BAB 2
TINJAUAN PUSTAKA

2.1 Pengolahan Citra Digital

Pengolahan citra digital merupakan sebuah cabang ilmu yang mempelajari tentang teknik-teknik mengelola citra. citra yang dimaksud adalah gambar diam (foto) maupun gambar bergerak. sedangkan digital yang mempunyai maksud bahwa pengolahan citra atau gambar dilakukan secara digital menggunakan komputer. citra merupakan fungsi kontinu dengan intensitas cahaya pada bidang dua dimensi, agar dapat diolah menggunakan komputer digital maka sebuah citra harus dipresentasikan secara numerik yang dapat diwakili oleh sebuah Matrikss dua dimensi f (x,y) yang terdiri dari M kolom dan N baris[4]. setiap foto dalam bentuk citra digital dapat diolah menggunakan perangkat lunak tertentu. misalnya merubah format warna citra menjadi Grayscale, Resize citra dan lain sebagainya.

Pada pengolahan citra digital untuk klasifikasi gambar, sistem *file* yang didukung paling banyak digunakan adalah JPG atau PNG jadi sangat sulit jika dalam penelitian menggunakan data gambar dengan format DISCOM untuk mempermudah dalam penelitian ini maka dilakukan perubahan format gambar dari DISCOM ke PNG dan juga merubah format warna yang semula menggunakan citra RGB ke Grayscale.

2.1.1 Discom

Digital Imaging Communication Medicine (DICOM) merupakan standar internasional untuk penyimpanan dan pertukaran gambar medis. standar DICOM diciptakan untuk membuat pertukaran gambar medis lebih mudah dan independen dari produsen peralatan pencitraan. Selain data gambar, format file DICOM mendukung informasi lain yang berguna untuk menggambarkan gambar. Ini membuat DICOM mudah digunakan dan pertukaran data cepat dan aman sambil menghindari kemungkinan kebingungan yang disebabkan oleh beberapa file untuk studi yang sama. Ini telah menjadi standar terkemuka yang digunakan oleh semua vendor utama peralatan medis diagnostik yang menggunakan pencitraan, misalnya:

kardiologi, mamografi, radiologi, operasi, endoskopi, kedokteran gigi, patologi, dan lain-lain [4].

2.1.2 PNG

Portable Network Graphics (PNG) merupakan format gambar yang dirancang untuk menggantikan format GIF dalam pengeditan gambar PNG memiliki keunggulan karena bisa didapat tanpa kehilangan informasi dari format gambar sebelumnya dan tanpa membayar biaya paten. PNG mendukung tiga jenis gambar *true color*, *grayscale* dan *palette-based* [5].

2.1.3 RGB

Citra RGB, yang disebut juga citra "*true color*", disimpan dalam citra berukuran (m x n) x 3 yang mendefinisikan warna merah (*red*), hijau (*green*), dan warna biru (*blue*) untuk setiap pikselnya. Warna pada tiap piksel ditentukan berdasarkan kombinasi dan warna *red*, *green*, dan *blue* (RGB). RGB merupakan citra 24bit dengan komponen merah, hijau, biru yang masing-masing umumnya bernilai 8bit sehingga intensitas kecerahan warna sampai 256 level dan kombinasi warnanya kurang lebih sekitar 16 juta warna sehingga disebut "*true color*"[6].

2.1.4 Grayscale

Citra *Grayscale* berbeda dengan citra RGB, citra ini didefinisikan oleh satu nilai derajat warna. Umumnya bernilai 8bit sehingga intensitas kecerahan warna sampai 256 level dan kombinasi warnanya 256 varian. Tingkat kecerahan paling rendah yaitu 0 untuk warna hitam dan putih bernilai 255 [6].

Untuk mengkonfersikan citra yang memiliki warna RGB ke derajat keabuan bias menggunakan:

$$Gray = \frac{R + G + B}{3} \tag{2.1}$$

Atau

$$Gray = 0.299.R + 0.587.G + 0.114.B \tag{2.2}$$

2.1.5 Resize Image

Resize Image merupakan tahapan di mana proses mengubah ukuran suatu citra menjadi lebih besar atau kecil dari ukuran citra awal dengan ukuran yang telah ditetapkan sebelumnya. Proses bertujuan untuk mengurangi beban perangkat

disebabkan objek deteksi metode sangat komplek dan banyak membutuhkan ruang penyimpanan yang besar.

2.2 Image Classification

Image Classification Dalam beberapa tahun terakhir, mengelompokkan gambar berdasarkan objek di dalamnya telah mendapatkan popularitas. Ini karena kemajuan dalam algoritma serta ketersediaan dataset besar. Algoritma pembelajaran mendalam untuk klasifikasi gambar telah secara signifikan meningkatkan keakuratan sambil dilatih pada dataset seperti ImageNet. Model yang terlatih sering digunakan lebih lanjut untuk meningkatkan algoritma pengenalan lainnya seperti deteksi objek, serta kategorisasi gambar dalam aplikasi online menggunakan model *Deep Learning* Klasifikasi Objek [7].

2.3 *Deteksi Object*

Deteksi Object adalah masalah yang lebih kompleks dari kedua pelokalan posisi suatu objek dalam suatu gambar serta mengatakan apa jenis objek itu. Karena itu, ini memerlukan teknik yang lebih kompleks. Pertama, harus melokalkan objek atau beberapa objek di dalam gambar. Kedua, ini memberikan kelas prediksi untuk masing-masing objek yang dilokalkan. Ada beberapa metode deteksi objek yang menggunakan pendekatan berbasis jendela geser. Salah satu teknik deteksi yang populer adalah pendekatan deteksi wajah, yang dikembangkan oleh Viola dan Jones. jurnal ini mengeksploitasi fakta bahwa wajah manusia memiliki fitur deskriptif yang kuat seperti daerah di dekat mata yang lebih gelap daripada di dekat mulut. Jadi mungkin ada perbedaan yang signifikan antara area persegi panjang yang mengelilingi mata sehubungan dengan area persegi panjang di dekat hidung. Menggunakan ini sebagai salah satu dari beberapa pola yang telah ditentukan sebelumnya dari pasangan persegi panjang, metode mereka menghitung perbedaan area antara persegi panjang di setiap pola[7].

2.4 *Image Segmentation*

Image segmentation iyalah membuat daerah *cluster* dalam gambar, sehingga satu *cluster* memiliki sifat yang serupa. Pendekatan yang biasa dilakukan adalah mengelompokkan piksel gambar milik objek yang sama. Aplikasi baru-baru ini telah berkembang dalam mobil *self-driving* dan analisis kesehatan menggunakan

wilayah gambar. contoh image segmentasi dari dataset PascalVOC dapat dilihat pada Gambar 2.1 :

Gambar 2. 1 Contoh *Image Segmentasi*

Pada Gambar 2.1 di sebelah kiri, ada beberapa pesawat kecil di latar belakang dan, oleh karena itu, dapat melihat piksel kecil berwarna sesuai pada gambar yang sesuai di sebelah kanan. Di Gambar 2.1 kiri bawah, ada dua hewan peliharaan yang berdekat bersama, oleh karena itu, gambar mereka yang tersegmentasi di sebelah kanan memiliki warna yang berbeda untuk piksel masing-masing milik kucing dan anjing. Dalam Gambar 2.1 ini, batas berwarna berbeda untuk kenyamanan dan tidak menyiratkan kategori yang berbeda.

Dalam teknik *segmentasi* tradisional, properti utama yang digunakan adalah tingkat intensitas gambar. Pertama, daerah kecil yang berbeda dari nilai intensitas yang sama ditemukan, dan kemudian mereka digabung menjadi daerah yang lebih besar. Untuk mendapatkan kinerja terbaik, titik awal dipilih oleh pengguna untuk algoritma. Pendekatan terbaru menggunakan deep learning telah menunjukkan kinerja yang lebih baik tanpa perlu inisialisasi [7].

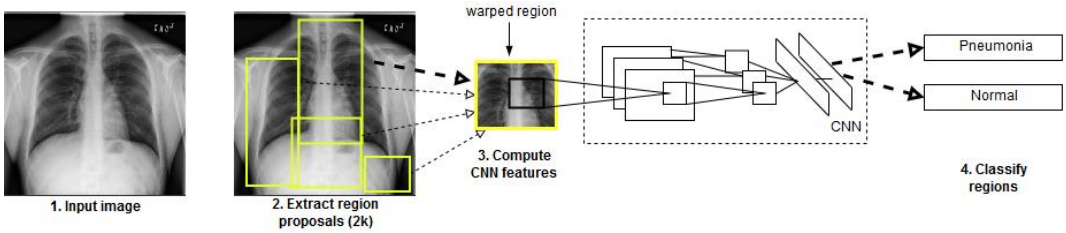
2.5 *Deep Learning*

Deep Learning memungkinkan model komputasi yang terdiri dari beberapa lapisan pengolahan untuk mempelajari representasi data dengan berbagai tingkat abstraksi. Metode ini telah secara dramatis memperbaiki state-of-the-art dalam

speech recognition, pengenalan objek visual, deteksi objek dan banyak domain lainnya seperti penemuan obat dan genomik. *Deep learning* menemukan struktur yang rumit dalam kumpulan data yang besar dengan menggunakan algoritma *backpropagation* untuk menunjukkan bagaimana sebuah mesin harus mengubah parameter internalnya yang digunakan untuk menghitung representasi pada setiap lapisan dari representasi pada lapisan sebelumnya. Jaring *konvolusi* yang dalam telah membawa terobosan dalam memproses gambar, video, ucapan dan audio, sedangkan jala berulang telah menyoroti data *sekuensial* seperti teks dan ucapan[8].

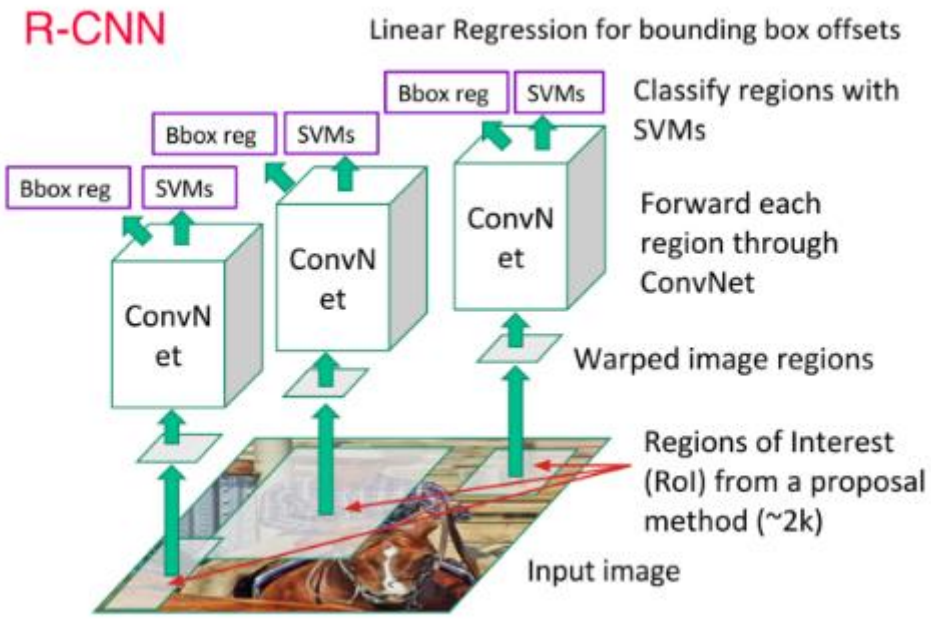
2.6 *Region Convolutional Neural Network* (R-CNN)

Region-based Convolutional Neural Network (R-CNN) merupakan metode pengenalan objek yang diperkenalkan oleh Ross Girshick, Jeff Donahue, Trevor Darrell, dan Jitendra Malik pada tahun 2014. Ross Girshick, et al. memperkenalkan arsitektur R-CNN dalam 3 tahap, yaitu *extract region proposals* (~2k), *compute CNN feature*, dan *classify regions*[2]. Arsitektur tersebut dapat dilihat pada Gambar 2.2 :



Gambar 2.2 Arsitektur R-CNN

Pada tahap *extract region proposals*, R-CNN melakukan *region search* (pencarian wilayah) menggunakan metode *selective search* hingga menghasilkan 2000 kandidat region proposal dalam bentuk persegi. Pada tahap *compute CNN feature*, *Region proposal* tersebut di masukan kedalam *Convolutional Neural Network* (CNN) hingga menjadi fitur. Pada tahap *classify region*, fitur tersebut kemudian menjadi masukan untuk *Support Vector Machine* (SVM) untuk di klasifikasi dan dilakukan perhitungan *bounding-box regression* untuk menghasilkan *bounding box*. Arsitektur ketiga tahap dan *bounding-box regression* tersebut dapat dilihat pada Gambar 2.3 :



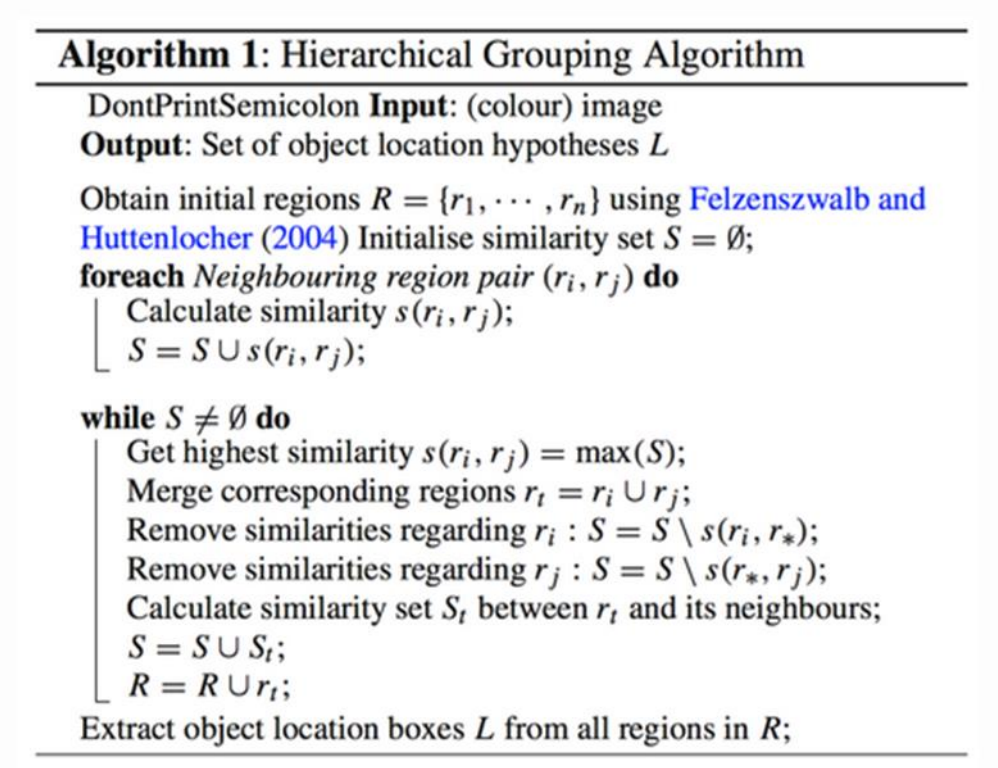
Gambar 2.3 Arsitektur Ketiga Tahap Dan Bounding-box Regression

2.6.1 *Selective Search*

Selective search merupakan metode yang digunakan untuk mencari *region proposal* yang diperkenalkan oleh J.R.R. Uijlings, K.E.A. Van de Sande, T. Gevers, dan A.W.M. Smeulders pada tahun 2012[9]. Secara garis besar, *selective search* mengelompokan suatu region atau daerah pada citra berdasarkan warna, tekstur, ukuran dan kompatibilitas bentuk sehingga *selective search* dapat menghasilkan kemungkinan lokasi objek yang ada pada citra. Kemudian lokasi objek tersebut akan digunakan oleh metode-metode pengenalan objek sebagai awal pembentukan model. *Selective search* memiliki 3 tahap *Selective search by hierarchical grouping, Diversification strategies*, dan *Combining locations* dalam melokalisasi objek.

2.6.1.1 Selective Search by Hierarchical Grouping

Selective Search by Hierarchical Grouping merupakan tahap awal dalam *selective search*, di mana pada tahap ini dilakukan segmentasi citra menggunakan pendekatan *bottom-up grouping*. Proses *grouping* yang terjadi bersifat hierarkis, hingga seluruh citra menjadi *single region*[9][10]. Adapun algoritma dari hierarchical grouping dapat di lihat pada Gambar 2.4:



Algorithm 1: Hierarchical Grouping Algorithm

DontPrintSemicolon **Input**: (colour) image
Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using Felzenszwalb and Huttenlocher (2004) Initialise similarity set $S = \emptyset$;
foreach *Neighbouring region pair* (r_i, r_j) **do**
  Calculate similarity $s(r_i, r_j)$;
  $S = S \cup s(r_i, r_j)$;

while $S \neq \emptyset$ **do**
  Get highest similarity $s(r_i, r_j) = \max(S)$;
  Merge corresponding regions $r_t = r_i \cup r_j$;
  Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$;
  Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$;
  Calculate similarity set S_t between r_t and its neighbours;
  $S = S \cup S_t$;
  $R = R \cup r_t$;
Extract object location boxes L from all regions in R;

Gambar 2. 4 Algoritma Hierarchical Grouping

Prosedur *grouping* ini memiliki proses sebagai berikut:

a. Membuat *initial region* (Wilayat awal)
b. Algoritma *greedy* untuk mengelompokan wilayah tersebut
c. Menghitung setiap region terdekat
d. Mengelompokkan region yang memiliki kemiripan
e. Menghitung kemiripan di setiap neighbouring region (wilayah tetangga). Proses ini diulang sampai seluruh citra menjadi *single* region (satu wilayah).

2.6.1.2 Diversification Strategies

Diversification Strategies merupakan proses diversifikasi pengambilan sampel dan membuat serangkaian strategi komplementer yang lokasinya digabungkan sesudahnya. proses diversifikasi dilakukan dengan menggunakan berbagai ruang warna dengan sifat invarian yang berbeda, menggunakan langkah-langkah kesamaan yang berbeda dan memvariasikan sebagai *starting regions*. pada proses *diversifikasi* memperhitungkan berbagai kondisi pemandangan dan pencahayaan dengan melakukan algoritma pengelompokan hierarkis di berbagai

ruang warna dengan berbagai properti variant untuk $s_{size}(r_i, r_j)$ gambar yang hitam dan putih perubahan ruang warna memiliki dampak kecil pada hasil akhir dari algoritma[9]. Untuk mendorong daerah-daerah kecil untuk bergabung lebih awal. $s_{size}(r_i, r_j)$ didefinisikan dengan persamaan berikut :

$$s_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)} \quad (2.3)$$

$s_{fill}(r_i, r_j)$ merupakan persamaan untuk mengukur seberapa baik wilayah dan kesesuaian satu sama lain. Idenya adalah untuk mengisi kekosongan: jika r_i terkandung dalam r_j, jika r_i dan r_j hampir hampir tidak saling bersentuhan, kemungkinan akan membentuk wilayah yang aneh dan tidak boleh digabung. Untuk menjaga agar pengukuran tetap cepat, maka digunakan ukuran wilayah dan kotak yang berisi. secara khusus, BB_{ij} didefinisikan menjadi kotak terikat di sekitar r_i dan r_j Sekarang $s_{fill}(r_i, r_j)$ adalah sebagain kecil dari gambar yang terkandung dalam BB_{ij} di mana $s_{fill}(r_i, r_j)$ didefinisikan dengan persamaan berikut :

$$fill(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)} \quad (2.4)$$

Sehingga persamaan *Diversification Strategies* merupakan kombinasi dari persamaan di atas. persamaan tersebut dapat dilihat pada persamaan 2.5

$$s(r_i, r_j) = a_1 s_{colour}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j) \quad (2.5)$$

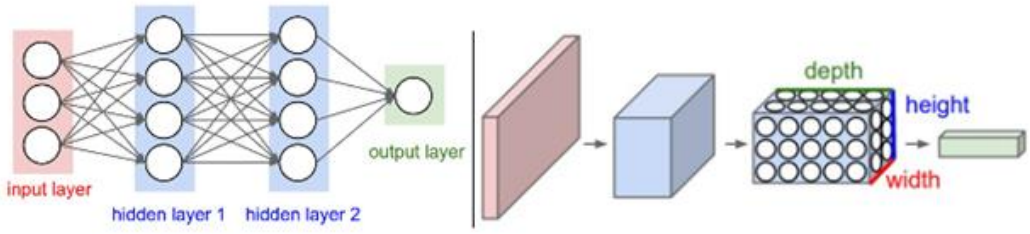
2.6.1.3 Combining Locations

Combining Locations merupakan proses menggabungkan objek hipotesis dari beberapa variasi algoritma pengelompokan hirarkis dengan memesan objek hipotesis sedemikian rupa sehingga lokasi yang paling mungkin menjadi objek yang didahulukan. dalam memesan hipotesis objek gabungan yang ditetapkan berdasarkan urutan hipotesis yang dihasilkan dalam setiap strategi pengelompokan individu. Namun, saat menggabungkan hasil hingga 80 strategi yang berbeda, urutan seperti itu akan terlalu menekankan wilayah besar. Untuk mencegah hal ini, maka strategi pengelompokan j diabiarkan r_i^j menjadi wilayah yang dibuat pada posisi i dalam hierarki, di mana $i = 1$ mewakili atas hierarki (yang wilayah terkait

mencakup lengkap gambar). untuk menghitung nilai posisi v_i^j sebagai $RND\ x\ i$, di mana RND adalah angka acak dalam kisaran [0,1]. Peringkat akhir diperoleh dengan memesan wilayah menggunakan v_i^j. untuk menggunakan lokasi dalam hal kotak pembatas, maka pertama-tama memberi peringkat pada semua lokasi. kemudian memfilter duplikat peringkat lebih rendah. Ini memastikan bahwa kotak duplikat memiliki peluang yang lebih baik untuk memperoleh peringkat tinggi. Ini diinginkan karena jika beberapa strategi pengelompokan menyarankan lokasi kotak yang sama, itu kemungkinan berasal dari bagian gambar yang jelas secara visual[9].

2.6.2 *Convolutional Neural Network* (CNN)

Convolution Neural Network merupakan salah satu kelas *deep feed-forward artificial neural network* yang banyak diaplikasikan pada analisis citra. CNN terdiri dari satu lapisan masukan (input layer), satu lapisan keluaran (output layer), dan sejumlah lapisan tersembunyi (hidden layer). lapisan tersembunyi umumnya berisi *Convolutional layer*, *pooling layer*, *relu layer*, *fully connected layer*, dan *loss layer*. arsitektur CNN secara umum dapat dilihat pada Gambar 2.4 :



Gambar 2.5 Arsitektur CNN Secara Umum

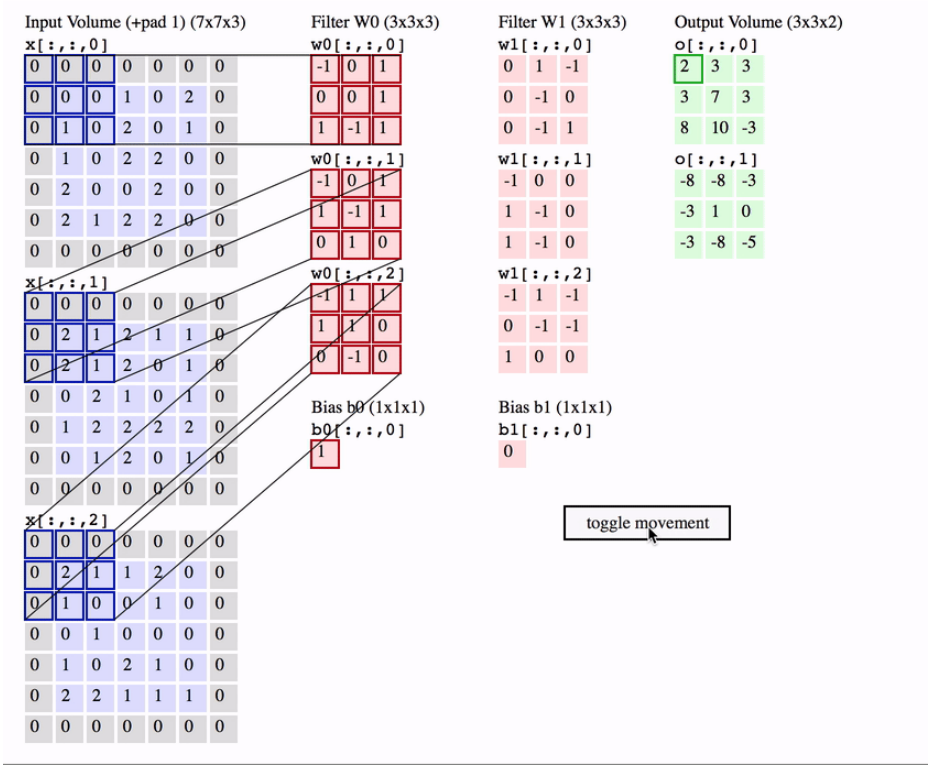
2.6.2.1 Convolutional Layer

Lapisan Konvolusional *(convolutional layer*) merupakan blok bangunan inti CNN, di mana sebagian besar komputasi dilakukan di lapisan ini *Convolutional layer* yang melakukan produk titik dari Matrikss kernel dengan bagian dari gambar dan menghasilkan *output*. Ini diikuti dengan menggeser dan mengulangi operasi yang sama pada gambar yang lengkap dan disebut konvolusi. Wilayah input yang diambil untuk produk titik disebut bidang reseptif dari lapisan konvolusi. Di setiap lapisan konvolusi, ada satu set kernel dan mereka secara kolektif disebut filter.

Input untuk lapisan konvolusi adalah array n-dimensi, artinya *input* adalah gambar dari bentuk Lebar x Tinggi x Kedalaman. Misalnya, jika kita memiliki

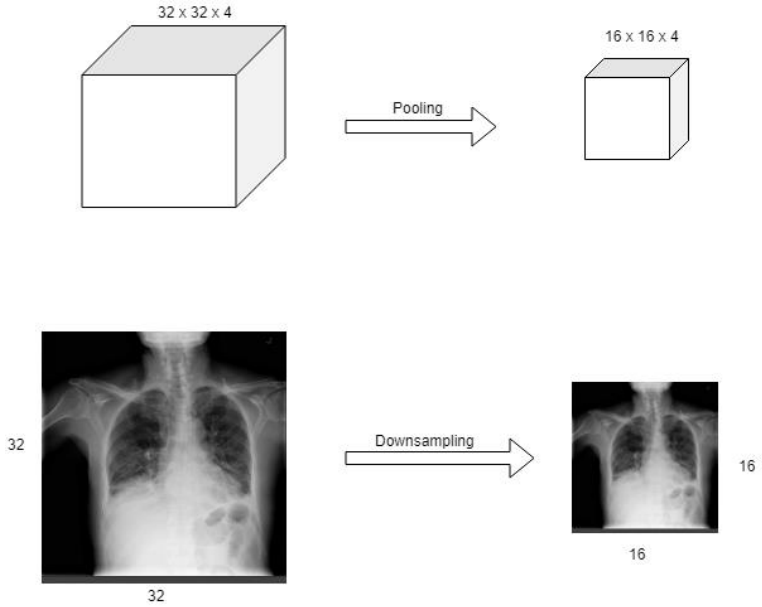
gambar skala abu-abu dengan ukuran 32 x 32, lebar dan tinggi, maka inputnya adalah 32 x 32 x 1 di mana kedalaman adalah jumlah saluran warna dalam kasus ini, dan diwakili oleh dimensi ketiga. Demikian pula, untuk gambar berwarna ukuran 512, inputnya adalah 512 x 512 x 3. Semua kernel di filter juga memiliki kedalaman yang sama dengan *input*.

Parameter untuk layer adalah jumlah filter, ukuran filter, langkah, dan nilai padding. Dari jumlah tersebut, nilai filter adalah satu-satunya parameter yang bisa dipelajari. Langkah mengacu pada jumlah pergeseran piksel untuk sebuah kernel. Dengan langkah 1, kernel dipindahkan ke kiri sebesar 1 piksel dan produk titik diambil dengan wilayah input yang sesuai. Dengan langkah 2, kernel digerakkan 2 piksel dan operasi yang sama terjadi. Pada setiap input, pada batas, kernel hanya bisa tumpang tindih dengan wilayah tertentu di dalam gambar. Batas karena itu diisi dengan nol untuk kernel untuk menangkap wilayah gambar lengkap. Nilai padding mengatur cara untuk memberi batas pada Gambar[11]. proses *convolutional layer* dapat dilihat pada Gambar 2.5 :



Gambar 2.6 Proses Convolutional Layer

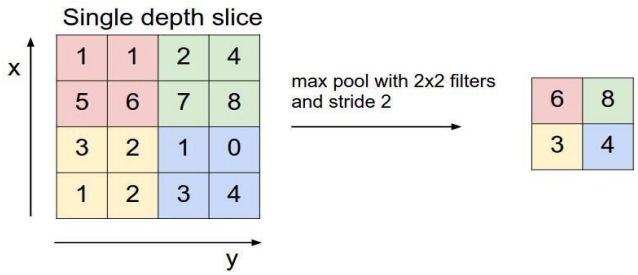
2.6.2.2 Pooling Layer



Gambar 2.7 proses Pooling layer

Pooling mengambil suatu wilayah *input* dan nilai maksimum atau rata-rata dari wilayah tersebut dihasilkan sebagai *output*. Akibatnya, ini mengurangi ukuran *input* dengan pengambilan sampel di wilayah lokal. Lapisan ini disisipkan antara 2 hingga 3 lapisan konvolusi untuk mengurangi resolusi *output*, sehingga mengurangi persyaratan untuk parameter. *Pooling Layer* merupakan fungsi untuk menjaga ukuran data ketika *convolution*, yaitu dengan melakukan *downsampling*. dengan *pooling*, kita dapat merepresentasikan data menjadi lebih kecil, mudah dikelola dan mudah mengontrol *overfitting*. proses *pooling* dapat dilihat pada Gambar 2.6 :

proses *Pooling* yang umum digunakan adalah max *Pooling*, memilih nilai maksimum dalam suatu area tertentu, proses *max pooling* dapat dilihat pada Gambar 2.7 :



Gambar 2.8 Operasi Max Pooling

2.6.2.3 *Relu Layer*

Aktivasi *ReLU* (*Rectified inear Unit*) merupakan lapisan aktivasi pada model CNN yang mengaplikasikan fungsi $f(x) = \max(0, x)$ yang berarti fungsi ini melakukan *Thresholding* dengan nilai nol terhadap nilai *piksel* pada *input* citra. Aktivasi ini membuat seluruh nilai piksel yang bernilai kurang dari nol pada suatu citra akan dijadikan 0.

$$f(x) = \max(0, x) \tag{2. 6}$$

2.6.2.4 Fully Connected Layer

Lapisan *Fully-Connected* adalah lapisan di mana semua neuron aktivasi dari lapisan sebelumnya terhubung semua dengan neuron di lapisan selanjutnya seperti halnya jaringan syaraf tiruan biasa. Setiap aktivasi dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua neuron di lapisan *Fully-Connected*. Lapisan *Fully-Connected* biasanya digunakan pada metode Multi Lapisan Perceptron dan bertujuan untuk mengolah data sehingga bisa diklasifikasikan.

Perbedaan antara lapisan *Fully-Connected* dan lapisan konvolusi biasa adalah neuron di lapisan konvolusi terhubung hanya ke daerah tertentu pada input, sementara lapisan *Fully-Connected* memiliki neuron yang secara keseluruhan terhubung. Namun, kedua lapisan tersebut masih mengoperasikan produk dot, sehingga fungsinya tidak begitu berbeda.

Fully Connected Layer pada intinya adalah sebuah neural *network multilayer perceptron* (MLP), yang memiliki beberapa hidden layer, *activation function*, output layer dan *loss function. Fully connect*ed layer-lah yang akan berperan untuk mengklasifikasi data masukan. Bentuk persamaan *hidden* layer dan output layer sebagai berikut.

Hidden layer:

$$z\_in_i = \sum_{i=1}^{n} X_j * V_{j,i} + V_{0,i} \tag{2. 7}$$

$z\_in_i =$ masukan untuk *node hidden layer z ke i* dengan jumlah node n

$X_j =$ node *X ke j*

$V_{j,i} =$ weight V untuk node X_j dan node Z_i

$V_{0,i}$ = bias V untuk node z_in_i

Output layer:

$$y\_in_i = \sum_{i=1}^{m} Z_j * W_{j,i} + W_{0,i} \quad (2.8)$$

y_in_i = masukan untuk *node hidden layer z ke i* dengan jumlah node m

Z_j = node Z *ke* j

$W_{j,i}$ = weight W untuk node Z_j dan node Y_i

$W_{0,i}$ = bias W untuk node W_in_i

Output yang dihasilkan dari pooling layer masih berbentuk multidimensional array, sehingga akan di-flatten terlebih dahulu untuk mengubah data menjadi vektor sebelum dijadikan *input* untuk fully connected layer

2.6.2.5 Loss Layer

Loss layer merupakan lapisan terakhir dalam CNN, menentukan bagaimana pelatihan memberikan penalti atas penyimpangan antara hasil prediksi dan label. terdapat sejumlah variasi *loss function*, salah satunya ialah *softmax classifier* yang digunakan untuk memprediksi satu dari sejumlah kelas yang saling eksklusif. *Softmax Classifier* merupakan bentuk lain dari algoritma *Logistic Regression* yang dapat kita gunakan untuk mengklasifikasi lebih dari dua kelas. Standar klasifikasi yang biasa dilakukan oleh algoritma *Logistic Regression* adalah tugas untuk klasifikasi kelas biner. Pada *i* bentuk persamaan yang muncul adalah sebagai berikut.

$$Y_i = \frac{e^{y\_in_i}}{\sum_{i=1}^{m} e^{M}} \quad (2.9)$$

Y_i= keluaran untuk node output layer ke-i

y_in_i = masukan untuk node output layer ke-i

M = semua masukan untuk node output layer, berjumlah m buah

Softmax juga memberikan hasil yang lebih intuitif dan juga memiliki interpretasi probabilistic yang lebih baik dibanding algoritma klasifikasi lainnya. Softmax memungkinkan kita untuk menghitung probabilitas untuk semua label. Dari label

yang ada akan diambil sebuah vektor nilai bernilai riil dan merubahnya menjadi vector dengan nilai antara nol dan satu yang bila semua dijumlah akan bernilai satu.

2.6.2.6 Backpropagation

Backpropagation adalah salah satu algoritma *supervised learning* yang digunakan dalam *artificial neural networks*. *Backpropagation* mencari kombinasi bobot untuk meminimalkan kesalahan output untuk dianggap menjadi solusi yang benar. Secara sederhana *Backpropagation* dilakukan dalam dua tahap, yaitu:

1. *Feedfoward*

Pola yang akan dilatih diset ke setiap unit di *input layer*, lalu *output* yang dihasilkan ditransmisikan ke *layer* selanjutnya terus sampai *output layer*.

2. *Backpropagation*

Berdasarkan *output* yang diharapkan, setiap bobot disesuaikan agar menghasilkan *error* yang minimal mulai dari bobot yang terhubung ke *output neuron*, lalu terus mundur sampai ke *input layer*. Berikut tahap-tahap *backpropagation*.

1. Perhitungan *loss function* (*Cross Entropy Loss*)

$$\mathrm{L} = -\sum_i^m t_i \log(Y_i) \qquad (2.11)$$

2. Perhitungan gradien kesalahan terhadap parameter bobot W_{ji} menggunakan rumus *chain rule.*

$$\frac{\partial L}{\partial W_{ji}} = \frac{\partial L}{\partial Y_i}\frac{\partial Y_i}{\partial y\_in_i}\frac{\partial y\_in_i}{\partial W_{ji}}$$

$$\frac{\partial L}{\partial Y_i} = \frac{Y_i - t_i}{Y_i(1 - Y_i)}$$

$$\frac{\partial Y_i}{\partial y\_in_i} = Y_i(1 - Y_i)$$

$$\frac{\partial y\_in_i}{\partial W_{ji}} = Z_j$$

$$\frac{\partial L}{\partial W_{ji}} = (Y_i - t_i)Z_j \qquad (2.12)$$

3. Perhitungan gradien kesalahan terhadap parameter bobot V_{kj} menggunakan rumus *chain rule*.

$$\frac{\partial L}{\partial V_{kj}} = \sum_{i}^{m} \frac{\partial L}{\partial Y_i} \frac{\partial Y_i}{\partial y\_in_i} \frac{\partial y\_in_i}{\partial Z_j} \frac{\partial Z_j}{\partial z\_in_j} \frac{\partial z\_in_j}{\partial V_{kj}}$$

$$\frac{\partial L}{\partial Y_i} = \frac{Y_i - t_i}{Y_i(1 - Y_i)}$$

$$\frac{\partial Y_i}{\partial y\_in_i} = Y_i(1 - Y_i)$$

$$\frac{\partial y\_in_i}{\partial Z_j} = W_{ji}$$

$$\frac{\partial Z_j}{\partial z\_in_j} = Z_j\left(1 - Z_j\right)$$

$$\frac{\partial z\_in_j}{\partial V_{kj}} = X_k$$

$$\frac{\partial L}{\partial V_{kj}} = \sum_{i}^{m}(Y_i - t_i)(W_{ji})(Z_j\left(1 - Z_j\right))(X_k) \qquad (2.13)$$

4. *Update* nilai parameter filter W

$$W_{1,1}{}' = W_{1,1} - \alpha(\frac{\partial L}{\partial W_{1,1}}) \qquad (2.14)$$

Parameter filter F

$$\mathrm{F}[1]_{1,1}{}' = \mathrm{F}[1]_{1,1} - \alpha(\frac{\partial \mathrm{P}[1]}{\partial \mathrm{F}[1]_{1,1}}) \qquad (2.15)$$

Parameter filter V

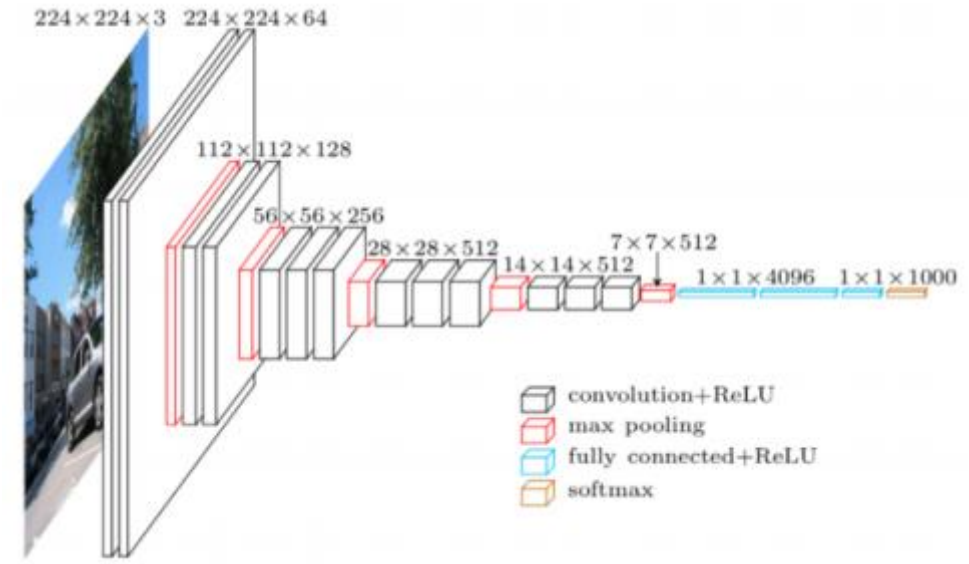
$$V_{1,1}{}' = V_{1,1} - \alpha(\frac{\partial L}{\partial V_{1,1}}) \qquad (2.16)$$

2.6.2.7 Arsitektur CNN

VGG-16 ada beberapa model pra-terlatih yang mudah diakses tersedia dan telah digunakan cukup banyak bahwa ada tolak ukur yang kuat untuk perbandingan. Selain itu, VGG-16 secara langsung berdasarkan *AlexNet* CNN arsitektur yang digunakan dalam R-CNN [11]. Tampaknya tepat untuk menggunakan model awal dalam rangka menghasilkan hasil yang berlaku dan relevan dalam penelitian ini.

Jaringan telah dilatih terlebih dahulu pada dataset kelas 1000 ImageNet. Ini untuk menghitung fitur dari wilayah gambar proposal objek, pertama akan dikonversi ke 224 × 224 ukuran *input* yang dibutuhkan oleh CNN. Ada sejumlah kemungkinan untuk mengubah *image* region, termasuk Scaling kemudian *center-cropping* dan padding ke aspek rasio persegi. Selain itu, R-CNN menambahkan

cukup *padding* untuk sedemikian rupa sehingga hasil yang melengkung persis 16 piksel dari *padding* pada semua sisi, VGG-16 memiliki 16 lapisan bobot dan 5 lapisan *max polling layer*[12], terlihat pada Gambar 2.9.

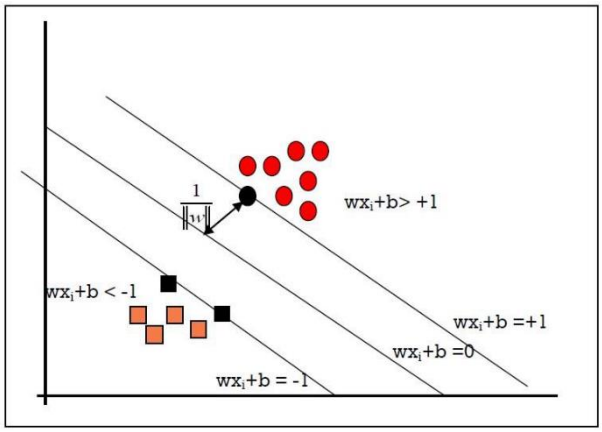


Gambar 2.9 Arsitektur CNN VGG-16

Filter *convolutional* dipelajari melalui *Mini-batch Gradient Descent*, dan beberapa lapisan 3 × 3 filter diikuti oleh ReLU *non-linearity* mendahului masing-masing operasi *max polling layer*. *Network layer* diberi nama berdasarkan jenis dan posisi ditumpuk, sehingga pool5 mengacu pada kelima *max polling layer*, FC6 adalah layer yang terhubung sepenuhnya langsung mengikuti pool5, dan conv5 2 adalah lapisan *convolutional* kedua dalam tumpukan langsung sebelum pool5.

2.6.3 *Support Vector Machine* (SVM)

Support Vector Machine (SVM) merupakan salah satu metode *machine learning* yang memaksimumkan akurasi prediksi dengan mencari bidang pembatas (*hyperplane*) terbaik dari dua kelas dalam ruang fitur. Metode SVM membutuhkan data latih positif dan negatif. Data latih positif dan negatif ini dibutuhkan SVM



Gambar 2. 10 Mencari Fungsi Pemisah Yang Optimal Untuk Obyek Yang Bisa Dipisahkan Secara Linier

untuk membuat keputusan terbaik dalam memisahkan data positif dengan data negatif di ruang n-dimensi atau pembatas (*hyperplane*). Metode ini lebih dikenal dengan metode klasifikasi *supervised learning* untuk mencari garis pemisah *hyperplane* dengan mengoptimalkan *hyperplane*, dan memaksimalkan margin antara dua kelas. Data yang paling dekat dengan bidang pembatas disebut *support vector*[13].

Metode SVM dalam mengklasifikasikan data uji ke dalam kelas positif dan negatif. Untuk kelas positif diberi label +1 yang artinya data uji tersebut memiliki potensi terdapat anomali *pneumonia* , sedangkan untuk kelas negatif diberi label -1 yang artinya data uji tersebut memiliki potensi normal. Klasifikasi data uji dengan metode ini dilihat dari vektor data uji, jika nilai vektor melebihi nilai *hyperplane* maka data uji tersebut masuk ke dalam kelas positif dan jika tidak maka data uji tersebut masuk ke dalam kelas negatif.

Data pada ruang *input* (*input space*) *dataset* dinotasikan dengan $\{x_i, .., x_n\}$ sedangkan label kelas dinotasikan dengan $y_i \in \{-1, +1\}$ untuk i = 1, 2,…,n. Dimana n adalah banyaknya data. Diasumsikan kedua kelas -1 dan +1 dapat

terpisah secara linear bidang pembatas [14], maka persamaan bidang pembatasnya didefinisikan pada persamaan berikut :

$$w * x_i + b = 0 \quad (2.10)$$

Data xi yang terbagi ke dalam dua kelas, yang termasuk kelas -1 (sampel negative didefinisikan sebagai vektor yang memenuhi pertidaksamaan (2.11) berikut :

$$w * x_i + b < 0 \text{ untuk } y_i = -1 \quad (2.11)$$

Sedangkan yang termasuk kelas +1 (sampel positif) memenuhi pertidaksamaan (2.12) berikut:

$$w * x_i + b > 0 \text{ untuk } y_i = +1 \quad (2.12)$$

Di mana:

xi = data *input*

yi = label yang diberikan

w = nilai dari bidang normal

b = posisi bidang relatif terhadap pusat koordinat

Parameter w dan b adalah parameter yang akan dicari nilainya. Bila label data yi = -1, maka pembatas menjadi persamaan (2.13) berikut:

$$wx_i + b \leq -1 \quad (2.13)$$

Bila label data yi = +1, maka pembatas menjadi persamaan (2.14) berikut:

$$wx_i + b \geq +1 \quad (2.14)$$

Margin terbesar dapat dicari dengan cara memaksimalkan jarak antar bidang pembatas kedua kelas dan titik terdekatnya, yaitu 2/|w|. Hal ini dirumuskan sebagai permasalahan quadratic programming (QP) problem yaitu mencari titik minimal persamaan (2.15) dengan memperhatikan persamaan (2.16) berikut:

$$minimize \ \frac{1}{2}||w||^2 \quad (2.15)$$

$$y_i(wx_i + b) - 1 \geq 0, (i = 1, .., n) \quad (2.16)$$

Permasalahan ini ini dapat dipecahkan dengan berbagi teknik komputasi. Lebih mudah diselesaikan dengan mengubah persamaan (2.15) ke dalam fungsi *Lagrangian* pada persamaan (2.17), dan menyederhanakannya menjadi persamaan (2.18) berikut:

$$L(w,b,a) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} a_i(y_i((w^T x_i + b) - 1)) \quad (2.17)$$

$$L(w,b,a) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} a_i y_i(w^T x_i + b) + \sum_{i=1}^{n} a_i \quad (2.18)$$

Di mana ai adalah lagrange multiplier yang bernilai nol atau positif ($ai \geq 0$) dan $||w||2= wTw$. Nilai optimal dari persamaan (2.18) dapat dihitung dengan meminimalkan L terhadap w, b dan memaksimalkan L terhadap a. Dapat dilihat pada persamaan (2.19) sampai (2.21) berikut:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} a_i\, y_i x_i = 0 \quad (2.19)$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} a_i\, y_i = 0 \quad (2.20)$$

$$\frac{\partial L}{\partial a} = \sum_{i=1}^{n} a_i y_i(w^T x_i + b) - \sum_{i=1}^{n} a_i = 0 \quad (2.21)$$

Maka masalah *Lagrange* dapat dinyatakan pada persamaan (2.22) berikut:

$$Min\ L(w,b,a) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} a_i y_i(w^T x_i + b) - \sum_{i=1}^{n} a_i \quad (2.22)$$

Dengan memperhatikan persamaan (2.23) dan (2.24) berikut:

$$w = \sum_{i=1}^{n} a_i\, y_i x_i \quad (2.23)$$

$$\sum_{i=1}^{n} a_i y_i = 0 \quad (2.24)$$

Jadi persoalan pencarian bidang model atau bidang pemisah terbaik persamaan (2.22) dengan memaksimalkan L terhadap a_i, persamannya menjadi persamaan (2.25) dan berikut:

$$Max \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1,j=1}^{n} a_i a_j y_i {y_j}^T x_i x_j^T \quad (2.25)$$

Dengan memperhatikan persamaan (2.26) berikut:

$$\sum_{i=1}^{n} a_i y_i = 0,\ a_i \geq 0\ (i,j = 1,..,n) \quad (2.26)$$

Untuk mencari nilai x_i dapat dilakukan ketika sudah didapatkan nilai data masukan yang telah dinormalisasi dan diubah ke dalam bentuk format data svm, sedangkan nilai y_i merupakan label pada data masukan ditentukan oleh peraihan medali.

Pada data latih, untuk mendapatkan nilai a_i, langkah pertama adalah mengubah setiap data latih menjadi nilai vektor (*support vector*) = $\binom{x}{y}$. Kemudian nilai vektor dari setiap data latih dimasukkan ke persamaan (2.27) kernel *trick phi* berikut:

$$S_i = \phi \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sqrt{x_n^2 + y_n^2} - x + (x-y)^2 \\ \sqrt{x_n^2 + y_n^2} - y + (x-y)^2 \end{bmatrix} \quad (2.27)$$

Nilai x didapatkan dari persamaan (2.28) kernel linear untuk x berikut:

$$\sum_{i=1,j=1}^{n} x_i x_j^T\ ,(i,j = 1,..,n) \quad (2.28)$$

Nilai y didapatkan dari persamaan (2.29) kernel linear untuk y berikut:

$$\sum_{i=1,j=1}^{n} y_i y_j^T\ ,(i,j = 1,..,n) \quad (2.29)$$

Untuk mendapatkan jarak tegak lurus yang optimal dengan mempertimbangkan vektor positif, maka hasil perhitungan dari substitusi nilai x dan nilai y ke persamaan (2.27) diberi nilai bias = 1 . Kemudian cari parameter a_i, dengan terlebih dahulu mencari nilai fungsi setiap data latih menggunakan persamaan (2.30) lalu mencari nilai a_i pada persamaan linear menggunakan persamaan (2.31) dengan memperhatikan $i,j = 1,..,n$ berikut:

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j \quad (2.30)$$

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j = y_i \quad (2.31)$$

Setelah parameter a_i didapatkan, kemudian masukan ke persamaan (2.32) berikut:

$$\widetilde{W} = \sum_{i=1}^{n} a_i S_i \quad (2.32)$$

Hasil yang didapatkan menggunakan persamaan, selanjutnya digunakan persamaan untuk mendapatkan nilai w dan b:

$$y = wx + b \quad (2.33)$$

Sedemikian sehingga didapatkanlah nilai w dan nilai b sebagai model fitur untuk mengklasifikasikan kedua kelas.

Maka masalah *Lagrange* dapat dinyatakan pada persamaan berikut:

$$Min\ L(w,b,a) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} a_i y_i (w^T x_i + b) - \sum_{i=1}^{n} a_i \quad (2.34)$$

Dengan memperhatikan persamaan berikut:

$$w = \sum_{i=1}^{n} a_i\, y_i x_i \quad (2.35)$$

$$\sum_{i=1}^{n} a_i y_i = 0 \quad (2.36)$$

Jadi persoalan pencarian bidang model atau bidang pemisah terbaik persamaan dengan memaksimalkan L terhadap a_i, persamannya menjadi persamaan berikut:

$$Max \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1,j=1}^{n} a_i a_j y_i {y_j}^T x_i x_j^T \quad (2.37)$$

Dengan memperhatikan persamaan berikut:

$$\sum_{i=1}^{n} a_i y_i = 0,\ a_i \geq 0\ (i,j = 1,..,n) \quad (2.38)$$

Untuk mencari nilai x_i dapat dilakukan ketika sudah didapatkan nilai data masukan yang telah dinormalisasi dan diubah ke dalam bentuk format data SVM, sedangkan nilai y_i merupakan label pada data masukan ditentukan oleh anomali *Pneumonia*.

Pada data latih, untuk mendapatkan nilai a_i, langkah pertama adalah mengubah setiap data latih menjadi nilai vektor (*support vector*) = $\binom{x}{y}$. Kemudian nilai vektor dari setiap data latih dimasukkan ke persamaan kernel *trick phi* berikut:

$$S_i = \phi \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sqrt{x_n^2 + y_n^2} - x + (x - y)^2 \\ \sqrt{x_n^2 + y_n^2} - y + (x - y)^2 \end{bmatrix} \quad (2.\ 39)$$

Nilai x didapatkan dari persamaan kernel linear untuk x berikut:

$$\sum_{i=1,j=1}^{n} x_i x_j^T \,, (i, j = 1, .., n) \quad (2.\ 40)$$

Nilai y didapatkan dari persamaan kernel linear untuk y berikut:

$$\sum_{i=1,j=1}^{n} y_i y_j^T \,, (i, j = 1, .., n) \quad (2.\ 41)$$

Untuk mendapatkan jarak tegak lurus yang optimal dengan mempertimbangkan vektor positif, maka hasil perhitungan dari substitusi nilai x dan nilai y ke persamaan diberi nilai bias = 1 [13]. Kemudian cari parameter a_i, dengan terlebih dahulu mencari nilai fungsi setiap data latih menggunakan persamaan lalu mencari nilai a_i pada persamaan linear menggunakan persamaan dengan memperhatikan $i, j = 1, .., n$ berikut:

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j \quad (2.\ 42)$$

$$\sum_{i=1,j=1}^{n} a_i S_i^T S_j = y_i \quad (2.\ 43)$$

Setelah parameter a_i didapatkan, kemudian masukan ke persamaan berikut:

$$\widetilde{W} = \sum_{i=1}^{n} a_i S_i \quad (2.\ 44)$$

Hasil yang didapatkan menggunakan persamaan, selanjutnya digunakan persamaan untuk mendapatkan nilai w dan b:

$$y = wx + b \quad (2.\ 45)$$

Sedemikian sehingga didapatkanlah nilai w dan nilai b sebagai model fitur untuk mengklasifikasikan kedua kelas.

2.6.4 *Bounding-box regression*

Bounding-box regression merupakan tahap regresi kotak-terikat sederhana untuk meningkatkan kinerja pelokalan. Setelah menilai setiap proposal pencarian selektif dengan deteksi kelas khusus SVM, untuk memperkirakan kotak pembatas baru untuk deteksi menggunakan regresi kotak pembatas khusus kelas. Ini serupa dalam semangat dengan regresi kotak-terikat yang digunakan dalam model bagian

yang dapat dideformasi .Perbedaan utama antara dua pendekatan adalah bahwa di sini kita mundur dari fitur yang dihitung oleh CNN, daripada dari fitur geometris yang dihitung pada lokasi bagian DPM yang disimpulkan.

Untuk *input* algoritma pelatihan N menggunakan satu set pasangan pelatihan$\{(P^i, G^i)\}_{i=1,..,N}$,dimana $P^i = (P_x^i, P_y^i, P_w^i, P_h^i)$ menentukan koordinat piksel dari pusat proposal P^i kotak pembatas bersama dengan lebar dan tinggi P^i dalam piksel. Setiap ground-truth bounding box G ditentukan dalam sama cara:$G = G_x, G_y, G_w, G_h$. untuk mempelajari transformasi yang memetakan kotak P yang diusulkan ke *ground-truth bounding box* G.

Untuk melakukan parameterisasi transformasi proposal P memiliki empat fungsi $D_x(P), D_y(P), D_w(P)$, dan $D_h(P)$. Pertama menentukan dua *translation* skala-invarian dari pusat Kotak pembatas P, sedangkan yang kedua menentukan ruang log *translation* lebar dan tinggi kotak pembatas P. setelah mendapatkan fungsi-fungsi ini selanjutnya mengubah input proposal P ke dalam *ground-truth box* G dengan menerapkan persamaan transformasi berikut :

$$G_x = P_w d_x(P) + P_x \tag{2.46}$$

$$G_y = P_h d_y(P) + P_y \tag{2.47}$$

$$G_w = P_w \exp(d_w(P)) \tag{2.48}$$

$$G_h = P_h \exp(d_h(P)) \tag{2.49}$$

Setiap fungsi $d_*(P)$ (di mana * Adalah salah satu dari x, y, h, w) merupakan pemodelan sebagai fungsi linier dari fitur pool5 proposal P, dilambangkan dengan $\emptyset_5(P)$. (Ketergantungan $\emptyset_5(P)$. Pada citra thorax diasumsikan secara implisit), maka $\emptyset_*(P) = w_*^t \emptyset_5(P)$ dimana w_* adalah vektor parameter model dengan mengoptimalkan yang diatur tujuan kuadrat terkecil (regresi ridge), maka dapat disimpulkan dalam persamaan (2.50).

$$w_* = \underset{\widehat{w}}{argmin} \sum_i^n (t_*^i - \widehat{w}_*^T \ \emptyset_5(P))^2 + \lambda \left\| \widehat{W}_* \right\|^2 \tag{2. 50}$$

Target regresi t_* untuk pasangan pelatihan (P, G) didefinisikan sebagai

$$t_x = (G_x - P_x)/P_w \tag{2.51}$$

$$t_y = (G_y - P_y)/P_h \quad (2.52)$$

$$t_w = \log(G_w/P_w) \quad (2.53)$$

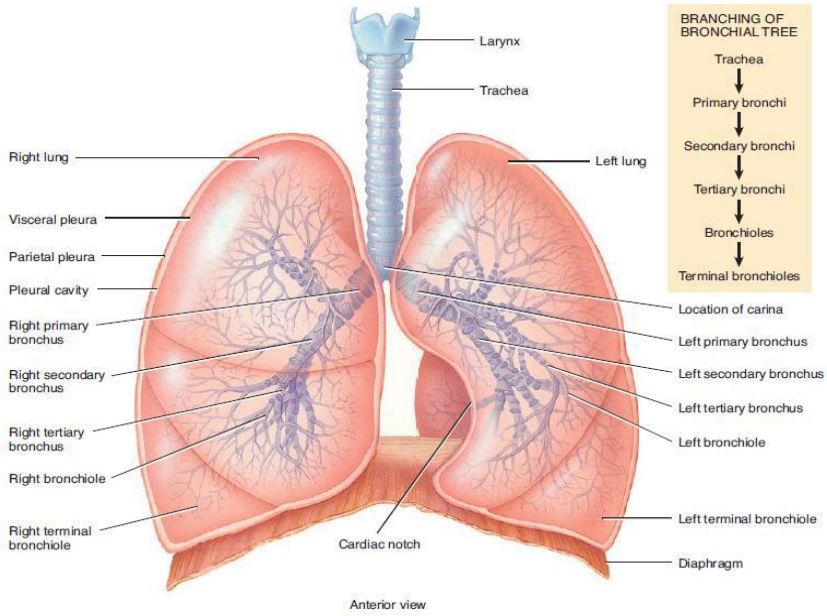
$$t_h = \log(G_h/P_h) \quad (2.54)$$

Sebagai masalah standar kuadrat terkecil yang diatur, ini bisa jadi diselesaikan secara efisien dalam bentuk tertutup. menerapkan "kedekatan" dengan menugaskan P ke kotak G-ground-kebenaran yang dengannya ia memiliki tumpang tindih IoU maksimum jika tumpang tindih lebih besar dari ambang batas Semua proposal yang tidak ditugaskan dibuang[2][15].

2.7 Anatomi Paru

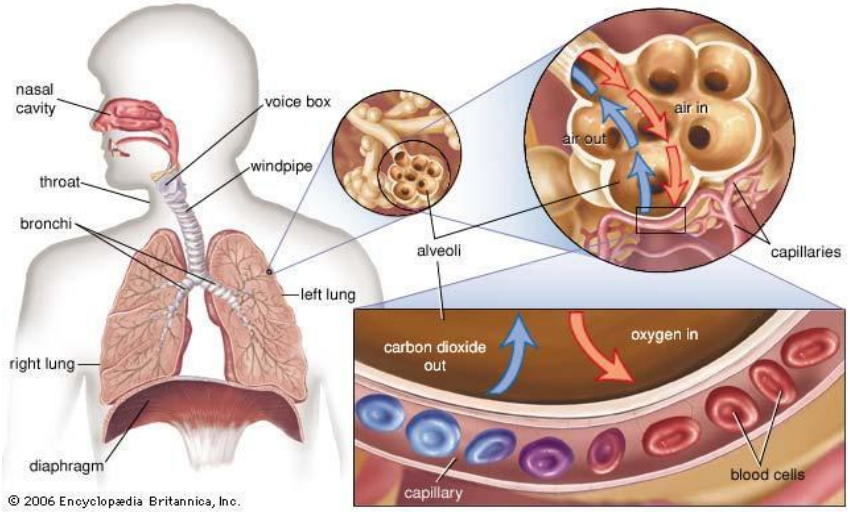
Paru merupakan organ sebagai alat pernapasan yang berbentuk seperti bunga karang besar yang terletak di dalam torak pada sisi lain jantung dan pembuluh darah besar. Paru-paru memanjang mulai dari akar leher menuju diagfragma dan secara kasar berbentuk kerucut dengan puncak di sebelah atas dan alas di sebelah bawah. Di antara paru-paru mediastinum, yang dengan sempurna memisahkan satu sisi rongga torasik sternum di sebelah depan. Di dalam mediastinum terdapat jantung, dan pembuluh darah besar, trakea dan esofagus, dustuk torasik dan kelenjar timus.Paru-paru dibagi menjadi lobus-lobus. Paru-paru

sebelah kiri mempunyai dua lobus, yang dipisahkan oleh belahan yang miring. Lobus superior terletak di atas dan di depan lobus inferior yang berbentuk kerucut. Paru-paru sebelah kanan mempunyai tiga lobus. Lobus bagian bawah dipisahkan oleh fisura oblik dengan posisi yang sama terhadap lobus inferior kiri. Sisa paru lainnya dipisahkan oleh suatu fisura horisontal menjadi lobus atas dan lobus tengah. Setiap lobus selanjutnya dibagi menjadi segmen-segmen yang disebut bronko-pulmoner, mereka dipisahkan satu sama lain oleh sebuah dinding jaringan koneknif masing-masing satu arteri dan satu vena. Masing-masing segmen juga dibagi menjadi unit-unit yang disebut lobulus[16]. Adapun anatomi paru dapat dilihat pada Gambar 2.11 Anatomi Paru :



Gambar 2.11 Anatomi Paru

Fungsi utama paru adalah sebagai alat pernapasan yaitu melakukan pertukaran udara (ventilasi), yang bertujuan menghirup masuknya udara dari atmosfer ke dalam paru-paru (inspirasi) dan mengeluarkan udara dari alveolar ke luar tubuh (ekspirasi).Adapun sistem saluran pernapasan dapat dilihat pada Gambar 2.9 saluran pernapasan manusia.



Gambar 2.12 Saluran Pernapasan Manusia

Secara anatomi, fungsi pernapasan ini dimulai dari hidung sampai ke parenkim paru. Secara fungsional saluran pernapasan dibagi atas bagian yang berfungsi sebagai konduksi (pengantar gas) dan bagian yang berfungsi sebagai respirasi

(pertukaran gas). Pernapasan dapat berarti pengangkutan oksigen (O2) ke sel dan pengangkutan CO2 dari sel kembali ke atmosfer[16]. Proses saluran pernapasan terdiri dari 4 tahap yaitu sebagai berikut :

1. Pertukaran udara paru, yang berarti masuk dan keluarnya udara ke dan dari alveoli. Alveoli yang sudah mengembang tidak dapat mengempis penuh, karena masih adanya udara yang tersisa di dalam *alveoli* yang tidak dapat dikeluarkan walaupun dengan ekspirasi kuat. Volume udara yang tersisa ini disebut volume residu. Volume ini penting karena menyediakan O2 dalam alveoli untuk mengoperasikan darah.
2. Difusi O2 dan CO2 antara alveoli dan darah.
3. Pengangkutan O2 dan CO2 dalam darah dan cairan tubuh menuju ke dan dari sel-sel.
4. Regulasi pertukaran udara dan aspek-aspek lain pernapasan

Dari aspek fisiologis, ada dua macam pernapasan yaitu sebagai berikut :

1. Pernapasan luar (*eksternal respiration*) yaitu penyerapan O2 dan pengeluaran CO2 dalam paru-paru.
2. Pernapasan dalam (*internal respiration*) yang *aktifitas* utamanya adalah pertukaran gas pada metabolisme energi yang terjadi dalam sel.

2.7.1 *Pneumonia*

Pneumonia merupakan penyakit dari paru-paru dan sistem pernapasan di mana alveoli mikroskopik udara mengisi kantong dari paru yang bertanggung jawab untuk menyerap oksigen dari atmosfer) menjadi radang dan dengan penimbunan cairan.*Pneumonia* disebabkan oleh berbagai macam sebab,meliputi infeksi karena bakteri, virus, jamur atau parasit. *Pneumonia* juga dapat terjadi karena bahan kimia atau kerusakan fisik dari paruparu, atau secara tak langsung dari penyakit lain seperti kanker paru atau penggunaan alkohol. Gejala khas yang berhubungan dengan *pneumonia* meliputi batuk, nyeri dada demam, dan sesak nafas. Alat diagnosa meliputi sinar-x dan pemeriksaan sputum. Pengobatan tergantung penyebab dari *pneumonia*, *pneumonia* kerena bakteri diobati dengan antibiotika. *Pneumonia* merupakan penyakit yang umumnya terjadi pada semua kelompok umur, dan menunjukan penyebab kematian pada orang tua dan orang

dengan penyakit kronik. Tersedia vaksin tertentu untuk pencegahan terhadap jenis pnuemonia. Prognosis untuk tiap orang berbeda tergantung dari jenis *pneumonia*, pengobatan yang tepat, ada tidaknya komplikasi dan kesehatan orang tersebut dan Tes penting untuk mendeteksi *pneumonia* pada keadaan yang tidak jelas ialah dengan foto thorax. Foto thorax dapat menampakkan daerah opak(terlihat putih) yang menggambarkan konsolidasi. *Pneumonia* tidak selalu dilihat oleh sinar x. selain karena penyakitnya hanya pada tingkat permulaan atau karena mengenai bagian paru tertentu yang sulit dilihat dengan sinar x, Foto thorax juga digunakan untuk evaluasi adanya komplikasi dari *pneumonia*[16].

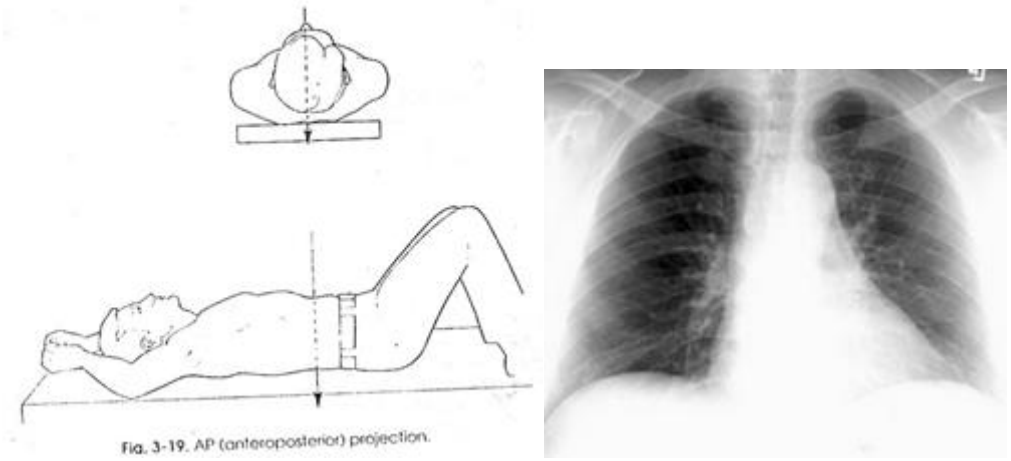
Pada pembacaan foto rontgen dada, pendekatan secara sistematis adalah penting, berdasarkan penilaian pertama pada anatomi dan selanjutnya fisiologi. Jantung mudah dibedakan dari paru-paru karena jantung lebih mengandung darah dengan densitas air lebih besar dibanding udara. Karena darah melemahkan x-ray lebih kuat dibanding udara, jantung relatif tampak berwarna putih dan paru-paru relatif hitam[8].

2.8 FOTO THORAX

Foto thorax atau sering disebut chest x-ray (CXR) adalah suatu proyeksi radiografi dari thorax untuk mendiagnosis kondisi-kondisi yang mempengaruhi thorax, isi dan struktur-struktur di dekatnya. Foto thorax menggunakan radiasi terionisasi dalam bentuk x-ray. Dosis radiasi yang digunakan pada orang dewasa untuk membentuk radiografi adalah sekitar 0.06 mSv.

Foto thorax digunakan untuk mendiagnosis banyak kondisi yang melibatkan dinding thorax, tulang thorax dan struktur yang berada di dalam kavitas thorax termasuk paru-paru, jantung dan saluran-saluran yang besar. *Pneumonia* dan gagal jantung kongestif sering terdiagnosis oleh foto thorax.

Pemeriksaan foto thorax Rontgenography Adalah pembuatan foto rontgen thorax, yang biasanya dibuat dengan arah postero-anterior (PA) dan lateral bila perlu. Agar distorsi dan magnifikasi yang diperoleh menjadi sekecil mungkin, maka jarak antara tabung dan film harus 1,80 meter dan foto dibuat sewaktu penderita sedang bernapas dalam (inspirasi maksimal). Adapun pemilihan proyeksi pada posisi foto thorax PA (Postero Anterior) Gambar 2.12 Proyeksi PA :



Gambar 2. 13 Proyeksi PA

2.9 Pemograman Berbasis Objek

Pemrograman berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek[17].

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemrograman, dan pengujian perangkat lunak. Berikut adalah beberapa konsep dasar yang harus dipahami pemrograman berorientasi objek [17]:

1. Kelas (*Class*)

Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (relationship) dan arti. Suatu kelas dapat diturunkan dan kelas semula dapat diwariskan ke kelas yang baru.

2. Objek (*Object*)

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

3. Metode (Method)

Operasi atau metode atau method pada sebuah kelas hampir sama dengan fungsi atau prosedur pada terstruktur. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

4. Atribut (*Attribute*)

Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5. Abstraksi (*Abstraction*)

Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*Encapsulation*)

Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja.

7. Pewarisan (*Inheritance*)

Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

8. Antarmuka (*Interface*)

Antarmuka atau interface sangat mirip dengan kelas, tetapi tanpa atribut kelas dan tanpa memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah interface dapat diimplementasikan oleh kelas lain.

9. *Reusability*

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi antar objek

Komunikasi antar-objek dilakukan lewat pesan (message) yang dikirim dan satu objek ke objek lainnya.

12. *Polimorfisme (Polymorphism*)

Kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13. Package

Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda [17].

2.10 *Unified Modeling Language*

UML (*Unified Modeling Language*) adalah salah satu standar Bahasa yang banyak digunakan didunia industri untuk mendefinisikan requirement, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. Ada beberapa diagram yang digunakan proses pembuatan perangkat lunak berorientasi objek di antaranya, *use case diagram, activity diagram, class diagram dan sequence diagram* [17].

2.10.1 *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Ada

dua hal utama pada use case yaitu pendefinisian apa yang disebut actor dan use case.

1) Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibangun.
2) *Use Case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor [17].

2.10.2 *Class Diagram*.

Diagram kelas atau *Class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas [17].

2.10.3 *Sequence Diagram*

Diagram Sekuen atau *Sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang di instansiasi menjadi objek itu.

Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka *diagram sekuen* yang harus dibuat juga semakin banyak. Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu [17].

2.10.4 *Activity Diagram*

Diagram Aktivitas atau *Activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada

perangkat lunak. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1) Rancangan proses bisnis di mana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2) Urutan atau pengelompokan tampilan dari sistem/user interface dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3) Rancangan pengujian di mana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4) Rancangan menu yang ditampilkan pada perangkat lunak [17].

2.11Software Pendukung

Pada penelitian yang dilakukan diperlukan *software* pendukung yang membantu dalam pembangunan aplikasi. Adapun *software* pendukung, diantaranya:

2.11.1 Python

Python adalah bahasa pemrograman *interpretatif multiguna*. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, *python* lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Hal ini membuat *Python* sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain.

Bahasa ini muncul pertama kali pada tahun 1991, dirancang oleh seorang bernama *Guido van Rossum*. Sampai saat ini *Python* masih dikembangkan oleh *Python Software Foundation*. Bahasa *Python* mendukung hampir semua sistem operasi, bahkan untuk sistem operasi *Linux*, hampir semua distronya sudah menyertakan *Python* di dalamnya.

Dengan kode yang simpel dan mudah diimplementasikan, seorang programmer dapat lebih mengutamakan pengembangan aplikasi yang dibuat, bukan malah sibuk mencari *syntax error*.

2.11.2 *Python Flask*

Flask adalah *microframework* untuk Bahasa pemrograman *Python* yang berbasis *Werkzeug* dan *Jinja* 2, sekaligus memiliki lisensi BSD yang berarti siapapun bisa menggunakan *flask* untuk berbagai keperluan baik pribadi maupun

komersial tanpa dipungut bayaran (*free software*) asalkan hak cipta *flask* tetap disertakan [18].

Keunggulan utama dari *framework flask* memang bisa digunakan untuk menerapkan konsep MVC, namun tidak memaksa pengguna (*strict*) untuk menerapkannya, maka developer tetap bias menggunakan paradigma pemrograman terstruktur.

Masalah utama dalam menggunakan *flask* sebagai *framework* adalah karena terlalu sederhana maka *developer* perlu menerapkan *desain* yang *scalable* dan *modular* (penulisan *script* berdasarkan modul-modul) ketika membuat program dengan *flask*, salah satu contoh terbaik adalah penerapan konsep MVC.

2.11.3 Python Library

Python library adalah kumpulan fungsi dan metode yang memungkinkan untuk melakukan banyak tindakan tanpa menulis kode.

1. *Scikit-learn*

Scikit-learn merupakan sebuah proyek yang dikerjakan oleh David Cournapeau pada tahun 2007 pada proyek Google Summer of Code, Beberapa tahun kemudian proyek ini dijadikan sebuah penelitian tesis oleh Matthieu Brucher [19].

Di tahun 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort dan Vincent Michel dari INRIA mengambil kepemimpinan proyek ini dan membuat rilis yang pertama untuk publik tepatnya Februari 1 Februari 2010[19].

Scikit-learn yaitu sebuah modul *Python* yang mengintegrasikan berbagai algoritma machine learning yang canggih untuk masalah berskala menengah yang diawasi ataupun tidak terawasi. pada modul ini berfokus pada machine learning ke non-spesialis menggunakan bahasa tingkat tinggi sebagai tujuan umum, yang bertujuan kemudahan dalam penggunaan, kinerja, dokumentasi, dan konsistensi API. yang di peruntukan untuk kademik dan pengaturan komersial yang memiliki ketergantungan yang minimal dan didistribusikan di bawah lisensi BSD yang disederhanakan[20] .

berikut paket yang dapat diimplementasikan pada liberary Scikit-learn :

a. Classification

Classification yaitu mengidentifikasi sebuah kategori mana yang dimiliki *object.* pada pengaplikasiannya yaitu *Spam detection*, Image *recognition* dengan berbagai *algoritma* yang terdapat pada paket ini diantaranya SVM, *nearest neighbors*, dan *random forest*[19].

b. Regression

Regression yaitu memprediksi sebuah atribut yang bernilai *kontinyu* yang terkait pada sebuah objek. pada pengaplikasiannya yaitu *Drug response* dan *Stock prices* dengan beberapa *algoritma* yang dapat diterapkan diantaranya SVR, *ridge regression* dan *Lasso*[19].

c. Clustering

Clustering yaitu pengelompokkan secara otomatis pada objek yang serupa yang terdapat pada data set. pada pengaplikasiannya yaitu *Customer segmentation, Grouping experiment outcomes* dengan algoritma yang sudah tersedia pada paket ini diantaranya k-*Means, spectral clustering*,*mean-shift* dll[19].

d. *Dimensionality reduction*

Dimensionality reduction yaitu dengan mengurangi jumlah variabel acak untuk dipertimbangkan. pada pengaplikasiannya yaitu untuk *Visualization* dan *Increased efficiency* dengan *algoritma* yang bisa diterapkan diantaranya PCA, *feature selection* dan *non-negative matrix factorization*[19].

e. Model selection

Model selection yaitu untuk membandingkan, *memvalidasi* dan memilih parameter dan model yang bertujuan untuk peningkatan pada parameter yang sesuai. berikut modul yang dapat diterapkan diantaranya *grid search, cross validation* dan *metrics*[19].

f. Preprocessing

Preprocessing yaitu pengekstraksi fitur dan normalisasi dengan beberapa modul yang dapat diterapkan diantaranya *preprocessing* dan *feature extraction*. pada pengaplikasiannya mengubah input data ke text dengan algoritma *machine learning*[19].

2. Tensorflow

Tensorflow yaitu perpustakaan perangkat lunak yang dikembangkan oleh Tim *Google Brain* dalam organisasi penelitian Mesin *Cerdas Google* yang bertujuan untuk *machine learning* dan penelitian dalam jaringan syaraf. perpustakaan ini menggabungkan aljabar komputasi teknik pengoptimalan kompilasi, mempermudah penghitungan banyak ekspresi matematis dimana masalahnya adalah waktu yang dibutuhkan untuk melakukan perhitungan.

Fitur utamanya meliputi:

1. Mendefinisikan, *mengoptimalkan,* dan menghitung secara *efisien ekspresi* matematis yang melibatkan *array multidimension* (*tensors*).
2. Pemrograman pendukung jaringan syaraf dalam dan teknik pembelajaran mesin.
3. Penggunaan GPU yang transparan, mengotomatisasi manajemen dan optimalisasi memori yang sama dan data yang digunakan. *Tensorflow* bisa menulis kode yang sama dan menjalankannya baik di CPU atau GPU. Lebih khususnya lagi, *Tensorflow* akan mengetahui bagian perhitungan yang harus dipindahkan ke GPU.
4. Skalabilitas komputasi yang tinggi di seluruh mesin dan kumpulan data yang besar.

3. OpenCV

OpenCV (*Open Computer Vision*) adalah sebuah API (*Application Programming Interface*) yang di rilis di bawah lisensi BSD yang dapat di pakai oleh siapa saja karena bersifat *open source* di peruntukan untuk komersial dan akademis. *OpenCV* sendiri di rancang untuk efisiensi komputasi yang berfokus pada aplikasi secara Real – time. Dapat di tulis dengan berbagai bahasa pemograman seperti C / C++ , *Phyton*, *Java* dan *Php* yang telah dioptimalkan. Pada *OpenCV* mampu dijalankan pada berbagai jenis *Flatform* seperti *Windows, Linux, Android* dan *Mac OS*. Pada perpustakaan dapat memanfaatkan *Multi – core* yang diaktifkan dengan *OpenCL* dan *CUDA* yang sedang dikembangkan pada saat ini. *OpenCV* sendiri ditulis secara asli pada bahasa pemograman C++ dan mempunyai template antarmuka yang bekerja dengan mulus dengan kontainer STL. *OpenCV* dibangun untuk media pembelajaran

pada bidang ilmu visi komputer yang menyediakan infrastruktur umum untuk mempercepat pada penggunaan persepsi mesin dalam produk komersial, memudahkan bisnis dalam memanfaatkan dan memodifikasi kode.

Pada *OpenCV* sudah ada 2500 algoritma yang sudah dioptimalkan yang mencangkup satu set lengkap visi komputer klasik dan algoritma pembelajaran pada mesin. Algoritma ini dapat digunakan untuk *Face Recognition*, *Face Detection*, *Face*/*Object Tracking* dan lain – lain.

4. *TF.Learn*

   TF.Learn adalah modul *Python* tingkat tinggi untuk *machine learning* terdistribusi di dalam *TensorFlow*. Ini menyediakan antarmuka gaya *Scikit*-learning yang mudah digunakan untuk menyederhanakan proses membuat, mengkonfigurasi, melatih, mengevaluasi, dan bereksperimen dengan model pembelajaran mesin.