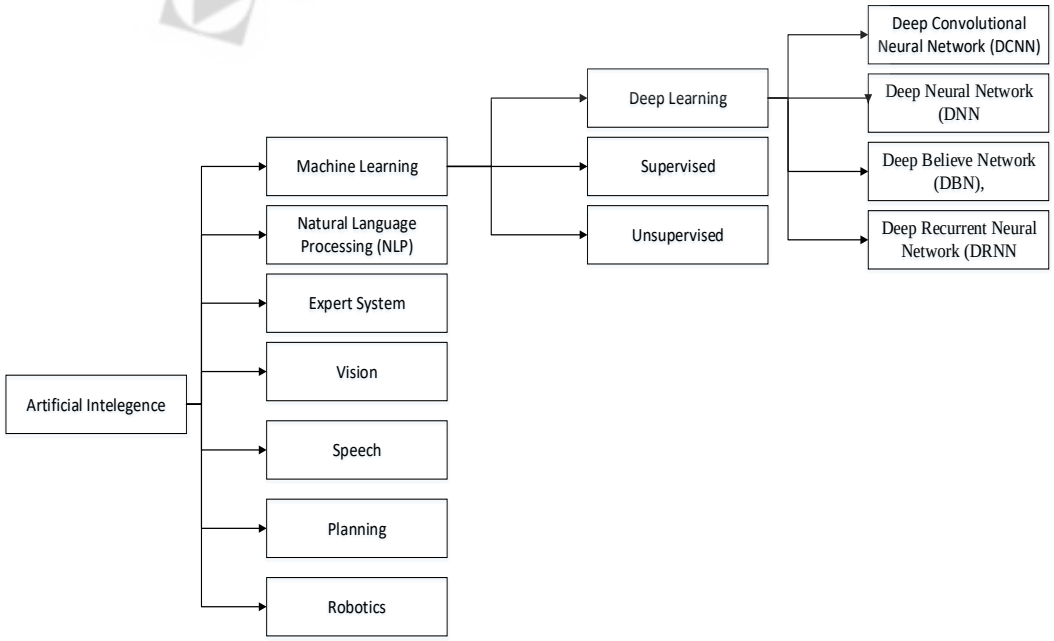
BAB II
LANDASAN TEORI

Pada bab ini dijelaskan tentang teori-teori yang rnendukung dalam melakukan penelitian yang dilakukan serta menjelaskan konsep-konsep yang diperlukan untuk memecahkan permasalahan yang ada dalam penelitian.

2.1 Kecerdasan Buatan

Menurut Rich dan knight (1991), "kecerdasan buatan merupakan sebuah studi tentang bagaimana membuat computer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia."

Tujuan kecerdasan buatan adalah menciptakan program yang mampu untuk belajar, sama halnya dengan proses manusia belajar yaitu mampu memperbaharui parameter dimana parameter tersebut kurang-lebih merepresentasikan pengetahuan mesin. Keterkaitan antara kecerdasan buatan, *Machine Learning*, dan Metode yang digunakan dalam penelitian yaitu *Deep Convolutional Neural Network* direpresentasikan dalam bentuk diagram pada gambar 2.1



Gambar 2. 1 Keterkaitan antara Kecerdasan Buatan dan *Deep Convolutional Neural Network*

2.2 *Machine Learning*

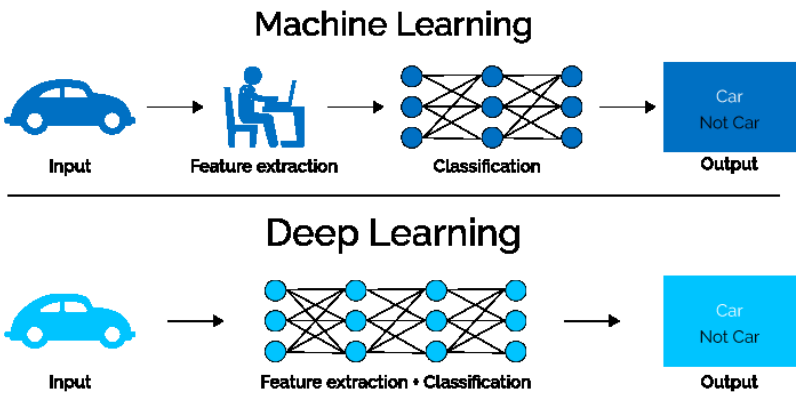
Machine Learning atau Pembelajaran mesin adalah salah satu metode kecerdasan buatan yang memberikan sistem kemampuan untuk belajar dan meningkatkan secara otomatis dari pengalaman tanpa diprogram secara eksplisit. Proses pembelajaran dimulai dengan observasi atau data, seperti contoh, pengalaman langsung, atau instruksi, guna mencari pola dalam data dan mengambil keputusan yang lebih baik di masa depan berdasarkan contoh yang diberikan. Tujuan utamanya adalah untuk memungkinkan komputer belajar secara otomatis tanpa campur tangan manusia dan menyesuaikan tindakan yang sesuai.

Klasifikasi adalah metode dalam *machine learning* yang digunakan oleh mesin untuk memilah atau mengklasifikasikan obyek berdasarkan ciri tertentu sebagaimana manusia mencoba membedakan benda satu dengan yang lain. Sedangkan prediksi atau regresi digunakan oleh mesin untuk menerka keluaran dari suatu data masukan berdasarkan data yang sudah dipelajari dalam training. Metode *machine learning* yang paling populer yaitu Sistem Pengambil Keputusan, *Support Vector Machine* (SVM) dan *Neural Network*. (Abu Ahmad, 2017)

Teknik dalam *Neural Network* salah satunya adalah *Deep Learning* (DL) yang digunakan untuk mempercepat proses pembelajaran dalam *Neural Network* yang menggunakan lapis banyak atau lebih dari 7 lapis. Dengan adanya *Deep Learning*, waktu yang dibutuhkan untuk training akan semakin sedikit karena masalah hilangnya gradien pada propagasi balik akan semakin rendah. (Bengio, Y. 1994)

2.3 *Deep Learning*

Deep Learning merupakan teknik dalam *machine learning* yang memiliki arsitektur yang lebih "dalam" dibanding dengan teknik *machine learning* lainnya dalam menyelesaikan masalah prediksi, maupun klasifikasi. Perbedaan *machine learning* dengan *deep learning* dapat dilihat secara visual pada gambar 2.2.



Gambar 2. 2 Perbedaan antara machine learning dengan deep learning

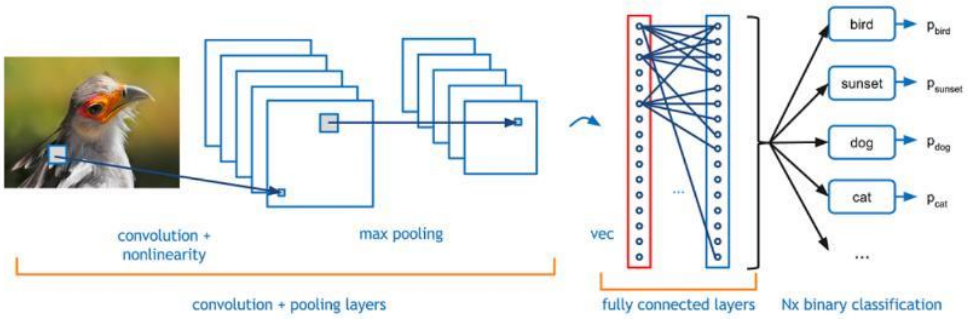
(Sumber : B.Rajalingam et al, 2018)

Perbedaan *machine learning* dengan *deep learning* terletak pada *feature extraction* / ekstraksi fitur, pada *machine learning* dilakukan secara terpisah dengan proses klasifikasinya, namun pada deep learning proses tersebut dapat disatukan.

Beberapa arsitektur pada perspektif *deep learning* adalah *Deep Neural Network (DNN), Deep Believe Network (DBN), Deep Convolutional Neural Network (DCNN), dan Deep Recurrent Neural Network (DRNN).* DNN sesuai untuk data berjenis multivariasi dengan banyak input neuron, yang secara umumnya adalah proses DBN yang kemudian dilakukan *feed-forward* satu arah terhadap DBN tersebut. Sedangkan *Deep Convolutional Neural Network* DCNN menggunakan *max and pool layer*, serta dense layer yang lebih sesuai dengan klasfikasi citra gambar. Untuk pengenalan *text, speech, language*, atau data dengan *tipe time series* maka DRNN akan lebih sesuai (Nikoskinen, 2015).

2.4 *Convolutional Neural Network*

Convolutional Neural Network merupakan salah satu jenis *neural network* yang biasanya digunakan dalam pengolahan citra digital. Konvolusi atau biasa yang disebut dengan convolution adalah matriks yang memiiki fungsi melakukan filter pada gambar (Ludwig, 2012). *Convolutional Neural Network* memiliki beberapa layer yang difungsikan untuk melakukan filter pada setiap prosesnya. Prosesnya disebut dengan proses *training*. Pada proses *training* terdapat 3 tahapan yaitu *Convolutional layer, Pooling layer,* dan *Fully connected layer*.Gambar 2.3

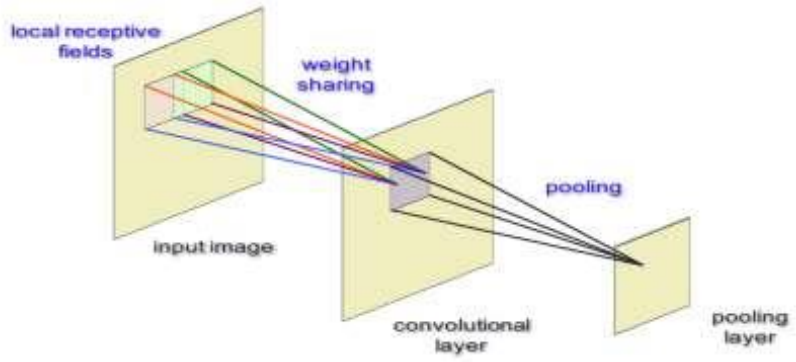


Gambar 2. 3 Ilustrasi dari *Convolutional Neural Network*

2.4.1 *Convolutional Layer*

Seluruh data yang menyentuh lapisan konvolusional akan mengalami proses konvolusi. lapisan akan mengkonversi setiap filter ke seluruh bagian data masukan dan menghasikan sebuah *activation map* atau *feature map*. Filter yang terdapat pada *Convolutional Layer* memiliki panjang, tinggi dan tebal sesuai dengan channel data masukan. Setiap filter akan mengalami pergeseran dan operasi "dot" antara data masukan dan nilai dari filter. Lapisan konvolutional secara signifikan mengalami kompleksitas model melalui optimalisasi outputnya. (O'Sheal and Nash, 2015) ilustrasi lapisan konvolusional terdapat pada gambar 2.4

(sumber : Srikanth Tammina, 2019)



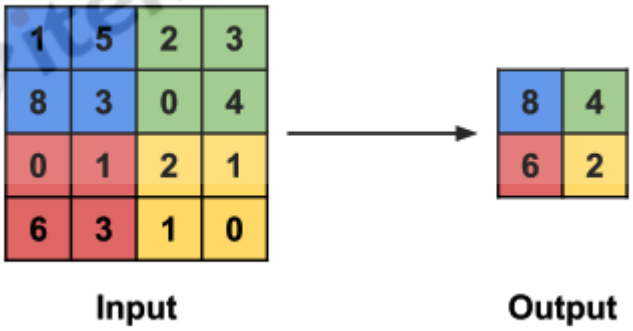
Gambar 2. 4 *Convolutional Layer*

(Sumber : B.Rajalingam et al, 2018)

2.4.2 *Pooling layer*

Polling layer biasanya berada setelah *conv. layer*. Pada prinsipnya *pooling layer* terdiri dari sebuah filter dengan ukuran dan *stride* tertentu yang bergeser pada seluruh area *feature map*. *Pooling* yang biasa digunakan adalah *Max Pooling* dan *Average Pooling*. Tujuan dari penggunaan *pooling layer* adalah mengurangi dimensi dari feature map (*downsampling*), sehingga mempercepat komputasi karena parameter yang harus di update semakin sedikit dan mengatasi *overfitting* (Qalbiyatul Lina, 2019).

Hal terpenting dalam pembuatan model CNN adalah dengan memilih banyak jenis lapisan *pooling*. Hal ini dapat menguntungkan kinerja model (Lee, Gallagher, & Tu, 2015). Lapisan *pooling* bekerja di setiap tumpukan *feature map* dan mengurangi ukurannya. Bentuk lapisan *pooling* yang paling umum adalah dengan menggunakan filter berukuran 2x2 yang diaplikasikan dengan langkah sebanyak 2 dan kemudian beroperasi pada setiap irisan dari input. Pada gambar 2.5 merupakan contoh ilustrasi dari penggunaan *Max Pooling.*



Gambar 2. 5 Ilustrasi *Max Pooling*

(Sumber : Thomas da Silva Paula, 2017)

2.4.3 ReLu

Rectified Linear Units (ReLU) untuk fungsi aktivasi yang diperkenalkan oleh Geoffrey Hinton dan Vinod Nair dan digunakan pada *neural network*, digunakan untuk mengubah nilai x menjadi 0 jika nilai x tersebut bernilai negatif, sedangkan

sebaliknya untuk nilai x tetap dipertahankan apabila nilai tidak kurang dari 0 (Agarap, 2018). Operasi ReLU dirumuskan dengan persamaan sebagai berikut

$$f(xi) = \max(0, xi)\{^{xi, i\ f\ xi \geq 0}_{0, i\ f\ xi < 0} \quad (2.1)$$

Keterangan :

$f(xi) =$ nilai dari ReLU activation

Xi = nilai dari matriks

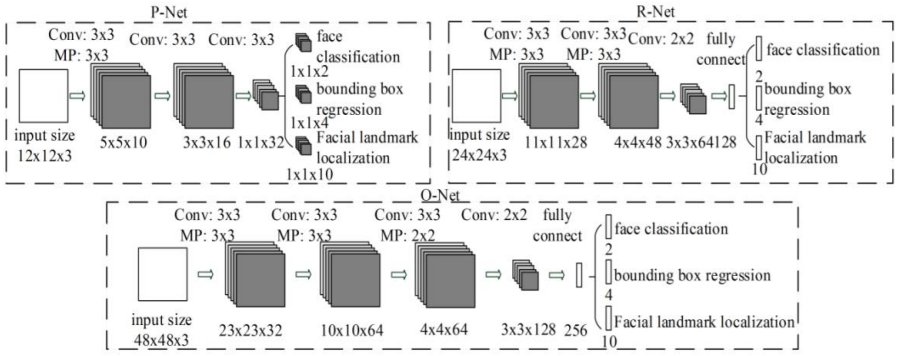
2.4.4 *Stride*

Stride adalah parameter yang menentukan berapa jumlah pergeseran *filter*. Jika nilai *stride* adalah 1, maka *convolutional filter* akan bergeser sebanyak 1 piksel secara horizontal lalu vertikal. Semakin kecil *stride* maka akan semakin detail informasi yang kita dapatkan dari sebuah input, namun membutuhkan komputasi yang lebih jika dibandingkan dengan *stride* yang besar. Namun perlu diperhatikan bahwa dengan menggunakan *stride* yang kecil kita tidak selalu akan mendapatkan performa yang bagus. (Qalbiyatul Lina, 2019).

2.5 Multi-task Cascaded Convolutional Neural Network (MTCNN).

Multi-task Cascaded Convolutional Neural Network (MTCNN) merupakan salah satu variasi dari metode *Deep Convolutional Neural Network* yang biasa digunakan untuk mengenali atau mendeteksi wajah, MTCNN terdiri dari 3 jaringan terpisah: P-Net, R-Net, dan O-Net.

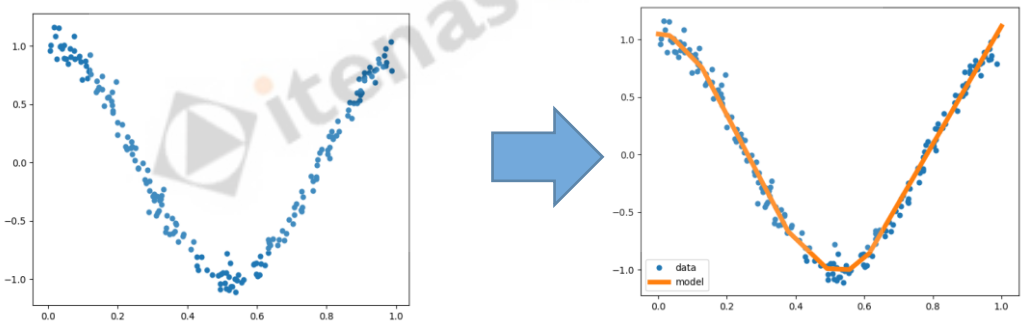
Di jaringan P-Net, untuk setiap gambar, kernel 12x12 menelusuri gambar, mencari wajah. Kernel 12x12, perlahan-lahan bergerak melintas secara horizontal sampai sudut bawah gambar, mencari wajah. Begitupun di jaringan R-Net dan O-Net melakukan penelusuran mendeteksi wajah yang membedakan di setiap jaringan ada pada bagian kernel, yaitu P-Net menggunakan kernel 12x12, R-Net menggunakan kernel 24x24 dan O-Net menggunakan kernel 48x48. Seperti yang direpresentasikan pada Gambar 2.5



Gambar 2. 6 Arsitektur MTCNN

(sumber : Kaipeng Zhang et al., 2016)

Dalam masing-masing 12x12 kernel ini, dibagi menjadi tiga tahapan konvolusi dijalankan dengan 3x3 kernel. Setelah setiap lapisan konvolusi selesai dijalankan, lapisan Parametric Rectified Linear Unit (pReLU) akan menghilangkan vanishing gradient dengan cara menerapkan fungsi aktivasi element. Contoh implementasi pReLU dapat dilihat pada gambar 2.6

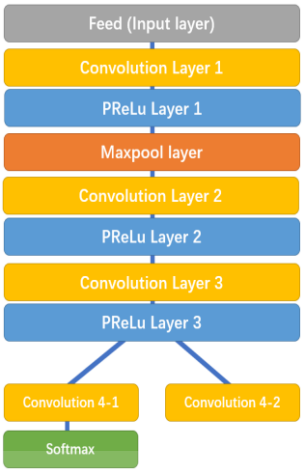


Gambar 2. 7 Ilustrasi penggunaan fungsi pReLU

(sumber : medium.com)

Setelah lapisan konvolusi ketiga, jaringan terbagi menjadi dua lapisan. Aktivasi dari lapisan ketiga dibagi ke dua lapisan konvolusi yang terpisah, dan lapisan *softmax* memberikan probabilitas desimal untuk setiap hasil, dan probabilitas bertambah hingga satu. Dalam kasus ini, ia menghasilkan dua probabilitas, yaitu probabilitas bahwa ada wajah di daerah tersebut dan probabilitas bahwa tidak ada wajah. Pada gambar 2.8 merupakan representasi dari cara kerja

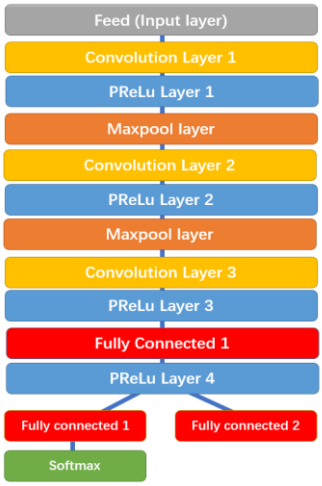
Multi-task Cascaded Convolutional Neural Network (MTCNN) pada bagian jaringan P-Net



Gambar 2. 8 Representasi cara kerja MTCNN pada jaringan P-Net

(sumber : Thomas da Silva Paula, 2017)

Konvolusi 4-1 menghasilkan probabilitas suatu wajah berada di setiap kotak pembatas (*bounding box*), dan konvolusi 4–2 menghasilkan koordinat dari *bounding box*. R-Net memiliki struktur yang serupa, tetapi dengan lebih banyak lapisan. Menggunakan vektor *bounding box* P-Net sebagai inputnya, dan menyempurnakan koordinatnya. Gambar 2.9 menunjukan bagaimana cara kerja MTCNN pada jaringan R-Net

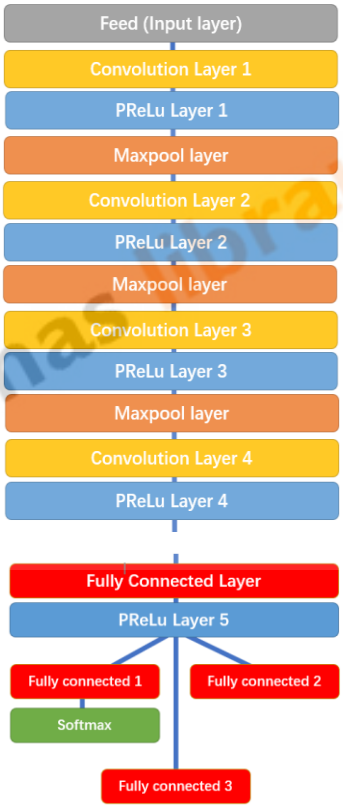


Gambar 2. 9 Representasi cara kerja MTCNN pada jaringan R-Net

(sumber : Thomas da Silva Paula, 2017)

Demikian pula, R-Net terbagi menjadi dua lapisan pada ujungnya, memberikan dua output : yaitu *Fully Connected* 1 berisi probabilitas suatu wajah berada di setiap kotak pembatas (*bounding box*) dan *Fully Connected* 2 berisi koordinat kotak pembatas (*bounding box*) yang baru dan menambah tingkat kepastian deteksi pada setiap kotak pembatas (*bounding box*).

Jaringan terakhir yaitu, O-Net mengambil kotak pembatas (*bounding box)* R-Net sebagai input dan bertugas menandai koordinat landmark wajah. Pada gambar 2.10 merupakan representasi dari cara kerja *Multi-task Cascaded Convolutional Neural Network* (MTCNN) pada bagian jaringan O-Net



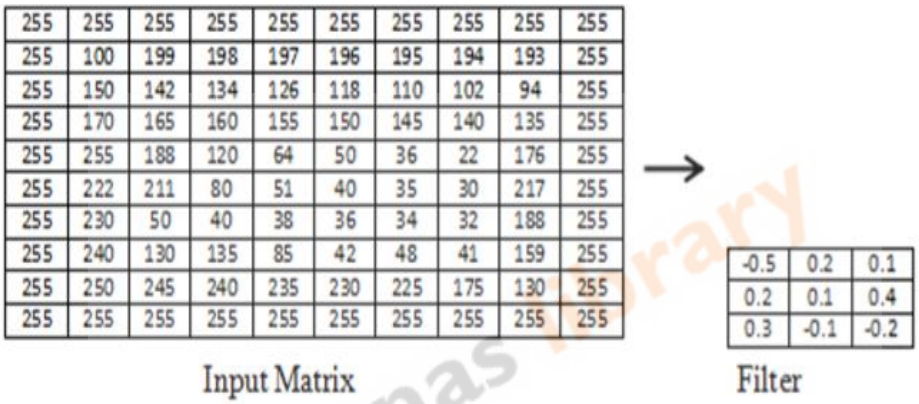
Gambar 2. 10 Representasi cara kerja MTCNN pada jaringan O-Net

(sumber : Thomas da Silva Paula, 2017)

O-Net terbagi menjadi tiga lapisan diujungnya, memberikan tiga output berbeda: *Fully Connected* 1 yaitu probabilitas wajah berada di dalam *bounding box*, *Fully Connected* 2 yaitu koordinat kotak pembatas (*bounding box*), dan *Fully Connected* 3 yaitu koordinat landmark wajah (lokasi mata, hidung, dan mulut).

2.6 Studi Kasus

Contoh proses konvolusi dengan input berupa citra satu channel yang diilustrasikan Pada gambar 2.11, sebuah citra berukuran 10x10 piksel direpresentasikan sebagai matriks. Matriks awal diproses dengan dua layer konvolusi untuk mendapatkan *feature map*. Pada layer konvolusi pertama, filter yang digunakan berukuran 3x3 dengan bobot yang telah ditentukan. Hasil dari konvolusi pertama berupa matriks dengan ukuran 9x9. Diilustrasikan pada Gambar 2.11 dan 2.12



255	255	255	255	255	255	255	255	255	255
255	100	199	198	197	196	195	194	193	255
255	150	142	134	126	118	110	102	94	255
255	170	165	160	155	150	145	140	135	255
255	255	188	120	64	50	36	22	176	255
255	222	211	80	51	40	35	30	217	255
255	230	50	40	38	36	34	32	188	255
255	240	130	135	85	42	48	41	159	255
255	250	245	240	235	230	225	175	130	255
255	255	255	255	255	255	255	255	255	255

Input Matrix

-0.5	0.2	0.1
0.2	0.1	0.4
0.3	-0.1	-0.2

Filter

Gambar 2. 11 Ilustrasi Input Matrix

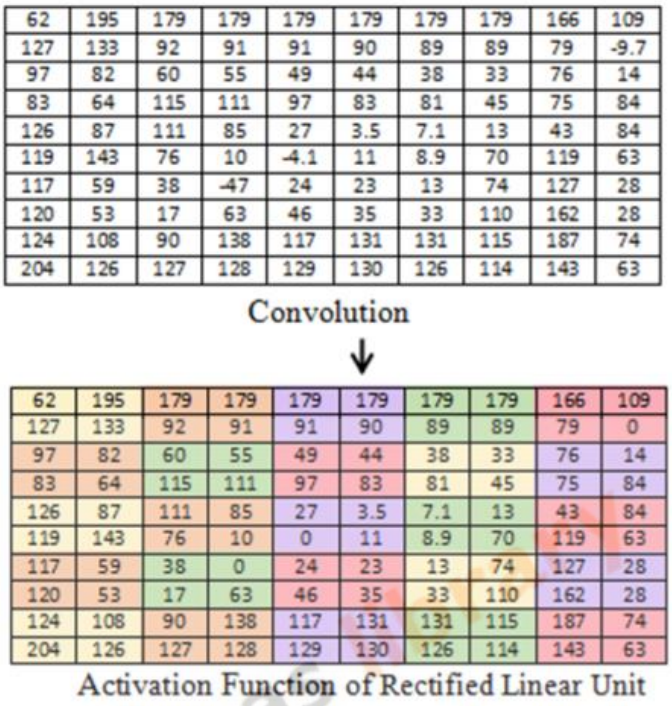
62	195	179	179	179	179	179	179	166	109
127	133	92	91	91	90	89	89	79	-9.7
97	82	60	55	49	44	38	33	76	14
83	64	115	111	97	83	81	45	75	84
126	87	111	85	27	3.5	7.1	13	43	84
119	143	76	10	-4.1	11	8.9	70	119	63
117	59	38	-47	24	23	13	74	127	28
120	53	17	63	46	35	33	110	162	28
124	108	90	138	117	131	131	115	187	74
204	126	127	128	129	130	126	114	143	63

Convolution

Gambar 2. 12 Hasil Convolution dengan Filter

Setelah melalui proses konvolusi, fungsi aktivasi akan diterapkan pada hasil konvolusi. Fungsi aktivasi yang digunakan adalah reLu yaitu mengubah matriks yang bernilai negatif menjadi nol. Output dari fungsi reLu kemudian dilakukan *pooling* dengan filter berukuran 2x2 dan stride dua piksel. Sebelum melakukan

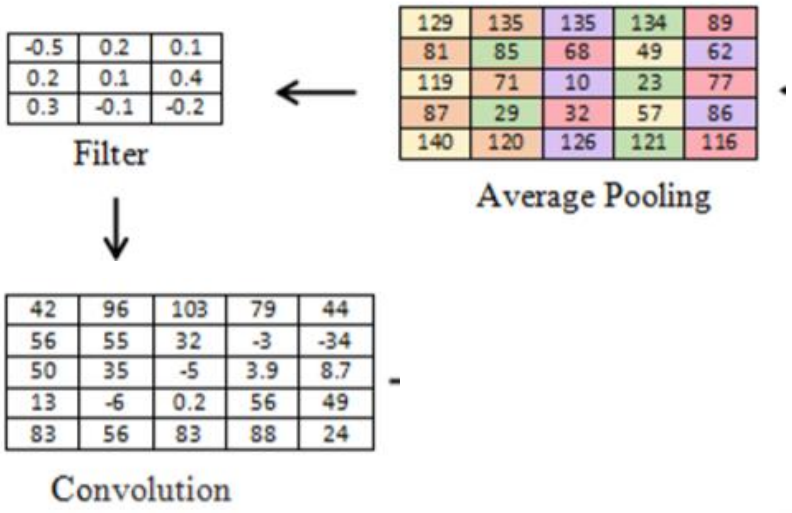
pooling, dapat digunakan *zero padding* sehingga matriks hasil pooling berukuran 5x5. Diilustrasikan pada gambar 2.13 dan 2.14



Gambar 2. 13 Fungsi aktivasi ReLu diterapkan

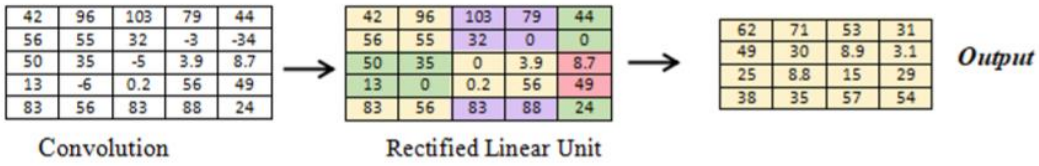
Gambar 2. 14 Proses Average Pooling

Matriks ini kemudian melalui tahap konvolusi kedua dengan ukuran filter sama seperti sebelumnya, tetapi dengan bobot yang berbeda, ukuran tidak harus sama dengan konvolusi tahap pertama dan merupakan parameter yang bisa dioptimalkan. Sementara bobot matriks merupakan nilai yang sudah didapatkan melalui proses *training*. Diilustrasikan dengan gambar 2.15



Gambar 2. 15 Proses Convolutional kedua

Output dari proses konvolusi tahap kedua diterapkan lagi fungsi aktivasi yang sama, yaitu reLu. *Pooling* yang dikenakan berukuran 2x2 dengan *stride* satu, sehingga menghasilkan matriks dengan ukuran 4x4. Proses konvolusi bisa dilanjutkan sesuai dengan matriks akhir yang diinginkan, jika konvolusi dihentikan sampai tahap kedua, maka matriks berukuran 4x4 tersebut menjadi input bagi *neural network*. Jika filter yang digunakan sejumlah n, maka input bagi *neural network* adalah nx4x4 nodes. Proses konvolusi kedua sampai dengan output diilustrasikan pada gambar 2.16



Gambar 2. 16 Proses output *Convolution Layer*

2.7 *Confusion matrix*

Confusion matrix adalah suatu metode yang biasanya digunakan untuk melakukan perhitungan akurasi pada konsep data mining atau Sistem Pendukung Keputusan. Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat 4 (empat) istilah sebagai representasi hasil proses klasifikasi. Keempat istilah tersebut

adalah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN).

Ilustrasi hubungan antar atribut direpresentasikan pada tabel 2.1

Tabel 2. 1 Hubungan antar *Confusion Matrix*

	Positif	Negatif
True	True Positif (TP)	False Positif (FP)
False	False Negatif (FN)	True Negatif (TN)

Jika dalam kasus pendeteksian wajah maka akan menjadi seperti berikut

- True Positif (TP) adalah gambar wajah yang benar terdeteksi sebagai wajah
- False Positif (FP) adalah gambar bukan wajah tetapi dideteksi sebagai wajah
- False Negatif (FN) adalah gambar wajah yang tidak terdeteksi sebagai wajah
- True Negatif (TN) adalah gambar bukan wajah dan tidak terdeteksi sebagai wajah.

Precision atau nilai *confidence* merupakan ketepatan deteksi yang menyatakan seberapa besar deteksi dengan benar dapat diraih, yang dapat dirumuskan sebagai berikut :

$$precission = \frac{TP}{TP+FP} \quad (2.2)$$

Recall atau *sensivity* merupakan wajah yang terdeteksi dengan benar dibagi wajah yang dideteksi dengan benar ditambah wajah yang tidak terdeteksi atau dapat diambil diartikan juga wajah yang terdeteksi dengan benar dibagi jumlah target wajah yang ada pada citra. Dirumuskan sebagai berikut

$$recall = \frac{TP}{TP+FN} \quad (2.3)$$

Sedangkan untuk mengukur akurasi sebuah sistem dengan menggunakan metode *Confusion matrix* maka dapat dirumuskan sebagai berikut :

$$accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (2.4)$$

Untuk mengukur kinerja dari sebuah sistem menggunakan *f-measure* yang dirumuskan sebagai berikut :

$$F\ Measure: 2\ x\ \frac{(Precision\ x\ Recall)}{(Precision+Recall)} \quad (2.5)$$