BAB 2
TINJAUAN PUSTAKA

2.1 Computer Vision

Computer vision adalah kemampuan komputer untuk mendeteksi gambar dan "melihat" apa yang dilihatnya. Misalnya, bagian proses ini dapat mendeteksi bentuk, warna, atau kontras dalam foto dan menggambar fitur tertentu.[10]

2.2 Face Recognition

Face Recognition merupakan tugas yang dilakukan manusia secara rutin dan mudah dalam kehidupan sehari-hari. *Face recognition* telah menciptakan minat yang sangat besar dalam pemrosesan otomasi gambar digital dalam berbagai aplikasi, termasuk otentikasi biometrik, pengawasan, interaksi manusia-komputer, dan manajemen multimedia.[11]

2.3 Citra Grayscale

Dalam tahap Pre-processing peregangan kontras dilakukan pada gambar yang diperoleh di mana piksel putih dibuat lebih putih dan piksel hitam dibuat lebih hitam atau disebut juga Grayscaling [11]. Berikut adalah Rumus dari grayscaling.

$$s = 0.2989 * R + 0.5870 * G + 0{,}1140 * B \quad (2{,}1)$$

2.4 Resize

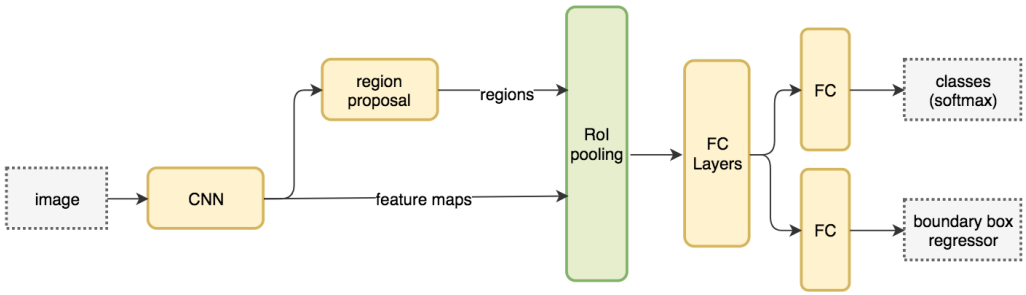
Resizing merupakan proses mengubah dan menyamakan citra masukan, dengan mengubah resolusi horizontal dan vertikal citra masukan tersebut. Kinerja sistem akan melambat apabila terdapat keanekaragaman ukuran citra masukan. Resizing dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit.

2.5 Faster R-CNN

Faster R-CNN adalah salah satu metode pendeteksi objek yang menggunakan *Convolutional Neural Network* (CNN) sebagai dasar dari *deep learning* dengan kata lain Faster R-CNN adalah CNN yang telah berevolusi dari CNN menjadi R-CNN lalu Fast R-CNN dan sekarang menjadi Faster R-CNN. Apa yang menjadi perbedaan dengan versi sebelumnya adalah pengenalan terhadap *Region Proposal Network* (RPN). RPN adalah fully trained convolutional network yang secara bersamaan memprediksi batas objek dan skor objek di setiap deteksi.[12]

Berikut adalah bagian dari Faster R-CNN di gambar 2.1:

1. *Convolutional Neural Network*
2. *Region Proposal Network*
3. *Fully Connected Layer (detector)*



Gambar 2.1 Faster R-CNN[13]

2.5.1 CNN

Convolutional Neural Network (CNN) merupakan pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada kasus klasifikasi citra, MLP kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra dan menganggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik [14].

CNN telah mendominasi banyak tugas dalam *computer vision*[15]. Untuk deteksi objek, metode deteksi *Region-based* CNN sekarang menjadi paradigma utama. Ini adalah area yang berkembang pesat sehingga tiga generasi model deteksi

Region-based CNN, dari R-CNN, hingga Fast R-CNN, dan akhirnya Faster R-CNN, telah diusulkan dalam beberapa tahun terakhir, dengan semakin akurasi yang lebih baik dan kecepatan pemrosesan yang lebih cepat. [12]

2.5.2 ResNet50

ResNet-50 merupakan salah satu arsitektur dari CNN yang memperkenalkan sebuah konsep baru yaitu shortcut connections. Munculnya konsep shortcut connections yang ada pada arsitektur ResNet-50 memiliki keterkaitan dengan vanishing gradient problem yang terjadi ketika usaha memperdalam struktur suatu network dilakukan. Bagaimanapun memperdalam suatu network dengan tujuan meningkatkan performansinya tidak bisa dilakukan hanya dengan cara menumpuk layer. Semakin dalam suatu network dapat memunculkan vanishing gradient problem yang bisa membuat gradient menjadi sangat kecil yang berakibat pada menurunnya performansi atau akurasi[16].

2.5.3 Convolutional Layer

Seluruh data yang menyentuh lapisan konvolusional akan mengalami proses konvolusi. Lapisan akan mengkonversi setiap filter ke seluruh bagian data masukan dan menghasilkan sebuah activation map atau feature map 2D. Filter yang terdapat pada *Convolutional Layer* memiliki panjang, tinggi (*pixels*) dan tebal sesuai dengan channel data masukan. Setiap filter akan mengalami pergeseran dan operasi "*dot*" antara data masukan dan nilai dari filter. Lapisan konvolusional secara signifikan mengalami kompleksitas model melalui optimalisasi outputnya.

Di mana P adalah ukuran padding dan F adalah ukuran bidang reseptif atau tingkat spasial yang tentu saja sama dengan ukuran filter. Konvolusi pada dasarnya hanya berupa dot product antara filter dengan sebuah bidang reseptif kecil pada citra masukan yang berukuran sama dengan ukuran filter, dimana filter dilambangkan $k \times k$.

Persamaan diatas hanya berlaku untuk operasi konvolusi pada layer pertama saja, untuk melakukan operasi konvolusi ke tahap layer selanjutnya persamaan yang

digunakan akan sedikit berbeda, dimana inputan dari layer sebelumnya sebanyak 21 jumlah filter masing-masing akan di konvolusi dan kemudian akan dijumlahkan.

Stride adalah parameter yang menentukan berapa jumlah pergeseran filter. Jika nilai stride adalah 1, maka conv. filter akan bergeser sebanyak 1 pixels secara horizontal lalu vertical.

Sedangkan *Padding* atau *Zero Padding* adalah parameter yang menentukan jumlah pixels (berisi nilai 0) yang akan ditambahkan di setiap sisi dari input. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi output dari conv. layer (Feature Map).

Untuk menghitung dimensi dari *feature map* kita bisa gunakan rumus seperti dibawah ini.

$$featuremap = \frac{W - N + 2P}{S} + 1 \quad (2,2)$$

Dimana,

W = Panjang/Tinggi Input

N = Panjang/Tinggi Filter

P = *Zero Padding*

S = *Stride*

Operasi konvolusi yang dilakukan dengan cara perkalian antara matriks filter dan matriks citra dapat didefinisikan sebagai berikut :

$$f(x,y) = f1.(1,1) + f2.(2,1) + f3.(3,1) + \cdots + fn(x,y) \quad (2,3)$$

2.5.4 Fungsi Aktivasi ReLu

Di dalam CNN terdapat suatu fungsi bernama aktivasi ReLu. Fungsi aktivasi ReLU, dimana hasil dari konvolusi setiap *pixel* yang memiliki nilai < 0 akan dijadikan 0 dan yang memiliki nilai ≥ 0 nilai tersebut ditetapkan, berikut adalah rumus dari aktivasi ReLu.

$$f(x) = \max(0, x) \quad (2,4)$$

2.5.5 Pooling Layer

Pooling Layer merupakan tahap setelah *Convolutional Layer*. Pooling Layer terdiri dari sebuah filter dengan ukuran dan stride tertentu. Setiap pergeseran akan ditentukan oleh jumlah stride yang akan digeser pada seluruh area feature map atau activation map. Dalam penerapannya, *pooling Layer* yang biasa digunakan adalah *Max Pooling* dan *Average Pooling*. Pada penelitian kali ini fungsi *Max Pooling* akan digunakan, adapun rumus fungsi *Max pooling* adalah sebagai berikut:

$$Pool_{x,y} = Max\left(Conv_{x,y}, Conv_{x+1,y}, Conv_{x,y+1}, Conv_{x+1,y+1}\right) \quad (2.5)$$

Keterangan :

$Pool_{x,y}$ = Hasil dari pooling layer.

$Conv_{x,y}$ = Nilai pixel dari hasil convolutional layer.

2.5.6 RPN

RPN adalah sebuah neural network yang menggantikan peran *Selective Search* untuk mengajukan region (bagian-bagian mana dari sebuah gambar yang perlu dilihat lebih jauh). RPN menghasilkan beberapa bounding box, setiap box memiliki 2 skor probabilitas apakah pada lokasi tersebut terdapat objek atau tidak. Penggunaan RPN untuk mengganti *Selective Search* ini mengurangi kebutuhan komputasi yang cukup signifikan, dan membuat keseluruhan model dapat di-*train* secara *end-to-end*[17].

Selain itu, Faster R-CNN juga menghasilkan performa yang lebih cepat dan lebih akurat bila dibandingkan dengan Fast R-CNN, dan saat ini telah menjadi pilihan model yang umum digunakan untuk deteksi objek dengan solusi berbasis *deep learning* [18].

RPN akan menerima input berupa feature map yang telah diproses oleh konvolusi lalu menjalankan sliding window untuk menghasilkan proposal berdasarkan anchor yang telah ditentukan. Untuk setiap proposal akan dilakukan konvolusi untuk menghasilkan nilai objektivitas (0-1) dimana nilai tersebut menandakan keberadaan objek atau foreground.

Pada *Region Proposal Network* awalnya gambar dimasukkan ke dalam jaringan *Convolutional Neural Network*. Gambar input diteruskan ke jaringan

convolutional layer terakhir yang menampilkan *feature map*[9]. Biaya waktu untuk menghasilkan proposal wilayah jauh lebih kecil di RPN daripada pencarian selektif, saat RPN berbagi komputasi paling banyak dengan jaringan deteksi objek. Secara singkat, RPN memberi peringkat pada *region box* (disebut *anchor*) dan mengusulkan kotak yang kemungkinan besar berisi objek.

Training loss untuk RPN juga merupakan *multitask loss*, yang diberikan oleh:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (2,6)$$

$p_i = Predicted\ Probability\ of\ Anchor\ i\ being\ an\ object$
$p_i^* = Ground\ truth\ label\ , 1\ if\ positive\ 0\ if\ negative$
$t_i = Predicted\ four\ parameterized\ coordinates$
$N_{cls} = Number\ of\ anchor\ in\ minibatch(512)$
$L_{cls} = softmax\ loss\ function$

2.5.6.1 Anchor

Anchor memainkan peran penting dalam Faster R-CNN. *Anchor* adalah sebuah box. Dalam *Faster R-CNN*, ada 9 anchor pada posisi gambar. Untuk setiap titik dalam peta fitur keluaran, jaringan harus mempelajari apakah suatu objek ada dalam citra masukan di lokasi yang sesuai dan memperkirakan ukurannya. Hal ini dilakukan dengan menempatkan satu set "*Anchors*" pada citra masukan untuk setiap lokasi pada peta fitur keluaran dari jaringan *backbone*. Jangkar ini menunjukkan kemungkinan objek dalam berbagai ukuran dan rasio aspek di lokasi ini.

Banyaknya anchor ditentukan dengan rumus :

$$anchor = Fw * Fh * 9 \quad (2,7)$$

Keterangan : Fw = Panjang Feature Map

Fh = Lebar Feature Map

2.5.6.2 IoU

Intersection over Union (IoU) digunakan saat menghitung mAP. Ini adalah angka dari 0 hingga 1 yang menentukan jumlah kesamaan tindih antara Boundary box asli dan prediksi.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \quad (2,8)$$

$$A \cup Gt = A * Gt$$

$$A \cap Gt = [(tw1 - tx1) * (th1 - ty1)] + [(tw2 - tx2) * (th2 - yh2)] - A \cup Gt$$

Keterangan :

$A = \min(t_{w1}, t_{w2}) - \max(t_{x1}, t_{x2})$

$Gt = \min(t_{h1}, t_{h2}) - \max(t_{y1}, t_{y2})$

$t_{x1}, t_{x2} = Koordinat\ X$

$t_{y1}, t_{y2} = Koordinat\ Y$

$t_{w1}, t_{w2} = Lebar\ box\ regressor$

$t_{x1}, t_{x2} = Tinggi\ box\ regressor$

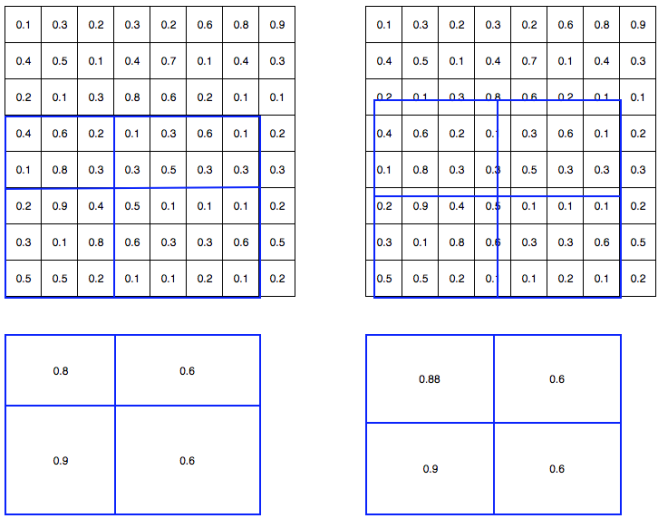
Klasifikasi menunjukkan probabilitas 0 atau 1 yang menunjukkan apakah wilayah tersebut berisi objek atau tidak dengan p*= 1 jika IoU >0,7, p* = .1 jika IoU <o,3, p* = 0.

2.5.7 ROI Pooling

Region of Interest pooling, yaitu pemetaan ROI dan kemudian *pooling*. Pemetaan adalah untuk memetakan ROI gambar yang digunakan untuk pelatihan ke lapisan fitur terakhir.

Setelah mendapatkan koordinat ROI, kami membagi area ROI ini menjadi bagian H x W , setiap bagian dikumpulkan, dan akhirnya setiap gabungan gabungan

dimasukkan ke lapisan berikutnya. Melalui operasi ROI *Pooling Layer*, semua ROI, terlepas dari ukuran ROI, menghasilkan *vektor* dengan panjang tetap ke lapisan berikutnya[19]. Gambaran RoI Pooling dapat dilihat di gambar 2.2.



Gambar 2.2 RoI Pooling[19]

2.5.8 Fully Connected Layer

Feature map yang dihasilkan oleh tahap sebelumnya berbentuk *multidimensional array*. Sehingga, Sebelum masuk pada tahap *Fully Connected Layer*, *Feature Map* tersebut akan melalui proses "*flatten*" atau reshape. Proses flatten menghasilkan sebuah vektor yang akan digunakan sebagai input dari *Fully Connected Layer*. *Fully Connected Layer* memiliki beberapa *Hidden Layer*, *Action Function*, *Output Layer* dan *Loss Function*. Adapun persamaan yang akan digunakan pada tahapan ini adalah sebagai berikut :

$$Fully_i = \sum_{j}^{Length(flatten)} W_{i,j} * flatten_j + b_i \quad (2.9)$$

Keterangan:

$Fully_i$ = Hasil dari perhitungan pada fully-connected layer.

$W_{i,j}$ = Nilai bobot yang digunakan dari hasil convolutional layer.

$flatten_j$ = Nilai dari vektor ke-j.

i = Kelas ke-i (i=1,2,3,4,5,6,7).

Kemudian hasil dari $Fully$ akan diaktivasi dengan menggunakan fungsi softmax, digunakan fungsi softmax karena pada penelitian ini menghasilkan output atau jumlah kelas lebih dari dua, atau biasa disebut dengan multi-class. Tujuan dari fungsi softmax sendiri adalah agar dapat diketahui prediksi yang dihasilkan dari arsitektur yang dibangun, dimana rumus untuk fungsi softmax ditulis pada persamaan

$$\widehat{y_i} = \frac{e^{Fully_i}}{(\sum_{j=1}^{kelas} e^{Fully_i})} \tag{2.10}$$

Keterangan:

$\widehat{y_i}$ = Hasil dari aktivasi fungsi *softmax.*

$Fully_i$ = Hasil dari perhitungan pada *fully-connected* layer ke-i.

$flatten_j$= Nilai dari vektor ke-j

2.6 Tensorflow

TensorFlow adalah sistem pembelajaran mesin yang beroperasi dalam skala besar dan dalam lingkungan yang heterogen. Tensor- Flow menggunakan grafik aliran data untuk merepresentasikan komputasi, status bersama, dan operasi yang mengubah status tersebut. Ini memetakan node grafik aliran data di banyak mesin dalam sebuah cluster, dan di dalam mesin di beberapa perangkat komputer, termasuk CPU multicore, GPU tujuan umum, dan ASIC yang dirancang khusus yang dikenal sebagai Tensor Processing Units (TPUs). Arsitektur ini memberikan fleksibilitas kepada pengembang aplikasi: sedangkan pada desain "server parameter" sebelumnya, pengelolaan status bersama dibangun ke dalam sistem, TensorFlow memungkinkan pengembang untuk bereksperimen dengan pengoptimalan baru dan melatih algoritma. TensorFlow mendukung berbagai aplikasi, dengan fokus pada pelatihan dan inferensi pada jaringan neural dalam [20].

2.7 Keras

Keras adalah salah satu library yang biasa digunakan dalam machine learning. Keras memiliki antarmuka yang sangat produktif dan sangat bagus untuk dipakai didalam *machine learning*, dengan fokus pada *deep learning*. Sudah banyak penelitian yang memakai library keras untuk aplikasi yang memiliki deep learning[21].

2.8 Confusion Matrix

Confusion matrix juga sering disebut *error matrix*. Pada dasarnya confusion matrix memberikan informasi perbandingan hasil klasifikasi yang dilakukan oleh sistem (model) dengan hasil klasifikasi sebenarnya. *Confusion matrix* berbentuk tabel matriks yang menggambarkan kinerja model klasifikasi pada serangkaian data uji yang nilai sebenarnya diketahui. Gambar dibawah ini merupakan confusion matrix dengan 4 kombinasi nilai prediksi dan nilai aktual yang berbeda[22]. Berikut adalah rumus untuk menghitung jumlah akurasi, presisi, dan recall pada *Confusion Matrix.*

$$Akurasi = \frac{Tp + Tn}{Tp + Fp + Tn + Fn} * 100\%$$

$$Presisi = \frac{Tp}{Tp + Fp} * 100\%$$

$$Recall = \frac{Tp}{Tp + Fn} * 100\%$$

$Tp = True\ Positive$
$Tn = True\ Negative$
$Fp = False\ Positive$
$Fn = False\ Negative$