

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Coronavirus atau yang sering disebut *covid-19* merupakan virus yang baru ditemukan sekitar akhir tahun 2019 pertama kali di Wuhan, China [20]. Virus ini menyerang sistem pernafasan, biasanya ditandai dengan gejala flu, demam, pilek, sakit tenggorokan. Virus ini akan cepat menular jika seseorang mempunyai riwayat penyakit lain yang bisa mempercepat pertumbuhan virus ini didalam tubuh seseorang. Virus ini sangat cepat menyebar di kota tersebut dalam hitungan beberapa hari yang mengakibatkan kota itu lumpuh dan terpaksa lockdown sementara agar penyebarannya tidak semakin parah. Namun, tidak lama kemudian virus tersebut sangat cepat menyebar ke seluruh negara lainnya, salah satunya Indonesia. Pada akhir Maret Indonesia terkonfirmasi dua orang positif *covid-19*, dan semakin hari kasus bertambah banyak. Data yang terdapat pada website *covid19.go* dikonfirmasi per tanggal 01 November yaitu 412.784 orang positif *covid-19*.

Social distancing atau jaga jarak merupakan salah satu upaya yang dilakukan untuk memutus rantai penyebaran *covid-19*, setiap orang harus berjarak kurang lebih satu hingga dua meter karena virus bisa menular lewat percikan air dari mulut dan hidung orang yang sedang berbicara, batuk, dan bersin. *Social distancing* ini dihimbau pemerintah untuk dipatuhi supaya rantai penyebarannya tidak semakin luas, *social distancing* biasanya diterapkan di mall, rumah sakit, tempat umum, dan lain sebagainya terutama tempat yang ramai agar tidak terjadi kerumunan [22].

Dalam penerapan *social distancing* mampu mencegah penularan virus, karena virus ini merupakan virus yang menular pemerintah Indonesia bersama Menteri Kesehatan telah mengeluarkan keputusan pada No.HK.01.07/MENKES.104/2020 tentang Penetapan Infeksi Corona Virus sebagai Penyakit dapat Menimbulkan Wabah dan Penanggulangannya. Namun masyarakat masih cukup banyak yang belum faham bagaimana cara mengatasi penyebaran virus ini, sehingga pemerintah membuat protokol kesehatan 5M (mencuci tangan, memakai masker, menjaga jarak, menjauhi kerumunan, dan mengurangi mobilitas)

ini dibuat agar mudah diingat masyarakat dan menerapkannya. Karena masih banyaknya masyarakat yang belum menerapkan *social distancing* dan kurangnya pengawasan yang dilakukan, maka diperlukan suatu program yang mampu membantu dalam pengawasan *social distancing* ini terutama di lingkungan yang ramai dan sering dikunjungi banyak orang.

Pendeteksian *social distancing* sebelumnya pernah diteliti oleh beberapa peneliti, salah satu hasil penelitian yang saya dapatkan yaitu berjudul “*The Visual Social Distancing Problem*” [28]. Pada penelitian ini menjelaskan bahwa *Visual Social Distancing* (VSD) akan dilakukan dan dengan cara memberitahu kepada masyarakat bisa jadi akan digunakan alarm peringatan ataupun label yang akan ditempel pada sekitar area tertentu. *Visual Social Distancing* (VSD) ini akan menggunakan penelitian dari bidang *Computer Vision* dan *Social Signal Processing*.

Maka penelitian ini akan mengusulkan untuk menggunakan algoritma *Faster Region based Convolutional Neural Network* (Faster R-CNN), algoritma ini merupakan salah satu algoritma yang cukup unggul untuk melakukan pendeteksian sebuah objek dan dapat dilakukan secara *real-time*, algoritma ini merupakan algoritma deteksi wajah berbasis wilayah terbaru yang menunjukkan hasil luar biasa pada pendeteksian objek dan juga kinerja *Faster R-CNN* ini akan bergantung pada kualitas data *training* yang cukup besar [6], [21]. Hasil akhir dari pengerjaan Tugas Akhir ini adalah sebuah program dengan *interface* yang dapat digunakan kapan saja dengan kamera untuk mendeteksi pelanggaran *Social Distancing*.

1.2 Rumusan Masalah

Adapun rumusan masalah dari penelitian ini adalah :

1. Bagaimana mendeteksi manusia pada sistem deteksi *social distancing* ?
2. Bagaimana mendeteksi *social distancing* pada ruang terbuka?
3. Bagaimana mengetahui pelanggaran *social distancing* pada ruang terbuka secara *real-time*?

1.3 Tujuan dan Manfaat

Tujuan dari penulisan penelitian ini antara lain :

1. Membuat sistem untuk mendeteksi manusia dengan algoritma *Faster R-CNN*.
2. Membuat sebuah sistem yang dapat mendeteksi pelanggaran social distancing secara efektif dan akurat.
3. Mengimplementasikan pendeteksian pelanggaran *social distancing* secara *real-time* dan akurat yang ditampilkan pada *website*.

1.4 Batasan Masalah

Batasan masalah dalam pembuatan penelitian ini adalah :

1. Metode untuk pendeteksian jarak yaitu *Euclidean Distance*.
2. Metode yang dipakai untuk pengenalan objek yaitu *Faster R-CNN*.
3. Pendeteksian berdasarkan postur tubuh manusia
4. Perhitungan jarak antar objek dihitung dari *center bounding box*.
5. Tempat pengujian dilakukan di luar ruangan pada sore hari.
6. Pelanggaran social distancing hanya dalam dua dimensi.
7. Dikatakan melakukan pelanggaran *social distancing* jika dua orang atau lebih berdekatan dengan jarak kurang dari satu meter.

1.5 Sistematika Penulisan

Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, rumusan masalah, batasan masalah, metode penelitian, dan sistematika penulisan.

Bab II Dasar Teori

Bab ini berisi penjelasan teori dasar yang digunakan dalam penyusunan tugas akhir.

Bab III Perancangan Sistem

Bab ini berisi gambaran umum sistem, desain umum, arsitektur umum, pembuatan model, alur kerja sistem, perangkat keras yang digunakan, perangkat lunak yang digunakan, perancangan website, dan parameter performansi.

Bab IV Pengujian dan Analisis

Bab ini berisi simulasi dari sistem yang telah dibuat serta pengujian terhadap hasil yang telah di implementasikan beserta analisisnya.

Bab V Kesimpulan dan Saran

Bab ini berisi tentang kesimpulan dari implementasi sistem dan hasil dari kesimpulan diambil saran untuk pengembangan penelitian selanjutnya agar sistem dapat lebih baik.

BAB II

DASAR TEORI

2.1 *Social Distancing*

Social distancing merupakan pembatasan jarak antar orang yaitu satu hingga dua meter agar tidak saling berdekatan, pembatasan jarak ini dapat dilakukan pada lingkungan yang terdapat banyak orang ataupun di beberapa tempat yang memungkinkan untuk dilakukan *social distancing*. *Social distancing* ini sangat dibutuhkan pada saat masa pandemi *Covid-19* yang bertujuan untuk memutus rantai penyebarannya [1].

2.2 *Pengolahan Citra*

Pengolahan citra merupakan metode untuk melakukan beberapa operasi pada gambar tujuannya untuk meningkatkan kualitas citra atau untuk mengekstrak beberapa informasi yang berguna di dalamnya [2]. Pada pemrosesan pengolahan citra terdapat tiga langkah yaitu [2] :

1. Mengimpor gambar melalui alat akuisisi gambar,
2. Menganalisis dan memanipulasi gambar,
 1. Keluaran yang hasilnya dapat berupa gambar atau laporan yang diubah berdasarkan analisis gambar.

Pada pengolahan citra terdapat dua metode yang digunakan yaitu pengolahan citra analog dan pengolahan citra digital. Pengolahan citra analog dapat digunakan untuk salinan berbentuk *hard copy*. Teknik pemrosesan gambar digital membantu manipulasi gambar digital dengan menggunakan komputer. Terdapat tiga fase yang harus dilalui semua jenis data saat menggunakan teknik digital adalah pra-pemrosesan, peningkatan, dan tampilan, ekstrasi informasi [2].

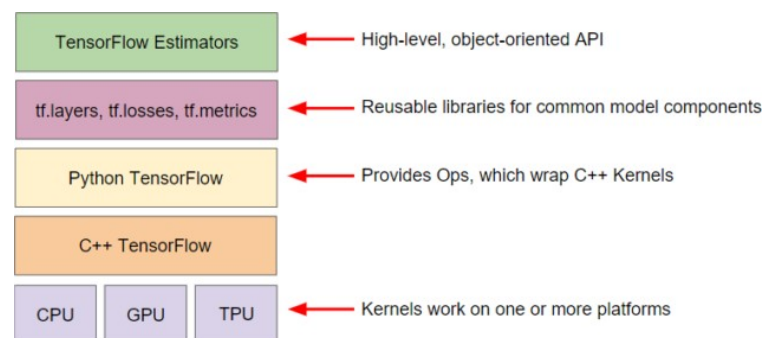
2.3 *Deep Learning*

Deep Learning merupakan salah satu cabang *machine learning* dan untuk menyelesaikan masalah pada *machine learning*. Sebuah model *machine learning* perlu ‘diberitahu’ bagaimana ia menciptakan prediksi akurat, dengan terus diberikan data. Sementara model *deep learning* dapat mempelajari metode komputasinya sendiri, dengan ‘otaknya’ sendiri, apabila diibaratkan. Sebuah model

deep learning dirancang untuk terus menganalisis data dengan struktur logika yang mirip dengan bagaimana manusia mengambil keputusan. Untuk dapat mencapai keputusan itu, deep learning menggunakan struktur algoritma berlapis yang disebut *artificial neural network* (ANN) [4].

2.4 *Tensorflow*

Tensorflow merupakan *computational framework* untuk membangun model *machine learning* seperti *neural networks*. *Tensorflow* dikembangkan oleh tim *Google Brain* untuk penggunaan *Google internal* dan dirilis di bawah lisensi open source Apache 2.0 pada 9 November 2015. *Tensorflow* adalah koleksi *software open source* untuk komputasi numerik yang menggunakan grafik aliran data (Toleubay, Y., James, A. P., 2020) [13].



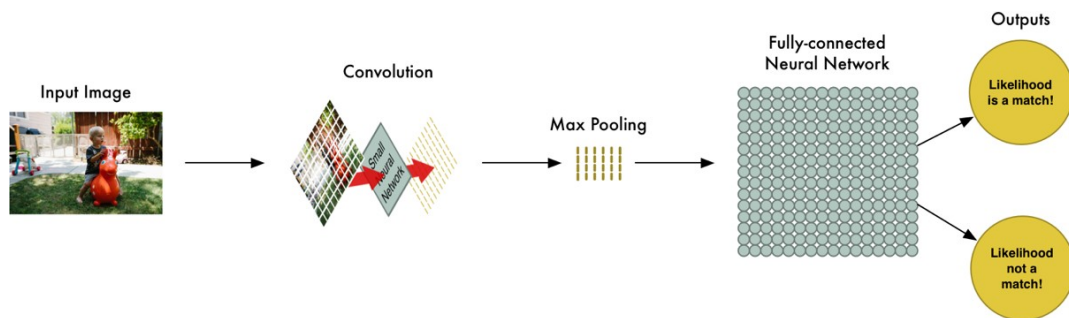
Gambar 2. 1 Tensorflow Toolkit Hierarchy [13]

2.5 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan jaringan syaraf yang digunakan khusus untuk mengolah data berstruktur grid, salah satunya berupa citra dua dimensi [7]. *Convolutional Neural Network* (CNN) adalah salah satu metode yang terdapat dalam *Deep Learning* yang dan banyak digunakan untuk menyelesaikan permasalahan yang berkaitan dengan *object detection* dan *image classification* [8].

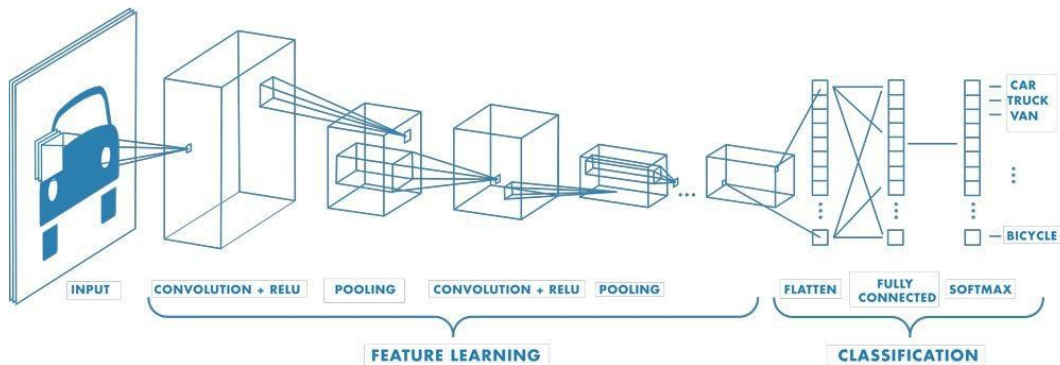
Secara garis besar *Convolutional Neural Network* (CNN) tidak jauh beda dengan neural network biasanya. CNN terdiri dari neuron yang memiliki weight, bias dan activation function. *Convolutional layer* juga terdiri dari neuron yang

tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (*pixels*) [14].



Gambar 2. 2 Cara Kerja CNN [14]

Arsitektur dari CNN dibagi menjadi 2 bagian besar, *Feature Extraction Layer* dan *Fully-Connected Layer* (MLP) [14].



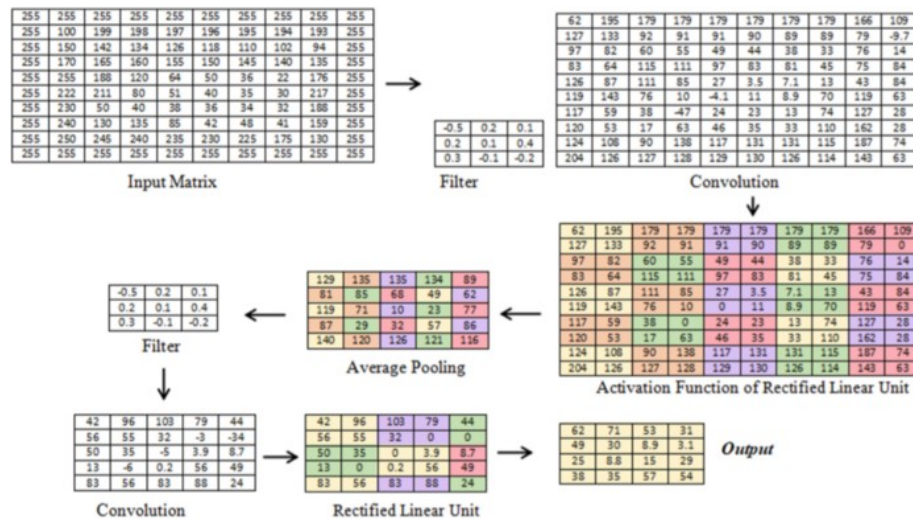
Gambar 2. 3 Arsitektur CNN [11]

2.5.1 Convolution Layer

Convolution layer merupakan lapisan pertama yang akan mengekstrak fitur dari gambar input. *Layer* ini bertugas untuk menjaga hubungan antara piksel dengan mempelajari fitur citra menggunakan kotak kecil data masukan. Tahap ini merupakan operasi matematika yang mengambil dua masukan yaitu matriks citra dan filter atau kernel [12].

Pada gambar 2.4 menunjukkan ilustrasi proses konvolusi, sebuah citra berukuran 10x10 piksel direpresentasikan sebagai matriks. Matriks awal diproses dengan dua layer konvolusi untuk mendapatkan *feature map* [14]. Setelah melalui proses konvolusi, kemudian dilakukan fungsi aktivasi dan menggunakan reLu. Output dari fungsi reLu kemudian dikenakan pooling dengan filter berukuran 2x2

dan *stride* sebesar dua. Sebelum melakukan *pooling*, dapat digunakan *zero padding* sehingga matriks hasil *pooling* berukuran 5x5. Matriks ini kemudian melalui tahap konvolusi kedua dengan ukuran filter sama seperti sebelumnya, tetapi dengan bobot yang berbeda [14].



Gambar 2. 4 Ilustrasi Proses Konvolusi [12]

Pada Gambar 2.5 merupakan ilustrasi dari proses konvolusi, pada proses ini terjadi pemecahan gambar menjadi 77 bagian yang lebih kecil dengan konvolusi yang sama [14].

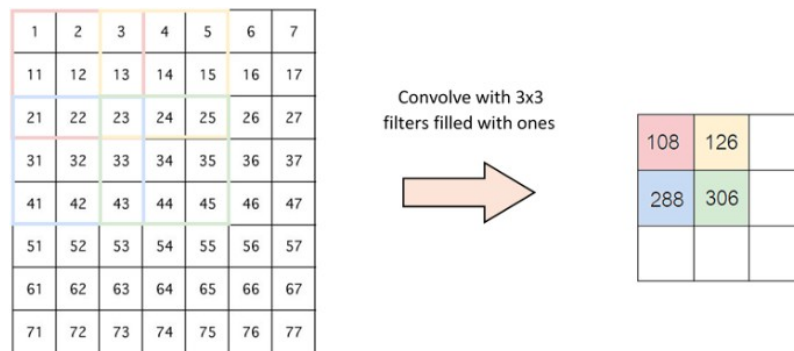


Gambar 2. 5 Ilustrasi Proses Konvolusi [14]

2.5.2 Stride

Stride merupakan parameter yang menentukan berapa jumlah pergeseran filter. Apabila nilai *stride* adalah 1, maka *conv. filter* akan bergeser sebanyak 1 pixel secara *horizontal* lalu *vertical*. Pada gambar 2.6 menunjukkan *stride* yang digunakan

adalah 2. Semakin kecil *stride* maka akan semakin detail informasi yang kita dapatkan dari sebuah input, namun membutuhkan komputasi yang lebih jika dibandingkan dengan *stride* yang besar. Namun perlu diperhatikan bahwa dengan menggunakan *stride* yang kecil kita tidak selalu akan mendapatkan performa yang bagus.



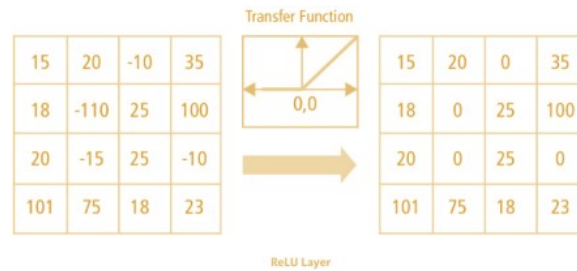
Gambar 2. 6 Contoh Stride 2 [12]

2.5.3 Padding

Padding atau *zero padding* adalah parameter menentukan jumlah *pixel* (berisi nilai 0) yang akan ditambahkan di setiap sisi dari input. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi output dari *convolution layer (feature map)* [14]. Untuk menghitung dimensi dari feature map dapat dilihat pada persamaan dibawah ini [14].

2.5.4 Fungsi Aktivasi

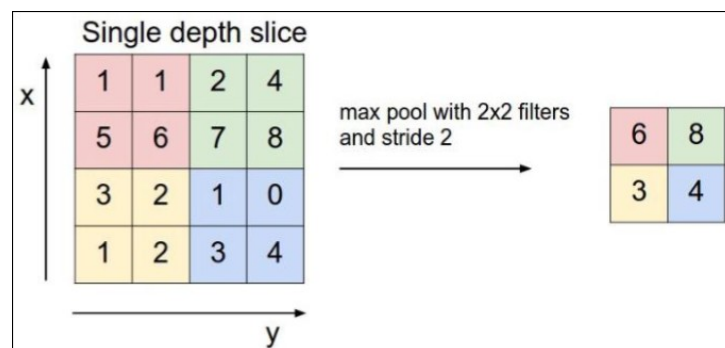
Fungsi aktivasi berada pada tahap sebelum melakukan pooling layer dan setelah melakukan proses konvolusi. Pada tahap ini, nilai hasil konvolusi dikenakan fungsi aktivasi atau *activation function*. Fungsi yang digunakan untuk aktivasi pada reLU, fungsi reLU adalah nilai output dari neuron bisa dinyatakan sebagai 0 jika inputnya adalah negatif. Jika nilai input dari fungsi aktivasi adalah positif, maka output dari neuron adalah nilai input aktivasi itu sendiri [14].



Gambar 2. 7 Operasi ReLU [12]

2.5.5 Pooling Layer

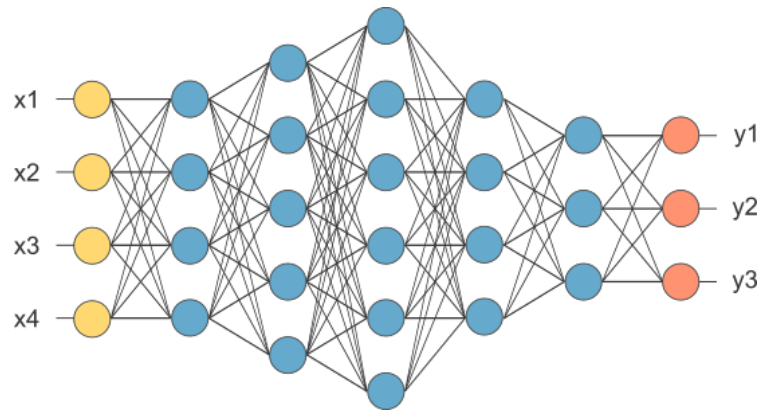
Pooling layer biasanya berada setelah *convolution layer*. Pada prinsipnya pooling layer terdiri dari sebuah filter dengan ukuran dan *stride* tertentu yang bergeser pada seluruh area *feature map*. Pooling yang biasa digunakan adalah *Max Pooling* dan *Average Pooling*. Tujuan dari penggunaan pooling layer adalah mengurangi dimensi dari *feature map* (*downsampling*), sehingga mempercepat komputasi karena parameter yang harus di update semakin sedikit dan mengatasi *overfitting* [14].



Gambar 2. 8 Contoh Max Pooling [12]

2.5.6 Fully-Connected Layer (MLP)

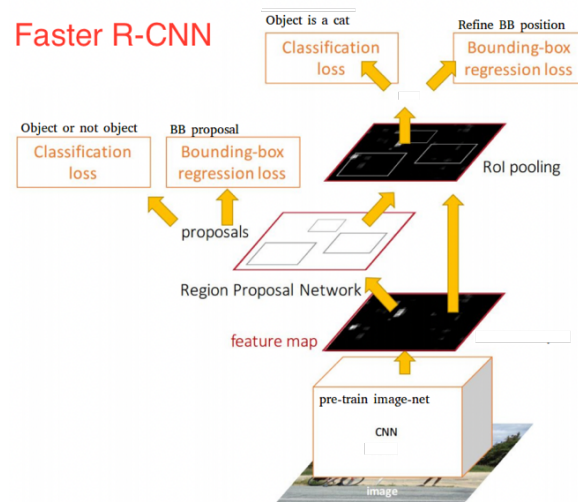
Fully connected layer adalah lapisan dimana semua neuron aktivitas dari lapisan sebelumnya terhubung semua dengan neuron di lapisan selanjutnya seperti halnya jaringan syaraf tiruan bisa. *Fully connected layer* umumnya digunakan pada metode Multi lapisan Perceptron dan bertujuan untuk mengolah data sehingga bisa diklasifikasikan [14].



Gambar 2. 9 Fully Connected Layer [12]

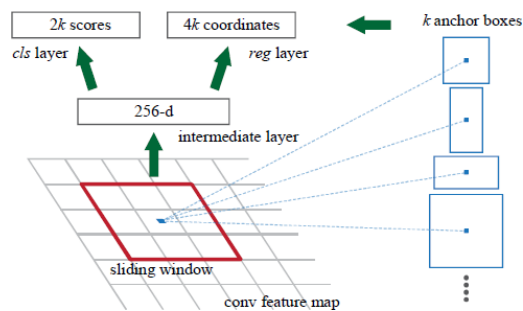
2.6 Algoritma *Object Detection* pada *Faster R-CNN*

Faster Region based Convolutional Neural Network (Faster R-CNN) yaitu salah satu algoritme pendeteksian objek yang termasuk dalam computer vision berbasis jaringan konvolusi. Pada awalnya *R-CNN* pada tahun 2015 dibuat untuk melakukan pendeteksian objek yang mengkombinasikan algoritme *Region Proposal Network* (RPN) dan *Convolutional Neural Network* (CNN) [5]. *Convolutional Neural Network* (CNN) termasuk dalam satu turunan jaringan saraf tiruan yang paling berkembang [18].

Gambar 2. 10 Arsitektur *Faster R-CNN* [17]

Pada *R-CNN* ataupun *Fast R-CNN* masih terdapat kekurangan, salah satunya adalah adanya *bottleneck*, dimana kompleksitas perhitungan *region proposal* pada *Region Proposal Network* (RPN) yang tidak dapat menyamai kecepatan komputasi

pada CNN[5]. Untuk mengatasi masalah *bottleneck*, dilakukan penelitian dengan optimisasi pada penggunaan fitur konvolusional untuk mempercepat proses *Region Proposal Network* (RPN), sehingga dapat mengurangi terjadinya *bottleneck* menggunakan *Faster R-CNN* yang merupakan kombinasi dari *Fast R-CNN* dan *Region Proposal Network* (RPN) [6]. *Faster R-CNN* menggunakan kombinasi arsitektur utamanya dari *Fast R-CNN* dan RPN. Algoritme ini merupakan pengembangan dari *Fast R-CNN* dengan mengubah bagian *selective search* pada *Fast RCNN* menjadi RPN [6].

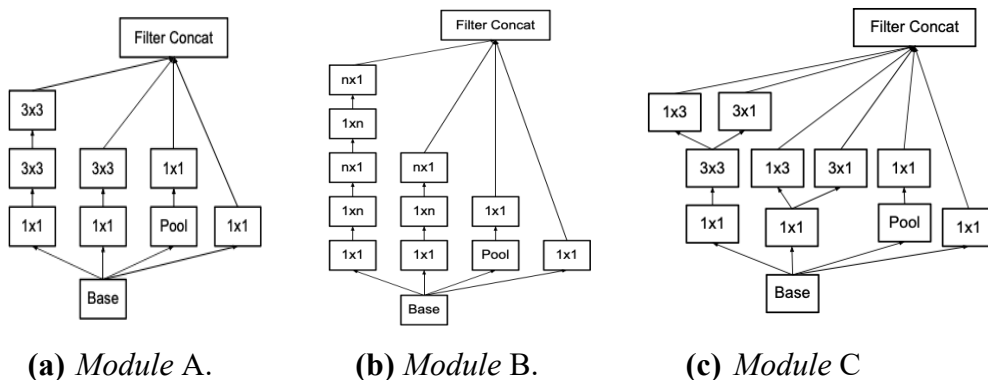


Gambar 2. 11 Arsitektur *Region Proposal Network* (RPN) [19]

Region Proposal Network (RPN) memetakan lapisan terakhir CNN ke dimensi yang lebih rendah (256-d) ke dalam *feature map* [19].

2.7 GoogLeNet Inception-V2

CNN memiliki berbagai arsitektur, salah satunya yaitu *Inception-V2* yang akan digunakan dalam penelitian ini. Arsitektur dari *Inception-V2* dirancang untuk mengurangi kompleksitas CNN, yang dilakukan dengan cara menyusun arsitektur yang lebih melebar dari pada mendalam. *Inception-V2* memiliki 3 modul seperti dibawah ini.



Gambar 2. 12 Modul *Inception-V2* [10]

Gambar 2.12 merupakan modul asli dari *Inception-V2*. *Module A* bertujuan untuk menggantikan konvolusi 5 x 5 menjadi 3x 3 [9]. Ini mengikuti prinsip yang mengatakan agregasi spasial dapat dilakukan pada penyisipan dimensi yang lebih rendah tanpa banyak atau kehilangan kekuatan representasional. Dengan melakukan konvolusi 3x3, performa konvolusi meningkat [19]. *Module B* ini diubah lebih melebar untuk mengurangi kompleksitas jaringan konvolusi [9]. Selain itu, *filter* diperluas yang mengikuti prinsip representasi dimensi yang lebih tinggi lebih mudah untuk diproses secara lokal dalam jaringan. Modul yang diperluas ditunjukkan oleh *module c*.

2.8 Algoritma Pengukur Jarak pada *Euclidean Distance*

Algoritma *Euclidean Distance* adalah algoritma yang digunakan untuk mencari pembagian persekutuan terbesar antara dua bilangan. *Euclidean* adalah suatu masalah yang kompleksitas waktunya dibatasi oleh fungsi kuadrat dari panjang nilai input. Konsekuensi yang paling penting untuk menggunakan algoritma ini adalah harus ditemukannya nilai x dan y [4].

$$d(x,y) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (2.1)$$

Keterangan :

- d : jarak antar fasilitas 1 dan 2
- x1 : koordinasi x untuk fasilitas 1
- x2 : koordinasi x untuk fasilitas 2
- y1 : koordinasi y untuk fasilitas 1
- y2 : koordinasi y untuk fasilitas 2

2.9 *Confusion Matrix*

Confusion Matrix adalah suatu metode penampilan informasi mengenai prediksi dan kebenaran dari suatu sistem klasifikasi yang mempunyai label kebenaran [23]. *Confusion matrix* mempunyai beberapa terminologi yaitu *True Positive* (TP), *True Negative* (TN), *False Positif* (FP), dan *False Negative* (FN). Contoh pada tugas akhir ini sistem harus mendeteksi orang dengan benar.

Tabel 2. 1Tabel *Confusion Matrix*

		<i>True Class</i>	
		<i>Positive</i>	<i>Negative</i>
<i>Predicted Class</i>	<i>Positive</i>	TP	FN
	<i>Negative</i>	FP	TN

1. *Positive*, menunjukkan faktanya adalah orang,
2. *Negative*, menunjukkan faktanya adalah bukan orang,
3. *True Positive* (TP), ketika model memprediksi objek manusia, faktanya terdapat objek manusia didalam citra tersebut,
4. *True Negative* (TN), ketika model memprediksi tidak ada objek manusia, faktanya memang tidak ada objek manusia didalam citra tersebut,
5. *False Postive* (FP), ketika model memprediksi objek manusia, faktanya tidak terdapat objek manusia didalam citra tersebut,
6. *False Negative* (FN), ketika model memprediksi tidak ada objek manusia, faktanya terdapat objek manusia didalam citra tersebut.

Terminologi pada *confusion matrix* tidak hanya memberikan informasi error yang dibuat oleh sistem, tetapi juga menentukan jenis error pada sistem [16].

- ***Accuracy***

Memberikan nilai akurasi yang didapat dari pengolahan model secara keseluruhan, yang merupakan fraksi dari total sampel yang diklasifikasikan dengan benar oleh pengklasifikasi [17]. Akurasi dapat ditentukan menggunakan rumus :

$$Accuracy = \frac{T\#TN}{T\#TN\&\#\&\#N} \quad (2.2)$$

- ***Precision***

Memberi tahu berapa fraksi prediksi sebagai kelas positif yang benar-benar positif [17]. *Precision* dapat ditentukan menggunakan rumus :

$$Precision = \frac{T\#}{T\#\&\#} \quad (2.3)$$

- ***Recall***

Memberi tahu berapa fraksi dari semua sampel positif yang diprediksi dengan benar sebagai positif oleh pengklasifikasi. Ini juga dikenal sebagai *True Positive Rate* (TPR), Sensitivitas, Probabilitas Deteksi [17]. *Recall* dapat ditentukan menggunakan rumus :

$$Recall = \frac{T}{T+F} \quad (2.4)$$

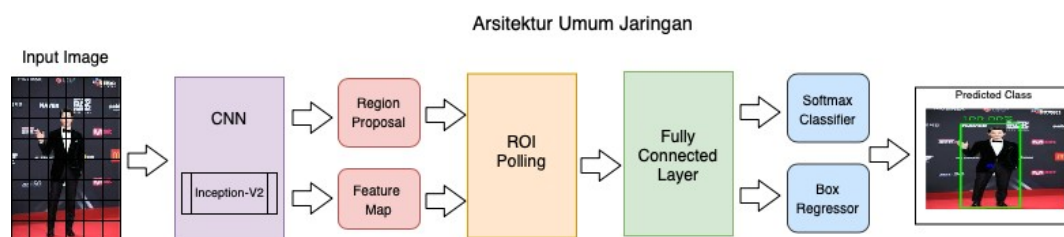
BAB III

PERANCANGAN SISTEM

Pada bab ini akan membahas analisis dan gambaran umum untuk memberikan informasi secara singkat bagaimana proses pada pengimplementasian metode *Faster R-CNN* untuk deteksi objek manusia dan *Euclidean Distance* untuk pengukuran jarak. Parameter performansi yang dibahas pada bab ini di antaranya adalah *Accuracy*, *Precision*, dan *Recall*.

3.1 Gambaran Umum Sistem *Faster R-CNN* untuk Deteksi Manusia

Desain sistem yang dirancang pada penelitian ini yaitu untuk mendeteksi objek manusia didalam video yang diambil oleh kamera, kemudian dilakukan pengolahan citra digital menggunakan bahasa Pemrograman Python dan *OpenCV* sebagai detectornya serta *Tensorflow* digunakan untuk pembuatan model klasifikasi melalui proses *training*. Setelah dilakukannya pendeteksian objek manusia akan dilakukan perhitungan titik *center* setiap *bounding box* yang akan digunakan untuk menghitung jarak objek. *Output* keseluruhan dari sistem ini yaitu objek yang telah diberi *bounding box* yang ditampilkan pada *website*.



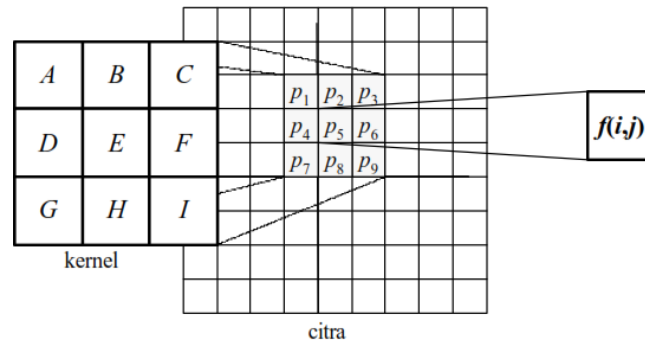
Gambar 3.1 Arsitektur Umum Jaringan

Gambar 3.1 merupakan arsitektur umum yang digunakan pada penelitian ini, arsitektur ini memiliki beberapa bagian yang digunakan untuk deteksi objek seperti lapisan CNN yang menggunakan arsitektur CNN yang menggunakan *inception-V2*, *region proposal*, *feature map*, *ROI Pooling*, *fully connected layer*, dan *softmax classifier* beserta *box regressor* yang menghasilkan *predicted class*. Untuk gambar yang dijadikan inputan akan di *resize*, gambar tidak lebih dari 200 KB dan dimensi citra tidak lebih dari 720 x 1280. Setiap gambar memiliki 3 channel dengan warna

RGB (*Red, Green, Blue*) yang digunakan menjadi *input* pada *layer* pertama pada arsitektur ini

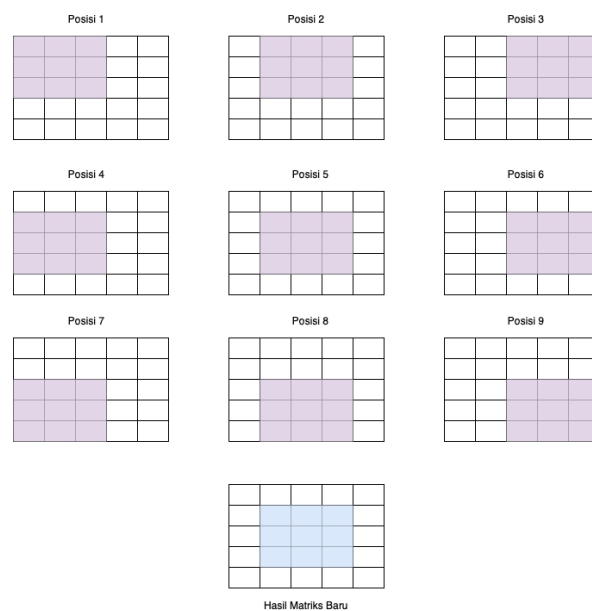
3.1.1 Proses Konvolusi

Konvolusi merupakan operasi matematis pada *Image Processing* yang mengalikan sebuah citra dengan sebuah *mask* atau kernel. Gambar 3.2 merupakan ilustrasi dari proses konvolusi.



Gambar 3. 2 Ilustrasi Proses Konvolusi [26]

Operasi konvolusi dilakukan dengan cara menggeser kernel konvolusi *pixel* per *pixel*, kemudian hasil konvolusi akan disimpan didalam matriks yang baru. Misalkan memiliki input citra berukuran 5x5 yang dijadikan *input* untuk proses konvolusi yang menggunakan kernel (filter) berukuran 3x3. Filter tersebut akan mengalami pergeseran sebanyak 9kali karena *stride* 1. Gambar 3.3 merupakan contoh konvolusi dengan *stride* 1.



Gambar 3. 3 Ilustrasi Pergerakan *Stride*

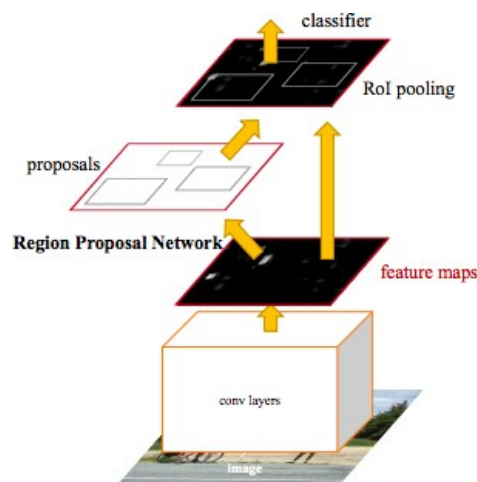
Hasil dari gambar 3.3 merupakan hasil dari proses konvolusi, dengan input citra berukuran 5x5 dan *kernel mask* 3x3 sehingga *output* menjadi berukuran 3x3.

3.1.2 Feature Map

Sebuah map yang dibuat oleh *convolutional layer* yang berisi informasi tentang representasi vector dari *image* yang ditangkap, *convolution layer* akan menghasilkan dua *feature map* yang sama, *feature map* pertama akan dilolah di RPN untuk menghasilkan *region proposal* dan *feature map* lainnya akan langsung dikirim ke *pooling layer*.

3.1.3 Region Proposal Network

Sebuah modul yang bekerja untuk mengolah *feature map* yang telah dibuat pada *convolution layer* untuk memprediksi bagian yang dianggap sebagai objek dan melakukan prediksi *bounding box* dari objek tersebut, RPN dibagi menjadi 2 *convolution layer* dimana 1 layer bertanggung jawab untuk mendeteksi letak objek dan 1 layer berfungsi memprediksi *bounding box*, keluaran dari RPN adalah *region proposal* dari *image*.



Gambar 3. 4 Ilustrasi RPN [27]

Pada RPN ini akan terdapat *layer convolution neural network* (CNN), *feature map*, dan *proposed regions*. Untuk melakukan proses ini akan dijelaskan pada ilustrasi pada gambar 3.5 berikut.