IMPLEMENTASI ALGORITMA YOLO *(YOU ONLY LOOK ONCE)* UNTUK DETEKSI API

[1]Mada Lazuardi Nazilly, [2]Basuki Rahmat [3]Eva Yulia Puspaningrum
Email : [1]madalazuardy@gmail.com, [2]basukirahmat.if@upnjatim.ac.id,
[3]evapuspaningrum.if@upnjatim.ac.id

[1,2,3] Teknik Informatika, Fakultas Ilmu Komputer, Universitas Pembangunan Nasional "Veteran" Jawa Timur

Abstrak

Kebakaran hutan dan lahan menjadi masalah yang selalu terjadi setiap tahun di wilayah Indonesia. Penyebabnya adalah cuaca panas pada musim kemarau yang dapat menyebabkan munculnya bara api maupun praktek pembukaan lahan pertanian atau perkebunan baru pada kawasan hutan yang dilakukan dengan cara pembakaran liar. Hal ini dapat dicegah dengan memanfaatkan *Drone*, yaitu teknologi pesawat tanpa awak yang dapat digunakan untuk memantau titik-titik api pada kawasan hutan/lahan melalui udara. Penelitian ini dilakukan untuk mengembangkan sistem deteksi api berbasis *drone* dengan menerapkan algoritma YOLO (*You Only Look Once*) yang menggunakan jaringan syaraf konvolusional untuk mendeteksi objek api pada citra. Hasil penelitian menunjukkan bahwa penggunaan algoritma YOLO berhasil untuk mendeteksi api dengan cukup baik dengan menghasilkan rata-rata nilai *confidence* sebesar 0.66 pada pengujian video. Sedangkan untuk pengujian menggunakan 100 citra pada nilai *threshold* 0.30, menghasilkan skor *precision* sebesar 98%, skor *recall* sebesar 95% dan skor *accuracy* sebesar 95% serta skor *mean average precison* (mAP) sebesar 72.63%.

Kata kunci : Kebakaran Hutan dan Lahan, *Drone,*, Deteksi Api, Algoritma YOLO (*You Only Look Once*)

1. PENDAHULUAN

Kebakaran hutan dan lahan menjadi masalah yang selalu terjadi setiap tahun di wilayah Indonesia. Penyebabnya adalah cuaca panas pada musim kemarau yang dapat menyebabkan munculnya bara api maupun praktek pembukaan lahan pertanian atau perkebunan baru pada kawasan hutan yang dilakukan dengan cara pembakaran liar. Berdasarkan data Kementrian Lingkungan Hidup dan Kehutanan (KLHK) Republik Indonesia, luas area hutan dan lahan yang terbakar pada periode 2014-2019 adalah lebih dari 3 juta hektar di seluruh wilayah Indonesia dengan kerugian materi pada tahun 2015 sudah mencapai 221 triliun rupiah dan menyebabkan ribuan orang terdampak bencana 'kabut asap' yang dapat berlangsung selama hitungan bulan. Tentu ini merupakan masalah yang harus diselesaikan dengan solusi yang tepat.

Salah satu solusi yang dapat dilakukan adalah tindakan pencegahan yaitu pemantauan titik api pada kawasan hutan dan lahan melalui udara. Hal ini tentu dapat dilakukan dengan memanfaatkan teknologi *drone* atau pesawat tanpa awak. Pesawat ini dikendalikan secara otomatis melalui program komputer (Utomo, 2017). Dalam hal ini, *drone* dapat dimanfaatkan dengan membawa kamera yang digunakan untuk mengambil citra foto maupun video untuk memantau keadaan lingkungan sekitar. Selanjutnya, citra maupun video tersebut dapat dianalisa terdapat titik api atau tidak.

Salah satu cara mendeteksi api pada citra digital adalah dengan melakukan teknik peningkatan kualitas citra (*image enhancement*). Dalam penggunaan teknik ini, fitur-fitur api dapat digunakan dalam mendeteksi api secara efisien seperti : warna api, asap, percikan api, tekstur api, sebaran area api, dan deteksi tepi (Pritam & Dewan, 2017). Selanjutnya,

citra hasil dari *image enhancement* dapat dilatih dengan memanfaatkan algoritma *deep learning* yaitu algoritma YOLO (*You Only Look Once)*.

Algoritma YOLO (*You Only Look Once)* adalah algoritma *deep learning* yang memanfaatkan jaringan syaraf konvolusional (CNN) dalam mendeteksi objek. Algoritma ini akan membagi citra ke dalam grid berukuran *s x s* yang kemudian pada tiap grid akan memprediksi *bounding box* serta peta kelas masing-masing grid. Apabila pada satu grid terprediksi objek, maka pada grid tersebut akan diprediksi *bounding box* yang mengelilingi objek tersebut. Nilai *confidence* akan dihitung pada masing-masing *bounding box* yang kemudian akan diseleksi berdasarkan nilai yang didapat. (Redmon, Divvala, Girshick, & Farhadi, 2016)

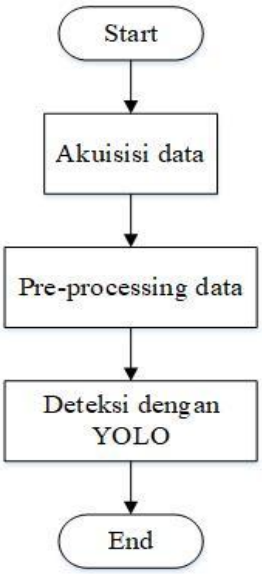
2. METODE PENELITIAN

2.1 Analisa Data

Data yang digunakan dalam penelitian ini adalah video hasil perekaman api yang diambil secara mandiri memanfaatkan *drone.* Video yang digunakan berjumlah 24 video dengan masing-masing berdurasi 10-30 detik dengan *framerate* 20 fps. Video kemudian dibagi menjadi 14 sebagai dataset pelatihan dan 10 video untuk pengujian.

Dataset pelatihan didapat dari ekstraksi frame video hasil perekaman api menggunakan *drone* dan sebagian dari internet. Dataset pelatihan berjumlah sebanyak 2000 citra, terbagi menjadi 2 yaitu Data Latih dan Data Validasi. Data latih yang digunakan adalah 1460 citra yang berasal dari ekstraksi frame dan 40 citra tambahan dari internet, sehingga total menjadi 1500 citra. Data validasi yang digunakan sebanyak 500 citra yang berasal dari ekstraksi frame video. Untuk data pengujian menggunakan 10 video uji dan 100 citra yang berasal dari ekstraksi frame video uji.

2.2 Analisa Sistem



Gambar 1. Alur kerja sistem

Pada Gambar 1. Alur sistem yang diusulkan berawal dengan proses Akuisisi data yaitu pengambilan dataset. Dataset diambil dari hasil ekstraksi frame video perekaman api menggunaakan *drone*. Setiap citra hasil ekstraksi frame tersebut selanjutnya masuk tahap *Pre-Processing,* pada tahap ini citra akan mengalami perbaikan dengan melakukan *contrast stretching* atau peregangan nilai kontras citra. Citra hasil peregangan kontras tersebut akan digunakan untuk melatih model jaringan pada algoritma YOLO. Hasil pelatihan ini adalah nilai bobot yang dapat digunakan untuk mendeteksi api pada citra.

2.3 Pre-processing Data

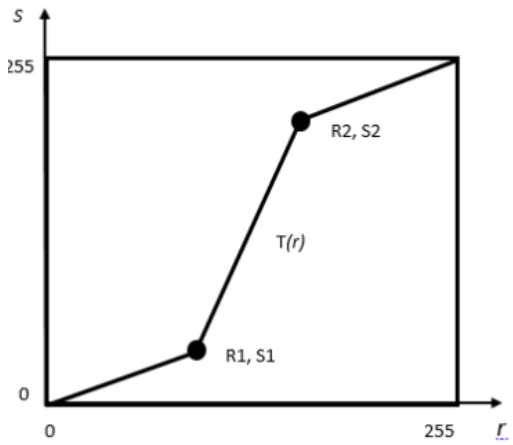
Pada tahap ini data citra akan mengalami perbaikan kualitas citra dengan menggunakan metode *contrast stretching* atau peregangan kontras. *Contrast stretching* merupakan teknik untuk mendapatkan citra dengan kontras yang lebih baik dari sebelumnya. Proses *contrast stretching* dilakukan bergantung pada *graylevel* satu pixel citra dan tidak bergantung pada piksel lain atau bersifat *point operations* (Wakhidah, 2011). *Point operations* dilakukan pada domain spasial, yakni operasi pada sumbu x dan y pada bidang kartesian, atau jika dalam citra disebut baris dan kolom [1].

Kontras citra merupakan distribusi pixel terang (*lightness*) dan pixel gelap (*darkness*) pada citra. Untuk menentukan transformasi citra digunakan 2 (dua) titik kontrol yaitu (*r1*,*s1*) dan (*r2*,*s2*). Nilai *r* adalah *graylevel* sebelum transformasi dan nilai *s* adalah *graylevel* sesudah transformasi. Untuk menghitung nilai hasil transformasi linear pada 2 titik kontrol (*r1*,*s1*) dan (*r2*,*s2*) dengan asumsi $r1 \leq r2$ dan $s1 \leq s2$ dapat menggunakan persamaan berikut (Nurliadi, Sihombing, & Ramli, 2016) :

$$\text{Untuk } 0 \leq \text{r} \leq r1, maka \;\; s = r\frac{s1}{r1} \tag{1}$$

$$\text{Untuk r1} \leq \text{r} \leq r2, maka \;\; s = s1 + \frac{(r - r1) * (s2 - s2)}{(r2 - r1)} \tag{2}$$

$$\text{Untuk r2} \leq \text{r} \leq 255, maka \;\; s = s2 + \frac{(r - r2) * (255 - s2)}{(255 - r2)} \tag{3}$$



Gambar 2. Transformasi kontras citra (Nurliadi, Sihombing, & Ramli, 2016)

Pada Gambar 2, merupakan visualisasi proses transformasi citra. Jika nilai *r1=s1* dan *r2=s2* maka bentuk transformasi akan berbentuk lurus karena tidak ada *graylevel* yang berubah. Jika $r1 \leq r2$ *dan* $s1 \leq s2$ maka fungsi akan menghasilkan nilai tunggal yang nilainya selalu naik [3]

2.4 Algoritma YOLO (*You Only Look Once*)

Algoritma YOLO merupakan algoritma *deep learning* untuk deteksi objek yang menggunakan pendekatan berbeda dari algoritma lain, yaitu menerapkan sebuah jaringan syaraf tunggal pada keseluruhan citra. YOLO mendeteksi sebuah objek dalam beberapa tahap yaitu [5]:

1. Membagi citra dalam region/grid berukuran *sxs*. Grid-grid tersebut bertanggung jawab untuk 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 :

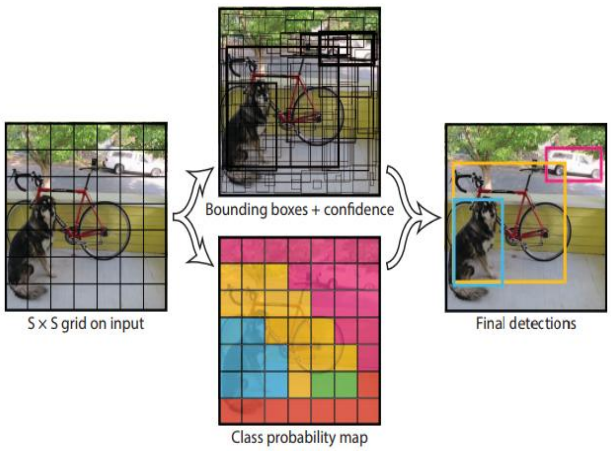
$$Conf(class) = \Pr(class) \times IOU_{Pred}^{Truth} \quad (4)$$

$Pr(class)$ adalah probabilitas objek yang muncul dalam suatu region dan IOU_{Pred}^{Truth} adalah 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.* Makin besar nilai IOU, maka makin tinggi tingkat akurasi pendeteksiannya [2].

2. Tiap *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, tiap grid akan memprediksi nilai *class probabilitas* jika diprediksi terdapat objek di dalamnya. Saat pengujian, YOLO akan mengkalikan nilai *class probability* dengan nilai *confidence* dari *bounding box*.

$$Pr\ (CLass_i|Object) \times \Pr(Object) \times\ IOU_{Pred}^{Truth} = Pr\ (Class_i) \times IOU_{Pred}^{Truth} \quad (5)$$

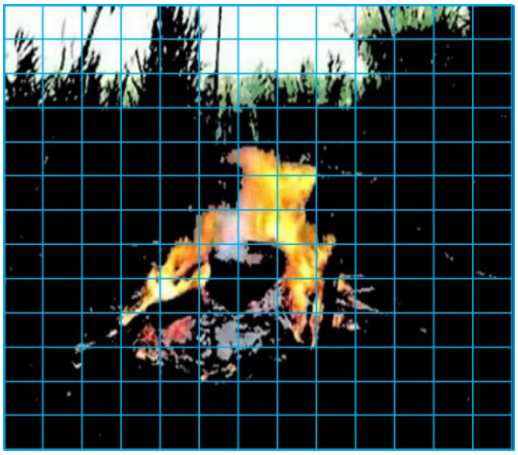
Sehingga menghasilkan nilai *confidence* kelas secara spesifik pada tiap *bounding box.* Nilai ini menunjukkan *class probability* yang muncul pada *bounding box* dan seberapa akurat *bounding box* memprediksi sesuai dengan objek [5].



Gambar 3. YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016)

2.5 YOLO Membagi Citra dalam Grid

Pada penelitian ini, algoritma YOLO yang digunakan adalah YOLOv2 yang membagi citra ke dalam grid berukuran 13 x 13 [5]



Gambar 4. Citra api yang dibagi dalam grid 13x13

Gambar 4 menunjukkan citra yang terbagi menjadi grid-grid berukuran 13x13. Tiap grid tersebut akan diprediksi 5 *anchor box* dengan 5+1 nilai yaitu *(x,y,w,h,c)* dan *class*. Karena pada penelitian ini hanya memprediksi 1 *class* yaitu api maka akan menghasilkan *feature map* dengan dimensi 13 x 13 x 30

2.6 Arsitektur CNN pada YOLO

Arsitektur jaringan syaraf konvolusional yang digunakan pada penelitian ini adalah arsitektur YOLOv2 atau YOLO9000 [5]. Berikut adalah arsitektur yang digunakan :

Tabel 1. Tabel Arsitektur CNN pada YOLOv2

Layer	Filter	Size	Stride	Output	Activation
Input				*416,416,3*	
Conv 1 (C1)	*32*	*3x3*	*1*	*416,416,32*	*Leaky ReLU*
Max Pooling 1(MP1)		*2x2*	*2*	*208,208,32*	
C2	*64*	*3x3*	*1*	*208,208,64*	*Leaky ReLU*
MP2		*2x2*	*2*	*104,104,64*	
C3	*128*	*3x3*	*1*	*104,104,128*	*Leaky ReLU*
C4	*64*	*1x1*	*1*	*104,104,64*	*Leaky ReLU*
C5	*128*	*3x3*	*1*	*104,104,128*	*Leaky ReLU*
MP3		*2x2*	*2*	*52,52,128*	
C6	*256*	*3x3*	*1*	*52,52,256*	*Leaky ReLU*
C7	*128*	*1x1*	*1*	*52,52,128*	*Leaky ReLU*
C8	*256*	*3x3*	*1*	*52,52,256*	*Leaky ReLU*
MP4		*2x2*	*2*	*26,26,256*	
C9	*512*	*3x3*	*1*	*26,26,512*	*Leaky ReLU*
C10	*256*	*1x1*	*1*	*26,26,256*	*Leaky ReLU*
C11	*512*	*3x3*	*1*	*26,26,512*	*Leaky ReLU*
C12	*256*	*1x1*	*1*	*26,26.256*	*Leaky ReLU*
C13	*512*	*3x3*	*1*	*26,26,512*	*Leaky ReLU*

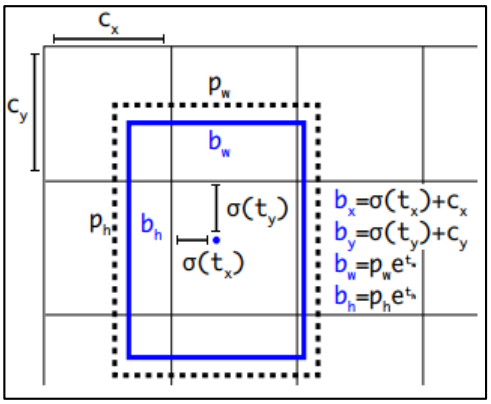
MP5		*2x2*	*2*	*13,13,512*	
C14	*1024*	*3x3*	*1*	*13,13,1024*	*Leaky ReLU*
C15	*512*	*1x1*	*1*	*13,13,512*	*Leaky ReLU*
C16	*1024*	*3x3*	*1*	*13,13,1024*	*Leaky ReLU*
C17	*512*	*1x1*	*1*	*13,13,512*	*Leaky ReLU*
C18	*1024*	*3x3*	*1*	*13,13,1024*	*Leaky ReLU*
C19	*1024*	*3x3*	*1*	*13,13,1024*	*Leaky ReLU*
C20	*1024*	*3x3*	*1*	*13,13,1024*	*Leaky ReLU*
Concat				*26,26,512*	
C21	*64*	*1x1*	*1*	*26,26,64*	*Leaky ReLU*
Flatten		*2x2*		*13,13,256*	
Concat				*13,13,1280*	
C22	*1024*	*3x3*	*1*	*13,13,1024*	*Leaky ReLU*
C23	*30*	*1x1*	*1*	*13,13,30*	*Linear*

Pada Tabel 1, terlihat bahwa arsitektur CNN yang digunakan terdiri dari 23 konvolusi layer, 5 *pooling* layer dan menghasilkan *feature map* berukuran 13x13x30 dengan tiap konvolusi layer menggunakan *batch normalization* dan fungsi aktifasi *leaky ReLU*. *Batch Normalization* berguna untuk menstabilkan model saat proses *training* dan dapat mempercepat nilai *loss* untuk konvergen, Sedangkan *aktivasi ReLU* digunakan karena cocok untuk model deep CNN [8]

2.7 Prediksi *Bounding Box*

Untuk memprediksi *bounding box*, YOLOv2 akan memprediksi koordinat titik tengah *bounding box* dengan relatif terhadap lokasi grid, dengan menggunakan *k-means clustering* dengan menggunakan nilai k = 5 yang memberikan perbandingan nilai *recall* dan kompleksitas model yang baik. Sedangkan untuk menghitung jarak *euclidean* YOLOv2 menggunakan pendekatan IOU dengan rumus[5]:

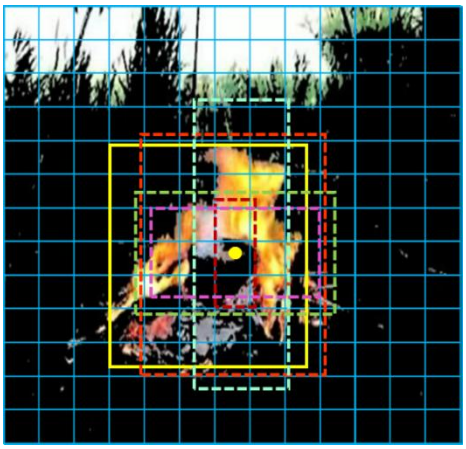
$$d(box, centroid) = 1 - IOU(box, centroid) \quad (6)$$



Gambar 5 Prediksi *Bounding Box* (Redmon & Farhadi, 2016)

Pada Gambar 5, menunjukkan prediksi *bounding box* pada YOLOv2. YOLOv2 memprediksi *bounding box* dari nilai-nilai anchor box yang terbentuk. Nilai *bx* diperoleh dari nilai sigma *tx*(ordinat x prediksi) dan nilai panjang dari *cx (cell x)*. Nilai *by* diperoleh dari nilai sigma *ty*(ordinat y prediksi) dan nilai panjang dari *cy (cell y)*. Nilai *bw* dan *bh*,

diperoleh dari nilai lebar dan nilai tinggi *bounding box*. Untuk prediksi objek diperoleh dari nilai *sigma to*



Gambar 6. *Bounding Box* yang terprediksi

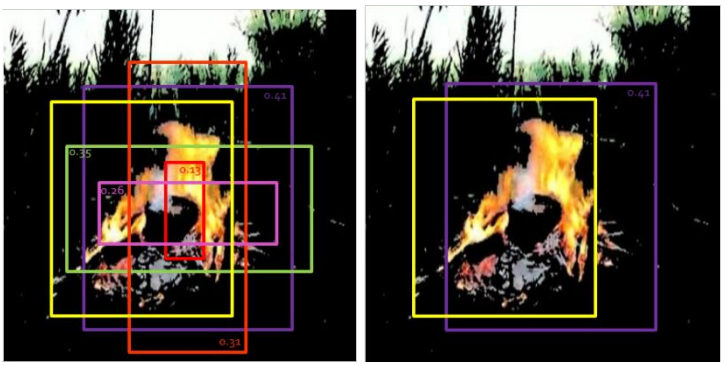
2.8 *Intersection Over Union (IOU)*

IOU menghitung luas area yang berpotongan lalu membaginya dengan luas area gabungan antar 2 *bounding box*. Nilai IOU digunakan untuk menentukan kesesuaian *bounding box* yang diprediksi dengan luas objek yang sesungguhnya pada citra (*ground truth*). IOU dapat dihitung dengan persamaan :

$$IOU = \frac{Area\ Overlap}{Area\ Union} \quad (7)$$

2.9 *Non-Max Supression*

Non-Max Suppression digunakan untuk menyeleksi *bounding box* yang muncul berlebihan pada objek yang sama dengan membandingkan nilai *confidence* masing-masing *bounding box*, hanya nilai *confidence* yang paling tinggi (maksimal) yang akan dipertahankan.



Gambar 7 *Non-Max Supression*

2.10 *Loss Function*

Untuk mengevaluasi *bounding box* yang terprediksi pada citra dapat menggunakan rumus perhitungan *Loss Function* sebagai berikut [5] :

$$Loss\ function = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2 \right]$$

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(\sqrt{w_i} - \sqrt{\hat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\hat{h_i}})^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C_i})^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C_i})^2$$

$$+ \sum_{i=0}^{S^2} 1_{ij}^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p_i}(c) \right)^2 \quad (8)$$

Pada baris pertama persamaan 8 merupakan perhitungan evaluasi dari prediksi koordinat titik tengah pada (x,y) *bounding box* jika terdapat objek pada sebuah grid, dengan (x_i , y_i) adalah koordinat objek yang sebenarnya (*ground truth*) dan ($\hat{x_i}$, $\hat{y_i}$) adalah koordinat *bounding box* yang terprediksi. Pada baris kedua, merupakan perhitungan evaluasi dari prediksi lebar dan tinggi (*widht* dan *height*) *bounding box*, dengan (w_i , h_i) adalah luas objek yang sebenarnya(*ground truth*) dan ($\hat{w_i}$, $\hat{h_i}$) adalah nilai *bounding box* yang terprediksi. Untuk baris ketiga merupakan perhitungan evaluasi nilai *confidence* dari *bounding box* yang terprediksi. Sedangkan pada baris keempat, merupakan evaluasi untuk prediksi kelas-kelas objek yang terdeteksi.

3 HASIL DAN PEMBAHASAN

Proses *training* menggunakan konfigurasi *yolov2-voc.cfg* dan bobot *yolov2-voc.weights* dan terbagi dalam 2 tahap yaitu *overfitting* dan *full training*. Tahap *overfitting* berguna dalam mendapat *loss* yang kecil. Selanjutnya, dilakukan *full training* dengan 2(dua) *learning rate* dan *batch* yang berbeda hingga *loss* mencapai konvergen. Maka, didapatkan hasil sebagai berikut :

3.1 Evaluasi Hasil Data Video

Gambar 8. Hasil Deteksi pada Video

Tabel 2. Tabel Hasil Deteksi pada Video

Video Uji	*Test 1*	*Test 2*	*Test 3*	*Test 4*	*Test 5*	*Test 6*	*Test 7*	*Test 8*	*Test 9*	*Test 10*	*Rata-rata*
Confidence	*0.55*	*0.65*	*0.62*	*0.57*	*0.67*	*0.73*	*0.74*	*0.68*	*0.67*	*0.7*	***0.66***

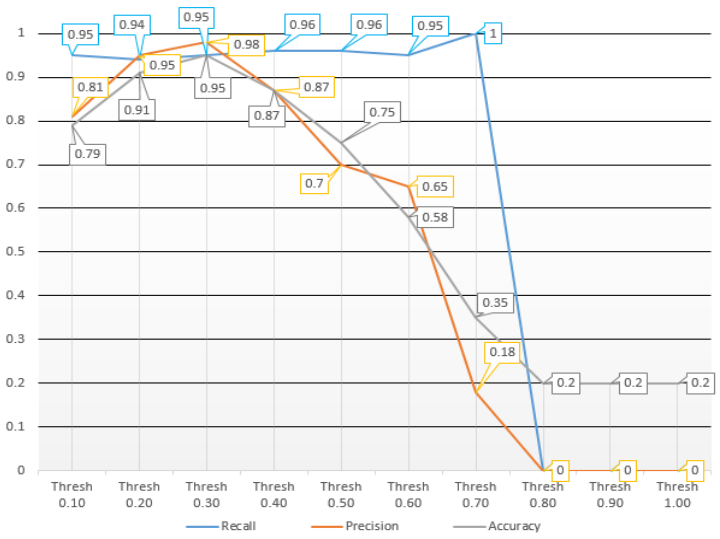
Pada pengujian yang dilakukan dengan 10 data video pada nilai *threshold*, didapati *bounding box* dapat terbentuk dari sekeliling objek api dengan menampilkan nilai *confidence*-nya pada tiap *frame*. Data pada Tabel 2, didapati bahwa nilai *confidence* terendah ada pada video test 1 yaitu 0.55 dan untuk nilai *confidence* tertinggi pada video test 7 dengan 0.74. Dengan mendapat rata-rata nilai *confidence* sebesar 0.66.

3.2 Evaluasi *Confusion Matriks*

Tabel 3 Tabel perbandingan *threshold*

Threshold	*TP*	*FP*	*FN*	*TN*	*Recall*	*Precision*	*Accuracy*
0.10	73	17	4	6	0.95	0.81	0.79
0.20	76	4	5	15	0.94	0.95	0.91
0.30	74	1	4	21	0.95	0.98	0.95
0.40	67	10	3	20	0.96	0.87	0.87
0.50	55	23	2	20	0.96	0.70	0.75
0.60	38	40	2	20	0.95	0.49	0.58
0.70	15	65	0	20	1.00	0.19	0.35
0.80	0	0	80	20	0	0	0.20
0.90	0	0	80	20	0	0	0.20
1.00	0	0	80	20	0	0	0.20

.

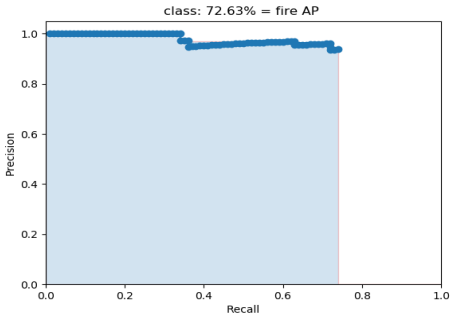


Gambar 9 Grafik perubahan nilai *threshold*

Pada Gambar 9, terlihat grafik perubahan nilai *precision, recall,* dan *accuracy* pada tiap nilai *threshold* yang digunakan berdasarkan pada Tabel 3. Dapat terlihat bahwa pada hasil deteksi menggunakan nilai *threshold* 0.30 adalah yang paling optimal, karena *precision* mendapatkan skor sebesar 0.98, *recall* mendapat skor sebesar 0.95 dan *accuracy* mendapat skor 0.95. Namun, seiring dengan bertambahnya nilai *threshold* yang digunakan, skor *precision* dan skor *accuracy* semakin turun, lain halnya dengan *recall* yang justru cenderung meningkat. Pada nilai *threshold* 0.80, didapati bahwa prediksi tidak ada yang mencapai nilai *confidence* melebihi 0.80.

3.3 Evaluasi mAP (*Mean Average Precision)*

Evaluasi mAP atau *mean average precision* dilakukan dengan melihat hasil deteksi yang menggunakan data uji berupa citra dengan jumlah sebanyak 100 citra pada nilai *threshold* 0.30. Maka, didapatkan skor mAP sebesar 72.63%.



Gambar 10 Grafik mAP pada *threshold* 0.30

4. KESIMPULAN

Berdasarkan hasil penelitian, dapat diambil beberapa kesimpulan di antaranya :

1. Algoritma YOLO dapat dengan baik mendeteksi objek api pada video dengan cukup baik dengan mendapat nilai *confidence* rata-rata 0.66, dengan nilai terendah pada 0.55 dan nilai tertinggi pada 0.71
2. Dengan menggunakan nilai *threshold* 0.30, didapatkan hasil deteksi dengan skor *precision* sebesar 0.98, *recall* sebesar 0.95 dan *accuracy* sebesar 0.95. Serta, menghsilkan skor mAP sebesar 72.63%

5. DAFTAR RUJUKAN

[1] Kusban, M. (2012). PERBAIKAN CITRA MELALUI PROSES PENGOLAHAN PIKSEL. *Prosiding Seminar Nasional Aplikasi Sains & Teknologi (SNAST) Periode III ISSN: 1979-911X*, (pp. 98-105). Yogyakarta.

[2] Lan, W., Dang, J., Wang, Y., & Wang, S. (2018). Pedestrian Detection Based on YOLO Network Model. *Proceedings of 2018 IEEE International Conference on Mechatronics and Automation*, 1547-1551.

[3] Nurliadi, Sihombing, P., & Ramli, M. (2016). Analisis Contrast Stretching Menggunakan Algoritma Euclidean Untuk Meningkatkan Kontras Pada Citra Berwarna. *Jurnal Teknovasi Volume 03, Nomor 1, ISSN : 2355-701X*, 26-38.

[4] Pritam, D., & Dewan, J. H. (2017). Detection Of Fire Using Image Processing Techniques With LUV Color Space. *2017 2nd International Conference for Convergence in Technology (I2CT) IEEE*, 1158-1162.

[5] Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. *arXiv:1612.08242v1*, 1-9.

[6] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 779-788.

[7] Shen, D., Chen, X., Nguyen, M., & Yan, W. Q. (2018). Flame Detection Using Deep Learning. *2018 4th International Conference on Control, Automation and Robotics*, 416-420.

[8] Torre, J. d., Valls, A., & Puig, D. (2017). A Deep Learning Interpretable Classifier for Diabetic. *arXiv:1712.08107v1* , 1-28.

[9] Utomo, B. (2017). Drone Untuk Percepatan Pemetaan Bidang Tanah. *Media Komunikasi Geografi, Vol 18, No. 2 Jurusan Pendidikan Geografi, Universitas PGRI Palembang*, 146-155.

[10] Wakhidah, N. (2011). Perbaikan Kualitas Citra Menggunakan Metode Contrast Stretching. *JURNAL TRANSFORMATIKA, Volume 8, No.2 Januari 2011*, 78-83.