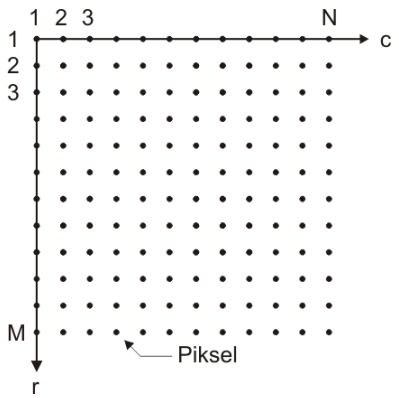
BAB 2
LANDASAN TEORI

2.1. Citra dan Matriks Citra

Citra dilambangkan dengan fungsi dua dimensi, f(x,y) dimana x dan y adalah koordinat spasial. Amplitudo merupakan fungsi yang terdapat pada setiap titik (x,y) merupakan intensitas atau kecerahan citra pada titik tersebut. *Gray level* adalah intensitas dari citra monokrom atau satu warna. Citra berwarna terdiri dari beberapa citra monokrom, misalnya RGB terdiri dari 3 kanal dan terbentuk dari kombinasi tiga warna yaitu, merah, hijau dan biru.

Citra digital digambarkan dalam bentuk matriks dimana indeks baris dan kolomnya menyatakan suatu titik pada citra yang disebut piksel dan elemen matriksnya menyatakan tingkat keabuan pada titik tersebut. Gambar berikut ini adalah contoh koordinat suatu piksel dari sebuah citra digital dengan x adalah baris dan y adalah kolom [11].



Gambar 2.1 Koordinat Piksel

2.2. Pengolahan Citra

Pengolahan citra adalah pemrosesan sebuah gambar dua dimensi secara digital, Proses – proses yang dapat dilakukan dalam pengolahan citra digital adalah sebagai berikut [11]:

1. Merubah dari citra berwarna ke dalam citra abu-abu
2. Merubah dari citra berwarna atau citra abu-abu ke dalam citra biner. Proses ini dapat dilakukan dengan deteksi tepi atau filtering.

Tujuan dari pengolahan citra digital adalah untuk memperbaiki kualitas suatu gambar agar dapat dengan mudah dipahami oleh mata manusia dan komputer untuk mengolah informasi yang terdapat pada gambar tersebut untuk keperluan pengenalan objek secara otomatis.

2.3. Kartu Tanda Penduduk Elektronik (E-KTP)

E-KTP atau KTP Elektronik adalah dokumen kependudukan yang berisi dengan sistem keamanan atau pengendalian, baik dari sisi administrasi maupun dari sisi teknologi informasi berbasis pada database kependudukan nasional. E – KTP mempunyai sebuah chip yang menyimpan berbagai data seperti data biometrik berupa sidik jadi. Chip tersebut telah dienkripsi atau diamakan dengan algoritma kriptografi tertentu. Sesuai dengan aturan yang berlaku setiap penduduk Indonesia yang telah menginjak usia 17 tahun atau lebih wajib memiliki E-KTP [12].

Setiap E-KTP memiliki perbedaan, perbedaan itu terletak pada nomor NIK. Nomor NIK banyak digunakan sebagai syarat dalam penerbitan Paspor, SIM, Nomor Pokok Wajib Pajak (NPWP), Polis Asuransi, Sertifikat ataa Hak Tanah dan penerbitan dokumen identitas lainnya (Pasal 13 UU No. 23 Tahun 2006 tentang Adminduk). NIK pada E-KTP memiliki karakteristik yang berbeda pada setiap E-KTP oleh karena itu NIK dijadikan sebagai *primary key* pada E-KTP [12].

2.4. Ekstraksi Informasi

Ekstraksi Informasi adalah proses pengambilan informasi secara otomatis dari informasi terstruktur seperti entitas, hubungan antar entitas, dan atribut deskripsi entitas dari sumber yang tidak terstruktur. Kegiatan ini biasanya berkaitan dengan pemrosesan teks bahasa manusia melalui pemrosesan bahasa alami (NLP). Kegiatan dalam pemrosesan dokumen multimedia seperti anotasi otomatis dan

ekstraksi konten dari gambar / audio / video / dokumen dapat disebut juga dengan ekstraksi informasi. Dalam penelitian ini akan melakukan ekstraksi informasi pada KTP [13]. Pada penelitian ini metode ekstraksi informasi yang digunakan adalah *rule-based*, Sistem berbasis aturan (*Rule Based System*) adalah sebuah program komputer yang melakukan pemrosesan informasi yang ada di *working memory* dengan menggunakan sekumpulan aturan yang terdapat di dalam dasar pengetahuan menggunakan mesin inferensi untuk menghasilkan informasi baru. Metode *rule-based* bisa digunakan ketika dokumen yang digunakan merupakan dokumen yang terstruktur [6]. Informasi dokumen yang dapat diperoleh dengan cara mencari struktur suatu dokumen salah satunya adalah KTP.

2.5. *Optical Character Recognition* (OCR)

(*Optical Character Recognition*) OCR merupakan sebuah aplikasi komputer yang digunakan untuk mengenali citra huruf maupun angka untuk dirubah ke dalam bentuk tulisan. Sistem pengenal huruf ini dapat meningkatkan fleksibilitas atau kemampuan dan kecerdasan sistem komputer. Sistem ini banyak digunakan karena sangat membantu usaha digitalisasi informasi dan pengetahuan, misalnya dalam pembuatan koleksi pustaka digital, koleksi sastra kuno digital, dan lain-lain yang saat ini dilakukan banyak pihak [14]

Secara umum proses OCR dapat dilihat pada gambar 2.2, dengan penjelasan sebagai berikut [15]:

a. File Input

   File input berupa file citra digital dengan format *.bmp atau *.jpg.

b. *Preprocessing*

   Preprocessing adalah proses yang bertujuan untuk menghilangkan bagian-bagian yang tidak diperlukan pada gambar *input* untuk proses selanjutnya

c. Segmentasi

   Segmentasi adalah proses membagi daerah yang ingin diamati(*region*) pada tiap karakter yang dideteksi.

d. Normalisasi

Normalisasi adalah proses merubah ukuran *region* tiap karakter dan ketebalan karakter.

e. Ekstraksi ciri

Ekstraksi ciri adalah proses untuk mengambil ciri-ciri tertentu dari karakter yang diamati.

f. *Recognition*

Recognition adalah proses pengenalan karakter yang dilakukan dengan cara membandingkan ciri-ciri karakter yang diperoleh dengan ciri-ciri karakter yang ada pada basis data

Gambar 2.2 Proses OCR Secara Umum

2.6. *Resize* Citra

Resize Citra atau penskalaan merupakan proses merubah ukuran citra menjadi lebih kecil atau lebih besar. Proses *resize* pada penelitian ini tidak menggunakan metode khusus. *Resize* dilakukan agar pada saat komputer sedang melakukan pemrosesan citra kinerja dari komputer akan lebih cepat dan tidak banyak menghabiskan ruang memori penyimpanan di dalam memori sementara [16]. Hasil *resize* dapat dilihat pada gambar berikut ini

Gambar 2.3 Contoh Citra *Resize*

2.7. *Grayscale*

Grayscale merupakan proses merubah citra dari awal mula RGB kanal menjadi citra 1 kanal, sehingga nilai yang ditampilkan hanyalah nilai intensitas atau dikenal juga dengan istilah derajat keabuan [17]. Jenis citra *grayscale* ini disebut juga sebagai 8-bit image karena untuk setiap pikselnya memerlukan ruang penyimpanan sebesar 8 bit. Proses *grayscale* bertujuan untuk merubah citra berwarna menjadi citra berskala keabuan. Secara umum perubahan citra berwarna menjadi citra *grayscale* menggunakan rumus [18]:

$$I = 0.2989 \; x \; R + 0.5870 \; x \; G + 0.1141 \; x \; B \qquad (2.1)$$

2.8. Segmentasi

Segmentasi adalah sebuah proses yang bertujuan untuk membagi sebuah citra menjadi daerah pilihan atau mengisolasi objek dari citra secara keseluruhan, segmentasi dapat dilakukan berdasarkan tekstur, kecerahan, serta intentitas jumlah piksel. Pada proses ini akan melakukan pembagian citra menjadi dua wilayah, yaitu wilayah latar dan wilayah teks. Setelah melakukan pembagian, OCR akan melakukan proses selanjutnya hanya pada wilayah teks yang sudah terbagi [19]. Pada penelitian ini metode segmentasi yang digunakan adalah profil proyeksi, metode ini bekerja dengan dua tahap yaitu mencari baris dari citra (*horizontal)* dan mencari karakter (*vertical)*. Profil Proyeksi akan mencari garis pada teks secara vertikan dan horizontal, dimana proyeksi horizontal akan mendapatkan baris teks kemudian proyeksi vertikal akan memisahkan setiap kolom karakter.

Dimana S(N, M) merupakan citra biner dengan N baris dan M kolom. Profil proyeksi terdapat 2 jenis yaitu [20]:

1. Profil Vertikal

   Menjumlahkan pixel putih yang tegak lurus dengan sumbu y, yang diwakili vektor P_{ver} dengan ukuran N yang didefinisikan sebagai:

$$P\ ver\ [k] = \sum_{j=1}^{m} I[bk, k] \qquad (2.2)$$

   Dengan ketentuan sebagai berikut :

   M = tinggi citra

   $P_{ver}[k]$ = jumlah piksel pada kolom k citra

2. Profil Horizontal

   Menjumlahkan pixel putih yang tegak lurus dengan sumbu x, yang diwakili vektor Phor dengan ukuran M yang didefinisikan sebagai:

$$P\ hor\ [b] = \sum_{j=1}^{n} I[b, kj] \qquad (2.3)$$

   Dengan ketentuan sebagai berikut :

   N = lebar citra

   $P_{hor}[b]$ = jumlah piksel pada baris b citra

2.9. *Thresholding*

Thresholding merupakan metode paling sederhana dan banyak digunakan pada segmentasi citra. Dari citra *grayscale*, *thresholding* dapat digunakan untuk membuat citra biner dengan cara merubah nilai pada piksel menjadi 0 dan 1. Pada proses *thresholding*, setiap piksel yang ada pada sebuah citra ditandai sebagai piksel milik objek jika nilainya lebih besar dari nilai *threshold*. Dengan asumsi objek tersebut lebih terang dari latar belakangnya, ini biasa disebut dengan *threshold above*. Sedangkan kebalikannya disebut *threshold below*. Jika piksel-piksel suatu objek berada diantara dua *threshold* disebut *threshold inside* dan

kebalikannya disebut *threshold outside*. Biasanya piksel suatu objek dilabeli dengan nilai "1" sementara piksel-piksel latar belakangnya dilabeli dengan nilai "0". *Thresholding* dapat dibentuk dalam persamaan sebagai berikut [18]:

$$g(x,y) = \begin{cases} 1 & if \quad f(x,y) \geq T \\ 0 & if \quad f(x,y) < T \end{cases} \qquad (2.4)$$

Pada penelitian ini metode *thresholding* yang digunakan adalah *hysteresis thresholding*, metode ini banyak digunakan untuk melakukan *edge detection* dalam metode canny, Pada metode *Hysteresis Thresholding* ini terdapat 2 nilai ambang batas yaitu Th_{high} dan Th_{low}, dapat dilihat dari persamaan berikut [21]:

$$I_t(x,y) = \begin{cases} foreground \ \ (strong) & if \ \ I_t(x,y) > Th_{high} \\ background & if \ I_t(x,y) < Th_{low} \\ candidate \ \ (weak) & otherwise \end{cases}$$

Gambar 2.4 *Hysteresis Thresholding*

Nilai piksel pada matriks citra yang lebih rendah dari nilai Th_{low} akan dianggap sebagai nilai dari background begitu sebaliknya jika nilai piksel dari citra melebihi Th_{high} maka nilai piksel tersebut menjadi foreground dan jika tidak berada diantara Th_{high} dan Th_{low} maka itu menjadi hanya kandidat (lemah). Untuk menghitung nilai *threshold* menggunakan persamaan sebagai berikut :

$$t1v = t1 * (maxv - minv) + minv \qquad (2.5)$$

$$t2v = t2 * (maxv - minv) + minv \qquad (2.6)$$

Dimana :

T1v : nilai *threshold* yang batas bawah

T2v : nilai *threshold* yang batas atas

T1 : ambang batas bawah

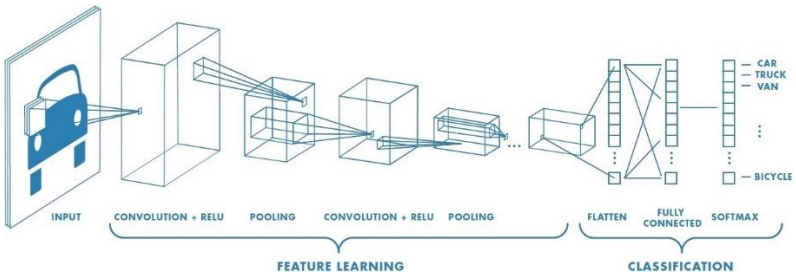
T2 : ambang batas atas

Maxv : nilai maksimal dari matriks piksel citra

Minv : nilai maksimal dari matriks piksel citra

2.10. *Convolutional Neural Network* (CNN)

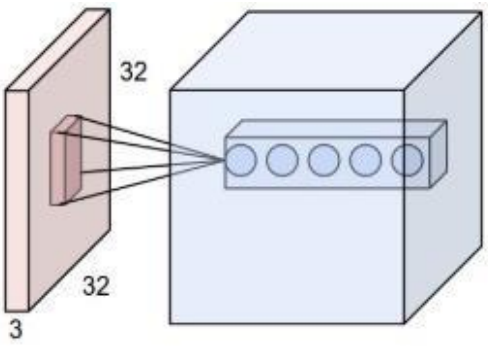
Convolutional Neural Networks, adalah salah satu jenis dari *Deep Learning, Deep Learning* merupakan bagian dari *Artificial Neural Network* ,yang banyak digunakan pada analisis citra. CNN terdiri dari satu lapisan masukan (*input layer*), satu lapisan keluaran (*Output Layer*), dan beberapa lapisan tersembunyi (*hidden layer*) [22]. Lapisan tersembunyi biasanya berisi *convolutional layers, pooling layers, normalization layers, ReLu layer, fully connected layers,* dan *loss layer.* Lapisan – lapisan tersebut disusun secara bertumpuk-tumpuk [23].



Gambar 2.5 Arsitektur CNN

2.10.1. *Convolutional Layer*

Convolutional layer terdiri dari neuron yang tersusun sehingga membentuk sebuah filter dengan panjang dan tinggi. Sebagai contoh pada volume *input* memiliki ukuran 32 x 32 x 3 [24]. Jika ukuran filter adalah 5 x 5, maka pada setiap neuron yang terdapat di lapisan *convolutional* akan mempunyai bobot ke wilayah 5 x 5 x 3 = 75 bobot dan +1 parameter bias.



Gambar 2.6 *Convolutional Layer*

CNN pada umumnya menggunakan lebar langkah atau stride dengan zero padding sebesar:

$$\boldsymbol{p = \frac{(f-1)}{2}} \qquad (2.7)$$

Di mana P adalah ukuran padding dan F adalah ukuran bidang reseptif atau tingkat spasial yang tentu saja sama dengan ukuran filter. Konvolusi sebenarnya adalah proses *dot product* antara filter dengan sebuah bidang reseptif kecil pada citra masukan yang berukuran sama dengan ukuran filter, dimana filter dilambangkan $k \times k$. Adapun persamaannya adalah sebagai berikut [24]:

$$Convj_{x,y} = (I \underset{3}{\otimes} C + B_j)\underset{3}{,}$$
$$dimana \otimes = \sum_{u=1} \sum_{v=1} I_{x+u,y+v} * C_{u,v} \qquad (2.8)$$

Keterangan:

$I_{x,y}$ = Nilai pixel dari input ke-x,y.

$C_{u,v}$ = Nilai pixel dari filter ke-u,v.

B_j = Nilai bias ke-j.

Persamaan diatas digunakan pada saat operasi konvolusi di layer pertama, untuk melakukan operasi konvolusi ke layer selanjutnya persamaan yang digunakan mengalami perubahan menjadi inputan dari layer sebelumnya sebanyak jumlah filter masing-masing yang akan dikonvolusi kemudian akan dijumlahkan. Persamaan tersebut adalah sebagai berikut:

$$Convj = \sum_{i=1}^{filter} pool_i \otimes C_{i,j} + B_j \qquad (2.9)$$

Keterangan:

$Convj$ = Layer konvolusi ke-j.

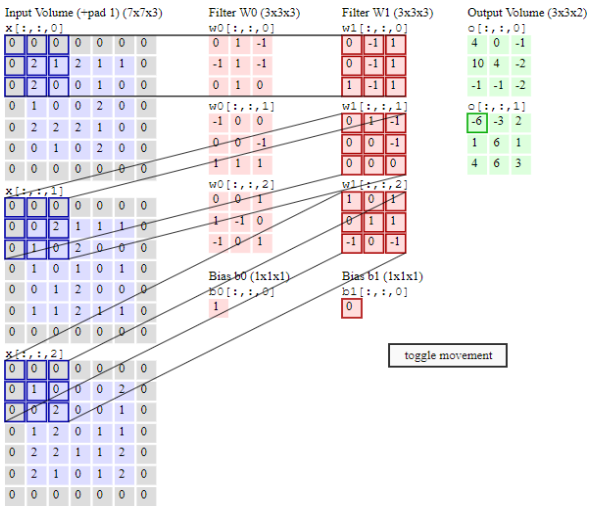
$pool_i$ = Layer pooling ke-i.

$C_{i,j}$ = Kernel index ke-j dan filter ke-i.

Bj = Bias ke-j.

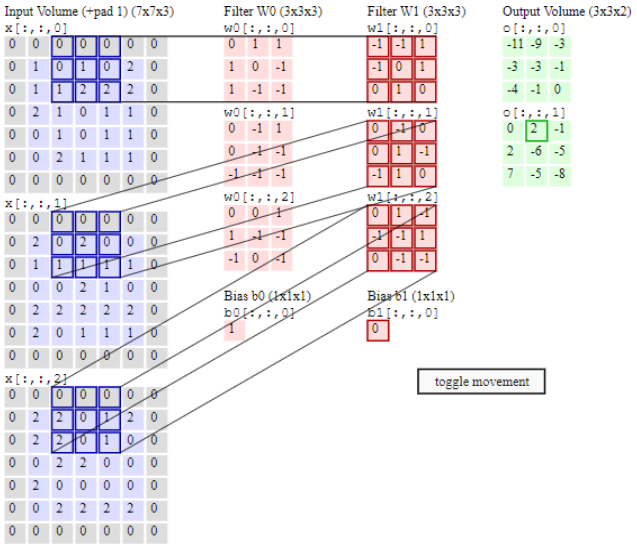
$\otimes$ = Operasi konvolusi .

Adapun gambaran dari proses konvolusi telah tergambarkan pada Gambar 2.7



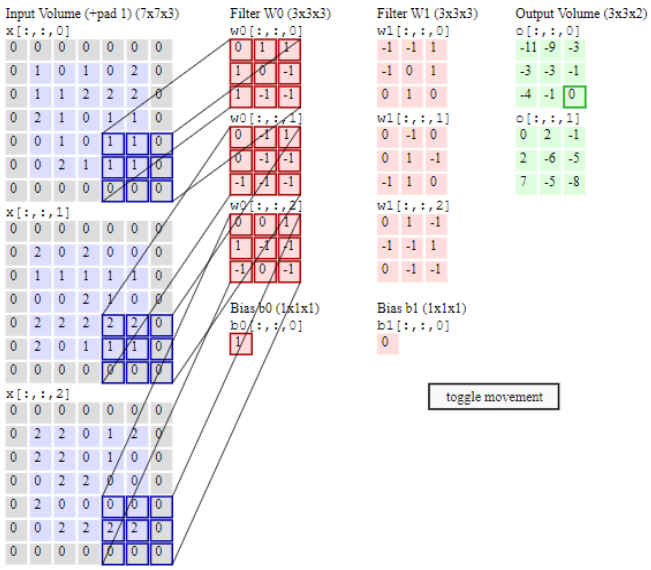
Gambar 2.7 Proses Konvolusi Pada Bagian Kiri Atas Citra Masukan

Selanjutnya filter bergeser dua piksel ke kanan yang digambarkan pada Gambar 2.8



Gambar 2.8 Proses Konvolusi Setelah Pergeseran Dua Piksel Ke Kanan

Proses konvolusi akan terus dilakukan dengan cara menggeser filter dua piksel ke kanan hingga sampai pada bagian paling kanan bawah citra masukan yang digambarkan pada gambar 2.9.



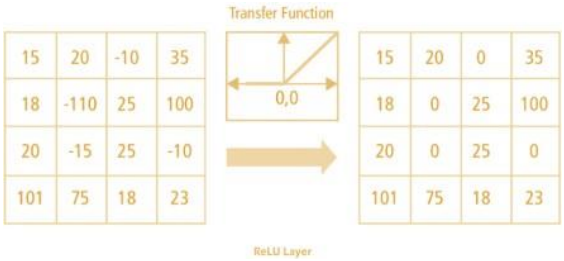
Gambar 2.9 Proses Konvolusi Setelah Pergeseran

2.10.2. *ReLu Layer*

Rectified Linear Units (*ReLu*) *layer* adalah Lapisan yang didalamnya terdapat fungsi aktivasi f (x) = max (0, x). Cara kerja dari fungsi aktivasi ini adalah ketika ada nilai piksel yang memiliki nilai < 0 akan dirubah menjadi 0, fungsi ini

meningkatkan sifat nonlinearitas fungsi keputusan dan jaringan secara keseluruhan dengan tidak memberikan dampak terhadap bidang-bidang di *convolutional layer*. Tujuan dari *ReLu* adalah untuk memperkenalkan nonlinearitas kepada CNN. Karena data pada dunia nyata ingin agar CNN mempelajari nilai-nilai nonnegatif. Adapun rumus pada *ReLu layer* adalah [23]:

$$f(x) = ReLu(x) = \begin{cases} Cj_{x,y} & Jika(Cj_{x,y} \geq 0) \\ 0 & Jika\ Tidak \end{cases} \quad (2.10)$$



Gambar 2.10 *ReLu* Operation

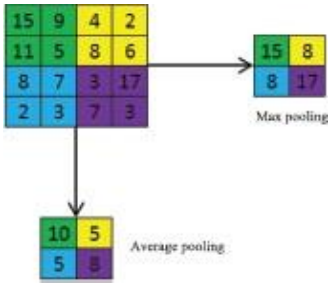
2.10.3. *Pooling Layer*

Pooling Layer memiliki fungsi untuk menjaga ukuran data ketika proses *convolution*,dengan cara melakukan *downsampling* atau mengurangi sampel. Dengan *pooling* kita dapat merepresentasikan data menjadi lebih kecil, mudah dikelola, dan mudah mengontrol *overfitting*. Proses *pooling* yang sering digunakan adalah *max pooling*, cara kerjanya dengan memilih nilai maksimum pada suatu daerah tertentu [23]. Dari empat nilai di bagian kiri atas akan dihasilkan satu nilai maksimum yaitu 100, untuk mengisi data hasil *pooling*. Hal yang sama dilakukan untuk empat nilai di bagian kanan atas, kanan bawah dan kiri bawah. Teknik *max pooling* lebih baik digunakan karena dapat memberikan performansi lebih baik dibanding 2 proses *pooling* lainnya yaitu *average pooling* dan *L2-norm pooling*. Pada penelitian kali ini fungsi *Max Pooling* akan digunakan, adapun rumus fungsi Max pooling adalah sebagai berikut:

$$Pool_{x,y} = Max(Conv_{x,y}, Conv_{x+1,y}, Conv_{x,y+1}, Conv_{x+1,y+1}) \quad (2.11)$$

Keterangan:
$Pool_{x,y}$ = Hasil dari *pooling layer*.

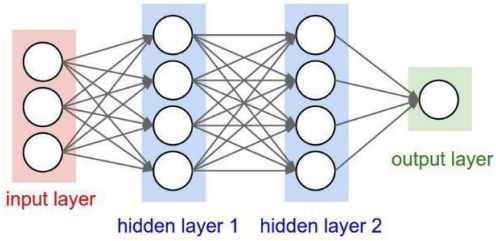
$Conv_{x,y}$ = Nilai pixel dari hasil *convolutional layer*.



Gambar 2.11 *Max Pooling* dan *Average Pooling*

2.10.4. *Fully Connected Layer*

Pada lapisan ini setiap neurons saling terhubung dan memiliki koneksi penuh ke semua aktivasi dalam lapisan sebelumnya. Hal ini sama dengan MLP. Model aktivasinya pun sama persis dengan MLP, yaitu komputasi menggunakan suatu perkalian matriks yang diikuti dengan bias *offset*. Sesuai dengan namanya, *Multi Layer Perceptron* memiliki lapisan yang berlapis dan memiliki beberapa *hidden layer*, *activation function* dan *output layer* [25].



Gambar 2.12 Arsitektur MLP Secara Umum

Fully connected layer berfungsi untuk melakukan mengklasifikasi data masukan. Keluaran yang dihasilkan dari *pooling layer* masih berbentuk *multidimensional array*, sehingga dibutuhkan proses *flatten* terlebih dahulu untuk mengubah data menjadi vektor sebelum input untuk *fully connected layer* [24]. Adapun persamaan yang akan digunakan pada tahapan ini adalah sebagai berikut:

$$Fully_i = \sum_{j}^{Length(flatten)} W_{i,j} * flatten_j + b_i \quad (2.12)$$

Keterangan:

$Fully_i$ = Hasil dari perhitungan pada *fully-connected layer*.

$W_{i,j}$ = Nilai bobot yang digunakan dari hasil *convolutional layer*.

$flatten_j$ = Nilai dari vektor ke-j.

i = Kelas ke-i (i = 1,2,3,4,5,6,7).

hasil dari $Fully$ akan diaktifasi lagi dengan menggunakan fungsi *softmax*, penggunaan fungsi softmax dilakukan karena pada penilitian ini menghasilkan *output* atau jumlah kelas lebih dari dua, yang biasa disebut dengan multi-class. Fungsi softmax bertujuan agar dapat diketahui prediksi yang dihasilkan dari arsitektur yang digunakan, dimana rumus untuk fungsi *softmax* ditulis pada persamaan.

$$Softmax_i = \frac{e^{Fully_i}}{\left(\sum_{j=1}^{kelas} e^{Fully_j}\right)} \quad (2.13)$$

Keterangan:

^y = Hasil dari aktifasi fungsi softmax.

$Fullyi$ = Hasil dari perhitungan pada fully-connected layer ke-i.

$flattenj$ = Nilai dari vektor ke-j.

2.10.5. *Cross-Entropy Loss Function*

Selanjutnya akan dicari nilai *error* yang didapat dari hasil prediksi pada *fully-connected layer* sebelumnya, nilai *error* akan digunakan untuk menentukan hasil prediksi dari CNN sudah mencapai target atau belum, oleh karena itu dilakukan proses perbandingan antara *error* dengan *target error* dimana, jika *error* masih dibawah dari target, maka akan dilakukan *loss function*. *Loss function* memiliki beberapa jenis yang ada, salah satu-nya adalah *Cross-entropy Loss Function*, dimana rumus dari *cross-entropy* tersebut dituliskan pada persamaan dibawah ini [23]:

$$Loss = -\sum_{i}^{kelas} Y_i * Log(\hat{y}) \quad (2.14)$$

Keterangan:

$\hat{y}i$ = Hasil dari aktifasi fungsi softmax.

t_i = Nilai dari target, bernilai 1 apabila target adalah kelas yang dituju,
dan bernilai 0 apabila nilai target bukanlah kelas yang dituju.

i = Kelas ke-i (i = 1,2,3,4,5,6,7).

2.10.6. *Backpropagation*

Backpropagation adalah salah satu algoritma *supervised learning* yang digunakan dalam *artificial neural networks*. *Backpropagation* akan mencari nilai bobot terbaik agar dapat meminimalkan kesalahan *output* agar dapat menjadi solusi yang dianggap benar [26]. Adapun tahapan *Backpropagation* adalah sebagai berikut:

1. Turunan gradien error terhadap softmax

   Tahapan turunan gradien error terhadap softmax berdasarkan rumus yang ditulis oleh Peter Sadowski.

$$\Delta\hat{y}i = \frac{\partial Loss}{\partial\hat{y}i} = \hat{y}i - ti \quad (2.15)$$

   Keterangan:

   $\Delta\hat{y}$ = Nilai turunan dari fungsi *softmax*.

   t_i = Nilai dari target, bernilai 1 apabila target adalah kelas yang dituju, dan bernilai 0 apabila nilai target bukanlah kelas yang dituju.

   i = Kelas ke-i (i = 1,2,3,4,5,6,7).

2. Turunan gradien error terhadap bobot

   Tahapan turunan gradien error terhadap bobot berdasarkan rumus berikut:

$$\Delta W_{i,j} = \frac{\partial Loss}{\partial W_i} = \Delta\hat{y} * flatten_j \qquad (2.16)$$

3. Turunan gradien error terhadap bias

   Tahapan turunan gradien *error* terhadap bobot berdasarkan rumus berikut:

$$\Delta b_i = \frac{\partial Loss}{\partial b_i} = \Delta\hat{y} \qquad (2.17)$$

2.10.7. *Stocastic Gradient Descent*

Stochastic Gradient Descent merupakan sebuah algoritma yang digunakan untuk mencari nilai minimum lokal dari sebuah fungsi, Algoritma dari *Stochastic Gradient Descent* dapat dilihat pada gambar dibawah:

function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ
    # where: L is the loss function
    # f is a function parameterized by θ
    # x is the set of training inputs $x^{(1)}$, $x^{(2)}$,..., $x^{(n)}$
    # y is the set of training outputs (labels) $y^{(1)}$, $y^{(2)}$,..., $y^{(n)}$

$\theta \leftarrow 0$
repeat T times
  For each training tuple ($x^{(i)}$, $y^{(i)}$) (in random order)
  Compute $\hat{y}^{(i)} = f(x^{(i)};\theta)$ # What is our estimated output $\hat{y}$?
  Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$) from the true output $y^{(i)}$?
  $g \leftarrow \nabla_\theta L(f(x^{(i)};\theta), y^{(i)})$ # How should we move θ to maximize loss ?
  $\theta \leftarrow \theta - \eta\, g$ # go the other way instead
return θ

Gambar 2.13 Algoritma *Stochastic Gradient Descent*

Dimana dari algoritma tersebut, bisa juga dirumuskan nilai perbaikan bobot dan bias baru seperti pada persamaan berikut:

1. Perbaikan nilai pada bobot

$$\theta W = W - \alpha(\Delta W) \qquad (2.18)$$

Keterangan:

ΔW = Nilai turunan dari bobot.

W = Nilai bobot.

α = Nilai *learning rate*.

θW = Nilai bobot baru.

2. Perbaikan nilai pada bias

$$\theta b = b - \alpha(\Delta b) \quad (2.19)$$

Keterangan:

Δb = Nilai turunan dari bias.

b = Nilai bias.

α = Nilai *learning rate.*

θb = Nilai bias baru

2.11. Pemodelan Sistem

Pada suatu sistem terdapat beberapa proses di dalamnya, yang mana proses tersebut harus dimodelkan untuk memperjelas gambaran yang terjadi pada proses tersebut berjalan. Berikut ini adalah pemodelan sistem yang digunakan pada penelitian ini, yang terdiri dari Blok Diagram, *DFD*, Diagram Konteks, dan *Flowchart* [27].

2.12. Blok Diagram

Proses yang sedang berjalan di suatu sistem dapat digambarkan dengan melalui data masukan, proses, serta keluaran yang dihasilkan. Model yang dapat menggambarkan tiga tahapan tersebut adalah model blok diagram. Blok diagram berfungsi untuk merepresentasikan suatu sistem atau sejumlah blok dalam rangkaian beberapa proses dengan cara menggunakan blok. Komponen yang ada pada diagram blok terdiri dari sebab akibat *input* dan *output* [27]. Berikut gambaran blok diagram secara umum:

Gambar 2.14 Blok Diagram

Penggunaan blok diagram pada penelitian ini untuk menggambarkan beberapa proses yang terjadi pada sistem dari mulai data masukan sampai hasil yang diharapkan.

2.13. *Data Flow Diagram* (DFD)

Model diagram yang bisa mendeskripsikan keterkaitan antar proses yang ada pada sebuah secara mendetail dapat dibuat dengan menggunakan diagram Data Flow Diagram (DFD). DFD adalah suatu model logika data atau proses yang digunakan untuk menggambarkan darimana data berasal, dan tujuan data yang keluar dari sistem, tempat data disimpan, proses apa yang menghasilkan data tersebut, dan interaksi antara data yang tersimpan, serta proses yang dicocokan pada data tersebut [28]. Terdapat 4 simbol utama yang ada di DFD yaitu sebagai berikut:

1. Entitas Luar (*External Entity*)

   Entitas luar digunakan untuk menggambarkan departemen, kantor, organisasi, orang, maupun sekelompok orang.
2. Arus Data (*Data Flow*)

   Arus data digunakan untuk meilustrasikan masukan atau keluaran dari proses yang sedang berjalan.
3. Proses (*Process*)

   Proses digunakan untuk menggambarkan pekerjaan yang dilakukan oleh manusia maupun sistem.
4. Simpanan Data (*Store Data*)

   Simpanan data berfungsi untuk menggambarkan penyimpanan data dari suatu sistem maupun memanggil data dari simpanan data.

Pada penelitian ini, DFD digunakan untuk menggambarkan beberapa proses yang terjadi pada sistem *optical character recognition* menggunakan CNN dan ekstraksi informasi menggunakan *rule-based*.

2.14. Digram Konteks

Model sistem yang belum dianalisis secara detail pada tahapan proses yang ada di suatu sistem disebut dengan diagram konteks. Diagram konteks merupakan bagian *Data Flow Diagram* (DFD) yang hanya memiliki satu simbol. Diagram konteks disebut juga sebagai top level pada penggambaran *Data Flow Diagram* (DFD) [28]. Diagram konteks memiliki karakteristik, yaitu:

1. Hanya ada satu proses saja.
2. Tidak ada penomoran khusus pada proses yang ada.
3. Penggambaran arus data dan datanya dilakukan secara jelas.

Pada penelitian ini, diagram konteks digunakan untuk menjelaskan alur masukan, proses dan keluaran data secara menyeluruh.

2.15. *Flowchart*

Aliran data yang ada pada suatu sistem dapat digambarkan dengan menggunakan diagram *flowchart*. *Flowchart* atau bagan alir adalah suatu bagan yang berisi simbol – simbol grafis yang menggambarkan arah aliran kegiatan dan data-data yang dimiliki program saat program dijalankan. Pada *flowchart* terdapat tiga komponen yaitu terdapat *input*, proses dan *output*, dan sebagai tambahannya terdapat simbol kondisional yang terletak diantara *input* dan *output*. *Flowchart* pada penelitian ini digunakan untuk menggambarkan perancangan prosedur – prosedur yang terdapat pada sistem *optical character recognition* dan ekstraksi informasi [29].

2.16. Matlab

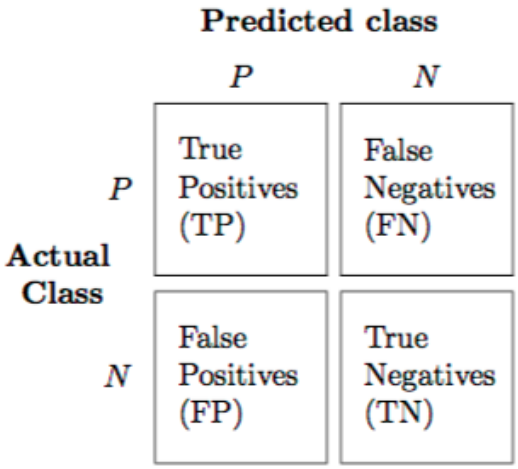
MATLAB (*Matrix Laboratory*) adalah suatu bahasa pemograman matematika yang dibuat didasari oleh pemikiran yang menggunakan sifat dan bentuk matriks dan juga suatu program untuk menganalisa dan melakukan komputasi numerik. Awal mulanya, program ini merupakan bagian dari proyek LINPACK dan EISPACK, lalu dikembangkan lagi menggunkan bahasa FORTRAN [30].

Perkembangan MATLAB membuat MATLAB menjadi sebuah pemograman canggih yang didalamnya terdapat fungsi-fungsi *built-in* yang dapat digunakan untuk mengolah sinyal, aljabar linier, dan kalkulasi matematis lainnya. MATLAB juga berisi beberapa toolbox yang terdiri dari fungsi-fungsi tambahan untuk aplikasi khusus. MATLAB memiliki sifat *extensible*, pemogram ini tidak terlalu rumit jika anda telah mempunyai pengalaman dengan bahasa lainnya eperti C++, PASCAL, atau FORTRAN. Biasanya MATLAB banyak dipakai pada Matematika dan Komputansi, Pengembangan dan Algoritma, Pemrograman modeling,

simulasi, dan pembuatan *prototype*, Analisa Data, eksplorasi dan visualisasi, Analisis numerik dan *statistic*, dan Pengembangan aplikasi teknik.

2.17. Confusion Matrix

Confusion Matrix merupakan sebuah tabel yang menggambar jumlah data uji yang terklasifikasi dengan benar dan jumlah data uji yang terklasifikasi dengan salah. *Confusion Matrix* juga adalah matriks yang didalamnya terdapat jumlah dari *True Positive, True Negative, False Positive,* dan *False Negative* [25]. TP dan TN merupakan hasil klasifikasi yang benar sedangkan FP dan FN merupakan hasil klasifikasi yang salah. Adapun gambaran *confusion matrix* dapat dilihat pada gambar berikut:



Gambar 2.15 *Confusion Matrix*

a. Akurasi merupakan persentase dari total hasil prediksi yang bernilai true dari semua data. Adapun untuk melakukan perhitungan tingkat akurasi menggunakan persamaan berikut.

$$Akurasi = \frac{TP}{TP + FN + FP + TN} \text{ x } 100\% \qquad (2.20)$$

Keterangan:

TP : *True Positive*

TN : *True Negative*

FP : *False Positive*

FN : *False Negative*

b. *Precision* adalah nilai ketepatan dari hasil suatu model. Persamaannya menggunakan perbandingan antara hasil *true positive* dengan total data dengan label *positive*. Adapun untuk perhitungan *precision* menggunakan persamaan berikut [31].

$$Precision = \frac{TP}{TP + FP} \quad (2.21)$$

c. *Recall* adalah nilai kelengkapan dari sebuah model. Persamaan *recall* menggunakan perbandingan antara *true positive* terhadap total contoh yang benar-benar *positive*. Adapun untuk perhitungan *recall* menggunakan persamaan berikut [31].

$$Recall = \frac{TP}{TP + FN} \quad (2.22)$$

d. *F-measure* merupakan perhitungan untuk mencari nilai *harmonic mean* dari *precision* dan *recall*, Adapun untuk perhitungan *recall* menggunakan persamaan berikut [31].

$$F1\ Score = \frac{2 * precision * recall}{precision + recall} \quad (2.23)$$

Jika dalam suatu pengujian jumlah kelas yang diklasifikasikan memiliki lebih dari dua kelas, maka rumus untuk menghitung *presicion*, dan *recall* adalah sebagai berikut [32] :

$$Precision = \frac{\sum_i^L \frac{TP_i}{TP_i + FP_i}}{L} X\ 100\% \quad (2.24)$$

$$Recall = \frac{\sum_i^L \frac{TP_i}{TP_i + FN_i}}{L} X\ 100\% \quad (2.25)$$

Keterangan

L : Jumlah Keseluruhan Data Yang Diuji/Jumlah Kelas

2.18. *Classification Accuracy*

Classification Accuracy merupakan sebuah metode yang digunakan untuk menghitung jumlah kebenaran suatu *classifier* dengan menampilkan persentase dari data yang berhasil di klasifikasikan [33]. Adapun untuk perhitungannya menggunakan persamaan berikut:

$$Accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions\ made}\ x\ 100\% \quad (2.26)$$

Keterangan

Number of correct Prediction : Jumlah data yang berhasil di kelompokkan dari *classifier*

Total number of predictions made : Total data yang digunakan pada proses klasifikasi

Jika *classifier* yang diuji memiliki jumlah lebih dari dua, maka untuk mencari rata-rata dari setiap *classifier* menggunakan persamaan sebagai berikut:

$$Classifier = \frac{\sum_{i=1}^{n} x_i}{n}\ x\ 100\% \quad (2.27)$$

Keterangan

x_i : Nilai akurasi dari *classifier* ke - i

n : Jumlah total yang data uji.


UNIVERSITAS KOMPUTER INDONESIA
UNIKOM