BAB 2
LANDASAN TEORI

Terdapat beberapa teori-teori dan metode yang digunakan yaitu *Intelligent Transportation System* (ITS), *Principal Component Analysis* (PCA), augmentasi data, *oversampling* dan *undersampling*, *Convolutional Neural Network* (CNN), *pre-trained* model, EfficientNet, dan metrik performa evaluasi.

2.1 Intelligent Transportation System (ITS)

ITS adalah penerapan teknologi yang mengintegrasikan antar sistem informasi dan teknologi dengan infrastruktur transportasi dan kendaraan. ITS bertujuan untuk memfasilitasi sistem transportasi sistem transportasi yang menghubungkan lingkungan transportasi dengan kendaraan-kendaraan yang ada untuk melayani kepentingan masyarakat dengan memanfaatkan teknologi untuk memaksimalkan keselamatan, mobilitas, dan kinerja lingkungan [21]. Informasi-informasi yang dibutuhkan mencakup kendaraan, infrastruktur, dan pengemudi. Dengan adanya informasi tersebut, ITS dapat meningkatkan pengambilan keputusan secara langsung, kontrol dan manajemen lalu lintas.

Vehicle Make and Model Recognition (VMMR) merupakan salah satu kategori yang ada dalam ITS. VMMR bergantung terhadap informasi yaitu klasifikasi merek atau model dari suatu kendaraan. Metode ini dapat diimplementasikan untuk membantu polisi dalam mencari merek mobil, sistem automasi pembayaran toll, dan sistem dalam membantu pengemudi[7].

2.2 Principal Component Analysis (PCA)

Sebuah teknik yang digunakan untuk menyederhanakan dengan cara melakukan reduksi dimensi yang menggunakan beberapa garis yang biasa disebut *principle components*. Dengan adanya bantuan reduksi dimensi dapat memudahkan model untuk membaca atau menginterpretasikan data dan melihat data dalam beberapa *cluster*. PCA mengurangkan dimensi tetapi tidak menghilangkan informasi penting dari data tersebut. PCA dapat digunakan pada gambar untuk melakukan reduksi pada dimensi gambar. Dengan cara menghitung *eigenvectors* dan *eigenvalues* dari kovarians matriks dan mengubah data gambar dengan basis baru [22].

Sebagai contoh, saat melihat suatu gambar, warna, tekstur fitur, bentuk tepi dan bentuk pada suatu gambar terdapat banyak sekali fitur yang perlu diperhatikan. Jika semua fitur dilihat maka akan terdapat banyak sekali fitur-fitur yang mungkin tidak diperlukan. Diperlukan metode yang dapat menyimpan fitur penting dari suatu gambar dan mengganti fitur awal dengan kombinasi linier. PCA menggunakan dimensi reduksi dengan mempertahankan komponen dengan variansi dan informasi yang besar dan menghapus komponen dengan variansi dan informasi yang kecil. Tahapan-tahapan pada PCA sebagai berikut [23]:

1. *Image Normalization*
   PCA akan melakukan tahap *preprocessing* dengan melakukan *normalization* terhadap gambar. Proses reduksi gambar dengan menggunakan nilai rata-rata dapat meningkatkan *signal-to-noise-ration* (SNR) dan *discrimination power* (DP).

$$F_{normalized}(x,y) = \begin{bmatrix} f(0,0) & \dots & f(0,m-1) \\ \vdots & \ddots & \vdots \\ f(n-1,0) & \dots & f(n-1,m-1) \end{bmatrix} - \begin{bmatrix} f(0,0) & \dots & f(0,m-1) \end{bmatrix} \tag{2.1}$$

   dimana [f(0,0) ... f(0,m-1)] merupakan kolom vector yang berisikan nilai rata rata

2. *Covariance Matrix of Image Data*
   Kovarians matriks dihitung untuk mencari nilai varians tertinggi dari suatu gambar. Kovarians matriks didapatkan dengan rumus :

$$Cov(x,y) = \frac{F_{normalized}(x,y) + F_{normalized}(x,y)^T}{m-1} \tag{2.2}$$

   dimana m adalah jumlah dari elemen y.

3. *Eigenvectors* dan *Eigenvalues of the Covariance Matrix*
   Eigenvector dengan *eigenvalues* terbesar adalah variasi terbesar dari gambar tersebut. Dimana *eigenvector* matriks merepresentasikan *principal feature* dari gambar. *Eigenvectors* dan *Eigenvalues* dapat dihitung menggunakan rumus SVD.

$$AA^T = Cov(x,y) = UD^2U^T \tag{2.3}$$

dimana U adalah *eigenvector* dari AA^T dan D kuadrat adalah *eigenvalue* dari AA^T.

4. *Transforming Image Data into New Basis* Melakukan transformasi gambar dengan dimensi yang sudah di reduksi. Rumus dibawah digunakan untuk melakukan proyeksi gambar asli dengan *eigenvector*.

$$F_{transformed}(x,y) = U^T F_{normalized}(x,y) \tag{2.4}$$

dimana U^T adalah *transpose* dari *eigenvector* matriks dan $F_{normalized}(x,y)$ merupakan gambar awal yang dinormalisasi.

2.3 Augmentasi Data

Teknik yang digunakan untuk menambah jumlah dari *training dataset* dengan membuat gambar dengan versi yang baru. Gambar dengan versi baru didapatkan dari *training dataset* yang dibuat secara artificial. Augmentasi data dapat digunakan pada *deep learning* dalam melakukan klasifikasi gambar sehingga dapat meningkatkan *generalization* pada *model*.

Terdapat beberapa teknik yang bisa diimplementasikan pada suatu gambar yaitu [24]:

1. *Flipping* : Gambar dapat dibalik berdasarkan sumbu x dan sumbu y.
2. *Cropping* : Memotong gambar secara acak.
3. *Rotation* : Rotasi gambar dari 1 derajat hingga 359 derajat.
4. *Translation* : Menggeser gambar ke kiri, ke kanan, ke atas, ke bawah.
5. *Noise injection* : Memberikan nilai di matriks pada gambar berdasarkan *gaussian distribution*.
6. *Color space transformation* : Memanipulasi nilai pixel pada suatu gambar.
7. *Mixing images* : Menggabungkan gambar dengan gambar-gambar lain.

Augmentasi data dapat digunakan untuk mengatasi permasalahan dimana terdapat beberapa kelas yang memiliki jumlah yang jauh lebih tinggi dibanding kelas yang lain atau *imbalance dataset*. Permasalahan ini akan membuat model menjadi *bias* dimana hanya data mayoritas yang digeneralisir dan data minoritas

akan dianggap sebagai bagian dari data mayoritas. Penelitian mengatakan bahwa *imbalance dataset* dikategorikan berada pada ratio 1:4 sampai dengan 1:100 [25]. Untuk mengatasi ratio antara data mayoritas dan data minoritas salah satu metode yang dapat digunakan adalah dengan *oversampling*.

2.4 Oversampling dan Undersampling

Oversampling merupakan suatu metode dimana data minoritas diperbanyak sehingga jumlah data minoritas tersebut kurang lebih sama dengan data mayoritas [26]. Dengan metode ini memungkinkan terjadinya *overfit* karena terdapat duplikat data pada data minoritas yang dapat menyebabkan model mempelajari fitur spesifik dari data tersebut. Oleh karena itu, digunakan augmentasi data sehingga data tidak sama dengan data awalnya.

Undersampling merupakan suatu metode dimana data mayoritas dihapus sehingga memiliki jumlah yang kurang lebih sama dengan data minoritas. Dari hasil yang didapatkan menggunakan metode *oversampling* dan *undersampling* didapatkan hasil yang lebih baik ketika menggunakan metode *oversampling* pada berbagai *classifier* dan mendapatkan metrik performa evaluasi yang lebih tinggi [27].

2.5 Convolutional Neural Network (CNN)

CNN merupakan salah satu tipe dari *Deep Learning* yang merupakan pengembangan dari *Multi Layer Perceptron* (MLP). Terdapat perbedaan utama pada MLP dan CNN yaitu pada neuron direpresentasikan. MLP merepresentasikan neuron dengan 1 dimensi sedangkan CNN dengan 2 dimensi. CNN menggunakan neuron 2 dimensi karena dengan 2 dimensi CNN dapat melihat korelasi antar piksel. Dengan korelasi piksel tersebut CNN dapat melakukan klasifikasi gambar dengan melihat fitur-fitur pada suatu gambar.

Terdapat 4 layer utama pada CNN arsitektur [28] :

1. *Input Layer*
   Layer ini digunakan untuk menerima *dataset* gambar yang direpresentasikan dengan 3 dimensi matriks jika memiliki warna dasar yaitu merah, hijau, dan biru (RGB) dan 1 dimensi matriks jika memiliki warna *grayscale*.

2. *Convolution Layer*
   Layer ini akan digunakan untuk mempelajari karakteristik atau fitur penting

dari gambar yang disebut *feature map*. Dari setiap *feature map* tersebut akan dilakukan perkalian matriks antar tiap kombinasi dari *feature map* yang ada. Hasil dari perhitungan tersebut akan menjadi hasil konvolusi dari setiap nilai terbesar pada setiap *feature map*. Pada *convolution layer*, terdapat *kernels* atau *learnable filters* yang digunakan untuk mengetahui fitur penting dari gambar. Setiap *filter* memiliki ukuran lebar dan tinggi. Dari hasil yang didapatkan melalui konvolusi dilanjutkan dengan *non linear activation function* seperti *sigmoid, tanh, ReLU etc.*. Terdapat 3 *hyperparameter* yang mempengaruhi hasil dari *convolution layer* yaitu [29]:

(a) *Depth* : merupakan jumlah *filter* yang akan digunakan untuk melakukan operasi konvolusi. Setiap filter akan mempelajari fitur penting seperti *input*, sisi tepi, warna dan lain-lain.

(b) *Stride* : merupakan jumlah dari berapa banyak pixel yang akan digeser setiap *filter* berjalan. Ketika jumlah *stride* sebanyak n satuan maka filter akan bergeser n *pixel* setiap bergeser.

(c) *Padding* : digunakan untuk mengatur ukuran dari output. Menggunakan *convolution layer* dapat mengurangkan informasi yang didapatkan dari gambar. Untuk menghindari hal tersebut, digunakan *padding* pada pinggiran dari matriks gambar.

3. *Pooling Layer*

Gambar yang direpresentasikan dengan data masuk ke dalam *convolutional layer* akan meningkatkan jumlah parameter dan komputasi menjadi semakin kompleks. Dengan menggunakan *pooling layer* dapat mengurangi kompleksitas komputasi. Selain itu juga dapat mengurangi matriks dengan menggunakan *stride*. Terdapat 2 tipe dalam *pooling* yaitu *max pooling* dimana hasil terbesar dari *pixel* akan digunakan dan *average pooling* dimana hasil rata-rata dari *pixel* akan digunakan. *Max pooling* lebih sering digunakan daripada *average pooling* [29] karena *max pooling* mengambil nilai terbesar dari kumpulan *pixel* yang memungkinkan adanya fitur penting dari gambar. Terdapat 2 *hyperparameter* pada *pooling layer* yaitu [29]:

(a) *Filter size* : merupakan jumlah *filter* yang akan digunakan untuk melakukan operasi *pooling*.

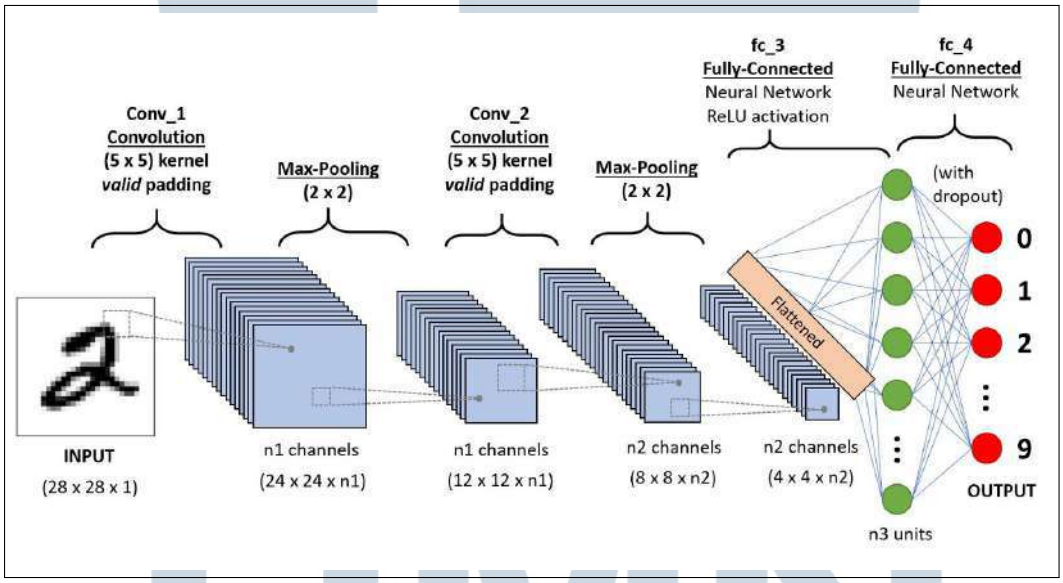
(b) *Stride* : merupakan jumlah dari berapa banyak pixel yang akan digeser setiap *filter* berjalan. Ketika jumlah *stride* sebanyak n satuan maka filter

akan bergeser n *pixel* setiap bergeser.

4. *Fully Connected Layer*

   Layer ini akan menghubungkan semua *neuron* dari 1 *layer* ke *layer* lainnya. Dapat juga digunakan untuk melakukan klasifikasi gambar sesuai dengan jumlah *class* yang telah ditentukan. Pada tahap ini, dapat menggunakan *dropout* atau teknik *regularization* pada *fully connected layer* untuk mencegah terjadinya *overfitting*[29]. Jumlah *layer* terakhir memiliki jumlah yang sama dengan banyaknya kelas yang ingin diklasifikasi.

Gambar CNN arsitektur [30] dapat dilihat pada gambar 2.1.



Gambar 2.1. CNN *sequence*

2.6 Pretrained Model

Pretrained model merupakan model yang sudah dilatih pada dataset yang besar. Secara general *pretrained* model dapat digunakan untuk melakukan klasifikasi sesuai dengan data yang sudah dilatih atau menggunakan *transfer learning* untuk melakukan klasifikasi pada kelas baru yang belum dilatih pada dataset awal [31]. Penggunaan *transfer learning* digunakan dengan menghilangkan *dense* layer pada *pretrained model*.

Transfer learning dapat digunakan untuk 2 cara yaitu *fine tuning* dan ekstraksi fitur.

1. *Fine tuning* digunakan dengan cara melakukan pelatihan ulang pada *pre-trained model* dengan memilih layer untuk dilakukan *unfreeze* layer. Dengan *fine tuning*, model memperbarui *weight* dari setiap layer berdasarkan *weight* awal yang telah dipelajari model dari dataset yang besar. Dari *weight* awal tersebut memungkinkan model sudah mengetahui beberapa fitur penting atau yang hanya menjadi latar belakang atau *noise* pada suatu gambar.

2. Ekstraksi fitur digunakan untuk melakukan ekstraksi fitur-fitur penting dari suatu gambar tanpa melatih model tersebut. Model tersebut dilakukan *freeze* layer sehingga *weight* tidak diperbarui. Pada layer terakhir setelah ekstraksi fitur dapat ditambahkan *classifier* seperti *pooling*, *flatten*, dan *dense*. Setelah itu, model akan dilatih kembali sesuai dengan *classifier* yang ditentukan sesuah dengan kebutuhan.

2.7 EfficientNet

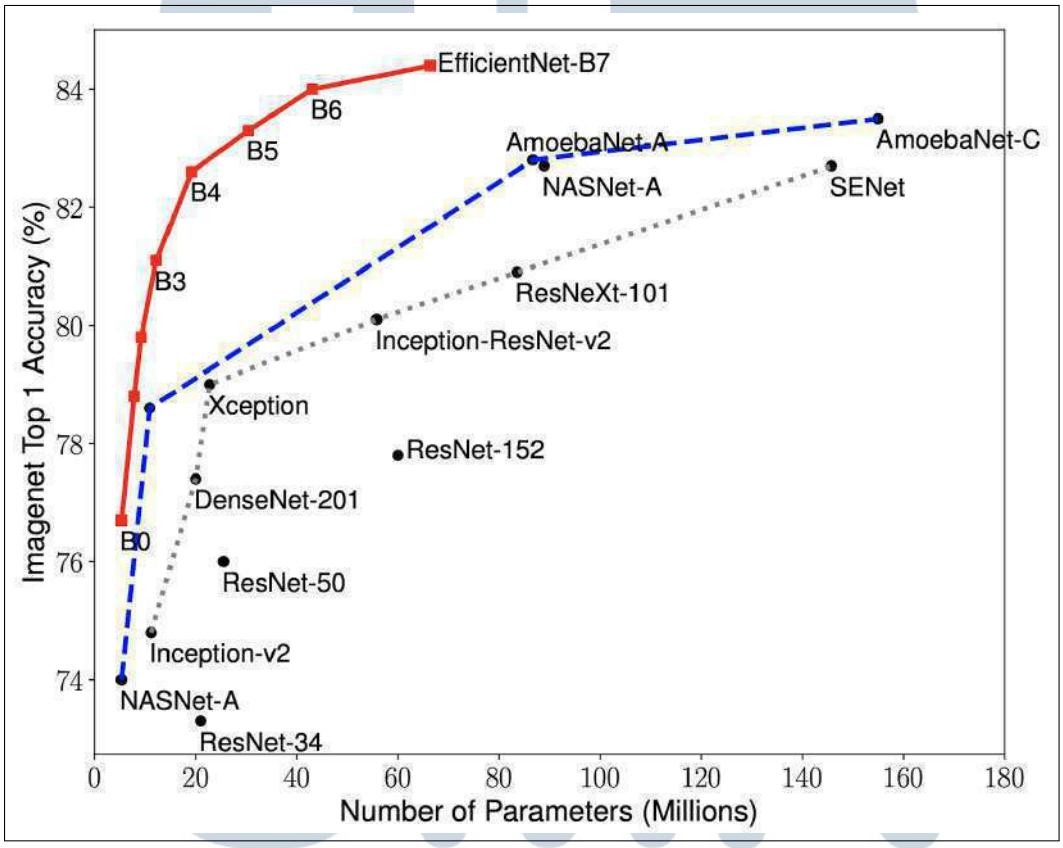
EfficientNet merupakan salah satu *pretrained* model CNN yang digunakan untuk melakukan klasifikasi gambar yang dikembangkan oleh peneliti pada Google AI. Terdapat beberapa model pada EfficientNet yaitu B0 sampai dengan B7. Beberapa model pada EfficientNet dengan resolusi dan jumlah parameter dapat dilihat pada tabel 2.1.

Tabel 2.1. Model resolusi dan parameter

Model	Resolusi	Parameter (juta)
EfficientNetB0	224	5.3
EfficientNetB1	240	7.8
EfficientNetB2	260	9.2
EfficientNetB3	300	12
EfficientNetB4	380	19
EfficientNetB5	456	30
EfficientNetB6	528	43
EfficientNetB7	600	66

Dalam meningkatkan akurasi pada model CNN, salah satu cara yang paling umum digunakan adalah dengan meningkatkan *depth* dengan cara menambahkan

layer atau *width* dengan cara memperbanyak *feature maps*. Selain itu terdapat juga cara lain yaitu meningkatkan resolusi pada gambar. EfficientNet sendiri menggunakan metode lain yaitu melakukan penskalaan pada *width*, *depth*, dan *resolution* dengan *compound coefficient* yang disebut *compound scaling method*. Dengan metode ini gambar dengan resolusi tinggi memerlukan penskalaan pada *width* dan *depth* untuk membantu dalam mencari fitur penting pada gambar dengan piksel yang lebih banyak [32]. Gambar grafik yang menunjukan perbandingan EfficientNet dengan *pretrained* model lainnya dapat dilihat pada gambar 2.2.



Gambar 2.2. Komparasi akurasi *pretrained* model

Arsitektur dalam EfficientNetB0 dibuat dengan komponen utama yaitu *mobile inverted bottleneck MBConv* dengan menambahkan *squeeze-and-excitation optimization*. EfficientNetB0 arsitektur dapat dilihat pada tabel 2.2.

Tabel 2.2. Arsitektur EfficientNetB0

Stage	Operator	Resolution	Channels	Layers
1	Conv3x3	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 224	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	Conv1x1 & Pooling & Fully-Connected	224 x 224	1280	1

2.8 Metrik Performa Evaluasi

Metrik performa berguna untuk membandingkan kualitas dari *model* yang telah dibuat dalam membuat suatu prediksi. Beberapa metrik yang digunakan yaitu akurasi, presisi, *recall*, dan skor F1. Untuk melakukan evaluasi diperlukan beberapa variabel yang dapat dilihat pada tabel *confusion matrix* dibawah ini. *Confusion matrix* merupakan suatu tabel yang menghitung munculnya nilai dari *actual classification* dan *predicted classification* [33].

Untuk menghitung nilai metrik performa evaluasi seperti akurasi, presisi, *recall*, dan skor F1 diperlukan beberapa istilah yang digunakan untuk menghitung nilai dari metrik tersebut yaitu *true positive*, *true negative*, *false positive*, dan *false negative* [34]

Tabel 2.3. Multiclass confusion matrix

		Predicted Values		
		A	B	C
Actual values	A	TP_A	E_{AB}	E_{AC}
	B	E_{BA}	TP_B	E_{BC}
	C	E_{CA}	E_{CB}	TP_C

1. *True positive* (TP)

   Hasil dari prediksi sama dengan nilai asilnya. Jika dilihat dari tabel 1 maka

kelas A memiliki nilai TP yaitu *predicted values* dan *actual values* pada kelas A. Nilainya adalah TP_A.

2. *True negative* (TN)
   Jumlah seluruh nilai dari baris dan kolom kecuali nilai dari kelas tersebut. Jika dilihat dari tabel 1 maka kelas A memiliki nilai TN yaitu jumlah seluruh cell terkecuali baris dan kolom kelas A. Nilainya adalah jumlah dari TP_B, E_{BC}, E_{CB}, TP_C.

3. *False positive* (FP)
   Jumlah dari seluruh nilai dari kolom selain TP. Jika dilihat dari tabel 1 maka kelas A memiliki nilai FP yaitu *predicted values* berada pada kelas A dan *actual values* pada kelas lain. Nilainya adalah jumlah dari E_{BA} dan E_{CA}.

4. *False negative* (FN)
   Jumlah dari seluruh nilai dari baris selain TP. jika dilihat dari tabel 1 maka kelas A memiliki nilai FN yaitu *actual values* berada pada kelas A dan *predicted values* pada kelas lain. Nilainya adalah jumlah dari E_{AB} dan E_{AC}.

Akurasi digunakan untuk melihat seberapa akurat *model* berdasarkan jumlah dari TP dan TN. Akurasi didapatkan dengan rumus :

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.5}$$

Presisi digunakan untuk melihat seberapa akurat *model* berdasarkan total dari FP dan TP. Presisi didapatkan dengan rumus :

$$Presisi = \frac{TP}{TP+FP} \tag{2.6}$$

Recall digunakan untuk melihat seberapa akurat *model* berdasarkan total dari TP dan FN. *Recall* didapatkan dengan rumus :

$$Recall = \frac{TP}{TP+FN} \tag{2.7}$$

Skor F1 menggabungkan nilai dari presisi dan *recall* dengan menggunakan konsep yaitu *harmonic mean* [33]. Skor F1 didapatkan dengan rumus :

$$SkorF1 = 2.\left(\frac{presisi\ .\ recall}{presisi+recall}\right) \tag{2.8}$$