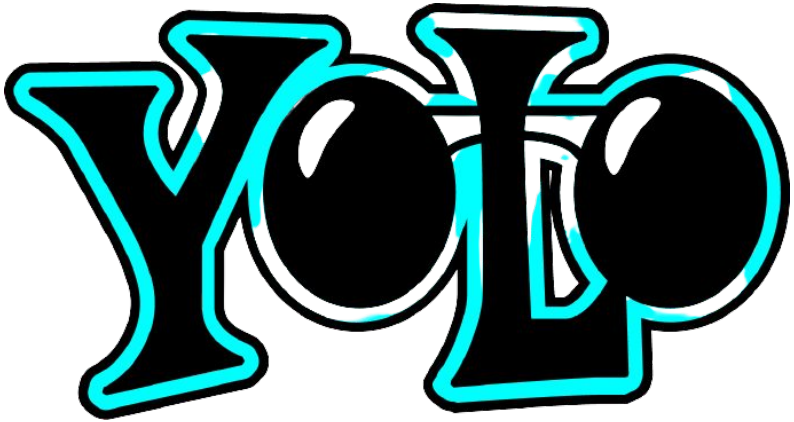
YOLO Theory

Christoffel Cleon / C14180064
Albert Bayu Sani / C14180079
Danny Yurista / C14180122



Paper yang digunakan :

- You Only Look Once:Unified, Real-Time Object Detection
- Oleh : Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

  Link :
  https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf

- YOLO9000
- Oleh : Joseph Redmon, Ali Farhadi

  Link : https://arxiv.org/pdf/1612.08242.pdf

YOLO

YOLO (*You Only Look Once*) pertama kali diciptakan oleh **Joseph Redmon** pada tahun 2015 adalah system deteksi objek secara *real time* berdasarkan CNN (*Convolutional Neural Network*)

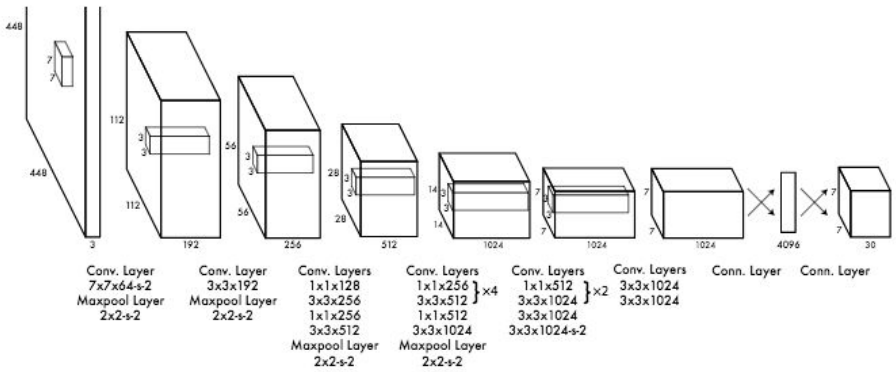
Apa itu YOLO?

YOLO singkatan dari You Only Look Once. YOLO merupakan algoritma yang berdasarkan pada regresi dimana dalam sekali proses running tersebut akan menghasilkan output prediksi kelas dan bounding box untuk setiap objek

Ide Dasar YOLO

Yolo membagi gambar atau video yang di input menjadi S x S grid atau persegi. Inovasi dari YOLO adalah mereformasi framework dari Region Proposal detection : seri R-CNN perlu menghasilkan Region Proposal untuk melengkapi proses klasifikasi dan regresi

Struktur Jaringan YOLO



YOLO v2

Joseph Redmon dan Ali Farmadi kemudian memutuskan untuk membuat metode baru untuk memanfaatkan jumlah besar data yang sudah disediakan untuk digunakan dan memperluas ruang lingkup *system* deteksi saat ini. Mereka juga mengusulkan algoritma untuk melakukan training data-data berbeda yang diinginkan sesuai dengan keinginan untuk proses deteksi dan klasifikasi objek. Metode dan algoritma tersebut adalah YOLOv2, dimana disediakan 9000 objek berbeda dan menghasilkan proses deteksi dan klasifikasi yang lebih akurat dibandingkan YOLOv1.Mereka menyederhanakan jaringan dan membuat representasi proses menjadi lebih mudah dipelajari.

Batch Normalization

Batch normalization mengarah ke peningkatan yang signifikan dalam konvergensi ketika menghilangkan kebutuhan bentuk regulasi yang lain. Dengan menambahkan *batch normalization* pada semua *convolutional layers* di YOLO kita mendapatkan peningkatan lebih dari 2% di maP. *Batch normalization* juga membantu meregulasi / mengatur model. Menggunakan *batch normalization* kita dapat menghilangkan *dropout* dari model tanpa *overfitting*.

High Resolution Classifier

Pada YOLO v1 training jaringan *classifier* pada 224 x 224 dan pada YOLO v2 ditingkatkan resolusinya menjadi 448 x 448 untuk proses deteksi. Pertama-tama dalam proses klasifikasi menggunakan resolusi 448 x 448 untuk 10 *epoch*. Ini memberikan waktu proses *filters* lebih lama dengan resolusi yang lebih tinggi untuk *input*. Hasil yang didapatkan dengan menggunakan resolusi ini adalah mendapatkan peningkatan hampir 4% tingkat akurasi maP.

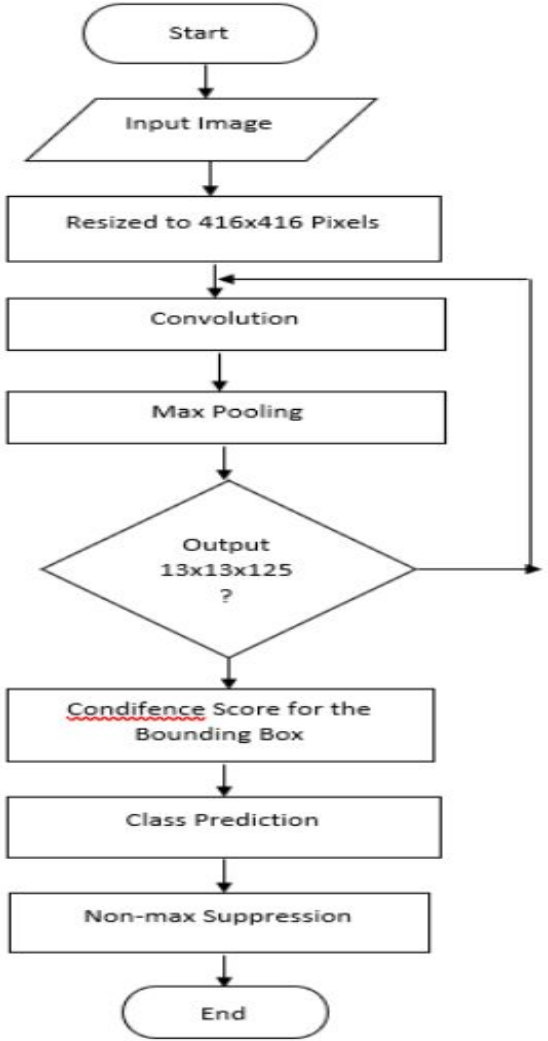
Convolutional with Anchor Boxes

Pada YOLOv1, *fully connected layer* digunakan untuk memprediksi koordinat *bounding box* secara langsung setelah *convolutional layer*. **YOLOv2 menghapus *fully connected layer* dengan menggunakan ide faster R-CNN, dan menambahkan *Anchor Boxes***, yang secara efektif meningkatkan tingkat recall. Pertama YOLO menghilangkan satu *pooling layer* untuk membuat *output* dari *convolutional layer* menjadi lebih tinggi resolusinya.**YOLO juga memperkecil jaringan dari 448 x 448 menjadi 416 x 416**. YOLO melakukan ini karena ingin jumlah ganjil pada lokasi pada *feature map* sehingga ada sel pusat tunggal. *Convolutional layers* pada **YOLO menurunkan sampel (downsample) atau *reduction factor* gambar dengan *factor* 32 sehingga dengan menggunakan *input* gambar 416 kita mendapatkan *output feature map* sebesar 13 x 13.** Menggunakan *anchor boxes* juga meningkatkan akurasi. YOLO hanya dapat memprediksi 98 bounding boxes per image tetapi dengan *anchor boxes model* dapat memprediksi lebih dari 1000.

Multi-Scale Training

Multi-Scale Training. Original YOLO menggunakan input dengan resolusi 448 x 448. Dengan menambahkan *anchor boxes* YOLO mengubah resolusi menjadi 416 x 416. **Namun, sejak YOLO model hanya menggunkan *convolutional* dan *pooling layers* itu bisa mengubah ukurannya secara cepat.** Kita ingin YOLOv2 kuat untuk dijalankan pada gambar dengan ukuran-ukuran yang berbeda sehingga YOLO melakukan training ini kedalam *model*. Alih-alih memperbaiki ukuran *image* pada *input*, kita mengubah jaringan setiap beberapa iterasi. Setiap 10 *batch* pada jaringan secara acak memilih ukuran dimensi gambar baru. Sejak *model downsample* menggunakan *factor* 32, kita tarik hasil dari kelipatan 32 yaitu 32: {320, 352, … 608}. Jadi pilihan terkecil adalah 320 x 320 dan terbesarnya 608 x 608. Ini artinya dengan jaringan yang sama YOLO dapat memprediksi deteksi dengan resolusi yang berbeda sehingga YOLO v2 menawarkan *tradeoff* yang mudah antara kecepatan dan akurasi pada ukuran yang lebih kecil.

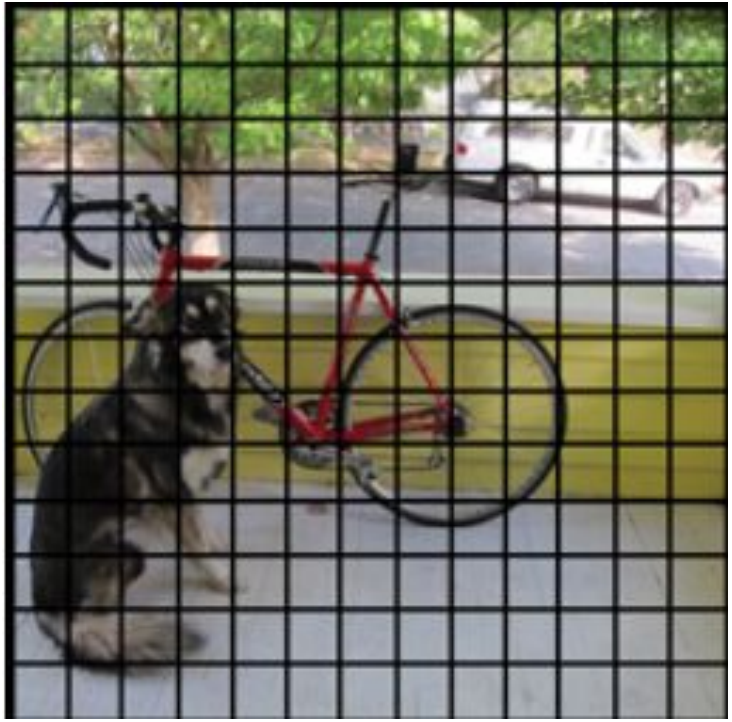
Langkah-langkah kerja YOLO



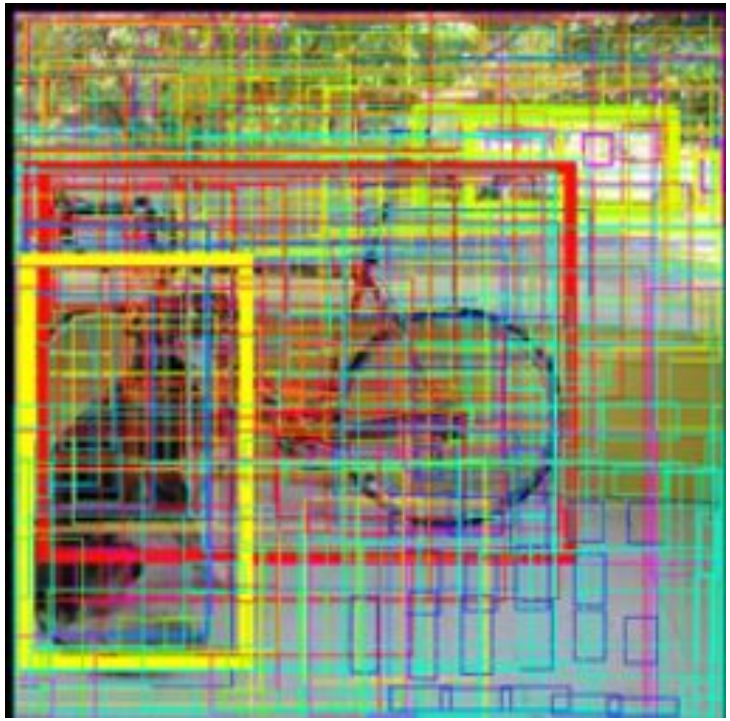
Convolution and Max Pooling

Layer	kernel	stride	output shape
Input			(416, 416, 3)
Convolution	3×3	1	(416, 416, 16)
MaxPooling	2×2	2	(208, 208, 16)
Convolution	3×3	1	(208, 208, 32)
MaxPooling	2×2	2	(104, 104, 32)
Convolution	3×3	1	(104, 104, 64)
MaxPooling	2×2	2	(52, 52, 64)
Convolution	3×3	1	(52, 52, 128)
MaxPooling	2×2	2	(26, 26, 128)
Convolution	3×3	1	(26, 26, 256)
MaxPooling	2×2	2	(13, 13, 256)
Convolution	3×3	1	(13, 13, 512)
MaxPooling	2×2	1	(13, 13, 512)
Convolution	3×3	1	(13, 13, 1024)
Convolution	3×3	1	(13, 13, 1024)
Convolution	1×1	1	(13, 13, 125)

Output 13 x 13 x 125



Confidence Score for the Bounding Box



Class Prediction & Non-max Suppression