

**IMPLEMENTASI DEEP LEARNING VGG16 DENGAN  
TRANSFER LEARNING PADA DETEKSI PENYAKIT  
TANAMAN SINGKONG**

SKRIPSI



Oleh:  
**Muhammad Fikri Syahid**  
NIM: 11180910000041

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS SAINS DAN TEKNOLOGI  
UNIVERSITAS ISLAM NEGERI SYARIF HIDAYATULLAH  
JAKARTA  
1442 H/2021 M**

# **IMPLEMENTASI DEEP LEARNING VGG16 DENGAN TRANSFER LEARNING PADA DETEKSI PENYAKIT TANAMAN SINGKONG**

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar  
Sarjana Komputer (S.Kom)



Oleh:

**Muhammad Fikri Syahid**  
NIM: 11180910000041

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS SAINS DAN TEKNOLOGI  
UNIVERSITAS ISLAM NEGERI SYARIF HIDAYATULLAH  
JAKARTA  
1442 H/2021 M**

## PERNYATAAN ORISINALITAS

Dengan ini saya menyatakan bahwa:

1. Skripsi ini merupakan hasil karya asli saya yang diajukan untuk memenuhi salah satu persyaratan memperoleh gelar strata 1 di UIN Syarif Hidayatullah Jakarta.
2. Semua sumber yang saya gunakan dalam penulisan ini telah saya cantumkan sesuai dengan ketentuan yang berlaku di UIN Syarif Hidayatullah Jakarta.
3. Jika dikemudian hari terbukti bahwa karya ini bukan hasil karya asli saya atau merupakan hasil jiplakan dari karya orang lain, maka saya bersedia menerima sanksi yang berlaku di UIN Syarif Hidayatullah Jakarta.

Depok, 21 November 2021



**Muhammad Fikri Syahid**

11180910000041

## PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI

Sebagai civitas akademik UIN Syarif Hidayatullah Jakarta, saya yang bertanda di bawah ini:

Nama : Muhammad Fikri Syahid  
NIM : 11180910000041  
Program Studi : Teknik Informatika  
Fakultas : Sains dan Teknologi  
Jenis Karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Islam negeri Syarif Hidayatullah Jakarta Hak Bebas Royalti Non Eksklusif (Non-exclusive Royalty Free Right) atas karya ilmiah saya yang berjudul:

**"IMPLEMENTASI DEEP LEARNING VGG16 DENGAN TRANSFER LEARNING PADA DETEKSI PENYAKIT TANAMAN SINGKONG"**

Beserta perangkat yang ada (Jika diperlukan). Dengan Hak Bebas Royalti Non Eksklusif ini Universitas Islam Negeri Syarif Hidayatullah Jakarta berhak menyimpan, mengalih media/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik hak. Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Sukabumi

Pada tanggal : 16 Desember 2021

Yang menyatakan



(Muhammad Fikri Syahid)

**Penulis** : Muhammad Fikri Syahid  
**Program Studi** : Teknik Informatika  
**Judul** : **IMPLEMENTASI DEEP LEARNING VGG16  
DENGAN TRANSFER LEARNING PADA DETEKSI  
PENYAKIT TANAMAN SINGKONG**

### **ABSTRAK**

Singkong merupakan salah satu bahan pangan pokok penting di Indonesia. Salah satu faktor yang mempengaruhi daya produksi singkong di Indonesia adalah hama penyakit. Penelitian ini bertujuan untuk membuat model deep learning yang dapat mengklasifikasikan penyakit pada tanaman singkong. Penelitian ini menggunakan 5000 dataset gambar dengan *augmentasi data* untuk melatih model deep learning yang akan digunakan untuk klasifikasi penyakit tanaman singkong. Model terbaik yang dihasilkan penelitian ini adalah model VGG16 *transfer learning* dengan *learning rate* 0.00001 pada *fine-tuning* dan *dropout* dengan rate 50% yang berhasil mengklasifikasikan penyakit pada tanaman singkong dengan akurasi 70%.

Kata kunci:

Singkong, penyakit tanaman, klasifikasi, *model deep learning*, VGG16, *augmentasi data*, *transfer learning*, *learning rate*, *fine-tuning*, *dropout*.

### **ABSTRACT**

Cassava is one of the most important staple foods in Indonesia. One of the factors that affects cassava production rate is pest disease. This research aims to make a deep learning model to classify cassava plant disease. This research uses 5000 image datasets with *data augmentation* to train deep learning model to classify cassava plant disease. The best model acquired from this research is VGG16 model with *transfer learning* and 0.00001 *learning rate* on *fine-tuning* and 50% *dropout rate* that successfully classify cassava plant disease with 70% accuracy.

Keywords:

Cassava, plant disease, classification, *deep learning model*, VGG16, *data augmentation*, *transfer learning*, *learning rate*, *fine-tuning*, *dropout*.

## KATA PENGANTAR



*Assalamu'alaikum Wr. Wb.*

Puji syukur senantiasa dipanjatkan kehadiran Allah SWT yang telah melimpahkan rahmat, hidayat serta nikmat-Nya sehingga penyusunan skripsi ini dapat diselesaikan. Sholawat dan salam senantiasa dihaturkan kepada junjungan kita baginda Nabi Muhammad SAW beserta keluarganya, para sahabatnya serta umatnya hingga akhir zaman. Penulisan skripsi ini mengambil tema dengan judul:

### **IMPLEMENTASI DEEP LEARNING VGG16 DENGAN TRANSFER LEARNING PADA DETEKSI PENYAKIT TANAMAN SINGKONG**

Penyusunan skripsi ini adalah salah satu syarat untuk memperoleh gelar Sarjana Komputer (S.Kom) pada program studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Syarif Hidayatullah Jakarta. Adapun bahan penulisan skripsi ini adalah berdasarkan hasil penelitian, pengembangan aplikasi, wawancara, observasi dan beberapa studi literatur.

Dalam penyusunan skripsi ini, telah banyak bimbingan dan bantuan yang didapatkan dari berbagai pihak sehingga skripsi ini dapat berjalan dengan lancar. Oleh karena itu, penulis ingin mengucapkan banyak terima kasih kepada:

1. Allah subhanahu wa ta'ala yang maha kuasa yang telah memberikan nikmat dan jalan agar penulis dapat menyelesaikan penulisan skripsi ini
2. Ir. Nashrul Hakiem, S.Si., M.T., Ph.D selaku dekan Fakultas Sains dan Teknologi.
3. Dr. Imam Marzuki Shofi, M.T. selaku Ketua Program Studi Teknik Informatika.
4. Ibu Siti Ummi Masruroh, M.Sc. selaku Dosen Pembimbing I yang senantiasa meluangkan waktu dan memberikan bimbingan, bantuan, semangat dan motivasi dalam menyelesaikan skripsi ini.



5. Ibu Luh Kesuma Wardhani, S.T., M.T. selaku Dosen Pembimbing II yang turut memberikan saran serta arahan mengenai pengerjaan dan dalam menyelesaikan skripsi ini.
6. Orang tua tercinta, Bapak Haris dan Ibu Yuli yang tidak pernah lelah mendo'akan serta memberi dukungan moril dan materil sepanjang perjalanan hidup penulis.
7. Seluruh dosen dan staff UIN Jakarta, khususnya Fakultas Sains dan Teknologi Prodi Teknik Informatika yang telah memberikan ilmu serta pengalaman yang berharga.
8. Teman seperjuangan Aditya Sidhiq Pratama dan Kahfi Del Vieri yang sudah memberi masukan untuk penelitian ini.
9. Seluruh pengajar di HimtiDev tahun 2020 yang sudah memberikan wawasan yang sangat bermanfaat pada penelitian ini.
10. Kepada Muhammad Jundi Rabbani selaku adik penulis yang sudah meminjamkan PC untuk membantu menunjang proses pembuatan skripsi.
11. Seluruh pihak yang tidak dapat disebutkan satu persatu yang secara langsung maupun tidak langsung telah membantu dalam menyelesaikan skripsi ini

Penulisan skripsi ini masih jauh dari kata sempurna. Untuk itu, sangat diperlukan kritik dan saran yang membangun bagi penulis. Akhir kata, semoga laporan skripsi ini dapat bermanfaat bagi penulis dan orang lain.

*Wassalamualaikum, Wr. Wb.*

Sukabumi, 21 November 2021

Penulis



**Muhammad Fikri Syahid**

11180910000041

## DAFTAR ISI

<b>PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI</b>	iv
<b>ABSTRAK</b>	v
<b>KATA PENGANTAR</b>	vi
<b>DAFTAR ISI</b>	viii
<b>DAFTAR GAMBAR</b>	x
<b>DAFTAR TABEL</b>	xii
<b>BAB I PENDAHULUAN</b>	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	4
1.5 Manfaat Penelitian	4
1.5.1 Bagi Penulis	4
1.5.2 Bagi Universitas	4
1.5.3 Bagi Masyarakat	4
1.6 Sistematika Penulisan	5
<b>BAB II LANDASAN TEORI</b>	6
2.1 CNN	6
2.1.1 Convolution Layer	6
2.1.2 Pooling Layer	8
2.1.3 Fully-Connected Layer	9
2.2 VGG16	12
2.3 Transfer Learning	12
2.3.1 Fixed Feature Extractor	13
2.3.2 Fine-tuning	14
2.4 Data Augmentation	14
2.5 Dropout	15
2.6 Tensorflow	15
<b>BAB III METODOLOGI PENELITIAN</b>	17
3.1 Metode Implementasi	17
3.2 Metode Pengumpulan Data	17
<b>BAB IV IMPLEMENTASI</b>	25
4.1 Data Collection	25
4.2 Data Preparation	25
4.2.1 Pemilihan Dataset	25
4.2.2 Data Augmentation	26



4.2.3	Resize Image .....	28
4.2.4	Pembagian Dataset .....	29
4.3	Modeling .....	29
4.3.1	Perancangan Model .....	29
4.3.2	Pelatihan Model.....	34
4.4	Evaluation.....	38
<b>BAB V</b>	<b>HASIL DAN PEMBAHASAN .....</b>	<b>39</b>
5.1	Pencarian Model Terbaik .....	39
5.1.1	CNN .....	39
5.1.2	CNN (DO).....	40
5.1.3	CNN (DA).....	40
5.1.4	CNN (DA, DO) .....	41
5.1.5	VGG16 (TL).....	41
5.1.6	VGG16 (DA, TL).....	42
5.1.7	VGG16 (TL, DO).....	42
5.1.8	VGG16 (DA, TL, DO) .....	43
5.1.9	VGG16 (TL, FT).....	43
5.1.10	VGG16 (DA, TL, FT) .....	44
5.1.11	VGG16 (TL, FT, DO) .....	45
5.1.12	VGG16 (DA, TL, FT, DO) .....	45
5.1.13	Ringkasan Pencarian Model Terbaik .....	46
5.2	Uji Coba Parameter Model Terbaik .....	46
5.3	Mencari Titik Konvergen Model Terbaik .....	49
5.4	Implementasi 10-Fold Cross Validation pada Model Terbaik .....	50
<b>BAB VI</b>	<b>KESIMPULAN DAN SARAN.....</b>	<b>51</b>
6.1	Kesimpulan.....	51
6.2	Saran.....	51
<b>DAFTAR PUSTAKA .....</b>		<b>52</b>

## DAFTAR GAMBAR

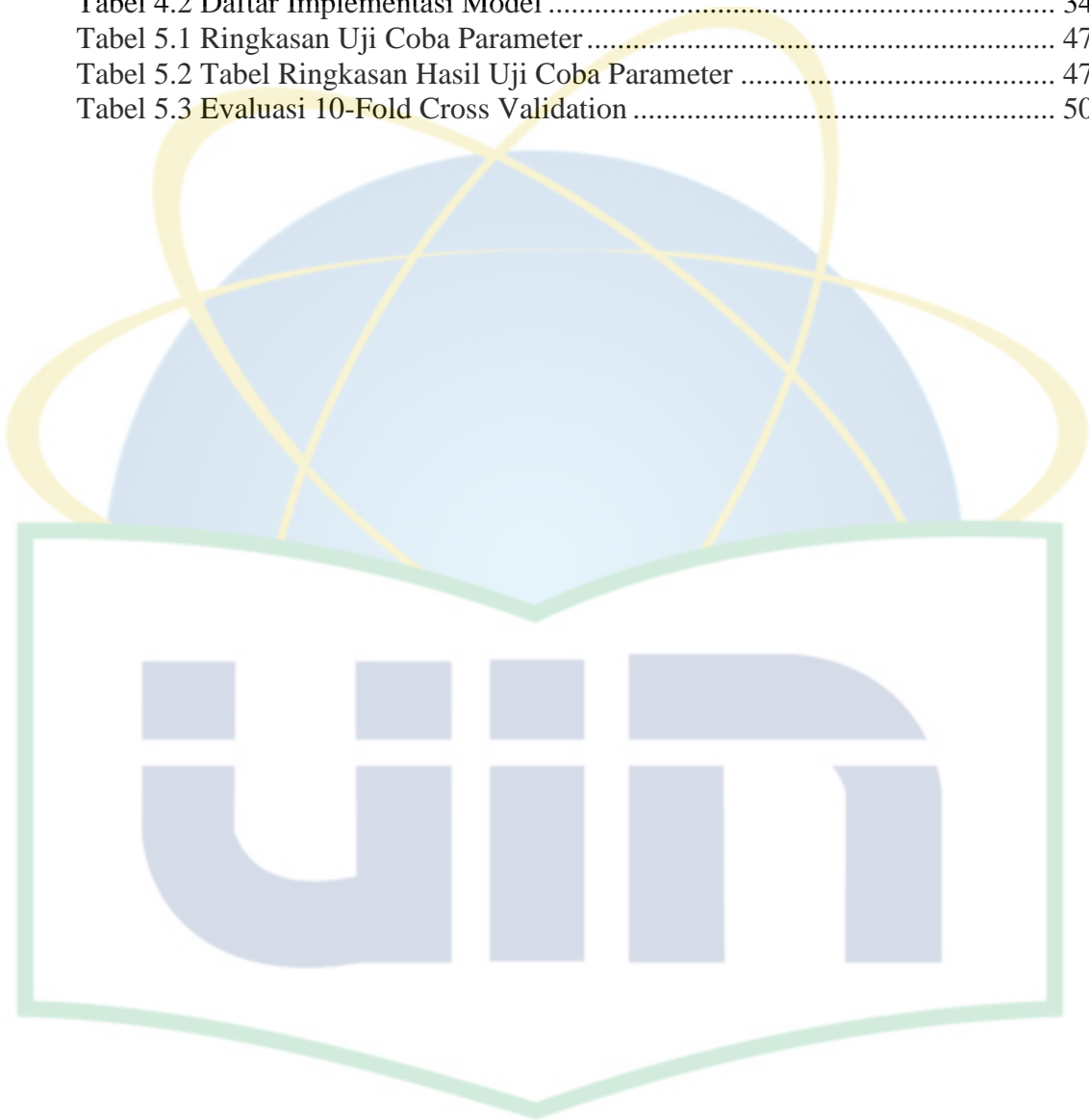
Gambar 2.1. Convolution Layer.....	7
Gambar 2.2. Stride .....	7
Gambar 2.3. Padding.....	8
Gambar 2.4. Pooling Layer dengan Max Pooling.....	9
Gambar 2.5. Fully-Connected Layer.....	10
Gambar 2.6. Sigmoid Function .....	10
Gambar 2.7. Softmax Function .....	11
Gambar 2.8. ReLu Function.....	11
Gambar 2.9 Visualisasi Model VGG16 .....	12
Gambar 2.10 Visualisasi Transfer Learning .....	13
Gambar 2.11 Visualisasi Fixed Feature Extractor .....	13
Gambar 2.12 Visualisasi Fine-Tuning .....	14
Gambar 2.13. Visualisasi Data Augmentation.....	15
Gambar 2.14. Visualisasi Layer Sebelum dan Sesudah Dropout.....	15
Gambar 2.15. Logo Tensorflow .....	16
Gambar 3.1 Alur Penelitian.....	17
Gambar 4.1. Visualisasi Pembagian Dataset.....	26
Gambar 4.2 Kode Implementasi Data Augmentation .....	28
Gambar 4.3 Visualisasi Resize Image.....	28
Gambar 4.4 Kode Implementasi Resize Image.....	29
Gambar 4.5 Pembagian dataset dan 10-fold cross validation .....	29
Gambar 4.6 Summary Model CNN .....	30
Gambar 4.7 Kode Implementasi Model CNN.....	30
Gambar 4.8 Summary Model VGG16 .....	31
Gambar 4.9 Summary Convolution Layer pada Model VGG16 .....	32
Gambar 4.10 Kode Implementasi Model VGG16 .....	33
Gambar 4.11 Kode Implementasi Pelatihan Model CNN dan VGG16 .....	35
Gambar 4.12 Deklarasi batch_size saat proses Resize Image.....	35
Gambar 4.13 Output dari train_generator dan validation_generator .....	36
Gambar 4.14 Kode Implementasi Pelatihan 10 Epoch Pertama .....	37
Gambar 4.15 Kode Implementasi Pelatihan 10 Epoch Kedua.....	37
Gambar 5.1 Evaluasi Model CNN .....	39
Gambar 5.2 Evaluasi Model CNN dengan DO .....	40
Gambar 5.3 Evaluasi Model CNN dengan DA.....	40
Gambar 5.4 Evaluasi Model CNN dengan DA dan DO .....	41
Gambar 5.5 Evaluasi Model VGG16 dengan TL.....	41
Gambar 5.6 Evaluasi Model VGG16 dengan DA dan TL .....	42
Gambar 5.7 Evaluasi Model VGG16 dengan TL dan DO .....	42
Gambar 5.8 Evaluasi Model VGG16 dengan DA, TL, dan DO .....	43
Gambar 5.9 Evaluasi Model VGG16 dengan TL dan FT .....	43
Gambar 5.10 Evaluasi Model VGG16 dengan DA, TL, dan FT .....	44
Gambar 5.11 Evaluasi Model VGG16 dengan TL, FT, dan DO .....	45
Gambar 5.12 Evaluasi Model VGG16 dengan DA, TL, FT, dan DO.....	45
Gambar 5.13 Ringkasan Performa Model.....	46

Gambar 5.14 Tabel Ringkasan Performa Model.....	46
Gambar 5.15 Visualisasi Hasil Model A.....	48
Gambar 5.16 Visualisasi Hasil Model B.....	48
Gambar 5.17 Visualisasi Hasil Model C.....	48
Gambar 5.18 Visualisasi Hasil Model D.....	48
Gambar 5.19 Visualisasi Hasil Model E.....	48
Gambar 5.20 Visualisasi Hasil Model F.....	48
Gambar 5.21 Ringkasan Evaluasi Hasil Uji Coba Parameter.....	48
Gambar 5.22 Evaluasi Hasil Model A dengan 50 Epoch.....	49



## DAFTAR TABEL

Tabel 3.1 Literature Review.....	21
Tabel 3.2 Perbandingan Literature Review.....	24
Tabel 4.1 Jenis Data Augmentasi yang diimplementasikan.....	27
Tabel 4.2 Daftar Implementasi Model .....	34
Tabel 5.1 Ringkasan Uji Coba Parameter.....	47
Tabel 5.2 Tabel Ringkasan Hasil Uji Coba Parameter .....	47
Tabel 5.3 Evaluasi 10-Fold Cross Validation .....	50



# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Singkong merupakan salah satu bahan pangan pokok yang penting di Indonesia karena produksinya yang tinggi. Produksi singkong di Indonesia diperkirakan mencapai 19,3 juta ton pada tahun 2018 (Statistik, 2018). Salah satu faktor yang mempengaruhi tingkat produktivitas tanaman singkong yaitu adanya serangan hama dan penyakit (Minarni., Indra, W., & Wenda, H, 2017). Jenis penyakit yang terdapat pada tanaman singkong diantaranya adalah *Bacterial Blight*, *Brown Streak Disease*, *Green Mottle*, dan *Mosaic Disease*.

Klasifikasi penyakit tanaman singkong dapat meningkatkan daya produksi singkong. Model AI yang dapat mengklasifikasikan penyakit pada tanaman singkong merupakan langkah awal dalam usaha meningkatkan produksi singkong di Indonesia. Salah satu cara untuk mengklasifikasikan penyakit tersebut adalah menggunakan metode *machine learning*. Metode seperti *CBR* (Minarni., Indra, W., & Wenda, H, 2017), *Forward Chaining* (Ahmad et al., 2020), *SVM* (Padol & Yadav, 2016), *Random Forest*, *SGD* (Sujatha et al., 2021), *FCNN*, *k-NN* (Radovanovic & Dukanovic, 2020), dan *CNN* (Susila et al., 2020) diusulkan oleh banyak peneliti untuk mendeteksi penyakit pada tanaman.

*Forward Chaining* dan *CBR* merupakan metode *machine learning* dengan akurasi tinggi namun masih bergantung pada pakar, karena itu pula mereka disebut metode sistem pakar (Ahmad et al., 2020). *SVM* merupakan metode *machine learning* yang cukup baik untuk klasifikasi gambar, namun *SVM* memiliki akurasi yang rendah apabila jumlah fitur nya lebih banyak dari pada jumlah sampel. *CNN* merupakan metode *deep learning* yang dapat memecahkan masalah pada *Forward Chaining* dan *CBR*. *CNN* juga memiliki performa yang lebih baik dibanding metode *machine learning* dan *deep learning* lainnya dalam pengklasifikasian gambar (Radovanovic & Dukanovic, 2020). Namun untuk memperoleh akurasi yang tinggi diperlukan dataset yang besar (Radovanovic & Dukanovic, 2020).

Memperoleh dataset yang besar dapat dibantu dengan metode data augmentation, yaitu metode untuk mentransformasi dataset sehingga memperbanyak varian sampel dari dataset (Taylor & Nitschke, 2017). Penggunaan data augmentation diharapkan dapat meningkatkan akurasi pada CNN juga menghindari over-fitting. Tidak hanya menggunakan data augmentation, metode dropout juga dapat mengurangi over-fitting dengan cara menghentikan pendeteksi fitur secara acak dengan kemungkinan tertentu pada tiap training epoch (Qian et al., 2020).

Performa CNN dapat ditingkatkan dengan menggunakan arsitektur berbeda yaitu VGG16 yang memiliki akurasi yang paling tinggi (Sujatha et al., 2021) dibandingkan dengan beberapa arsitektur CNN lainnya seperti AlexNet, InceptionV3 dan VGG19. Metode lain untuk meningkatkan akurasi dari CNN adalah menggunakan *transfer learning*. Transfer learning adalah metode dimana model deep learning yang sudah dilatih oleh dataset sebelumnya digunakan kembali pada dataset lain (Rozaqi et al., 2021). Transfer learning dapat meningkatkan akurasi model dengan dataset yang sedikit (Rismiyati & Luthfiarta, 2021). Fine tuning merupakan konsep dari transfer learning, model perlu dilatih sebelumnya untuk mengimplementasikan fine tuning. Fine tuning memiliki performa yang lebih baik dari pada model yang disusun secara manual (Too et al., 2019).

Dataset yang akan digunakan berasal dari kaggle dengan judul “*Cassava Leaf Disease Merged*”. Dataset ini terdiri dari 26.338 file gambar singkong beserta label penyakitnya. Terdapat 5 kelas pada dataset ini diantaranya *Bacterial Blight*, *Brown Streak Disease*, *Green Mottle*, *Mosaic Disease*, dan *Healthy*. Hanya sebagian dataset yang akan digunakan, hal ini dikarenakan alasan teknis ketidakmampuan hardware penulis juga tidak diperlukannya dataset melebihi 1000 tiap kelas untuk mendapatkan model yang baik, hal ini merujuk kepada kemenangan arsitektur VGG16 menempati posisi pertama dan kedua pada *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) tahun 2014 dimana pada challenge tersebut digunakan 1000 data pada 1000 kategori data. Dataset akan dibagi menjadi 9:1 antara training dan validasi karena menghasilkan akurasi



tertinggi (Susila et al., 2020). Pembagian dataset 9:1 juga mendukung penggunaan 10-fold cross validation untuk mengevaluasi performa dari model.

Penerapan data augmentation, arsitektur VGG16, transfer learning, finetuning, dan dropout untuk meningkatkan performa CNN dalam mengklasifikasikan penyakit tanaman singkong.

## 1.2 Rumusan Masalah

- Bagaimana performa CNN dalam mengklasifikasikan penyakit pada tanaman singkong?
- Seberapa besar perbandingan akurasi CNN sebelum dan sesudah diterapkan data augmentation, VGG16, transfer learning, dan dropout?

## 1.3 Batasan Masalah

- Metode yang akan digunakan adalah CNN dengan arsitektur VGG16 , transfer learning dengan fine-tuning, data augmentation, dropout.
- Dataset yang akan digunakan didapat dari Kaggle dengan judul “*Cassava Leaf Disease Merged*”.
- Data yang digunakan sebanyak 1000 dari tiap kelas dengan pemilihan secara acak.
- Dataset akan dibagi menjadi 9:1 (*training:validasi*) dan akan menggunakan 10-fold cross validation untuk evaluasi model final.
- Performa saat pencarian model terbaik akan dilihat dari *training accuracy* dan *validation accuracy*.
- Pengolahan citra gambar menggunakan *Jupyter Notebook* dengan menggunakan bahasa pemrograman Python.
- VGG16 dengan transfer learning dan data augmentation akan diimplementasikan menggunakan library Tensorflow dan Keras pada Python.

- Spesifikasi hardware yang digunakan untuk melakukan penelitian:
  - Processor: Intel Core I7 7700HQ @2.8GHz (4 Cores, 8 Threads)
  - RAM: 16GB DDR4 @2400MHz
  - GPU: NVIDIA GTX 1060 3GB

#### **1.4 Tujuan Penelitian**

Menerapkan arsitektur VGG16 dengan transfer learning, data augmentation, fine tuning, dan dropout untuk meningkatkan performa CNN dalam mengklasifikasikan penyakit tanaman singkong.

#### **1.5 Manfaat Penelitian**

##### **1.5.1 Bagi Penulis**

1. Penulis dapat mengaplikasikan pengetahuan akademis yang didapat selama perkuliahan pada penelitian ini.
2. Menambah wawasan penulis dalam bidang penelitian juga image classification.
3. Sebagai salah satu syarat dalam penyelesaian gelar Strata Satu (S1) program studi Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Syarif Hidayatullah Jakarta.

##### **1.5.2 Bagi Universitas**

1. Menambah kumpulan skripsi dari salah satu mahasiswa Teknik Informatika Fakultas Sains dan Teknologi dalam bidang computer vision menggunakan metode deep learning.
2. Menjadi sebuah tolak ukur bagi universitas dalam menentukan keberhasilan dan kemampuan penulis dalam mengimplementasikan ilmu yang sudah didapatkan selama menempuh pendidikan perkuliahan di universitas.

##### **1.5.3 Bagi Masyarakat**

Berguna dalam segi keilmuan sehingga dapat menambah wawasan bagi masyarakat umum.

## 1.6 Sistematika Penulisan

### **BAB 1 PENDAHULUAN**

Bab ini membahas hal umum dalam penelitian, seperti latar belakang dari dari sebuah permasalahan yang diangkat, tujuan penelitian, manfaat penelitian, rumusan masalah, batasan masalah, metodologi penelitian, dan sistematika penulisan.

### **BAB II LANDASAN TEORI**

Bab ini menjelaskan beberapa materi dan teori yang dibutuhkan dalam melaksanakan penelitian ini.

### **BAB III METODE PENELITIAN**

Bab ini menjelaskan tentang metode yang digunakan untuk mendapatkan data dan metode untuk pengembangan sistem yang telah dibuat serta kerangka berpikir tugas akhir ini.

### **BAB IV IMPLEMENTASI**

Bab ini menjelaskan tentang proses implementasi dari metode yang digunakan untuk menyelesaikan permasalahan penelitian.

### **BAB V HASIL DAN PEMBAHASAN**

Bab ini membahas tentang hasil yang telah didapat dari proses implementasi dan eksperimen yang telah dilakukan pada bab sebelumnya.

### **BAB VI KESIMPULAN DAN SARAN**

Bab ini menjelaskan tentang kesimpulan dari hasil yang telah didapat dan menjawab semua pokok permasalahan yang dirancang, serta saran-saran yang digunakan untuk penelitian selanjutnya.

## **BAB II**

### **LANDASAN TEORI**

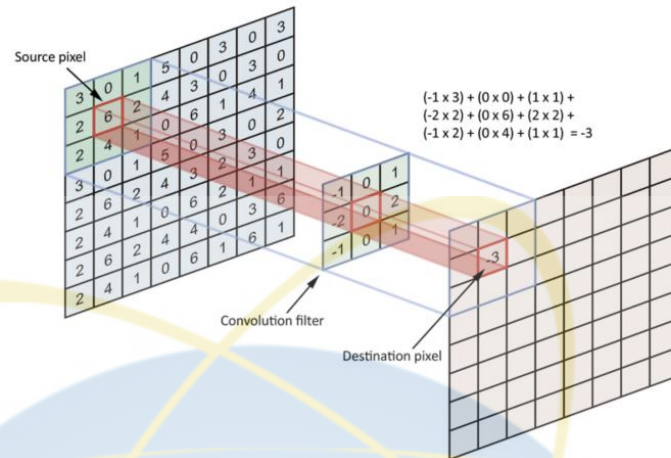
#### **2.1 CNN**

Convolutional Neural Network (CNN) pertama kali diperkenalkan oleh Yann LeCun pada sekitar tahun 1980. CNN adalah salah satu metode deep learning yang paling sering digunakan untuk memproses data visual (Roy et al., 2020). CNN berperan penting dalam sejarah dari deep learning, CNN merupakan contoh sukses penerapan cara kerja otak ke dalam bentuk machine learning. CNN juga merupakan salah satu model pertama yang dapat bekerja dengan baik dalam penggunaan komersial (Goodfellow et al., 2016).

CNN mengatasi masalah yang ada pada MLP yaitu apabila sebuah gambar yang digunakan untuk melatih model berada pada posisi yang berbeda dengan gambar yang digunakan untuk validasi, maka model MLP akan mengklasifikasikan gambar validasi tersebut berbeda dengan gambar yang digunakan untuk melatih model. CNN dapat menyelesaikan masalah ini dengan menggunakan sebuah filter pendeteksi yang fungsinya mendeteksi gambar di posisi manapun. CNN terdiri dari 3 jenis layer yaitu convolution layer, pooling layer, fully-connected layer.

##### **2.1.1 Convolution Layer**

Convolution layer merupakan layer yang terdiri dari neuron yang membentuk filter dengan ukuran panjang dan tinggi. Input yang didapatkan layer ini akan diteruskan pada sebuah filter yang akan menghasilkan output berupa feature map.

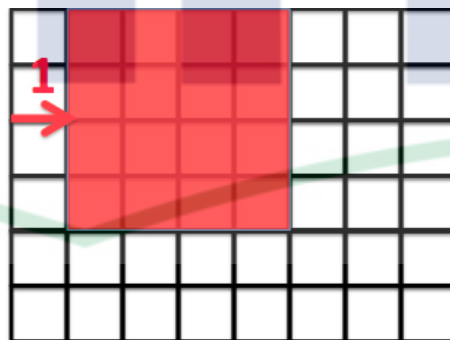


Gambar 2.1. Convolution Layer  
(Sumber: freecodecamp.org)

Output yang dihasilkan oleh convolution layer akan diteruskan ke Pooling layer untuk dikurangi dimensinya.

#### 2.1.1.1 Stride

Stride adalah parameter pada convolution layer yang berfungsi untuk menentukan jumlah pergeseran filter pada convolution layer. Semakin besar parameter stride maka semakin besar pula pergeseran yang dilakukan filter yang akan menghasilkan waktu komputasi yang lebih kecil namun informasi yang didapatkan lebih sedikit.

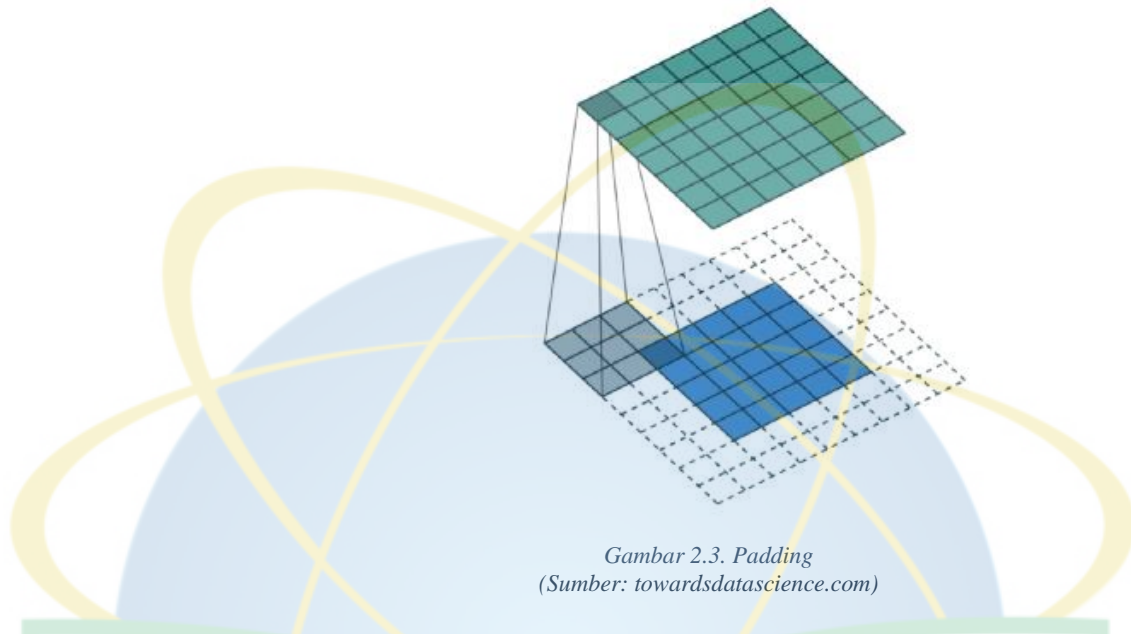


Gambar 2.2. Stride  
(Sumber: medium.com)

#### 2.1.1.2 Padding

Padding digunakan untuk memperbesar dimensi dari convolution layer. Tujuan dari penggunaan padding adalah supaya

tiap pixel dari convolution layer akan dikunjungi dalam jumlah yang sama.



Penggunaan padding dapat mengurangi resiko over-fitting dikarenakan seluruh pixel dikunjungi secara merata, namun output dari padding juga akan menghasilkan feature map yang lebih besar.

$$output = \frac{W - N + 2P}{S} + 1$$

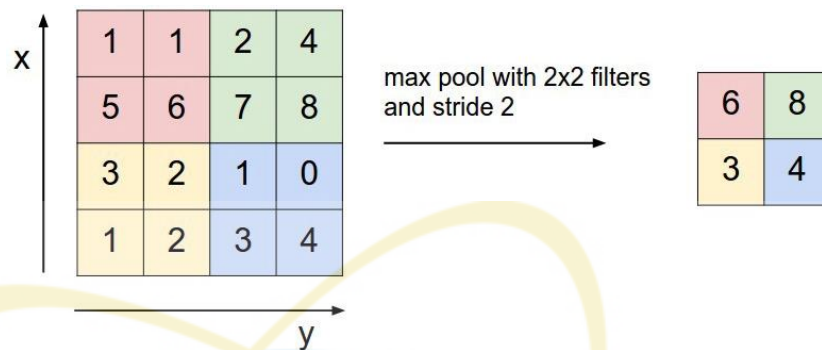
Dimana:

- W = Panjang / Tinggi input
- N = Panjang / Tinggi filter
- P = Zero Padding
- S = Stride

### 2.1.2 Pooling Layer

Pooling layer memiliki cara kerja yang sama seperti convolution layer, yang membedakan adalah filter yang digunakan pooling layer bukan bertujuan untuk mendeteksi sesuatu melainkan antara mencari nilai terbesar atau rata-rata yang biasa disebut *Max Pooling* dan *Average Pooling*.





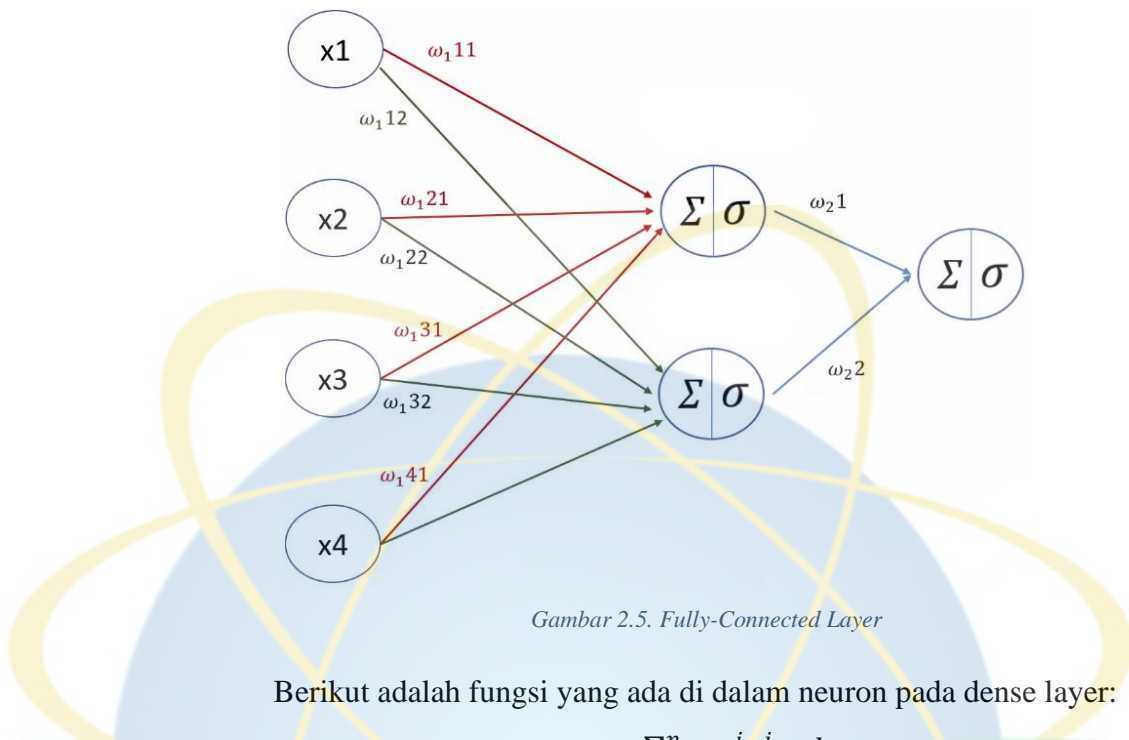
Gambar 2.4. Pooling Layer dengan Max Pooling  
(Sumber: [machinelearning.mipa.ugm.ac.id](http://machinelearning.mipa.ugm.ac.id))

Tujuan dari penggunaan pooling adalah mengurangi dimensi dari output convolution layer atau disebut feature map, sehingga dapat mengurangi parameter yang harus diubah dan mempercepat proses pelatihan model.

### 2.1.3 Fully-Connected Layer

Fully-Connected (FC) layer merupakan layer yang bertugas untuk mengklasifikasikan data berdasarkan output yang dihasilkan dari convolution dan pooling layer. FC layer menerima input berupa 1D array sehingga output dari layer sebelumnya harus diubah ke dalam bentuk 1D array terlebih dahulu.

Setiap neuron yang ada pada FC layer terdiri dari fungsi regresi dan activation function. Fungsi regresi digunakan untuk memprediksi output sedangkan activation function digunakan untuk menentukan output untuk neuron selanjutnya.



Gambar 2.5. Fully-Connected Layer

Berikut adalah fungsi yang ada di dalam neuron pada dense layer:

$$y = \sum_{i=0}^n w^i x^i + b$$

Dimana:

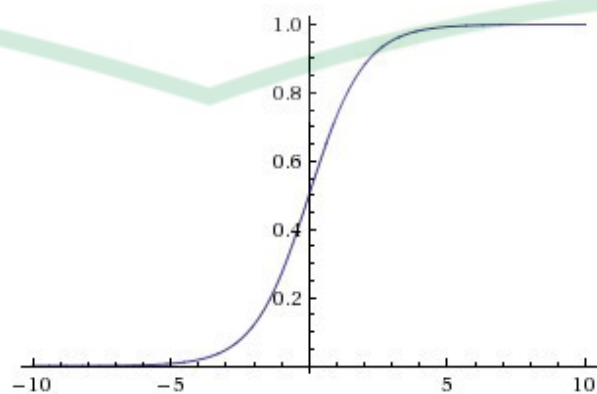
$W$  = Weight

$X$  = Input dari neuron sebelumnya

$B$  = Bias

Terdapat beberapa jenis activation function pada FC layer, 3 diantaranya yang paling sering digunakan adalah *Sigmoid*, *Softmax*, dan *ReLU*.

### Sigmoid

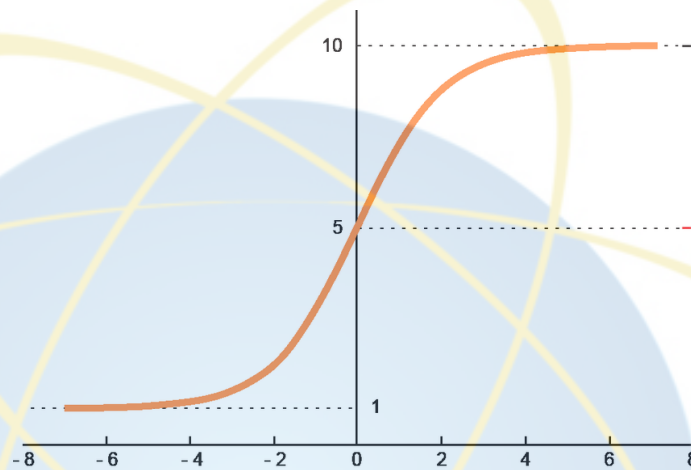


Gambar 2.6. Sigmoid Function

Sigmoid memiliki fungsi untuk mengubah logits menjadi 0 dan 1, sehingga Sigmoid cocok untuk mengklasifikasikan dataset binary.

$$S(x) = \frac{1}{1 + e^{-x}}$$

### Softmax

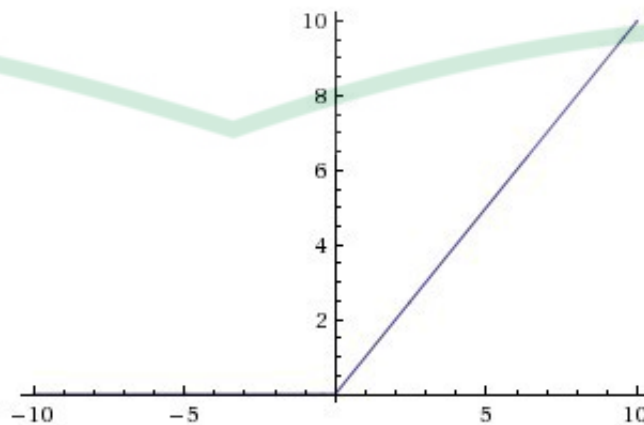


Gambar 2.7. Softmax Function

Softmax memiliki fungsi untuk mengubah logits menjadi 3 output, sehingga Softmax cocok untuk dijadikan activation function untuk dataset dengan 3 atau lebih kategori. Softmax juga menjadikan outputnya berupa probabilitas dengan total seluruhnya adalah 1.

$$S(y) = \frac{e^{y_i}}{\sum_{j=1}^n e^{y_j}}$$

### ReLU



Gambar 2.8. ReLu Function

ReLU atau Rectified Linear Unit adalah salah satu activation function yang populer digunakan. ReLU biasa digunakan pada convolution layer ataupun FC layer. ReLU membatasi output negatif menjadi 0 sehingga ReLU hanya mengeluarkan output positif. Fungsi ReLU selain mudah dipahami juga cepat dalam performa sehingga banyak dipilih dalam pelatihan model AI.

$$f(x) = \max(0, x)$$

## 2.2 VGG16

VGG16 adalah salah satu VGGnet model yang menggunakan 16 layer sebagai model arsitekturnya. VGG16 normalnya menggunakan 5 convolutional blocks yang kemudian terhubung ke 3 MLP classifiers. Layer output menggunakan *sigmoid activation function* apabila terdapat 2 atau kurang kategori, dan *softmax activation function* apabila terdapat 3 atau lebih kategori pada dataset (Hridayami et al., 2019).

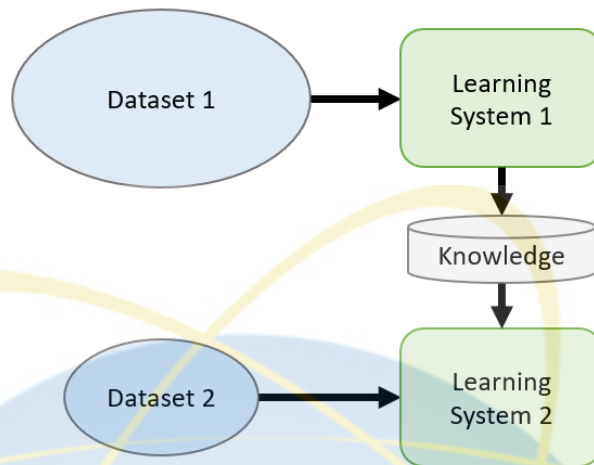


Gambar 2.9 Visualisasi Model VGG16

VGG16 memenangkan *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) tahun 2014 pada posisi pertama dan kedua, dimana pada challenge tersebut digunakan 1000 kategori dan 1000 data pada tiap kategori.

## 2.3 Transfer Learning

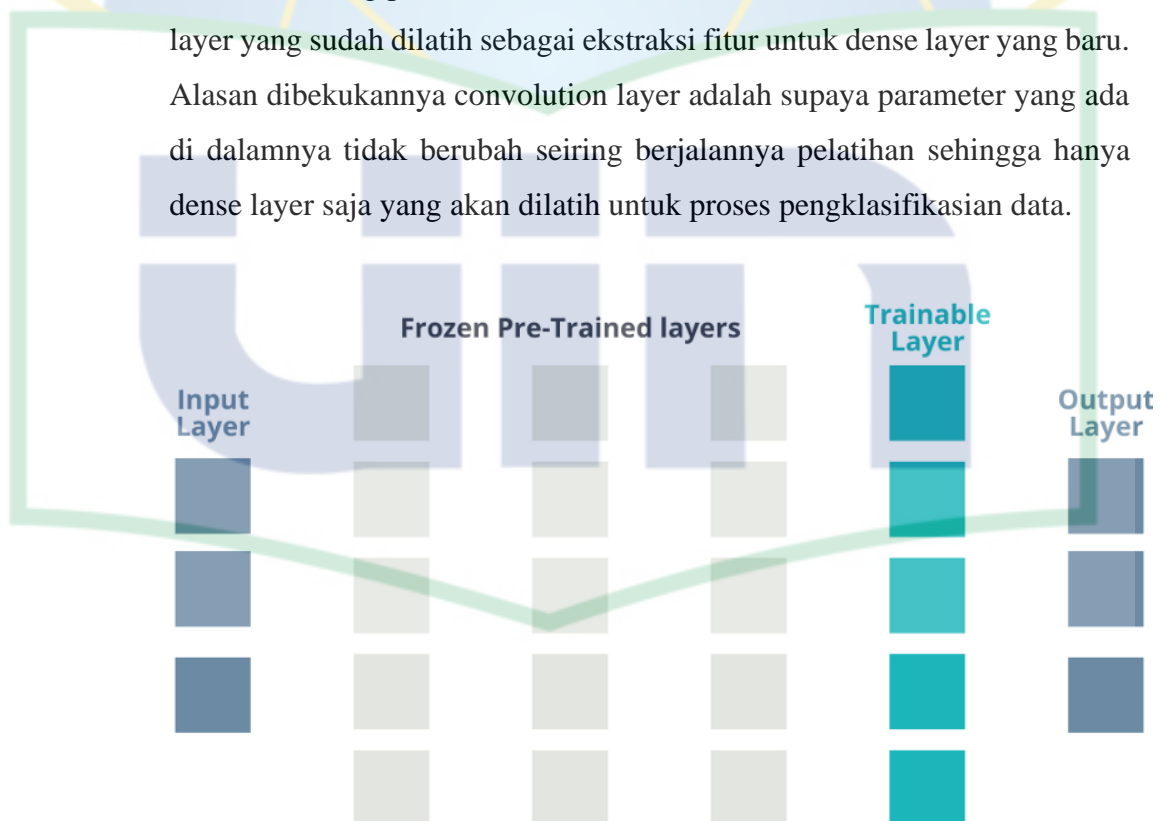
Transfer learning adalah sebuah metode dimana model yang sudah dilatih oleh dataset sebelumnya digunakan kembali pada model dengan dataset yang berbeda (Rozaqi et al., 2021). Metode transfer learning cocok digunakan pada model dengan dataset yang sedikit (Risdiyati & Luthfiarta, 2021).



Gambar 2.10 Visualisasi Transfer Learning

### 2.3.1 Fixed Feature Extractor

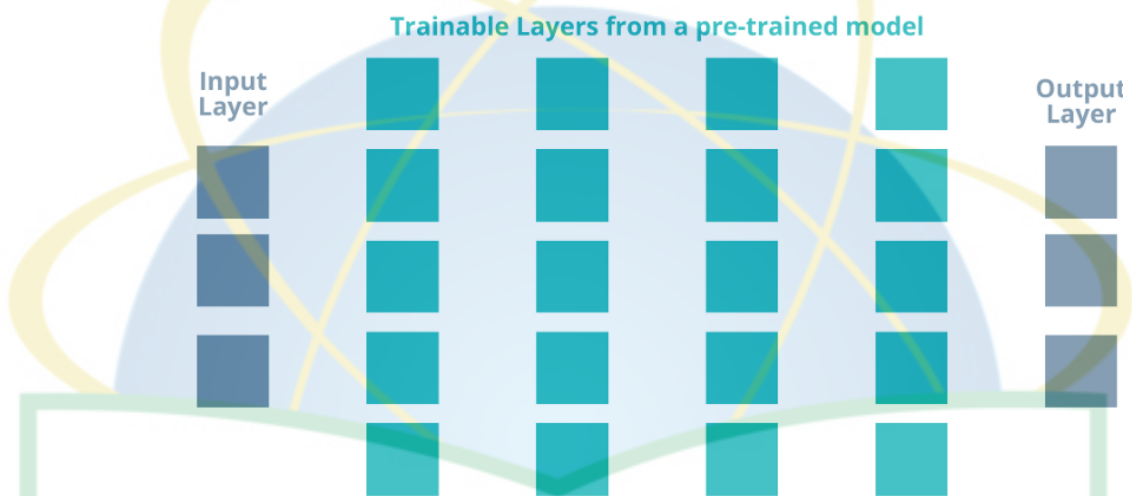
Fixed feature extractor merupakan sebuah salah satu cara penerapan transfer learning pada sebuah model. Metode ini membekukan convolution layer yang sudah dilatih sebagai ekstraksi fitur untuk dense layer yang baru. Alasan dibekukannya convolution layer adalah supaya parameter yang ada di dalamnya tidak berubah seiring berjalannya pelatihan sehingga hanya dense layer saja yang akan dilatih untuk proses pengklasifikasian data.



Gambar 2.11 Visualisasi Fixed Feature Extractor  
(Sumber: [thumarakesh.medium.com](https://thumarakesh.medium.com))

### 2.3.2 Fine-tuning

Fine-tuning merupakan sebuah metode dimana dataset yang sudah dilatih dengan transfer learning dilatih ulang dengan layer yang tidak dibekukan. Hal ini dapat meningkatkan performa dari model yang sudah terlatih, namun fine-tuning harus dilakukan dengan learning rate yang kecil supaya tidak merusak parameter yang sudah dipelajari oleh model.

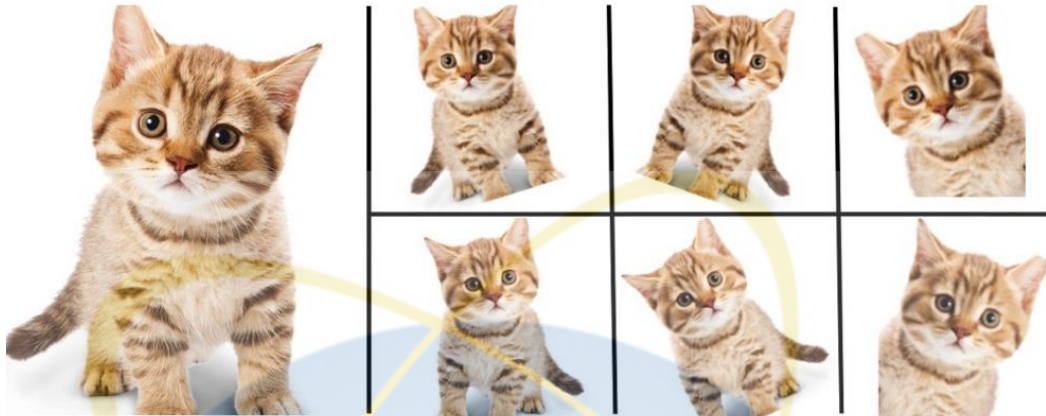


Gambar 2.12 Visualisasi Fine-Tuning  
(Sumber: [thuwarakesh.medium.com](https://thuwarakesh.medium.com))

### 2.4 Data Augmentation

Data augmentation adalah sebuah metode untuk memperbanyak dataset. Deep learning merupakan metode yang membutuhkan dataset dengan jumlah besar untuk mencapai performa yang baik, data augmentation dapat membantu hal tersebut. Selain memperbanyak dataset, data augmentation juga dapat meminimalisir resiko over-fitting dikarenakan beragamnya dataset yang dihasilkan oleh metode data augmentation.

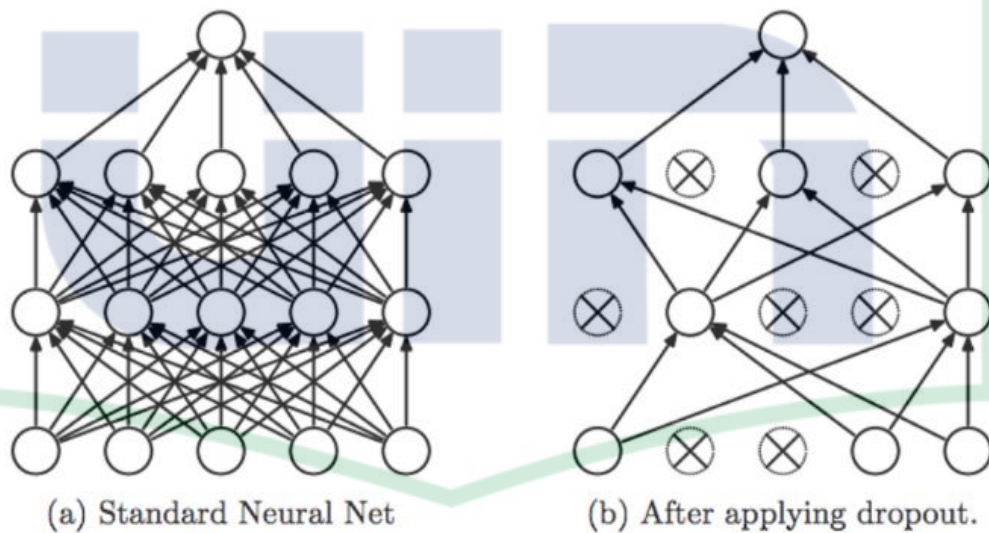




Gambar 2.13. Visualisasi Data Augmentation  
(Sumber: kdnuggets.com)

## 2.5 Dropout

Dropout adalah sebuah metode untuk tidak menggunakan sebagian neuron secara acak pada sebuah layer dengan kemungkinan  $p$ . Tujuan dihapusnya neuron pada layer supaya neuron berikutnya tidak bergantung pada neuron tertentu, hal ini dapat meminimalisir over-fitting.



Gambar 2.14. Visualisasi Layer Sebelum dan Sesudah Dropout  
(Sumber: www.tech-quantum.com)

## 2.6 Tensorflow

Tensorflow adalah sebuah open source platform yang dikembangkan oleh google pada tahun 2015 untuk machine learning. Tensorflow menyediakan

berbagai API pada berbagai bahasa pemrograman yang memudahkan proses pembuatan model AI.



# TensorFlow

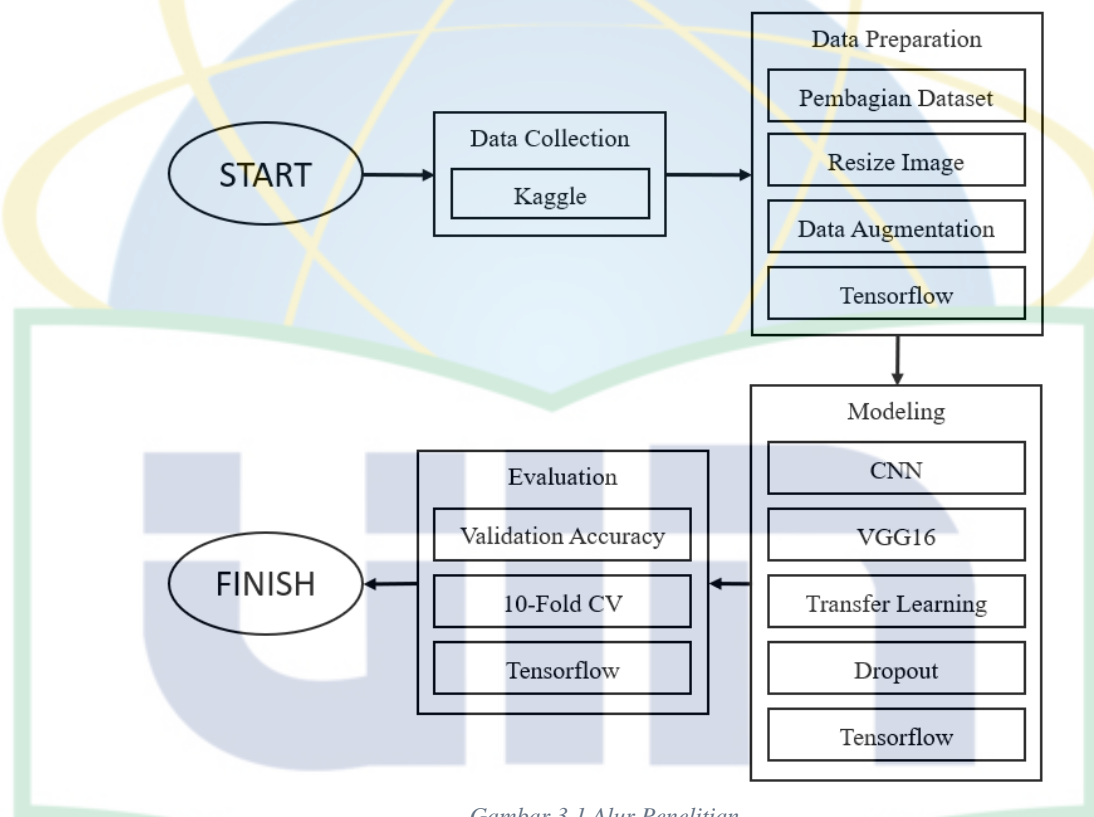
*Gambar 2.15. Logo Tensorflow  
(Sumber: dlmacedo.com)*



## BAB III METODOLOGI PENELITIAN

### 3.1 Metode Implementasi

Implementasi deep learning VGG16 dengan transfer learning pada deteksi penyakit tanaman singkong disusun melalui beberapa tahapan untuk memudahkan proses penelitian. Alur tahapan metode yang digunakan dapat dilihat pada gambar 3.1 di bawah.



*Gambar 3.1 Alur Penelitian*

### 3.2 Metode Pengumpulan Data

Tahap pengumpulan data dilakukan dengan cara studi pustaka dan literatur, penulis mencari referensi-referensi yang berkaitan dengan penelitian. Pencarian referensi dilakukan secara online melalui internet. Literatur sejenis yang didapatkan berupa skripsi, jurnal, dan produk sejenis yang kemudian dikaji sehingga penelitian ini dapat menambah wilayah berkembangnya keilmuan dan dapat menjadi pelengkap dari penelitian atau alat yang sudah ada, kemudian dipilih berbagai informasi yang dibutuhkan dalam penelitian ini.

NO	JUDUL	PENULIS	TAHUN	METODE	JENIS LITERATUR	PROSES	HASIL
1	Metode <i>Forward Chaining</i> untuk Deteksi Penyakit Pada Tanaman Kentang	<ul style="list-style-type: none"> <li>Ahmad Nazarudin</li> <li>Iskandar</li> </ul>	2020	Forward Chaining	Jurnal	Merancang sistem pakar menggunakan metode forward chaining untuk mendeteksi penyakit pada tanaman kentang	Sistem pakar berhasil dibuat dan diimplementasikan pada aplikasi
2	Deteksi Penyakit Pada Daun Pakcoy Dengan Pengolahan Citra Menggunakan Metode <i>Convolutional Neural Network</i>	<ul style="list-style-type: none"> <li>Susila, Michael Ferdy</li> <li>Irawan, Budhi</li> <li>Setianingsih, Casi</li> </ul>	2020	CNN	Proceeding	Mendeteksi penyakit pada tanaman pakcoy menggunakan metode CNN dengan jenis penyakit yaitu daun berlubang dan leaf miner, menggunakan partisi data latih dan uji 9:1	Akurasi yang diperoleh dari model CNN yang dibuat dalam mengklasifikasikan penyakit pada tanaman pakcoy adalah 86,67%
3	SVM Classifier Based Grape Leaf Disease Detection	<ul style="list-style-type: none"> <li>Padol, Pranjali B.</li> <li>Yadav, Anjali A.</li> </ul>	2016	SVM	Proceeding	Menggunakan metode SVM untuk mendeteksi penyakit pada anggur, bagian yang terkena penyakit akan dideteksi menggunakan K-means clustering lalu kemudian melakukan ekstraksi fitur dari warna dan tekstur. Lalu pengklasifikasian penyakit menggunakan SVM	SVM berhasil mendeteksi penyakit pada anggur dengan akurasi sebesar 88.89%

4	Image-Based Plant Disease Detection: A Comparison of Deep Learning and Classical Machine Learning Algorithms	<ul style="list-style-type: none"> <li>• Radovanovic, Drasko</li> <li>• Dukanovic, Slobodan</li> </ul>	2020	<ul style="list-style-type: none"> <li>• SVM</li> <li>• KNN</li> <li>• FCNN</li> <li>• CNN</li> </ul>	Proceeding	Membandingkan algoritma machine learning dan deep learning pada pengklasifikasian penyakit pada tanaman	CNN memiliki akurasi tertinggi dengan nilai 99.3%. SVM 91.7%. FCNN 91.4% dan k-NN 78%
5	Performance of deep learning vs machine learning in plant leaf disease detection	<ul style="list-style-type: none"> <li>• Sujatha, R.</li> <li>• Chatterjee, Jyotir Moy</li> <li>• Jhanjhi, N. Z.</li> <li>• Brohi, Sarfraz Nawaz</li> </ul>	2020	<ul style="list-style-type: none"> <li>• Random Forest</li> <li>• SGD</li> <li>• SVM</li> <li>• VGG-19</li> <li>• Inception-V3</li> <li>• VGG-16</li> </ul>	Jurnal Q3	Membandingkan algoritma machine learning dan deep learning diantaranya random forest, SGD, SVM, VGG-19, Inception-V3, dan VGG-16 pada dataset Citrus Leaf Disease	VGG16 memiliki akurasi tertinggi dengan nilai precision 0.896, F1 dan CA 0.895 dan AUC 0.97
6	Case-Based Reasoning (CBR) pada Sistem Pakar Identifikasi Hama dan Penyakit Tanaman Singkong dalam Usaha Meningkatkan Produktivitas Tanaman Pangan	<ul style="list-style-type: none"> <li>• Minarni., Indra, W.</li> <li>• Wenda, H, .</li> </ul>	2017	CBR	Jurnal	Mengaplikasikan sistem pakar menggunakan metode CBR untuk mengidentifikasi penyakit pada tanaman singkong lalu membandingkan hasilnya menggunakan metode k-NN	CBR memiliki akurasi 100% dalam mengklasifikasikan penyakit pada tanaman singkong sedangkan k-NN memiliki akurasi sebesar 67.65%

7	Improving Deep Learning using Generic Data Augmentation	<ul style="list-style-type: none"> <li>Taylor, Luke</li> <li>Nitschke, Geoff</li> </ul>	2017	CNN	Jurnal arXiv	Meningkatkan akurasi metode CNN dengan metode data augmentation	Dengan menggunakan data augmentation akurasi pada metode CNN meningkat hingga 13.82%
8	Implementasi Transfer Learning pada Algoritma Convolutional Neural Network Untuk Identifikasi Penyakit Daun Kentang	<ul style="list-style-type: none"> <li>Rozaqi, Jalil</li> <li>Sunyoto, Andi</li> <li>Arief, Rudyanto</li> </ul>	2021	<ul style="list-style-type: none"> <li>ResNet-50</li> <li>Inception-V3</li> <li>VGG-16</li> </ul>	Proceeding	Mengimplementasikan transfer learning pada arsitektur CNN sederhana, VGG16, Inception-V3, dan ResNet-50	CNN sederhana mendapatkan akurasi sebesar 80%, Inception-V3 78%, VGG16 95%, dan ResNet-50 78%
9	VGG16 Transfer Learning Architecture for Salak Fruit Quality Classification	<ul style="list-style-type: none"> <li>Rismiyati, Rismiyati</li> <li>Luthfiarta, Ardytha</li> </ul>	2021	<ul style="list-style-type: none"> <li>VGG16</li> </ul>	Jurnal S3	Mengimplementasikan transfer learning model VGG16.	Metode transfer learning dengan arsitektur VGG16 dapat mengklasifikasikan buah salak dengan akurasi sebesar 95,83%.
10	Sequence-Dropout Block for Reducing Overfitting	<ul style="list-style-type: none"> <li>Qian, Ledan</li> <li>Hu, Libing</li> <li>Zhao, Li</li> <li>Wang, Tao</li> </ul>	2020	<ul style="list-style-type: none"> <li>VGG16</li> <li>ResNet50</li> </ul>	Jurnal Q1	Mengimplementasikan dropout dan batch normalization pada 2 arsitektur CNN yaitu VGG16 dan ResNet50	Performa terbaik didapati oleh VGG16 + Sequence-Dropout



	Problem in Image Classification	<ul style="list-style-type: none"> <li>• Jiang, Runhua</li> </ul>				dengan akurasi sebesar 89.04%
11	Fish species recognition using VGG16 deep convolutional neural network	<ul style="list-style-type: none"> <li>• Hridayami, Praba</li> <li>• Putra, I. Ketut Gede Darma</li> <li>• Wibawa, Kadek Suar</li> </ul>	2019	<ul style="list-style-type: none"> <li>• VGG16</li> </ul>	Jurnal Q3	<p>Mengklasifikasikan ikan menggunakan VGG16 pada 4 jenis dataset yaitu RGB color space image, canny filter image, blending image, blending image + RGN image.</p> <p>Performa terbaik didapat pada dataset jenis blending image + RGB dengan acceptance rate (GAR) sebesar 96.4%, diikuti dengan RGB color space image dengan GAR 92.4%, lalu canny filter image dengan GAR 80.4%, dan terakhir blending image dengan GAR 75.6%</p>
12	A comparative study of fine-tuning deep learning models for plant disease identification	<ul style="list-style-type: none"> <li>• Yingchun, Liu</li> <li>• Too, Edna Chebet</li> <li>• Yujian, Li</li> <li>• Njuki, Sam</li> </ul>	2018	<ul style="list-style-type: none"> <li>• ResNet-50</li> <li>• ResNet-101</li> <li>• ResNet-152</li> <li>• VGG16</li> <li>• Inception V4</li> <li>• DenseNet</li> </ul>	Jurnal Q1	<p>Mengimplementasikan berbagai arsitektur model deep learning menggunakan transfer learning dan fine tuning untuk mengklasifikasikan 38 jenis penyakit pada tanaman</p> <p>Model DenseNet dengan memperoleh akurasi terbaik di antara model lainnya dengan akurasi sebesar 99.75% pada epoch 30 .</p>

Tabel 3.1 Literature Review

NO	JUDUL	PENULIS	ARSITEKTUR	DATA AUGMENTATION	TRANSFER LEARNING	FINE TUNING	DROPOUT
-	Implementasi Deep Learning VGG16 dengan Transfer Learning pada Deteksi Penyakit Tanaman Singkong	<ul style="list-style-type: none"> <li>Muhammad Fikri Syahid</li> </ul>	VGG16	<ul style="list-style-type: none"> <li>Rotation Range</li> <li>Width Shift Range</li> <li>Height Shift Range</li> <li>Horizontal Flip</li> <li>Vertical Flip</li> <li>Shear Range</li> <li>Zoom Range</li> </ul>	ImageNet	✓	50%
1	Metode <i>Forward Chaining</i> untuk Deteksi Penyakit Pada Tanaman Kentang	<ul style="list-style-type: none"> <li>Ahmad Nazarudin</li> <li>Iskandar</li> </ul>	-	-	-	-	-
2	Deteksi Penyakit Pada Daun Pakcoy Dengan Pengolahan Citra Menggunakan Metode <i>Convolutional Neural Network</i>	<ul style="list-style-type: none"> <li>Susila, Michael Ferdy</li> <li>Irawan, Budhi</li> <li>Setianingsih, Casi</li> </ul>	CNN	-	-	-	-
3	SVM Classifier Based Grape Leaf Disease Detection	<ul style="list-style-type: none"> <li>Padol, Pranjali B.</li> <li>Yadav, Anjali A.</li> </ul>	-	-	-	-	-
4	Image-Based Plant Disease Detection: A Comparison of Deep Learning and Classical	<ul style="list-style-type: none"> <li>Radovanovic, Drasko</li> <li>Dukanovic, Slobodan</li> </ul>	GoogleNet	-	ImageNet	-	-

	Machine Learning Algorithms						
5	Performance of deep learning vs machine learning in plant leaf disease detection	<ul style="list-style-type: none"> <li>• Sujatha, R.</li> <li>• Chatterjee, Jyotir Moy</li> <li>• Jhanjhi, N. Z.</li> <li>• Brohi, Sarfraz Nawaz</li> </ul>	<ul style="list-style-type: none"> <li>• VGG19</li> <li>• Inception-V3</li> <li>• VGG16</li> </ul>	-	-	-	-
6	Case-Based Reasoning (CBR) pada Sistem Pakar Identifikasi Hama dan Penyakit Tanaman Singkong dalam Usaha Meningkatkan Produktivitas Tanaman Pangan	<ul style="list-style-type: none"> <li>• Minarni., Indra, W.</li> <li>• Wenda, H, .</li> </ul>	-	-	-	-	-
7	Improving Deep Learning using Generic Data Augmentation	<ul style="list-style-type: none"> <li>• Taylor, Luke</li> <li>• Nitschke, Geoff</li> </ul>	CNN	<ul style="list-style-type: none"> <li>• Baseline</li> <li>• Flipping</li> <li>• Rotating</li> <li>• Cropping</li> <li>• Color Jittering</li> <li>• Edge Enhancement</li> <li>• Fancy PCA</li> </ul>	Xavier	-	-
8	Implementasi Transfer Learning pada Algoritma Convolutional Neural Network Untuk Identifikasi Penyakit Daun Kentang	<ul style="list-style-type: none"> <li>• Rozaqi, Jalil</li> <li>• Sunyoto, Andi</li> <li>• Arief, Rudyanto</li> </ul>	<ul style="list-style-type: none"> <li>• VGG19</li> <li>• Inception-V3</li> <li>• VGG16</li> <li>• ResNet-50</li> </ul>	-	✓	-	-

9	VGG16 Transfer Learning Architecture for Salak Fruit Quality Classification	<ul style="list-style-type: none"> <li>• Rismiyati, Rismiyati</li> <li>• Luthfiarta, Ardytha</li> </ul>	VGG16	-	ImageNet	-	-
10	Sequence-Dropout Block for Reducing Overfitting Problem in Image Classification	<ul style="list-style-type: none"> <li>• Qian, Ledan</li> <li>• Hu, Libing</li> <li>• Zhao, Li</li> <li>• Wang, Tao</li> <li>• Jiang, Runhua</li> </ul>	<ul style="list-style-type: none"> <li>• VGG16</li> <li>• ResNet-50</li> </ul>	<ul style="list-style-type: none"> <li>• Color Jittering</li> <li>• Random Crop</li> <li>• Scale Jittering</li> <li>• Flip</li> <li>• Random Erasing</li> <li>• Noise Injection</li> </ul>	✓	-	50%
11	Fish species recognition using VGG16 deep convolutional neural network	<ul style="list-style-type: none"> <li>• Hridayami, Praba</li> <li>• Putra, I. Ketut Gede Darma</li> <li>• Wibawa, Kadek Suar</li> </ul>	VGG16	<ul style="list-style-type: none"> <li>• Zoom Range</li> <li>• Rotation Range</li> <li>• Horizontal Flip</li> <li>• Vertical Flip</li> </ul>	ImageNet	-	-
12	A comparative study of fine-tuning deep learning models for plant disease identification	<ul style="list-style-type: none"> <li>• Yingchun, Liu</li> <li>• Too, Edna Chebet</li> <li>• Yujian, Li</li> <li>• Njuki, Sam</li> </ul>	<ul style="list-style-type: none"> <li>• Inception-V4</li> <li>• VGG16</li> <li>• ResNet-50</li> <li>• ResNet-101</li> <li>• ResNet-152</li> <li>• DenseNet-121</li> </ul>	-	ImageNet	✓	✓

Tabel 3.2 Perbandingan Literature Review

## **BAB IV IMPLEMENTASI**

### **4.1 Data Collection**

Pada tahap ini, penulis akan mencari dataset untuk digunakan pada penelitian ini. Data yang akan digunakan berupa gambar tanaman singkong beserta keterangan penyakitnya. Penulis akan mencari dataset publik dari internet, website pertama yang akan menjadi sasaran penulis adalah *kaggle*.

Penulis menemukan dataset yang cocok untuk penelitian ini pada website *kaggle* yang berjudul *Cassava Leaf Disease Merged*. Dataset ini berukuran 3.13GB dengan file berupa gambar sebanyak 26.337 gambar beserta file *.csv* yang berisi keterangan penyakit atau kelas untuk setiap gambar. Kelas yang terdapat pada dataset tanaman singkong ini di antaranya adalah *Bacterial Blight*, *Brown Streak Disease*, *Green Mottle*, *Mosaic Disease*, *Healthy*.

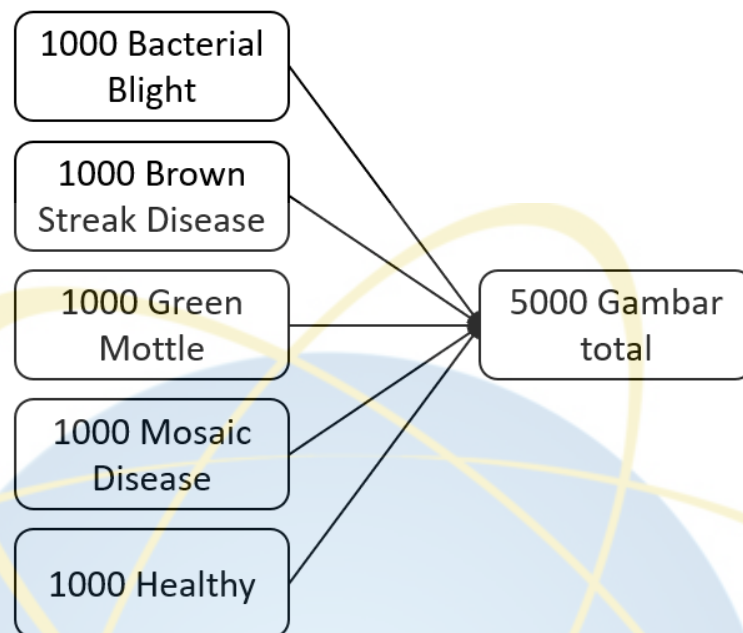
### **4.2 Data Preparation**

Setelah mendapatkan dataset yang akan digunakan, selanjutnya adalah melakukan persiapan pada dataset untuk digunakan pada pelatihan model.

#### **4.2.1 Pemilihan Dataset**

Dataset dengan total 26.337 gambar yang dibagi ke dalam 5 kelas akan diambil 1000 dataset dari setiap kelasnya, alasan dari tidak digunakan keseluruhan dari dataset ini adalah tidak diperlukannya dataset melebihi dari 1000 untuk mendapatkan performa model yang baik, pernyataan ini dirujuk dari challenge yang diadakan oleh *ImageNet* dimana pada setiap challenge akan digunakan 1000 data untuk setiap kategori.

Pemilihan 1000 dataset ini masih dilakukan secara acak dikarenakan tidak ada parameter untuk mengukur gambar yang bagus atau yang tidak, proses manual dapat dilakukan untuk memilih dataset yang bagus namun proses tersebut akan sangat menggunakan banyak waktu.



Gambar 4.1. Visualisasi Pembagian Dataset

#### 4.2.2 Data Augmentation

Setelah dilakukan pembagian dataset selanjutnya adalah proses data augmentasi dimana setiap dataset akan didiklasi sebanyak 7 jenis dengan proses yang berbeda di antaranya yaitu *rotation\_range*, *width\_shift\_range*, *height\_shift\_range*, *horizontal\_flip*, *vertical\_flip*, *shear\_range*, *zoom\_range*. Penelitian ini akan menerapkan perbandingan model dengan dan tanpa data augmentasi.









Normal



Rotation Range



 <p>Width Shift Range</p>	 <p>Height Shift Range</p>
 <p>Horizontal Flip</p>	 <p>Vertical Flip</p>
 <p>Shear Range</p>	 <p>Zoom Range</p>

Tabel 4.1 Jenis Data Augmentasi yang diimplementasikan



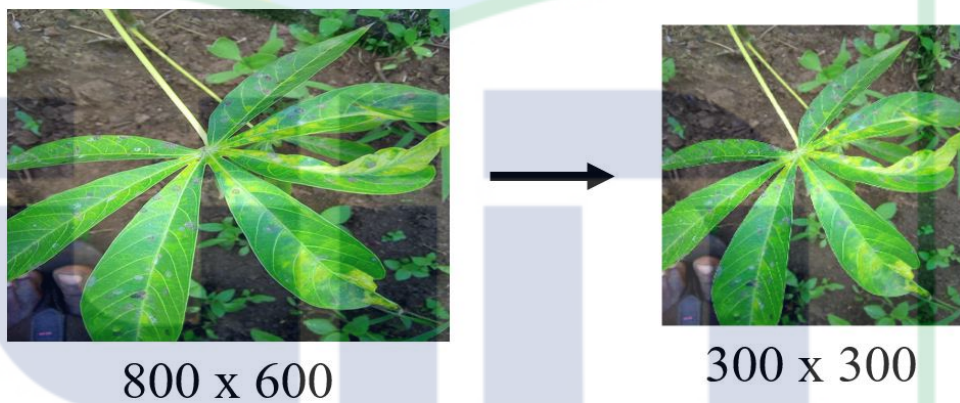
```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    vertical_flip = True,
    shear_range = 0.2,
    zoom_range=0.2,
    fill_mode = 'nearest')
```

Data Augmentation  
menggunakan  
ImageDataGenerator  
dari library tensorflow

Gambar 4.2 Kode Implementasi Data Augmentation

### 4.2.3 Resize Image

Selain data augmentation, hal lain yang dilakukan pada tahap persiapan data adalah merubah ukuran gambar. Hal ini dikarenakan gambar akan dijadikan sebagai input pada model yang akan dilatih. Model akan menerima input gambar dengan ukuran tertentu, maka dari itu diperlukan adanya penentuan ukuran gambar untuk seluruh dataset.



Gambar 4.3 Visualisasi Resize Image

Seperti yang terlihat pada Gambar 4.2 di sisi kiri adalah dataset dengan ukuran yang sebenarnya yaitu  $800 \times 600 \text{ px}$ , dan di sisi kanan adalah dataset yang sudah dilakukan perubahan ukuran menjadi  $300 \times 300 \text{ px}$ . Pengurangan ukuran pada dataset akan membantu saat proses pelatihan model terutama pada perangkat dengan spesifikasi hardware yang rendah.

```

train_generator = train_datagen.flow_from_dataframe(
    dataframe=df_train,
    x_col="image_id",
    y_col="id",
    directory=data_dir,
    target_size=(300, 300),
    batch_size=10,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_dataframe(
    dataframe=df_test,
    x_col="image_id",
    y_col="id",
    directory=data_dir,
    target_size=(300, 300),
    batch_size=10,
    class_mode='categorical')

```

Resize image untuk dataset training dan validation menggunakan ImageDataGenerator pada parameter *target\_size*

Gambar 4.4 Kode Implementasi Resize Image

#### 4.2.4 Pembagian Dataset

Dataset gambar singkong yang berjumlah 5000 dengan 1000 dari tiap kelasnya akan diberlakukan 10-fold cross validation sehingga dataset akan dibagi menjadi 9:1 bagian. Pembagian 9:1 pada dataset akan terdiri dari 4500 data training dan 500 data validasi. Pengaturan pembagian partisi dataset akan dilakukan secara otomatis dari library *sklearn*.

```

kfold = KFold(n_splits=10, shuffle=True)

acc_per_fold = []
loss_per_fold = []
fold_no = 1

for train, test in kfold.split(df):

    df_train = df.iloc[train, :]
    df_test = df.iloc[test, :]

```

*KFold* merupakan merupakan fungsi dari library *sklearn*.

List *acc\_per\_fold* digunakan untuk menyimpan akurasi dari tiap fold  
List *loss\_per\_fold* digunakan untuk menyimpan loss dari tiap fold

*fold\_no* akan digunakan untuk menyimpan keterangan fold saat perulangan

*kfold.split* akan menerima dataframe yang akan dibagi secara otomatis menjadi *train* dan *test*.

Gambar 4.5 Pembagian dataset dan 10-fold cross validation

### 4.3 Modeling

#### 4.3.1 Perancangan Model

Setelah dilakukan pre-processing pada dataset selanjutnya adalah perancangan model yang akan digunakan yaitu VGG16 dengan weight dari *ImageNet* dan CNN.

#### 4.3.1.1 Perancangan Model CNN

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 298, 298, 32)	896
max_pooling2d (MaxPooling2D)	(None, 149, 149, 32)	0
conv2d_1 (Conv2D)	(None, 147, 147, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 73, 73, 64)	0
flatten (Flatten)	(None, 341056)	0
dense (Dense)	(None, 64)	21827648
dense_1 (Dense)	(None, 32)	2080
dense_2 (Dense)	(None, 5)	165
Total params: 21,849,285		
Trainable params: 21,849,285		
Non-trainable params: 0		

Gambar 4.6 Summary Model CNN

```

model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3),
        activation='relu', input_shape=(300,300,3)),
    tf.keras.layers.MaxPooling2D((2, 2)),

    tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3),
        activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(5, activation='softmax')
])

model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
    optimizer=tf.optimizers.Adam(),
    metrics=['accuracy'])

```

Gambar 4.7 Kode Implementasi Model CNN

Perancangan model CNN dapat dilihat pada *Gambar 4.6* dan *Gambar 4.7*. Model ini memiliki 5 layer diantaranya adalah 2 convolution layer dan 3 dense layer. Layer pertama pada convolution layer juga merupakan input layer dengan ukuran 300,300,3 dimana 300,300 adalah ukuran gambar dan 3 adalah nilai RGB untuk warna pada gambar. Setiap convolution layer

menggunakan *max pooling* dengan ukuran  $2 \times 2$  dengan stride 2 untuk mengecilkan ukuran convolution pada layer berikutnya.

Dense layer hanya menerima input berupa vector 1 dimensi sehingga layer flatten digunakan sebelum dense layer. Dense layer memiliki jumlah neuron sebanyak 64,32,5 secara berturut-turut. Jumlah neuron pada dense layer didapatkan dari trial and error. Jumlah neuron 64,32 merupakan jumlah neuron yang biasa dipakai pada model CNN pada umumnya sehingga digunakan sebagai jumlah neuron pada model ini, sedangkan 5 neuron pada dense layer terakhir digunakan untuk output dari klasifikasi. Penggunaan *softmax activation function* pada dense layer terakhir dikarenakan jumlah kategori kelas pada dataset lebih dari 3.

Selanjutnya model akan disusun menggunakan *categorical\_crossentropy* sebagai loss function dikarenakan dataset yang digunakan memiliki lebih dari 3 kelas lalu model juga menggunakan *adam optimizer* sebagai optimizer serta pengukuran performa model akan berdasarkan *accuracy*.

#### 4.3.1.2 Perancangan Model VGG16

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 9, 9, 512)	14714688
global_average_pooling2d (G1	(None, 512)	0
dense_1 (Dense)	(None, 4096)	2101248
dropout (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 4096)	16781312
dropout_1 (Dropout)	(None, 4096)	0
dense (Dense)	(None, 5)	20485
Total params: 33,617,733		
Trainable params: 18,903,045		
Non-trainable params: 14,714,688		

Gambar 4.8 Summary Model VGG16

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 300, 300, 3)]	0
block1_conv1 (Conv2D)	(None, 300, 300, 64)	1792
block1_conv2 (Conv2D)	(None, 300, 300, 64)	36928
block1_pool (MaxPooling2D)	(None, 150, 150, 64)	0
block2_conv1 (Conv2D)	(None, 150, 150, 128)	73856
block2_conv2 (Conv2D)	(None, 150, 150, 128)	147584
block2_pool (MaxPooling2D)	(None, 75, 75, 128)	0
block3_conv1 (Conv2D)	(None, 75, 75, 256)	295168
block3_conv2 (Conv2D)	(None, 75, 75, 256)	590080
block3_conv3 (Conv2D)	(None, 75, 75, 256)	590080
block3_pool (MaxPooling2D)	(None, 37, 37, 256)	0
block4_conv1 (Conv2D)	(None, 37, 37, 512)	1180160
block4_conv2 (Conv2D)	(None, 37, 37, 512)	2359808
block4_conv3 (Conv2D)	(None, 37, 37, 512)	2359808
block4_pool (MaxPooling2D)	(None, 18, 18, 512)	0
block5_conv1 (Conv2D)	(None, 18, 18, 512)	2359808
block5_conv2 (Conv2D)	(None, 18, 18, 512)	2359808
block5_conv3 (Conv2D)	(None, 18, 18, 512)	2359808
block5_pool (MaxPooling2D)	(None, 9, 9, 512)	0
Total params: 14,714,688		
Trainable params: 0		
Non-trainable params: 14,714,688		

Gambar 4.9 Summary Convolution Layer pada Model VGG16



```

VGG16_MODEL=tf.keras.applications.VGG16(
    input_shape=(300,300,3),
    include_top=False,
    weights='imagenet')

VGG16_MODEL.trainable = False

global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
prediction_layer = tf.keras.layers.Dense(5,activation='softmax')

model = tf.keras.Sequential([
    VGG16_MODEL,
    global_average_layer,
    tf.keras.layers.Dense(4096, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(4096, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    prediction_layer
])

model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
              optimizer=tf.optimizers.Adam(),
              metrics=['accuracy'])

```

Gambar 4.10 Kode Implementasi Model VGG16

Perancangan model VGG16 dapat dilihat pada Gambar 4.8, Gambar 4.9 dan Gambar 4.10. Layer *VGG16\_MODEL* merupakan 13 convolution layer dari arsitektur VGG16 tanpa adanya fully-connected (FC) / dense layer. Alasan ditiadakan FC / dense layer dikarenakan model akan digunakan untuk mengklasifikasikan dataset yang baru dengan jumlah kategori yang berbeda. Weight yang digunakan oleh 13 convolution layer pada model ini berasal dari *ImageNet*. Convolution layer dari *VGG16\_MODEL* dibekukan untuk menjaga parameter weight yang sudah didapat dari *ImageNet*, penjelasan lebih detail terdapat pada bagian pelatihan model.

Setelah convolution layer terdapat 2 fully-connected / dense layer yang digunakan pada model dengan jumlah neuron untuk keduanya sebanyak 4096 dan layer output dengan 5 neuron menggunakan *softmax* activation function. Metode dropout akan diterapkan pada setiap dense layer selain output dengan probabilitas 50%.

Selanjutnya model disusun dengan loss function berupa *categorical crossentropy* dikarenakan jumlah kategori pada dataset yang digunakan lebih banyak dari atau sama dengan 3. Optimizer yang digunakan adalah

*adam optimizer* dengan accuracy sebagai metrics pengukuran performa model.

#### 4.3.2 Pelatihan Model

Pada tahap ini penulis akan menerapkan beberapa jenis model dengan penerapan metode yang berbeda namun dengan parameter yang sama. Tujuan dari digunakannya parameter dan dataset yang sama adalah supaya pengukuran performa dari setiap model diberlakukan secara adil.

Berikut adalah daftar model yang akan dilakukan uji coba performa:

NO	ARSITEKTUR MODEL	TRANSFER LEARNING	FINE TUNING	DATA AUGMENTATION	DROPOUT
1	VGG16	✓	✓	✓	✓
2	VGG16	✓	✓	✓	-
3	VGG16	✓	✓	-	✓
4	VGG16	✓	-	✓	✓
5	VGG16	✓	-	-	✓
6	VGG16	✓	✓	-	-
7	VGG16	✓	-	✓	-
8	VGG16	✓	-	-	-
9	CNN	-	-	✓	✓
10	CNN	-	-	✓	-
11	CNN	-	-	-	✓
12	CNN	-	-	-	-

Tabel 4.2 Daftar Implementasi Model

Model pada Tabel 4.2 akan diterapkan menggunakan parameter yang sama. Fine-tuning akan diimplementasikan menggunakan *learning rate* sebesar 0.00001 dan *dropout layer* akan menggunakan *probabilitas* 50%. Seluruh model akan menggunakan *epoch* sebanyak 20, namun ada perbedaan implementasi *epoch* pada arsitektur VGG16 dan CNN terutama pada saat melakukan fine-tuning pada transfer learning VGG16.



#### 4.3.2.1 Pelatihan Model CNN dan VGG16 (Tanpa Fine-Tuning)

```
history = model.fit(
    train_generator,
    epochs=20,
    steps_per_epoch=450,
    validation_data=validation_generator,
    validation_steps=50,
    verbose=1
)
```

Gambar 4.11 Kode Implementasi Pelatihan Model CNN dan VGG16

Dapat dilihat pada *Gambar 4.11* pelatihan model CNN dan VGG16 (tanpa fine-tuning) dilakukan dengan 20 epoch, 450 steps\_per\_epoch, dan 50 validation\_steps. Setiap proses epoch menandakan seluruh dataset sudah dijadikan sebagai input pada model. 450 steps\_per\_epoch dan 50 validation\_steps didapat dari pembagian dari jumlah dataset dan batch\_size.

```
train_generator = train_datagen.flow_from_dataframe(
    dataframe=df_train,
    x_col="image_id",
    y_col="id",
    directory=data_dir,
    target_size=(300, 300),
    batch_size=10,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_dataframe(
    dataframe=df_test,
    x_col="image_id",
    y_col="id",
    directory=data_dir,
    target_size=(300, 300),
    batch_size=10,
    class_mode='categorical')
```

Gambar 4.12 Deklarasi batch\_size saat proses Resize Image

Parameter batch\_size dideklarasikan pada train\_generator dan validation\_generator. Generator ini sudah pernah ditunjukkan pada tahap Data Preparation tepatnya pada tahap Resize Image. Generator ini memiliki fungsi untuk membaca dataset dari sebuah direktori dengan pengaturan kelas yang terdapat pada data frame df\_train dan df\_test.

```
Found 4500 validated image filenames belonging to 5 classes.
Found 500 validated image filenames belonging to 5 classes.
```

Gambar 4.13 Output dari *train\_generator* dan *validation\_generator*

Dapat dilihat pada Gambar 4.13 bahwa *train\_generator* menemukan 4500 dataset training sedangkan *validation\_generator* menemukan 500 dataset validasi. Kedua generator memiliki *batch\_size* sebesar 10 yang berarti *steps\_per\_epoch* untuk masing-masing *train* dan *validation* adalah  $4500/10 = 450$ , dan  $500/10 = 50$ .

Parameter *batch\_size* merupakan jumlah data (gambar) yang dimasukkan ke dalam model dalam satu waktu, pengaturan *batch\_size* dapat membantu performa model pada hardware dengan spesifikasi rendah. Dikarenakan hardware penulis tidak sanggup menggunakan *batch\_size* 30 maka penulis menurunkan *batch\_size* menjadi 10.

#### 4.3.2.2 Pelatihan Model VGG16 dengan Fine-Tuning

Pelatihan model VGG16 dengan fine-tuning memiliki proses yang berbeda dengan pelatihan model tanpa fine-tuning. Fine-tuning hanya dapat dilakukan apabila model menggunakan transfer learning sehingga hanya model VGG16 dengan transfer learning yang dapat menggunakan fine-tuning.

Pada tahap perancangan model VGG16, convolution layer dibekukan dan hanya dense layer yang dapat dilatih, hal ini disebut sebagai *fixed feature extractor* yang berarti model hanya menggunakan convolution layer sebagai *feature extractor* tanpa melatihnya. Fixed feature extractor perlu dilakukan sebelum dilakukan fine-tuning, hal ini disebabkan dense layer masih memiliki random value yang perlu dilatih terlebih dahulu sebelum model melatih keseluruhan layer.

Pelatihan model dengan fine-tuning akan tetap menggunakan 20 epoch namun pelatihan akan dibagi menjadi 2 yaitu 10 epoch pertama dan kedua.

```
history = model.fit(
    train_generator,
    epochs=10,
    steps_per_epoch=450,
    validation_data=validation_generator,
    validation_steps=50,
    verbose=1
)
```

Gambar 4.14 Kode Implementasi Pelatihan 10 Epoch Pertama

Setelah dilakukan pelatihan 10 epoch pertama dengan *fixed feature extractor* maka dense layer sudah memiliki nilai yang sudah terlatih untuk melakukan proses fine-tuning. Langkah selanjutnya adalah melatih seluruh layer termasuk convolution layer dan dense layer dengan cara tidak membekukan convolution layer dan menyusun ulang model lalu melatihnya kembali dengan epoch awalan 10 yaitu epoch terakhir pada saat *fixed feature extractor*.

```
VGG16_MODEL.trainable = True
model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
              optimizer=tf.optimizers.Adam(1e-5),
              metrics=['accuracy'])

history_fine = model.fit(
    train_generator,
    steps_per_epoch=450,
    epochs=20,
    initial_epoch=history.epoch[-1],
    validation_data=validation_generator,
    validation_steps=50,
    verbose=1
)
```

Gambar 4.15 Kode Implementasi Pelatihan 10 Epoch Kedua

Pelatihan 10 epoch kedua menggunakan model yang sudah disusun ulang dan juga learning rate yang kecil, pada Gambar 4.15 dapat dilihat learning rate yang digunakan untuk fine-tuning pada VGG16 adalah  $1e-5$  atau  $0.00001$ . Learning rate yang kecil

digunakan supaya perubahan parameter saat pelatihan model terjadi secara perlahan.

#### 4.4 Evaluation

Setelah seluruh daftar model pada *Tabel 4.2* berhasil dilatih, evaluasi performa model akan dilihat dari *validation accuracy* atau akurasi validasi. Alasan digunakannya *validation accuracy* untuk parameter pengukuran performa model adalah dikarenakan dataset yang *balance* yaitu dari 500 dataset untuk validasi terdiri dari 100 data dari tiap kelas. Model dengan akurasi validasi terbaik akan diuji kembali dengan berbagai parameter dari metode yang dimilikinya, dengan harapan performa dari model yang terbaik dapat menjadi lebih baik lagi.

Model yang terbaik diantara yang terbaik akan dilakukan uji coba epoch untuk mencari titik konvergen dari model tersebut. Setelah didapatkan titik konvergen, model akan dievaluasi menggunakan *10-fold cross validation* untuk evaluasi terakhir pada model.



## BAB V

### HASIL DAN PEMBAHASAN

Langkah pertama pada penelitian ini secara garis besar adalah persiapan dataset. Setelah dataset sudah siap untuk diproses maka selanjutnya adalah perancangan dan pelatihan model. Daftar model pada *Tabel 4.2* akan dirancang, dilatih dan dievaluasi performanya. Penulis sudah menjelaskan proses implementasi dari awal persiapan data hingga akhir penelitian atau evaluasi pada BAB 4, selanjutnya adalah pembahasan hasil dari implementasi tersebut.

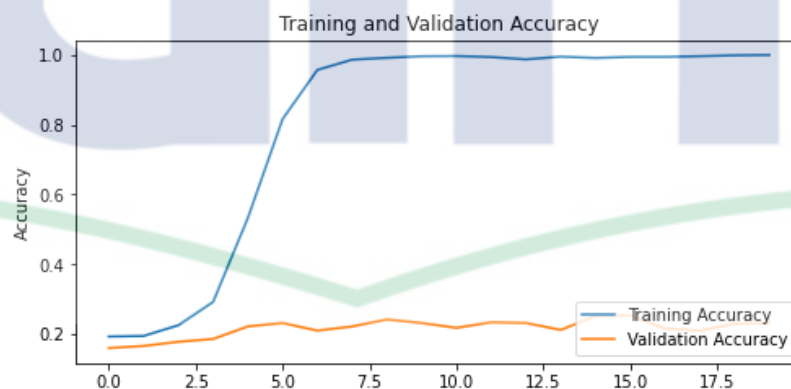
#### 5.1 Pencarian Model Terbaik

Demi memudahkan pembacaan model, penulis membuat kode untuk merepresentasikan metode pada model di antaranya adalah:

1. DA = Data Augmentation
2. TL = Transfer Learning
3. FT = Fine Tuning
4. DO = Dropout

Berikut adalah detail pelatihan model pada *Tabel 4.2*:

##### 5.1.1 CNN

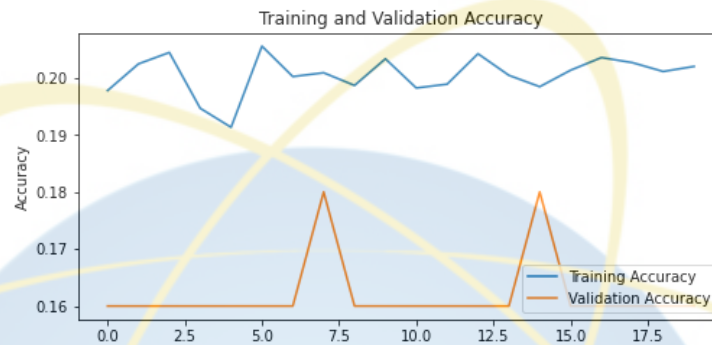


Gambar 5.1 Evaluasi Model CNN

Model CNN tanpa adanya penerapan metode lain menghasilkan performa dengan akurasi training dan validasi sebesar 100% dan 23.2% berturut-turut. Dapat dilihat pada *Gambar 5.1* bahwa grafik pada pelatihan

model ini menghasilkan overfitting yang sangat buruk, perbandingan antara akurasi training dan validasi mencapai 76.8%.

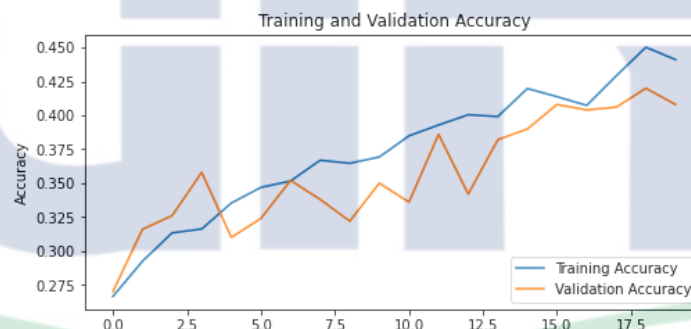
### 5.1.2 CNN (DO)



Gambar 5.2 Evaluasi Model CNN dengan DO

Model CNN dengan adanya tambahan dropout 50% mengakibatkan underfitting. Dapat dilihat pada *Gambar 5.2* bahwa akurasi training berawal di sekitar 20% dan tidak meningkat sama sekali, begitu juga dengan akurasi validasi berawal dari 16% dan hanya meningkat ke 18% sesekali.

### 5.1.3 CNN (DA)

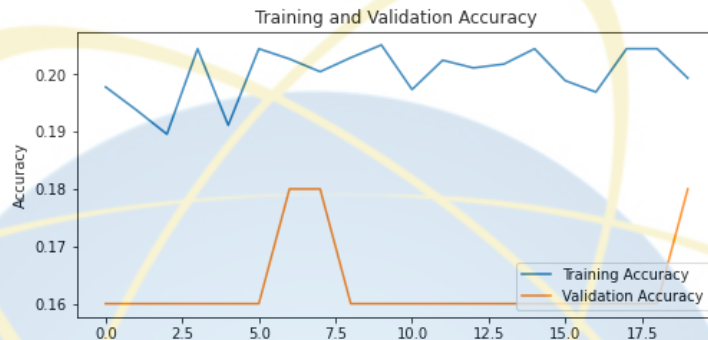


Gambar 5.3 Evaluasi Model CNN dengan DA

Model CNN dengan tambahan implementasi data augmentation mendapatkan performa yang lebih baik dibanding CNN tanpa tambahan metode dan CNN dengan dropout seperti yang terlihat pada *Gambar 5.3*. Model ini menghasilkan akurasi training dan validasi sebesar 44.1% dan 40.8% berturut-turut. Hal ini membuktikan bahwa penggunaan data augmentation lebih berpengaruh pada pengurangan overfitting

dibandingkan dengan dropout. Sampai sini dapat disimpulkan bahwa dataset lebih berperan penting pada performa model dari pada pengaturan layer seperti dropout.

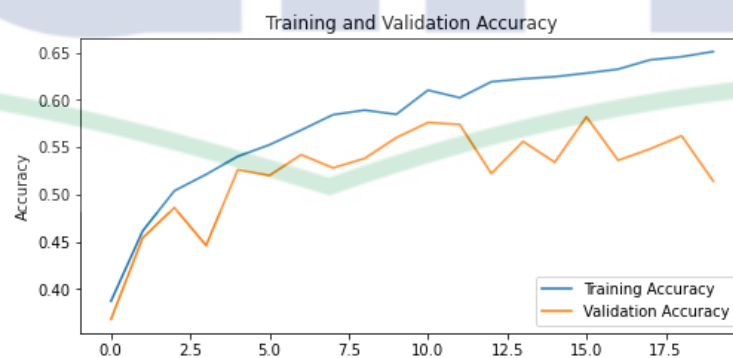
#### 5.1.4 CNN (DA, DO)



Gambar 5.4 Evaluasi Model CNN dengan DA dan DO

Model CNN dengan tambahan metode dropout dan data augmentation pada Gambar 5.4 menunjukkan underfitting yang tidak berbeda jauh dari model CNN (DO) pada Gambar 5.2. Model ini menghasilkan akurasi training dan validasi sebesar 19.9% dan 18% berturut-turut. Dari 4 model ini dapat disimpulkan bahwa penggunaan dropout pada model CNN untuk klasifikasi image pada dataset penelitian ini belum menghasilkan peningkatan pada performa model.

#### 5.1.5 VGG16 (TL)



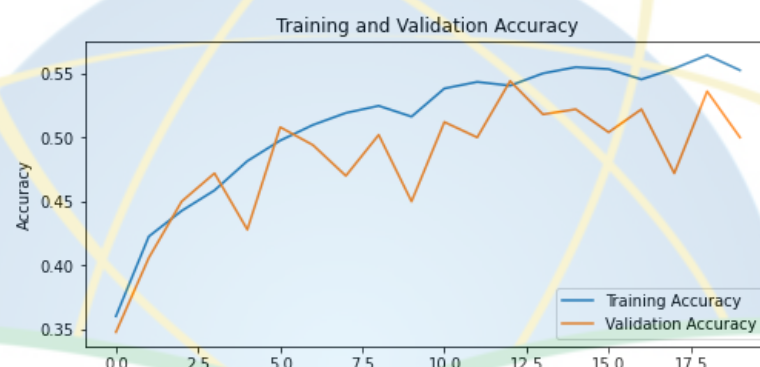
Gambar 5.5 Evaluasi Model VGG16 dengan TL

Model VGG16 dengan transfer learning menunjukkan performa yang jauh lebih baik dibandingkan model CNN sebelumnya. Model ini



menghasilkan akurasi training dan validasi sebesar 65.1% dan 51.4% berturut-turut. Perbedaan antara akurasi training dan validasi mencapai 13.7% dan dapat dilihat pada *Gambar 5.5* bahwa model ini menghasilkan hasil yang overfitting. Meski hasil dari model ini overfitting namun model ini memiliki performa yang jauh lebih bagus dari model CNN bahkan model ini belum ditambahkan metode seperti data augmentation dan dropout.

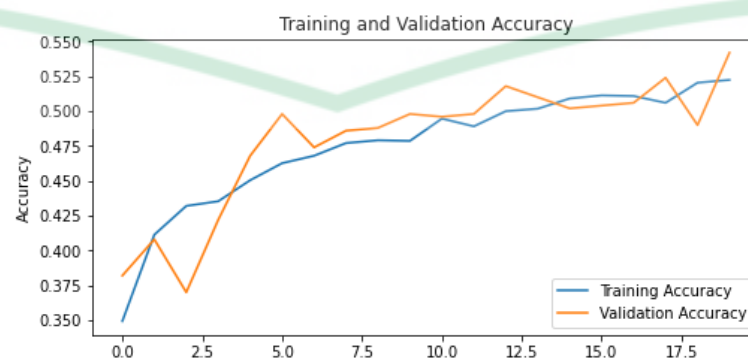
#### 5.1.6 VGG16 (DA, TL)



Gambar 5.6 Evaluasi Model VGG16 dengan DA dan TL

Model VGG16 dengan transfer learning dan data augmentation menghasilkan hasil yang lebih baik dari segi overfitting dibandingkan dengan model VGG16 dengan transfer learning saja, namun model ini memiliki akurasi validasi yang lebih rendah. Model ini menghasilkan akurasi training dan validasi sebesar 55.2% dan 50% berturut-turut.

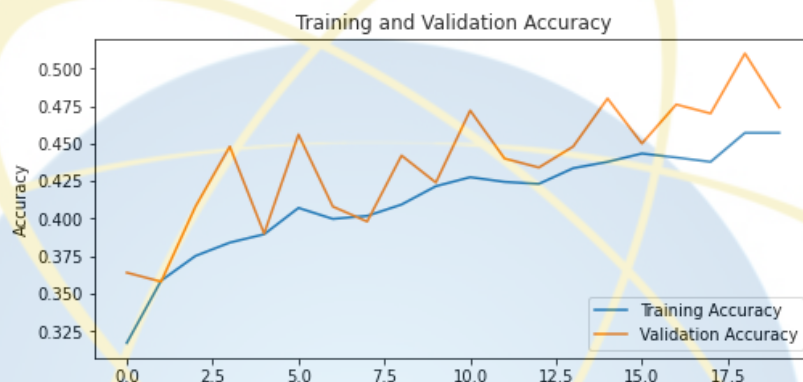
#### 5.1.7 VGG16 (TL, DO)



Gambar 5.7 Evaluasi Model VGG16 dengan TL dan DO

Model VGG16 dengan transfer learning dan dropout menghasilkan model yang balance dengan akurasi training dan validasi sebesar 52.2% dan 54.2% berturut-turut. Penggunaan dropout pada model VGG16 dengan transfer learning meningkatkan akurasi validasi dan mengurangi overfitting.

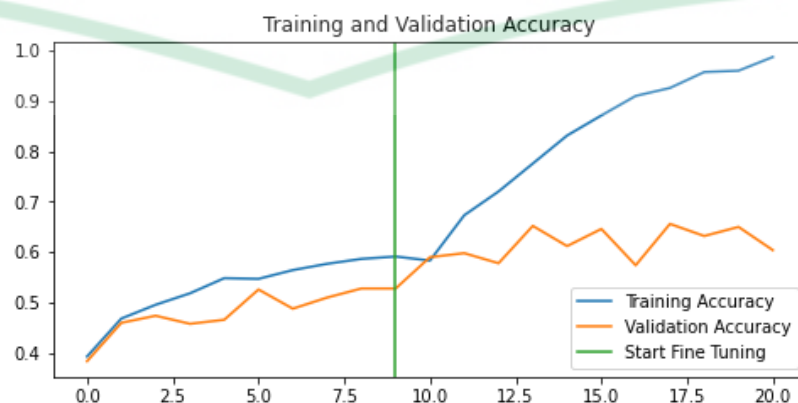
### 5.1.8 VGG16 (DA, TL, DO)



Gambar 5.8 Evaluasi Model VGG16 dengan DA, TL, dan DO

Model VGG16 dengan transfer learning, data augmentation, dan dropout menghasilkan model yang balance, namun dibandingkan dengan model VGG16 dengan transfer learning dan dropout model ini menghasilkan akurasi yang lebih rendah yaitu 47.4% sedangkan akurasi training sebesar 45.7%. Penggunaan dropout lebih berpengaruh pada peningkatan akurasi dan pengurangan overfitting pada model VGG16 dengan transfer learning.

### 5.1.9 VGG16 (TL, FT)

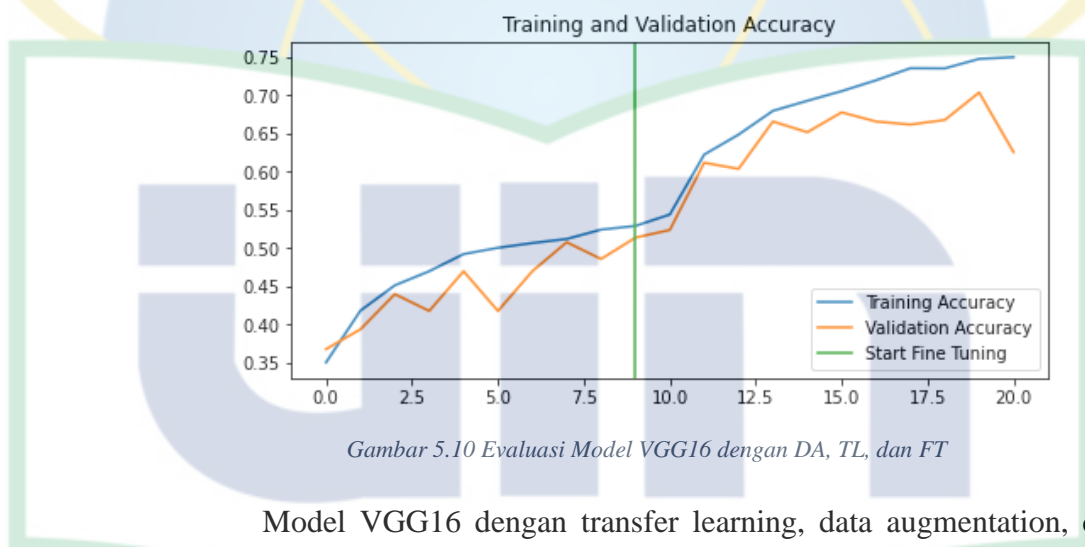


Gambar 5.9 Evaluasi Model VGG16 dengan TL dan FT

Model VGG16 dengan transfer learning dan fine-tuning menghasilkan akurasi training dan validasi sebesar 98.6% dan 60.4% berturut-turut. Dapat dilihat garis hijau pada *Gambar 5.9* yang menunjukkan mulainya proses fine-tuning pada model, fine-tuning sangat meningkatkan akurasi training pada model VGG16 dengan transfer learning.

Model ini adalah model dengan akurasi validasi tertinggi sejauh ini, namun model ini memiliki overfitting yang cukup buruk yang menghasilkan perbedaan antara akurasi training dan validasi sebesar 38.2%. Penggunaan fine-tuning pada model VGG16 dengan transfer learning sangat meningkatkan performa model, namun masih diperlukan metode lain untuk menanggulangi overfitting pada model ini.

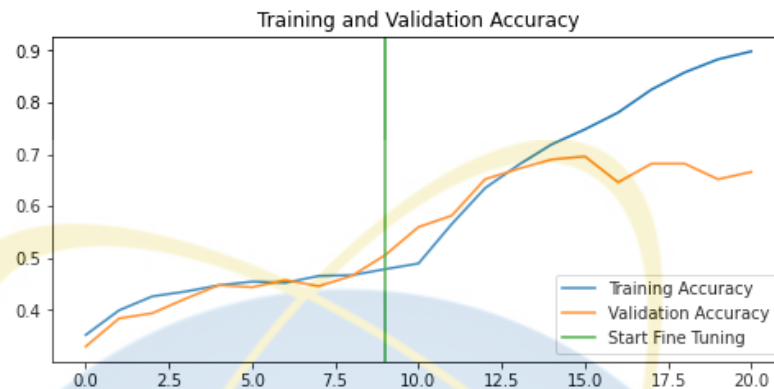
#### 5.1.10 VGG16 (DA, TL, FT)



*Gambar 5.10 Evaluasi Model VGG16 dengan DA, TL, dan FT*

Model VGG16 dengan transfer learning, data augmentation, dan fine-tuning menghasilkan hasil yang cukup baik namun sedikit overfitting pada sekitar epoch 17. Model ini menghasilkan akurasi training dan validasi sebesar 75% dan 62.6% berturut-turut. Se jauh ini akurasi validasi tertinggi diperoleh oleh model ini.

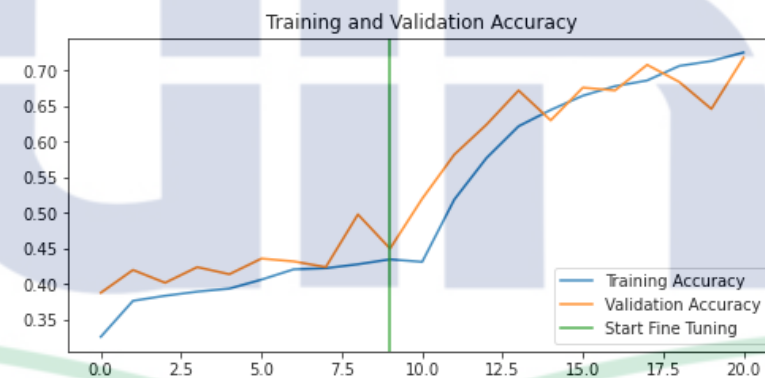
### 5.1.11 VGG16 (TL, FT, DO)



Gambar 5.11 Evaluasi Model VGG16 dengan TL, FT, dan DO

Model VGG16 dengan transfer learning, fine-tuning, dan dropout menghasilkan akurasi training dan validasi sebesar 89.8% dan 66.6% berturut-turut. Model ini memiliki akurasi validasi tertinggi sejauh ini, namun model ini juga menghasilkan hasil yang overfitting dengan perbedaan antara akurasi training dan validasi yaitu 23.2%.

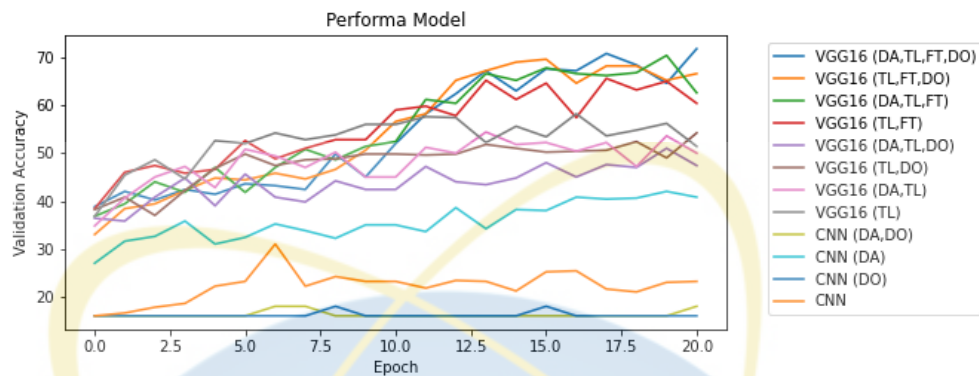
### 5.1.12 VGG16 (DA, TL, FT, DO)



Gambar 5.12 Evaluasi Model VGG16 dengan DA, TL, FT, dan DO

Model VGG16 dengan tambahan seluruh metode yaitu data augmentation, transfer learning, fine-tuning, dan dropout memiliki akurasi training dan validasi sebesar 72.5% dan 71.8% berturut-turut. Model ini memiliki hasil yang balance dengan akurasi validasi tertinggi di antara seluruh model yang diuji.

### 5.1.13 Ringkasan Pencarian Model Terbaik



Gambar 5.13 Ringkasan Performa Model

Pada Gambar 5.13 dapat dilihat performa dari seluruh model pada Tabel 4.2. Di antara seluruh model yang diujikan, model dengan arsitektur VGG16 dengan tambahan metode data augmentation, dropout, transfer learning, dan fine-tuning memiliki performa terbaik (VGG16 dengan DA, TL, FT, DO). Maka dari itu penulis akan menjadikan model ini sebagai model terbaik dan akan menerapkan pengujian parameter pada model ini.

ARSITEKTUR MODEL	TL	FT	DA	DO	STATUS	TRAINING ACCURACY	VALIDATION ACCURACY
VGG16	✓	✓	✓	✓	BALANCE	72.5%	71.8%
VGG16	✓	✓	-	✓	OVERFITTING	89.8%	66.6%
VGG16	✓	✓	✓	-	OVERFITTING	75.0%	62.6%
VGG16	✓	✓	-	-	OVERFITTING	98.6%	60.4%
VGG16	✓	-	-	✓	BALANCE	52.2%	54.2%
VGG16	✓	-	-	-	OVERFITTING	65.1%	51.4%
VGG16	✓	-	✓	-	OVERFITTING	55.2%	50.0%
VGG16	✓	-	✓	✓	BALANCE	45.7%	47.4%
CNN	-	-	✓	-	BALANCE	44.1%	40.8%
CNN	-	-	-	-	OVERFITTING	100.0%	23.2%
CNN	-	-	✓	✓	UNDERFITTING	19.9%	18.0%
CNN	-	-	-	✓	UNDERFITTING	20.0%	16.0%

Gambar 5.14 Tabel Ringkasan Performa Model

## 5.2 Uji Coba Parameter Model Terbaik

Uji coba parameter dilakukan pada model terbaik yang didapatkan dari pencarian model terbaik pada Tabel 4.2. Uji coba parameter dilakukan dengan

harapan model yang sudah baik akan menjadi lebih baik lagi. Model terbaik yang terdapat pada *Tabel 4.2* adalah model **VGG16 (DA, TL, FT, DO)**. Model ini memiliki parameter default yaitu:

1. Fine-tuning (Learning rate) : *0.00001*
2. Dropout : *50%*

Uji coba parameter akan mencoba untuk melatih ulang model ini dengan parameter yang berbeda, di antara parameter tersebut adalah:

NO	FINE TUNING	DROPOUT
1	<i>0.0001</i>	<i>25%</i>
2	<i>0.00001</i>	<i>50%</i>
3	<i>0.00005</i>	-

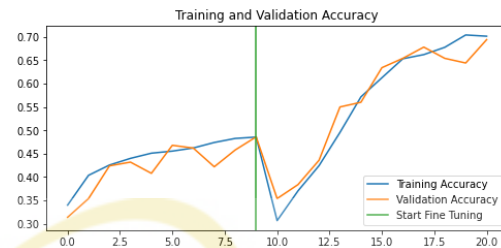
*Tabel 5.1 Ringkasan Uji Coba Parameter*

KODE	FT	DO	STATUS	TRAINING ACCURACY	VALIDATION ACCURACY
A	<i>0.00001</i>	<i>50%</i>	BALANCE	72.5%	71.8%
B	<i>0.0001</i>	<i>25%</i>	BALANCE	70.1%	69.4%
C	<i>0.00001</i>	<i>25%</i>	BALANCE	73.8%	68.6%
D	<i>0.00005</i>	<i>25%</i>	BALANCE	73.5%	67.2%
E	<i>0.0001</i>	<i>50%</i>	BALANCE	68.7%	66.8%
F	<i>0.00005</i>	<i>50%</i>	OVERFITTING	58.6%	51.2%

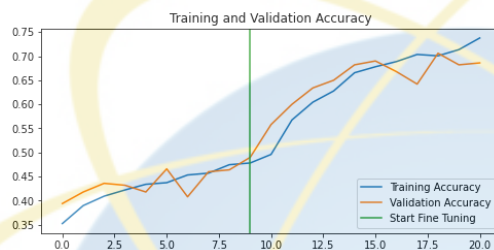
*Tabel 5.2 Tabel Ringkasan Hasil Uji Coba Parameter*



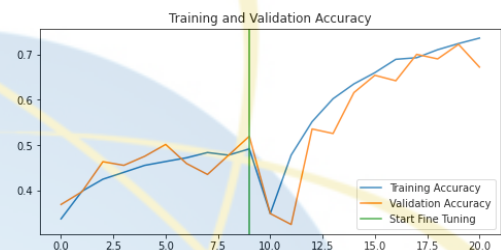
Gambar 5.15 Visualisasi Hasil Model A



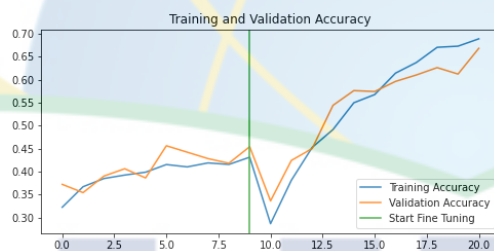
Gambar 5.16 Visualisasi Hasil Model B



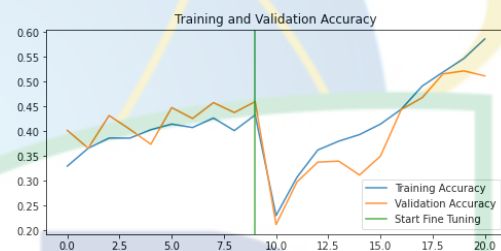
Gambar 5.17 Visualisasi Hasil Model C



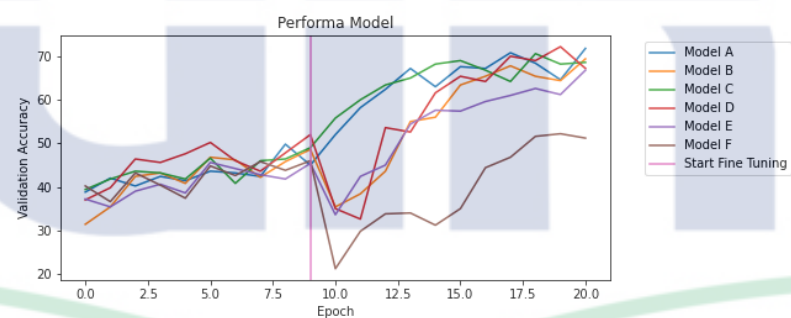
Gambar 5.18 Visualisasi Hasil Model D



Gambar 5.19 Visualisasi Hasil Model E



Gambar 5.20 Visualisasi Hasil Model F



Gambar 5.21 Ringkasan Evaluasi Hasil Uji Coba Parameter

Berdasarkan hasil dari *Tabel 5.2* dan *Gambar 5.15* sampai *Gambar 5.21* dapat disimpulkan hal berikut:

1. Learning rate lebih besar dari  $0.00001$  akan menimbulkan penurunan akurasi saat epoch dimulainya fine-tuning, sedangkan learning rate  $0.00001$  memberikan hasil yang stabil.

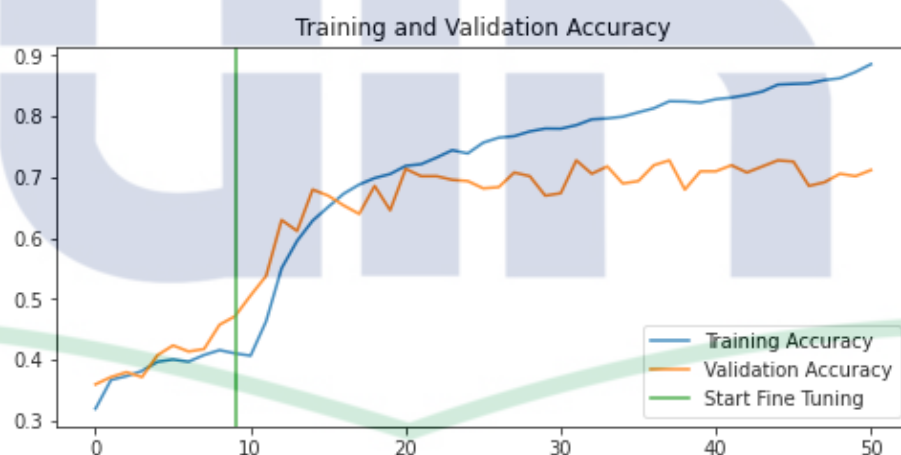


- Dropout 25% cenderung memiliki akurasi validasi yang lebih besar namun hal tersebut hanya berlaku untuk learning rate yang lebih besar dari  $0.00001$ , sedangkan dropout 50% akan menghasilkan akurasi validasi lebih besar hanya pada learning rate  $0.00001$ .

Berdasarkan akurasi validasi yang paling besar dan tidak overfitting, maka model terbaik dari model terbaik adalah **model A**. Model ini adalah model yang sama pada saat pencarian model terbaik yaitu model **VGG16 (DA, TL, FT, DO)** sehingga usaha merubah parameter demi meningkatkan performa tidak membuahkan hasil yang diharapkan.

### 5.3 Mencari Titik Konvergen Model Terbaik

Setelah menentukan model terbaik, selanjutnya adalah melihat potensi dari model apakah model masih dapat berkembang atau sudah mencapai titik konvergen. Maka langkah selanjutnya adalah menambahkan epoch pada model A dari 20 menjadi 50 dengan harapan penulis dapat melihat titik konvergen pada model A.



Gambar 5.22 Evaluasi Hasil Model A dengan 50 Epoch

Pada Gambar 5.21 dapat disimpulkan bahwa model A mulai mencapai titik konvergen pada epoch 20 dikarenakan setelah epoch 20 akurasi validasi tidak mengalami peningkatan yang signifikan. Maka model A dengan 20 epoch adalah model terbaik yang dihasilkan penelitian ini, langkah selanjutnya adalah evaluasi

model A menggunakan *10-fold cross validation* untuk mendapatkan performa model A secara merata.

#### 5.4 Implementasi 10-Fold Cross Validation pada Model Terbaik

Evaluasi final dari penelitian ini akan diperoleh dengan menggunakan 10-fold cross validation pada model terbaik, dimana model terbaik sejauh ini adalah model A. Setelah dilakukan 10-fold cross validation pada model A didapatkan hasil sebagai berikut:

FOLD	VALIDATION ACCURACY
1	71.4%
2	72.0%
3	68.0%
4	72.6%
5	72.8%
6	71.2%
7	71.2%
8	64.0%
9	62.2%
10	74.8%
RATA-RATA	70.0%

Tabel 5.3 Evaluasi 10-Fold Cross Validation

## **BAB VI**

### **KESIMPULAN DAN SARAN**

#### **6.1 Kesimpulan**

Penelitian ini telah mengimplementasikan berbagai model menggunakan deep learning CNN dan VGG16 untuk klasifikasi penyakit pada tanaman singkong. Model CNN tanpa tambahan metode mendapatkan performa terburuk di antara model yang ada dengan akurasi validasi mencapai 16%. Sedangkan performa terbaik diperoleh oleh model VGG16 dengan tambahan 4 metode yaitu transfer learning, data augmentation, fine-tuning dan dropout dengan akurasi validasi yang diperoleh dengan 10-fold cross validation sebesar 70%.

Perbandingan performa antara model CNN dan VGG16 dengan transfer learning, data augmentation, fine-tuning dan dropout adalah 54%. Hal ini menunjukkan peningkatan performa model yang signifikan. Dari kesimpulan ini, dapat dikatakan bahwa penelitian ini sukses menerapkan arsitektur VGG16 dengan transfer learning, fine-tuning, data augmentation dan dropout untuk meningkatkan akurasi CNN pada klasifikasi penyakit tanaman singkong.

#### **6.2 Saran**

Berikut adalah beberapa saran untuk penelitian selanjutnya yaitu:

1. Menggunakan dataset yang lebih baik lagi dikarenakan dataset yang digunakan pada penelitian ini masih kurang baik.
2. Apabila menggunakan dataset yang sama, harap dilakukan proses pemilihan dataset yang bagus dengan tidak menggunakan pemilihan secara acak.
3. Menambahkan dataset testing untuk pengukuran performa model pada dataset baru.
4. Penggunaan arsitektur CNN lainnya seperti GoogleNet, AlexNet, VGG19, ResNet-50, Inception-V3, MobileNet.
5. Implementasi metode *L1* atau *L2 Regularization* pada model.

### DAFTAR PUSTAKA

- Ahmad, N., Ahmad, N., Informasi, P. T., Ar-raniry, U. I. N., Aceh, B., Informatika, P. T., Ghafur, U. J., Pakar, S., Chaining, F., & Informasi, T. (2020). *Metode Forward Chaining untuk Deteksi Penyakit Pada Tanaman Kentang*. 1(2), 7–19.
- Goodfellow, I., Bengio, Y., & Aaron, C. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>
- Hridayami, P., Putra, I. K. G. D., & Wibawa, K. S. (2019). Fish species recognition using VGG16 deep convolutional neural network. *Journal of Computing Science and Engineering*, 13(3), 124–130. <https://doi.org/10.5626/JCSE.2019.13.3.124>
- Minarni., Indra, W., & Wenda, H. . (2017). Case-Based Reasoning (CBR) pada Sistem Pakar Identifikasi Hama dan Penyakit Tanaman Singkong dalam Usaha Meningkatkan Produktivitas Tanaman Pangan. *Jurnal TEKNOIF*, 5(1), 41–47. <https://doi.org/10.21063/JTIF.2017.V5.1.41-47>
- Padol, P. B., & Yadav, A. A. (2016). SVM classifier based grape leaf disease detection. *Conference on Advances in Signal Processing, CASP 2016*, 175–179. <https://doi.org/10.1109/CASP.2016.7746160>
- Qian, L., Hu, L., Zhao, L., Wang, T., & Jiang, R. (2020). Sequence-Dropout Block for Reducing Overfitting Problem in Image Classification. *IEEE Access*, 8, 62830–62840. <https://doi.org/10.1109/ACCESS.2020.2983774>
- Radovanovic, D., & Dukanovic, S. (2020). Image-Based Plant Disease Detection: A Comparison of Deep Learning and Classical Machine Learning Algorithms. *2020 24th International Conference on Information Technology, IT 2020, February*, 1–4. <https://doi.org/10.1109/IT48810.2020.9070664>
- Rismiyati, R., & Luthfiarta, A. (2021). VGG16 Transfer Learning Architecture for Salak Fruit Quality Classification. *Telematika*, 18(1), 37. <https://doi.org/10.31315/telematika.v18i1.4025>
- Roy, S. K., Krishna, G., Dubey, S. R., & Chaudhuri, B. B. (2020). HybridSN: Exploring 3-D-2-D CNN Feature Hierarchy for Hyperspectral Image Classification. *IEEE*

*Geoscience and Remote Sensing Letters*, 17(2), 277–281.  
<https://doi.org/10.1109/LGRS.2019.2918719>

Rozaqi, J., Sunyoto, A., & Arief, R. (2021). *Implementasi Transfer Learning pada Algoritma Convolutional Neural Network Untuk Identifikasi Penyakit Daun Kentang*  
*Implementation of Transfer Learning in the Convolutional Neural Network Algorithm for Identification of Potato Leaf Disease*. 1(1).  
<https://press.umsida.ac.id/index.php/PELS/article/view/820/478>

Statistik, B. P. (2018). *Produksi Ubi Kayu Menurut Provinsi (ton), 1993-2015*.  
[https://www.pertanian.go.id/Data5tahun/TPATAP-2017\(pdf\)/27-ProdUbikayu.pdf](https://www.pertanian.go.id/Data5tahun/TPATAP-2017(pdf)/27-ProdUbikayu.pdf)

Sujatha, R., Chatterjee, J. M., Jhanjhi, N. Z., & Brohi, S. N. (2021). Performance of deep learning vs machine learning in plant leaf disease detection. *Microprocessors and Microsystems*, 80(November 2020), 103615.  
<https://doi.org/10.1016/j.micpro.2020.103615>

Susila, M. F., Irawan, B., & Setianingsih, C. (2020). *DETEKSI PENYAKIT PADA DAUN PAKCOY DENGAN PENGOLAHAN CITRA MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK DISEASES DETECTION OF BOK CHOY LEAF BY IMAGE PROCESSING USING CONVOLUTIONAL NEURAL NETWORK METHOD*. 7(3), 9347–9354.

Taylor, L., & Nitschke, G. (2017). *Improving Deep Learning using Generic Data Augmentation*. <http://arxiv.org/abs/1708.06020>

Too, E. C., Yujian, L., Njuki, S., & Yingchun, L. (2019). A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*, 161(February), 272–279.  
<https://doi.org/10.1016/j.compag.2018.03.032>