BAB 2
DASAR TEORI

2.1 KAJIAN PUSTAKA

Penelitian [3] meneliti mengenai pengenalan pola karakter pelat nomor kendaraan menggunakan algoritma *Momentum Backpropagation Neural Network*. Sebelum citra karakter masuk ke tahap pengenalan, akan digunakan *Haar Wavelet* untuk mereduksi dan menyeragamkan dimensi citra. Selanjutnya citra karakter dari langkah sebelumnya akan dikenali menggunakan Jaringan Saraf Tiruan Backpropragation. Jaringan Saraf Tiruan adalah sekumpulan node yang saling terhubung, serupa dengan jaringan neuron yang luas di otak. Hasilnya menunjukkan dari total 276 karakter yang terdiri dari huruf dan angka, sistem mampu mengenali 268 karakter diantaranya. Akurasi sistem yang didapatkan pada penelitian ini mencapai 97,01%.

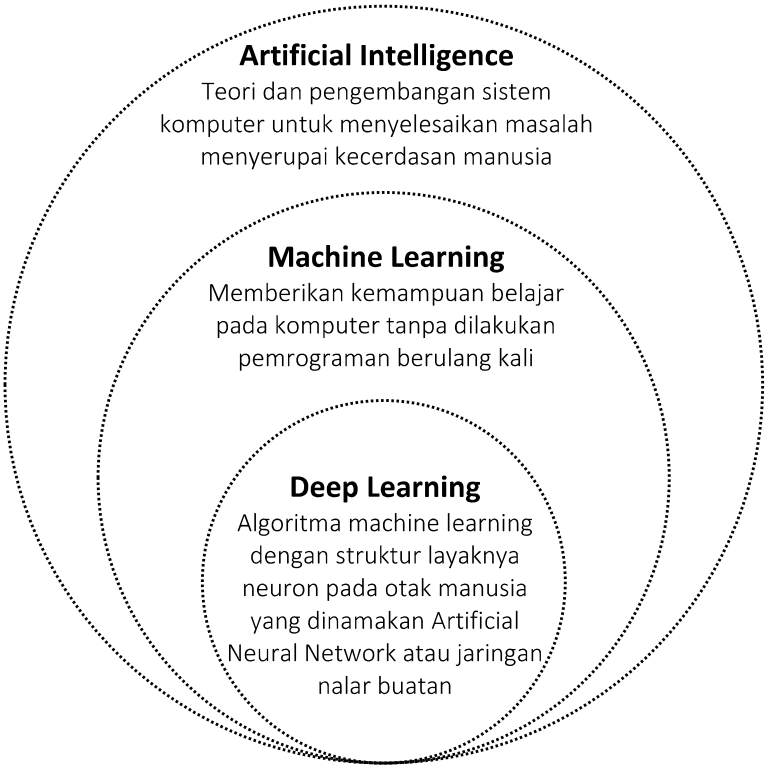
Penelitian [4] membahas mengenai unjuk kerja identifikasi pelat nomor kendaraan secara *off-line* berbasis pengolahan citra dan jaringan syaraf tiruan. Sama seperti penelitian sebelumnya, digunakan metode jaringan syaraf tiruan backpropragation. Data masukan merupakan citra plat utuh yang sudah dilakukan *cropping*. Digunakan 220 neuron *input*, dengan satu *hidden layer* dan 40 buah neuron, 37 neuron *output*. Hasilnya sistem dapat mengenali sebanyak 96% data citra pelat nomor yang diikutkan dalam tahap pelatihan dan 90% dari data citra pelat nomor yang tidak diikutkan dalam tahap pelatihan.

Penelitian [5] membahas mengenai unjukkerja sistem identifikasi pelat nomor kendaraan menggunakan metode *Robert Filter* dan *Framming Image* berbasis pengolahan citra digital. Metode ini memindai warna hitam hasil gambar pada Robert secara vertikal, maka akan memunculkan nilai-nilai dimana terdapat warna hitam diwakilkan dengan nilai 0, sedangkan warna putih diwakilkan dengan nilai 255. Didapatkan nilai penyimpangan rata-rata yang paling kecil (0,21%) dari pengujian dengan jarak 100cm dengan jarak sudut 0 derajat.

2.2 DASAR TEORI

2.2.1 Konsep dasar *Deep Learning*

Deep Learning merupakan sub-bidang *Machine Learning* yang algoritmanya terinspirasi dari struktur otak manusia. Struktur tersebut dinamakan Artificial *Neural Network*s atau disingkat ANN. Pada dasarnya, ia merupakan jaringan saraf yang memiliki tiga atau lebih lapisan ANN. Ia mampu belajar dan beradaptasi terhadap sejumlah besar data serta menyelesaikan berbagai permasalahan yang sulit diselesaikan dengan algoritma *Machine Learning* lainnya.



Gambar 2.1 *Deep Learning* merupakan bagian kompleks dari *Machine Learning* [6]

Deep Learning merupakan salah satu bidang dari *Machine Learning* yang memanfaatkan jaringan syaraf tiruan untuk mengimplementasikan permasalahan dengan dataset yang besar. Arsitektur *Deep Learning* sangat baik untuk model pembelajaran terbimbing (*Supervised Learning)*. Dengan menambahkan lebih banyak lapisan maka model pembelajaran tersebut bisa mewakili data citra berlabel dengan lebih baik. Pada *Machine Learning* terdapat teknik ekstraksi fitur dari data pelatihan dan algoritma pembelajaran khusus untuk mengklasifikasi citra maupun mengenali suara. Namun, metode ini masih memiliki beberapa kekurangan baik

dalam hal kecepatan dan akurasi. Aplikasi konsep *Deep Artificial Neural Network* dapat digunakan pada algoritma *Machine Learning* yang sudah ada sehingga komputer sekarang bisa belajar dengan kecepatan, akurasi, dan skala yang besar [7].

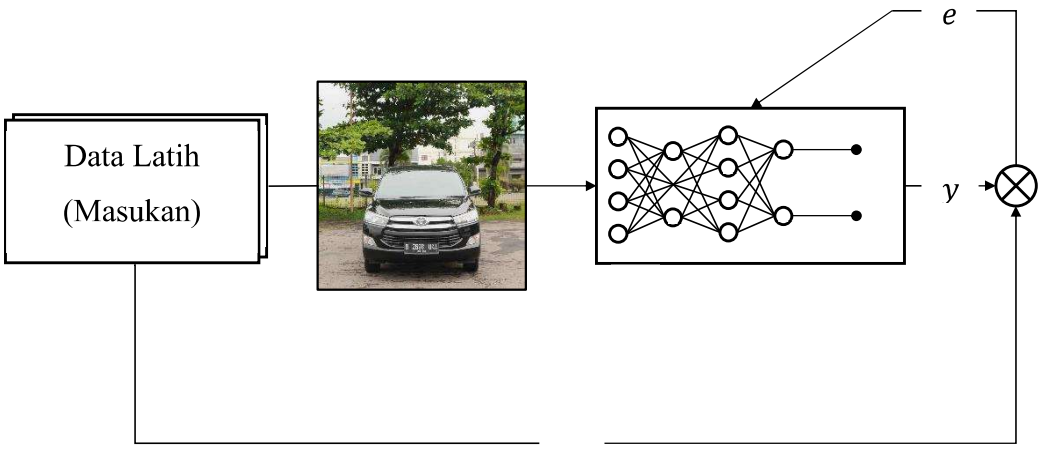
2.2.2 Arsitektur *Convolutional Neural Network* (CNN)

Algoritma CNN bukan hanya sistem syaraf buatan mendalam yang mempunyai banyak *hidden layer*. Algoritma ini merupakan *deep Network* yang prinsip kerjanya menyerupai bagaimana *visual cortex* di dalam otak manusia mengenali gambar. Oleh karena itu, ahli dalam bidang *Neural Network* pun terkadang kesulitan dalam memahami konsep ini saat pertama kali. Ini menunjukkan betapa berbedanya konsep dan operasi dari *Neural Network* sebelumnya [8].

Pada dasarnya, pengenalan citra merupakan konsep klasifikasi. Sebagai contoh, untuk membedakan antara citra kucing dengan citra anjing prinsip kerjanya sama dengan mengklasifikasikan kucing dan anjing ke dalam kelas. Sama halnya dengan pengenalan huruf; mengenali huruf dari sebuah citra sama saja dengan mengklasifikasikan citra ke dalam kelas huruf. Oleh karena itu, *layer* keluaran CNN umumnya menggunkan banyak kelas klasifikasi *Neural Network.*

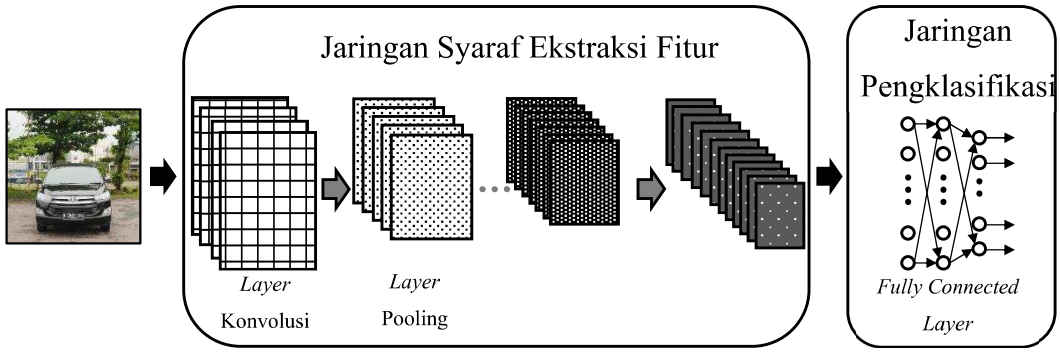
Namun, penggunaan gambar asli untuk pengenalan citra memberikan hasil yang kurang baik, tanpa memperhatikan metode pengenalan; citra harus diproses ke dalam fitur kontras agar hasil yang didapatkan lebih maksimal. Dari alasan tersebut , berbagai macam teknik dalam ekstraksi fitur dari sebuah citra terus dikembangkan.

CNN memiliki pengekstraksi fitur di dalam proses pelatihan dan tidak merancangnya secara manual. Pengekstraksi fitur dalam CNN tersusun dari jaringan syaraf (*Neural Network*) khusus yang bobotnya ditentukan dalam proses pembelajaran. CNN mengubah ekstraksi fitur secara manual dan merancangnya ke dalam proses otomatis yang merupakan fitur dan keuntungan utamanya [9]. Gambar 2.1 menunjukkan konsep pelatihan dalam CNN.



Gambar 2.2 Pengekstraksi Fitur CNN tersusun dari jaringan syaraf khusus [10]

CNN dapat mengenali citra dengan baik apabila ekstraksi fitur dari jaringan syarafnya lebih mendalam (lebih banyak lapisan atau *layer*). CNN terdiri atas jaringan syaraf yang mengekstrak dan mengklasifikasikan fitur dari citra masukan. Gambar 2.2 menunjukkan arsitektur dari CNN.



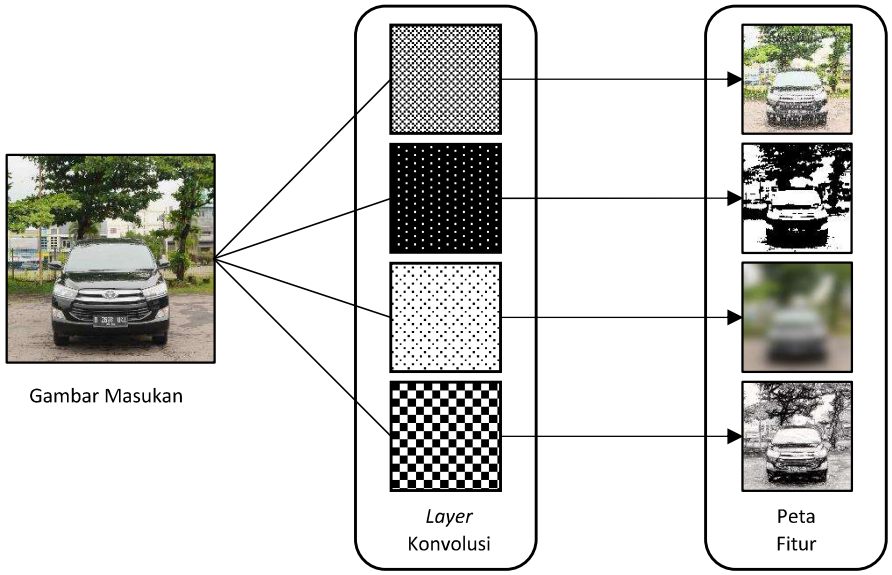
Gambar 2.3 Arsitektur CNN [10]

Dari gambar 2.2 dapat dijelaskan bahwa citra masukan masuk ke dalam jaringan ekstraksi fitur. Sinyal hasil ekstraksi fitur kemudian masuk ke dalam jaringan syaraf klasifikasi. Kemudian jaringan syaraf klasifikasi bekerja berdasarkan fitur dari citra dan menghasilkan keluaran. Jaringan syaraf ekstraksi fitur terdiri dari tumpukan lapisan konvolusi dan *pooling layer***.** Sesuai Namanya, lapisan konvolusi mengonversi citra menggunakan operasi konvolusi. Lapisan konvolusi juga bisa disebut kumpulan filter digital. *Pooling Layer* menggabungkan piksel yang berdekatan menjadi satu buah piksel sehingga memperkecil dimensi

citra. Karena perhatian utama dalam CNN adalah gambar, operasi konvolusi dan *pooling layer* secara konsep merupakan perhitungan dalam bidang dua dimensi. Hal tersebut merupakan perbedaan mendasar antara CNN dengan jaringan syaraf tiruan lain [10].

2.2.3 Lapisan Konvolusi (*Convolutional Layer*)

Layer konvolusi menghasilkan citra baru yang dimakan peta fitur (*feature maps*). Peta fitur berfokus pada fitur unik dari sebuah citra masukan yang membuat cara kerja *layer* konvolusi berbeda dengan *layer* jaringan syaraf lain. *Layer* ini berisi filter atau bisa disebut kernel yang fungsinya untuk mengkonversi citra (*convolution filters*). Citra yang dimasukkan ke dalam *layer* kovolusi akan menghasilkan peta fitur [11]. Gambar 2.3 menunjukkan proses dalam *layer* konvolusi.



Gambar 2.4 Proses di dalam *layer* konvolusi [10]

Dari gambar 2.3 dapat dijelaskan bahwa gambar masukan akan dilakukan operasi konvolusi yang menghasilkan filter konvolusi atau kernel. Gambar yang sudah dilakukan konvolusi akan menghasilkan peta fitur setelah melewati fungsi aktivasi. *Layer* konvolusi menghasilkan jumlah peta fitur yang sama sesuai dengan jumlah filter konvolusi yang digunakan. Sebagai contoh, apabila jumlah *layer* konvolusi berisi empat buah filter, maka akan dihasilkan empat buah peta fitur juga.

Filter dalam *layer* konvolusi merupakan matriks dua dimensi. Nilai dalam matriks ditentukan dalam proses pelatihan. Kemudian, nilai tersebut dilatih secara berulang selama proses pelatihan. Aspek ini sama dengan proses *update* dalam jaringan syaraf buatan lainnya [12]. Sebagai contoh, diberikan gambar dengan ukuran 4 × 4 piksel yang ditunjukkan pada Gambar 2.4..

1	**1**	**1**	**3**
4	**6**	**4**	**8**
30	**0**	**1**	**5**
0	**2**	**2**	**4**

Gambar 2.5 Citra dengan ukuran 4 × 4 piksel [10]

Digunakan dua buah filter konvolusi dengan matriks ordo 2 × 2. Sebagai catatan, filter di dalam CNN ditentukan pada saat proses pelatihan dan tidak dibuat secara manual.

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Operasi konvolusi dimulai dari bagian kiri atas piksel. Piksel yang diambil memiliki ukuran yang sama dengan matriks filter konvolusi. Proses ini juga dinamakan dengan *Padding.* Gambar 2.5 menunjukkan proses konvolusi tahap pertama.

1	**1**	**1**	**3**
4	**6**	**4**	**8**
30	**0**	**1**	**5**
0	**2**	**2**	**4**

$$\times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} =$$

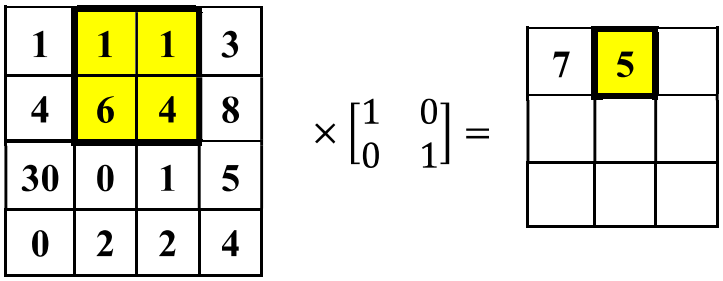
7		

Gambar 2.6 Proses konvolusi filter pertama dimulai dari bagian kiri atas piksel [10]

Operasi konvolusi merupakan *sum product* dari elemen yang memiliki posisi sama dengan matriks. Nilai 7 dalam gambar 2.5 didapatkan dari perhitungan 1 :

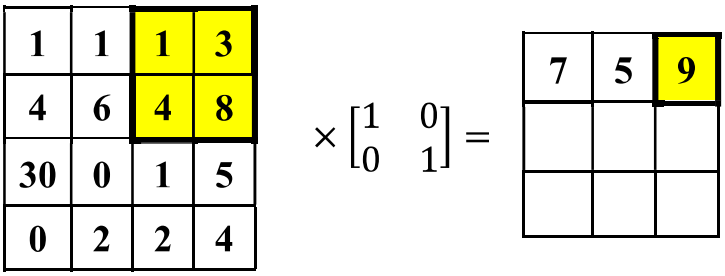
$$(1 \times 1) + (1 \times 0) + (4 \times 0) + (6 \times 1) = 7 \ldots (1)$$

Operasi konvolusi dilakukan pada piksel berikutnya. Gambar 2.6 menunjukkan proses konvolusi tahap kedua.



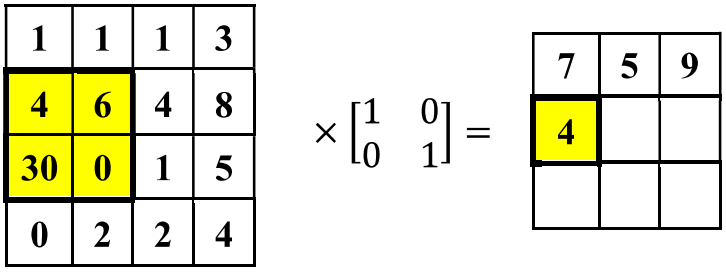
Gambar 2.7 Operasi konvolusi filter pertama tahap kedua [10]

Dengan cara yang sama, operasi konvolusi dilakukan pada piksel berikutnya. Konvolusi tahap ketiga ditunjukkan pada gambar 2.7.



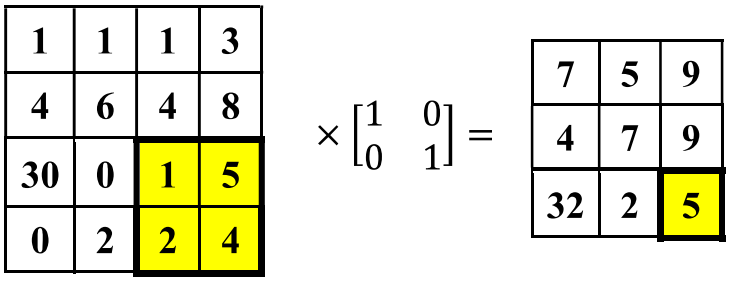
Gambar 2.8 Operasi konvolusi filter pertama tahap ketiga [10]

Jika baris pertama sudah terpenuhi, dilanjutkan dengan operasi konvolusi pada baris kedua dimulai dari bagian kiri piksel. Konvolusi tahap keempat ditunjukkan pada gambar 2.8.



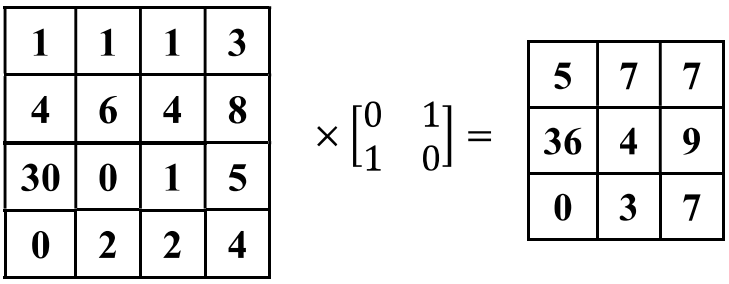
Gambar 2.9 Operasi konvolusi filter pertama tahap keempat [10]

Proses konvolusi dilakukan sampai dengan peta fitur dari filter yang diberikan terpenuhi seperti pada gambar 2.9.



Gambar 2.10 Peta fitur dari filter pertama sudah terpenuhi [10]

Dengan cara yang sama, operasi konvolusi filter kedua didapatkan hasil yang ditunjukkan pada gambar 2.10.



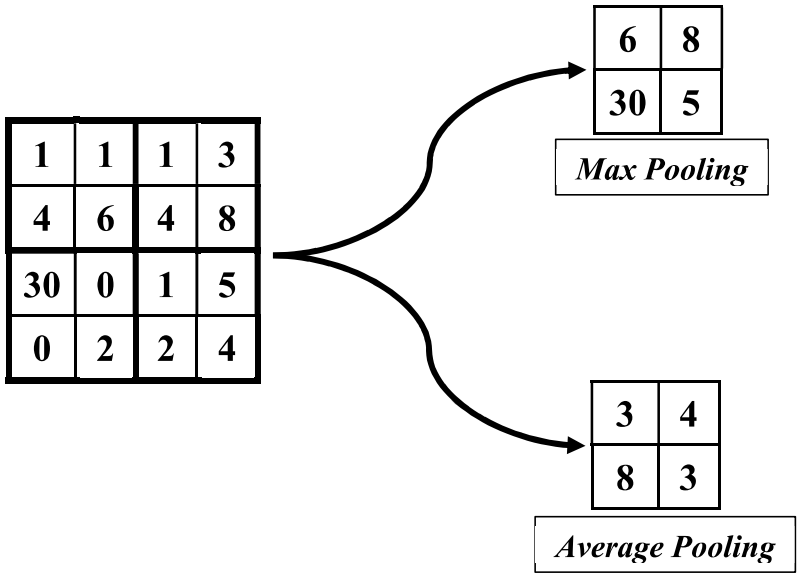
Gambar 2.11 Peta fitur dari filter kedua [10]

Sama dengan hasil operasi konvolusi filter pertama, nilai dari masing-masing peta fitur bergantung pada apakah matriks gambar cocok dengan filter konvolusi atau tidak.

2.2.4 *Pooling Layer*

Pooling layer bekerja dengan mereduksi ukuran spasial dari fitur konvolusi sehingga dapat mengurangi kinerja komputasi. Cara kerja dari *pooling layer* yaitu mengurangi dimensi dari peta fitur (*downsampling*) sehingga mempercepat proses komputasi karena parameter yang diperbarui semakin sedikit. *Pooling layer* juga dapat berfungsi untuk mengekstraksi fitur dominan sehingga proses pelatihan model lebih efektif. Ada dua jenis *pooling layer*, yaitu *max pooling* dan *average pooling*. *Max pooling* menampilkan nilai maksimum dari bagian gambar yang dicakup oleh kernel sedangkan *average pooling* menampilkan nilai rata-rata dari

gambar yang dicakup oleh kernel [13]. Contoh kasus *max pooling average pooling* ditunjukkan pada gambar 2.11.



Gambar 2.12 Contoh kasus *max pooling* dan *average pooling* [10]

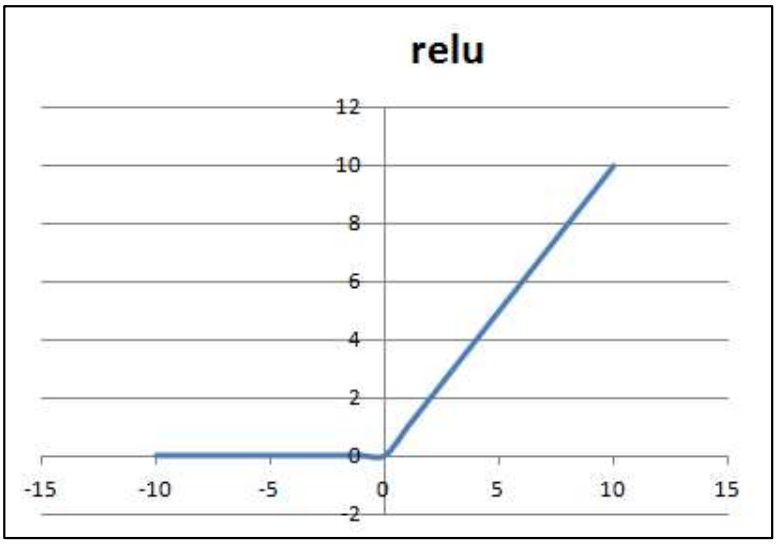
2.2.5 *Fully Connected Layer* atau *Dense Layer*

Dalam CNN, dibutuhkan *Fully connected layer* untuk transformasi data agar data dapat diklasifikasikan secara linear. Keluaran dari lapisan ini merupakan hasil komputasi perkalian matriks yang diikuti dengan *bias offset,* tidak lagi menggunakan operasi konvolusi. Setiap *neuron* memiliki koneksi penuh ke semua aktivasi dari lapisan sebelumnya sehingga lapisan ini dinamakan dengan *fully connected layer.*

Fungsi aktivasi atau *neuron* merupakan fungsi *non-linear* yang memungkinkan sebuah jaringan syaraf tiruan untuk menyelesaikan permasalahan *non-trivial.* Setiap *neuron* mengambil sebuah nilai dan melakukan operasi matematika. Pada arsitektur CNN, *neuron* terletak pada perhitungan akhir keluaran peta fitur atau sesudah proses perhitungan konvolusi atau *pooling* untuk menghasilkan suatu pola fitur [14]. Berikut beberapa fungsi aktivasi yang digunakan pada penelitian ini.

A. Fungsi Aktivasi *Rectified Linear Unit* (RELU)

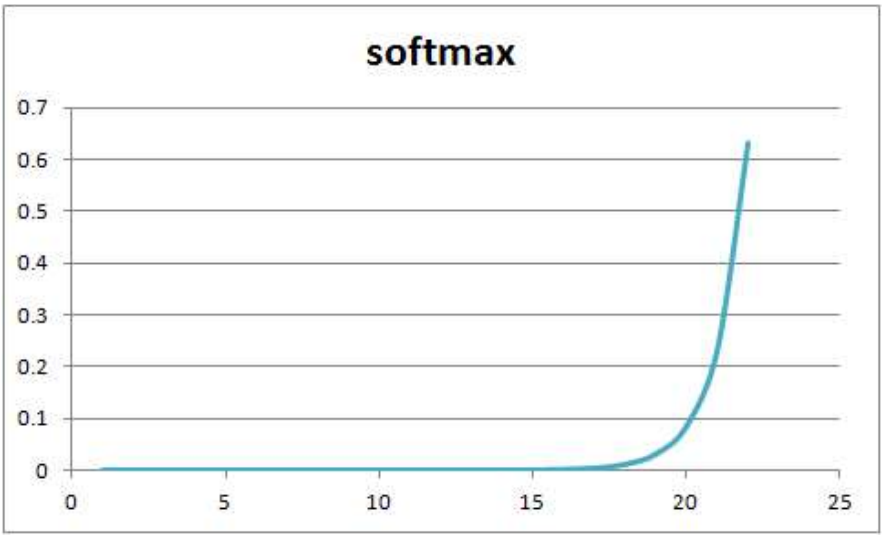
Merupakan fungsi aktivasi yang paling sering digunakan pada *hidden layer*. ReLU menghasilkan keluaran 0 apabila masukannya kurang dari 0 dan menghasilkan keluaran mentah sampai dengan tak terhingga [15].



Gambar 2.13 Grafik fungsi ReLU [15]

B. Fungsi Aktivasi *Softmax*

Merupakan fungsi aktivasi yang umumnya digunakan pada lapisan terakhir jaringan syaraf tiruan. *Softmax* menghitung ditribusi probabilitas dari bobot masukan yang diberikan. Rentang probabilitas keluarannya dalah antara 0 sampai dengan 1 dan jumlah semua probabilitas akan sama dengan 1. Keuntungan dari fungsi ini adalah mampu menangani beberapa kelas [15].

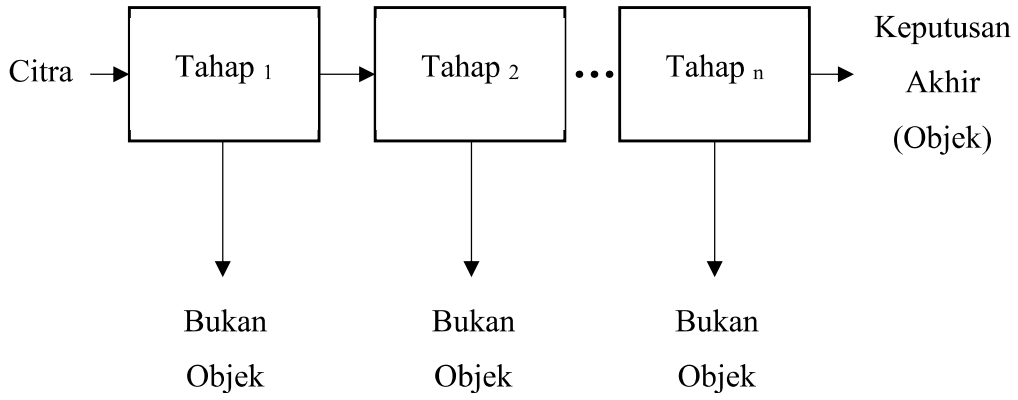


Gambar 2.14 Grafik Fungsi *Softmax* [15]

2.2.6 *Haar Cascade Classifier*

Cascade Classifier terdiri dari susunan tahapan dimana setiap tahap terdiri dari susunan *weak classifier*. Dinamakan *weak classifier* dikarenakan tidak mampu mengklasifikasikan daerah pada tingkat objek. Sistem mendeteksi objek dengan memindahkan jendela diatas citra. Setiap tahap *classifier* memberikan label positif atau negatif pada daerah tertentu berdasarkan lokasi jendela. Positif menandakan objek yang dicari ditemukan sedangkan negatif menandakan objek yang dicari tidak ditemukan [16].

Jika dari proses pelabelan menghasilkan nilai negatif, maka klasifikasi pada daerah tersebut selesai dan jendela bergerak ke bagian lain di dalam citra. Jika pelabelan menghailkan nilai positif maka proses klasifikasi masuk ke tahap selanjutnya. Pengklasifikasi akan menghasilkan keputusan akhir dengan nilai positif apabila di semua tahap (sampai dengan tahap terakhir) objek ditemukan dalam citra [17]. Gambar 2.12 menunjukkan proses klasifikasi menggunakan *Haar Cascade classifier*.



Gambar 2.15 Alur klasifikasi *Haar Cascade Classifier* [18]