



Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

BAB 2

LANDASAN TEORI

2.1 Machine Learning

Teknologi *Machine Learning* merupakan teknologi yang dikembangkan untuk mesin agar dapat belajar sendiri tanpa arahan dari penggunanya. Pembelajaran mesin dikembangkan berdasarkan disiplin ilmu lainnya seperti statistika, matematika dan data mining sehingga mesin dapat belajar dengan menganalisa data tanpa perlu di program ulang atau diperintah[9].

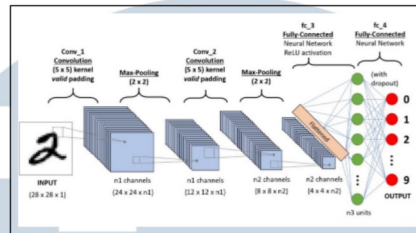
Secara prinsip, *machine learning* adalah suatu penerapan dalam kecerdasan buatan yang menyediakan sistem yang bekerja secara otomatis dan mampu belajar dari pengalaman atau pengetahuan tanpa perlu diprogram secara eksplisit. Machine learning berfokus pada pengembangan program komputer yang dapat mengakses data dan menggunakan data tersebut untuk *self learning* [10]. Proses *self learning* dimulai dengan mengamati data, baik melalui interaksi langsung maupun instruksi untuk mencari atau menyelidiki pola *data*, sehingga dapat membuat keputusan atau kesimpulan yang lebih baik di masa depan berdasarkan contoh-contoh tersebut. Tujuan utama dari *machine learning* adalah memberikan kemampuan kepada komputer untuk belajar secara otomatis tanpa intervensi, campur tangan, atau bantuan manusia, serta menyesuaikan aktivitas yang sesuai [11].

2.2 Convolutional Neural Network

Convolutional Neural Network merupakan salah satu jenis dari neural network yang biasa digunakan untuk mendeteksi sebuah data yang berbasis *image* [12]. CNN dikenalkan oleh seorang ilmuwan Yann LeCun pada tahun 1988. CNN merupakan salah satu metode yang menjadi asal mulanya kemunculan dan suksesnya *Deep Learning*[13]. Komponen utama dari *Convolutional Neural Network* adalah:

1. *input layer*
2. *convolution layer*
3. *activation layer*
4. *pooling layer*

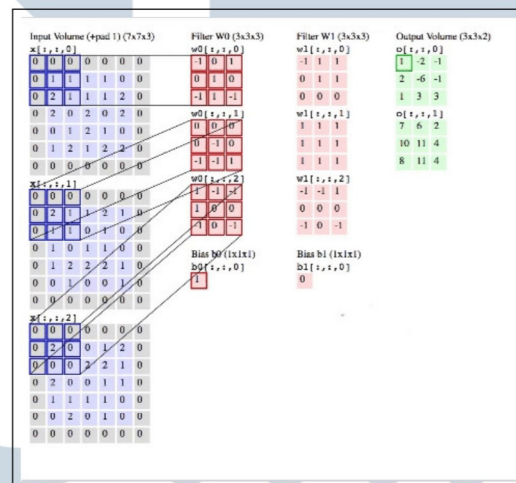
5. fully connected layer



Gambar 2.1. Alur CNN

2.2.1 Input Layer

Input layer merupakan bagian yang berisi *filter matrix* dua dimensi yang berukuran kecil. Untuk setiap *filter* akan mengikuti *input matrix* atau gambar yang nantinya akan menghasilkan sebuah *Activation map* berukuran dua dimensi [14].

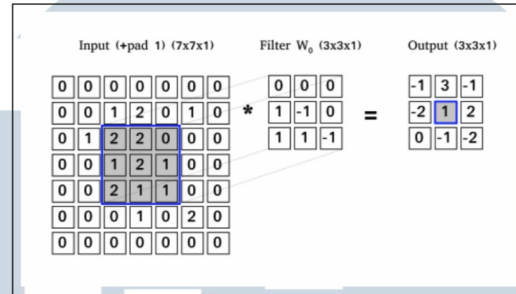


Gambar 2.2. Input Layer

2.2.2 Convolution Layer

Pada *Convolutional Neural Network* (CNN), *convolution layer* menjadi komponen utama. Fungsi utama dari *convolution layer* adalah menghasilkan citra baru yang menampilkan fitur-fitur dari citra *input*. Dalam proses ini, *convolution layer* menggunakan *filter* yang diterapkan pada setiap citra yang menjadi *input*. *Filter* ini berupa matriks dua dimensi dengan ukuran yang bervariasi, seperti

5x5, 3x3, atau 1x1. Melalui proses konvolusi dengan menggunakan *filter* pada *convolution layer*, akan dihasilkan peta fitur (*feature map*) yang selanjutnya digunakan pada *activation layer* [15].



Gambar 2.3: Contoh *Convolution Layer*

2.2.3 Activation Layer

Activation layer adalah lapisan di mana *feature map* disertakan dalam fungsi aktivasi. Fungsi aktivasi digunakan untuk mengubah nilai peta objek pada area tertentu sesuai dengan fungsi aktivasi yang digunakan. Tujuannya adalah untuk membawa nilai yang menunjukkan fitur dominan dari citra yang masuk ke tingkat berikutnya[16]. Fungsi aktivasi dapat dilihat pada rumus berikut

$$sigmoid = \frac{1}{1 + e^{-\theta^T x}} \quad (2.1)$$

$$tanh = \tanh(x) \quad (2.2)$$

$$ReLU = \max(0, x) \quad (2.3)$$

$$LReLU = \max(0.1x, x) \quad (2.4)$$

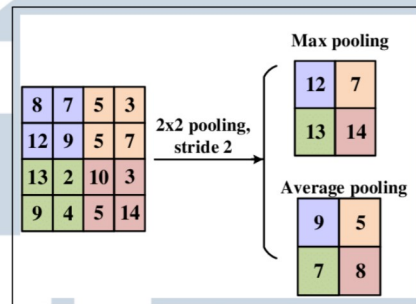
$$Maxout = \max(w1^t + b1, w2^t + b2) \quad (2.5)$$

$$ELU = (x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases} \quad (2.6)$$

2.2.4 Pooling Layer

Fungsi *pooling layer* adalah mengurangi dimensi spasial fitur konvolusi yang telah dihasilkan. Tujuannya adalah untuk mengurangi beban komputasi dalam memproses data dengan mengurangi dimensi tersebut. Selain itu, lapisan ini juga

berguna dalam mengekstraksi fitur yang paling menonjol dan tidak dipengaruhi oleh rotasi atau posisi, sehingga memastikan bahwa pelatihan model berjalan dengan efektif [17]. contoh *pooling layer* digambarkan pada Gambar 2.4

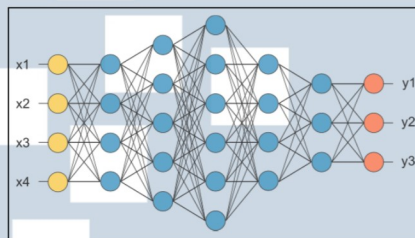


Gambar 2.4: Cara kerja *Pooling Layer*

Berdasarkan Gambar 2.4 yang dijadikan sebagai contoh adalah matriks 4x4, maka dilakukanlah penyederhanaan menjadi matrix 2x2 dan dibagi menjadi 2 bagian yaitu *max pooling* dan *average pooling*. Untuk mendapatkan *max pooling* diambil nilai terbesar pada setiap jendela matrix. pada jendela biru diambil angka 13, jendela oranye 7, jendela hijau 13, dan jendela merah 14. Lalu, untuk mendapatkan *average pooling* didapat dari hasil rata-rata setiap jendela. Fungsi *max pooling* adalah untuk mengembalikan nilai maksimum dari bagian gambar yang dicakup oleh kernel, sedangkan fungsi dari *average pooling* adalah untuk mengembalikan nilai rata – rata dari bagian gambar yang dicakup oleh kernel [18].

2.2.5 Fully Connected Layer

Fully connected layer merupakan lapisan di mana semua aktivitas neuron dari lapisan sebelumnya terhubung dengan semua neuron pada lapisan berikutnya, mirip dengan cara jaringan saraf tiruan bekerja. Sebelum dapat dihubungkan ke semua neuron pada *Fully Connected layer*, setiap aktivitas dari lapisan sebelumnya perlu diubah menjadi data satu dimensi [19].



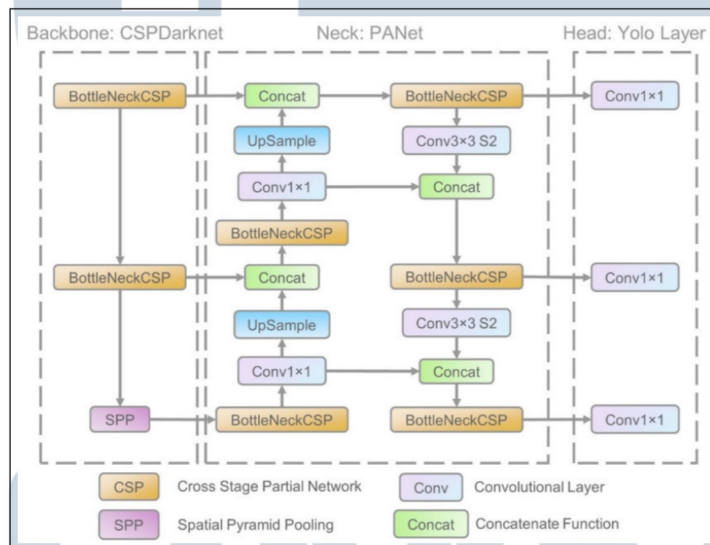
Gambar 2.5: *Fully Connected Layer*

Pada Gambar 2.5 dapat dilihat bahwa sebelum hasil *pooling* digunakan sebagai input, hasil *pooling* akan diubah menjadi vektor x_1 , x_2 , dan seterusnya. Kemudian, vektor ini akan diproses melalui *fully connected layer*. Pada lapisan terakhir dalam *fully connected layer*, akan digunakan fungsi aktivasi *sigmoid* atau *softmax* untuk menentukan klasifikasi dari citra *input* atau masukan yang berasal dari *input layer* CNN.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

2.3 YOLOv5 (You Look Only Once)

You Look Only Once merupakan *Single Convolutional Network* yang bekerja melihat gambar secara keseluruhan lalu memprediksi *bounding box*, yang kemudian akan memprediksi untuk probabilitas kelas *bounding box* tersebut, *bounding box* ini diberi bobot sesuai dengan probabilitas yang diharapkan[2]. Metode *You Look Only Once* pada gambar membuat prediksi setelah satu propagasi melewati *neural network*. Kemudian mengirimkan item yang terdeteksi setelah *non-max suppression* (yang memastikan bahwa algoritma deteksi objek hanya mengidentifikasi setiap objek satu kali).



Gambar 2.6: Arsitektur pada YOLOv5 *You Look Only Once*

Terdapat tiga arsitektur utama pada algoritma *You Look Only Once*[2]:

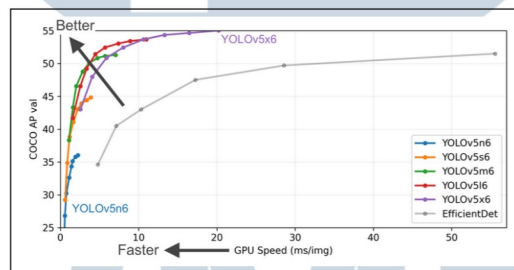
1. **Backbone:** merupakan pengekstrak untuk fitur utama pada input sebuah gambar. CSP (Cross Stage Partial Networks) digunakan sebagai penopang di YOLO v5 untuk mengekstrak kaya akan karakteristik yang berguna dari sebuah *input* gambar.
2. **Neck:** Model Leher sebagian besar digunakan untuk membuat *pyramid feature*. *pyramid feature* membantu model dalam mengeneralisasi dengan baik dalam hal *scaling object*. Ini membantu dalam identifikasi objek yang sama dalam berbagai ukuran dan skala. *Feature pyramid* cukup bermanfaat dalam membantu model bekerja secara efektif pada data yang sebelumnya

tidak terlihat. Model lain, seperti FPN, BiFPN, dan PANet, menggunakan berbagai macam pendekatan *pyramid feature*.

3. *Head*: Model *head* bertanggung jawab atas pendeteksian akhir. Penggunaan *anchor boxes* untuk membuat *final output* vektor dengan *class probabilities*, *objectness scores*, and *bounding boxes*.

YOLOv5 dalam penggunaan modelnya memiliki beberapa model yaitu:

1. n untuk model yang berukuran sangat kecil (nano).
2. s untuk model yang berukuran kecil.
3. m untuk model yang berukuran sedang.
4. l untuk model yang berukuran besar.
5. x untuk model yang berukuran sangat besar



Gambar 2.7: Performa YOLOv5 dalam beberapa ukuran model

YOLO sendiri memiliki banyak sekali versi dan varian modelnya, dan yang paling sering digunakan adalah YOLOv5. YOLOv5 sering digunakan karena algoritma ini menjadi salah satu algoritma yang cepat dan akurat dalam pendeteksian sebuah objek. Metode ini mampu mendeteksi 2 kali lebih cepat dibandingkan versi YOLO sebelumnya yaitu YOLOv4 dan YOLOv3 [20]. Berikut perbandingan antara YOLOv3, YOLOv4 dan YOLOv5.

	YOLOv3	YOLOv4	YOLOv5
Neural Network Type	Fully convolution	Fully convolution	Fully convolution
Backbone Feature Extractor	Darknet-53	CSPDarknet53	CSPDarknet53
Loss Function	Binary cross entropy	Binary cross entropy	Binary cross entropy and Logits loss function
Neck	FPN	SSP and PANet	PANet
Head	YOLO layer	YOLO layer	YOLO layer

Gambar 2.8: Perbedaan YOLOv5 dengan versi sebelumnya

2.3.1 Head Of The Network

YOLOv5 memiliki cara memprediksi yang sama dengan dua versi sebelumnya yaitu YOLOv3 dan YOLOv4 dimana saat akan memprediksi *bounding box* (x,y,tinggi, lebar), skor, dan kelas objeknya. Perhitungan untuk menghitung *target* koordinat dari *bounding box* terdapat perubahan dari versi sebelumnya.

$b_x = \sigma(t_x) + c_x$	$b_x = (2 \cdot \sigma(t_x) - 0.5) + c_x$
$b_y = \sigma(t_y) + c_y$	$b_y = (2 \cdot \sigma(t_y) - 0.5) + c_y$
$b_w = p_w \cdot e^{t_w}$	$b_w = p_w \cdot (2 \cdot \sigma(t_w))^2$
$b_h = p_h \cdot e^{t_h}$	$b_h = p_h \cdot (2 \cdot \sigma(t_h))^2$
(a)	(b)

Gambar 2.9: perhitungan YOLO versi sebelumnya dan yolo5

2.4 OpenCV

OpenCV (*Open Source Computer Vision Library*) adalah sebuah *open source computer vision* dan *machine learning software library*. OpenCV dibangun untuk memenuhi keseluruhan infrastruktur *computer vision* dan mempercepat persepsi mesin dalam produk komersial[21].

Library ini memiliki 2500 algoritma yang sudah di optimalkan, yang mencakup serangkaian algoritma *computer vision* dan *machine learning*. Algoritma ini dapat digunakan untuk mendeteksi dan mengenali wajah, mengidentifikasi sebuah objek, *tracking* pergerakan kamera, *tracking* pada sebuah objek yang bergerak, mencari gambar yang serupa dari sebuah *database*, mendeteksi pergerakan mata, dan lain-lain. OpenCV memiliki lebih dari 47 ribu pengguna komunitas dan perkiraan jumlah unduhan mencapai 18 juta. *Library*

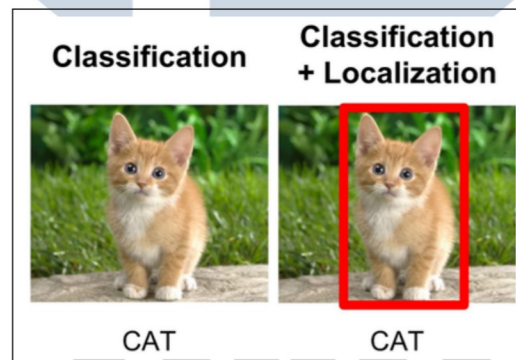
ini biasanya banyak digunakan di pemerintahan, tim peneliti, dan perusahaan-perusahaan besar[22].

2.5 Object Localization

Object localization merupakan pengembangan dari *object classification*. Dimana gambar yang dihasilkan tidak lagi dalam bentuk kelas dan kategori, namun pada umumnya gambar tersebut akan dalam bentuk bounding box[23].

2.6 Object Classification

Object classification merupakan bagian dari *computer vision*. *Object classification* merupakan tugas dari *machine learning* untuk memilah apakah suatu objek berupa gambar atau video. Komputer akan melakukan identifikasi pada suatu *input* gambar sehingga nantinya gambar yang sudah di-*input* dapat terdeteksi termasuk kategori atau kelas yang mana sebagai *output*[24].



Gambar 2.10: *Object classification* dan *Object localization*

2.7 Metriks Evaluasi

Tahapan perancangan yang dilakukan adalah dengan cara melakukan *testing* pada hasil *training* yang sudah dilakukan, penulis menguji coba langsung menggunakan *data* hasil *splitting* untuk diuji coba. Untuk mendapatkan hasil performa model yang sudah di-*training* untuk melakukan *object detection* secara lebih jelas dan terukur, akan dilakukan menghitung nilai *mAP* (*mean Average precission*), *precision*, *recall*, *F1-score*, dan nilai *inference time*. *mAP* berfungsi

untuk mengevaluasi kinerja keseluruhan dari sebuah model *object detection* dengan menghitung rata-rata dari nilai AP menggunakan *IoU threshold* tertentu.

$$mAP = \frac{\sum_{i=1}^n AP_i}{n} \quad (2.7)$$

Kecepatan model dapat diukur dengan melihat nilai waktu inferensinya, yang mencerminkan seberapa cepat model dapat melakukan deteksi objek. Semakin cepat prosesnya, semakin baik performanya dalam penggunaan *real-time*.

Untuk *precision* sendiri adalah bentuk *matrix* penilaian yang digunakan untuk menghitung dan melihat nilai sebuah *prediction* yang benar berdasarkan nilai *true positive* dan *false positive*.

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (2.8)$$

Recall menggambarkan sejauh mana suatu model dapat memprediksi semua True Positive yang ada, yaitu sejauh mana model mampu menemukan semua objek yang seharusnya diprediksi.

$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (2.9)$$

Untuk melihat keterkaitan antara presisi (*precision*) dan *recall*, untuk mencapai keseimbangan dimana model tidak hanya melakukan prediksi dengan benar (presisi), tetapi juga menemukan semua objek yang seharusnya diprediksi (*recall*). Tujuan utamanya adalah memprediksi semua objek dengan benar tanpa mengabaikan satupun atau melakukan prediksi berlebihan. untuk melakukan perhitungan *F1 Score* menggunakan persamaan berikut

$$F1 - Score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.10)$$