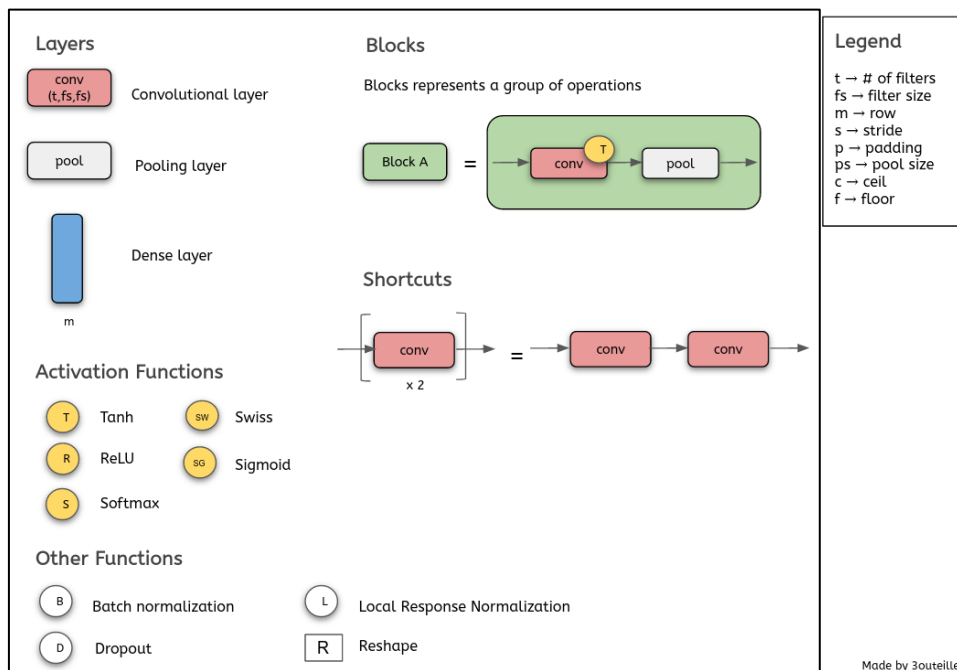
*Transfer learning* adalah suatu teknik atau metode yang memanfaatkan model yang sudah dilatih terhadap suatu *dataset* untuk menyelesaikan permasalahan lain yang serupa dengan cara menggunakannya sebagai *starting point*, memodifikasi dan mengupdate parameternya sehingga sesuai dengan dataset yang baru (Sena, 2018)

##### - Arsitektur model *Efficient-Net*

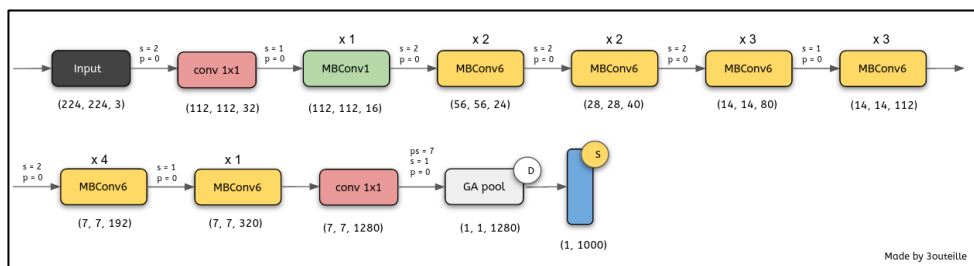
*Efficient-Net* merupakan salah satu arsitektur CNN yang dapat memprediksi dan mengklasifikasi objek secara akurat dan dapat diaplikasikan ke peralatan *mobile* yang memiliki komputasi terbatas (Duang et al., 2020). *Efficient-Net* ditemukan dengan melakukan scaling secara teratur pada tiga komponen; yaitu *depth*, *width*, dan *resolution*. Dalam penambahan *depth* (kedalaman), *width* (lebar), ataupun *resolution* (resolusi) akan membutuhkan waktu yang lebih lama untuk memproses suatu data. Dalam arsitektur *EfficientNet*, penambahan ketiga komponen tersebut dilakukan dengan sangat teratur sehingga didapatkan jumlah parameter yang lebih sedikit yang membuat waktu proses menjadi lebih cepat namun juga didapatkan juga akurasi yang baik dari model sebelumnya. Sehingga model ini

menawarkan efisiensi, dan performa yang lebih baik dari model lain yang ada. (Bima Kusuma, 2020).

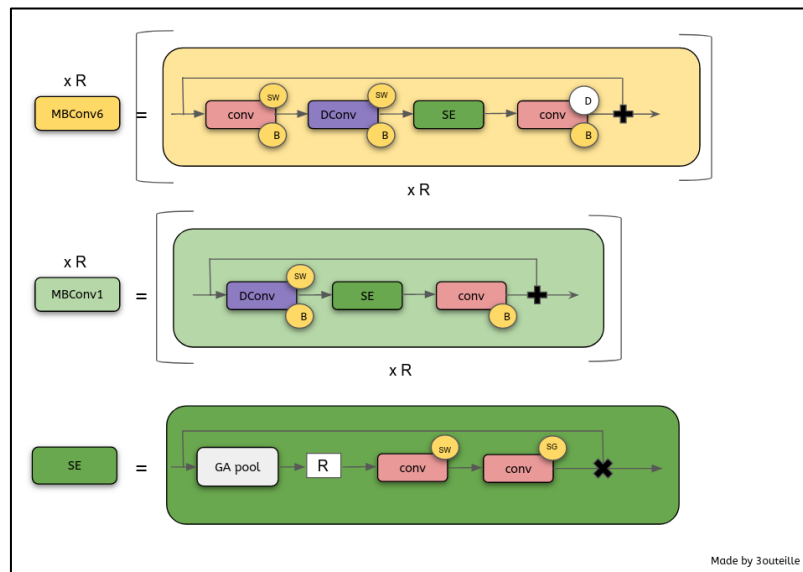
Berikut penjelasan lengkap mengenai arsitektur model *EfficientNet* beserta detail layer konvolusinya ditunjukkan pada gambar dibawah ini.



**Gambar 2.14** Layer dan Activation Functions dalam Model *EfficientNet*



**Gambar 2.15** Struktur Model *EfficientNet*



**Gambar 2.16** Deskripsi Layer Arsitektur Model *EfficientNet*

Stage $i$	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	$224 \times 224$	32	1
2	MBCConv1, k3x3	$112 \times 112$	16	1
3	MBCConv6, k3x3	$112 \times 112$	24	2
4	MBCConv6, k5x5	$56 \times 56$	40	2
5	MBCConv6, k3x3	$28 \times 28$	80	3
6	MBCConv6, k5x5	$14 \times 14$	112	3
7	MBCConv6, k5x5	$14 \times 14$	192	4
8	MBCConv6, k3x3	$7 \times 7$	320	1
9	Conv1x1 & Pooling & FC	$7 \times 7$	1280	1

**Gambar 2.17** Arsitektur model *Efficient-Net*  
Spesifikasi arsitektur Efficientnet (Tan & Le, 2019)

Adapun parameter *default* dalam *EfficientNet* seperti pada gambar 2.16

```
tf.keras.applications.efficientnet.EfficientNetB0(
    include_top=True, weights='imagenet', input_tensor=None,
    input_shape=None, pooling=None, classes=1000,
    classifier_activation='softmax', **kwargs
)
```

**Gambar 2.18** Parameter *default* EfficientNet dalam Tensorflow

## 2.5 Confusion Matrix

Secara umum penentuan baik atau tidaknya performa suatu model klasifikasi dapat dilihat dari parameter pengukuran performanya, yaitu *accuracy*, *recall*, dan *precision*. Untuk menghitung faktor-faktor tersebut

diperlukan sebuah matriks yang biasa disebut dengan *confusion matrix*. *Confusion matrix* adalah cara untuk memvisualisasikan kinerja model dalam bentuk tabel. Dalam pengukuran kinerja menggunakan *confusion matrix*, terdapat empat kondisi untuk merepresentasikan hasil proses klasifikasi, yaitu (Mohajon, 2020):

- *True Positive* (TP): Keadaan di mana *classifier* memprediksi dengan benar kelas positif sebagai positif.
- *True Negative* (TN): Keadaan di mana *classifier* memprediksi dengan benar kelas negatif sebagai negatif.
- *False Positive* (FP): Keadaan di mana *classifier* salah, yaitu memprediksi kelas negatif sebagai positif.
- *False Negative* (FN): Keadaan di mana *classifier* salah, yaitu memprediksi kelas positif sebagai negatif.

Penelitian ini merupakan *multi-class classification* yang mana memiliki 3 kelas keluaran kelas dalam *confusion matrix*.

**Tabel 2.1** Letak TP dalam *confusion matrix* yang digunakan

		Predicted		
		A	B	C
True	A	TP <sub>A</sub>		
	B		TP <sub>B</sub>	
	C			TP <sub>C</sub>

Untuk menghitung nilai akurasi sistem dapat menggunakan rumus pada Persamaan 10.

$$Accuracy = \frac{TP+FN}{TP+FP+TN+FN} \dots (10)$$

## 2.6 Tensorflow

TensorFlow merupakan *open source framework* yang dapat digunakan untuk mengembangkan, melatih, dan menggunakan model deteksi objek. Sistem ini sudah banyak diterapkan pada berbagai produk google antara lain pencarian image, deteksi wajah, dan plat nomor kendaraan pada google street view, Google assistant, way mo atau self driving car, dan lain-lain.

TensorFlow bekerja dengan computational untuk membuat model machine learning. TensorFlow menyediakan berbagai toolkit yang memungkinkan anda membuat model pada tingkat yang lebih rendah untuk membuat model dengan menentukan serangkaian matematis. Sebagai alternative, anda dapat menggunakan API dengan tingkat yang lebih rendah untuk membuat model dengan menentukan API dengan tingkat yang lebih tinggi (seperti tf.estimator) untuk menentukan arsitektur yang telah ditetapkan, seperti regresi linear atau neural network.

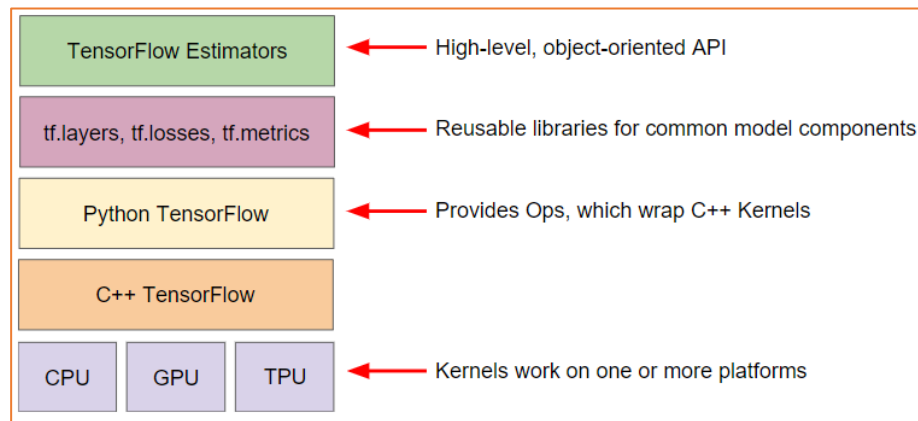
Framework *TensorFlow* digunakan pada proses pembuatan sistem objek deteksi agar memudahkan implementasi algoritma dan penggunaan bahasa pemrograman kemudian terdapat GPU untuk mempercepat proses training. *TensorFlow* sebagai kerangka machine learning yang dapat digunakan untuk mengolah banyak data atau ingin mempelajari kecerdasan buatan (artificial Intellegence) secara mendalam.

*TensorFlow* adalah library open source untuk perhitungan numeric berkinerja tinggi. Arsitekturnya yang fleksibel memungkinkan penyebaran

komputasi dengan mudah di berbagai platform (CPU, GPU, TPU), dan dari desktop ke cluster server hingga perangkat seluler (Nisa Hanum Harani dan Miftahul Hasanah, 2020).

Awalnya dikembangkan oleh para peneliti dan insinyur dari tim google Brain dalam organisasi AI Google, ia hadir dengan dukungan kuat untuk machine learning dan deep learning dan inti perhitungan numerik yang fleksibel digunakan dibanyak domain ilmiah lainnya.

Gambar berikut menunjukkan hierarki toolkit *TensorFlow* saat ini:



**Gambar 2.19** *TensorFlow* Toolkit Hierarchy

Tabel berikut berisi ringkasan tujuan dari berbagai lapisan:

**Tabel 2.1** Hierarki Toolkit *TensorFlow*

Toolkit	Deskripsi
Estimator( <i>tf.estimator</i> )	Merupakan sebuah API tingkat tinggi yang berorientasi pada objek
<i>tf.layer/tf.loses/tf.metrics</i>	<i>Library</i> untuk komponen model umum
<i>TensorFlow</i>	API dengan tingkat lebih rendah

### 2.6.1 Komponen *TensorFlow*

Menurut Lu Yifei (2017) terdapat 4 komponen *computational graph* terkait *TensorFlow* yaitu:

#### 1. *Operations*

Dalam *TensorFlow*, node mewakili operasi. Node menggambarkan bagaimana data input mengalir melalui node dalam grafik yang diarahkan. Suatu operasi dapat memperoleh nol atau banyak input kemudian menghasilkan nol atau banyak output. Operasi semacam itu dapat berupa persamaan matematika, konstanta atau variabel. Konstanta diperoleh dengan operasi tanpa input dan menghasilkan output sama dengan konstanta yang sesuai. Demikian pula, variabel adalah operasi yang tidak mengambil input dan menghasilkan nilai saat ini dari variabel itu. Setiap operasi perlu diimplementasikan oleh kernelnya yang dapat dieksekusi pada perangkat keras seperti CPU atau GPU.

#### 2. *Tensors*

Dalam *TensorFlow*, data diwakili oleh tensor yang mengalir antara node dalam grafik komputasi. Tensor adalah array multi dimensi dengan tipe statis dan dimensi dinamis. Jumlah dimensi suatu tensor disebut pangkatnya. Bentuk tensor menggambarkan jumlah komponen di setiap dimensi. Dalam grafik komputasi, saat membuat operasi, tensor dikembalikan yang akan dikirim oleh tepi terarah sebagai input ke operasi yang terhubung.



### 3. *Variables*

Di seluruh evaluasi pelatihan, sebagian besar tensor tidak bertahan sedangkan kondisi model seperti bobot dan bias perlu dipertahankan. Karenanya variabel ditambahkan ke grafik komputasi sebagai operasi khusus. Variabel menghemat tensor yang disimpan secara terus-menerus dalam buffer memori. Nilai variabel dapat dimuat saat melatih dan mengevaluasi model. Saat membuat variabel, perlu diberikan tensor sebagai nilai awal pada saat eksekusi. Bentuk dan tipe data dari tensor itu secara otomatis menjadi bentuk dan tipe variabel.

Inisialisasi variabel harus dieksekusi sebelum pelatihan. Ini dapat dilakukan dengan menambahkan operasi untuk menginisialisasi semua variabel dan menjalankannya sebelum melatih jaringan.

### 4. *Session*

Eksekusi operasi dan evaluasi tensor dilakukan dalam konteks sesi. Sesi menggunakan Run rutin sebagai entri untuk mengeksekusi grafik komputasi. Dengan invocation run, input dimasukkan ke dalam proses komputasi seluruh grafik untuk mengembalikan output sesuai dengan definisi grafik. Grafik perhitungan akan dieksekusi berulang kali untuk melatih jaringan dengan menjalankan Run rutin. Sesi ini mendistribusikan operasi grafik ke perangkat seperti CPU atau GPU pada mesin yang berbeda sesuai dengan algoritma

penempatan *TensorFlow* yang akan disajikan nanti. Selain itu, urutan eksekusi node didefinisikan secara eksplisit, yaitu dependensi kontrol. Mengevaluasi model memastikan bahwa dependensi kontrol dipertahankan.

### 2.6.2 Kelebihan *TensorFlow*

*TensorFlow* memiliki kelebihan antara lain:

- Cepat

Performa merupakan faktor penting dalam mengembangkan dan menerapkan sistem machine learning. Karena itulah *TensorFlow* menggunakan XLA, pengkompilasi aljabar linear canggih yang membuat kode *TensorFlow* mampu berjalan secepat mungkin pada prosesor, CPU, GPU, TPU, dan platform hardware lain yang digunakan.

- Fleksibel

*TensorFlow* menyediakan API level tinggi yang memudahkan dalam mengembangkan dan melatih model, serta kontrol level rendah untuk mendapatkan fleksibilitas dan performa yang maksimal.

- Siap Produksi

Lingkup penggunaan *TensorFlow* meliputi riset penyelidikan hingga produksi skala besar. Gunakan API *TensorFlow* yang sama dan telah dipahami, baik untuk membuat

jenis model baru maupun memproses jutaan permintaan dalam produksi.

### 2.6.3 Cara Kerja *TensorFlow*

- *Import or generate datasets*

Data merupakan hal yang paling penting dalam pengolahan machine learning, dengan adanya data maka machine learning akan berjalan dengan baik. Pada penelitian ini data yang digunakan adalah data yang dikumpulkan dan diambil sendiri oleh peneliti.

- *Transform and normlize data*

Umumnya, dataset yang diinput tidak memiliki bentuk yang dapat dibaca langsung *TensorFlow*, jadi kita perlu mengubah dataset yang diinput agar dapat dibaca deng *TensorFlow*. Inilah yang dimaksud dengan Transform and normalize data.

- *Partition datasets into train, and validation sets*

Membagi data set menjadi dua bagian yaitu sebagai data train dan data validation, masing-masing akan digunakan pada proses yang berbeda.

- *Set algoritm parameters (hyperparameters)*

Algoritma yang digunakan dalam *training* dan *validation* data adalah *convolutional neural network (cnn)*. Penentuan parameter sangat penting demi memberi hasil akurasi yang baik. Parameter yang dimaksud antara lain adalah *epoch*, jumlah layar konvolusi dan masih banyak lagi.

- *Initialize variable and placeholders*

Kita perlu menginisialisasi variable dan *placeholder* ini dengan ukuran dan jenis yang sesuai, sehingga *TensorFlow* tahu apa yang diharapkan. *TensorFlow* juga perlu mengetahui jenis data yang diharapkan, sehingga dapat bekerja sesuai dengan apa yang kita harapkan.

- *Define the model structure*

Setelah kita memiliki data, dan menginisialisasi variabel dan *placeholder*, kita harus mendefinisikan modelnya. Ini dilakukan dengan membuat grafik komputasi. *TensorFlow* memilih operasi dan nilai apa yang harus menjadi variabel dan *placeholder* untuk sampai pada hasil model kita.

- *Declare the loss functions*

Setelah menentukan model, kita harus mengevaluasi keluarannya. Di sinilah kami mendeklarasikan *loss function*. *Loss function* sangat penting karena memberi tahu kita seberapa jauh prediksi kita dari nilai sebenarnya.

- *Initialize and train the model*

Selanjutnya kita perlu membuat contoh grafik kita, memasukkan data melalui *placeholder* dan membiarkan *TensorFlow* mengubah variabel untuk memprediksi data pelatihan kita dengan lebih baik.

- *Evaluate the model*

Setelah membuat dan melatih model, selanjutnya adalah mengevaluasi model dengan melihat seberapa baik performanya dengan data baru melalui beberapa kriteria yang ditentukan.

- *Tune hyperparameters*

Sering kali, kami ingin kembali dan mengubah beberapa parameter sehingga kinerja model dapat disesuaikan dengan apa yang kita harapkan. Untuk itu kita akan mengulangi langkah sebelumnya dengan parameter yang berbeda dan mengevaluasi model pada set validasi.

- *Deploy/predict new outcomes*

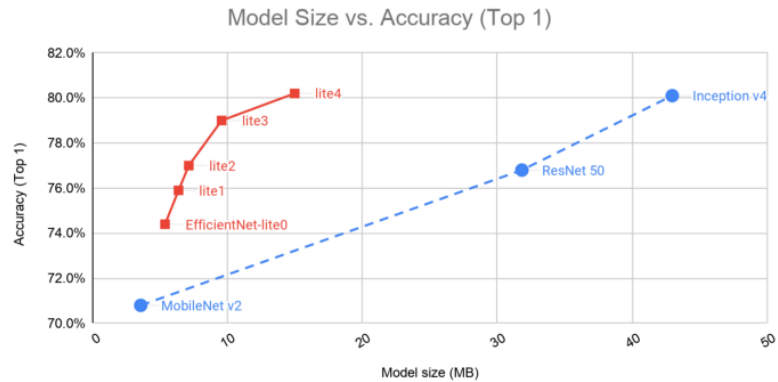
Penting juga untuk mencoba data baru sehingga menghasilkan hasil akurasi pada data prediksi yang baru.

#### **2.6.4 TensorFlow Lite**

*TensorFlow lite* adalah suatu library yang dikembangkan oleh perusahaan *Google* pada tahun 2018. *TensorFlow* ini merupakan perangkat alat yang memungkinkan pembelajaran pada suatu perangkat yang dapat membantu developer menjalankan model mereka di perangkat seluler, tersemat, dan IoT.

##### **- TFLite Model Maker**

*TFLite Model Maker* adalah API Python yang dapat membantu pembuatan training model. Untuk pembuatan model, API ini menyediakan model *Efficient-Net* sebagai model *default* yang siap untuk dilatih.



**Gambar 2.20** Perbandingan performa algoritma *Efficient Net*, *MobileNet*, *ResNet50* dan *Inception v4*

*EfficientNet* termasuk dalam keluarga model klasifikasi gambar yang mampu mencapai akurasi mutakhir pada perangkat edge. Grafik diatas menunjukkan akurasi perbandingan model *EfficientNet* terhadap *MobileNet*, *ResNet* dan *Inception v4*.

## 2.7 Android Studio

Android studio adalah sebuah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi android yang berbasis pada IntelliJ IDEA. Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, android studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi android, misalnya:

- a. Sistem versi berbasis Gradle yang fleksibel
- b. Emulator yang cepat dan kaya fitur
- c. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat android
- d. Instant Run untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru

- e. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh
- f. Dukungan C++ dan NDK
- g. Dukungan bawaan untuk Google Cloud Platform, mempermudah pengintegrasian Google Cloud Messaging dan App Engine.
- h. Dukungan untuk pembuatan aplikasi *machine learning* dengan mengintegrasikan *model tensorflow lite*.

## 2.8 Penelitian Terkait

Di bawah ini adalah beberapa hasil dari penelitian yang berkaitan dengan penelitian yang akan dilakukan :

### 2.8.1 Penelitian 1

Dengan judul penelitian “**Identifikasi Penyakit Cabai Berdasarkan Gejala Bercak Daun dan Penampakan Conidia Menggunakan Probabilistic Neural Network (Jaka Permadi dkk, 2015)**”. Penelitian yang dilakukan adalah identifikasi jenis penyakit tanaman cabai yang dibatasi pada penampakan bercak daun dan *conidia* yang diperoleh dari daun tersebut. Pengenalan dilakukan menggunakan *probabilistic neural network* (PNN) dan data yang dikenali merupakan citra dari daun cabai dan citra mikroskopis yang memperlihatkan *conidia* yang diekstraksi dari permukaan daun. Pengambilan data dilakukan di Yogyakarta dan didapatkan tiga jenis penyakit yang disebabkan oleh bercak daun dan dua jenis *conidia* yang diekstraksi dari setiap daun yang diambil. Ketiga jenis penyakit tersebut adalah bercak

daun serkospora, bercak karena bakteri dan bercak kelabu stemfilium. Sementara kedua jenis *conidia* adalah *Cercospora capsici* dan *Leveillula taurica*. GLCM, fitur warna dan *circularity ratio* digunakan sebagai fitur-fitur dalam mengenali jenis bercak daun. Sementara fitur-fitur bentuk digunakan dalam mengenali jenis *conidia*. Akurasi yang didapat dari pengujian PNN untuk mengenali bercak daun sebesar 94,74%, sedangkan akurasi hasil pengujian dalam mengenali *conidia* sebesar 95,31%.

### **2.8.2 Penelitian 2**

Dengan judul penelitian **“Klasifikasi Penyakit Tanaman Padi Menggunakan Model Deep Learning Efficientnet B3 Dengan Transfer Learning”** Penelitian ini bertujuan untuk mengidentifikasi penyakit padi berdasarkan citra daun. Fitur/ciri terbaik diperoleh dari arsitektur Efficientnet B3 yang ditunjukkan oleh tingginya akurasi pelatihan dan rendahnya loss pelatihan yaitu 99% dan 0,012, namun akurasi yang lebih rendah dihasilkan oleh arsitektur Mobilenet V3 yaitu 57% dengan loss 0,007. Kedua arsitektur tersebut menggunakan transfer learning untuk model pelatihannya Berdasarkan hasil akurasi pelatihan kedua arsitektur maka dapat menentukan akurasi pengujian dengan menggunakan data test yang belum dipelajari sebelumnya. Akurasi pengujian untuk klasifikasi penyakit brown spot dan bacterial leaf yang dihasilkan pada arsitektur EfficientnetB3 lebih tinggi dibandingkan dengan arsitektur Mobilenet V3 yaitu 79,53%.



Penggunaan data test secara acak antara kedua kelas penyakit. Kesimpulannya yaitu klasifikasi penyakit padi berdasarkan fitur yang diperoleh dari citra daun dapat diklasifikasikan dengan baik oleh Efficientnet B3.

### 2.8.3 Penelitian 3

Dengan judul penelitian *“Early Detection of Chili Plant Leaf Diseases using Machine Learning (M. Karuna dkk, 2019).”* Penelitian ini bertujuan untuk mendeteksi dini penyakit pada daun tanaman cabai menggunakan *machine learning*. Penelitian ini merancang sebuah sistem pendeteksi dengan menggunakan kamera *smartphone* secara real-time sehingga dapat mendeteksi secara langsung di lapangan tanpa perlu pemeriksaan di laboratorium. Terdapat 4 objek yang diteliti pada penelitian ini, salah satu di antaranya adalah Kutu Kebul atau dikenal dengan istilah *whitefly* salah satu Organisme Pengganggu Tanaman (OPT) yang mampu merusak tanaman dengan cepat. Tiga objek tersisa adalah penyakit *Yellowish* (virus gemini), *curl leaf* (keriting daun) dan *gray mould (Botrytis)* serta daun sehat (*Healthy leaf*). Hasil dari penelitian ini menunjukkan akurasi yang sangat baik yaitu sebesar 96,4%. Akan tetapi kendala yang ditemukan dari penelitian ini adalah sulitnya menemukan objek penelitian sehingga jumlah data untuk setiap kelas berbeda.

#### 2.8.4 Penelitian 4

Dengan judul penelitian “**Klasifikasi Penyakit Tanaman Padi Menggunakan Metode *Deep Learning* (Tedi Setiady, 2021).**” Pada tahun 2021, Tedi Setiady melakukan penelitian ini terhadap tanaman cabai dengan 4 kategori, 3 kelas dari jenis penyakit yaitu *blight*, *blast* dan tungro kemudian *healthy* untuk kategori sehat/tidak berpenyakit. Penelitian tersebut menggunakan *Convolutional Neural Network (CNN)* sebagai metode *classifier* dan *VGG16* sebagai arsitektur modelnya. Menggunakan 320 data dan berhasil mendapatkan tingkat akurasi sebesar 97,6% dengan melakukan 5 kali percobaan.

#### 2.8.5 Penelitian 5

Dengan judul penelitian “**Deteksi Penyakit Tanaman Cabai Menggunakan Metode Convolutional Neural Network (ATR Dzaky, 2021).**” Pada tahun 2021, ATR Dzaky melakukan penelitian deteksi Penyakit Tanaman Cabai Menggunakan Metode CNN dengan arsitektur AlexNet dan menggunakan library dari Keras *TensorFlow*, dan proses training dilakukan pada *Google Colab*. Penelitian ini mendapatkan hasil yang cukup baik dengan menggunakan parameter *learning rate* 0,01 dan 0,001 mendapatkan akurasi yang cukup bagus hingga mencapai sekitar angka 80% untuk *epoch* 10 dan 90% untuk *epoch* 40.