PROPOSAL

DETEKSI PENYAKIT TANAMAN DENGAN MENGGUNAKAN METODE DEEP LEARNING PADA PERANGKAT MOBILE



RESEARCH

*Burhanudin Syamsuri*

*2001855113*

Program Pascasarjana Ilmu Komputer

PROGRAM STUDI TEKNIK INFORMATIKA JENJANG S2

UNIVERSITAS BINA NUSANTARA

JAKARTA

2019

PROPOSAL

DETEKSI PENYAKIT TANAMAN DENGAN MENGGUNAKAN METODE DEEP LEARNING PADA PERANGKAT MOBILE



RESEARCH

*Burhanudin Syamsuri*

*2001855113*

Pembimbing:

I Gede Putra Kusuma Negara, B.Eng., PhD.

23– 02 – 2019

DAFTAR ISI

[DAFTAR GAMBAR v](#_Toc7084773)

[DAFTAR TABEL vi](#_Toc7084774)

[BAB 1 PENDAHULUAN 7](#_Toc7084775)

[1.1 Latar Belakang 7](#_Toc7084776)

[1.2 Rumusan Masalah 11](#_Toc7084777)

[1.3 Tujuan Penelitian 11](#_Toc7084778)

[1.4 Manfaat Penelitian 11](#_Toc7084779)

[1.5 Ruang Lingkup 12](#_Toc7084780)

[BAB 2 LANDASAN TEORI 13](#_Toc7084781)

[2.1 Deep Learning 13](#_Toc7084782)

[2.2 *Convolution*al Neural Network (CNN) 14](#_Toc7084783)

[2.3 CNN’s Arsitektur Overview 17](#_Toc7084784)

[2.4 CNN’s Layers 20](#_Toc7084785)

[2.4.1 *Convolution*al Layer 20](#_Toc7084786)

[2.4.2 Active Function (ReLU) 22](#_Toc7084787)

[2.4.3 Pooling Layer 23](#_Toc7084788)

[2.4.4 Fully-Connected Layer 24](#_Toc7084789)

[2.5 Hyperparameters 24](#_Toc7084790)

[2.5.1 Hyperparamater pada Struktur Jaringan. 24](#_Toc7084791)

[2.5.2 Hyperparameter pada training 25](#_Toc7084792)

[2.6 Loss Function 28](#_Toc7084793)

[2.7 CNN’s Arsitektur 28](#_Toc7084794)

[2.7.1 AlexNet 29](#_Toc7084795)

[2.7.2 GoogLeNet / Inception-V1 30](#_Toc7084796)

[2.7.3 VGG 32](#_Toc7084797)

[2.7.4 Inception-V4 dan Inception-ResNet 33](#_Toc7084798)

[2.7.5 MobileNet 35](#_Toc7084799)

[2.7.6 MNasNet 36](#_Toc7084800)

[2.8 Tinjauan Pustaka 38](#_Toc7084801)

[2.8.1 Deteksi dan Klasifikas Penyakit Tanaman dengan CNN 38](#_Toc7084802)

[2.8.2 Penelitian Deteksi dan Klasifikasi Penyakit Tanaman Kopi 43](#_Toc7084803)

[2.8.3 CNN pada Perangkat Mobile dan Embedded Device. 45](#_Toc7084804)

[2.8.4 Rangkuman Tinjauan Pustaka 48](#_Toc7084805)

[BAB 3 METODOLOGI 50](#_Toc7084806)

[3.1 Kerangka Pikir 50](#_Toc7084807)

[3.2 Deteksi dan Klassifikasi Penyakit Tanaman Dengan *Deep Learning* 51](#_Toc7084808)

[3.2.1 Implementasi Sistem 52](#_Toc7084809)

[3.2.2 Deployment 54](#_Toc7084810)

[3.3 Pengumpulan Data 55](#_Toc7084811)

[3.4 Evaluasi state of the art CNN 59](#_Toc7084812)

[3.4.1 Evaluasi Tahap Implementasi 59](#_Toc7084813)

[3.4.2 Evaluasi Tahap Deployment 63](#_Toc7084814)

[DAFTAR PUSTAKA 66](#_Toc7084815)

# DAFTAR GAMBAR

[Gambar 1.1 Tingkat Produktivitas Kopi Terbesar di Dunia 7](#_Toc7285936)

[Gambar 1.2 Tingkat Kerusakan Tanaman oleh Karat Daun 8](#_Toc7285937)

[Gambar 2.1 Arsitektur Convolutional Network 18](#_Toc7285938)

[Gambar 2.2 Visualisasi Input dan Filter pada Convolutional Layer 20](#_Toc7285939)

[Gambar 2.3 Operasi Convolution 21](#_Toc7285940)

[Gambar 2.4 ReLU Function 23](#_Toc7285941)

[Gambar 2.5 Max Pooling sample 23](#_Toc7285942)

[Gambar 2.6 AlexNet Arsitektur 30](#_Toc7285943)

[Gambar 2.7 GoogLeNet Arsitektur Diagram 31](#_Toc7285944)

[Gambar 2.8 VGGNet Konfigurasi Diagram 33](#_Toc7285945)

[Gambar 2.9 Konfigurasi Inception-V4 34](#_Toc7285946)

[Gambar 2.10 Skema Inception-ResNet2 34](#_Toc7285947)

[Gambar 2.11 Depthwise Separable Convolutional 35](#_Toc7285948)

[Gambar 2.12 Mobile Arsitektur Diagram 36](#_Toc7285949)

[Gambar 2.13 MnasNet Arsitektur Digaram 37](#_Toc7285950)

[Gambar 3.1 Kerangka Pikir 50](#_Toc7285951)

[Gambar 3.2 Diagram Alir Implementasi 53](#_Toc7285952)

[Gambar 3.3 Diagram Alir deployment pada Mobile Deployment 55](#_Toc7285953)

[Gambar 3.4 Contoh Kelas Daun Plantvillage 58](#_Toc7285954)

[Gambar 3.5 Dataset Tambahan Daun Kopi 59](#_Toc7285955)

[Gambar 3.6 Putaran Proses Training 61](#_Toc7285956)

[Gambar 3.7 Confusion Matrix Calculator 64](#_Toc7285957)

# DAFTAR TABEL

[Tabel 2.1 Riset Model CNN dengan Dataset PlantVillage 16](#_Toc7252085)

[Tabel 2.2 Perbandingann Riset Mengenai Plant Disease Detection 42](#_Toc7252086)

[Tabel 3.1 Spesifikasi Mesin untuk Proses Training 53](#_Toc7252087)

[Tabel 3.2 Spesifikasi Perangkat Mobile 55](#_Toc7252088)

[Tabel 3.3 Dataset PlantVillage 55](#_Toc7252089)

[Tabel 3.4 Dataset Daun Kopi 58](#_Toc7252090)

[Tabel 3.5 Evaluasi Kebutuhan Waktu Training 61](#_Toc7252091)

[Tabel 3.6 Evalusi Model Re-Training 62](#_Toc7252092)

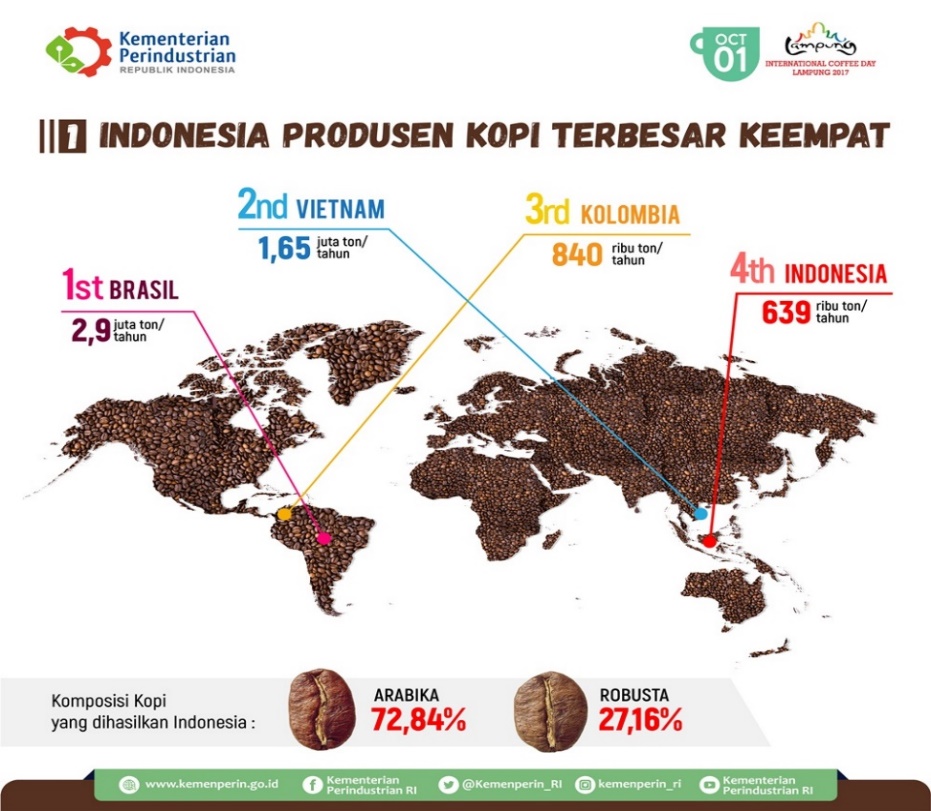
[Tabel 3.7 Evaluasi untuk Perangkat Mobile 64](#_Toc7252093)

[Tabel 3.8 Evaluasi Empiris Kebutuhan Komputasi Perangkat Mobile 64](#_Toc7252094)

# PENDAHULUAN

## Latar Belakang

Berdasarkan data Kementerian Perindustrian Republik Indonesia, Indonesia adalah produsen kopi nomor empat di dunia (KOMINFO, 2017), seperti diilustrasikan Gambar 1.1 yang menunjukkan tingkat produktivitas kopi terbesar di dunia.



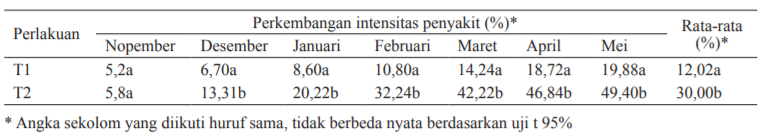
Gambar 1.1 Tingkat Produktivitas Kopi Terbesar di Dunia

Statistik di atas menunjukkan masih rendahnya produktivitas kopi Indonesia, yaitu 539 kilogram biji kering/ha/tahun dibandingkan negara produsen lain seperti Vietnam (1.540 kg/ha/tahun), Kolombia (1.220 kg/ha/tahun), dan Brasil (1.000 kg/ha/tahun) (Mahfud, Siti, Ismiyati, & Ardiansyah, 2010).

Karat daun (*coffee leaf rust*) adalah jenis penyakit pada tanaman kopi yang memberikan kontribusi besar terhadap penurunan produksi dan dilaporkan sebagai penyakit paling merusak pada tanaman kopi (Agrios, 1988). Di perkebunan kopi Indonesia, penyakit karat daun dilaporkan menurunkan produksi 20-70% (Mahfud et al., 2010).

Kesulitan dalam pengendalian gangguan penyakit karat daun ini disebabkan kesalahan dalam penerapan praktik kultur teknis pada perkebunan rakyat yang luasnya mencapai 1.241.500 ha atau 95,5% dari keseluruhan luas tanaman kopi di Indonesia. Hal ini menyebabkan tingginya gangguan penyakit karat daun yang berujung pada penurunan produksi kopi (Sugiarti, 2017).

Penyakit karat daun pada tanaman kopi disebabkan oleh Jamur *H. Vastatrix.* Jamur ini menimbulkan bercak berwarna kuning yang menjadikan daun berwarna coklat. Serangan penyakit karat daun akan menyebabkan daun gugur, pohon menjadi gundul, dan mematikan tanaman kopi jika tidak ditanggulangi (Mahfud et al., 2010). Gambar 1.2 menunjukkan statistik tingkat kerusakan tanaman kopi oleh penyakit karat daun di Indonesia (Mahfud & Cholil, 2012).



Gambar 1.2 Tingkat Kerusakan Tanaman oleh Karat Daun

Sistem pendeteksian dini gejala penyakit karat daun dengan memanfaatkan teknologi terbaru di bidang pertanian telah dikembangkan dalam sejumlah penelitian. Sistem tersebut mengaplikasikan metode otomatisasi untuk menghindari kerusakan tanaman pada saat proses pendeteksian penyakit ini. Penelitian mengenai penyakit karat daun kopi yang dilakukan oleh (Castro, Oblitas, Maicelo, & Avila-George, 2018) menggunakan *Advance Hyperspectral Images System* (HIS’s) mengklasifikasikan penyakit karat daun menjadi lima kelas yaitu *Healthy, Initial, Intermediate, Advanced,* dan *Necrotic*. HIS’s pada dasarnya menggunakan metode khusus *machine learning* seperti *Artificial Neural Networks (ANN’s)*, *Decision Tree (DT)*, *K-means (KM)*, *K-Nearest Neighbour (KNN)*, dan *Support Vector Mesin (SVM)* untuk melakukan data analisis. Teknik HIS’s mencapai akurasi 90,30% dengan metode *Decision Tree*, 93% dengan metode K-NN, dan akurasi tertinggi 94,70% dicapai dengan metode *Support Vector Mesin* *(SVM)*.

Teknik HIS’s memperoleh gambar hiperspektral dari sistem sensor yang terdiri dari beberapa komponen yaitu kamera dan spektograf, penutup, komputer, sumber daya, sumber cahaya yang terdiri dari empat lampu halogen yang diatur pada ketinggian variabel dan dapat disesuaikan, dan platform sampel untuk pengguliran horizontal selama pemindaian yang dikendalikan kecepatan. Komputer yang cepat, detektor yang sensitif, dan kapasitas penyimpanan data yang besar diperlukan untuk menganalisis data hiperspektral. Kapasitas penyimpanan data yang signifikan diperlukan karena kubus hiperspektral adalah kumpulan data multidimensi yang besar, yang berpotensi melebihi ratusan *megabyte*. Komponen-komponen tersebut di atas menyebabkan sistem HIS memiliki biaya dan kompleksitas yang tinggi. Hal ini menjadi kekurangan paling utama dari sistem ini.

Beberapa penelitian berbeda dengan menggunakan metode *deep learning* yang menyederhanakan kompleksitas dan berbiaya lebih rendah dengan model *Convolutional Neural Network* (CNN) arsitektur terbaru seperti AlexNet, GoogLeNet, ResNet, VGG, dan Cifar10 menghasilkan tingkat akurasi yang lebih tinggi. Penelitian ini menggunakan *color image* sebagai input yang langsung diolah tanpa melalui sensor khusus seperti yang dilakukan pada penelitian dengan hiperspektral.

Penelitian dengan membuat perbaikan pada arsitektur CNN GoogLeNet dan Cifar10 untuk mengidentifikasi penyakit pada tanaman tomat, mencapai tingkat akurasi 98,9% (Zhang, Qiao, Meng, Fan, & Zhang, 2018). Dengan beberapa arsitektur CNN yang berbeda (Brahimi et al., 2018) melakukan klasifikasi penyakit tanaman menggunakan dataset publik PlantVillage dan mencapai akurasi 99,76%.

Berdasarkan hasil penelitian sebelumnya, belum ada penelitian yang memberikan fokus pada aplikasi teknik *deep learning* untuk dapat digunakan pada perangkat mobile khususnya untuk penelitian deteksi penyakit pada tanaman kopi di Indonesia. Sehingga penelitian ini nantinya akan memberikan kontribusi dengan menambahkan data set daun kopi Indonesia.

Dengan aplikasi yang dapat berjalan di perangkat mobile, diharapkan sistem dapat dieksekusi secara lokal tanpa memerlukan proses pengiriman gambar ke server untuk proses identifikasi, menghindari adanya keterbatasan pada koneksi internet dan juga menghindari latensi pemrosesan pada server yang berada di *cloud*. Hal ini dapat memberikan kontribusi langsung bagi petani dalam sistem deteksi dini penyakit tanaman khususnya pada tanaman kopi di Indonesia.

Maka dalam penelitian ini kami mengajukan sebuah pendekatan metode CNN pada perangkat mobile untuk mendeteksi penyakit karat daun (*Hemileia vastatrix*) dan bercak daun (*Cercospora coffeicola*) yang dievaluasi menggunakan dataset publik PlantVillage yang diperluas pada dataset daun tanaman kopi. Implementasi pada penelitian ini akan diujikan dengan memanfaatkan model terbaru framework Tensorflow-lite yang optimal digunakan di perangkat mobile dengan arsitektur MobileNet, RasNet Mobile, [PNASNet-5 Mobile](https://arxiv.org/abs/1712.00559), [NASNet-A Mobile](https://arxiv.org/abs/1707.07012), [VGG](http://arxiv.org/abs/1409.1556.pdf) dan Inception.

## Rumusan Masalah

Berdasarkan uraian di latar belakang, maka masalah penelitian yang akan diselesaikan dapat dirumuskan sebagai berikut:

* Bagaimana mengevaluasi model-model CNN yang dirancang khusus untuk dapat dijalankan di perangkat mobile seperti MobileNet, RasNet Mobile, [PNASNet-5 Mobile](https://arxiv.org/abs/1712.00559), [NASNet-A Mobile](https://arxiv.org/abs/1707.07012), [VGG](http://arxiv.org/abs/1409.1556.pdf), dan Inception.
* Bagaimana mengoptimalkan model CNN tersebut sehingga lebih efisien dalam kebutuhan sumber daya komputasi dan energi pada perangkat mobile dengan melakukan penyederhanaan model tersebut.

## Tujuan Penelitian

Berdasarkan perumusan masalah di atas, tujuan penelitian disusun sebagai berikut:

* Mengevaluasi model CNN yang khusus dirancang untuk digunakan pada perangkat mobile.
* Mengoptimalkan model CNN yang lebih efisien dalam kebutuhan sumber daya komputasi (CPU, Memori, dan kapasitas penyimpanan) dan mengurangi kebutuhan energi pada perangkat mobile.

## Manfaat Penelitian

Dari penelitian yang dilakukan diharapkan memberikan kontribusi manfaat sebagai berikut:

* Mendapatkan kesimpulan model yang terbaik dan optimal digunakan pada perangkat mobile.
* Memiliki sistem deteksi dini dari kinerja metode *image detection* dengan *deep learning* terhadap penyakit pada tanaman dengan menggunakan perangkat mobile.
* Mengurangi probabilitas kegagalan panen dengan tindakan pencegahan.

## Ruang Lingkup

Ruang lingkup penelitian ini adalah sebagai berikut:

* Data yang digunakan menggunakan PlantVillage data set, data daun tanaman akan diambil langsung dari perkebunan bekerjasama dengan pusat penelitian kopi dan kakao di Indonesia, juga gambar-gambar dari internet.
* Model yang akan digunakan adalah model dan arsitektur terbaru MobileNet (Howard et al., 2017), ResNet (He, Zhang, Ren, & Sun, 2016), [PNASNet-5 Mobile](https://arxiv.org/abs/1712.00559) (Liu et al., 2018), [NASNet-A Mobile](https://arxiv.org/abs/1707.07012) (Zoph, Vasudevan, Shlens, & Le, 2017), [VGG](http://arxiv.org/abs/1409.1556.pdf) (Simonyan & Zisserman, 2015), dan Inception V4 (Szegedy, Ioffe, Vanchouke, & Alemi, 2016) yang telah disederhanakan yang disediakan oleh Tensorflow-Lite.
* Menggunakan media perangkat mobile dengan *operating system android* dan dibandingkan dengan PC untuk melakukan pengujian dan hasil.

# LANDASAN TEORI

## Deep Learning

Landasan teori dalam penelitian ini adalah teori *deep learning* dengan model *Convolutional Neural Network* (CNN) yang banyak digunakan dalam melakukan deteksi, klasifikasi, dan prediksi pada gambar. Teori dalam penelitian ini antara lain diambil dari buku yang berjudul “Deep Learning” oleh (Goodfellow, Bengio, & Courville, 2013), *course* di Standford University yaitu “CS231n: Convolutional Neural Networks for Visual Recognition” (Karpathy, 2018), dan berbagai jurnal penelitian dengan model CNN.

(Lecun, Bengio, & Hinton, 2015) dalam penelitian berupa *review* yang berjudul *Deep Learning* mendefinisikan *deep learning* sebagai aplikasi dari *machine learning* yang digunakan untuk mengidentifikasi objek dalam gambar, menyalin pidato menjadi teks, mencocokan item berita, dan menganalisa sentimen pengguna dari postingan di media sosial, yang memanfaatkan kelas teknis.

Pada awalnya teknik *machine learning* konvensional terbatas pada kemampuan untuk memproses data alami dalam bentuk data mentah. Dalam beberapa dekade *machine learning* membutuhkan rekayasa yang cermat dan dengan mempertimbangkan bidang keahlian yang cukup untuk merancang fitur ekstraktor yang mengubah data mentah (seperti nilai piksel gambar) menjadi data yang matang (seperti dalam bentuk data yang representatif secara internal atau fitur vektor yang berasal dari *learning subsistem*, dimana pengklasifikasi dapat mendeteksi atau mengklasifikasi pola di dalam input. *Representation-learning* adalah kumpulan metode yang memungkinkan mesin untuk diinput dengan data mentah dan secara otomatis menemukan representasi yang diperlukan untuk deteksi atau klasifikasi.

Metode *deep learning* adalah metode *representation-learning* dengan berbagai tingkat representasi, diperoleh dengan menyusun modul sederhana namun non-linear yang masing-masing mengubah representasi pada satu tingkat tertentu (dimulai dengan input mentah) menjadi representasi pada tingkat yang lebih tinggi, ke tingkat yang abstrak. Dengan komposisi transformasi yang cukup, fungsi yang sangat kompleks dapat dipelajari.

Sehingga dapat disimpulkan aspek kunci dari metode *deep learning* ini adalah bahwa layer-layer fitur tidak dirancang oleh seorang *engineer* atau orang dengan keahlian tertentu, tetapi layer fitur ini dipelajari dari data menggunakan prosedur pembelajaran.

## Convolutional Neural Network (CNN)

Definisi mengenai *convolutional network* yang dikenal sebagai *Convolutional Neural Network* adalah jenis jaringan saraf tiruan khusus untuk memproses data yang dikenal memiliki *grid-like* topologi (Goodfellow et al., 2013). Contoh penggunaannya adalah *data time series*, yang dapat dianggap sebagai grid Dimensi-1 yang mengambil sampel pada interval waktu regular. Kemudian pada data gambar, yang dapat dianggap sebagai grid Dimensi-2 yang berbentuk piksel. Dalam CNN digunakan operasi matematika yang disebut *convolution*. *Convolution* adalah jenis operasi linier khusus. Jaringan *convolutional* adalah jaringan saraf yang secara sederhana menggunakan *convolution* sebagai pengganti perkalian matriks umum dalam setidaknya satu layernya (Goodfellow et al., 2013).

Di dalam *deep learning*, jaringan saraf *convolutional* merupakan kelas pada *deep neural network* yang paling umum diterapkan untuk menganalisa citra visual, dan digunakan untuk melakukan pengenalan gambar, klasifikasi gambar, deteksi objek, dan pengenalan wajah.

CNN terdiri dari neuron yang memiliki bobot dan bias yang dapat dipelajari. Setiap neuron dapat menerima beberapa input, dengan menghasilkan titik sebagai produk dan secara opsional mengikutinya dengan non-linearitas. Seluruh jaringan masih mengekspresikan fungsi skor tunggal yang dapat dibedakan dari piksel gambar mentah di satu titik ke skor kelas di ujung lainnya.

Secara algoritma, CNN merupakan *deep learning*, dimana dapat mengambil gambar sebagai input, dengan menetapkan nilai bobot dan bias yang dapat dipelajari dari berbagai aspek/objek dalam gambar dan dapat membedakan satu dari yang lain. Pra-pemrosesan yang diperlukan dalam CNN jauh lebih ekonomis dibandingkan dengan algoritma klasifikasi lainnya. Sementara dalam metode yang terdahulu filter direkayasa secara manual, sedangkan CNN dengan pelatihan yang cukup, mampu untuk mempelajari filter/karakteristik dari sebuah objek.

CNN mempunyai keunggulan dalan efisiensi secara komputasi dimana menggunakan operasi *convolution*, penyatuan khusus dan menggunakan *parameter sharing*. Sehingga memungkinkan model CNN berjalan di perangkat apa pun termasuk perangkat mobile. Hal ini menjadikan CNN menarik secara universal.

 Dengan karakteristik dan keunggulan yang disebutkan di atas membuat CNN model menarik untuk digunakan dalam penelitian untuk mengidentifikasi gambar, termasuk pendeteksian penyakit pada tanaman. Terinspirasi dari sebuah kompetisi Plant Village yang diadakan oleh organisasi crowdAI, menggunakan CNN dengan arsitektur yang telah dikembangkan, dan menggunakan dataset yang disediakan oleh (Hughes & Salathe, 2015). Beberapa penelitian bertopik deteksi penyakit tanaman metode *deep learning* model CNN dengan dataset PlantVillage, dirangkum dalam Table 2.1.

Tabel 2.1 Riset Model CNN dengan Dataset PlantVillage

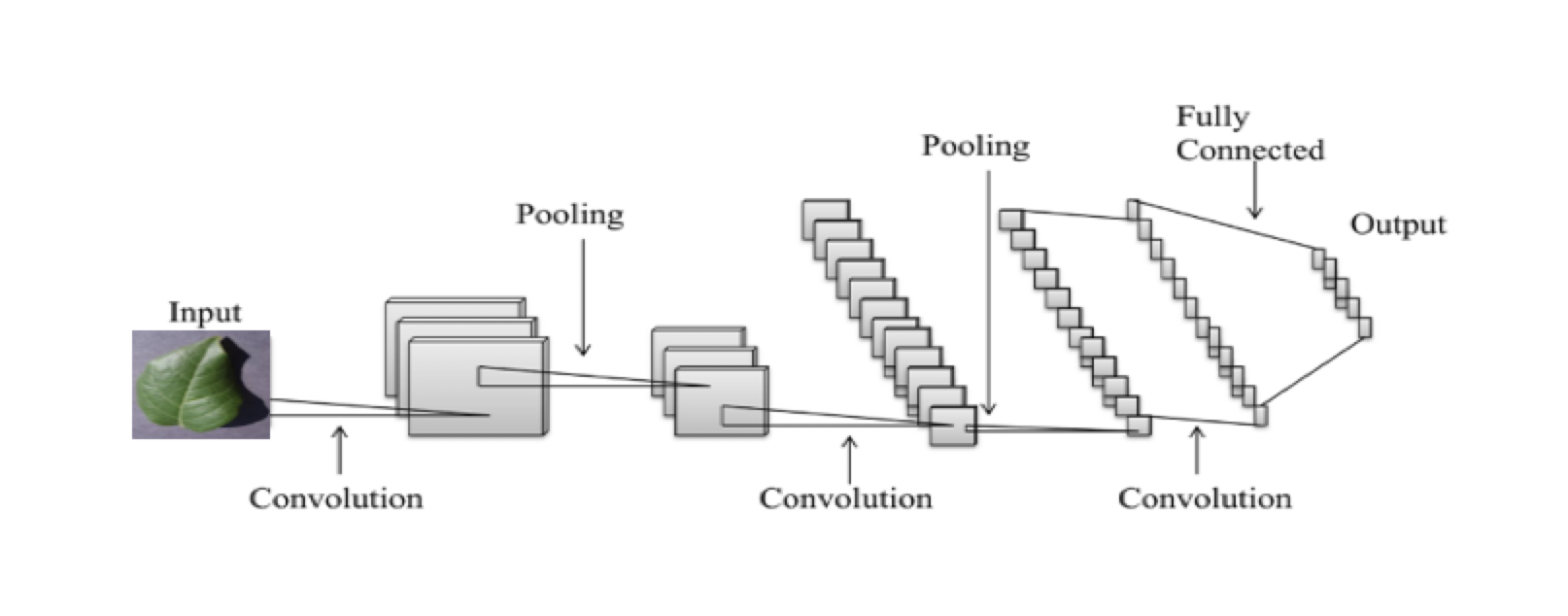
|  |  |  |  |
| --- | --- | --- | --- |
| No | Ref | Topik Penelitian | CNN Arsitektur |
| 1 | (Zhang et al., 2018) | Identifikasi penyakit daun jagung | - GoogleNet.  - Cifar10. |
| 2 | (Brahimi et al., 2018) | Deteksi dan *sailency visualization* penyakit tanaman | * AlexNet. * DenseNet. * Inception-v3. * ResNet-34. * SqueezeNet-1.1. * VGG13 |
| 3 | (Sladojevic, Arsenovic, Anderla, Culibrk, & Stefanovic, 2016) | Deteksi penyakit tanaman dengan klasifikasi daun. | -CaffeNet |
| 4 | (Mohanty, Hughes, & Salathé, 2016) | Penggunaan *deep learning* untuk mendeteksi penyakit tanaman | -AlexNet  -GoogleNet |
| 5 | (Brahimi, Boukhalfa, & Moussaoui, 2017) | Penggunaan *deep learning* untuk penyakit tanaman tomat | -AlexNet  -GoogleNet |
| 6 | (Garcia & Barbedo, 2018) | Dampak jumlah data | -GoogleNet |

## 

## CNN’s Arsitektur Overview

Arsitektur CNN dianalogikan sebagai pola konektivitas Neuron di otak manusia yang diilhami oleh cara kerja dari *Visual Cortex*. Setiap neuron merespon rangsangan pada batas bidang visual yang dikenali sebagai bidang reseptif. Kumpulan bidang tersebut tumpang tindih untuk menutupi seluruh area visual.

Pada dasarnya semua model CNN mengikuti arsitektur yang sama, seperti pada Gambar 2.1, diilustrasikan menggunakan gambar sebagai input kemudian dilakukan operasi *convolutional*, operasi *pooling*, diikuti oleh sejumlah layer yang terhubung sepenuhnya (*fully connected*).



Gambar 2.1 Arsitektur Convolutional Network

Dengan memanfaatkan gambar berwarna sebagai input, CNN membatasi arsitektur dengan cara yang lebih sederhana untuk diterapkan. Secara khusus, layer-layer dari CNN memiliki neuron yang tersusun dalam tiga dimensi: **Lebar (W), Tinggi (H), Kedalaman (D)**. Kedalaman di sini merujuk pada tiga dimensi volume aktivasi, bukan kedalaman dari jaringan neural penuh, yang dapat merujuk pada jumlah total layer dalam jaringan.

Pada setiap layer CNN mengubah isi input 3D menjadi isi output 3D dari aktivasi neuron. Dalam contoh di atas, layer input gambar daun hijau, lebar, dan tingginya akan menjadi dimensi gambar, dan kedalamannya adalah tiga yaitu Merah, Hijau dan Biru sebagai *channel* warna gambar.

Arsitekstur CNN menyerupai urutan yang terdiri dari beberapa layer, dimana setiap layer CNN mengubah sebuah isi dengan mengaktivasi sebuah fungsi yang merubah menjadi bentuk yang lain sehingga dapat dibedakan. Kemudian layer-layer tersebut akan bertumpuk membentuk arsitektur CNN secara lengkap.

Dalam pengklasifikasian *convolutional network* yang paling sederhana, dapat memiliki arsitektur dengan urutan sebagai berikut [*INPUT - CONV - RELU - POOL - FC*]. Urutan detailnya adalah seperti dibawah ini: (Karpathy, 2018).

**1.** **INPUT**Misalkan input yang digunakan ***[32x32x3]*** yang menampung nilai piksel mentah dari gambar, dalam hal ini gambar dengan lebar 32, tinggi 32, dan dengan kedalaman tiga saluran warna Merah, Hijau, dan Biru.

**2. CONV Layer**, akan menghitung output neuron yang terhubung ke wilayah lokal dalam input, masing-masing menghitung produk titik antara bobotnya dan wilayah kecil yang terhubung ke dalam volume input. Sehingga menghasilkan isi seperti [32x32x12], dengan menggunakan 12 filter.

**3. RELU Layer**, akan menerapkan fungsi aktivasi *element-wise*, seperti ambang maksimum (0, x) = maks (0, x) nol. Fungsi ini tidak merubah ukuran isi yaitu tetap ([32x32x12]).

**4. POOL Layer*,*** akan melakukan operasi *downsampling* di sepanjang dimensi spasial (lebar, tinggi), menghasilkan isi seperti [16x16x12].

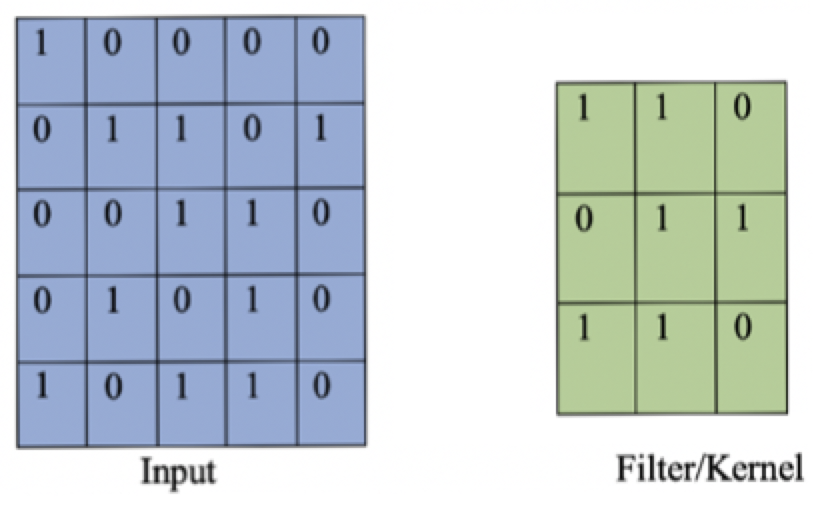
**5. FC Layer**, akan menghitung skor kelas, menghasilkan isi dengan ukuran [1x1x10], di mana masing-masing dari 10 angka sesuai dengan skor kelas, seperti di antara 10 kategori. Sama halnya dengan *Neural Networks* biasa, setiap neuron pada layer ini akan terhubung kesemua angka di dalam nilai isi sebelumnya.

## CNN’s Layers

Layer-layer utama untuk membangun arsitektur CNN adalah *Convolutional Layer (CONV), ReLU, Pooling Layer (POOL), dan Fully-Connected Layer (FC).*

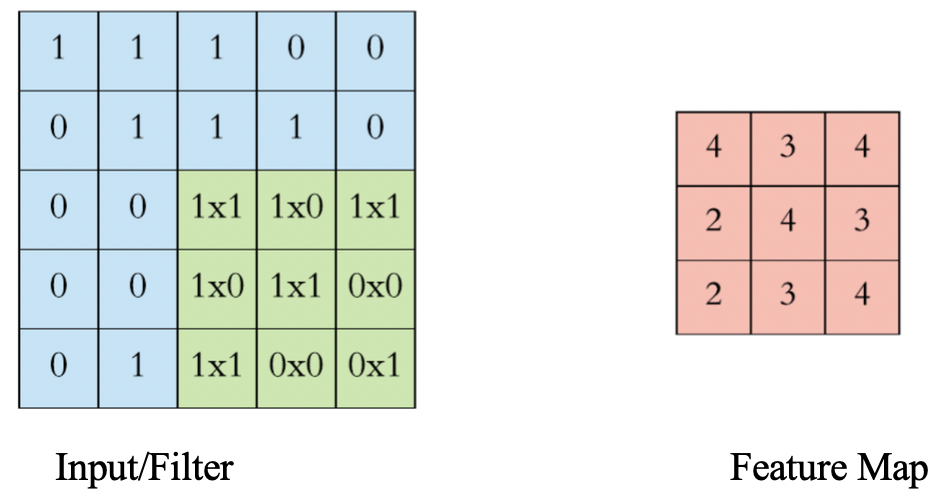
### Convolutional Layer

Adalah bagian paling utama dari *convolutional network* di mana pada layer ini sebagian besar proses komputasi yang berat dilakukan yaitu operasi *convolution*. Dengan tujuan utama untuk melakukan ekstraksi fitur gambar, *convolution* melakukan konservasi spasial antara piksel dengan mempelajari fitur gambar menggunakan kotak kecil pada data input. Dalam setiap layer *convolution* menggunakan berbagai filter untuk mendeteksi dan mengekstraksi fitur seperti mendeteksi tepian gambar, mempertajam, mengaburkan, dan lain lain (Loussaief & Abdelkrim, 2018). Setelah menggeser filter di atas gambar, kita mendapatkan matriks yang dikenal sebagai peta fitur (*feature map*). Untuk lebih memudahkan visualisasinya adalah sebagai berikut:



Gambar 2.2 Visualisasi Input dan Filter pada Convolutional Layer

Dari visual di atas digambarkan sebelah kiri adalah input dari *convolutional* *layer*. Sedangkan di sebelah kanan merupakan *convolutional filter* yang juga disebut sebagai kernel. Ini merupakan *convolutional* 3x3 berdasarkan dari filternya.



Gambar 2.3 Operasi Convolution

Operasi *convolution* dilakukan dengan menggeser filter ini dalam input. Di setiap lokasi, dilakukan perkalian matriks berdasarkan elemen dan menjumlahkan hasilnya. Hasil penjumlahan ini dimasukan kedalam *feature map*. Area hijau tempat operasi *convolution* disebut bidang reseptif. Berdasarkan ukuran filter, maka bidang reseptif di sini adalah 3x3.

Perhitungan *convolution* dari gambar dua dimensi dapat dipetakan ke jendela *convolution* yang digeser secara berkelanjutan untuk memperoleh nilai *convolution* yang sesuai.

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

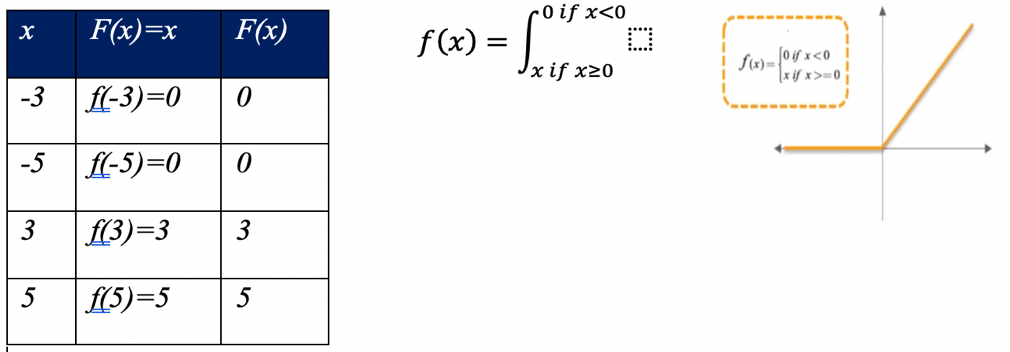
Di mana \* mewakili operasi *convolution*, Wi mewakili filter *convolution* pada layer, dan f mewakili fungsi aktivasi. Wi D [Wi1; Wi2; WiK], K adalah nomornya kernel *convolution* dari layer. Setiap kernel WK i adalah M x M x N matriks berat dengan M menjadi ukuran jendela dan N menjadi jumlah saluran input

### Active Function (ReLU)

Layer berikutnya setelah *convolutional layer* adalah *ReLU layer* yang merupakan sebuah konvensi untuk menerapkan layer non-linier atau fungsi aktivasi setelah *layer convolutional*. Tujuan layer ini adalah untuk memperkenalkan non-linier ke sistem yang menghitung operasi linear pada *layer convolutional*. Hasil *convolution* dan operasi *ReLU* disebut *rectified feature map*.

Pada awalnya banyak digunakan fungsi non-linier seperti tanh dan sigmoid, tetapi peneliti menemukan bahwa layer *ReLU* bekerja jauh lebih baik karena jaringan mampu melatih lebih cepat dan melakukan komputasi yang lebih efisien tanpa membuat perbedaan yang signifikan dalam hasil akurasi. *ReLU* juga meringankan masalah gradien, yang merupakan masalah di mana layer bawah dari jaringan melatih sangat lambat karena gradien berkurang secara eksponensial melalui layer.

*ReLU layer* menerapkan fungsi f (x) = maks (0, x) untuk semua nilai dalam volume input. Secara dasar, layer ini hanya mengubah semua aktivasi negatif menjadi 0. Layer ini meningkatkan sifat nonlinear dari model dan jaringan keseluruhan tanpa mempengaruhi bidang reseptif dari *layer convolutional*. Hasil operasi *convolution* dimasukkan ke dalam fungsi aktivasi *ReLU*. Jadi nilai-nilai di *feature map* akhir sebenarnya bukan jumlah, tetapi merupakan penerapan fungsi *ReLU*.

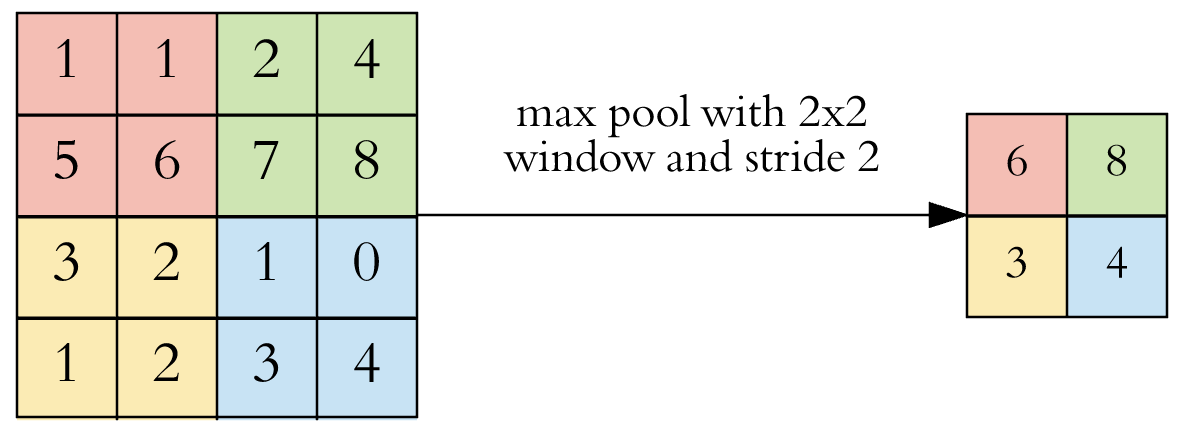


Gambar 2.4 ReLU Function

### Pooling Layer

Disebut juga sebagai *downsampling,* dengan *max pooling* menjadi yang paling popular walau pun ada beberapa opsi yang lain. Layer ini pada dasarnya menggunakan filter (biasanya berukuran 2x2) dan langkah (*stride*) dengan panjang yang sama. Kemudian menerapkannya pada volume input dan menghasilkan jumlah maksimum di setiap subregion dimana filter *convolution* bergeser.

Operasi *pooling* secara efektif dapat mengurangi jumlah jaringan parameter. Untuk mengurangi parameter di semua posisi, operasi pengumpulan dilakukan dengan menghitung statistik karakteristik suatu posisi untuk mewakili keseluruhan karakteristik posisi. Untuk lebih mudah memahaminya diilustrasikan pada Gambar 2.5.



Gambar 2.5 Max Pooling sample

### Fully-Connected Layer

Setelah layer *convolution*al dan *pooling layer*, layer berikutnya adalah *fully-connected layer* sepenuhnya untuk membungkus arsitektur CNN. *Fully-Connected layer* dengan menggunakan vektor fitur dapat mengklasifikasikan gambar input ke dalam beberapa kelas berdasarkan pada dataset pelatihan yang berlabel, *fully-connected layer* berupa vektor angka 1D. Dengan meratakan output dari layer kumpulan terakhir ke vektor dan itu menjadi input ke *fully-connected layer*. Sehingga pada saat meratakan merubah isi angka 3D menjadi sebuah vektor 1D.

## Hyperparameters

*Hyperparameters* adalah variabel yang ditentukan sebelum proses pelatihan. Berdasarkan keterhubungannya, *Hyperparameter* dibedakan menjadi *hyperparameter* yang menentukan struktur jaringan dan *hyperparameter* yang mempengaruhi jaringan pelatihan.

### Hyperparamater pada Struktur Jaringan.

Berikut adalah *hyperparameter* yang akan mempengaruhi struktur jaringan:

1. Jumlah layer tersembunyi (*Hidden Layer*) dan jumlah unit gambar.

*Hidden layer* adalah layer diantara input dan output. Dengan analogi terus menambahkan *hidden layer* sehingga mencapai nilai kesalahan yang tidak meningkat saat fase testing. Menambah banyak *hidden layer* dan jumlah unit gambar dengan teknik regularisasi akan meningkatkan akurasi. Jumlah unit yang lebih sedikit dapat menyebabkan *under-fitting*, yaitu dimana model tidak bisa memodelkan data pelatihan atau mengeneralisasi ke data baru.

1. Dropout

*Dropout* adalah teknik regularisasi untuk meningkatkan akurasi validasi dengan menghindari *overfitting*, dimana sebuah model memiliki *Loss* yang rendah selama pelatihan tetapi berfungsi dengan buruk saat memprediksi data baru. Sehingga dapat meningkatkan kekuatan generalisasi.

- Biasanya nilai *dropout* lebih kecil 20%-50% dari neuron, dengan menetapkan awal *dropout* 20% akan menjadi titik awal yang baik. Dengan probabilitas jika *dropout* terlalu rendah memiliki efek minimal dan memberikan nilai yang terlalu tinggi menyebabkan *under-learning* oleh jaringan.

- Memperbesar Jaringan. Kinerja yang lebih baik didapat ketika *dropout* digunakan pada jaringan yang lebih besar, memberikan model lebih banyak kesempatan untuk mempelajari representasi independen.

1. Network Weight Initialization

Inisilisasi bobot jaringan idealnya menggunakan berat yang berbeda sesuai dengan fungsi aktivasi yang digunakan pada setiap layer.

### Hyperparameter pada training

Berikut adalah *hyperparameter* yang berhubungan dengan proses *training.*

1. Learning Rate

Akan menentukan seberapa cepat jaringan memperbarui parameternya. *Learning rate* yang rendah memperlambat proses pembelajaran tetapi memperbarui dengan lancar. Tingkat pembelajaran yang lebih besar mempercepat pembelajaran tetapi mungkin sulit untuk memperbarui. Biasanya tingkat pembelajaran yang berubah secara perlahan (*decaying learning rate*) lebih banyak digunakan.

1. Momentum

Momentum membantu untuk mengetahui arah langkah selanjutnya dengan mendapatkan pengetahuan dari langkah-langkah sebelumnya. Ini membantu untuk mencegah osilasi. Pilihan momentum yang biasanya digunakan adalah antara 0,5 hingga 0,9.

1. Epoch

*Epoch* adalah berapa kali seluruh data pelatihan ditampilkan ke jaringan saat pelatihan. Tingkatkan epoch sampai akurasi validasi mulai berkurang bahkan ketika akurasi pelatihan meningkat (*overfitting*).

1. Batch Size

Ukuran *batch* adalah jumlah sub sampel yang diberikan ke jaringan setelah pembaruan parameter terjadi. *Default* yang baik untuk ukuran kumpulan mungkin 32. Ukuran *batch* yang dapat digunakan 64, 128, 256, dan sebagainya.

Secara umum ada empat *hyperparameter* yang penting ditentukan saat melakukan *training* yaitu sebagai berikut:

1. Ukuran filter: Dalam banyak contoh banyak digunakan filter 3x3, tetapi 5x5 atau 7x7 juga bisa digunakan tergantung pada aplikasi. Filter yang dimaksud disini adalah filter pada 3D dan yang memiliki dimensi kedalaman (*depth*) juga, tetapi karena kedalaman filter pada layer tertentu sama dengan kedalaman inputnya, sehingga kedalaman terkadang diabaikan.
2. Jumlah filter: ini adalah parameter yang paling bervariabel, yang paling sering digunakan antara 32 dan 1024. Menggunakan lebih banyak filter menghasilkan model yang lebih kuat, tetapi berisiko *overfitting* karena peningkatan jumlah parameter. Biasanya penentuan filter dimulai dengan filter kecil pada layer awal, dan semakin bertambah jumlahnya ketika masuk lebih dalam ke jaringan.
3. *Stride*: menentukan seberapa banyak kita memindahkan filter *convolution* pada setiap langkah. Secara *default* nilainya adalah 1. *Stride* yang lebih besar dapat digunakan untuk mengurangi *overlap* di antara bidang reseptif. Ini juga membuat *feature map* yang dihasilkan lebih kecil karena dapat melewatkan lokasi yang potensial.
4. *Padding*: *Padding* biasanya digunakan dalam CNN untuk mempertahankan ukuran peta fitur, jika tidak mereka akan menyusut pada setiap layer, yang tidak diinginkan. Angka *convolution* 3D yang kami lihat di atas menggunakan pelapis, itulah sebabnya tinggi dan lebar peta fitur sama dengan input (keduanya 32x32), dan hanya kedalaman yang berubah.

## Loss Function

Digunakan untuk mengukur perbedaan antara hasil yang diprediksi dan label input. Seperti dijelaskan di bawah ini:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

di mana W menunjukkan bobot matriks *convolution*al dan layer yang terhubung penuh, n menunjukkan jumlah sampel *training*, i adalah indeks sampel *training*, dan k adalah indeks kelas. Jika sampel dengan milik kelas k, yik D 1; selain itu yik D 0. P (xi D k) adalah probabilitas input xi milik kelas k yang diprediksi oleh model, yaitu fungsi dari parameter W.

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

di mana α adalah tingkat pembelajaran, yang merupakan parameter yang sangat penting yang menentukan ukuran langkah pembelajaran. K adalah indeks kelas, artinya sama dengan persamaan (2).

## CNN’s Arsitektur

Dalam perkembangan sejarah pengembangan CNN yang paling pertama adalah LeNet pada tahun 1990, dalam karya inovatif oleh Yann LeCun(LeCun & Bengio, 1995) yang merupakan hasil dari banyak pengulangan yang sukses sejak tahun 1988. LeNet mendorong berkembangnya *deep learning*, dimana saat itu arsitektur LeNet digunakan untuk aplikasi pengenalan karakter.

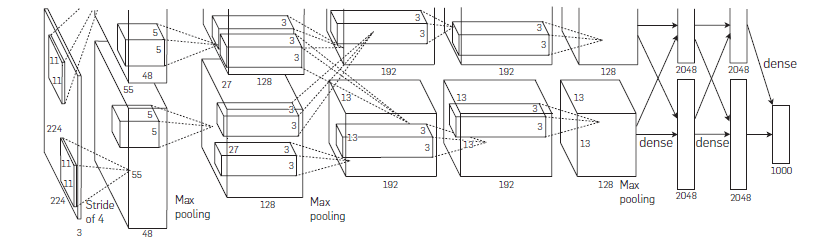
Ada banyak arsitektur CNN baru yang dikembangkan dalam beberapa tahun terakhir. Puncak pengembangan CNN’s arsitektur dimulai dengan AlexNet pada 2012 dan telah tumbuh secara eksponensial sejak itu. Penelitian terhadap arsitektur jaringan *convolutional* berlangsung begitu cepat, kompetisi *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* telah menghasilkan arsitektur baru terbaik dengan menggunakan tolak ukur penilaian yang telah ditentukan.

Selain model-model yang umum digunakan juga telah dikembangkan model yang dioptimalkan untuk dapat digunakan pada perangkat mobile dan *embedeed-device*, yang memungkinkan inferensi pembelajaran mesin pada perangkat dengan latensi rendah dan ukuran biner kecil.

### AlexNet

Merupakan karya pertama yang mempopularkan *convolutional networks* dalam Computer Vision, AlexNet(Krizhevsky, Sutskever, & Hinton, 2012) merupakan pemenang ImageNet ILSVRC pada 2012 dengan keunggulan yang signifikan dengan Top 5 *error* besar 16% dibandingkan dengan pemenang kedua dengan Top-5 *error* 26% kesalahan. Arsitektur jaringan AlexNet berisi delapan *learned layer*, lima *convolutional layer* dan tiga *fully connected layer.* Arsitektur AlexNet mengikuti pola desain yang sama dengan arsitektur LeNet-5(LeCun, 1989), menggunkan satu set layer *convolution* bertumpuk diikuti oleh satu atau lebih *fully connected layer.*

Gambar 2.6 yang merupakan gambar asli yang mengilustrasikan model AlexNet.



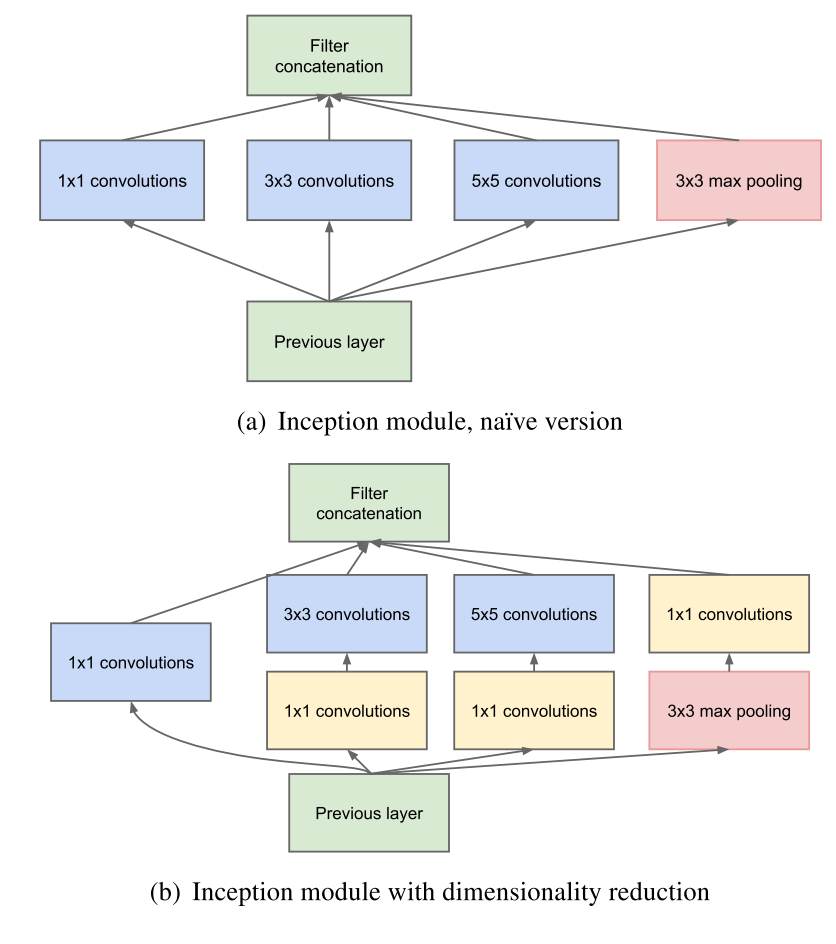
Gambar 2.6 AlexNet Arsitektur

Jaring ini menggunakan delapan layer dengan bobot, dengan lima yang pertama bersifat *convolutional* dan tiga sisanya adalah *FC-Layer*, dua *convolution* pertama (konv {1,2}) masing-masing diikuti oleh normalisasi dan layer penyatuan, dan layer *convolution* terakhir (conv5) diikuti oleh satu layer *pooling.* Output dari *FC-Layer* terakhir dilanjutkan ke softmax yang menghasilkan distribusi lebih dari 1000 label kelas. Jaringan AlexNet memaksimalkan tujuan regresi logistik multinomial, yang setara dengan memaksimalkan rata-rata di seluruh kasus pelatihan dari log-probabilitas label yang benar di bawah distribusi prediksi.

### GoogLeNet / Inception-V1

Nama "GoogLeNet" (Szegedy et al., 2015) merujuk pada arsitektur *inception* dengan inkarnasi tertentu yang digunakan dalam pengajuan untuk kompetisi LSVRC 2014. GooLeNet juga menggunakan satu jaringan *inception* yang lebih dalam dan lebih luas dengan kualitas yang sedikit lebih unggul.

Struktur GoogLeNet memiliki 22 layer, ditandai dengan layer yang lebih dalam. Model GoogLeNet memiliki lebih banyak fitur daripada struktur *deep learning* sebelumnya, karena peningkatan kedalaman, lebar, dan data training. Namun demikian, GoogLeNet memiliki lebih sedikit parameter daripada model VGG dan AlexNet, yang mana VGG dan AlexnNet adalah aplikasi yang mudah dipahami dari konsep *deep learning*. GoogLeNet menggunakan struktur *sparse network* untuk menangani kekurangan dari sumber daya komputasi yang terlalu kecil dan juga terlalu sibuk. Sehingga model piramida digunakan untuk menambah lebar dan mengedepankan konsep "*Inception Module*", seperti terlihat pada Gambar 2.7 yang mengilustrasikan model ini.



Gambar 2.7 GoogLeNet Arsitektur Diagram

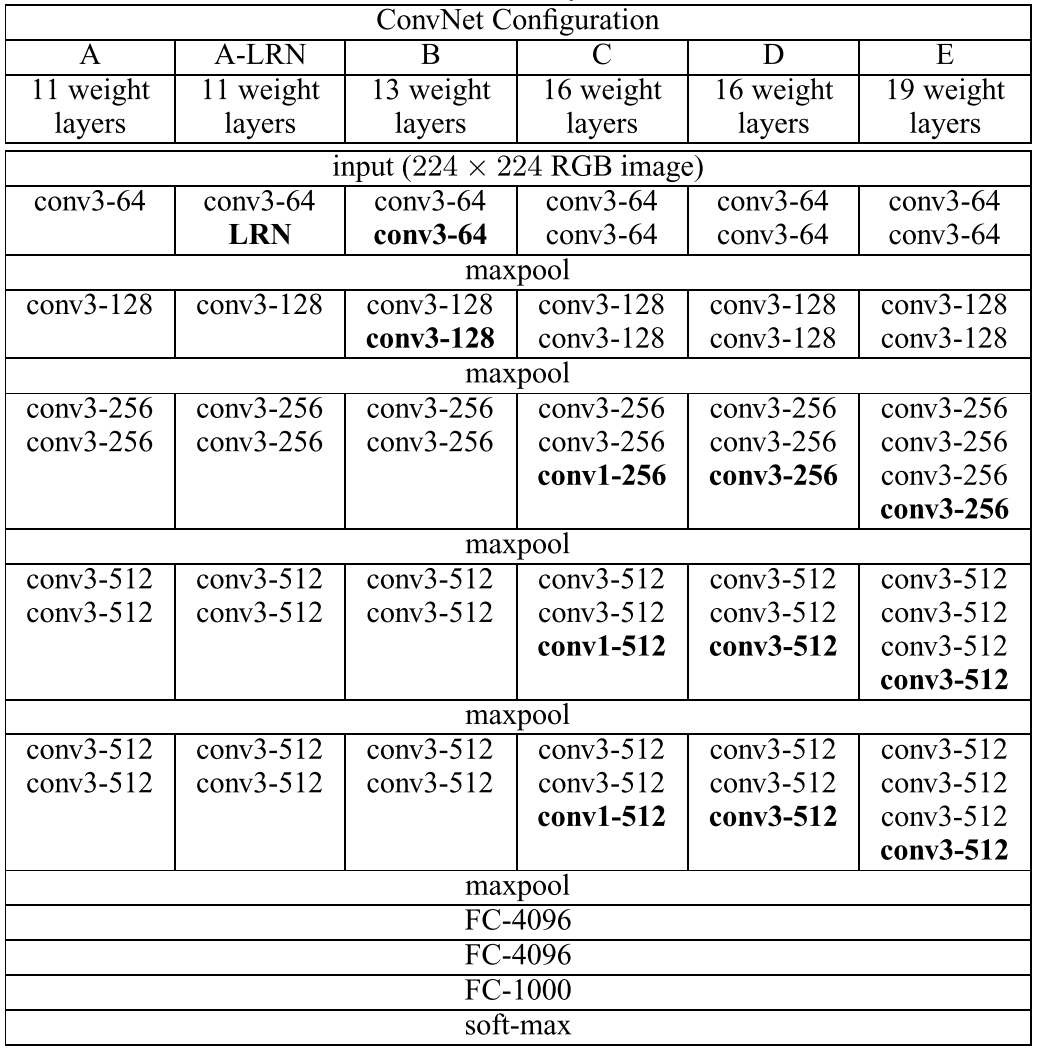
Gagasan utama dari "*Inception Module*" adalah menggunakan komponen yang padat untuk mendekati struktur sparse lokal yang optimal. Sebanyak sembilan modul awal digunakan dalam struktur GoogLeNet. Setiap modul menyertakan beberapa layer *convolutional* paralel dengan ukuran 1 x 1, 3 x 3, 5 x 5 dan layer pengumpulan maksimal untuk menangkap berbagai fitur secara bersamaan.

### VGG

VGGNet(Simonyan & Zisserman, 2015) merupakan Runner-up di ILSVRC 2014 karya dari Karen Simonyan dan Andrew Zisserman. Dengan kontribusi utamanya yaitu memberikan kesimpulan bahwa kedalaman jaringan adalah komponen penting untuk sebuah kinerja arsitektur yang baik. VGGNet berisi 16 layer CONV/FC dan, secara menarik, menampilkan arsitektur yang sangat homogen yang hanya melakukan *convolution* 3x3 dan pooling 2x2 dari awal hingga akhir.

Kelemahan dari VGGNet adalah membutuhkan *resource* yang lebih banyak dalam proses evaluasi, dimana menggunakan lebih banyak memori dan banyak parameter sebesar 140M. Sebagian besar parameter ini digunakan pada FC-Layer pertama, selanjutnya ditemukan bahwa FC-Layer ini dapat dihapus tanpa mengakibatkan penurunan kinerja, sehingga secara signifikan mengurangi jumlah parameter yang diperlukan. VGGNet sangat berbeda dari yang digunakan AlexNet. Menggunakan bidang reseptif yang lebih kecil di *convolution* pertama, dimana VGGNet menggunakan bidang reseptif yang relatif kecil 3 × 3, dengan operasi *convolution* pada input di setiap piksel dengn stradding 1.

Konfigurasi VGGNet diilustrasikan pada Gambar 2.8 dengan peningkatan kedalaman dari kiri (A) ke kanan (E), dengan menambahkan banyak layer ditambahkan (layer yang ditambahkan ditampilkan dalam huruf tebal).



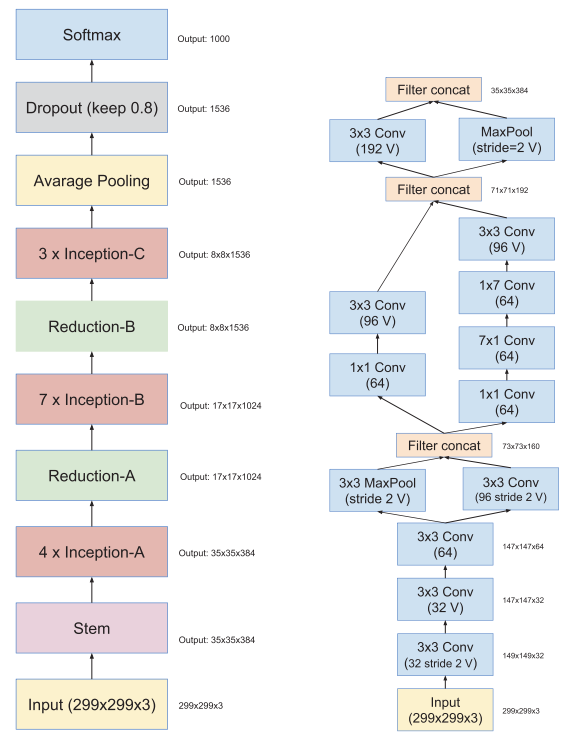
Gambar 2.8 VGGNet Konfigurasi Diagram

### Inception-V4 dan Inception-ResNet

Kombinasi dua arsitektur (Szegedy et al., 2016) dimana Arsitektur Inception-V4 menggunakan konsep dasar dari *inception* dimana arsitektur ini sangat memungkin untuk di-*tuning*, artinya sangat memungkinkan untuk dilakukan perubahan jumlah filter pada beberapa layer yang tidak mempengaruhi kualitas dari keseluruhan *network* yang dilatih.

Untuk mengoptimalkan kecepatan pelatihan, Inception-V4 menyesuaikan ukuran layer dengan hati-hati untuk menyeimbangkan perhitungan antara berbagai sub-jaringan model. Inception-V4 dapat dilatih tanpa mempartisi replika, dimana konsep ini berbeda saat awal pengenalan tensorflow.

Inception-V4 juga memungkinkan optimisasi memori terkini ke *backpropagation*, dengan cara mempertimbangkan dengan hati-hati tensor mana yang diperlukan untuk komputasi gradien dan penataan komposisi untuk mengurangi jumlah tensor tersebut. Konfigurasi penuh jaringan Inception-v4 di luar. Gambar 2.11 menunjukan skema keseluruhan dan konfigurasi stem pada Inception-V4.



Gambar 2.9 Konfigurasi Inception-V4

Sedangkan untuk Inception-ResNet2 menggunakan blok *inception* yang lebih ringan daripada *inception* asli. Dengan setiap blok *inception* yang kemudian diikuti oleh layer ekspansi filter (*convolution* 1 × 1 tanpa aktivasi) yang digunakan untuk meningkatkan dimensi penyimpan filter sebelum penambahan residual untuk menyesuaikan kedalaman input. Gambar 2.12 menggambarkan skema konfigurasi penuh jaringan Inception-Resnet-v2.

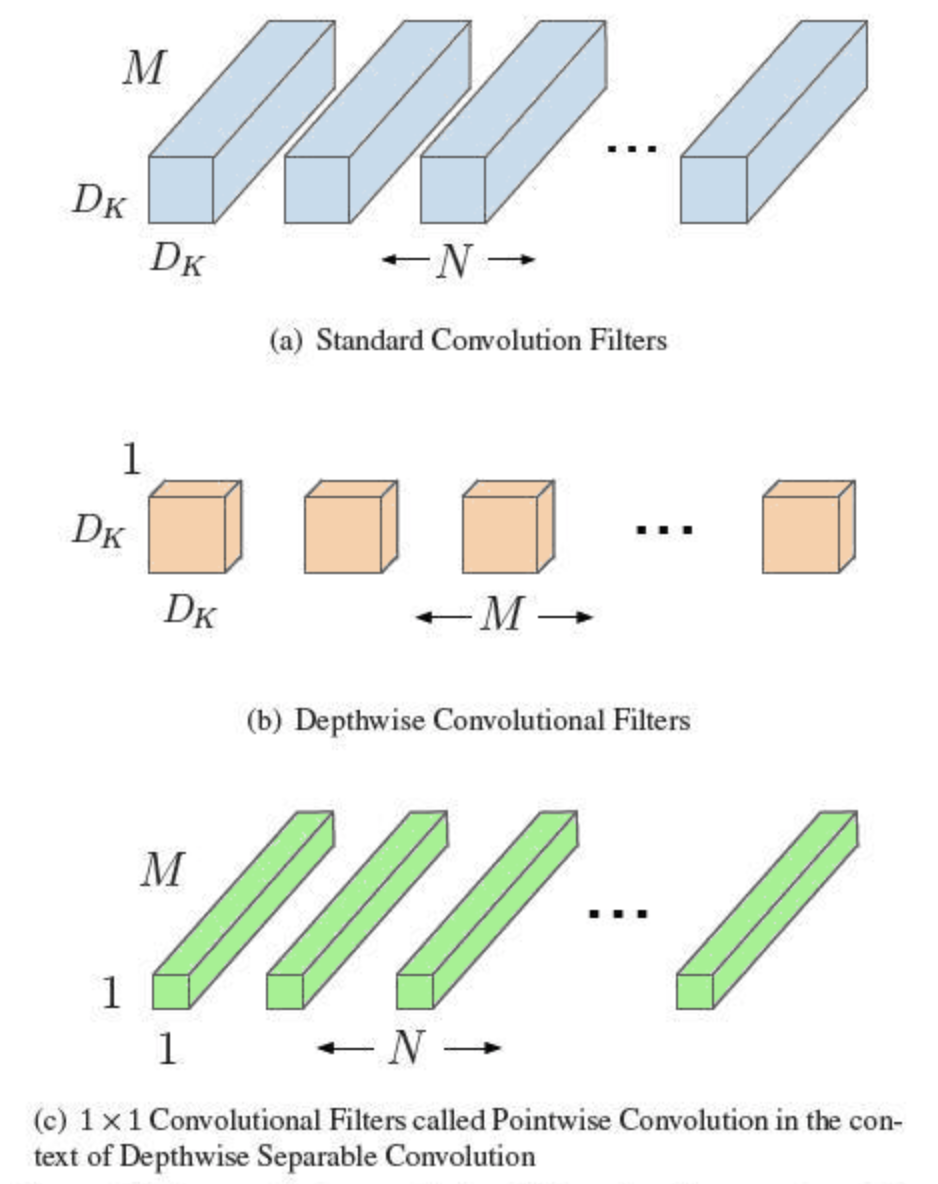


Gambar 2.10 Skema Inception-ResNet2

Di sebelah kiri adalah skema keseluruhan untuk Inception-Resnet-v2 dengan komposisi stem modul dan interior berbeda. Stem Inception-Resnet-v2 sama dengan jaringan Inception-v4 murni.

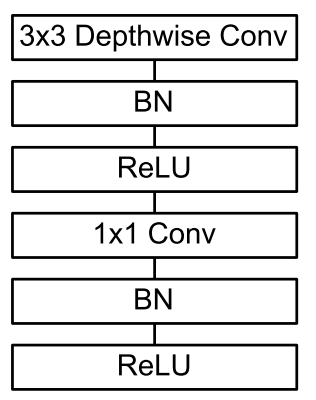
### MobileNet

Merupakan arsitektur CNN yang diciptakan untuk memenuhi kebutuhan design aplikasi pada perangkat mobile dan *embedded device*. MobileNet (Howard et al., 2017) mempunyai jaringan yang efisien dan berupa kumpulan jaringan yang terdiri dari dua hiper-parameter dengan tujuan membangun model dengan latensi yang sangat kecil dan rendah. MobileNets pada dasarnya dibuat dengan metode *depthwise separable convolutional* dengan tujuan untuk mengurangi perhitungan pada beberapa layer pertama.



Gambar 2.11 Depthwise Separable Convolutional

*Depthwise separable convolutional* merupakan bentuk *convolution* yang difaktorkan, yang mana memfaktorkan *convolution* standar menjadi *convolution* yang mendalam dan *convolution* 1 × 1 yang disebut *convolution* *pointwise*, untuk menggabungkan output dengan *deep convolutional*. *Convolution* standar menyaring dan menggabungkan input ke dalam set output baru dalam satu langkah. *Deep convolution* dipisahkan menjadi dua layer, yaitu layer terpisah untuk filter dan layer terpisah untuk *pooling*. Metode ini akan memberikan efek mengurangi perhitungan dan ukuran model secara drastis.

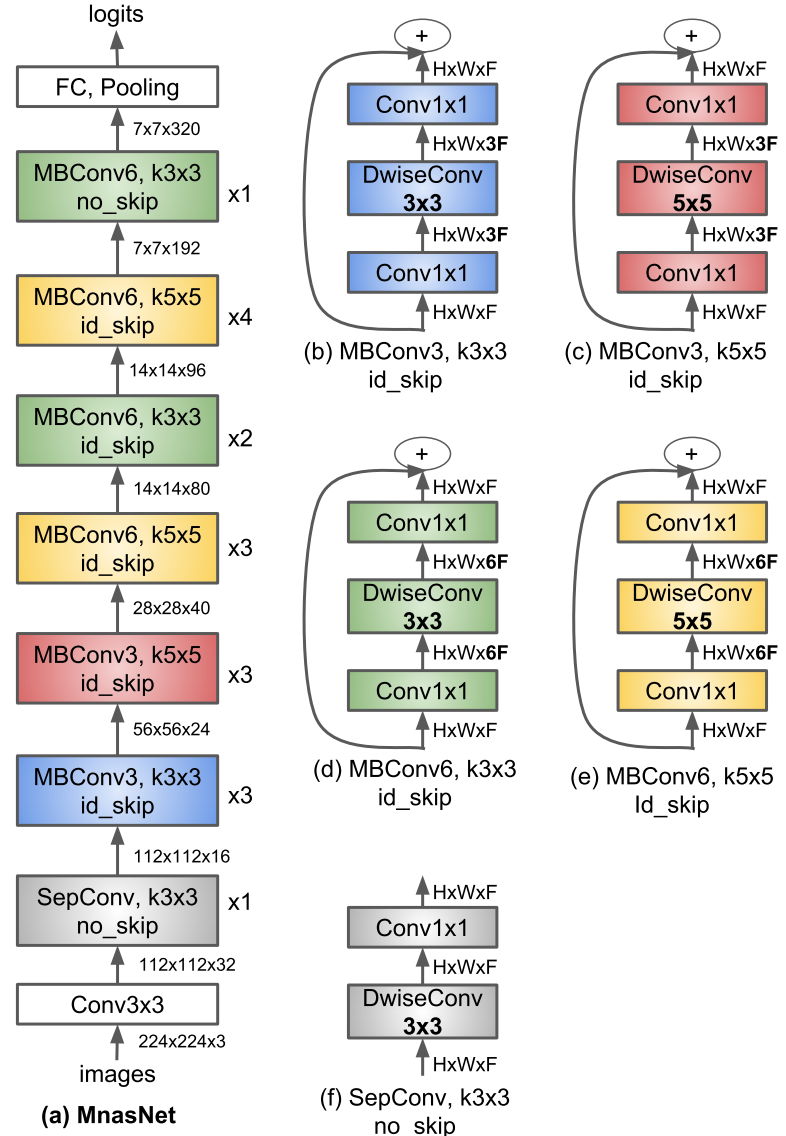


Gambar 2.12 Mobile Arsitektur Diagram

Gambar 2.10 mengilustrasikan diagram dari MobileNet arsitektur, dimana *deepwise seperable* *convolution* dengan *deepwise* dan *pointwise* layer diikuti oleh Bathchnorm dan aktivasi ReLU.

### MNasNet

Merupakan arsitektur Model CNN dengan metode ﻿*automated neural architecture search* dirancang untuk perangkat mobile yang hemat sumber daya (Tan, Chen, Pang, Vasudevan, & Le, 2018). Dengan latensi yang rendah sebagai tujuan utama sehingga pencarian dapat mengidentifikasi model yang mencapai pertukaran yang baik antara akurasi dan latensi. Berbeda dengan arsitektur sebelumnya dimana latensi selular dipertimbangkan melalui proksi lain, seringkali proksi yang sering tidak akurat (FLOPS), dengan langsung mengukur latensi inferensi sesungguhnya dengan mengeksekusi model pada platform tertentu dengan tujuan untuk lebih jauh mencapai keseimbangan yang tepat antara fleksibilitas dan ukuran ruang pencarian, maka menggunakan ruang pencarian hierarkis faktorisasi baru yang memungkinkan keanekaragaman layer di seluruh jaringan.



Gambar 2.13 MnasNet Arsitektur Digaram

Model MNasNet menggunakan lebih banyak *deep convolution* mendalam 5x5 daripada (Zhang et al. 2018; Huang et al. 2018; Sandler et al. 2018), dimana hanya 3x3 kernel yang biasanya digunakan. Bahkan, kernel 5x5 bisa lebih hemat sumber daya daripada dua kernel 3x3 untuk *convolution* terpisah yang mendalam. Secara formal, diberi bentuk input (H, W, M) dan bentuk output (H, W, N), misalkan C5 × 5 dan C3 × 3 menunjukkan biaya komputasi yang diukur dengan jumlah penambahan-ganda untuk *convolution* pemisahan terpisah dengan masing-masing kernel 5x5 dan 3x3.

MNasNet mencapai akurasi top-1 74,0% dengan latensi 76ms pada telepon Pixel, yang 1,5x lebih cepat dari MobileNetV2 (Sandler et al. 2018) dan 2,4 × lebih cepat dari NASNet (Zoph et al. 2018) dengan akurasi top-1 yang sama.

## Tinjauan Pustaka

Tinjuan pustaka akan dibagi menjadi tiga bagian yaitu tinjauan pada penelitian dalam mendeteksi penyakit pada tanaman dengan menggunakan *deep learning* arsitektur CNN secara umum, tinjauan pustaka pada penelitian yang mengkhususkan pada deteksi penyakit pada tanaman kopi, dan tinjauan pustaka pada penelitian-penelitian penggunaan *deep learning* CNN pada perangkat mobile atau *embedded device*.

### Deteksi dan Klasifikasi Penyakit Tanaman dengan CNN

Penelitian dengan topik deteksi dan klasifikasi penyakit tanaman dengan model CNN yang paling umum digunakan seperti AlexNet, GoogleNet, CIFAR-10, VGG, ResNet dll dengan menggunakan data set Plant Village dan memfokuskan pada salah satu jenis tanaman adalah sebagai berikut:

(Zhang et al., 2018) dalam penelitiannya membuat klasifikasi penyakit pada tanaman jagung dengan latar belakang masalah dimana penyakit daun jagung memiliki berbagai gejala yang menyebabkan kesulitan bagi petani yang tidak berpengalaman untuk mendiagnosis penyakit, sehingga memerlukan ahli patologi tanaman professional. Menggunakan dua model jaringan saraf *convolution* yang diperbaiki, yaitu GoogLeNet dan Cifar10. GoogleNet dan Cifar10 digunakan untuk melatih dan menguji sembilan jenis gambar daun jagung yang diperoleh dengan menyesuaikan parameter model, mengubah kombinasi penyatuan, menambahkan operasi pemutusan dan fungsi unit linear yang diperbaiki (Relu) fungsi, dan mengurangi jumlah pengklasifikasi. Kemudian hasil eksperimen dibandingkan dengan model yang tidak dimodifikasi. Dataset yang digunakan Plant Village dengan komposisi *train-test* adalah 80-20 (80% dari seluruh dataset digunakan untuk *train*, dan 20% untuk *test*. GoogLeNet dan Cifar10, masing-masing dapat mencapai akurasi identifikasi sangat tinggi yaitu 98,9% dan 98,8%.

Brahimi dkk mengangkat sebuah topik penelitian yang mengambil keunggulan dari metode visualisasi *deep learning* untuk memberikan informasi secara transparan sehingga peneliti mendapatkan penjelasan dan rincian mekanisme klasifikasi (Brahimi et al., 2018). Dalam penelitian ini dilakukan perbandingan antara kinerja arsitektur CNN dengan berbagai variasi arsitektur dan tipe pembelajaran yang berpengaruh kepada lamanya waktu yang dibutuhkan dalam proses pembelajaran. Perbandingan ini membantu para peneliti untuk memilih arsitektur mendalam terbaik untuk membangun sistem praktis untuk perlindungan penyakit tanaman. Data set menggunakan proses evaluasi akan dilakukan dengan menggunakan dataset PlantVillage. Untuk melatih dan mengevaluasi kinerja state-of-art CNN, menggunakan *open framework caffe*, *deep learning framework* yang dikembangkan oleh Berkley Vision and Learning Center. Di dalam penelitian dihasilkan akurasi paling tinggi 99,76% dengan model inception\_v3 dengan tipe *deep learning* dengan waktu 5.64 jam, sedangkan waktu yang paling singkat dalam proses pembelajaran yaitu 0.85 jam dengan tipe pembelajaran *shallow* pada model squeeznet, tetapi dengan akurasi lebih rendah yaitu 96,26%.

Sladojevic dkk dalam penelitiannya (Sladojevic et al., 2016) memberikan kontribusi penyediaan dataset yang dikumpulkan dengan cara men-*download* dari internet, dicari berdasarkan penyakit dan nama tanaman pada berbagai sumber dalam berbagai bahasa, seperti Latin, Inggris, Jerman, Serbia, dan Hongaria. Gambar dalam dataset dikelompokkan ke dalam 15 kelas yang berbeda. 13 kelas mewakili penyakit tanaman yang dapat ditentukan secara visual dari daun. Kemudian membuat *image processing* dan pelabelan dengan gambar yang digunakan untuk dataset adalah gambar yang diubah ukurannya menjadi 256 × 256 untuk mengurangi waktu *training*, yang secara otomatis dihitung dengan skrip tertulis dalam Python, menggunakan *open convolutioonal framework*. Adanya proses augmentasi dengan tujuan untuk meningkatkan dataset dan memperkenalkan sedikit distorsi pada gambar yang membantu mengurangi *overfitting* selama tahap *training*. Arsitektur CaffeNet dianggap sebagai titik awal, tetapi dimodifikasi dan disesuaikan untuk mendukung 15 kategori (kelas) nya. Layer terakhir diubah dan output dari layer softmax adalah parameter untuk persyaratan penelitian yang disajikan. Setelah menyempurnakan parameter jaringan, akurasi keseluruhan 96,3% tercapai, setelah training ke-100. Tingkat akurasi top-1 adalah 96,3% dan sukses top-5 adalah 99,99% setelah 100.000 iterasi. Model yang dikembangkan mampu mendeteksi keberadaan daun dan membedakan antara daun sehat dan 13 penyakit berbeda, yang dapat dilihat secara visual dan dapat didiagnosis secara visual.

(Mohanty et al., 2016) dalam penelitiannya fokus pada dua arsitektur popular, yaitu AlexNet dan GoogLeNet, membuat klasifikasi penyakit dengan menggunakan data set plantvillage, menganalisis 54.306 gambar daun tanaman, yang memiliki sebaran 38 label kelas, dimana setiap label kelas adalah pasangan penyakit tanaman, kemudian melakukan prediksi penyakit tanaman berdasarkan label kelasnya. Percobaan dilakukan dengan rangkaian uji *training*-pengujian, yaitu 80-20 (80% dari keseluruhan dataset digunakan untuk *training*, dan 20% untuk pengujian), 60–40 (60% dari keseluruhan dataset digunakan untuk *training*, dan 40% untuk pengujian), 50–50 (50% dari seluruh dataset digunakan untuk *training*, dan 50% untuk pengujian), 40–60 (40% dari seluruh dataset digunakan untuk *training*, dan 60% untuk pengujian) dan akhirnya 20–80 (20% dari seluruh dataset digunakan untuk *training*, dan 80% untuk pengujian). Akurasi keseluruhan bervariasi dari 85,53% dengan menggunakan AlexNet (Training from scratch, GrayScale, data 80–20) hingga 99,34% dengan percobaan dengan variasi GoogLeNet (TransferLearning, Color, data 80–20). Dari penelitian disimpulkan diantara arsitektur AlexNet dan GoogLeNet, dimana GoogLeNet secara konsisten berkinerja lebih baik daripada AlexNet dengan metode training *transfer learning* selalu menghasilkan hasil yang lebih baik.

(Brahimi et al., 2017) melakukan penelitian dengan membandingkan metode pembelajaran yang berbeda yaitu *shallow model’s* yang dikombinasikan dengan *hand-crafted model* (SVM, Random forest) dengan *deep model* CNN arsitektur (GoogLeNet dan AlexNet). *Deep model* (CNN) mengungguli yang *shallow* dimana akurasi terbaik dalam *shallow model* adalah 95,476 sedangkan 99,185 dalam *deep learning*. Nilai F makro (98.518) sangat tinggi pada *deep model* dibandingkan dengan nilai F makro dalam *shallow model* (94.185). Hasil ini menunjukkan kekuatan *deep model* dalam membangun fitur yang baik tanpa campur tangan manusia yang dengan keahlian. Sehingga dapat dikatakan *deep learning* yang dibangun secara otomatis memiliki kinerja yang lebih baik daripada fitur *hand-crafted*. Keakuratan pre-trained GoogleNet yang adalah 99,185 dan makro F1 =98,518 sedangkan AlexNet memiliki akurasi sama dengan 98,660 dan makro F=97,911. Keunggulan GoogleNet ini disebabkan oleh arsitektur baru yang digunakan dalam jaringan ini untuk meningkatkan non-linier tanpa ledakan jumlah bobot (Szegedy et al. 2015). GoogleNet menggunakan modul awal yang terinspirasi dari arsitektur *network-in-network* yang diusulkan dalam (Lin, Chen, dan Yan 2014). Modul *inception* menggunakan *convolution* dengan filter satu per satu (1 \* 1), yang mengarah pada penurunan kedalaman volume input.

(Garcia & Barbedo, 2018) dalam studi pengaruh data set dan variasi nya pada tingkat keefektifan *deep learning* untuk mendeteksi penyakit tanaman, mengangkat masalah banyak penelitian yang dilakukan menggunakan *tools* yang serupa pada sebuah dataset yang tidak mereproduksi berbagai kondisi yang diharapkan yang ditemukan di lapangan. Hal ini menjelaskan mengapa sebagian besar berhasil menunjukkan akurasi yang hampir sempurna seperti (Mohanty et al., 2016), (Brahimi et al., 2018) dan (Ferentinos, 2018), tanpa banyak variasi antar studi tersebut. Menyimpulkan bahwa meski pun menggunakan banyak variasi dalam hal spesies tanaman, penyakit dan kondisi pengambilan gambar, memiliki jumlah sampel yang terlalu kecil bagi CNN untuk secara menyeluruh menangkap karakteristik dan variasi yang terkait dengan masing-masing kelas. Perbandingan dari riset-riset dalam lima tahun terakhir mengenai *plant disease detection* dengan menggunakan metode CNN dapat dilihat pada Tabel 2.2.

Tabel 2.2 Perbandingan Riset Mengenai Plant Disease Detection

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Journal | Data Set | CNN Arsitektur | Akurasi |
| 1 | (Zhang et al., 2018) | -PlantVillage.  - CIFAR-10.  -Download Data sendiri. | - GoogleNet.  - Cifar10. | 98,9%.  98,8% |
| 2 | (Brahimi et al., 2018) | PlantVillage. | * AlexNet. * DenseNet. * Inception-v3. * ResNet-34. * SqueezeNet-1.1. * VGG13 | 94%-99%  96%-98%  91%-99%  94%-99%  92%-99% |
| 3 | (Sladojevic et al., 2016) | Semua gambar yang dikumpulkan untuk dataset diunduh dari Internet. | -CaffeNet | 96%-99% |
| 4 | (Mohanty et al., 2016) | PlantVillage | -AlexNet  -GoogleNet | 99,35% |
| 5 | (Brahimi et al., 2017) | PlantVillage | -AlexNet  -GoogleNet | GoogleNet=99,185-makro F1 =98,518  AlexNet memiliki akurasi sama dengan 98,660 dan makro F = 97,911 |
| 6 | (Garcia & Barbedo, 2018) | Pengaruh dataset dan variasi terhadap keefektifan DL | Literatur review |  |

### 

### Penelitian Deteksi Penyakit Tanaman Kopi

Beberapa penelitian yang khusus pada deteksi dan klasifikasi pada tanaman kopi antara lain, dengan menggunakan metode *expert system* berbasis logika fuzzy, dan *decision tree* menggunakan klasifikasi hirarkis, (Suhartono, Aditya, Lestari, & Yasin, 2013) membuat aplikasi yang dapat membantu peneliti atau pengamat yang bekerja di perkebunan kopi untuk mendiagnosis penyakit tanaman kopi. Pengetahuan tentang kopi, gejalanya, dan penyakitnya didapat dari pakar manusia dan kemudian diubah menjadi *decision tree*. Dari pengujian yang dilakukan, menghasilkan perhitungan akurasi sistem 85%.

Dengan menggunakan metode *image processing* dan *machine learning* (Mengistu, Alemayehu, & Mengistu, 2016), membuat dataset berupa gambar daun kopi sebanyak 9100, yang diambil menggunakan kamera, membuat tiga kelas penyakit daun tanaman kopi yaitu Coffee LeafRust, Coffee Berry Disease dan Coffee Wilt Disease. Gambar diolah dengan cara melakukan *pre-processing*, seperti menghapus frekuensi rendah, *noise* latar belakang, menormalkan intensitas partikel individu pada gambar, menghilangkan refleksi, dan menutupi bagian gambar. Segmentasi gambar dilakukan dengan menggunakan teknik K-means dan algoritma genetika digunakan untuk memilih fitur dengan teknik level *co-occurrence matrix* dan warna. Dengan ratio pemisahan data pelatihan (70%) dan pengujian (30%), menggunakan tools MATLAB2013Ra, proses klasifikasi menggunakan metode deteksi tepi sobel untuk menemukan batas gambar. Pengujian kinerja klasifikasi menggunakan model *Artificial Neural Network, K-Nearest Neighbor, Naive Bayes dan combination of Radial Basis Function,* dan *Self Srganizing Map*. Hasil tingkat akurasi KNN, ANN, Naive Bayes dan kombinasi RBF dan SOM masing-masing adalah 58,16%, 79,04%, 53,47%, dan 90,07%.

Metode *Multi Spectral Image* (MSI-2) digunakan oleh (Chemura, Mutanga, & Dube, 2016) memanfaatkan potensi pita sentinel-2 untuk membedakan tingkat infeksi karat daun kopi. Dengan melakukan sampel ulang pada spektrum yang ada di lapangan ke pengaturan pita Sentinel-2, kemudian dievaluasi menggunakan algoritma random forest (RF) dan *partial least squared analysis* (PLS-DA) dengan dan tanpa optimasi variabel. Diujikan dengan menggunakan variabel yang sama menghasilkan indeks vegetasi yang diterima Sentinel-2 *Multispectral Imager* (MSI) menghasilkan akurasi 76,2% lebih tinggi dari pita spektral mentah dengan akurasi 69,8%. Penelitian ini juga mendapatkan kesimpulan penting pengaturan variabel pita spektral dan indeks vegetasi turut mempengaruhi keakuratan dalam pengujian dimana diidentifikasi sebagai variabel penting dalam pemisahan tingkat infeksi karat daun, Meningkatkan akurasi perbedaan karat daun menjadi 79,4% dan 82,5%.

Masih berbasis teknik dengan menggunakan sensor *multi spectral image* penelitian pada tanaman kopi yaitu Spectroradiometer Data (Chemura, Mutanga, Sibanda, & Chidoko, 2018) dengan menggunakan ﻿*sensor satellite-based Sentinel-2 MultiSpectral Imager* (MSI). Tujuan dari studi yang dilakukan di rumah kaca menggunakan spektroradiometer genggam ini adalah untuk menilai kegunaan dari *waveband* yang sama seperti yang digunakan dalam MSI Sentinel-2 dalam menilai dan modeling karat daun kopi *leafrust* berdasarkan radial non-linear Algoritma *machine learning* dasar *function-partial least square* (RBF-PLS).

Metode *expert system* dan *hyperspectral imaging system* (Castro et al., 2018) menggunakan *Advance Hyperspectral Images System* (HIS’s) mengklasifikasi penyakit karat daun menjadi lima kelas yaitu *Healthy, Initial, Intermediate, Advanced,* dan *Necrotic*. HIS’s pada dasarnya menggunakan metode khusus *machine learning* seperti *Artificial Neural Networks (ANN’s)*, *Decision Tree (DT)*, *K-means (KM)*, *K-Nearest Neighbour (KNN)*, dan *Support Vector Mesin (SVM)* untuk melakukan data analisis. Teknik HIS’s mencapai akurasi 90,30% dengan metode *Decision Tree*, 93% dengan metode K-NN, dan akurasi tertinggi 94,70% dicapai dengan metode *Support Vector Mesin* *(SVM)*.

### CNN pada Perangkat Mobile dan Embedded Device.

Penelitian model *deep learning* yang dapat dieksekusi pada *runtime selular* saat ini menjadi bidang penelitian yang banyak dilakukan dalam bidang kecerdasan buatan. Adanya keterbatasan kemampuan kinerja dan konsumsi energi yang diperlukan untuk menjalankan komputasi algoritma secara intensif menjadi hambatan penggunaan *deep learning* pada perangkat mobile, hal ini memicu penelitian untuk mengembangkan model yang efisien dan hemat energi.

CNNdroid diajukan oleh (Salar, Oskouei, Golestani, & Hashemi, 2016) berupa sebuah Library GPU-accelerated, untuk mengeksekusi *deep learning* CNN yang sudah dilatih pada perangkat mobile berbasis Android. Hasil evaluasi empiris menunjukkan bahwa CNNdroid mencapai kecepatan hingga 60X dan penghematan energi 130X pada perangkat mobile saat ini. CNNdroid mendukung semua jenis umum layer CNN, yaitu *convolution*, *max pooling*/rata-rata, FC-Layer, ReLU, *batch normalization*, dan *softmax*. Di dalam CNNdroid juga menyediakan satu set skrip yang dapat merubah menjadi format CNNdroid dari model-model yang dilatih dan disediakan oleh *Library desktop/server* umum, yaitu Caffe, Torch, dan Theano, sehingga model-model tersebut dapat dijalankan di perangkat mobile.

Penelitian (Yanai, Tanno, & Okamoto, 2016) melakukan evaluasi dan menguji memeriksa arsitektur CNN yang cocok untuk dijalankan pada perangkat mobile dan mengusulkan *multi-scale network-in-networks* (NIN) dimana pengguna dapat menyesuaikan pertukaran antara waktu pengenalan (*recognition*) dan akurasi. Dengan menerapkan aplikasi selular multi-utas pada iOS dan Android menggunakan instruksi NEON SIMD atau library BLAS untuk perhitungan cepat layer *convolutional*, dan membandingkannya dalam hal waktu pengenalan pada perangkat mobile. Penelitian ini menghasilkan kesimpulan bahwa BLAS lebih baik untuk iOS, sementara NEON lebih baik untuk Android, dan juga menyimpulkan dengan mengurangi ukuran gambar input dengan mengubah ukuran sangat efektif untuk mempercepat pengenalan berbasis CNN.

Masih mengangkat rumusan masalah yang sama dalam penggunaan algoritma *deep learning* pada perangkat mobile, yaitu akurasi tinggi, latensi pemrosesan tinggi dan konsumsi daya, penelitian (Huynh, Lee, & Balan, 2017) mengusulkan *“DeepMon”*, sebuah sistem inferensi *deep learning selular* untuk menjalankan berbagai kesimpulan *deep learning* pada perangkat mobile dengan cara yang cepat dan hemat energi. Penelitian merancang serangkaian teknik optimisasi untuk secara efisien dengan Teknik *offloading* layer *convolution* ke GPU selular dan mempercepat pemrosesan. Hasil eksperimen menunjukkan bahwa *“DeepMon”* dapat mengklasifikasi gambar melalui model pembelajaran mendalam VGG-16 dalam 644ms pada Samsung Galaxy S7, tanpa menggangu privasi atau pun menambah biaya jaringan.

Penelitian dengan tujuan yang sama untuk mengatasi masalah latensi dan hemat energi dalam penerapan *deep learning* CNN pada perangkat mobile (Cao, Balasubramanian, & Balasubramanian, 2017) terus menggali optimisasi untuk menjalankan *recurrent neural network* (RNN) secara lokal di perangkat mobile. kebanyakan Model RNN banyak digunakan untuk NLP, terjemahan mesin, dan tugas-tugas lainnya. Biasanya aplikasi mobile yang ada yang menggunakan model RNN dijalankan di *cloud*. Untuk mengatasi masalah privasi dan efisiensi, penelitian menunjukkan bagaimana model RNN dapat dijalankan secara lokal di perangkat mobile. Pekerjaan yang ada pada *porting* model *deep learning* ke perangkat mobile fokus pada *Convolution* Neural Networks (CNNs) dan tidak dapat diterapkan secara langsung ke model RNN. Dengan kerangka kerja pengoptimalan khusus selular, MobiRNN menerapkan *offloading* GPU khusus untuk GPU selular. Evaluasi menggunakan model RNN untuk pengenalan aktivitas menunjukkan bahwa MobiRNN secara signifikan mengurangi latensi menjalankan model RNN di perangkat mobile.

Pengembangan perangkat lunak yang dilakukan oleh ﻿MyungJoo Ham dan kawan-kawan dari badan riset Samsung (Ham et al., 2019) mengembangkan *NNStreamer* yaitu sistem perangkat lunak yang menangani jaringan saraf sebagai filter *stream line*, menerapkan paradigma pemrosesan *stream* untuk aplikasi jaringan saraf. Menjadi tren baru dengan penyebaran luas aplikasi *deep neural network* yang merupakan AI pada perangkat. ﻿ *NNStreamer* secara efisien menangani *neural network* dengan *stream pipelines* yang kompleks pada perangkat, mampu meningkatkan kinerja keseluruhan secara signifikan dengan upaya minimal. NNStreamer juga menyederhanakan implementasi *neural network pipeline* dan memungkinkan penggunaan ulang *filter streaming multimedia* langsung, sehingga mengurangi biaya *development* secara signifikan. NNStreamer akan di-*deployed* dengan produk yang segera dirilis dan perangkat lunak *open source* dapat diterapkan pada beragam arsitektur perangkat keras dan platform perangkat lunak. NNStreamer bukan prototipe untuk percobaan melainkan perangkat lunak dengan jadwal segera dan mempunyai kepentingan secara komersial sehingga harus siap secara kualitas.

### Rangkuman Tinjauan Pustaka

Dari penelitian yang dilakukan sebelumnya, dihasilkan keakuratan yang cukup tinggi, bahkan mendekati nilai sempurna yaitu 99,3%. Adanya penelitian pada model-model yang dapat dijalankan atau digunakan pada perangkat mobile, yang dapat mendorong pemanfaatannya di lapangan dengan melakukan deteksi menggunakan perangkat mobile mendorong penelitian lanjutan yang dapat membandingkan model-model yang sudah ada dengan model yang dapat berjalan di perangkat mobile dan juga akan menguji dengan menggunakan dataset pada tanaman kopi.

Sedangkan penelitian mengenai deteksi tanaman dengan menggunakan metode *Deep Learning* CNN pada perangkat mobile dan khususnya untuk tanaman kopi belum ada yang secara khusus melakukannya sehinga mendorong penelitian ini untuk mengevaluasi model-model khusus *Deep Learning* CNN yang dapat dijalankan khusus pada perangkat mobile untuk membantu mengklasifikasi penyakit tanaman khususnya pada tanaman kopi, sehingga dapat mengoptimalkan kebutuhan komputasi dan penghematan sumber daya tanpa mengurangi akurasi, privasi dari perangkat mobile itu sendiri.

# METODOLOGI

## Kerangka Pikir

Kerangka pikir dalam penelitian ini terdiri dari tiga tahap yaitu: tahap perencanaan, tahap implementasi, dan tahap *deployment*. Ilustrasi pada Gambar 3.1 menggambarkan urutan proses kerja penelitian ini.



Gambar 3.1 Kerangka Pikir

Diawali dari tahap perencanaan untuk dapat mengidentifikasi masalah dengan melakukan studi literatur untuk dijadikan topik penelitian, kemudian dilanjutkan ke tahap implementasi dengan mengumpulkan data untuk diproses sehingga mendapatkan hasil evaluasi dan *graph file* hasil dari pembelajaran. Tahap berikutnya yaitu *deployment* dan pengujian pada perangkat mobile dari masing-masing model yang diujikan.

Tahap awal pada penelitian ini adalah perencanaan, yang berisi gagasan atau ide yang melatarbelakangi penelitian seperti tertuang dalam subbab 1.1. yaitu memberikan kontribusi di bidang agrikultur khususnya pada perkebunan tanaman kopi melalui pembuatan sistem deteksi penyakit pada tanaman kopi. Langkah selanjutnya adalah studi literatur dengan mengumpulkan jurnal penelitian mengenai identifikasi penyakit pada tanaman. Dari studi literatur teridentifikasi masalah untuk melakukan penelitian lanjutan yang memberikan fokus pada aplikasi teknik *deep learning* untuk digunakan pada perangkat mobile yang memberikan kontribusi langsung bagi petani dalam sistem deteksi dini penyakit tanaman khususnya pada tanaman kopi di Indonesia.

Tahap implementasi meliputi implementasi dari *state of art Convolutional Neural Network* yang akan dibagi dalam tiga proses utama yaitu pengumpulan data, implementasi berupa pembuatan program *Deep Learning* CNN, dan evaluasi dengan melakukan *training* pada data dan tingkat akurasi.

Tahap akhir yang akan dilakukan adalah *deployment* menggunakan perangkat mobile. Pada tahap ini akan dilakukan perbandingan model yang digunakan dan evaluasi kebutuhan *resource* dari perangkat mobile.

## Deteksi Penyakit Tanaman Dengan Deep Learning

Dalam penelitian ini *state of the art* CNN model akan diujicoba pada enam arsitektur berikut: MobileNet (Howard et al., 2017), ResNet (He et al., 2016), PNASNet-5 Mobile (Liu et al., 2018), NASNet-A Mobile (Zoph et al., 2017), VGG (Simonyan & Zisserman, 2015) dan Inception V4 (Szegedy et al., 2016). Sistem identifikasi pada penyakit tanaman metode *deep learning* model CNN akan diimplementasi dan diteliti pada *pre-trained* arsitektur dimana arsitektur ini telah di-*training* pada dataset *image* klasifikasi ILSVRC-2012-CLS. Arsitektur yang dipilih khusus pada arsitektur yang telah dikembangkan dan disediakan oleh tensorflow-lite yang dioptimalkan untuk berjalan pada perangkat mobile. Model-model ini diciptakan dengan *setting* untuk mobile dengan menekan kemampuan CNN model dengan ukuran input 244x244 dan jumlah dari operasi *multiply-add* di bawah 600M (Liu et al., 2018) sehingga dapat berjalan dengan optimal pada perangkat mobile.

### Implementasi Sistem

Dalam tahap implementasi sistem akan melakukan perancangan dan pembuatan sistem diilustrasikan dalam diagram alirpada Gambar 3.2.



Gambar 3.2 Diagram Alir Implementasi

Dalam persiapan data, akan dikumpulkan dataset public PlantVillage dan dataset pribadi daun tanaman kopi yang akan dipisah menjadi dua bagian yaitu *data training* dan *data testing*. Pelabelan akan dilakukan menggunakan *function* untuk meng-*encode* label. Proses selanjutnya adalah *pre-prosessing* data dengan melakukan *cropping, resize* data menjadi 224x224 untuk arsitektur MobilNet, ResNet, PASNet, VGG. Sementara untuk *inception* data di *resize* menjadi 299x299 pixel.Augmentasi dilakukan untuk memperbanyak data. Proses selanjutnya adalah *training* dengan menentukan model yang akan digunakan dan *parameter list* yang akan ditentukan seperti *learning rate* dan jumlah *epoch training*. Pada proses ini keakuratan akan dihitung menggunakan *loss function*, kategori *cross entropi* dan perhitungan waktu *training*. Proses *testing* dengan menggunakan PC akan dilakukan setelah proses *training* selesai dengan memberikan data *testing* sehingga dapat dilakukan evaluasi akurasi.

Untuk melatih kinerja *state of the art* CNN, penelitian ini menggunakan *open source framework Tesorflow*, khususnya tensorflow-lite Deep Learning model, Python 3.6, Pycharm community edition, dan Android developer. Menggunakan PC dengan spesifikasi seperti pada Tabel 3.1.

Tabel 3.1 Spesifikasi Mesin untuk Proses Training

|  |  |  |
| --- | --- | --- |
| No | Hardware dan software | Spesifikasi |
| 1 | Memory | 32Gb |
| 2 | Processor (CPU) | Intel Core i7-8750 CPU @ 2.20 GHz x8 |
| 3 | Graphics (GPU) | GeForce GTX 1050 4GB |
| 4 | Operating system Linux | Ubuntu 18.04 64 bits |

### Tahap Deployment CNN pada Perangkat Mobile



Gambar 3.3 Diagram Alir deployment pada Mobile Deployment

Proses *deployment* pada perangkat mobil diilustrasikan dalam Gambar 3.3, Setelah selesai melakukan *training* ulang model, *deployment* akan dilakukan ke perangkat mobile. Proses konversi untuk mengoptimalkan file yang berjalan di perangkat mobile, file yang dihasilkan dari proses training, misalnya “*retrained\_graph.pb”* akan dikonversi menjadi “*retrained\_graph.lite”* menggunakan program "*TensorFlow Lite Optimizing Converter*" atau *tflite\_convert* yang merupakan konverter grafik baru yang sekarang disertakan dengan instalasi TensorFlow.

TFLite menggunakan format serialisasi yang berbeda dari TensorFlow biasa. TensorFlow menggunakan Protokol Buffer sedangkan TFLite menggunakan FlatBuffers. Manfaat utama FlatBuffers adalah dapat dipetakan dalam memori dan digunakan langsung dari *disk* tanpa dimuat dan diuraikan. Sehingga waktu *startup* menjadi jauh lebih cepat dan memberikan pilihan kepada sistem operasi untuk *load* dan *unload pages* yang diperlukan dari file model, untuk menghindari matinya aplikasi ketika kehabisan memori. File hasil konversi selanjutnya akan disalin ke *folder asset* pada *folder assets* di perangkat mobile.

*Deployment* akan menggunakan perangkat mobile dengan spesifikasi seperti Tabel 3.2.

Tabel 3.2 Spesifikasi Perangkat Mobile

|  |  |  |
| --- | --- | --- |
| No | Hardware and software | Characteristics |
| 1 | Memory | 4 GB |
| 2 | Processor (CPU) | Quad-core Snapdragon 820 or Octa-core Samsung Exynos 8 |
| 4 | Operating system Linux | Android 8.0 Marshmallow |
| 5 | Rear Camera | 12MP f/1.7 1.4-micron pixels  OIS |

## Pengumpulan Data

Data yang akan digunakan dalam penelitian adalah dataset PlantVillage yang sudah direkayasa dengan dikumpulkan, dipisahkan, dan kemudian diberikan label (Hughes & Salathe, 2015). Dataset ini mencakup 54.323 gambar dari 14 spesies tanaman dengan 38 kelas daun dengan penyakit dan daun sehat. Dalam penelitian ini memisahkan gambar dengan membuat label ke dalam 38 kelas seperti Tabel 3.3.

Tabel 3.3 Dataset PlantVillage

|  |  |  |
| --- | --- | --- |
| No | Nama Gambar | Jumlah |
| 1 | Apple Scab, Venturia inaequalis | 630 |
| 2 | Apple Black Rot, Botryosphaeria obtuse | 621 |
| 3 | Apple Cedar Rust, Gymnosporangium juniperi-virginianae | 275 |
| 4 | Apple healthy | 1645 |
| 5 | Blueberry healthy | 1502 |
| 6 | Cherry healthy | 854 |
| 7 | Cherry Powdery Mildew, Podoshaera clandestine | 1052 |
| 8 | Corn Grey Leaf Spot, Cercospora zeae-maydis | 513 |
| 9 | Corn Common Rust, Puccinia sorghi | 1192 |
| 10 | Corn healthy | 1162 |
| 11 | Corn Northern Leaf Blight, Exserohilum turcicum | 985 |
| 12 | Grape Black Rot, Guignardia bidwellii | 1180 |
| 13 | Grape Black Measles (Esca), Phaeomoniella aleophilum, Phaeomoniella chlamydospora | 1383 |
| 14 | Grape Healthy | 423 |
| 15 | Grape Leaf Blight, Pseudocercospora vitis | 1076 |
| 16 | Orange Huanglongbing (Citrus Greening), Candidatus Liberibacter spp | 5507 |
| 17 | Peach Bacterial Spot, Xanthomonas campestris | 2297 |
| 18 | Peach healthy | 360 |
| 19 | Bell Pepper Bacterial Spot, Xanthomonas campestris | 997 |
| 20 | Bell Pepper healthy | 1478 |
| 21 | Potato Early Blight, Alternaria solani | 1000 |
| 22 | Potato healthy | 152 |
| 23 | Potato Late Blight, Phytophthora infestans 1000 | 1000 |
| 24 | Raspberry healthy | 371 |
| 25 | Soybean healthy | 5090 |
| 26 | Squash Powdery Mildew, Erysiphe cichoracearum | 1835 |
| 27 | Strawberry Healthy | 456 |
| 28 | Strawberry Leaf Scorch, Diplocarpon earlianum | 1109 |
| 29 | Tomato Bacterial Spot, Xanthomonas campestris pv. Vesicatoria. | 2127 |
| 30 | Tomato Early Blight, Alternaria solani | 1000 |
| 31 | Tomato Late Blight, Phytophthora infestans | 1591 |
| 32 | Tomato Leaf Mould, Passalora fulva | 1909 |
| 33 | Tomato Septoria Leaf Spot, Septoria lycopersici | 952 |
| 34 | Tomato Two Spotted Spider Mite, Tetranychus urticae | 1771 |
| 35 | Tomato Target Spot, Corynespora cassiicola | 1676 |
| 36 | Tomato Mosaic Virus | 1404 |
| 37 | Tomato Yellow Leaf Curl Virus | 373 |
| 38 | Tomato healthy | 5375 |



Gambar 3.4 Contoh Kelas Daun Plantvillage

Gambar 3.4. merupakan contoh gambar daun yang telah diklasifikasikan menjadi 38 kelas. Juga akan menambahkan satu kelas ke dalam dataset berisikan gambar daun sehat untuk membedakan daun sehat dari yang rusak.

Kelas tambahan berupa latar belakang yang dirancang oleh Stanford university (Gould, Fulton, & Koller, 2009) yang bermanfaat untuk mendapatkan klasifikasi yang lebih akurat, sehingga jaringan saraf yang dalam dapat dilatih untuk membedakan daun dari sekitarnya. Data set daun tanaman kopi di Indonesia akan ditambahkan sebagai obyek penelitian dengan mengambil data langsung dari perkebunan bekerja sama dengan Pusat Penelitian Kopi dan Kakao di Jember, Jawa Timur. Daun tanaman kopi juga diambil dari basis data yang digunakan yang disediakan secara gratis untuk tujuan akademik di repositori di alamat <https://www.digipathos-rep.cnptia.embrapa.br/> seperti yang dilakukan oleh (Garcia & Barbedo, 2018) diklasifikasi menjadi enam kelas yang berbeda seperti dalam Gambar 3.5.



Gambar 3.5 Dataset Tambahan Daun Kopi

Dalam penelitian ini daun kopi dipisahkan dengan memberikan label menjadi enam kelas, membentuk dataset karat daun seperti dalam Table 3.4.

Tabel 3.4 Dataset Daun Kopi

|  |  |  |
| --- | --- | --- |
| No | Nama Gambar | Jumlah |
| 1 | Daun kopi sehat | 750 |
| 2 | Daun kopi karat daun | 75 |
| 3 | Daun kopi brown spot | 39 |
| 4 | Daun kopi cercospora spot | 55 |
| 5 | Daun kopi bacterial blight | 37 |
| 6 | Daun kopi dengan miner | 12 |

Setelah pengumpulan data akan dilakukan pra-pemrosesan yang meliputi pelabelan dan augmentasi data. Proses pelabelan terdiri dari anotasi gambar yang dikumpulkan oleh ahli dibidang penyakit tanaman. Teknik augmentasi bertujuan memperbanyak jumlah data set dan menambahkan variasi. Teknik augmentasi yang digunakan meliputi transformasi geometrikal (*resizing, crop, rotation, horizontal flipping*) dan transformasi intensitas (*contrast & brightness enhancement, colour, noise*).

## Evaluasi State of The Art CNN

Evaluasi akan dilakukan dalam dua tahap, pertama pada tahap implementasi yaitu pada proses *training* di PC berupa akurasi dan *runtime* dan kedua pada tahap *deployment* yaitu pada proses deteksi di perangkat mobile dengan mengukur akurasi, *runtime*, dan konsumsi energi.

### Evaluasi Tahap Implementasi

Dalam tahap implementasi ada tiga hal utama yang akan dilakukan yaitu: menyiapkan dataset, proses *training* dengan metode *pretrained*, dan mengevaluasi model yang sudah di-*training* dan membuat analisis hasil yang akan dibahas secara rinci. *Pretrained*, merupakan *training* CNN mendalam yang dilakukan menggunakan dataset besar seperti ImageNet terlebih dahulu, sebelum dilakukan proses *training* menggunakan dataset sendiri. *Pretraining* dilakukan untuk mempersiapkan CNN dengan mentransfer pembelajaran dari dataset besar ke klasifikasi penyakit tanaman. Tahap ini digunakan untuk menangani kurangnya data dalam proses *training*.

#### Training Runtime

*Training runtime* adalah waktu yang dibutuhkan untuk menjalan proses *training*. Evaluasi akan dilakukan dengan menghitung waktu total yang dibutuhkan untuk satu proses *training*, dan juga menghitung rata-rata per *epoch* dari model-model CNN menggunakan strategi *pretarined*. Putaran per *epoch training* di ilustrasikan Gambar 3.6, dimulai dari *schedule epoch* dijalankan kemudian menentukan *hyperparameter* seperti jumlah ukuran *batch*, jumlah iterasi yang akan dilakukan, kemudian dilakukan optimisasi menggunakan *optimizer*, *loss* dihitung, akurasi, akurasi dicatat, dan diakhiri dengan *end\_epoch*.

Gambar 3.6 Siklus Proses Training

Proses *training* akan menjalan *script retrain* yang disediakan dari Tensorflow, secara *default* skrip ini akan menjalankan 4.000 langkah pelatihan. Setiap langkah memilih sepuluh gambar secara acak dari set pelatihan, menemukan *bottleneck* dari *cache*, dan memasukkannya ke dalam lapisan terakhir untuk mendapatkan prediksi. Prediksi tersebut kemudian dibandingkan dengan label dari folder-folder dataset untuk memperbarui bobot lapisan akhir melalui proses *back propagation*. Evaluasi akan dilakukan dalam proses *training* dengan menentukan *hyperparemeter* untuk mengukur seberapa besar pengaruh *hyperparameter* yang ditentukan terhadap *runtime* proses *training*, seperti dalam Tabel 3.5.

Tabel 3.5 Evaluasi Kebutuhan Waktu Training

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Batch Size  (100) | Jumlah Langkah (4000) | Learning Rate (0.1) | Runtime |
| MobileNet | x | x | x | x |
| ResNet | x | x | x | x |
| [PNASNet-5 Mobile](https://arxiv.org/abs/1712.00559) | x | x | x | x |
| [NASNet-A Mobile](https://arxiv.org/abs/1707.07012) | x | x | x | x |
| [VGG](https://arxiv.org/abs/1707.07012) | x | x | x | x |
| Inception V4 | x | x | x | x |

#### Akurasi

Nilai akurasi sebuah model biasanya ditentukan setelah parameter model selesai dipelajari dan diperbaiki dan tidak ada lagi proses pembelajaran yang dilakukan. Kemudian sampel tes dimasukkan ke model dan jumlah kesalahan yang dibuat model dicatat, setelah dibandingkan dengan target yang sebenarnya. Kemudian persentase kesalahan klasifikasi dihitung.

Dalam proses *training* akan memisahkan dataset antara data training, validasi, dan testing, dengan komposisi pemisahan data dapat diubah dengan menggunakan parameter, misalkan 80% *training* – 10% validasi dan 10% *testing*. Proses *training* yang berjalan akan menghasilkan serangkaian output dari setiap langkah, masing-masing menunjukkan akurasi pelatihan, akurasi validasi, dan *cross entrophy*.

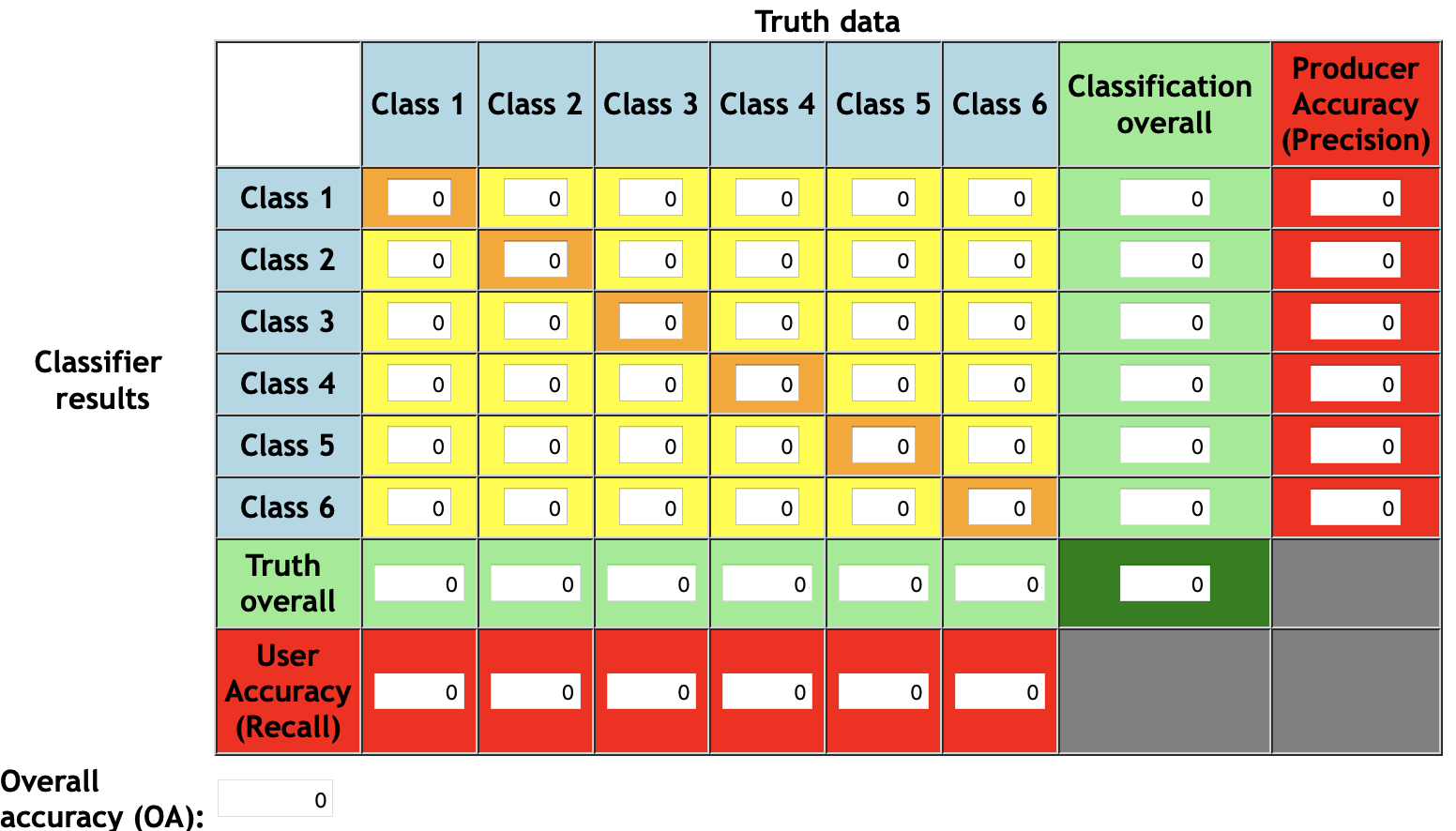
* Akurasi pelatihan menunjukkan berapa persen dari gambar yang digunakan dalam kumpulan pelatihan saat ini diberi label dengan kelas yang benar.
* Akurasi validasi adalah ketepatan pada kelompok gambar yang dipilih secara acak dari set yang berbeda. Perbedaan utama adalah bahwa keakuratan pelatihan didasarkan pada gambar yang telah dapat dipelajari oleh jaringan sehingga jaringan dapat menyesuaikan dengan *noise* dalam data pelatihan.
* *Cross entropy* adalah fungsi *loss* yang memberikan pandangan sekilas tentang seberapa baik proses pembelajaran berkembang.

Pengukuran akhir dari akurasi jaringan adalah dengan mengukur akurasi pada dataset diluar data pelatihan yang diukur dengan akurasi validasi. Jika akurasi *train* tinggi tetapi akurasi validasi rendah, itu berarti jaringan *overfitting* dan menghafal fitur-fitur tertentu dalam gambar pelatihan yang umumnya tidak membantu menambah akurasi. Evaluasi penghitungan akurasi untuk enam arsitektur CNN di diilustrasikan dengan Tabel 3.6.

Tabel 3.6 Evalusi Model Re-Training

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Akurasi Train | Akurasi  Validasi | Cross Validasi | Test Akurasi (%) |
| MobileNet | x | x | x | x |
| ResNet Mobile | x | x | x | x |
| [PNASNet-5 Mobile](https://arxiv.org/abs/1712.00559) | x | x | x | x |
| M[NASNet-A Mobile](https://arxiv.org/abs/1707.07012) | x | x | x | x |
| [VGG](https://arxiv.org/abs/1707.07012) | x | x | x | x |
| Inception V4 | x | x | x | x |

### Sedangkan untuk mengevaluasi akurasi klasifikasi setiap model terhadap enam kelas penyakit dari dataset tanaman kopi, digunakan fungsi *confusion\_matrix* dengan menghitung *confusion matriks* pada setiap gambar yang sesuai dengan kelas yang benar. *Output training* akan dikalkulasi untuk mengukur akurasi per kelas sebagi contoh menggunakan *calculator matrix online* (Vanetti, 2007) pada Gambar 3.7.



Gambar 3.7 Confusion Matrix Calculator

### Evaluasi Tahap Deployment

Evaluasi akan dilakukan secara empiris menggunakan perangkat mobile Huawei Nova 3i dengan OS system Android 8, terhadap model-model yang di optimalkan pada perangkat mobile yaitu MobileNet, RasNet Mobile, [PNASNet-5 Mobile](https://arxiv.org/abs/1712.00559), [NASNet-A Mobile](https://arxiv.org/abs/1707.07012), [VGG](http://arxiv.org/abs/1409.1556.pdf) dan Inception dengan menggunakan *library* yang disediakan oleh Tensorflow-lite framework.

Dalam penelitian ini evalusi terhadap penggunaan metode CNN pada perangkat mobile akan mengukur performa (akurasi dan *runtime*) dan kebutuhan sumber daya komputasi (CPU, memori dan konsumsi energi). Evaluasi akan dilakukan dengan mengeksekusi per satu gambar dan per 10 gambar secara bertahap. Untuk mendapatkan *baseline* yang baik perangkat mobile akan di *restart* kemudian, *battery* terisi penuh dan menggunakan *airplabe mode* untuk memastikan percobaan dilakukan secara lokal di perangkat mobile tanpa memerlukan koneksi internet.

#### Test Runtime Performance dan Akurasi

Evaluasi juga akan dilakukan dalam proses deteksi dan kebutuhan akan *resource* dari perangkat mobile yang digunakan, dalam Tabel 3.6 yang menunjukkan perbandingan performa dari masing-masing model. Tes evaluasi deteksi *runtime* menggunakan perangkat mobile akan dihitung dalam satu siklus pengambilan gambar dan *result* yang di hasilkan. Tes akan dilakukan dengan mengambil 10 gambar kemudian akan dihitung rata-rata runtime dan rata-rata akurasi.

Tabel 3.7 Evaluasi untuk Perangkat Mobile

|  |  |  |  |
| --- | --- | --- | --- |
| Perangkat Mobile | Model | Runtime | Akurasi (%) |
| Huawei Nova 3i | MobileNet | x | x |
| ResNet | x | x |
| [PNASNet-5 Mobile](https://arxiv.org/abs/1712.00559) | x | x |
| [NASNet-A Mobile](https://arxiv.org/abs/1707.07012) | x | x |
| [VGG](https://arxiv.org/abs/1707.07012) | x | x |
| Inception V4 | x | x |

#### Kebutuhan Sumber Daya Komputasi

Penelitian ini juga akan mengukur kebutuhan sumber daya komputasi perangkat mobile konsumsi daya yaitu CPU, memori dan konsumsi energi, seperti terlihat di Table 3.7 dengan mengeksekusi aplikasi beberapa input gambar secara bertahap, dengan diujikan untuk *benchmark* setiap arsitektur seperti yang dilakukan oleh (Salar et al., 2016) pada perangkat mobile dengan menggunakan aplikasi *Qual-comm Trepn Profiler*.

Tabel 3.8 Evaluasi Empiris Kebutuhan Komputasi Perangkat Mobile

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | CPU | | Memory (MB) | SD Card Storage  (MB) | Power Battery (mW) |
| Load  (%) | Frequency (MHz) |
| MobileNet | x | x | x | x | x |
| ResNet | x | x | x | x | x |
| [PNASNet-5 Mobile](https://arxiv.org/abs/1712.00559) | x | x | x | x | x |
| [NASNet-A Mobile](https://arxiv.org/abs/1707.07012) | x | x | x | x | x |
| [VGG](https://arxiv.org/abs/1707.07012) | x | x | x | x | x |
| Inception V4 | x | x | x | x | x |

# DAFTAR PUSTAKA

Agrios, G. N. (1988). *Plat Pathology* (Third; G. N. Agrios, Ed.). https://doi.org/https://doi.org/10.1016/C2012-0-01423-8

Brahimi, M., Arsenovic, M., Laraba, S., Sladojevic, S., Boukhalfa, K., & Moussaoui, A. (2018). Deep Learning for Plant Diseases: Detection and Saliency Map Visualisation. *Computational Economics*, 1–21. https://doi.org/10.1007/978-3-319-90403-0\_6

Brahimi, M., Boukhalfa, K., & Moussaoui, A. (2017). Deep Learning for Tomato Diseases: Classification and Symptoms Visualization. *Applied Artificial Intelligence*, *31*(4), 299–315. https://doi.org/10.1080/08839514.2017.1315516

Cao, Q., Balasubramanian, N., & Balasubramanian, A. (2017). *MobiRNN: Efficient Recurrent Neural Network Execution on Mobile GPU*. 1–6. https://doi.org/10.1145/3089801.3089804

Castro, W., Oblitas, J., Maicelo, J., & Avila-George, H. (2018). Evaluation of Expert Systems Techniques for Classifying Different Stages of Coffee Rust Infection in Hyperspectral Images. *International Journal of Computational Intelligence Systems*, *11*(1), 86. https://doi.org/10.2991/ijcis.11.1.8

Chemura, A., Mutanga, O., & Dube, T. (2016). Separability of coffee leaf rust infection levels with machine learning methods at Sentinel-2 MSI spectral resolutions. *Precision Agriculture*, *18*(5), 859–881. https://doi.org/10.1007/s11119-016-9495-0

Chemura, A., Mutanga, O., Sibanda, M., & Chidoko, P. (2018). Machine learning prediction of coffee rust severity on leaves using spectroradiometer data. *Tropical Plant Pathology*, *43*(2), 117–127. https://doi.org/10.1007/s40858-017-0187-8

Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, *145*(September 2017), 311–318. https://doi.org/10.1016/j.compag.2018.01.009

Garcia, J., & Barbedo, A. (2018). Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Computers and Electronics in Agriculture*, *153*(August), 46–53. https://doi.org/10.1016/j.compag.2018.08.013

Goodfellow, I., Bengio, Y., & Courville, A. (2013). *Deep Learning*. Retrieved from www.deeplearningbook.org

Gould, S., Fulton, R., & Koller, D. (2009). Decomposing a Scene into Geometric and Semantically Consistent Regions. *International Conference on Computer Vision (ICCV)*, (Iccv). https://doi.org/10.1109/ICCV.2009.5459211

Ham, M., Moon, J. J., Lim, G., Song, W., Jung, J., Ahn, H., … Kim, H.-S. (2019). *NNStreamer: Stream Processing Paradigm for Neural Networks, Toward Efficient Development and Execution of On-Device AI Applications*. Retrieved from http://arxiv.org/abs/1901.04985

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9908 LNCS*, 630–645. https://doi.org/10.1007/978-3-319-46493-0\_38

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., … Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. https://doi.org/10.1016/S1507-1367(10)60022-3

Hughes, D. P., & Salathe, M. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics. *ArXiv*. https://doi.org/10.1111/1755-0998.12237

Huynh, L. N., Lee, Y., & Balan, R. K. (2017). DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications. *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys ’17*, 82–95. https://doi.org/10.1145/3081333.3081360

Karpathy, A. (2018). CS231n Convolutional Neural Networks for Visual Recognition. Retrieved from http://cs231n.github.io/convolutional-networks/

KOMINFO. (2017). Rayakan Hari Kopi, Kemenperin Terus Tingkatkan Ekspor Kopi Nasional. Retrieved December 15, 2018, from https://kominfo.go.id/content/detail/10775/rayakan-hari-kopi-kemenperin-terus-tingkatkan-ekspor-kopi-nasional/0/artikel\_gpr

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks*. *12*, 04015009. https://doi.org/10.1061/(ASCE)GT.1943-5606.0001284

LeCun, Y. (1989). Generalization and network design strategies. *Univ. of Toronto*, 143–155. https://doi.org/10.1242/jeb.02092

LeCun, Y., & Bengio, Y. (1995). *Convolutional Networks for Images, Speech, and Time-Series*. (November). Retrieved from internal-pdf://189.226.74.163/Convolutional Networks for Images, Speech, and.pdf

Lecun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning*. *521*. https://doi.org/10.1038/nature14539

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L. J., … Murphy, K. (2018). Progressive Neural Architecture Search. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11205 LNCS*, 19–35. https://doi.org/10.1007/978-3-030-01246-5\_2

Loussaief, S., & Abdelkrim, A. (2018). Convolutional Neural Network Hyper-Parameters Optimization based on Genetic Algorithms. *International Journal of Advanced Computer Science and Applications*, *9*(10), 252–266. https://doi.org/10.14569/ijacsa.2018.091031

Mahfud, & Cholil, M. (2012). *Teknologi dan Strategi Pengendalian Penyakit Karat Daun Untuk Meningkatkan produksi Kopi Nasional*. *5*(1), 44–57.

Mahfud, M., Siti, N., Ismiyati, & Ardiansyah. (2010). Kajian penerapan teknologi produksi pada usahatani kopi robusta di lokasi prima tani kabupaten pasuruan. *Jurnal Pengkajian Dan Pengembangan Teknologi Pertanian*, *13*(2), 141–147.

Mengistu, A. D., Alemayehu, D. M., & Mengistu, S. G. (2016). Ethiopian Coffee Plant Diseases Recognition Based on Imaging and Machine Learning Techniques. *International Journal of Database Theory and Application*, *9*(4), 79–88. https://doi.org/10.14257/ijdta.2016.9.4.07

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). *Using Deep Learning for Image-Based Plant Disease Detection*. *7*(September), 1–10. https://doi.org/10.3389/fpls.2016.01419

Salar, S., Oskouei, L., Golestani, H., & Hashemi, M. (2016). CNNdroid : GPU-Accelerated Execution of Trained Deep Convolutional Neural Networks on Android Comparing Mobile and Desktop GPUs. *Proceedings of the 24th ACM International Conference on Multimedia*, 1201–1205. https://doi.org/10.1145/2964284.2973801

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Network For Large-Scale Image Recognation. *ICLR 2015: Very Deep Convolutional Network For Large-Scale Image Recognation*. https://doi.org/10.2146/ajhp170251

Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification. *Computational Intelligence and Neuroscience*, *2016*, 11. https://doi.org/10.1155/2016/3289801

Sugiarti, L. (2017). *Analisis Tingkat Keparahan Penyakit Karat Winaya Mukti Tanjungsari*. *1*(2), 80–89.

Suhartono, D., Aditya, W., Lestari, M., & Yasin, M. (2013). Expert System in Detecting Coffee Plant Diseases. *International Journal of Electrical Energy*, *1*(3), 156–162. https://doi.org/10.12720/ijoee.1.3.156-162

Szegedy, C., Ioffe, S., Vanchouke, V., & Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *Journal of Urology*. https://doi.org/10.1016/S0022-5347(17)50340-7

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Pierre, S., Reed, S., … Andrew, R. (2015). Going Deeper with Convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9. https://doi.org/10.1002/jctb.4820

Tan, M., Chen, B., Pang, R., Vasudevan, V., & Le, Q. V. (2018). *MnasNet: Platform-Aware Neural Architecture Search for Mobile*. Retrieved from http://arxiv.org/abs/1807.11626

Vanetti, M. (2007). Confusion Matrix Online Calculator. Retrieved April 27, 2019, from Confusion Matrix Online Calculator website: http://www.marcovanetti.com/pages/cfmatrix/?noc=8%0A

Yanai, K., Tanno, R., & Okamoto, K. (2016). *Efficient Mobile Implementation of A CNN-based Object Recognition System*. 362–366. https://doi.org/10.1145/2964284.2967243

Zhang, X., Qiao, Y., Meng, F., Fan, C., & Zhang, M. (2018). Identification of maize leaf diseases using improved deep convolutional neural networks. *IEEE Access*, *6*, 30370–30377. https://doi.org/10.1109/ACCESS.2018.2844405

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2017). *Learning Transferable Architectures for Scalable Image Recognition*. https://doi.org/10.1044/1092-4388(2002/060