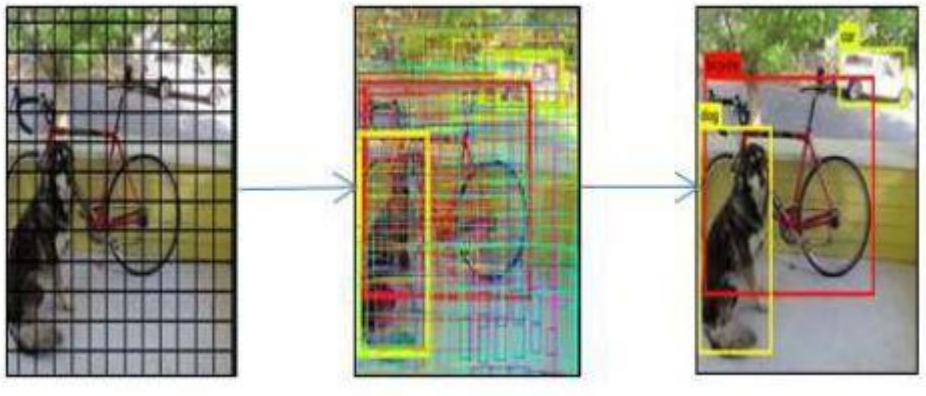
# BAB II

# TINJAUAN PUSTAKA

## II.1. Algoritma YOLO

Algoritma *You Only Look Once* (YOLO) adalah metode deep learning merupakan bagian dari *machine learning* yang sedang terkenal dalam pendeteksian suatu objek atau wajah. YOLO dikenal sebagai metode deteksi tercapai dengan akurasi tinggi. Dengan klasifikasi *multi purpose* dalam melakukan sebuah pendeteksian (Salamah, I. et al., 2022).

Pada Algoritma YOLO (*You Only Look Once*), masukan citra akan membagi dengan ukuran SxS dalam satu grid. Model YOLO didasari dengan regresi yang memprediksi menggunakan *class object* dan *bounding box* untuk menentukan lokasi dari sebuah objek pada citra dalam satu kali algoritma. Regresi model YOLO dapat dilihat pada gambar II.1.



**Gambar II.1. Regresi Model YOLO**

(Sumber: https://binus.ac.id/bandung/2021/01/deteksi-dan-klasifikasi-objek-berbasis-yolo-dalam-rekaman-video/

Model YOLO (*You Only Look Once*) mendeteksi objek dengan menggunakan *unified model* dimana sebuah *single convolutional network* memprediksi beberapa *bounding boxes* (kotak pembatas) serta probabilitas kelas di dalam kotak-kotak tersebut secara bersamaan. Pertama-tama, sistem YOLO membagi citra input ke dalam grid S × S. Jika pusat dari sebuah objek jatuh di dalam salah satu sel grid, maka sel grid itu bertanggung jawab untuk mendeteksi objek tersebut. Setiap sel grid memprediksi *bounding boxes* dan *confidence score* dari tiap *bounding box* tersebut. (Khairunnas, et al., 2021)

Faktor penentu prediksi akhir adalah *class confidence score* yang didapat, berdasarkan probabilitas kondisional kelas dan *box confidence score*. *Class confidence score* mengukur nilai kepercayaan pada klasifikasi dan lokalisasi objek. *Class confidence score* memberi nilai kepercayaan kelas spesifik untuk setiap kotak, yang mengkodekan kemungkinan kelas yang muncul di kotak dan seberapa sesuainya kotak yang diprediksi dengan objek . Jika tidak ada objek yang terdeteksi maka nilai *confidence* adalah nol. (Khairunnas, et al., 2021).

Pengujian dengan metode YOLO pada kasus deteksi jenis mobil, dilakukan berdasarkan 4 tahap *epoch* (iterasi) yang berbeda. Pengujian dilakukan dengan *epoch* 25, 50, 75, dan 100. Hasil akurasi klasifikasi jenis mobil yang didapat dengan epoch 50 diantaranya: mobil sedan (43,887%), mobil SUV (43,887%), mobil MPV (42,946%), mobil bus (45,329%). Namun, setelah melakukan pengembangan pengujian dengan metode Faster R-CNN (*Region-based Convolutional Neural Network*) dan YOLO, akurasi naik dengan *epoch* yang sama yaitu *epoch* 50 diantaranya: mobil sedan (71,724%), mobil SUV (71,473%), mobil

MPV (71,222%), mobil bus (75,360%) (Shianto, K.A., et al., 2019).

Pengujian dengan metode YOLO (*You Only Look Once*) pada kasus deteksi wajah manusia dengan dataset WIDER *face* berjumlah 1920 citra *training* dan 480 citra *testing*, didapatkan hasil terbaik adalah pada ukuran grid 3 x 3, ukuran citra 222x222 pixel dan jumlah *epoch* yang digunakan adalah 150. Hasil akurasi yang didapatkan adalah nilai *Precision* sebesar 0,253%, *Recall* sebesar 0,247%, dan *F1*-Score sebesar 0,25 %. Nilai akurasi data dipengaruhi oleh penggunaan model *Convolutional Neural Network* (CNN) yang dibangun dan keseimbangan data target positif dan negatif yang digunakan pada saat membangun model untuk deteksi wajah (Thoriq, Y.A., et al., 2023).

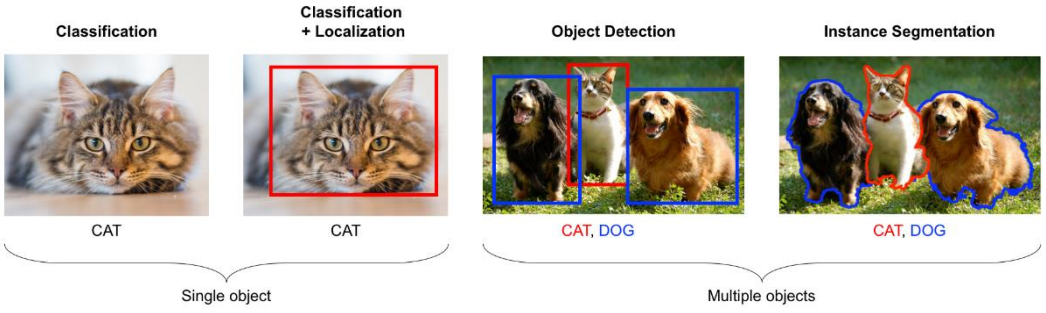
Pengujian dengan metode YOLO pada kasus mendeteksi nyeri bayi atau tingkat rasa nyeri pada bayi yang dibangun dengan *framework pytorch* dan algoritma YOLOv3 dengan dataset terdiri dari 600 emosi bayi sedih, netral dan nyeri dengan rentang umur 0-12 bulan dengan hasil mAP (*mean Average Precision*) dengan IoU (*Intersection over Union*) sebesar 0,5 diantaranya: sedih (97,9%), netral (99,2%), nyeri (96.9%) dan akurasi model 70%. Dengan kata lain, algoritma YOLO sudah dapat mendeteksi dengan sangat baik terhadap model yang diberikan dan saat dilakukan pembuktian dengan data acak pengambilan langsung di Puskesmas Imogiri I dan juga referensi lain terbukti nilai akurasi menjadi 90 % (Abuzairi, T., et al., 2019).

## II.2. Arsitektur YOLO

YOLO (*You Only Look Once*) adalah algoritma pembelajaran mendalam untuk pendeteksian objek yang menggunakan pendekatan yang berbeda dibandingkan dengan algoritma lain dengan menerapkan satu jaringan saraf ke seluruh objek pada gambar (Aprilino, A., et al., 2022).

Arsitektur YOLO dirancang untuk deteksi objek waktu nyata. Ini mengimplementasikan sistem deteksi menggunakan *classifier* atau *locator* yang digunakan kembali untuk deteksi. Gambar dibagi menjadi skala atau sel kisi tergantung pada ukuran input. Citra kemudian dideteksi berdasarkan skor tertinggi dari skala yang dibagi (Salamah, I., et al., 2022).

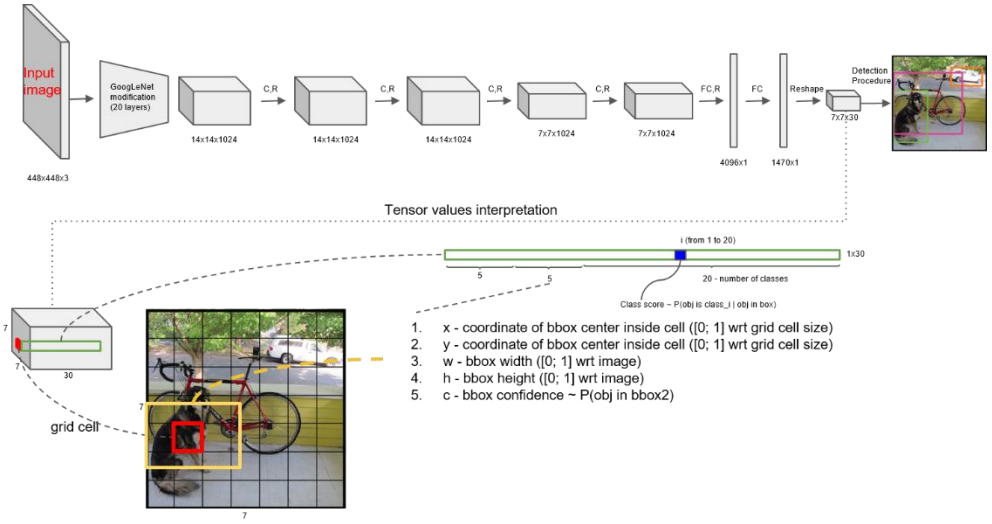
Selain itu, dengan munculnya YOLO, berbagai aplikasi telah memanfaatkan YOLO untuk deteksi dan pengenalan objek dalam berbagai konteks dan bekerja dengan sangat baik dibandingkan dengan detektor model lainnya yang berhubungan dengan pengenalan objek. (Diwan, T., et al., 2022).




**Gambar II.2. Klasifikasi, Lokalisasi dan Segmentasi Pada Citra Objek Tunggal dan Banyak**

(Sumber:https://www.oreilly.com/api/v2/epubs/9781788295628/files/assets/687bbea7-9a5b-49ed-8cc4-922287baf429.png)

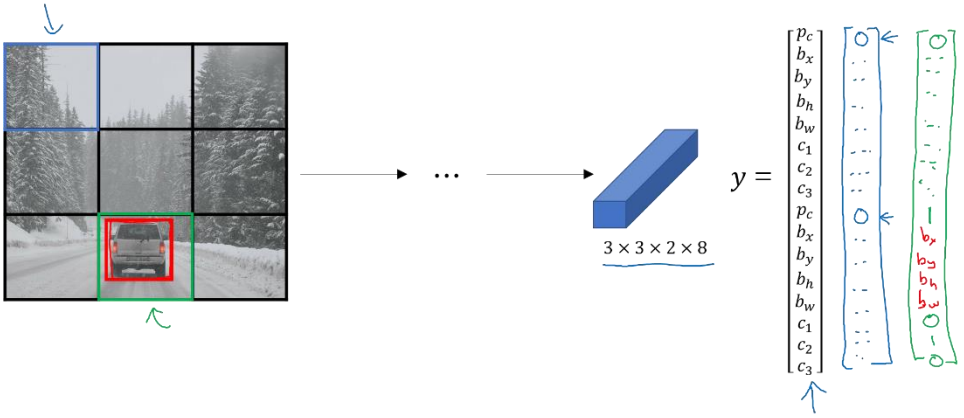
Versi dasar YOLO (*You Only Look Once*) tidak memiliki solusi untuk kesalahan pelokalan dan versi kedua gagal mendeteksi objek berukuran lebih kecil. Versi ketiga YOLO mencoba mengatasinya (Aprilino, A., et al., 2022). Hal tersebut dapat dilihat pada gambar II.3, dimana versi ketiga YOLO dapat mendeteksi dengan hasil yang akurat untuk objek berukuran sedang dan besar.




**Gambar II.3. Sistem Prediksi YOLO**

(Sumber: https://oi.readthedocs.io/en/latest/_images/training_yolo_detail.png)

Struktur yang menentukan mana dari dua kotak pembatas yang lebih baik untuk satu objek untuk satu kelas dalam satu sel *Grid*, x dan y merupakan titik pusat kotak pembatas, w dan h adalah panjang horizontal dan vertikal dari kotak pembatas, sedangkan c adalah probabilitas bahwa suatu objek ada di kotak pembatas.

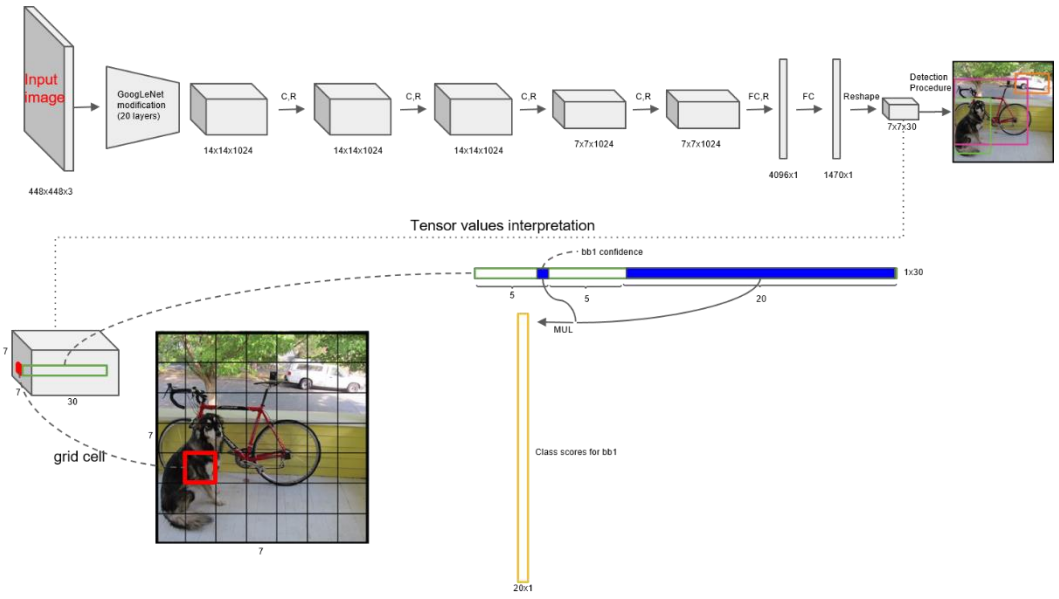



**Gambar II.4. Proses Sistem YOLO**

(Sumber: https://oi.readthedocs.io/en/latest/_images/predict_by_yolo.png)

Jika dilihat pada gambar II.4, pada prediksi pertama, vektor biru nilainya adalah 0, sehingga dapat dinilai tidak ada objek yang terdeteksi. Selanjutnya, prediksi pada vektor hijau nilainya adalah 1, hal tersebut terlihat pada kotak pembatas dengan memprediksi mobil dan sisanya adalah bx, by, bh dan bw.

Nilai probabilitas untuk nilai dan kelas ditampilkan, sehingga kotak pembatas untuk suatu objek dapat ditarik. Saat gambar dimasukkan ke model yang dilatih, hasilnya adalah *Tensor* berukuran 7x7x30 yang nantinya dapat menunjukkan tingkat probabilitas. Hal tersebut dapat dilihat pada gambar II.5.

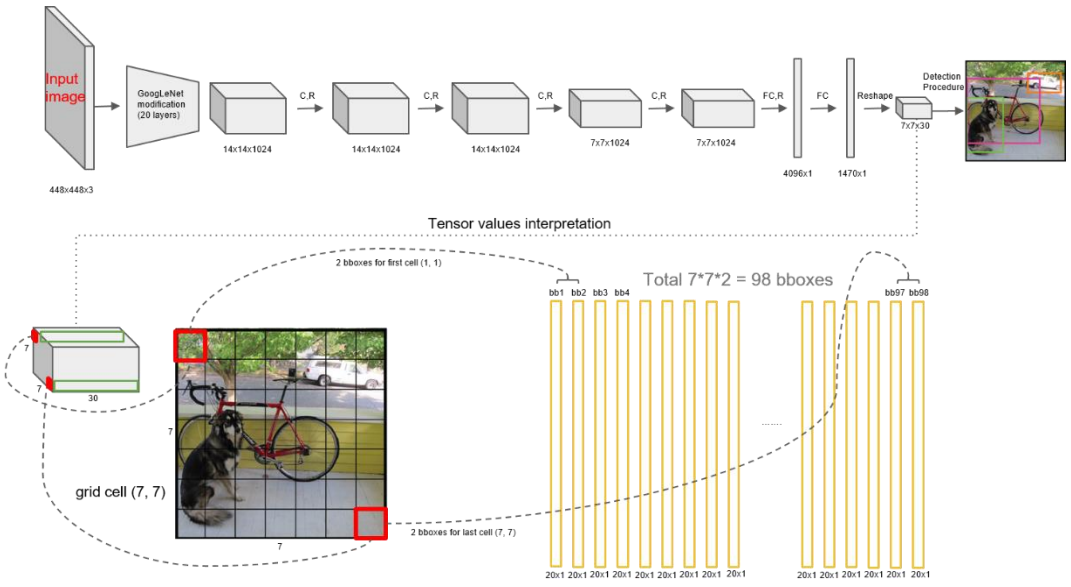



**Gambar II.5. Cara Kerja Sistem YOLO Dalam Memprediksi**

(Sumber: https://oi.readthedocs.io/en/latest/_images/yolo_inference_1.png)

Pada gambar II.5, tampak bagian 5 yang pertama adalah informasi tentang kotak pembatas pertama, 5 berikutnya adalah informasi tentang kotak pembatas kedua dan 20 berikutnya adalah nilai yang menunjukkan tingkat probabilitas bahwa objek yang sesuai dengan Grid saat ini adalah dari suatu kelas.

Proses mengalikan skor konferensi dari setiap kotak pembatas di setiap kisi dengan 20 probabilitas kelas menghasilkan 98 vektor berukuran 20x1. Sehingga dapat disimpulkan bahwa nilai yang dikalikan merupakan probabilitas objek dari kelas tertentu yang muncul dalam satu kotak pembatas di setiap *Grid*, seperti yang dapat dilihat pada gambar II.6.

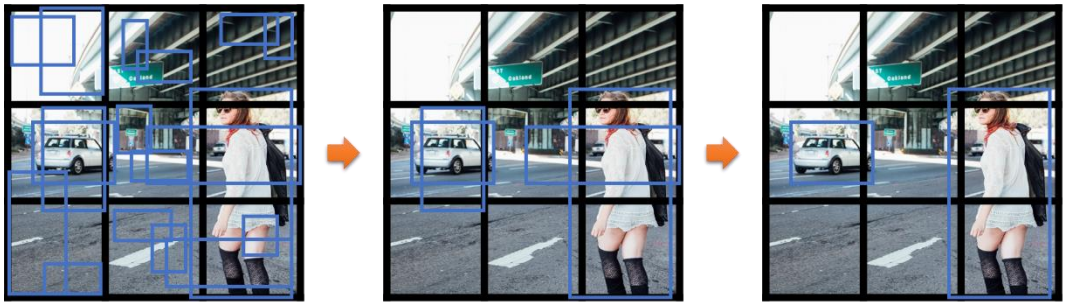



**Gambar II.6. Hasil Bounding Box Pada Deteksi YOLO**

(Sumber: https://oi.readthedocs.io/en/latest/_images/yolo_inference_2.png)

Kemudian, setelah *bounding box* dipetakan berdasarkan nilai *confidence* (kepercayaan) yang merujuk pada skor atau probabilitas hasil, YOLO (*You Only Look Once*) akan memprediksi kelas dari objek yang terdapat pada bounding box tersebut beserta kemungkinannya, sehingga terbentuklah *class probability map*.

Dari sekian banyak *bounding box* yang dihasilkan, untuk mendapatkan *bounding box* beserta kelas objek dengan probabilitas yang tinggi, maka dari seluruh hasil prediksi tersebut, hanya yang melampaui *threshold* (ambang batas) saja yang digunakan. Jika terdapat duplikasi *bounding box*, maka *Non-max Suppresion* (NMS) berperan untuk menghilangkan duplikat tersebut. Seperti yang tampak pada gambar II.7. (Aprilino, A., et al., 2022)

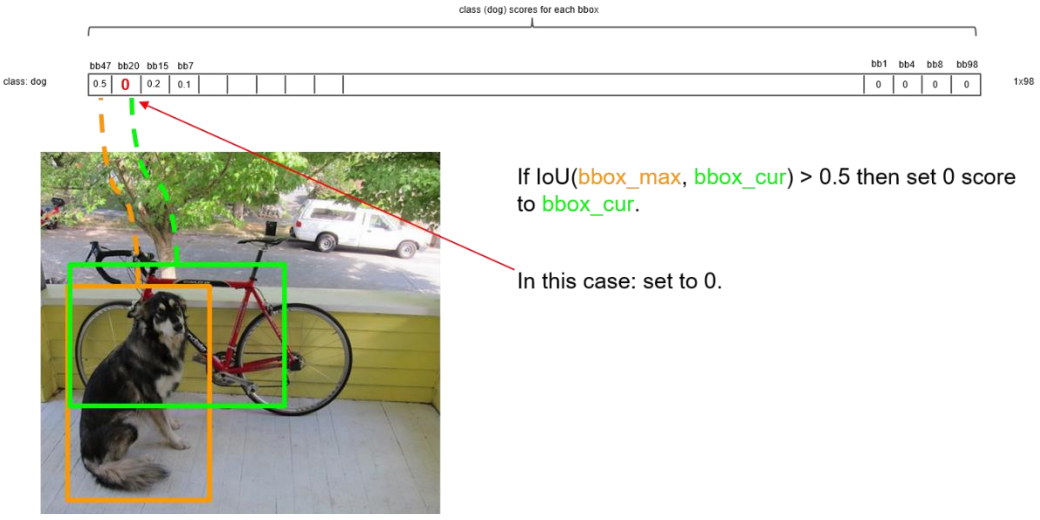


**Gambar II.7. Non-max Suppresion Pada YOLO**

(Sumber: https://oi.readthedocs.io/en/latest/_images/non-max_sup_by_yolo.png)

Dua kotak pembatas diprediksi untuk setiap kisi. Di antara kotak pembatas yang memiliki probabilitas terendah akan dieliminasi terlebih dahulu. Kemudian, di antara kotak pembatas dari setiap kelas, hapus semua kotak pembatas yang tumpang tindih dengan nilai terbesar yang akan muncul. Proses Ini adalah *Non-max suppression*, dan pada gambar II.7 di atas, karena ada 3 kelas, *Non-max suppression* dilakukan sebanyak 3 kali. Akibatnya, kotak pembatas dengan probabilitas tinggi dapat mendeteksi mobil dan orang, seperti yang ditunjukkan pada gambar II.7 paling kanan.

Seperti yang terlihat pada gambar II.7, kotak pembatas yang paling mungkin terletak di paling kiri. Kemudian, sambil mencentang kotak pembatas berikutnya, jika kotak pembatas tertinggi dan nilai IoU (*Intersection over Union*) melebihi 0,5, ubah semua nilainya menjadi 0. Jika proses ini dilakukan untuk semua kotak pembatas, semua nilai kotak pembatas duplikat dapat diatur ke 0.

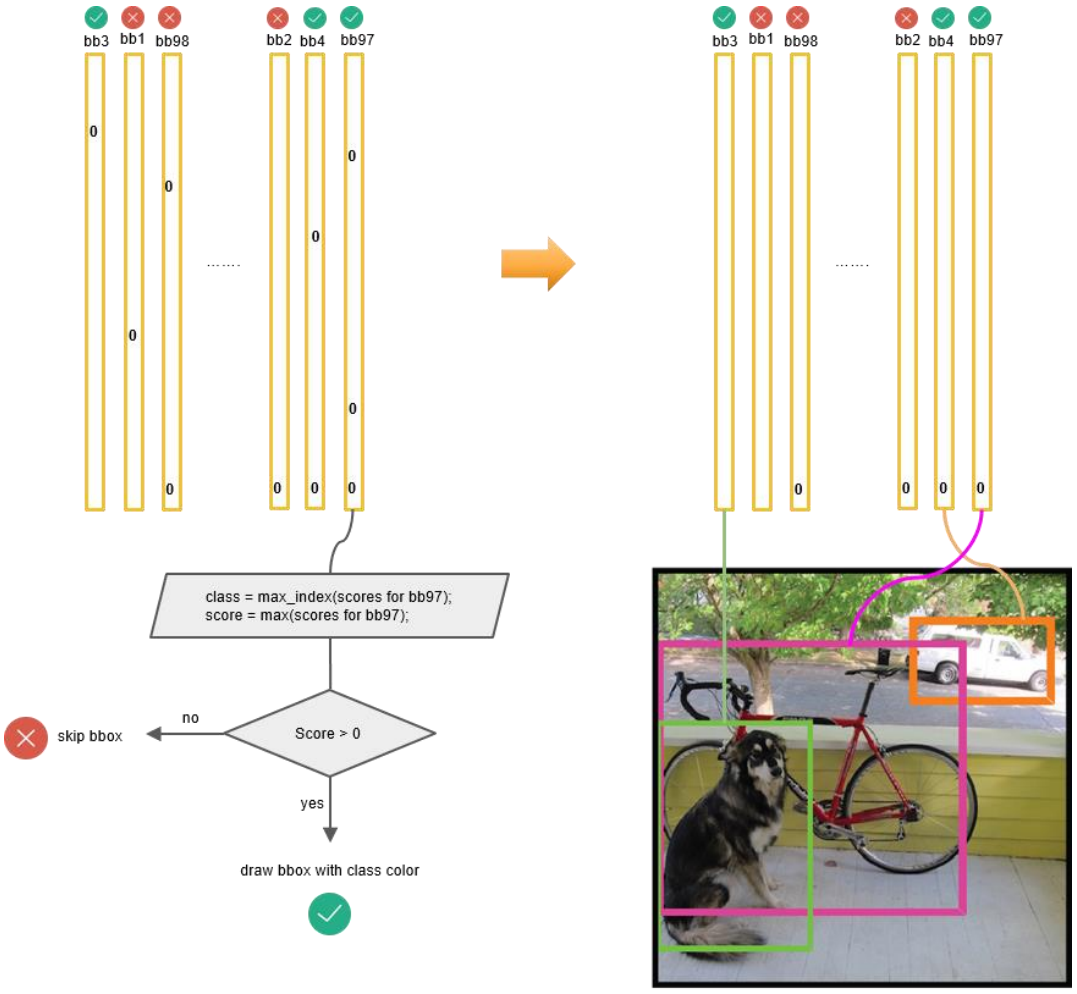



**Gambar II.8. Non-max Suppresion Detail**

(Sumber: https://oi.readthedocs.io/en/latest/_images/nms_for_yolo.png)

Kemudian, seperti yang ditunjukkan pada gambar II.8, skor terbesar untuk setiap kotak pembatas ditemukan dan hanya kotak pembatas yang ada yang diekstraksi. Kemudian, jika hanya kotak pembatas yang tersisa yang akhirnya ditarik, deteksi objek selesai.

Algoritma deteksi objek sering menggunakan Penekanan *Non-Maksimum Suppression* (NMS). Ini berperan dalam tahap akhir deteksi objek untuk mendapatkan efek terbaik dengan menentukan ukuran IoU (*Intersection Over Union*) untuk menghilangkan kandidat yang berlebihan kotak dan menemukan kotak pembatas terbaik dengan skor tertinggi dan tingkat tumpang tindih melebihi ambang batas untuk mendapatkan efek terbaik. (Tianjiao, L., et al., 2020)




**Gambar II.9. Deteksi Sistem YOLO**

(Sumber: https://oi.readthedocs.io/en/latest/_images/yolo_inference_4.png)

## II.3. Tahapan Deteksi YOLO

*You Only Look Once* (YOLO) memiliki beberapa tahapan dalam mendeteksi sebuah objek (Pamungkas, B., et al., 2021), adapun tahapan tersebut yakni:

1. Membagi citra dalam regian/grid s x s . Grid tersebut bertanggung jawab mendeteksi objek. Pada tiap grid juga akan diprediksi *bounding box* beserta

nilai *confidence*. Nilai *confidence* ini menunjukkan seberapa yakin *bounding box* tersebut berisi objek dan seberapa akurat prediksinya. Nilai *confidence* diperoleh melalui persamaan:

$$Conf\left(class\right) = \Pr\left(class\right) x \;\; IOU_{Pred}^{Truth}$$

Pr(class) adalah probabititas objek yang muncul dalam suatu region dan rasio tumpang tindih (*Intersection over Union*) antara kotak prediksi dan kotak *ground truth*. *Pred* adalah luas area dalam kotak prediksi. *Truth* adalah area dalam *ground truth*. Semakin besar nilai IoU , maka semakin tinggi tingkat akurasi pendeteksiannya.

2. Setiap *bounding box* memiliki 5 nilai informasi yaitu x,y,w,h dan c. Nilai x dan y adalah koordinat titik tengah *bounding box* yang terprediksi, nilai w dan h adalah rasio ukuran lebar dan tinggi relatif terhadap grid, dan c adalah nilai *confidence bounding box* tersebut.
3. Pada algoritma YOLO (*You Only Look Once*), tiap *grid* akan memprediksi nilai *class* probabilitas jika diprediksi terdapat objek didalamnya.

$$\Pr\left(Class_i \left| Object\right.\right) x \Pr\left(Object\right) x \;\; IOU_{Pred}^{Truth} = \Pr\left(Class_i\right) x \;\; IOU_{Pred}^{Truth}$$