BAB III
LANDASAN TEORI

3.1 Kendaraan Roda Dua / Motor

Kendaraan bermotor yaitu kendaraan yang digerakkan oleh peralatan teknik untuk pergerakannya, dan digunakan untuk transportasi darat. Umumnya kendaraan bermotor menggunakan mesin pembakaran dalam (perkakas atau alat untuk menggerakkan atau membuat sesuatu yg dijalankan dengan roda, digerakkan oleh tenaga manusia atau motor penggerak, menggunakan bahan bakar minyak atau tenaga alam).

Kendaraan bermotor memiliki roda, dan biasanya berjalan di atas jalanan. Berdasarkan UU No. 14 tahun 1992 , yang dimaksud dengan peralatan teknik dapat berupa motor atau peralatan lainnya yang berfungsi untuk mengubah suatu sumber daya energi tertentu menjadi tenaga gerak kendaraan bermotor yang bersangkutan. Pengertian kata kendaraan bermotor dalam ketentuan ini adalah terpasang pada tempat sesuai dengan fungsinya. Termasuk dalam pengertian kendaraan bermotor adalah kereta gandengan atau kereta tempelan yang dirangkaikan dengan kendaraan bermotor sebagai penariknya.

3.2 Citra

3.2.1 Definisi Citra

Citra adalah gambar yang terletak pada bidang dua dimensi. Dilihat dari sudut pandang matematis, citra adalah fungsi *continue* dari intensitas cahaya pada bidang dua dimensi. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pemantulan cahaya ini ditangkap oleh alat-alat optik antara lain layaknya mata pada manusia, alat sensor cahaya, kamera, *scanner*, dan lain sebagainya. Sehingga bayangan objek tersebut dapat tersimpan kedalam format digital maupun analog (Munir, 2004).

Citra sebagai keluaran dari suatu sistem perekaman data dapat bersifat :

1. Optik berupa foto
2. Analog berupa sinyal video seperti citra pada monitor televisi.
3. Digital yang dapat langsung disimpan pada suatu pita magnetik.

Citra atau *image* juga dapat didefinisikan sebagai fungsi dua dimensi *f(x,y)* di mana x dan y adalah koordinat bidang datar, dan harga fungsi f disetiap pasangan koordinat *(x,y)* disebut intensitas atau level keabuan *(grey level)* dari gambar di titik yang ada (Hermawati, 2013).

3.2.2 Definisi Citra Digital

Secara umum, pengolahan citra digital merujuk pada pemrosesan gambar 2 dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada proses setiap data 2 dimensi. Citra digital merupakan sebuah struktur *(array)* yang berisi nilai-nilai *real* maupun komplek yang direpresentasikan dengan deretan *bit* tertentu. (Putra, 2010).

3.3 *Artificial Intelligence*

Artificial Intelligence adalah kemampuan komputer digital atau robot yang dikendalikan komputer untuk melakukan tugas yang umumnya dikaitkan dengan makhluk cerdas. *Artificial intelligence* biasanya berkaitan dengan sebuah mesin yang menyelesaikan tugas-tugas dengan melibatkan tingkat kecerdasan tertentu yang sebelumnya dianggap hanya dilakukan oleh manusia. Simulasi proses kecerdasan manusia oleh mesin, terutama sistem komputer meliputi pembelajaran, penalaran, dan koreksi diri (Kumar, 2018).

3.3.1 *Machine Learning*

Alasan utama para peneliti menemukan beberapa masalah menjadi lebih sulit adalah ketika masalah-masalah tersebut tidak sesuai dengan teknik yang digunakan dalam AI. Aturan sistem dasar *Hard-code* atau perbaikan algoritma tidak berfungsi dengan baik untuk hal-hal seperti pengenalan gambar atau ekstraksi makna dari teks. Kemudian didapatkan solusi bahwa tidak hanya meniru perilaku manusia (AI) tetapi juga meniru bagaimana cara manusia belajar. Kemudian ditemukan ide mengenai pembelajaran mesin *(Machine Learning)*. *Machine Learning* adalah suatu cara yang digunakan melalui algoritma pembelajaran dengan memberikan banyak data, kemudian mengajarkan mesin tentang sesuatu dan membiarkan mesin belajar untuk dirinya sendiri, sebagai hasilnya mesin dapat memprediksikan. Hal tersebut merupakan awal mula dari terbentuknya *Neural Network* (jaringan saraf) (Jeffcock, 2018).

3.3.2 *Deep Learning*

Secara definisi *deep learning* merupakan bagian dari *machine learning* yang digunakan untuk pemodelan abstraksi tingkat tinggi pada suatu data berdasarkan algoritma dengan menggunakan lapisan implementasi dan menggunakan struktur yang kompleks atau sebaliknya, terdiri dari beberapa transformasi *non-linear*. (Fikrieabdillah, 2016). *Deep learning* biasanya menggunakan Teknik *Restricted Boltzmann Machine* (RBM) yang digunakan untuk mempercepat proses *training*. Lapis yang digunakan biasanya lebih dari 7, dengan bantuan *deep learning* waktu yang digunakan untuk proses *training* menjadi lebih sedikit hal itu dikarenakan semakin rendahnya masalah hilangnya gradien pada propagasi balik (Abu Ahmad, 2017).

Deep learning mempunyai sebuah fitur untuk mengekstraksi pola yang didapatkan dari data yang membantu model untuk membedakan kelas sehingga fitur ini juga berperan untuk pencapaian hasil prediksi yang baik, fitur ini disebut dengan *Feature Engineering*. Perkembangan *deep learning* membantu pemecahan permasalahan data besar sepeti *Computer Vision, Speech recognition,* dan *Natural Language Processing*. *Deep learning* merupakan cabang dari *Machine learning* yang terinspirasi dari *kortex* manusia dengan menerapkan jaringan syaraf buatan yang memiliki banyak *hiden layer* (Santoso & Ariyanto, 2018).

3.4 *Neural Network*

Neural Network (NN) adalah sebuah set algoritma, memiliki model rendah setelah otak manusia, yang didesain untuk mengenali pola. Mereka menginterpretasikan sensor data melalui seperti mesin persepsi, pelabelan atau pengelompokkan input mentah. Pola yang dikenali adalah *numeric*, terisi dalam *vector*, dimana semua data dunia nyata, baik itu gambar, suara, teks atau deret waktu, akan diterjemahkan.

Neural Network membantu manusia untuk mengelompokkan dan mengklasifikasikan. Manusia dapat memikirkan tentang *neural network* sebagai leyer pengelompokkan dan pengklasifikasian pada bagian data paling atas yang sudah di simpan dan dimanajement. NN membantu untuk menyatukan data yang tidak terlabel sesuai dengan kesamaan diantara contoh input, dan NN

mengklasifikasikan data ketika NN memiliki dataset terlabel untuk dilatih. NN juga dapat mengekstrak fitur yang diumpankan ke algoritma lain untuk mengelompokkan dan mengklasifikasikan, sehingg manuda dapat memikirkan NN dalam pelajaran mendalam *(Deep Learning)* sebagai komponen besar aplikasi *Machine Learning* melibatkan algoritma untuk pembelajaran penguatan, klasifikasi dan regresi.

Deep Learning adalah nama yang digunakan untuk *"Stacked Neural Network"* yaitu, *networks* tersusun atas beberapa *layer. Layer* ini terbuat sebagai *node*. Sebuah *node* hanya ditempati dimana komputasi terjadi, bermotif longgar pada *neuron* diotak manusia yag terbakar ketika bertemu rangsangan yang cukup. Sebuah *nod*e berisi masukan dari data denga seset koefisien atau bobot yang memperkuat dan mengurani input yang ada, dengan demikian memberikan signifikansi pada input terkai dengan algorimta tugas yang dicoba. Produk masukan input ini dijumlahkan kemudian dieruskan melewati fungsi aktivasi yang disebut sebagai *node,* untuk menentukan apa dan sejauh mana sinyal harus berkembang lebih jauh melalui jaringan untuk mempengaruhi hasil akhir, katakanlah tindakan klasifikasi. Jika sinyal melewati, neuron telah "diaktifkan".

3.5 *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah salah satu jenis jaringan saraf *feedforward.* Pada tahun 1960-an, ketika Hubel dan Wiesel meneliti saraf yang digunakan untuk orientasi sensitif-selektif lokal dalam visual kucing, mereka menemukan struktur jaringan khusus dapat secara efektif mengurangi kompleksitas umpan balik dari jaringan saraf kemudian mengusulkan *Convolutional Neural Network (CNN).* CNN diakui sebagai algoritma yang efisien, yang banyak digunakan dalam pengenalan pola dan pemrosesan gambar. Ini memiiki banyak fitur seperti stuktur sederhana, parameter pelatihan kurang dan kemampuan beradaptasi. Hal ini telah menjadi topik hangat pada analisis suara dan pengenalan gambar. Bobot struktur jaringan bersama membuatnya lebih mirip dengan jaringan saraf biologis. Ini mengurangi kompleksitas model jaringan dan jumlah bobot (Liu, et al., 2015).

Secara umum, struktur CNN mencakup dua lapisan, lapisan pertama adalah lapisan ekstraksi, masukan dari setiap neuron terkoneksi ke bidang reseptif lokal dari lapisan sebelumnya, kemudian mengekstrak fitur lokal. Setelah fitur lokal diekstraksi, hubungan posisi antara itu dan fitur lainnya juga akan ditentukan. Yang lainnya adalah layer peta fitur, masing-masing layer komputasi jaringan terdiri dari sejumlah peta fitur. Setiap peta fitur adalah pesawat, bobot neuron di pesawat itu sama. Struktur peta fitur menggunakan fungsi sigmoid sebagai fungsi aktivasi jaringan konvolusi, yang membuat peta fitur memiliki invarian bergeser. Selain itu, karena neuron dalam bidang pemetaan yang sama berbagi bobot, jumlah parameter jaringan bebas berkurang. Setiap lapisan konvolusi dalam CNN diikuti oleh lapisan komputasi yang digunakan untuk menghitung rata-rata lokal dan ekstrak kedua, ini struktur ekstraksi dua fitur yang unik untuk mengurangi resolusi (Liu, et al., 2015).

3.5.1 *Design Architecture*

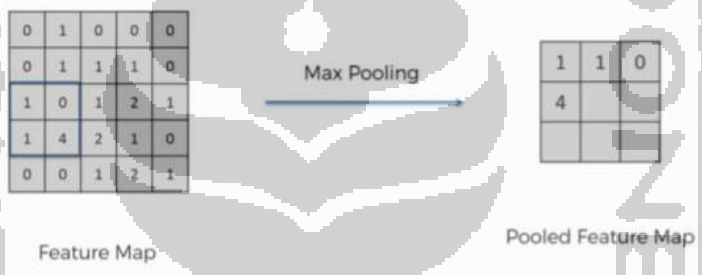
Algoritma *CNN* membutuhkan pengalaman dalam desain arsitektur, dan membutuhkan debug tanpa berhenti pada praktik aplikasinya, untuk mendapatkan aplikasi tertentu yang lebih cocok pada arsitektur CNN. Berdasarkan pada gambar keabuan sebagai masukan 96 x 96, dalam bagian proses, ubah gambar menjadi 32 x 32 dari gambar aslinya. Desain kedalaman lapisan 7 model konvolusi: lapisan masukan, lapisan konvolusi C1, Pengambilan lapisan sub sampel S1, lapisan konvolusi C2, pengambilan lapisan sampel S2, lapisan tersembunyi H, dan lapisan keluaran F.

3.5.2 *Convolutional Layer*

Proses konvolusi memanfaatkan filter. Seperti layaknya gambar, filter memiliki ukuran tinggi, lebar, dan tebal tertentu. Filter ini diinisialisasi dengan nilai tertentu (random atau menggunakan teknik tertentu seperti Glorot), dan nilai dari filter inilah yang menjadi parameter yang di-*update* dalam proses *learning*. Pada setiap posisi gambar, dihasilkan sebuah angka yang merupakan *dot product* antara bagian gambar tersebut dengan filter yang digunakan. Dengan menggeser *(convolve)* filter disetiap kemungkinan poisisi filter pada gambar, dihasilkan sebuah *activation map* (Dharmadi, 2018).

3.5.3 *Pooling Layer*

Pooling layer memiliki fungsi untuk mereduksi input secara spasial (mengurangi jumlah parameter) dengan operasi *down-sampling*. Metode *pooling* terbagi menjadi 2 yaitu *max pooling* dan *average pooling (L2 – norm pooling)* (Dharmadi, 2018). Tujuan dari *Max pooling* adalah mengizinkan CNN untuk mendeteksi objek ketika di tampilkan dengan gambar dalam cara bagaimanapun. *Max pooling* terkonsentrasi untuk mengajarkan CNN mengenali objek terlepas dari semua perbedaan yang disebutkan. Operasi *max pooling* adalah mencari nilai maksimum dari setiap pecahan baris dan kolom peta fitur, kemudian nilai maksimum tersebut dimasukkan kedalam peta fitur *pooled* seperti yang terlihat pada gambar 3.1 (Sumit, 2018)



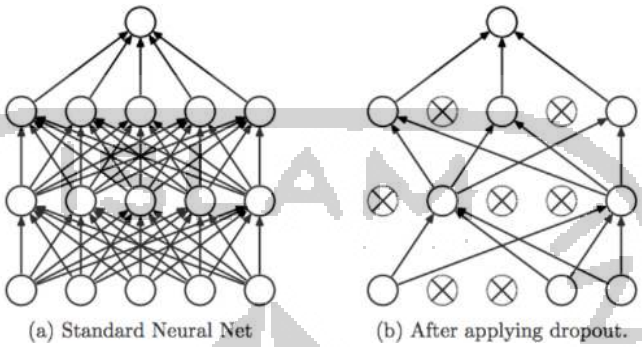
Gambar 3. 1 Contoh Operasi *Max Pooling*

Average Pooling yaitu pengambilan sebuah *grid* kecil dari *output convolutional layer* untuk diambil nilai rata-ratanya. Pada akhir proses, setiap *neuron* pada *convolutional layer* perlu ditransformasi menjadi data satu dimensi terlebih dahulu sebelum dapat dimasukkan kedalam sebuah *fully connected layer*. *Fully connected layer* tersebut dapat menyebabkan data kehilangan informasi spasialnya dan sifatnya yang tidak *reversible*, maka *fully connected layer* hanya diimplementasikan pada akhir jaringan *Invalid source specified.*

3.5.4 *Dropout Regularization*

Dropout regularization, Merupakan Teknik regulasi pada sebuah jaringan syaraf tiruan dimana beberapa *neuron* akan dipilih untuk tidak aktif agar sebuah jaringan syaraf tiruan tidak mengalami *overfitting*. *Dropout* ini merupakan sebuah Teknik yang digunakan untuk memilih secara acak *neuron* mana yang tidak di gunakan pada saat *training*. *Neuron* ini di "*Drop-out*" secara acak. Artinya kontribusi *neuron* yang di buang ini akan di berhentikan sementara pada saat

melakukan prosess *feedforward* dan tidak akan diberikan bobot baru pada *neuron* pada saat melakukan *backrpopagation* .



Gambar 3. 2 Penerapan *dropout regularization*

Pada gambar 3.2 di atas yang merupakan jaringan syaraf tiruan (a) yang merupakan jaringan syaraf tiruan biasa dengan 2 *Hidden layer* dan tanpa *dropout.* Sedangkan jaringan syaraf tiruan (b) merupakan jaringan yang memiliki 2 *Hidden layer* tetapi diberlakukan regulasi *dropout* sehingga beberapa *neuron* tidak aktif / digunakan.

3.5.5 *Fully Connected Layer*

Setelah melalui proses konvolusi, sebuah citra menjadi nilai input pada *fully-connected layer.* Ini merupakan lapisan dimana semua nilai dari lapisan *neuron* sebelumnya *saling* terhubung seperti jaringan syaraf tiruan pada umumnya. Pada lapisan ini semua nilai yang sebelumnnya berbentuk matriks berubah menjadi sebuah matriks satu dimensi, atau yang lebih di kenal sebagai vektor sebelum nilai dari aktivasi tersebut dapat masuk pada *layer* ini.

3.5.6 *Softmax Classifier*

Softmax Classifier adalah sebuah fungsi aktivasi yang di gunakan untuk permasalahan klasifikasi, biasanya fungsi aktivasi ini digunakan pada *output layer.* Pada dasarnya fungsi ini adalah probabilitas eksponential yang dinormalisasi dari pengamatan kelas yang di wakili sebagai aktivasi neuron. Fungsi eksponensial akan meningkatkan probabilitas nilai maksimum lapisan sebelumnnya dibandingkan dengan nilai lainnya.

Softmax function adalah perhitungan kemungkinan dari masing-masing kelas target ata semua kelas target yang memungkinkan dan membantuk untuk

menentukan kelas target untuk *input* yang diberikan. *Sotfmax* mempunyai keuntungan, seperti nilai rentang probabilitas yang dihasilkan dari 0 hingga 1, dan jumlah semua kemungkinan sama dengan satu. Ketika *softmax* digunakan untuk model klasifikasi multi, maka akan mengembalikan peluang dari masing-masing kelas dan kelas target akan memiliki probabilitas lebih tinggi dari kela yang yang lain. Perhitungan *softmax* menggunakan eksponensial dari nilau masukan yang diberikan dan *sum* dari nilai *eksponensial* dari semua nilai dalam masukan (Sofia, 2018).

3.5.7 Forward Propagation

Proses *forward propagation* pada jaringan *CNN* dilakukan untuk meneruskan nilai pada lapisan masukan hingga pada lapisan keluaran. Nilai ini diteruskan melalui lapisan konvolusi, subsampling dan lapisan *fully connected* sesuai dengan urutan lapisan tersebut ditempatkan pada jaringan yang digunakan. Maka dari itu perlu dilakukan perancangan bentuk struktur *CNN* yang akan digunakan terlebih dahulu. Urutan proses runut maju pada *CNN* dapat diringkas sebagai berikut:

1. Inisialisasi nilai awal pada *filter* pada lapisan konvolusi dan bobot pada lapisan *fully connected* dengan nilai acak, dan bias dengan nilai awal 0.
2. Melakukan proses konvolusi gambar masukan sesuai dengan *filter* pada lapisan konvolusi. Proses konvolusi dilakukan sesuai dengan persamaan (3.1) (Zhang, 2016) untuk menghasilkan *feature maps* ke *p* (C_p^1) dari *filter* ($k_{1,p}^1$) dan bias (b_p^1) yang dioperasikan pada gambar masukan (*I*). Tanda $*$ menotasikan proses konvolusi, dan $\sigma(x)$ menotasikan fungsi aktivasi.

$$C_p^1 = \sigma\left(I * k_{1,p}^1 + b_p^1\right) \tag{3.1}$$

3. *Feature maps* yang didapatkan akan dikurangi ukurannya untuk mengurangi kompleksitas perhitungan pada lapisan selanjutnya. Proses ini dilakukan pada lapisan subsampling. Proses *subsampling* dengan menggunakan *max pooling*, atau meloloskan nilai tertinggi dari *feature maps* yang ada dalam sebuah jendela *subsampling*.
4. Hasil dari lapisan subsampling merupakan *feature maps* yang telah direduksi ukurannyaya, jika pada struktur lapisan *CNN* yang digunakan terdapat lapisan

konvolusi setelah lapisan subsampling, maka tahapan selanjutnya adalah sama dengan tahap 1-3, jika tidak maka lanjutkan ke tahap 5.

5. *Feature maps* yang didapat dari lapisan subsampling terakhir merupakan *feature maps* yang akan digunakan pada lapisan *fully connected* sebagai fitur untuk melakukan klasifikasi. *Feature maps* yang berupa matriks akan diuraikan menjadi vector yang panjang seperti pada Gambar 3.2. Proses ini disebut *vectorization and concatenation* (Zhang, 2016) yang dinotasikan pada persamaan (3.24). Fitur yang masuk ke dalam lapisan *fully connected* (f) merupakan hasil proses vektorisasi ($F(x)$) dari hasil subsampling pada lapisan sebelumnya (S_p^1), proses ini menggabungkan seluruh n buah *feature maps*.

$$f = F\left(\{S_p^1\}p = 1,2,3,\dots n\right) \tag{3.2}$$

6. Selanjutnya adalah proses perhitungan prediksi target dari fitur yang masuk ke dalam lapisan *fully connected.* Nilai prediksi kelas ($y(i)$) ini dilakukan dengan melakukan perhitungan menggunakan persamaan (3.2). Perhitungan pada persamaan ini menggunakan fitur dari lapisan subsampling sebelumnya ((j)) yang dikalikan dengan bobot yang terkoneksi ((i,)) dan ditambahkan dengan bias ($b(i)$).

$$\hat{y}(i) = \sigma\ \left(\sum_{j=1}^{n} W(i,j)f(j) + b(i)\right) \tag{3.3}$$

7. Untuk mengetahui seberapa baik proses pembelajaran telah dilakukan, maka nilai *Loss* dihitung dengan persamaan (3.3).

3.5.8 Backward Propagation

Proses untuk memperbaharui nilai *filter* dan bobot pada jaringan adalah proes propagasi balik. Perhitungan perubahan nilai bobot dihitung dimulai dari lapisan *fully connected.* Pada lapisan ini perubahan bobot dicari dengan mencari derivatif *loss function* terhadap bobot (Zhang, 2016). Perhitungan perubahan ($\Delta W(i,j)$) yang terhubung dengan node penghasil nilai fitur $f(j)$ berdasarkan selisih prediksi kelas dari data ke i ($\hat{y}(i)$) dengan target aktual dari data ke i ($y(i)$) pada lapisan *fully connected* dapat dilihat pada persamaan .

$$\Delta W(i,j) = \begin{cases} \left(\hat{y}(\mathrm{i}) - \mathrm{y}(1)\right).f(j), & if\ \hat{y}(\mathrm{i}) < 0 \\ 0, & otherwise \end{cases} \tag{3.4}$$

Perubahan bias ($\Delta(i,j)$) juga dapat dilakukan dengan mencari derivatif *loss function* terhadap bias. Perubahan bias dapat dihitung menggunakan persamaan (3.5)

$$\Delta b(i) = \begin{cases} \left(\hat{y}(\mathrm{i}) - \mathrm{y}(1)\right), & if\ \hat{y}(\mathrm{i}) < 0 \\ 0, & otherwise \end{cases} \tag{3.5}$$

Selanjutnya adalah menghitung perubahan nilai *filter* pada lapisan konvolusi, perubahan ini diasarkan atas galat pada lapisan subsampling. Sehingga, sebelum menghitung perubahan bobot pada lapisan konvolusi, perlu dilakukan *upsampling* dari galat, karena setelah melakukan konvolusi *feature maps* melewati lapisan subsampling dan proses vektorisasi. Perhitungan perubahan *feature maps* (Δf) dilakukan dengan persamaan (3.6).

$$\Delta f = \begin{cases} \left(\hat{y}(\mathrm{i}) - \mathrm{y}(1)\right).W(i,j), & if\ \hat{y}(\mathrm{i}) < 0 \\ 0, & otherwise \end{cases} \tag{3.6}$$

Setelah didapat perubahan dari *feature maps* yang masih berbentuk vector panjang, maka dilakukan proses untuk membalikkan vector ini ke bentuk matriks 2 dimensi. Perubahan ini dapat dinotasikan pada persamaan (3.7)

$$\{S_p^1\}_{p=1,2,3,\ldots n} = f^{-1}(\Delta f) \tag{3.7}$$

Proses *upsampling* adalah merubah matriks $\{\Delta S_p^1\}$ yang merupakan matriks hasil subsampling kembali ke ukuran awal sebelum dilakukan proses subsampling. Hal ini dilakukan dengan meneruskan nilai matriks $\{\Delta S_p^1\}$ kepada koordinat dari *feature maps* yang diloloskan nilainya pada proses subsampling (berkontribusi).

Sedangkan untuk koordinat yang tidak diloloskan nilainya pada proses subsampling dapat diberi nilai 0. Penerusan nilai ini dinotasikan pada persamaan (3.8)

$$\Delta b(i) = \begin{cases} \Delta S_p^1\left(\left[\frac{i}{2}\right], \left[\frac{j}{2}\right]\right), & if\ C_p^1(i,j)\ contributed \\ 0, & otherwise \end{cases} \tag{3.8}$$

Setelah proses *upsampling*, maka $\Delta C_p^1(i,j)$ dapat digunakan untuk menghitung perubahan nilai pada *filter* konvolusi di lapisan sebelumnya. Pencarian perubahan nilai *filter* ($\Delta k_{1,p}^1$) dilakukan dengan melakukan konvolusi gambar

masukan (I) dengan menggunakan ($\Delta C_{1,\sigma}^{1}$). Proses pencarian nilai perubahan nilai *filter* konvolusi dapat dinotasikan pada persamaan (3.9).

$$\Delta C_{1,\sigma}^{1}(i,j) = \begin{cases} \Delta C_{p}^{1}(i,j), & if\ C\ {}_{p}^{1}(i,j)\ > 0 \\ 0, & otherwise \end{cases}$$

$$\Delta k_{1,\mathrm{p}}^{1} = I_{rot180} * \Delta C_{p,\sigma}^{1} \tag{3.9}$$

Pada lapisan konvolusi juga terdapat bias, nilai bias juga diperbaharui untuk mendukung proses pembelajaran. Perhitungan perubahan nilai bias ($\Delta b_{1,p}^{1}$) dilakukan hampir sama dengan perhitungan perubahan nilai *filter* konvolusi, namun tidak melibatkan nilai masukan. Sehingga perubahan nilai bias sama dengan jumlah seluruh ($\Delta C_{1,\sigma}^{1}$) setelah *upsampling* seperti yang telah dinotasikan pada persamaan (3.10)

$$\Delta b_{1,\mathrm{p}}^{1} = \sum_{i=1}^{row\ of\ C^{1}} \sum_{j=1}^{column\ of\ C^{1}} \Delta C_{p}^{1}(i,j) \tag{3.10}$$

Setelah menghitung perubahan pada tiap-tiap lapisan, maka proses memperbaharui nilai *filter*, bias pada lapisan konvolusi, bobot pada lapisan *fully connected*, serta bias yang lama dapat dilakukan sebagaimana dijabarkan pada persamaan (3.11), (3.12), (3.13) dan (3.14)

$$k_{1,\mathrm{p}}^{1} = k_{1,\mathrm{p}}^{1} - \alpha \cdot \Delta k_{1,\mathrm{p}}^{1} \tag{3.11}$$

$$b_{1,\mathrm{p}}^{1} = b_{1,\mathrm{p}}^{1} - \alpha \cdot \Delta b_{1,\mathrm{p}}^{1} \tag{3.12}$$

$$W = W - \alpha \cdot \Delta W \tag{3.13}$$

$$b = b - \alpha \cdot \Delta b \tag{3.14}$$

Proses ini dilakukan hingga kondisi terhenti ditemukan, kondisi terhenti ini bisa saja berupa epoch maksimum yang tercapai atau nilai *loss* yang berada dibawah batasan yang ditetapkan. Proses perubahan nilai bobot, bias dan *filter* dilakukan setiap satu data masuk ke dalam jaringan.

3.6 *Activation Function*

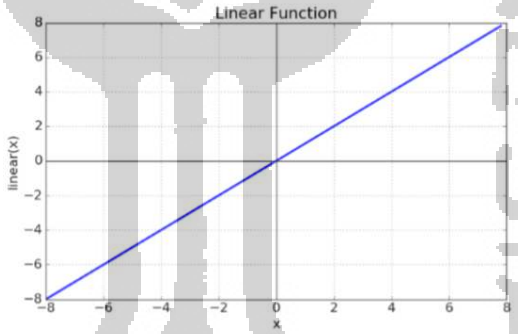
Activation Function merupakan operasi matematik yang dikenakan pada sinyal *output* y. Fungsi ini digunakan untuk mengaktifkan atau menonaktifkan neuron. Perilaku dari JST di tentukan oleh bobot-bobot dan *input-output* fungsi aktivasi yang ditetapkan.

Fungsi aktivasi adalah fungsi yang menentukan apakah suatu neuron akan di aktifkan atau tidak. Fungsi yang di pakai bisa berupa fungsai linear dengan nilai batas atau ReLu, atau fungsi non-linear seperti fungsi sigmoid dan tanh.

Activation function adalah sebuah titik yang ditambahkan di akhir *output* dari setiap jaringan syaraf. *Activation function* juga dikenal sebagai *Transfer Function* yang digunakan untuk menentukan *output neural network. Activation function* dibagi menjadi dua tipe yaitu *linear* dan *non linear. Activation function* digunakan memutuskatn hasil dari *neural network* seperti "setuju" atau "tidak setuju". Nilai peta hasil berkisar diantara 0 atau 1, -1 sampai 1, dan lain sebagainya (berdasarkan fungsinya) (Sharma, 2017).

3.6.1 *Linear Activition Function*

Fungsi aktivasi linier dapat dilihat pada gambar 3.3, pada gambar tersebut dapat diketahui fungsinya adalah garis atau linier. Hal ini menyebabkan output dari fungsi tidak dibatasi antara suatu rentang.



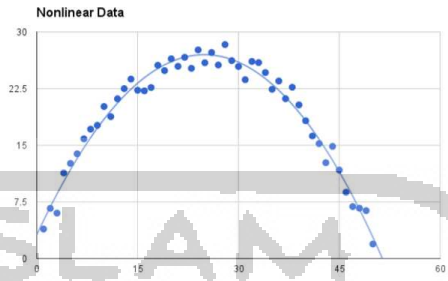
Gambar 3. 3 *Linear Activation Function*

(Sumber: https://towardsdatascience.com/activation-functions-neuralnetworks-1cbd9f8d91d6)

Fungsi persamaan linier activation function adalah f(x) = x dengan nilai x berkisar antara −∞ sampai ∞. Hal ini biasanya tidak membantu terhadap parameter kompleksitas atau variasi data yang diteruskan ke jaringan saraf (Sagar, 2017)

3.6.2 Non - Linier Activition Function

Non – linier Activation Function adalah fungsi aktivasi yang sering digunakan. Non-linieritas membantu untuk membuat grafik terlihat seperti pada gambar 3.4



Gambar 3. 4 *Non – linear Activation Function*

Hal ini membuat kemudahan dalam menggeneralisasi model atau beradaptasi dengan variasi data dan untuk membedakan antar output. Terminology utama yang perlu dimengerti adalah:

Derivative atau Differential : Perubahan sumbu y menjadi sumbu x. hal ini juga disebut dengan kemiringan.

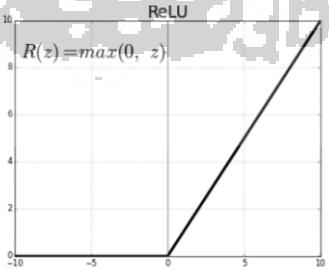
Fungsi Monotonic : Sebuah fungsi yang Suatu fungsi yang seluruhnya tidak meningkat atau tidak menurun.

Fungsi Aktivasi non-linier terfokus pada pembagian dasar dari range atau kurva.

3.6.3 Rectifield Linier Unit (ReLU)

ReLU (Rectified Linier Unit) merupakan fungsi aktivasi yang biasanya digunakan dalam pemodelan *deep learning*. Fungsi kembali ke angka 0 jika mereka menerima input yang negatif, tetapi untuk nilai positif x mereka akan kembali kenilai asal. Perumusan dapat dituliskan sebagai berikut (x) = max (0,) (3.13)

Hal yang mengejutkan bahwa fungsi sederhana (terdiri atas 2 bagian linier) dapat mengizinkan model untuk perhitungan non-linieritas dan interaksi secara baik (Becker, 2018). Grafik terlihat seperti pada gambar 3.5



Gambar 3. 5 *ReLU Activation Function*

3.7 Classification

Secara etimologi klasifikiasi berasal dari bahasa Inggris dari kata "*classification*" dan kata ini berasal dari kata "*to classy*" yang berarti menggolongkan dan menempatkan benda-benda disuatu tempat. Klasifikasi adalah pengelompokkan yang sistematis pada sejumlah objek, gagasan, buku atau benda-benda lain kedalam kelas atau golongan tertentu berdasarkan ciriciri yang sama (Hamakonda, 2008)

3.8 Image Processing

Image Proccessing adalah suatu bentuk pengolahan atau pemrosesan sinyal dengan input berupa gambar *(image)* dan ditransformasikan menjadi gambar lain sebagai keluarannya dengan teknik tertentu. *Image processing* dilakukan untuk memperbaiki kesalahan data gambar yang terjadi akibat transmisi dan selama akuisisi sinyal, serta untuk meningkatkan kualitas penampakan gambar agar lebih mudah diinterpretasi oleh system penglihatan manusia baik dengan melakukan manipulasi dan juga penganalisisan terhadap gambar.

Berdasarkan transformasinya perasi pemrosesan gambar dapat digolongkan sebagai berikut:

1. *Image Enhancement* (Peningkatan Kualitas Gambar)
2. *Image Restoration* (Pemulihan Gambar)
3. *Image Compression* (Kompresi Gambar)

Image Refresentation & Modelling (Representasi dan permodelan gambar) (Ade, 2017)

3.9 Pattern Recognition

Pengenalan pola telah dikembangkan secara konstan selama bertahun-tahun. Komponen dasar dalam pengenalan pola adalah *preprocessing*, ekstraksi fitur, dan klasifikasi. Setelah dataset diperoleh, gambar diproses terlebih dahulu, sehingga menjadi cocok untuk sub-proses berikutnya. Langkah selanjutnya adalah ekstraksi fitur, di mana, dataset diubah menjadi seperangkat vektor fitur yang seharusnya mewakili data asli. Fitur-fitur ini digunakan pada langkah klasifikasi untuk memisahkan titik data menjadi kelas yang berbeda berdasarkan masalah. (Saliba, 2014).

Pola adalah entitas yang terdefinisi dan dapat diidentifikasi melalui ciri-cirinya *(features)*. *Features* digunakan untuk membedakan suatu pola dengan pola lainnya. *Features* pada suatu pola diperoleh dari hasil pengukuran terhadap objek uji. Khusus pada pola yang terdapat didalam gambar, *features* yang dapat diperoleh berasal dari informasi:

a. Spasial : intensitas pixel, histogram

b. Tepi : arah, kekuatan

c. Kontur : garis, elips, lingkaran

d. Wilayah/bentuk : Keliling, luas, pusat massa

e. Transformasi Fourier : frekuensi

Pengenalan pola bertujuan menentukan kelompok atau kategori pola berdasarkan features yang dimiliki oleh pola tersebut. Dengan kata lain, pengenalan pola membedakan suatu objek dengan objek lain. Terdapat dua pendekatan yang dilakukan dalam pengenalan pola: pendekatan secara statistik dan pendekatan secara sintaktik atau struktural (Hendradjaya, 1995).

Ada dua fase dalam sistem pengenalan pola yaitu training step dan recognition step. Pada training step, beberapa contoh citra dipelajari untuk menentukan ciri yang akan digunakan dalam proses pengenalan serta prosedur klasifikasinya. Pada recognition step, pengambilan features pada citra kemudian ditentukan kelas kelompoknya (Munir, 2004).

3.10 *Computer Vision*

Computer Vision atau *Machine Vision* meupakan salah satu *terminology* yang berkaitan erat dengan pengolahan citra. Pada dasarnya, *computer vision* mencoba meniru kerja sistem visual manusia *(human vision)*. *Human vision* sesungguhnya sangat kompleks. Manusia melihat objek dengan indera penglihatan (mata), lalu citra objek diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti objek yang tampak dalam pandangan matanya. *Computer Vision* adalah proses otomatis yang mengintegrasikan sejumlah besar proses untuk persepsi visual, seperti akuisi citra, pengolahan citra, klasifikasi, pengenalan *(recognition),* dan membuat keputusan.

Computer Vision merupakan salah satu bagian dari ilmu komputer yang mempelajari sebuah komputer dapat mendeteksi objek/melihat objek seperti mata manusia. *Computer Vision* dipengaruhi oleh sebuah pencahayaan untuk menentukan kualitas hasil. Semakin banyak cahaya akan membuat kontras gambar menjadi tinggi, begitu sebaliknya jika kekurangan cahaya gambar akan semakin buruk (Wahyudi & Kartowisastro, 2011). *Computer Vision* juga di definisikan tiruan dari cara kerja system visual/ indra pengelihatan dari manusia. Penglihatan sebuah objek melalui proses dari indera penglihatan mata, kemudian diteruskan ke otak untuk di interprestasikan sehingga dapat mengenali sebuah objek untuk pengambilan keputusan, (Munir, 2004).

Table 3.1 Perbandingan *Human Vision* dan *CV*

Human Vision	*Computer Vision*
Menggunakan indera penglihatan yaitu mata untuk mendeteksi sebuah objek yang diteruskan ke otak.	Menggunakan kamera-kamera yang terhubung pada sistem komputer.
Dapat langsung mengenali objek yang di lihat, seperti bentuk objek, warna objek, lokasi objek dan pergerakan objek.	Secara otomatis melakukan interpretsi terhadap gambar-gambar dan mencoba mengerti isi dari gambar tersebut seperti halnya Human Vision dalam bentuk citra.

Secara garis besar, *Computer Vision* adalah sebuah teknologi yang di gunakan oleh sebuah komputer untuk dapat melihat. Kemampuan untuk mengenali ini merupakan kembinasi dari pengolahan citra dan pengenalan pola. Pengolahan citra adalah sebuah proses dalam *Computer Vision* untuk menghasilkan citra yang lebih baik dan/atau mudah diinterpretasikanm, sedangkan pengenalan pola adalah proses indentifikasi objek pada sebuah citra. Menurut (Basuki, 2016), Proses dalam *Computer Vision* terbagi menjadi 4 :

1. Proses Mendapatkan citra digital (Image Acquisition)
2. Proses pengolahan citra (Image Processing)
3. Proses Analisis data citra (Image Analysis)
4. Proses Pemahaman data citra (Image Understanding)

Computer Vision merupakan kombinasi antara pemrosesan citra dan proses pengenalan pola. *Computer Vision* adalah pembangunan deskripsi dari objek fisik yang jelas dari sebuah citra/gambar. *Output* dari *Computer Vision* adalah deskripsi atau interpretasi atau beberapa pengukuran kuantutatif struktur dalam adegan 3D.

3.11 Deteksi Objek *(Object Detection)*

Dalam banyak sistem visi komputer, deteksi objek adalah tugas pertama yang dilakukan karena memungkinkan untuk mendapatkan informasi lebih lanjut mengenai objek yang terdeteksi. Setelah instance objek terdeteksi, dimungkinkan untuk mendapatkan informasi lebih lanjut, termasuk:

- Untuk mengenali contoh spesifik (misalnya, untuk mengidentifikasi wajah subjek),
- Untuk melacak objek pada urutan gambar (misalnya, untuk melacak wajah dalam video), dan
- Untuk mengekstraksi informasi lebih lanjut tentang objek (misalnya, untuk menentukan jenis kelamin subjek), sementara juga dimungkinkan untuk Menyimpulkan keberadaan atau lokasi objek lain dalam adegan dan untuk memperkirakan informasi lebih lanjut tentang adegan tersebut di antara informasi kontekstual lainnya.

Deteksi objek sudah digunakan dalam banyak aplikasi, yang paling populer adalah interaksi manusia-komputer (HCI), Robotika (misal. Robot servis), Elektronik konsumen (misal Ponsel pintar), Keamanan (misalnya pengenalan, pelacakan), Pengambilan (misalnya mesin pencari, manajemen foto), dan Transportasi (misalnya mengemudi mandiri dan berbantuan). Masing-masing aplikasi ini memiliki persyaratan yang berbeda, termasuk: waktu pemrosesan (off-*line, on-line, atau real-time*), ketahanan terhadap oklusi, invarian terhadap rotasi dan deteksi dalam pose perubahan.

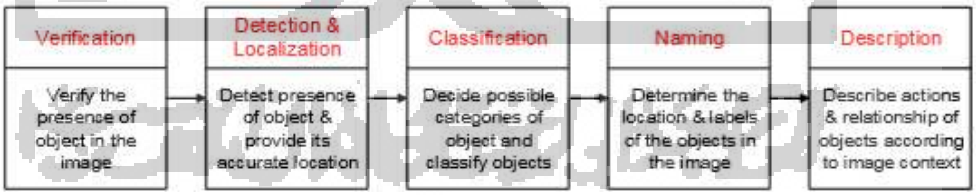
Sementara banyak aplikasi mempertimbangkan deteksi kelas objek tunggal (misalnya, wajah) dan dari tampilan tunggal (misalnya, wajah frontal), yang lain membutuhkan deteksi beberapa objek kelas (manusia, kendaraan, dll.), Atau dari kelas tunggal dari berbagai tampilan (misal tampilan samping dan depan kendaraan). Secara umum, sebagian besar sistem hanya dapat mendeteksi satu kelas

objek tunggal dari sekumpulan tampilan dan pose terbatas (Rodrigo & Ruiz-delSolar, 2015)

Deteksi objek adalah prosedur menentukan instance kelas yang menjadi objek dan memperkirakan lokasi objek dengan mengeluarkan kotak pembatas di sekitar objek. Mendeteksi satu *instance* kelas dari gambar disebut deteksi objek kelas tunggal, sedangkan mendeteksi kelas semua objek yang ada dalam gambar dikenal sebagai deteksi objek multi kelas. Berbagai tantangan seperti oklusi parsial / penuh, berbagai kondisi pencahayaan, pose, skala, dll perlu ditangani saat melakukan deteksi objek (Pathak,dkk, 2018).

Dalam pendeteksian objek biasanya dilakukan dengan mencari setiap bagian gambar untuk melakukan lokalisasi bagian yang memiliki sifat fotometrik atau geometriknya sesuai dengan objek yang di latih pada proses pelatihan dan pengumpulan data. Hal ini dapat dilakukan dengan memindai template objek di gambar di lokasi, skala, dan rotasi yang berbeda, dan deteksi dideklarasikan jika kemiripan antara template dan gambar cukup tinggi. Kemiripan antara template dan area gambar dapat diukur dengan korelasi. Selama beberapa tahun terakhir telah ditunjukkan bahwa pendeteksian objek berbasis gambar sensitif terhadap data pelatihan (Jalled & Voronkov, 2016).

Berikut ini adalah Deteksi objek sebagai langkah terpenting dalam aktivitas pengenalan visual.



Gambar 3. 6 Perbedaan *Object Detection* dan *Image Classification*

Sumber : (Pathak,2018)

3.12 TensorFlow

TensorFlow adalah kerangka *google* yang sangat kuat untuk membangun aplikasi menggunakan *Machine Learning*. *TensorFlow* adalah sebuah alat *opensource* yang dirilis oleh Google. *TensorFlow* dapat dijalankan dengan

menggunakan *Central Processing Unit* (CPU) dan *General Processing Unit* (GPU) (Scarpino , 2018).

Tensorflow Meupakan salah satu *library* yang *open-source* untuk menangai konputasi numerikal menggunakan *data-flow graphs*. *Tensorflow* di kembangkan oleh *Google Brain Team* yang merupakan organisasi peneliti dibawah google yang meneliti tentang *machine learning* dan *deep neural network*. Pertama kali mencapai versi pertamanya Versi 1.0 ke publik pada Februari 2017, dan perkembangannya terus meningkat

Tensorflow sangat populer karena dapat menggunakan sekeluruhan sistem pada komputer, Tepatnya *Tensorflow* dapat menggunakan komputasi CPU dan GPU. Dan bahkan termasuk pada perangkat *mobile* seperti *smartphone,* dan perangkat pendukung *IoT (Internet of Things)* seperti *Raspbery Pi* dan *Arduino.*

3.13 Akurasi Klasifikasi

Akurasi klasifikasi digunakan untuk mengukur kinerja model. Akurasi klasifikasi dapat diketahui dari rumus dibawah ini:

$$Akurasi = \frac{jumlah\ prediksi\ yang\ benar}{total\ jumlah\ sampel\ input}$$

Gambar 3. 7 Akurasi Klasifikasi

Akurasi klasifikasi dapat bekerja dengan baik jika jumlah sampel dari masing-masing kelas yang akan diklasifikasikan sama besar. Semakin tinggi tingkat akurasi (mendekati 100%) berdasarkan jumlah sampel besar dan jumlah masingmasing kelas sama besar maka kinerja model dalam mengklasifikasikan semakin baik. (Mishra, 2018)

3.14 *Python*

Python merupakan Bahasa pemrograman beraras-tinggi yang diciptakan oleh Guido Van Rossum pada tahun 1989 di Amsterdam, Belanda. Sebagai Bahasa beraras-tinggi, *python* menawarkan berbagai kemudahan menulis program. Seiring dengan kecenderungan pengguna pemrograman berorientasi objek dewasa ini, *python* juga sangat tepat digunakan, mengingat *python* memang Bahasa pemrograman yang berorientasi objek. Oleh karena itu, keistimewaan tentang

pewarisan dan instansi yang ditawarkan pada Bahasa yang berorientasi pada objek juga dapat diwujudkan pada *python,* dengan kata lain*, python* mendukung konsep *reusability* (suatu kemudahan untuk mengembangkan kode terhadap kode yang sudah tersedia) (Kadir, 2005)