BAB II

LANDASAN TEORI

2.1 Pengenalan Wajah

Teknologi pengenalan wajah (*Facial Recognition*) telah ditemukan sejak pertengahan abad 20 oleh Woody Bledsoe, yang merupakan seorang saintis komputer serta matematikawan, namun karena pembiayaan dilakukan oleh badan intelijen tidak bernama membuat hasil dari penelitian tidak begitu banyak dipublikasikan. Kini, telah banyak sekali penelitian dan studi terkait pengembangan teknologi pengenalan wajah dengan banyak jenis metode yang telah dilakukan. Metode pengembangan sistem pengenalan wajah yang berfokus pada intensitas gambar terdiri dari dua kategori pendekatan yaitu *featured-based* dan *holistic* (Liu, 2014).

Proses dari pengenalan wajah *featured-based* dimulai dengan melakukan suatu pengukuran dan ekstraksi dengan membagi wajah menjadi beberapa komponen seperti mata, mulut, hidung, dan lain-lain. Kemudian dilakukan suatu komputasi geometris relasi antara komponen wajah, sehingga terjadi suatu perubahan setiap data *raw image* menjadi suatu fitur berbentuk vektor (Manjunath, 2002).

Pengenalan wajah dengan pendekatan *holistic* merupakan pendekatan yang mencoba untuk mengidentifikasikan suatu wajah menggunakan representasi global, seperti melakukan pengamatan langsung terhadap seluruh gambar, berbeda dengan *feature based* yang fokusnya pada menelaah fitur satu gambar.

Pengenalan wajah *holistic* terbagi menjadi dua pendekatan yaitu, pendekatan statistik dan AI (Karamizadeh, 2013).

Pengenalan wajah atau *facial recognition* merupakan salah satu teknik otentikasi yang mampu untuk mengidentifikasi seseorang dari suatu citra digital atau baik dari suatu frame citra suatu video. Sistem pengenalan wajah memiliki 2 tugas utama yaitu verifikasi dan identifikasi. Verifikasi wajah berarti adanya kecocokan 1:1 atau adanya similaritas antara citra gambar input dibandingkan suatu kecocokan citra wajah yang terdapat dalam dataset. Identifikasi merupakan problem ber kompleksitas 1:N membandingkan sebuah citra wajah terhadap semua gambar yang terdapat dalam basis data sebanyak N (Kakade, 2016).

Secara garis besar, proses pengenalan wajah terbagi menjadi empat proses pengolahan data citra yaitu, input gambar, pra proses, ekstraksi ciri dan pembuatan model pembelajaran klasifikasi wajah. Salah satu studi terbaru dari pengenalan wajah adalah pengembangan ekstraksi ciri untuk mengimprovisasi hasil dari pengenalan wajah dengan menggunakan ekstraksi ciri menggunakan *Additive Angular Margin Loss* yang berguna untuk menambah ekstraksi penciri data dari model pengenalan wajah dan untuk menstabilkan proses *training* dataset (Deng J. , 2019).

Pra proses merupakan suatu tahap tiap gambar dinormalisasikan untuk mengimprovisasi hasil dari sistem pengenalan. Hasil dari implementasi tahap pra proses terhadap citra yaitu, data yang telah dinormalisasi ukurannya, *background* citra yang telah dibuang, normalisasi dimensi translasi dan rotasi, dan normalisasi penerangan (Jalled, 2017).

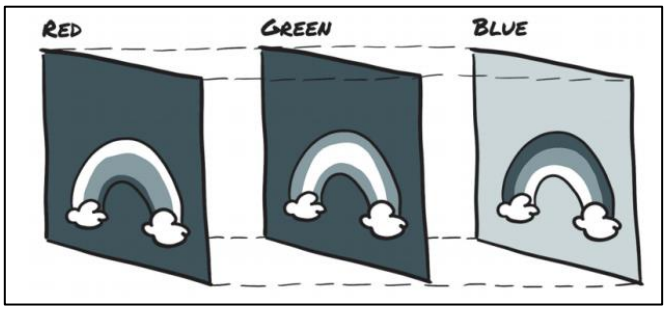
Ekstraksi ciri merupakan pengurangan reduksi suatu data dan bagian dari langkah pra-proses pembelajaran mesin untuk mengimprovisasi hasil yang komprehensif. Tujuan dari ekstraksi fitur ini adalah untuk menghapus data yang tidak relevan dan redundan (Khalid, 2014). Teknik ekstraksi ciri telah banyak yang telah diimplementasi dan populer digunakan seperti contoh, *Gabor Filters*, *Local Binary Patterns* (LBP), *Principal Component Analysis* (PCA), *Independent Component Analysis* (ICA), *Linear Discriminant Analysis* (LDA), *Local Gradient Code* (LGC), dan *Local Directional Pattern* (LDP) (Kumari, 2015).

2.2 Citra Digital

Citra digital merupakan suatu representasi diskrit yang tersusun dari informasi data tata letak dan intensitas warna. Citra terbentuk dari data diskrit yang berukuran dua dimensi sehingga memiliki batasan sejumlah spasial seperti baris dan kolom (*matrix)*. Suatu individu citra digital memiliki suatu *pixels* dari citra yang menjadi penyusun tiap koordinat data dua dimensi. (Solomon dan Breckon, 2011).

Sebuah citra tersusun dari suatu kumpulan data skalar sehingga memiliki tinggi dan lebar dalam *pixels* dan memiliki keunikan nilai tersendiri dibandingkan citra lain. Dalam citra digital *grayscale* data skalar yang menjadi komponen gambar tersusun dari satu skalar, sedangkan untuk citra digital yang banyak skalar mampu merepresentasikan berbagai warna atau fitur. Skalar merepresentasikan nilai warna individu pixel yang di-*encode* dalam data yang bertipe 8-bit *integer*. Terdapat beberapa cara dari meng-*encode* bilangan menjadi warna. Salah satu cara umum adalah dengan RGB, yang mendefinisikan suatu warna dengan tiga

bilangan yang merepresentasi intensitas dari komponen warna merah, hijau, dan biru. Masing-masing komponen warna tersebut untuk dapat menampilkan suatu citra dikombinasikan berdasarkan porsi spektrum tiap warna RGB. (Stevens dan Antiga, 2019).



Gambar 2. 1 Spektrum Pembentuk Citra
(Chollet, 2018)

Kemudian, untuk dapat melakukan pengolahan data agar proses pembentukan model *training* dapat cepat dan semakin akurat dibutuhkan beberapa pengolahan citra digital, seperti melakukan resize pada gambar, mengubah format gambar menjadi grayscale, menerapkan suatu filter, melakukan *crop* pada gambar dan lain-lain.

Grayscale merupakan suatu pengolahan citra digital yang tujuannya untuk mengurangi dimensi suatu citra yang awalnya tersusun dari tiga dimensi warna pembentuk citra (RGB) menjadi satu dimensi, tanpa mengurangi representasi kontras tiap *pixel* dalam citra digital. Proses pengubahan citra menjadi citra abu-abu ini dapat dihitung dengan suatu persamaan berikut:

$$Gray = W_R * \mathrm{R} + W_G * \mathrm{G} + W_B * \mathrm{B} \quad \ldots(1)$$

$$W_R + W_G + W_B = 1 \quad \ldots(2)$$

Batasan:

$$R, G, B <= 255$$

Keterangan:

$Gray$: Nilai representasi *pixel* grayscale

R: Nilai komponen warna merah

G: Nilai komponen warna hijau

B: Nilai komponen warna biru

W_R: Bobot komponen warna merah

W_G: Bobot komponen warna hijau

W_B: Bobot komponen warna biru

2.3 Ekstraksi Ciri

Ekstraksi ciri merupakan proses untuk melakukan reduksi pada data namun tidak mengurangi ciri unik dari tiap data dan mengurangi redudansi pada data. Salah satu proses utama dari ekstraksi ciri pada data berjenis citra digital adalah melakukan deteksi tepi, berikutnya dapat secara optional ditambahkan beberapa tahap tambahan seperti blur *filter* dan *thresholding* yang tujuannya untuk memperbaiki serta meningkatkan hasil deteksi tepi.

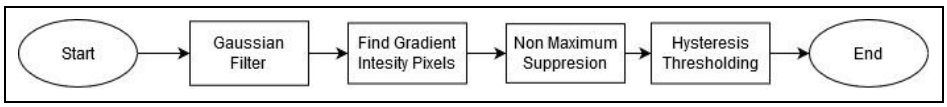
A. Edge Detection

Edge detection atau deteksi tepi merupakan sebuah teknik pengolahan citra untuk mencari suatu batas-batas dari objek dalam citra. Teknik ini umumnya

digunakan untuk melakukan segmentasi citra dan ekstraksi ciri dalam pengolahan citra. Teknik Deteksi tepi memiliki bermacam-macam jenis dan memiliki keunggulan tersendiri seperti Sobel, Prewitt, Robert dan Canny.

Deteksi tepi Canny merupakan deteksi tepi yang paling berguna, kokoh, dan populer digunakan untuk klasifikasi CNN, sebab telah sukses dapat mencari informasi tepi dari citra yang telah dibersihkan *noise* atau yang telah dilakukan praproses, sebab mempengaruhi performa deteksi tepi *Canny* (Mafi, dkk., 2018).

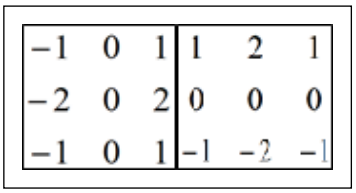
Proses dari deteksi tepi Canny terdiri dari empat tahap yaitu, pengurangan *noise* dengan *Gaussian Filter* yang dijelaskan pada bagian *Blur Filtering*, mencari intensitas gradien dari citra, melakukan *non maximum suppression*, dan *hysteresis thresholding* (Gary, 2000). *Flowchart* dari deteksi tepi Canny ditunjuk dalam Gambar 2.2.



Gambar 2. 2 Flowchart Canny Edge Detection (Gary, 2000)

Fungsi dari deteksi tepi Canny memiliki dua parameter yang mempengaruhi proses *thresholding*, dua parameter tersebut digunakan untuk menjadi batas nilai minimal dan nilai maximal dalam *thresholding*. Proses deteksi tepi Canny dapat dimulai ketika citra yang telah diakuisisi di-*grayscale*. Tahap selanjutnya adalah mencari nilai dan arah sudut gradien tepi tiap *pixel* yang digunakan juga pada proses deteksi tepi Sobel. Nilai gradien tepi ini terdiri dari dua nilai turunan berupa nilai representasi *direction* horizontal (G_x) dan vertikal (G_y). Untuk mendapatkan kedua nilai ini, dibutuhkan suatu matriks Sobel *mask*

3x3 yang dioperasikan dengan nilai *pixel* dalam citra untuk masing-masing nilai turunan (Vincent, 2009). Nilai matriks dari Sobel *mask* ditunjuk pada Gambar 2.3.



Gambar 2. 3 Matriks Sobel M*ask* untuk Mencari Nilai G_x dan G_y

Kemudian, untuk mendapatkan nilai dan arah sudut dari gradien tepi tiap *pixel* dalam citra, dibutuhkan suatu persamaan sebagai berikut.

$$|\mathrm{G}| = \sqrt{G_x^2 + G_y^2} \quad \ldots(3)$$

$$\theta = tan^{-1}\left(\frac{G_x}{G_y}\right) \quad \ldots(4)$$

Keterangan:

G: Nilai Gradien suatu *pixel*

G_y: Nilai turunan gradien *pixel* vertikal

G_x: Nilai turunan gradien *pixel* horizontal

θ: Arah gradien tepi

Untuk dapat melakukan *thresholding*, dibutuhkan suatu nilai yang telah dinormalisasi untuk representasi *pixel*. Nilai tersebut dapat diperoleh dari persamaan sebagai berikut.

$$G_v = \frac{G}{Max} * 255 \quad \ldots(5)$$

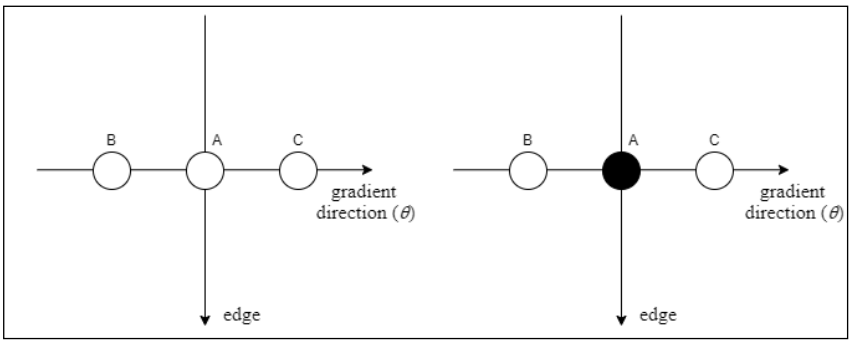
Keterangan:

G_v : Nilai gradien yang telah dinormalisasi.

G : Nilai gradien suatu *pixel* dalam citra.

Max : Nilai gradien maximum dalam satu citra.

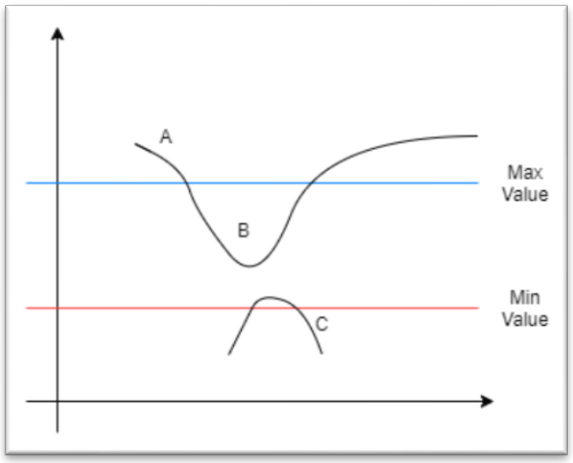
Tahap berikutnya, untuk membuang beberapa *pixels* yang bukan merupakan suatu tepi pada citra dilakukan suatu proses *non maximum suppression*. Proses ini membutuhkan suatu arah sudut gradien tepi (θ) yang telah didapat. Ilustrasi dari proses sebelum dan sesudah proses *non maximum suppression* ditunjuk dalam Gambar 2. 4



Gambar 2. 4 Ilustrasi Proses Non-maximum Suppression (Gary, 2000)

Pada ilustrasi terdapat tiga buat *pixel* gambar yang saling bersebelahan sesuai arah sudut gradien dimana awalnya ketiga *pixel* memiliki dideteksi sebagai suatu tepi dengan nilai kontras masing-masing *pixel* berbeda, kemudian jika *pixel* A dicek merupakan lokal maximum diantara dua pixel lainnya maka akan dilanjutkan proses *non maximum suppression* ke *pixel* lain jika bukan, maka *pixel* A tidak dijadikan tepi atau nilainya diubah menjadi 0, seperti pada sisi kanan Gambar 2.4. Hal ini digunakan untuk mempertipis garis tepis hasil operasi gradien sebelumnya.

Tahap terakhir pada deteksi tepi Canny yaitu *Hysteresis Thresholding*. Proses ini 2 buah nilai yang menjadi nilai minimum dan maximum yang mempengaruhi *thresholding*. Ilustrasi proses dari *Hysteresis Thresholding* ditunjuk dalam Gambar 2.5.



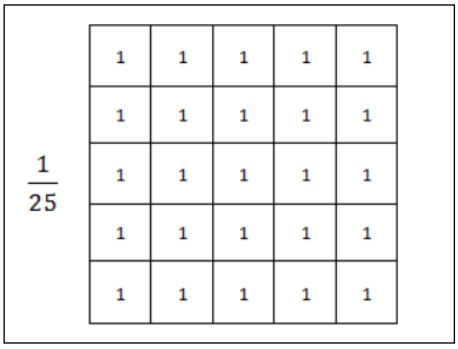
Gambar 2. 5 Ilustrasi *Hysteresis Thresholding* (Gary, 2000)

Prosesnya, jika suatu *pixel* memiliki nilai diatas maximum maka akan dianggap sebagai tepi (A), sedangkan jika suatu *pixel* memiliki nilai diantara maximum dan minimum serta terhubung dengan *pixel* yang dianggap sebagai tepi maka *pixel* tersebut akan dianggap sebagai tepi (B), terakhir jika nilai *pixel* berada dibawah nilai minimum maka tidak dianggap sebagai tepi (C). Tepi pada *thresholding* ini maksudnya adalah mengubah nilai suatu *pixel* menjadi 255, sedangkan bukan tepi mengubah nilai *pixel* menjadi 0.

B. Blur Filter

Setelah proses deteksi tepi, proses lanjutan ini adalah untuk meng-*improve* hasil dari deteksi tepi. Hasil dari *filter* ini nilai citra hasil deteksi tepi Canny ini tidak berupa bilangan biner. *Blur filter* merupakan salah satu proses pengolahan

citra yang digunakan untuk membuang *noise* pada citra dan detail kaku pada citra. Salah satu tekniknya adalah dengan menggunakan *Gaussian Filter*. Untuk dapat melakukan *Gaussian Filter* dibutuhkan suatu parameter berupa ukuran *n* matriks *kernel* yang dioperasikan ke citra. Ilustrasi dari matriks kernel berukuran 5x5 yang diterapkan untuk *Gaussian Blue* ditunjuk dalam gambar 2.6.



Gambar 2. 6 Matriks *kernel Gaussian Filter* 5x5 (Gary, 2000)

C. Thresholding

Thresholding merupakan salah satu tahap akhir dalam pengolahan citra dimana tujuannya mengubah data komponen citra yang sebelumnya berbentuk bilangan desimal dengan *range* 0 sampai 255 menjadi biner, sehingga sangat ringan untuk diolah ketika melakukan *training* suatu model. Salah satu teknik *thresholding* yang populer adalah *otsu threshold*. *Otsu thresholding* merupakan salah satu teknik *thresholding* yang dapat digunakan untuk *thresholding* secara global, teknik ini dipercaya dapat menyelesaikan masalah sensitivitas terhadap segmentasi objek dan meng-*improve* hasil dari segmentasi suatu citra (Zhang dan Hu, 2008).

Proses dari *otsu thresholding* sendiri mula-mula membagi citra yang berformat *grayscale* menjadi beberapa kelas berdasarkan tingkat kontras *grayscale*, hasil dari kelas ini merupakan pembagian numerical (0 hingga kelas terkontras). Setelah didapat tingkatan kelas berdasarkan kontras pada suatu *pixel*, dalam proses perhitungannya terdapat suatu pembatas kelas atau suatu golongan yang dinamakan *background* dan *foreground* untuk menghitung varians. Jika golongan *background* mendapat kelas 0 dan 1 (*subsequent*), maka golongan *foreground* mendapatkan kelas sisanya. Nilai varians setiap kelas dapat dicari dengan beberapa persamaan sebagai berikut

$$M_b = \frac{\sum_{k=0}^{i-1}(x_k * c_k)}{n_b}, M_f = \frac{\sum_{k=i}^{N}(x_k * c_k)}{n_f} \quad \ldots(6)$$

$$w_b = \frac{\sum(x_b)}{n}, w_f = \frac{\sum(x_f)}{n} \quad \ldots(7)$$

$$\sigma_b^2 = \frac{\sum_{k=0}^{i-1}(c_k - M_b)}{n_b}, \sigma_f^2 = \frac{\sum_{k=i}^{N}(c_k - M_f)}{n_f} \quad \ldots(8)$$

$$\sigma_i^2 = W_b * \sigma_b^2 + W_f * \sigma_f^2 \quad \ldots(9)$$

Batasan:

$$b < i, i <= f <= N$$

Keterangan:

i: Kelas yang sedang dihitung.

x : Frekuensi suatu kelas.

M: Mean suatu golongan.

n : Frekuensi suatu golongan.

w : Bobot suatu golongan.

c : Nilai tingkatan kelas berdasarkan kekontrasan.

N : Banyak tingkatan kelas.

σ : Varians suatu kelas atau golongan.

Nilai dari varians suatu kelas yang terkecil menjadikan *threshold*, dimana *foreground* akan menjadi tepi (255) dan pixel *background* akan dijadikan bukan tepi (0).

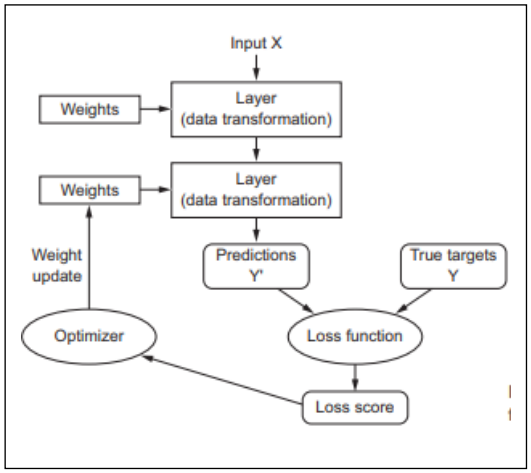
2.4 Deep Learning

Deep Learning merupakan bagian dari ranah *machine learning* baru yang menekankan pendekatan pembelajaran yang membangun banyak *layer* untuk menambah tingkat pembelajaran. Sedangkan bagian *machine learning* yang melibatkan satu atau dua *layer* sebagai representasi data disebutkan *shallow learning*.

Pengembangan *Deep Learning* dapat dibangun dengan suatu model yang kompleks yang dapat atau dapat disebut dengan nama *Neural Network*. *Neural Network* merupakan suatu set dari banyak algoritma, yang memiliki banyak jenis model yang dari tiap jenis algoritmanya, dan dibentuk untuk mengenali suatu pola. Pola yang dikenali dapat berupa pola numeric, data di dalam vektor dan matriks seperti citra, suara, teks, dan waktu yang harus diolah sebelumnya. (Guide to Neural Networks | Skymind, 2014).

Dalam *deep learning*, representasi dari *layer* yang membentuk model dapat disebutkan dengan nama *neural network*, tiap *layer* tersusun secara

terstruktur dan bertumpuk satu sama lain. Istilah *neural network* merupakan referensi dari ilmu neurobiologi, dimana beberapa bagian konsep utama dalam *deep learning* dikembangkan dari jaringan syaraf otak manusia. Dalam proses pengembangan algoritma *deep learning* untuk dapat mengatur hasil dari *deep learning* dibutuhkan suatu observasi. Pengamatan dapat dilakukan dengan membuat suatu *loss function* atau *objective function* dalam jaringan untuk dapat menghitung sebaik apa suatu jaringan menghasilkan output. Selain *objective function*, tiap *layer* dalam *deep learning* memiliki suatu bobot *weight* yang menjadi parameter untuk dapat mempengaruhi hasil. Selanjutnya terdapat suatu trik dasar dalam deep learning yang digunakan nilainya digunakan sebagai sinyal *feedback* untuk mengatur kembali nilai dari tiap *weight* dalam layer. Pengaturan ini dapat disebut suatu *optimizer* atau algoritma *backpropagation* (Chollet, 2018). Flowchart dasar dari *Deep Learning* ditunjuk dalam Gambar 2.7.



Gambar 2. 7 Flowchart Dasar Proses *Deep Learning*
(Chollet, 2018)

2.5 Convolutional Neural Network

Convolutional neural network adalah salah satu algoritma *Deep Learning* yang dapat mengolah sebuah citra gambar (*input image*), mencari suatu fitur penting (berdasarkan bobot dan bias) yang berdasar banyak aspek dan objek dalam citra sehingga dapat membedakan suatu citra dengan citra lain. CNN pengaplikasian utamanya hingga saat ini digunakan untuk pengenalan citra, klasifikasi citra, serta deteksi objek. Konvolusi yang berarti berbelit merupakan suatu operasi matematika atas dua atau lebih fungsi untuk menghasilkan fungsi baru dan merepresentasikan bentuk dari banyak fungsi yang terkait (Kumar, 2018).

CNN memiliki suatu keterbatasan seperti terlalu banyak memakan waktu jika tidak dilakukan pra proses dan ekstraksi dengan tepat. Hal ini menjadi masalah krusial. Batasan ini membuat dibutuhkannya suatu algoritma pengekstrak data yang memungkinkan training pada CNN lebih cepat (Seuret, 2017).

CNN secara garis besar terdiri dari 2 tahap. Tahap pertama merupakan tahap mengklasifikasi citra menggunakan *feedforward*. Tahap kedua merupakan tahap pembelajaran dengan metode *backpropagation*.untuk mengoptimalisasi pembelajaran (Eka, 2016).

Layer penyusun CNN terbentuk secara sekuensial, dimana data telah diolah menjadi input *layer*. Setiap *layer* baru yang tercipta merupakan hasil transformasi dari *layer* sebelumnya, proses ini terus dilakukan hingga *fully connected layer* terbentuk. Terdapat beberapa jenis fungsi yang digunakan untuk mentransformasi data dan membentuk suatu *layer* seperti *Conv2D*,

MaxPooling2D, *Flatten* dan *Dense*. Fungsi *Conv2D* dan *MaxPooling2D* digunakan untuk membentuk *layer extraction* sedangkan, *Flatten* dan *Dense* digunakan untuk membentuk *fully connected layer*. Penjelasan mengenai fungsi-fungsi yang digunakan adalah sebagai berikut:

- Conv2D

Conv2D merupakan fungsi pembentuk *layer convolution* 2 dimensi, dimana terdapat suatu *kernel* (matriks *convolution)* yang menjadi *filter* untuk merubah *layer input* menjadi *layer* baru. Terdapat dua parameter utama dalam fungsi ini yaitu jumlah *filter* (*f*) dan ukuran *kernel* (*n*) yang dipakai untuk operasi pembentukan *layer*. Fungsi ini selain mengembalikan *output layer* mengembalikan sejumlah parameter dengan sebanyak persamaan berikut.

$$p = (n * n) * f + 1 * f' \quad \ldots(10)$$

Keterangan:

p: Jumlah Parameter.

n: Ukuran Kernel Matriks.

f: Jumlah Filter Matriks Input.

f': Jumlah Filter Matriks Output.

- MaxPooling2D

MaxPooling2D merupakan fungsi pembentuk *layer* baru dengan menyusutkan dimensi panjang dan lebar layer menjadi *s* (*stride*) kali lebih kecil. Fungsi ini memiliki proses mengambil nilai terbesar dari suatu bagian dari matriks *layer* sebesar nilai ukuran *kernel*, hingga membentuk *layer* baru. Fungsi ini mengembalikan parameter *layer* 0.

- BatchNormalization

BatchNormalization menerapkan normalisasi data dalam dimensi *layer* dengan melakukan transformasi dengan me-*maintain* aktivasi rata-rata mendekati ke 0 dan aktivasi standar deviasi mendekati ke 1. Batch Normalization mengembalikan parameter sejumlah eksponen 2 pangkat.

- Dropout

Dropout merupakan *method* yang mengatur *rate* input menjadi suatu nilai antara 0 dan 1 setiap kali *epoch training* dilakukan supaya hasil *training* sebelumnya tidak mengganggu *training* baru hingga dapat mencegah *overfitting*. Dropout memiliki satu parameter yang menerima tipe data *float* antara 0 dan 1 menjadi nilai *rate* input.

- Flatten

Flatten adalah fungsi yang membentuk suatu *layer* baru dari *layer input* dan membuat dimensi *layer* baru sebesar satu. Jumlah kapasitas *layer* satu dimensi yang terbentuk adalah hasil kali dari semua multi dimensi *layer input*.

- Dense

Dense adalah fungsi yang digunakan untuk membuat *layer* baru dari *layer input* sebesar satu dimensi dengan perubahan mengkompres suatu data hingga memiliki kapasitas *layer* sebesar nilai bilangan desimal masukkan pada fungsi.

$$P = f + 1 * c \qquad \ldots(11)$$

Keterangan:

p: Jumlah Parameter.

f: Dimensi *Layer* Input.

c: Dimensi *Layer Output*.

Berikut contoh daftar bentuk dimensi tiap *layer* model dari suatu data dari dataset citra MNIST (*Modified National Institute of Standard Technology)*:

Tabel 2. 1 Contoh Penyusunan Layer CNN untuk Dataset MNIST (Chollet, 2018)

Layer	Output Shape	Parameter
conv2d_1 (Conv2D)	(26, 26, 32)	320
maxpooling2d_1 (MaxPooling2D)	(13, 13, 32)	0
conv2d_2 (Conv2D)	(11, 11, 64)	18496
maxpooling2d_2 (MaxPooling2D)	(5, 5, 64)	0
conv2d_3 (Conv2D)	(3, 3, 64)	36928
flatten_1 (Flatten)	(576)	0
dense_1 (Dense)	(64)	36928
dense_2 (Dense)	(10)	650

Dari transformasi-transformasi dimensi yang diolah fungsi terdapat suatu variabel / parameter yang mempengaruhi perubahan dimensi seperti ukuran *kernel*, *padding*, dan *stride*. Sehingga, muncul suatu rumus persamaan yang menghitung hasil dimensi / fitur output dari suatu input suatu nilai dimensi input fitur sebagai berikut:

$$n_{out} = \left[\frac{n_{in} + 2p - k}{s}\right] + 1 \quad \ldots(12)$$

Keterangan :

n_{in}: Jumlah dimensi input.

n_{out}: Jumlah dimensi output.

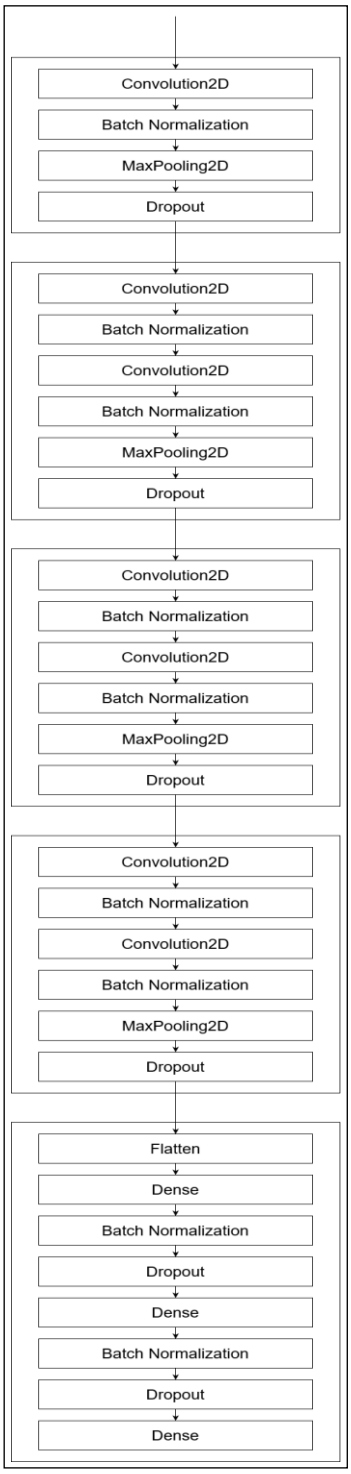
K: Ukuran dari kernel / matriks filter pada fungsi seperti Conv2D.

p: Ukuran *padding*.

S: Ukuran dari *stride* / matriks filter pada fungsi Pooling.

Sifat dari CNN sendiri adalah melakukan pembelajaran dari *layer* yang dibangun dengan mengenali pola lokal pada gambar, sebab *layer* pertama dari CNN akan mengenali pola lokal data seperti suatu ujung (*edges*), *layer* berikutnya akan mengenali pola yang lebih mendalam dari fitur yang didapat dari *layer* sebelumnya (Chollet, 2018). Hingga jika suatu *layer* telah selesai dibangun, maka dapat dilanjutkan dengan proses *training* pembentukan model.

Pada penelitian ini jumlah arsitektur dari model CNN yang digunakan memiliki referensi dari *kaggle* dengan *username* bernama trolukovich yang diaplikasikan untuk mengklasifikasi nama pokemon dari citra pokemon dengan akurasi tertinggi sebesar 94%. Arsitektur model terdiri dari 7 *layer* Conv2D dan *fully connected layer* yang terdiri dari 1 *layer* Flatten, dan 3 *layer* Dense (Trolukovich, 2019). Ilustrasi struktur dari model CNN ditunjuk dalam Gambar 2.8.



Gambar 2. 8 Struktur Model CNN

2.6 Sistem Presensi

Presensi adalah suatu pendataan kehadiran, bagian dari pelaporan aktivitas suatu institusi, atau komponen institusi itu sendiri yang berisi data-data kehadiran yang dibuat untuk keperluan pihak yang memiliki otoritas dan kepentingan. Sistem presensi memiliki banyak jenis mulai dari yang manual hingga otomatis yang terkomputerisasi, salah satunya adalah pengenalan wajah (Wardoyo, dkk., 2014).

Sedangkan sistem merupakan suatu kumpulan dari komponen-komponen yang dimiliki unsur keterkaitan antara satu dengan lainnya. Sistem terbagi menjadi dua kategori pendekatan yaitu, sebagai prosedur di mana menjadikan sistem sebagai jaringan dari banyak prosedur yang saling terhubung untuk menyelesaikan masalah dan sebagai elemen atau komponen yang berinteraksi untuk mencapai tujuan bersama (Hartono, 2005).