BAB 2
LANDASAN TEORI

2.1 Manga

Manga merupakan jenis komik atau novel grafis yang berasal dari jepang ataupun menggunakan bahasa jepang. Sejak tahun 1950 manga telah menjadi bagian utama dari industri penerbitan jepang. Pada tahun 1959, penerbit ternama kepang *Kondansha* menerbitkan majalah manga pertama bernama *Shonen Magazine,* yang menjadi awal naiknya kepopuleran komik jepang saat itu. Manga mempunyai beragam macam pembaca dengan berbagai rentang usia dikarenakan mencakup banyak genre yang diantaranya: aksi, komedi, romansa, drama, sejarah, horor, misteri, fiksi ilmiah dan lain-lain. [9]

2.2 Computer Vision

Computer Vision merupakan bidang yang berkembang pesat dikhususkan untuk menganalisa, memodifikasi, dan pemahaman gambar tingkat tinggi. Tujuannya adalah untuk menentukan apa yang terjadi di depan kamera dan menggunakan pemahaman tersebut untuk mengendalikan komputer, sistem robot, atau untuk memberikan citra baru yang lebih informatif daripada gambar kamera asli. Aplikasi area untuk komputer visi teknologi termasuk video surveillance, biometrik, otomotif, fotografi, produksi film, pencarian web, obat-obatan, game augmented reality, antarmuka pengguna yang baru, dan banyak lagi. [10]

2.3 Citra Digital

Secara umum, pengolahan citra digital menunjuk pada pemrosesan gambar 2 dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data 2 dimensi. Citra digital merupakan sebuah lark (array) yang berisi nilai-nilai real maupun komplek yang direpresentasikan dengan deretan bit tertentu.

Suatu citra dapat didefinisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x, y, dan nilai amplitudo f secara keseluruhan berhingga (finite) dan bernilai diskrit maka dapat dikata-kan bahwa citra tersebut adalah citra digital. [11]

2.3.1 Citra RGB

Citra RGB disebut juga citra true color. Citra RGB merupakan citra digital yang mengandung matriks data berukuran M × N × 3 yang merepresentasikan warna merah, hijau, dan biru untuk setiap pikselnya. Setiap warna dasar diberi rentang nilai tersendiri. Nilai rentang paling kecil yaitu 0 dan paling besar yaitu 255. Warna dari tiap piksel ditentukan oleh kombinasi dari intensitas merah, hijau, dan biru. [11]

2.3.2 Citra Grayscale

Citra grayscale merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pixelnya, dengan kata lain nilai bagian RED = GREEN = BLUE. Nilai tersebut digunakan untuk menunjukkan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan, dan putih. Tingkatan keabuan di sini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih. Citra grayscale berikut memiliki kedalaman warna 8 bit (256 kombinasi warna keabuan). [11] Rumus untuk mengubah citra RGB menjadi citra *grayscale* adalah sebagai berikut [12] [13].

$$Grayscale_{(x,y)} = (0.2989 * R) + (0.5870 * G) + (0.1141 * B) \qquad (2.1)$$

R = Nilai *Red*.

G = Nilai *Greeen.*

B = Nilai *Blue.*

2.3.3 Citra Binary

Citra biner adalah citra digital yang hanya memiliki dua kemungkinan nilai pixel yaitu hitam dan putih. Citra biner juga disebut sebagai citra B&W (black and white) atau citra monokrom. Hanya dibutuhkan 1 bit untuk mewakili nilai setiap pixel dari citra biner. Citra biner sering kali muncul sebagai hasil dari proses pengolahan seperti segmentasi, pengambangan, morfologi, ataupun dithering [11]. Rumus untuk mengubah citra RGB menjadi citra *grayscale* adalah sebagai berikut.

$$g(x,y) = \begin{cases} 1, jika\ f(x,y) \geq T \\ 0, jika\ f(x,y) < T \end{cases} \quad (2.2)$$

2.4 Pengolahan Citra Digital

Pengolahan citra digital secara umum merujuk pada pemrosesan komputerisasi pada citra dua dimensi. Bahkan disebutkan Anil K Jain, dalam pengertian yang luas pengolahan citra digital adalah suatu implementasi dari proses digital pada data dua dimensi.

Proses pengolahan citra digital meliputi akuisisi citra, praproses, segmentasi, representasi dan deskripsi, pengenalan dan interpretasi. [9]

2.4.1 Akuisisi Citra

Proses akuisisi citra adalah pemetaan suatu pandangan (scene) menjadi kontinu dengan menggunakan sensor. Misalnya kita dapat membuat sebuah citra digital dengan kamera atau *scanner*.

2.4.2 Peningkatan Kualitas Citra

Contoh praproses: meningkatkan kualitas kecerahan dan kontras, penajaman citra, menghilangkan derau (noise), menghilangkan gangguan geometrik/ radiometrik.

2.4.3 Segmentasi Citra

Melakukan partisi citra menjadi region of interest (R01) atau wilayah-wilayah obyek (internal properties), menentukan garis batas wilayah objek (external shope characteristics).

2.4.4 Representasi dan Deskripsi

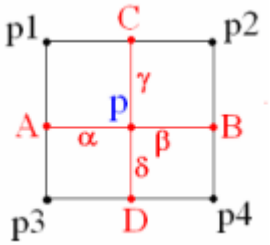
Suatu wilayah dapat direpresentasi sebagai suatu list titik-titik koordinat dalam loop yang tertutup, dengan deskripsi perimeternya

2.4.5 Pengenalan dan Interpretasi

Pengenalan dengan Labeling yaitu memberi label kategori objek pada setiap piksel citra berdasarkan informasi yang diberikan oleh descriptor. interpretasi: Memberikan arti pada objek yang sudah berhasil dikenali.

2.5 Interpolasi Bilinear

Interpolasi bilinear merupakan metode interpolasi yang menggunakan empat tetangga terdekat dari sebuah titik (nilai keabuan sebuah piksel) kemudian merubahnya menjadi nilai keabuan baru [14]. Inti algoritma bilinear adalah menganggap warna diantara dua *pixel* yang diketahui adalah linier (berubah secara beraturan). Pada model ini nilai satu *pixel* ditentukan oleh empat buah pixel yang mengelilinginya, untuk lebih jelas nya dapat dilihat pada Gambar 2.1 [15]



Gambar 2.1 Interpolasi Bilinier

Titik p1, p2, p3, dan p4 pada Gambar 2.1 adalah pixel yang diketahui nilainya sedangkan p adalah pixel yang ingin dicari nilainya dengan pendekatan interpolasi

bilinear. Diambil titik A, B, C, dan D yang merupakan proyeksi titik p ke garis p1-p3, p2-p4, p1-p2, dan p3-p4. Didefinisikan α, β, γ, dan δ sebagai berikut.

$$\alpha = \frac{|AP|}{|AB|}, \beta = \frac{|BP|}{|AB|}, \gamma = \frac{|CP|}{|CD|}, \delta = \frac{|DP|}{|CD|} \quad (2.3)$$

|AB| merupakan panjang AB, hal yang sama juga berlaku terhadap |AP|, dan lain-lain. Jarak antara dua buah *pixel* terdekat dianggap satu satuan. Nilai dari titik p akan ditentukan dari rumus berikut 2.4, dengan nilai(x) berarti nilai pada *pixel* x.

$$nilai(x) = \beta * \delta * nilai(p1) + \alpha * \delta * nilai(p2) + \beta * \gamma * nilai(p3) + \alpha * \gamma * nilai(p4) \quad (2.4)$$

2.6 Augmentasi Data

Augmentasi data merupakan serangkaian teknik meningkatkan varian citra memperbanyak jumlah dataset pelatihan untuk membangun model pembelajaran yang lebih baik berdasarkan dataset awal [16]. Pada transformasi geometrik metode augmentasi yang dilakukan berkaitan dengan posisi dari piksel-piksel di dalam citra, seperti rotasi, refleksi, dilatasi, translasi, shear, dan transformasi affine. Pada penelitian ini akan digunakan rotasi dan refleksi [17].

a. Rotasi merupakan salah satu transformasi geometri, dimana merupakan suatu proses perubahan posisi dari nilai-nilai piksel yang didasarkan pada nilai variabel rotasi sebesar θ terhadap sudut dan posisi titik pusat rotasi.

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} cos\theta^\circ & -sin\theta^\circ \\ sin\theta^\circ & cos\theta^\circ \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad (2.5)$$

b. Refleksi merupakan proses perubahan posisi dari nilai-nilai piksel pada koordinat awal (x1, y1) menuju koordinat akhir (x2, y2) pada citra sesuai dengan posisi pencerminan, dimana terdapat tiga jenis posisi pencerminan yaitu pencerminan

terhadap sumbu x, pencerminan terhadap sumbu y, dan pencerminan terhadap sumbu x dan sumbu y.

1. Refleksi sumbu x

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad (2.6)$$

2. Refleksi sumbu y

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad (2.7)$$

2.7 Run Length Smooth Algorithm

Run Length Smoothing Algorithm (RLSA) adalah metode yang dapat digunakan untuk segmentasi blok dan diskriminasi teks. Metode yang dikembangkan untuk sistem analisis dokumen ini terdiri dari dua langkah. Pertama, prosedur segmentasi membagi area dokumen menjadi beberapa wilayah (blok), yang masing-masing hanya dapat berisi satu jenis data (teks, grafik, gambar halftone, dll.). Kemudian beberapa karakteristik dasar dari blok-blok ini dihitung. RLSA dasar diterapkan pada urutan biner dimana piksel putih direpresentasikan sebagai 0s dan piksel hitam sebagai 1s. Algoritma mengubah string biner x menjadi string keluaran y sesuai dengan aturan berikut:

1. Angka nol dari x diganti dengan 1 dari y jika jumlah nol yang berdekatan kurang dari atau sama dengan batas C yang telah ditentukan.
2. Angka 1 dalam x tidak berubah dengan y.

Misalnya, untuk C = 4, barisan x dipetakan ke y sebagai berikut:

x: 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0

y: 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1

Saat diterapkan ke array sampel, RLSA memiliki efek menghubungkan wilayah hitam tetangga yang berjarak x kurang dari C piksel. Dengan pilihan C yang tepat, region terikat akan menjadi region dari tipe data generik. RLSA diterapkan baris demi

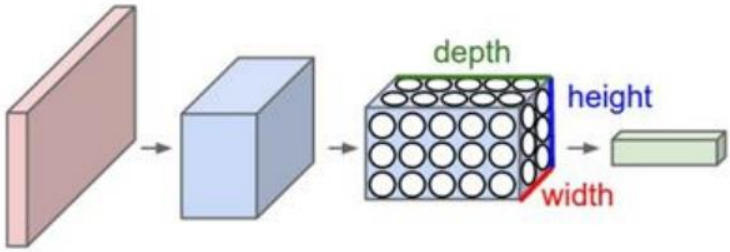
baris serta kolom demi kolom ke dokumen, membuat dua bitmap yang terpisah. Karena spasi elemen dokumen cenderung berbeda secara horizontal dan vertikal, nilai C yang berbeda digunakan untuk memproses baris dan kolom. Kedua bitmap tersebut kemudian digabungkan dalam operasi logika AND. Pemulusan horizontal tambahan menggunakan RLSA menghasilkan hasil akhir tersegmentasi. [18]

2.8 Convolution Neural Network

Convolutional Neural Network, disingkat CNN adalah salah satu kelas *deep feedforward artificial neural networks* yang banyak diaplikasikan pada analisis citra. CNN terinspirasi oleh proses-proses biologi dimana pola konektivitas antar *neurons* menyerupai organisasi *visual cortex* pada binatang. *Cortical neurons* menanggapi stimulus hanya dalam suatu area terbatas pada bidang visual atau bidang reseptif (*receptive field*). [6]

CNN terdiri atas satu lapisan masukan (*input layer*), suatu lapis keluaran dan sejumlah lapis tersembunyi umumnya berisi *convolutional layers, pooling layers, normalization layers, ReLU layer, fully connected layers dan loss layers.* Semua lapisan tersebut disusun secara bertumpuk-tumpuk, seperti sepotong *sandwich* yang terdiri atas roti bagian bawah, sayuran, daging, keju, saus tomat, mayonaise, saus sambal, dan roti bagian atas.

Berbeda dengan MLP, yang arsitekturnya disusun secara dua dimensi, CNN menggunakan arsitektur tiga dimensi: lebar (*width*), tinggi (*height*) dan dalam (*depth*). Setiap lapisan CNN mentransformasikan volume masukan tiga dimensi (3D input *volume*) ke dalam volume keluaran tiga dimensi aktivasi-aktivasi sel saraf (3D output volume of neuron activations). Berikut adalah Gambar 2.1 yang mengilustrasikan dimensi CNN:

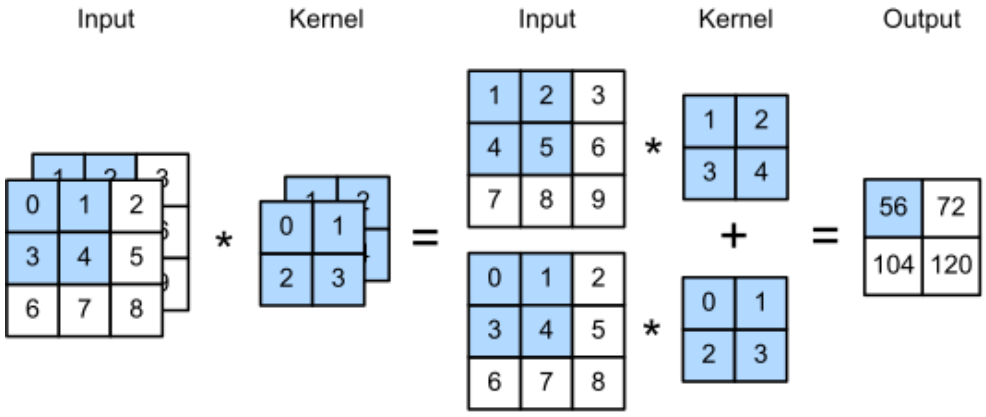


Gambar 2.2 Dimensi CNN

2.8.1 Convolutional Layer

Convolution layer inti dari CNN yang melakukan operasi konvolusi, yang berarti menerapkan beberapa filter ke gambar untuk mengekstrak fitur. *Filter* ini disebut kernel. Filter ini diterapkan pada setiap sub-bagian gambar, yang selanjutnya ditentukan oleh lapisan parameter konektivitas lokal. Setiap aplikasi filter menghasilkan nilai skalar untuk lokasi piksel tertentu, yang bila digabungkan di semua lokasi piksel, sering disebut sebagai peta fitur. Peta fitur akan dihitung sesuai dengan filter konvolusi tertentu.

Convolutional layer dalam arsitektur CNN umumnya menggunakan lebih dari satu *filter*. Jika kita menggunakan empat *filters*, maka *convolutional layer* akan berisi sejumlah *neurons* yang tersusun dalam kisi berukuran 28x28x4. Dengan demikian, akan ada empat *neurons* yang melihat area yang sama pada (citra) masukkan tersebut. [6]



Gambar 2.3 Proses Convolution Layer.

CNN umumnya menggunakan lebar langkah atau *stride* = 1 dengan *zero padding* sebesar

$$P = \frac{(F-1)}{2} \tag{2.8}$$

F = Ukuran filter.

Sementara untuk menghitung ukuran matriks keluaran dari proses konvolusi ini, dapat digunakan rumus sebagai berikut. [19] [17]

1. *Tanpa zero-padding*

$$H = \frac{I-K}{S} + 1 \tag{2.9}$$

2. *Half padding*

$$H = \frac{I-K+P}{S} + 1$$

(2.10)

3. *Full Padding*

$$H = \frac{I - K + 2P}{S} + 1$$

(**2.11)**

H = ukuran *feature maps* keluaran

I = ukuran citra masukan

K = ukuran kernel atau filter

P = jumlah padding

S = jumlah stride

2.8.2 Pooling Layer

Pooling Layer berfungsi menjaga ukuran data ketika *convolution*, yaitu dengan melakukan *downsampling* (pereduksian sampel). Dengan *pooling*, kita dapat merepresentasikan data menjadi lebih kecil, mudah dikelola, dan mudah mengontrol *overfitting*. [6]

Teknik *pooling* umumnya dibagi dua jenis yaitu *average pooling dan max pooling*. Average pooling mengambil nilai rata-rata di setiap grid, sementara Max Pooling mengambil nilai maksimal.

Untuk menghitung ukuran keluaran dari pooling layer, digunakan persamaan sebagai berikut. [19] [17]

$$u_K = \frac{w - f}{s} + 1$$

(2.12)

u_K = ukuran keluaran *pooling layer*

w = ukuran citra masukan

f = ukuran kernel atau filter

s = jumlah stride

2.8.3 Normalization Layer

Lapis normalisasi pada awalnya didesain untuk mengatasi adanya perbedaan rentang nilai yang signifikan pada citra masukan. Para ahli telah mengusulkan banyak jenis lapis normalisasi. Namun, lapis ini tidak banyak digunakan secara praktis karena dampaknya yang relatif kecil, atau bahkan tidak ada sama sekali. [6]

Perhitungan *batch normalization* dapat menggunakan rumus berikut.

$$BN = (\gamma \frac{I - \mu}{\sqrt{\sigma 2 + e}}) + \beta$$

(2.13)

I = nilai pada matriks input

μ = nilai rata-rata

γ = parameter pembelajaran

β = parameter pembelajaran

σ^2 = nilai varian

e = nilai konstan stabilitas numerik

2.8.4 ReLU Layer

Rectified Linear Units (*ReLu*) ini mengaplikasikan fungsi aktivasi f(x) = max(0,x). Ini meningkatkan sifat nonlinearitas fungsi keputusan dan jaringan secara keseluruhan tanpa mempengaruhi bidang-bidang reseptif pada *convolutional layer*. Bisa juga menggunakan fungsi lain untuk meningkatkan nonlinearitas, seperti tangen hiperbolik $f(x) = \tan(x)$, $f(x) = |\tan(x)|$ atau $f(x) = (1+e^{-x})^{-1}$.

2.8.5 Swish Layer

Fungsi aktivasi swish merupakan salah satu gabungan fungsi aktivasi dengan kombinasi fungsi aktivasi sigmoid dan fungsi input, untuk memperoleh fungsi aktivasi hybrid. Fungsi aktivasi swish menggunakan formasi pembelajaran berdasarkan teknik pencarian otomatis untuk menghitung fungsi. Fungsi aktivasi swish digambarkan oleh persamaan 2.14.

$$f_{(x)} = x * sigmoid(x) = \frac{x}{1 + e^{x}} \quad \textbf{(2.14)}$$

2.8.6 Fully Connected Layer

Pada lapisan yang terhubung seara penuh, setiap *neurons* memiliki koneksi penuh ke semua aktivasi dalam lapisan sebelumnya. Hal ini sama persis dengan MLP, yaitu komputasi menggunakan suatu perkalian matriks yang diikuti dengan *bias offset.* [6]

Perhitungan jaringan syaraf tiruan ini dilakukan dengan persamaan sebagai berikut.

$$fc = \sum_{j=1} w_{i,j}^{T} x_i + b_j$$

(2.15)

b = nilai vektor bias

w = nilai matriks bobot

x = vektor input

2.8.7 Loss Layer

Loss Layer, yang merupakan lapisan terakhir dalam CNN, menentukan bagaimana pelatihan memberikan penalti atas penyimpangan antara hasil prediksi dan label. Terdapat sejumlah variasi *loss function*, diantaranya adalah *softmax loss* yang digunakan untuk memprediksi satu dari sejumlah kelas yang saling eksklusif; *sigmoid cross-entropy loss* yang digunakan untuk memprediksi sejumlah nilai probabilitas dalam interval [0, 1]; dan *Euclidean loss* yang digunakan untuk regresi nilai kontinu. [6]

2.8.8 Cross Entropy

Secara teori, cross entropy umumnya digunakan untuk mengukur besar simpangan antara dua distribusi. Dalam neural network, cross entropy digunakan untuk menghitung nilai loss function atau mengukur tingkat pembelajaran model dengan mengetahui seberapa jauh nilai hasil prediksi dengan nilai sesungguhnya. Persamaan (2.16) merupakan persamaan cross entropy yang digunakan untuk mengukur besar simpangan nilai hasil prediksi t dengan y nilai sesungguhnya. [17]

$$J = -\sum_{j=1} t_j \log y_j \quad (2.16)$$

2.9 Stochastic Gradient Descent with Momentum

Stochastic gradient descent merupakan salah satu algoritma yang umumdigunakan pada neural network untuk melakukan optimasi. Algoritma ini melakukan pembaruan pada parameter bobot (weight) dan bias dengan cara mengurangi weight nilai awal dengan nilai gradient yang diperoleh.

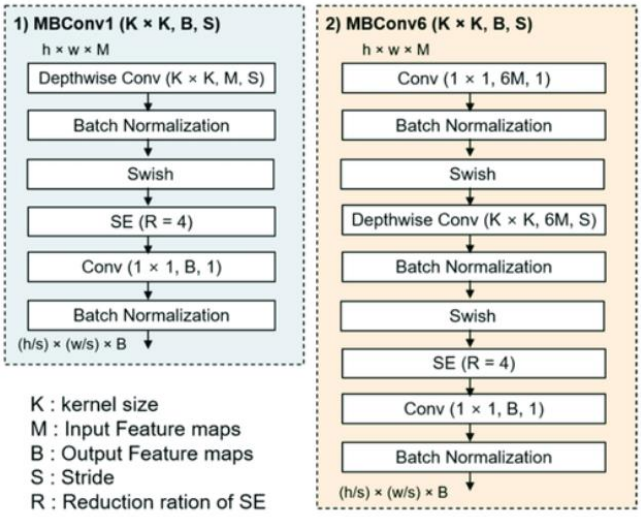
Dalam upaya memperoleh kinerja neural network yang optimal, diterapkan beberapa jenis pengembangan terhadap algoritma SGD, salah satunya adalah stochastic gradient descent with momentum. Algoritma tersebut mampu mencapai konvergensi dengan lebih cepat, sehingga sangat mungkin untuk menghindari lokal minimum yang buruk. [17]

2.10 MobileNet

MobileNets, merupakan salah satu arsitektur convolutional neural network (CNN) yang dibesut oleh peneliti ahili Google untuk mengatasi masalah kebutuhan komputasi dan digunakan secara mobile. Perbedaan mendasar antara arsitektur MobileNet dan arsitektur CNN pada umumnya adalah penggunaan lapisan atau layer konvolusi dengan ketebalan filter yang sesuai dengan ketebalan dari input image. MobileNet membagi konvolusi menjadi depthwise convolution. [20]

2.11 Inverted Residual Layer

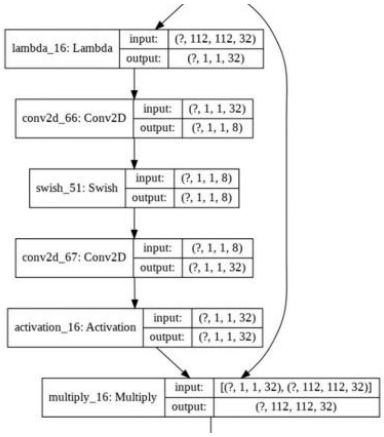
Inverted Residual Layer yang dipakai oleh EfficientNet secara umum adalah *module layer* yang digunakan MobileNetV2 yang terdiri dari beberapa layer. EfficientNet menggunakan dua jenis dari layer ini yaitu MBConv1 dan MBConv6. Berikut perbedaan antara *layer* MBConv1 dan MBConv6.



Gambar 2.4 Layer MBConv1 dan MBConv6.

Tahapan-tahapan MBConv1 adalah sebagai berikut: [21]

1. Pada tahap ini dilakukan operasi Depth Wise Convolutional dengan filter K x K dengan M channel.
2. Dilakukan proses batch normalization.
3. Kemudian dilakukan proses activation function menggunakan Swish.
4. Kemudian dilakukan Squeeze and Excitation yang mempunyai proses sebagai berikut.



Gambar 2.5 Layer Squeeze-and-Excitation

a. Pertama dilakukan Global Average Pooling
b. Selanjutnya lakukan konvolusi normal kernel 1x1 untuk melakukan reduksi channel berdasarkan *reduction rate* (R).
c. Dilakukan aktivasi Swish.
d. Lakukan konvolusi normal
e. Lakukan aktivasi Swish.
f. Lalu lakukan *multiply*.

5. Kemudian dilakukan batch normalization.

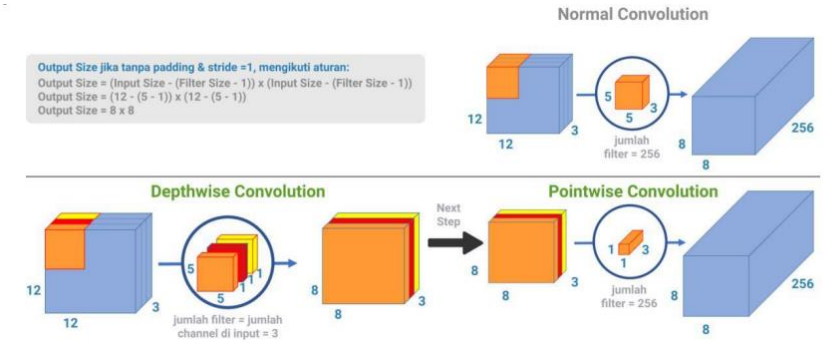
Tahapan-tahapan MBConv6 adalah sebagai berikut:

1. Pada tahap ini dilakukan operasi pemfilteran dengan menggunakan convolutional filter 1 x 1 dengan Mx6 channel.
2. Dilakukan proses batch normalization.
3. Kemudian dilakukan activation function menggunakan Swish.
4. Lalu dilakukan poses MBConv1.

2.12 Depthwise Convolution Layer

Model pada arsitektur MobileNet yang didasarkan pada konvolusi yang mendalam dapat dipisah-pisah (*depthwise separable convolution*) menjadi bentuk konvolusi yang menguraikan konvolusi standar (*standard convolution*) menjadi konvolusi mendalam (*depthwise convolution*) dan konvolusi 1x1 atau konvolusi

searah (*pointwise convolution*). *Depthwise convolution* memiliki penerapan satu filter pada setiap saluran masukan, kemudian *pointwise convolution* menggunakan konvolusi 1x1 untuk menggabungkan hasil keluaran dari *depthwise convolution*. Sedangkan standard convolution mem-filter dan melakukan penggabungan masukan pada sebuah set keluaran yang baru. *Deptwise separable convolution* membagi lapisan menjadi dua bagian berdasarkan fungsinya, yaitu lapisan untuk filter dan lapisan untuk penggabung. Pembagian lapisan ini dapat mengurangi komputasi dan ukuran model. [22]



Gambar 2.6 Proses *Normal Convolution* (atas) dan proses *Depthwise Separable Convolution* (bawah).

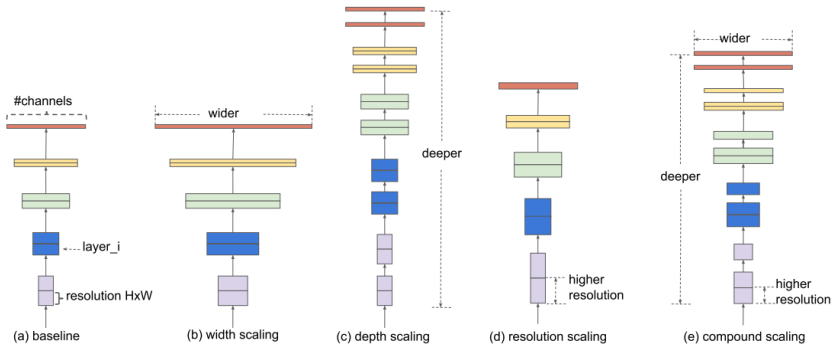
2.13 EfficientNet

EfficientNet merupakan arsitektur CNN yang diusulkan oleh Tan & Le [7]. Arsitektur ini mencoba menyelesaikan permasalahan dalam pelatihan data menggunakan metode CNN. Umumnya, pelatihan menggunakan metode CNN memakan cukup banyak waktu karena besarnya jumlah parameter pada arsitektur dan terbatas pada lapisan tertentu. Pada penelitian yang dilakukan oleh Tan & Le yang secara sistematis mempelajari penskalaan dan identifikasi model CNN menjelaskan bahwa pentingnya keseimbangan depth, width, dan resolution untuk menghasilkan kinerja model yang optimal. Dalam beberapa penelitian tentang arsitektur pada ImageNet sebelumnya, biasanya hanya melakukan penskalaan pada satu dari tiga dimensi depth, width dan resolution. Meskipun tetap memungkinan untuk melakukan

penskalaan dua atau tiga dimensi secara sembarangan, cara ini tentu membutuhkan penyetelan manual yang membosankan dan seringkali masih menghasilkan akurasi dan efisiensi yang kurang optimal. Oleh karena itu sangat penting memperhatikan keseimbangan semua dimensi depth, width, dan resolution.

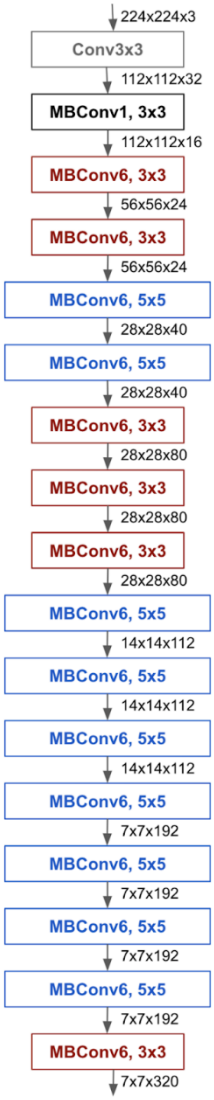
Keseimbangan tersebut dapat dicapai dengan hanya menskalakan masing-masing dimensi dengan rasio konstan. Tidak seperti praktik konvensional yang mengubah skala factor secara sembarangan, metode pada EfficientNet ini menskalakan depth, width, dan resolution dengan satu set koefisien penskalaan tetap. EfficientNet juga memiliki keunggulan dari segi kecilnya jumlah parameter serta akurasi yang tinggi.

Yang membedakan metode EfficientNet dengan arsitektur lainnya adalah bagaimana cara mereka melakukan *scaling* pada model. Terlihat pada Gambar 2.2, (a) adalah contoh jaringan awal. (b) - (d) adalah penskalaan yang biasa dilakukan arsitektur lain untuk meningkatkan akurasi. Sedangkan EfficientNet melakukan (e) yaitu *compound scaling*, metode penskalaan yang menggunakan ketiga metode sebelumnya di tiga dimensi secara bersamaan dengan rasio tetap.



Gambar 2.7 Macam-macam Model Scaling

Gambar 2.3 menunjukan layer model EfficientNet-B0.



Gambar 2.8 Arsitektur EfficientNet-B0

Faktor yang membedakan setiap arsitektur EfficientNet dapat dibedakan pada jumlah layer yang digunakan dari setiap tahapan. Berikut tahapan dari arsitektur EfficientNet-B0 dalam mengklasifikasi gambar [23] :

1. Stage 1 :

    a. Pada tahap ini dilakukan operasi pemfilteran dengan melakukan convolutional filter 3 x 3 dengan 32 channel.

b. Dilakukan proses Batch Normalization.

c. Kemudian dilakukan proses activation function menggunakan ReLU.

2. Stage 2:

a. Pada tahap ini dilakukan operasi Depth Wise Convolutional dengan filter 3 x 3 dengan 16 channel.

b. Dilakukan proses batch normalization

c. Kemudian dilakukan proses activation function menggunakan ReLU.

d. Kemudian dilakukan Global Average

e. Kemudian dilakukan reshape

f. Kemudian dilakukan 2 kali convolutional dengan filter 3 x 3 dengan channel 16 channel.

g. Kemudian dilakukan multiply.

h. Kemudian dilakukan batch normalization.

3. Stage 3:

a. Pada tahap ini dilakukan operasi pemfilteran dengan menggunakan convolutional filter 3 x 3 dengan 24 channel.

b. Dilakukan proses batch normalization.

c. Kemudian dilakukan activation function menggunakan Swish.

d. Kemudian dilakukan proses pada MBConv1 dengan filter 3x3.

4. Stage 4:

    a. Pada tahap ini dilakukan operasi pemfilteran dengan melakukan convolutional filter 5 x 5 dengan 40 channel.

    b. Dilakukan proses batch normalization.

    c. Kemudian dilakukan activation function menggunakan Swish.

    d. Kemudian dilakukan proses pada MBConv1 dengan filter 5 x 5.

5. Stage 5:

    a. Pada tahap ini dilakukan operasi pemfilteran dengan melakukan convolutional filter 3 x 3 dengan 80 channel.

    b. Dilakukan proses batch normalization.

    c. Kemudian dilakukan activation function menggunakan Swish.

    d. Kemudian dilakukan proses pada MBConv1 dengan filter 3 x 3.

6. Stage 6:

    a. Pada tahap ini dilakukan operasi pemfilteran dengan melakukan convolutional filter 5 x 5 dengan 112 channel.

    b. Dilakukan proses batch normalization.

    c. Kemudian dilakukan activation function menggunakan Swish.

    d. Kemudian dilakuakn activation convolutional filter 5 x 5.

7. Stage 7:

    a. Pada tahap ini dilakukan operasi pemfilteran dengan melakukan convolutional filter 5 x 5 dengan 192 channel.

b. Dilakukan proses batch normalization.

c. Kemudian dilakukan activation function menggunakan Swish.

d. Kemudian dilakuakn activation convolutional filter 5 x 5.

8. Stage 8:

a. Pada tahap ini dilakukan operasi pemfilteran dengan melakukan convolutional filter 3 x 3 dengan 320 channel.

b. Dilakukan proses batch normalization.

c. Kemudian dilakukan activation function menggunakan Swish.

d. Kemudian dilakukan proses pada MBConv1 dengan filter 3 x 3

9. Stage 9

a. Pada tahap ini dilakukan operasi pemfilteran dengan melakukan convolutional filter 1 x 1 menghasilkan 1280 channels.

b. Selanjutnya dilakukan Batch Normalization.

c. Kemudian dilakukan activation function menggunkana Swish.

10. Output dari proses sebelumnya akan dilakukan dengan menambahkan pooling seteleh itu di flatten yang menghasilkan output berupa array satu dimensi.

Berikut adalah perbedaan layer antara model B0 - B7:

1. EfficientNetB0

Tabel 2.1 Arsitektur EfficientNet-B0

Stage ***i***	**Operator** ***Fi***	**Resolusi** ***Hi x Wi***	**Channels** ***Ci***	**Layers** ***Li***
1	Conv 3x3	224 x 224	32	1
2	MBConv1, k3x3	112 x 112	16	1
3	MBConv6, k3x3	112 x 112	24	2
4	MBConv6, k5x5	56 x 56	40	2
5	MBConv6, k3x3	28 x 28	80	3
6	MBConv6, k5x5	14 x 14	112	3
7	MBConv6, k5x5	14 x 14	192	4
8	MBConv6, k3x3	7 x 7	320	1
9	Conv 1x1 & Pooling & FC	7 x 7	1280	1

2. EfficientNetB1

Tabel 2.2 Arsitektur EfficientNet-B1

Stage ***i***	**Operator** ***Fi***	**Resolusi** ***Hi x Wi***	**Channels** ***Ci***	**Layers** ***Li***
1	Conv 3x3	224 x 224	32	1

2	MBConv1, k3x3	112 x 112	16	2
3	MBConv6, k3x3	112 x 112	24	3
4	MBConv6, k5x5	56 x 56	40	3
5	MBConv6, k3x3	28 x 28	80	4
6	MBConv6, k5x5	14 x 14	112	4
7	MBConv6, k5x5	14 x 14	192	5
8	MBConv6, k3x3	7 x 7	320	2
9	Conv 1x1 & Pooling & FC	7 x 7	1280	1

3. EfficientNetB2

Tabel 2.3 Arsitektur EfficientNet-B2

Stage ***i***	**Operator** ***Fi***	**Resolusi** ***Hi x Wi***	**Channels** ***Ci***	**Layers** ***Li***
1	Conv 3x3	224 x 224	32	1
2	MBConv1, k3x3	112 x 112	16	2
3	MBConv6, k3x3	112 x 112	24	3
4	MBConv6, k5x5	56 x 56	48	3

5	MBConv6, k3x3	28 x 28	88	4
6	MBConv6, k5x5	14 x 14	120	4
7	MBConv6, k5x5	14 x 14	208	5
8	MBConv6, k3x3	7 x 7	352	2
9	Conv 1x1 & Pooling & FC	7 x 7	1408	1

4. EfficientNetB3

Tabel 2.4 Arsitektur EfficientNet-B3

Stage ***i***	**Operator** ***Fi***	**Resolusi** ***Hi x Wi***	**Channels** ***Ci***	**Layers** ***Li***
1	Conv 3x3	224 x 224	40	1
2	MBConv1, k3x3	112 x 112	24	2
3	MBConv6, k3x3	112 x 112	32	3
4	MBConv6, k5x5	56 x 56	48	3
5	MBConv6, k3x3	28 x 28	96	5
6	MBConv6, k5x5	14 x 14	136	5
7	MBConv6, k5x5	14 x 14	232	6

8	MBConv6, k3x3	7 x 7	384	2
9	Conv 1x1 & Pooling & FC	7 x 7	1536	1

5. EfficientNetB4

Tabel 2.5 Arsitektur EfficientNet-B4

Stage ***i***	**Operator** ***Fi***	**Resolusi** ***Hi x Wi***	**Channels** ***Ci***	**Layers** ***Li***
1	Conv 3x3	224 x 224	48	1
2	MBConv1, k3x3	112 x 112	24	2
3	MBConv6, k3x3	112 x 112	32	4
4	MBConv6, k5x5	56 x 56	56	4
5	MBConv6, k3x3	28 x 28	112	6
6	MBConv6, k5x5	14 x 14	160	6
7	MBConv6, k5x5	14 x 14	272	8
8	MBConv6, k3x3	7 x 7	448	2
9	Conv 1x1 & Pooling & FC	7 x 7	1792	1

6. EfficientNetB5

Tabel 2.6 Arsitektur EfficientNet-B5

Stage ***i***	**Operator** ***Fi***	**Resolusi** ***Hi x Wi***	**Channels** ***Ci***	**Layers** ***Li***
1	Conv 3x3	224 x 224	48	1
2	MBConv1, k3x3	112 x 112	24	3
3	MBConv6, k3x3	112 x 112	40	5
4	MBConv6, k5x5	56 x 56	64	5
5	MBConv6, k3x3	28 x 28	128	7
6	MBConv6, k5x5	14 x 14	176	7
7	MBConv6, k5x5	14 x 14	304	9
8	MBConv6, k3x3	7 x 7	512	3
9	Conv 1x1 & Pooling & FC	7 x 7	2048	1

7. EfficientNetB6

Tabel 2.7 Arsitektur EfficientNet-B6

Stage ***i***	**Operator** ***Fi***	**Resolusi** ***Hi x Wi***	**Channels** ***Ci***	**Layers** ***Li***

1	Conv 3x3	224 x 224	56	1
2	MBConv1, k3x3	112 x 112	32	3
3	MBConv6, k3x3	112 x 112	40	6
4	MBConv6, k5x5	56 x 56	72	6
5	MBConv6, k3x3	28 x 28	144	8
6	MBConv6, k5x5	14 x 14	200	8
7	MBConv6, k5x5	14 x 14	344	11
8	MBConv6, k3x3	7 x 7	576	3
9	Conv 1x1 & Pooling & FC	7 x 7	2304	1

8. EfficientNetB7

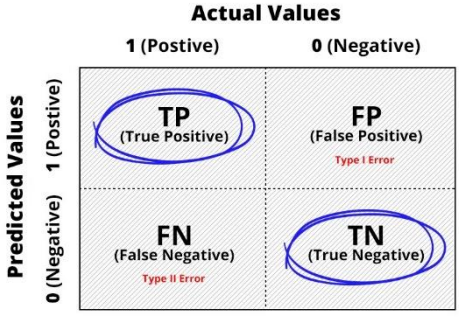
Tabel 2.8 Arsitektur EfficientNet-B7

Stage ***i***	**Operator** ***Fi***	**Resolusi** ***Hi x Wi***	**Channels** ***Ci***	**Layers** ***Li***
1	Conv 3x3	224 x 224	64	1
2	MBConv1, k3x3	112 x 112	32	4
3	MBConv6, k3x3	112 x 112	48	7

4	MBConv6, k5x5	56 x 56	80	7
5	MBConv6, k3x3	28 x 28	160	10
6	MBConv6, k5x5	14 x 14	224	10
7	MBConv6, k5x5	14 x 14	384	13
8	MBConv6, k3x3	7 x 7	640	4
9	Conv 1x1 & Pooling & FC	7 x 7	2506	1

2.14 Confusion Matrix

Confusion matrix atau error matrix digunakan untuk mengevaluasi kinerja dari algoritma biasanya digunakan pada supervised learning. Confusion matrix memberikan informasi perbandingan hasil klasifikasi yang telah dilakukan oleh sistem atau model dengan hasil sebenarnya. Confusion matrix berbentuk tabel matrik yang menampilkan hasil dari kinerja model dalam mengklasifikasikan data uji [24]. Gambar berikut ini merupakan contoh confusion matrix dengan empat kelas prediksi dan nilai aktual yang berbeda.



Gambar 2.9 Confusion Matrix

Pada gambar 2.23 hasil dari proses klasifikasi pada confusion matrix. Terdapat empat pada tabel confusion matrik yaitu True Positive (TP), True Negative (TN), False

Positive (FP) dan False Negative (FN). Berikut beberapa rumus yang diterapkan dalam mengevaluasi performa dari model:

2.14.1 Accuracy

Accuracy merupakan rasio antara prediksi benar (positif dan negatif) dengan keseluruhan data. Dengan kata lain, accuracy merupakan tingkat ketepatan nilai prediksi dengan nilai aktual (sebenarnya).

$$Akurasi = \frac{TP + TN}{P + N}$$

(2.16)

2.14.2 Precission

Precision merupakan rasio prediksi benar positif dengan keseluruhan hasil yang diprediksi positf. Dari semua kelas positif yang telah di prediksi dengan benar, berapa banyak data yang benar-benar positif. Nilai precision dapat diperoleh dengan persamaan.

$$Precission = \frac{TP}{TP + FP}$$

(2.17)

2.14.3 Recall

Recall merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Nilai recall dapat diperoleh dengan persamaan.

$$Recall = \frac{TP}{TP + FN}$$

(2.18)

2.15 Perangkat Lunak Pendukung

Berikut adalah beberapa perangkat lunak yang digunakan pada penelitian ini.

2.15.1 OpenCV

OpenCV adalah sebuah library (perpustakaan) yang digunakan untuk mengolah gambar dan video hingga kita mampu meng-ekstrak informasi didalamnya. OpenCV dapat berjalan di berbagai bahasa pemograman, seperti C, C++, Java, Python, dan juga support diberbagai platform seperti Windows, Linux, Mac OS, iOS dan Android. Salah satu contoh sederhana dalam penggunaan OpenCV adalah bagaimana kita dengan mudah bisa mendeteksi wajah dalam sebuah gambar. [25]

2.15.2 Tensorflow

TensorFlow adalah perpustakaan perangkat lunak, yang dikembangkan oleh Tim Google Brain dalam organisasi penelitian Mesin Cerdas Google, untuk tujuan melakukan pembelajaran mesin dan penelitian jaringan syaraf dalam. TensorFlow kemudian menggabungkan aljabar komputasi teknik pengoptimalan kompilasi, mempermudah penghitungan banyak ekspresi matematis dimana masalahnya adalah waktu yang dibutuhkan untuk melakukan perhitungan. [25]

2.15.3 Google Colab

Google Colab adalah alat yang diterbitkan oleh Google Internal Research. Google Colab adalah salah satu produk Google berbasis cloud yang dapat digunakan secara gratis. Google Colab dirancang khusus untuk peneliti dan pemrogram yang kesulitan mengakses komputer kelas atas. Google Colab adalah lingkungan pemrograman berukuran notebook untuk bahasa pemrograman Python. Alat Google Colab memberi pengguna layanan GPU gratis sebagai backend komputasi, masing-masing tersedia selama 12 jam. Google Colab berjalan di cloud Google, menyimpan

file ke Google Drive, serta memulai dengan tanda "!" untuk menjalankan baris perintah langsung di sel notebook. [26]

Manfaat menggunakan Google Colab

1. Free GPU

Google Colab memudahkan menjalankan program di komputer kelas atas (GPU Tesla, RAM 12 GB, HDD 300 GB, masih dapat dihubungkan ke Google Drive dan akses internet cepat sehingga dapat mengunduh file berukuran besar file) dan berjalan untuk waktu yang lama.

2. Kolaborasi

Memfasilitasi kolaborasi dengan pengguna lain dengan membagikan kode secara online. Lebih mudah untuk bereksperimen dan menggunakan fitur ini untuk mempelajari pengkodean orang lain.

3. Mudah berintegrasi

Dapat dengan mudah menghubungkan google colab ke notebook jupyter di komputer, terhubung ke google drive, terhubung ke github menggunakan runtime lokal.

4. Fleksibilitas

Google Colab hanya perlu dijalankan di browser, sehingga dapat dengan mudah menjalankan program deep learning dari ponsel. Selama ponsel cerdas terhubung ke Google Drive yang sama, dapat memantaunya dari browser ponsel cerdas .