

Miftahul Huda

hudamifta52@gmail.com

## **Implementasi *Optical Mark Recognition* (OMR) menggunakan YOLOv8 untuk Otomatisasi Pembacaan dan Penghitungan Surat Suara dalam Pemilihan Umum**

### **I. Latar Belakang Metode**

Pemilihan Umum atau yang lebih dikenal dengan singkatan sebagai Pemilu, dimulai pada tahun 1955 sebagai salah satu pilar penting dalam sistem demokrasi. Sejak saat itu, penyelenggaraan Pemilu di Indonesia telah melalui berbagai perkembangan dan penyempurnaan untuk menjamin terselenggaranya pemilu yang adil, jujur dan bermartabat. Namun, sampai saat ini masih terdapat tantangan dalam proses perhitungan suara untuk memastikan efisiensi dan transparansi hasil pemilu sehingga tercapai integritas dan legitimasi hasil pemilu serta menjaga kepercayaan publik terhadap proses demokrasi. Proses penghitungan suara yang masih dilakukan secara manual seringkali rentan terhadap kesalahan manusia, memakan waktu yang lama, dan berpotensi mengurangi kepercayaan masyarakat terhadap penyelenggaraan pemilu.

Salah satu tantangan utamanya yaitu bagaimana membaca data hasil surat suara pemilu dari gambar formulir pemungutan suara secara akurat dan efisien. Formulir hasil pemungutan suara seringkali ditulis tangan dan difoto dengan kondisi serta kualitas yang bervariasi, sehingga membuat proses pengenalan karakter menjadi tantangan tersendiri. Oleh sebab itu, diperlukan metode yang dapat mendeteksi dan mengekstraksi data secara otomatis dari formulir tersebut untuk meningkatkan efisiensi dan akurasi pembacaan data hasil surat suara. Untuk mengatasi tantangan ini, perkembangan teknologi informasi yang semakin pesat dalam bidang *Computer Vision* dapat menjadi solusi yang tepat untuk mengotomatisasi permasalahan tersebut. Terdapat beberapa metode yang dapat digunakan seperti *Optical Mark Recognition* (OMR) dan *Optical Character Recognition* (OCR).

Pemilihan metode OMR pada penelitian ini didasarkan pada gambar formulir kertas suara yang mengandung tanda (*mark*) berupa bulatan konsisten yang merepresentasikan nilai dari 0 sampai 9. *Optical Mark Recognition* (OMR) sendiri merupakan metode untuk melakukan proses ekstraksi informasi dengan mengenali tanda-tanda pada sebuah gambar atau dokumen (Kumar et al., n.d.). Keunggulan utama dari OMR adalah kemampuannya dalam mengenali tanda khusus seperti bulatan pada area tertentu dari formulir dengan akurasi yang tinggi. Untuk memaksimalkan proses ekstraksi OMR diintegrasikan dengan YOLOv8. Sistem ini dapat mendeteksi area yang relevan pada formulir kertas suara dan kemudian mengekstraksi informasi *mark* (titik) di dalamnya menggunakan OMR.

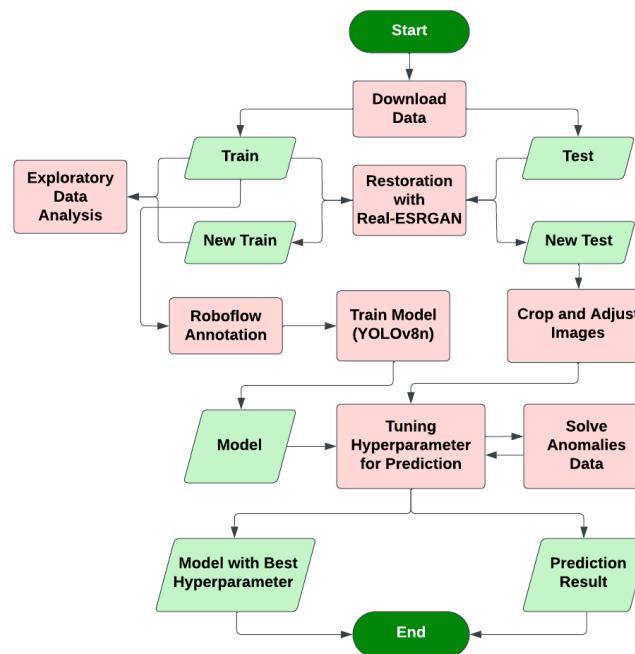
Penelitian ini mencoba mempraktekkan pendekatan yang sudah terbukti keberhasilannya dengan mengimplementasikan OMR menggunakan YOLOv8 dalam pembacaan dan perhitungan surat suara pemilihan umum. Penelitian ini memberikan hasil yang lebih akurat dan efisien, sehingga berperan penting dalam mendukung upaya peningkatan akurasi, efisiensi dan transparansi perhitungan surat suara dalam Pemilu.

## II. Metodologi

### 2.1 Data

Dataset yang digunakan dalam kompetisi ini merupakan kumpulan gambar formulir hasil penghitungan suara yang digunakan dalam pemilihan umum yang disebut Surat C1 Hasil (Sertifikat hasil dan rincian perhitungan perolehan suara di tempat perhitungan) yang menandakan surat sah dari setiap TPS yang tersebar di daerah-daerah. Setiap gambar menampilkan formulir yang terdiri dari angka-angka dan terbilang di *box* perhitungan suara paslon tertentu yang mewakili jumlah suara sah untuk setiap kandidat. Data terbagi menjadi data latih (*Data Train*) dan data uji (*Data Test*) dengan masing-masing berjumlah 500 gambar data latih dan 200 gambar data uji. Data latih digunakan untuk melatih model agar dapat menghasilkan prediksi yang akurat. Model yang telah dilatih ini kemudian digunakan untuk melakukan prediksi menggunakan data uji.

## 2.2 Prosedur Analisis



**Gambar I. Flowchart Prosedur Analisis**

Prosedur analisis dimulai dengan mendownload data yang ada didalam website *kaggle*, dengan pembuatan API yang terhubung pada saat pengambilan data dengan *google colab*, kemudian dilanjutkan mengekstrak file yang telah diunduh dan mengidentifikasi data yang ber-folder “Train” dan “Test”. Kemudian dilanjutkan dengan *preprocessing* gambar menggunakan Algoritma *Real-Enhance Super Resolution Generative Adversarial Network* (Real-ESRGAN). Setelah proses restorasi gambar dengan menggunakan Real-ESRGAN didapatkan data baru hasil perbaikan resolusi, proses selanjutnya dilakukan *Exploratory Data Analysis* (EDA). Setelah itu dilakukan proses anotasi bagian tertentu di dalam gambar menggunakan *web application* yang bernama “**Roboflow**” dengan memberikan label tertentu didalam gambar. Kemudian dilanjutkan dengan melatih data yang sudah dilakukan anotasi dengan

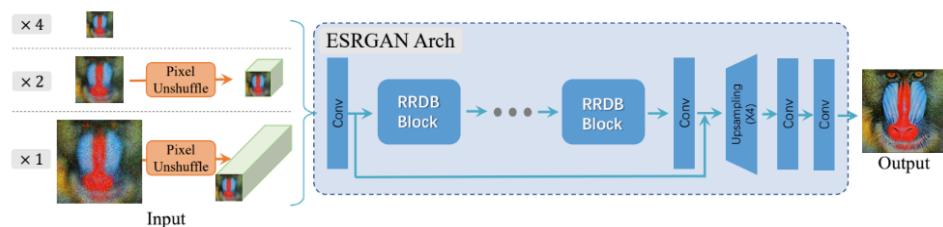
menggunakan model *pre-trained* dari YOLO versi 8. Setelah menjadi sebuah model dilakukan kembali *hyperparameter tuning* dengan tujuan mendapatkan kombinasi yang sempurna dari *hyperparameter* YOLO, proses ini dilakukan secara repetitif dan diiringi dengan memperbaiki anomali-anomali data yang menyebabkan kegagalan dalam deteksi. Setelah mendapatkan kombinasi *hyperparameter* yang sesuai dan tepat dari segi evaluasi model dan hasil prediksi, kemudian dilanjutkan dengan pengetesan dari model yang dibuat dan proses selesai.

### 2.2.1 Exploratory Data Analysis (EDA)

EDA merupakan proses mengeksplorasi data dengan tujuan untuk memahami dan memberikan pengetahuan (*insight*) tentang data (Wahyuni et al., 2019). Pada proses ini dilakukan eksplorasi tentang distribusi dari ukuran gambar ( $width \times height$ ) dan perbandingan rata-rata *width* dan *height* lalu menampilkan beberapa gambar sebelum dan sesudah direstorasi dengan *Real-Enhance Super Resolution Generative Adversarial Network* (Real-ESRGAN).

### 2.2.2 Restorasi dengan Real-ESRGAN

Real-ESRGAN bertujuan untuk meningkatkan kualitas dengan restorasi gambar dan video dengan mengembangkan kemampuan ESRGAN dalam melakukan restorasi. Real-ESRGAN dilatih menggunakan data sintetis murni (Xin Tao, 2024).



Gambar II. Real-ESRGAN dengan Arsitektur Generator ESRGAN

Real-ESRGAN menggunakan model generator yang sama dengan ESRGAN. Untuk input gambar skala  $\times 1$  dan  $\times 2$  diperkecil dengan operasi penguraian piksel untuk dimasukkan ke dimensi input yang dapat

mengurangi memori GPU dan sumber daya komputasi. Karena Real-ESRGAN bertujuan untuk mengatasi ruang degradasi yang jauh lebih besar daripada ESRGAN, maka Real-ESRGAN membutuhkan model diskriminasi yang lebih kuat daripada ESRGAN.



**Gambar III. Arsitektur Deskriminator U-Net pada Real-ESRGAN**

U-Net dapat menghasilkan piksel yang lebih realistik dan dapat memberikan umpan balik per-piksel yang terperinci ke generator (Wang et al., 2021). Pada penelitian ini menggunakan Real-ESRGAN agar dapat menghasilkan gambar dengan kualitas yang lebih baik dan lebih realitas.

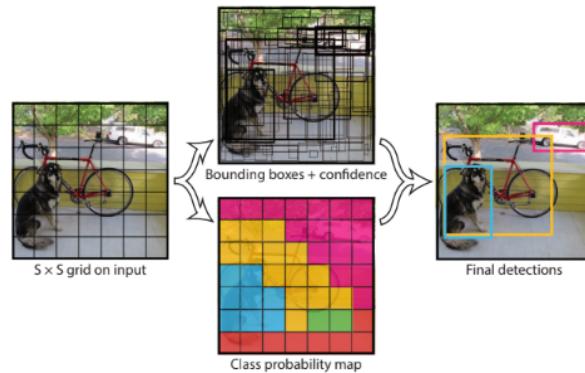
### 2.2.3 Anotasi Roboflow

Pada tahap ini gambar di anotasi menggunakan *tools* yang berbasis web aplikasi “**Roboflow**” *tools* ini merupakan project *open source* yang dapat digunakan tanpa hambatan. Pemrosesan gambar dengan anotasi diawali dengan memberikan *bounding box* (kotak pembatas) pada bagian yang mengandung *mark* (titik) dan memberikan label *class* yang sesuai dengan nama objek atau jenis objek yang ada. Kemudian data yang sudah di tandai dengan label akan disimpan di dalam sebuah dataset yang dapat digunakan dengan diintegrasikan dengan *library* python yang bernama “**Roboflow**”

### 2.2.4 Train Model You Only Look One (YOLO)

*You Only Look Once* (YOLO) adalah algoritma *object detection* “**one-step**”. Dalam YOLO, pendekripsi objek dianggap sebagai masalah regresi tunggal untuk menghasilkan prediksi langsung dari gambar ke koordinat kotak pembatas (*bounding box*) dan probabilitas kelas, yang berarti “**kamu hanya melihat satu kali**” (YOLO) pada sebuah gambar

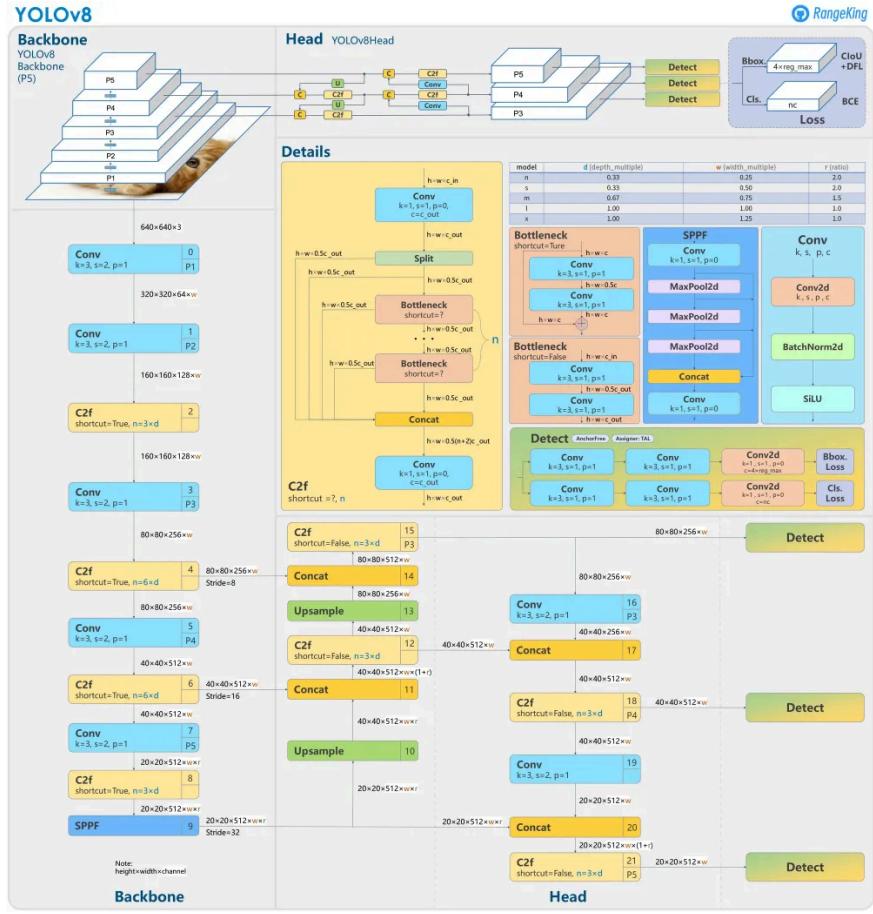
untuk memprediksi objek apa yang ada dan di mana lokasinya (Redmon et al., n.d.).



**Gambar V. YOLO Memprediksi *Bounding Box***

Pada penelitian ini kami menggunakan YOLO versi 8 yang merupakan versi terbaru sebelum versi 9 (YOLOv9) yang lebih banyak memakai memori GPU dan sumber daya komputasi. YOLOv8 memberikan kemampuan deteksi lebih cepat daripada pendahulunya (versi sebelumnya) yang sudah teruji [Ultralytics Documentation](#).

#### 2.2.4.1 Arsitektur



Gambar VI. Arsitektur YOLOv8

YOLOv8 mempunyai arsitektur yang kompleks yang terdiri dari *backbone* untuk ekstrasi fitur dan *head* untuk deteksi objek. YOLOv8 menggunakan *anchor free detection* dan SPPF untuk memprediksi langsung *class probability* dan koordinat *bounding box*. *Stem* awal menggunakan konvolusi  $6 \times 6$  diganti dengan konvolusi  $3 \times 3$ , lalu blok utama diubah dan *C3* digantikan *C2f*. Pada *C2f* seluruh output dari *Bottleneck* digabungkan dan pada *C3* hanya output dari *Bottleneck* terakhir yang digunakan. *Bottleneck* mirip dengan YOLOv5, tetapi ukuran kernel konvolusi pertama diubah dari  $1 \times 1$  menjadi  $3 \times 3$ . Pada bagian *neck* menggabungkan fitur-fitur dari berbagai lapisan secara langsung tanpa mengubah dimensi salurannya, yang dapat mengurangi jumlah parameter dalam jaringan dan ukuran tensor.

#### 2.2.4.2 Fungsi Kerugian dan Propagasi Balik

YOLOv8 menggunakan fungsi kerugian yang termasuk *box loss*, *class loss*, dan *dfl loss*. Fungsi kerugian mempertimbangkan berbagai parameter seperti *IoU*, lebar dan tinggi *bounding box*, titik sentral, dan label kelas. Aturan pembaruan bobot yang digunakan dalam *backpropagation* melibatkan gradien dari fungsi kerugian dan memperbarui bobot jaringan dengan menggunakan momentum dan laju pembelajaran (*Learning Rate*). Fungsi kerugian YOLOv8 didefinisikan sebagai:

$$\begin{aligned}
L = & \frac{\lambda_{box}}{N_{pos}} \sum_{x,y} P_{c^*_{x,y}} [1 - q_{x,y} + \frac{\|b_{x,y} - \hat{b}_{x,y}\|_2^2}{\rho^2} + \alpha_{x,y} v_{x,y}] \\
& + \frac{\lambda_{cls}}{N_{pos}} \sum_{x,y} \sum_{c \in classes} y_c \log(\hat{y}_c) + (1 - y_c) \log(1 - \hat{y}_c) \\
& + \frac{\lambda_{dfl}}{N_{pos}} \sum_{x,y} P_{c^*_{x,y}} [- (q_{(x,y)+1} - q_{x,y}) \log(\hat{q}_{x,y}) \\
& + (q_{x,y} - q_{(x,y)-1}) \log(\hat{q}_{(x,y)+1})] \quad (1)
\end{aligned}$$

di mana:

$$\begin{aligned}
q_{x,y} &= IoU(\hat{\beta}_{x,y}, \beta_{x,y}) = \frac{\hat{\beta}_{x,y} \cap \beta_{x,y}}{\hat{\beta}_{x,y} \cup \beta_{x,y}} \\
v_{x,y} &= \frac{4}{\pi^2} (\arctan(\frac{w_{x,y}}{h_{x,y}}) - \arctan(\frac{\hat{w}_{x,y}}{\hat{h}_{x,y}}))^2 \\
\alpha_{x,y} &= \frac{v}{1-q_{x,y}} \\
\hat{y}_c &= \frac{1}{1+e^{-z_c}} \\
\hat{q}_{x,y} &= \frac{e^{z_{x,y}}}{\sum_c e^{z_c}}
\end{aligned}$$

dan:

$N_{pos}$ : Jumlah total sel yang berisi objek.

$P_{c^*_{x,y}}$ : Fungsi penanda untuk sel-sel yang memuat sebuah objek.

$\beta_{x,y}$ : *Ground truth bounding box*.

$\hat{\beta}_{x,y}$ : *Predicted box*.

$b_{x,y}$ : Titik sentral *ground truth bounding box*.

$\hat{b}_{x,y}$ : *Predicted b*<sub>x,y</sub>.

$y_c$ : Label asli dari kelas c.

$w_{x,y}$  dan  $h_{x,y}$ : Lebar dan tinggi *bounding box*.

$\hat{w}_{x,y}$  dan  $\hat{h}_{x,y}$ : *Predicted w*<sub>x,y</sub> dan *h*<sub>x,y</sub>

$\rho$ : Panjang diagonal dari kotak terkecil yang mencakup *Predicted box* dan *Ground truth bounding box*.

Untuk melakukan *backpropagation* (propagasi balik) pada YOLOv8 menggunakan ketentuan update parameter bobot yang didefinisikan sebagai:

$$L(\theta) = \frac{\lambda_{box}}{N_{pos}} L_{box}(\theta) + \frac{\lambda_{cls}}{N_{pos}} L_{cls}(\theta) + \frac{\lambda_{dfl}}{N_{pos}} L_{dfl}(\theta) + \phi \|\theta\|_2^2 \quad (2)$$

$$\theta^t = \theta^{t-1} - \eta (\beta V^{t-1} + \nabla_{\theta} L(\theta^{t-1})) \quad (3)$$

Di mana  $L(\theta)$  adalah fungsi kerugian dengan parameter bobot  $\theta$  dan  $\theta^t$  adalah bobot baru yang diperbarui berdasarkan ketentuan update parameter bobot dengan laju pembelajaran (*learning rate*)  $\eta$  dan perubahan kecepatan  $V$  dengan momentum  $\beta$ .

### 2.2.5 Pemotongan dan Penyesuaian Gambar pada Data Test

Untuk mendapatkan informasi suara dari ketiga paslon dilakukan *filter area* yang berfokus mendekripsi 30% bagian kanan *width* (*width* × 0.7 sampai dengan *width* – 1) dan mengabaikan 5% *height* pada bagian atas (*height* × 0.05 sampai dengan *height* – 1) hal ini bertujuan untuk memfokuskan bagian yang akan didekripsi oleh model dan mengantisipasi adanya objek yang tidak diinginkan pada bagian lain yang terdeteksi.

Setelah *filter area* dan mendapatkan matriks  $M$  dengan dimensi  $((0.3 \times width) - 1) \times ((0.95 \times height) - 1)$  lalu selanjutnya mendapatkan informasi suara masing-masing paslon dengan membagi menjadi 3 *region*,  $R_1$  untuk paslon nomor urut 01,  $R_2$  untuk paslon nomor urut 02, dan  $R_3$  untuk paslon nomor urut 03 dengan:

$$R_1 = M[1:k], \quad k = \text{round}\left(\frac{(0.95 \times height) - 1}{3} + 20\right) \quad (4)$$

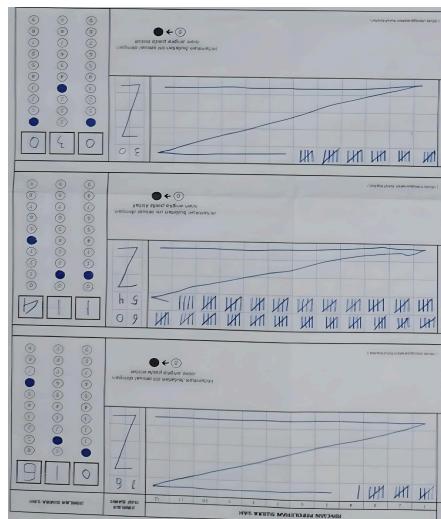
$$R_2 = M[k + 1:((0.95 \times height) - 1) - k] \quad (5)$$

$$R_3 = M[((0.95 \times height) - 1) - k + 1:((0.95 \times height) - 1)] \quad (6)$$

Dengan begini dapat memilih informasi suara paslon berdasarkan *region* tersebut.

#### 2.2.6 Penanganan Anomali Data pada Data Test

Anomali data adalah data yang menyimpang dari suatu *dataset* (Foorthuis, 2021). Pada *dataset* yang digunakan terdapat beberapa data yang menyimpang seperti data yang terbalik dan *mark* (titik) yang tidak ada, yang mana hal ini akan mempengaruhi algoritma dan model dalam mendapatkan informasi suara dari ketiga paslon.



**Gambar VII. Gambar Terbalik**

**Gambar VIII. Gambar Tanpa Mark**

Untuk memperbaiki gambar yang tidak ada *mark* (titik) diasumsikan model memprediksi dengan class 0 (sesuai label yang diberikan pada *data*

*train*) dan menambahkan 1. Sedangkan untuk memperbaiki gambar yang terbalik dengan cara memutar gambar  $180^\circ$  ( $180$  derajat) yang dapat dilakukan dengan:

$$x_{new} = x \times \cos(180^\circ) + y \times \sin(180^\circ) \quad (7)$$

$$y_{new} = y \times \cos(180^\circ) + x \times \sin(180^\circ) \quad (8)$$

### 2.2.7 Tuning Hyperparameter

Penggunaan hyperparameter default dari YOLO memungkinkan hasil yang kurang optimal seperti prediksi kotak pembatas yang tumpang tindih bersamaan dengan adanya objek yang salah terdeteksi, pada *eksperimen* ini dilakukan *hyperparameter tuning* untuk mencari parameter dalam model *pre-trained* YOLO yang paling optimal, parameter yang di-tuning antara lain *Confidence Score Threshold*, *Intersection Over Union (IoU) Threshold*, augmentasi waktu pengujian, dan *agnostic Non-Maximum Suppression (NMS)*.

#### 2.2.7.1 Confidence Score Threshold

*Confidence Score Threshold* digunakan untuk memberikan ambang batas pada kelas prediksi, jika terdapat kelas prediksi yang terdeteksi dengan *Confidence Score* ( $C_i$ ) dan  $C_i \leq C$  dimana  $C$  adalah *Confidence Score Threshold* yang diterapkan maka kelas  $i$  dianggap tidak valid. Hal ini berguna untuk mengatasi *bounding box* multi kelas yang *overlap*.

#### 2.2.7.2 Intersection Over Union (IoU) Threshold

Salah satu permasalahan dalam *object detection* adalah prediksi *bounding box* yang tumpang tindih dimana hanya terdapat 1 objek yang memiliki 2 *bounding box* sekaligus, hyperparameter *IoU Threshold* dapat mengatasi permasalahan tersebut dengan menganggap jika *IoU* kedua kotak pembatas yang terprediksi

$$IoU(\hat{\beta}_{pi}, \hat{\beta}_{pj}) = \frac{\hat{\beta}_{pi} \cap \hat{\beta}_{pj}}{\hat{\beta}_{pi} \cup \hat{\beta}_{pj}}$$

lebih besar dari *IoU Threshold* maka

dianggap *overlap* atau tumpang tindih. Misal  $\hat{\beta}_{pi}$  adalah prediksi *bounding box* objek ke  $i$  dan  $\hat{\beta}_{pj}$  adalah prediksi *bounding box* objek ke  $j$  maka  $\hat{\beta}_{pi}$  dan  $\hat{\beta}_{pj}$  dianggap *overlap* jika  $IoU(\hat{\beta}_{pi}, \hat{\beta}_{pj}) > IoU Threshold$ .

### 2.2.7.3 Augmentasi Waktu Pengujian

Saat *hyperparameter* Augmentasi waktu pengujian bernilai *True* akan Mengaktifkan augmentasi waktu pengujian untuk prediksi, yang memungkinkan meningkatkan presisi deteksi tetapi memperlambat proses komputasi. Dalam solusi permasalahan ini mengabaikan kecepatan komputasi karena presisi dan akurasi yang tinggi lebih penting.

### 2.2.7.4 Agnostic Non-Maximum Suppression (NMS)

Saat *hyperparameter* *Agnostic Non-Maximum Suppression* (NMS) bernilai *True* akan mengaktifkan *class-agnostic Non-Maximum Suppression* (NMS). Ini berarti akan menggabungkan kotak-kotak yang tumpang tindih dari berbagai kelas ketika objek dari kelas yang berbeda saling tumpang tindih (berdasarkan *Intersection Over Union* (*IoU*)).

---

#### Algorithm 1 Non-Max Suppression

---

```

1: procedure NMS( $B, c$ )
2:    $B_{nms} \leftarrow \emptyset$  Initialize empty set
3:   for  $b_i \in B$  do  $\Rightarrow$  Iterate over all the boxes
4:      $discard \leftarrow \text{False}$  Take boolean variable and set it as false. This variable indicates whether } b(i) \text{ should be kept or discarded}
5:     for  $b_j \in B$  do  $\Rightarrow$  Start another loop to compare with } b(i)
6:       if  $same(b_i, b_j) > \lambda_{nms}$  then  $\Rightarrow$  If both boxes having same IOU
7:         if  $score(c, b_j) > score(c, b_i)$  then
8:            $discard \leftarrow \text{True}$   $\Rightarrow$  Compare the scores. If score of } b(i) \text{ is less than that of } b(j), b(i) \text{ should be discarded, so set the flag to True.}
9:         if not  $discard$  then  $\Rightarrow$  Once } b(i) \text{ is compared with all other boxes and still the discarded flag is False, then } b(i) \text{ should be considered. So add it to the final list.}
10:         $B_{nms} \leftarrow B_{nms} \cup b_i$   $\Rightarrow$  Do the same procedure for remaining boxes and return the final list
11:      return  $B_{nms}$ 
```

---

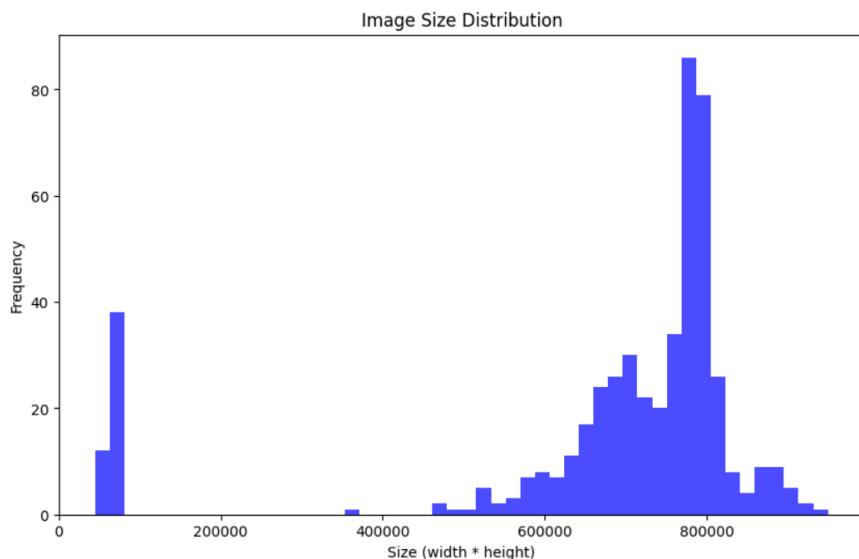
Gambar VI. Algoritma NMS

Hal ini berguna ketika ada banyak objek yang saling tumpang tindih dan berasal dari kelas yang berbeda.

### III. Hasil dan Pembahasan

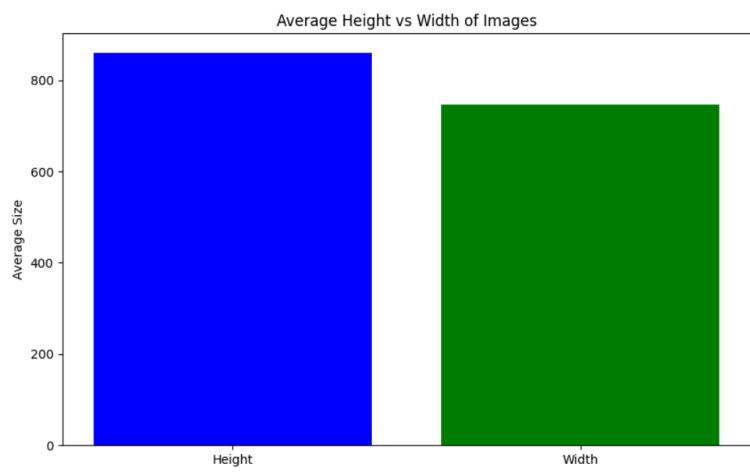
Merancang algoritma atau metode pemodelan komputasi yang efektif menjadi langkah penting untuk melakukan otomatisasi pembacaan dan perhitungan surat suara dalam pemilihan umum. Melibatkan teknologi *Computer Vision* menjadi dorongan untuk mencapai hasil prediksi yang akurat dan efisien. Penggunaan metode *Optical Mark Recognition* (OMR) menggunakan YOLOv8 memungkinkan untuk melakukan pembacaan data pada kertas suara untuk mengurangi *human error*, mempercepat perhitungan, dan meningkatkan kepercayaan masyarakat.

Dalam penyelesaian permasalahan ini, pertama-tama diperlukan pemahaman yang mendasar mengenai data-data dalam bentuk gambar dari segi ekstraksi dan juga pengolahan datanya. Proses pertama dilakukan pengecekan terhadap seluruh data gambar yang sudah diunduh sebelumnya yang terdiri dari folder *data train* dan *data test* dan selanjutnya dilakukan pengecekan dari segi ukuran dan resolusi dari masing-masing gambar yang berada didalam folder *data train* dan *data test* hasil menunjukkan,



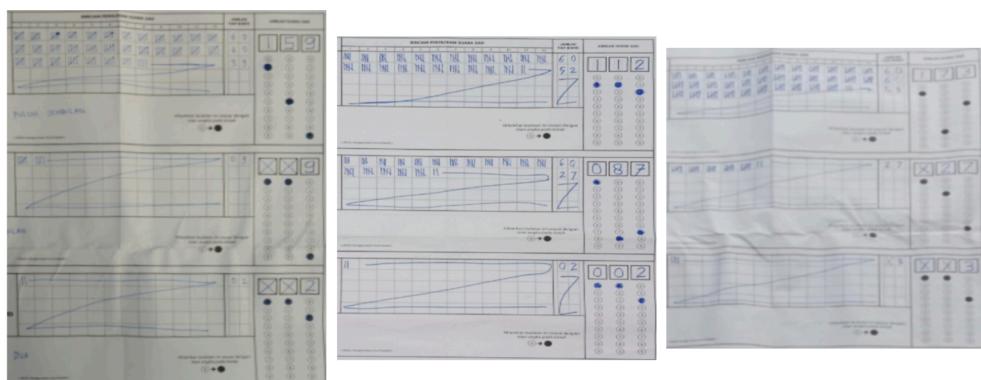
**Gambar VII. Hasil Distribusi Ukuran Gambar**

Resolusi pada gambar banyak terdistribusi diantara 600000 piksel hingga 800000 piksel dari hasil yang didapatkan gambar cukup banyak terdistribusi dalam resolusi yang tinggi dari hasil visualisasi dalam *barchart* (diagram batang). Namun ada beberapa gambar yang berdistribusi kurang dari 200000 piksel dengan rentang frekuensi pada munculnya gambar diatas 40 buah. Untuk itu, diperlukan pengukuran lebih lanjut dengan melihat hasil rataan dari tinggi dan lebar dari keseluruhan gambar untuk mengecek tinggi dan lebar dari gambar mempengaruhi secara signifikan untuk gambar yang lainnya dan analisis selanjutnya.



**Gambar VIII. Rataan Resolusi pada Seluruh Gambar**

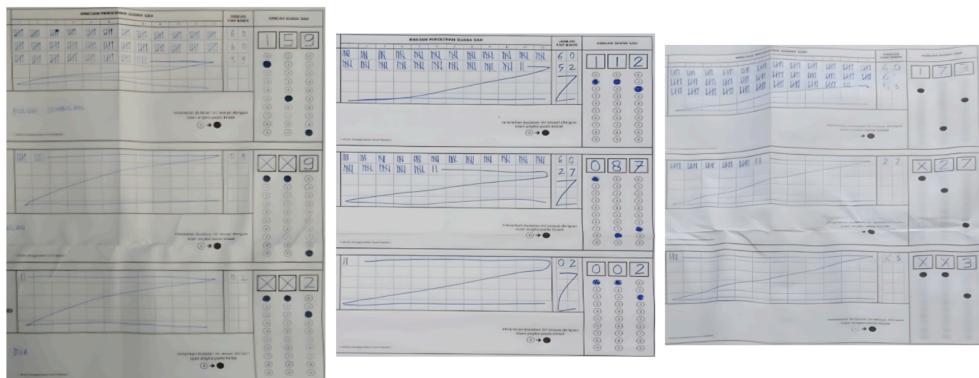
Hasil dalam pengukuran yang dari rataan tinggi dan lebar (resolusi) dari masing-masing gambar menunjukkan hasil yang tidak signifikan dan cukup ditoleransi untuk dilakukan preprocessing gambar lainnya. Proses dilanjutkan di dalam tahapan kedua dari pre-processing gambar dengan mengecek gambar-gambar yang direpresentasikan *blur* (kabur).



**Gambar IX. Gambar dalam data train dengan resolusi rendah**

Didalam proses pengecekan didapatkan hasil yang cukup banyak gambar yang memerlukan peningkatan dari sisi resolusi gambar. *Restoration Image Algorithm* atau yang lebih dikenal sebagai proses perbaikan gambar dari segi resolusi dapat diimplementasikan dengan menggunakan metode *deep learning* (pembelajaran mendalam) untuk permasalahan ini digunakan Real-ESRGAN.

Data *train* dan data *test* keseluruhan akan dilakukan pengimplementasian dari algoritma ini, dengan tujuan pergeseran secara serentak dari segi resolusi menjadi lebih baik dari gambar semula, hasil yang didapatkan setelah pengimplementasian algoritma Real-ESRGAN sebagai berikut

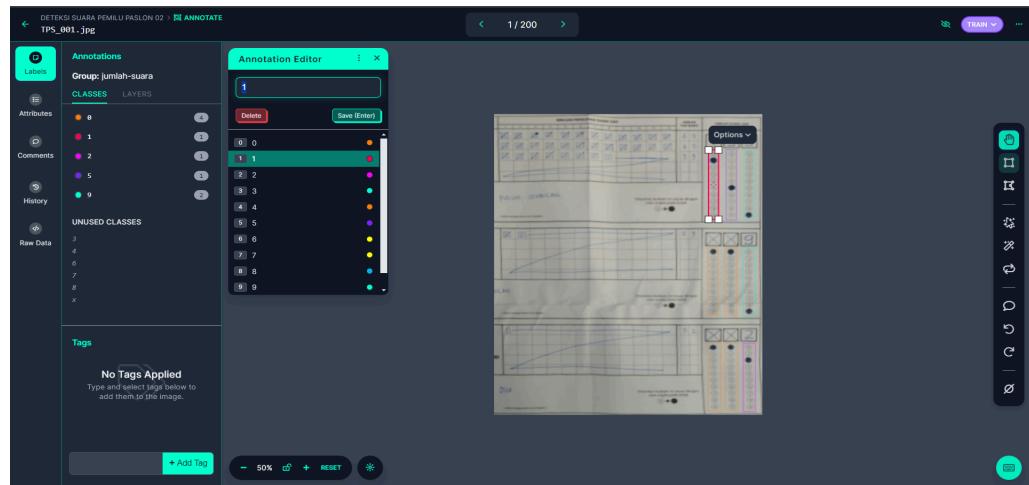


**Gambar X. Hasil peningkatan resolusi dengan Real-ESRGAN**

Hasil visualisasi gambar setelah diimplementasikan Real-ESRGAN tampak lebih tajam dibandingkan dengan sebelumnya, gambar lebih mudah dalam pembacaan mesin dan pada tahapan *preprocessing* ini cukup dalam penyelesaian *noise* (gangguan pada gambar) yang terjadi dalam permasalahan ini dan dapat dilanjutkan untuk tahapan selanjutnya.

Proses selanjutnya adalah tahapan *Annotation* (pemberian tanda), pada tahapan ini, dilakukan untuk membantu proses *Computer Vision* didalam mendapatkan sebuah *sight* (pandangan) dari data gambar yang ada, dengan bantuan anotasi juga data *Computer Vision* dapat menerka secara jelas dan membandingkan bidang-bidang yang ada dengan bantuan dari *bounding box* (kotak pembatas)

yang telah dibuat selama proses anotasi. Hal yang perlu diperhatikan didalam pembuatan anotasi dengan mengukur secara akurat *bounding box* (kotak pembatas) antara bidang yang ingin ditandai. Aturan dalam pembuatan *bounding box* tidak memotong dan juga tidak berlebihan didalam pengambilan bidang didalam dan di sekeliling *bounding box*.



Gambar X. Anotasi dengan Roboflow

Stage selanjutnya merupakan proses *training* model YOLO v8, untuk melakukan *training* kami menggunakan kode sumber dari repository *ultralytics*. Proses *training* menggunakan  $500 \times 3$  (Augmentasi) data *train* dengan  $100 + 10$  *epochs* dan input image  $640 \times 640$  piksel. Evaluasi proses *training* sepenuhnya dilakukan terhadap data *train* karena kami tidak menggunakan data validasi agar tidak kehilangan jumlah data dan karena kami pun dapat mengevaluasi manual menggunakan data *test*. Hasil proses *training* dievaluasi menggunakan *metrics-metrics* deteksi objek yang umum digunakan seperti Presisi ( $TP/(TP + FP)$ ), Recall ( $TP/(TP + FN)$ ), dan mAP (*Mean Average Precision*) pada nilai *IoU* tertentu. Pada eksperimen ini hasil evaluasi data *train* memiliki performa yang baik di semua kelas, dengan skor *Precision*, *Recall*, dan *mAP* yang tinggi. Nilai *Precision* (presisi) dan *Recall* untuk keseluruhan (semua kelas) adalah 0.997 menunjukkan tingkat akurasi yang tinggi dalam mendeteksi instansi (banyak kelas dalam dataset) kelas-kelas tersebut. Sedangkan untuk nilai mAP 50 dan mAP 50-95 juga tinggi yang masing-masing menyentuh skor 0.994 dan 0.841 menunjukkan model mencapai nilai rata-rata *Precision* yang tinggi

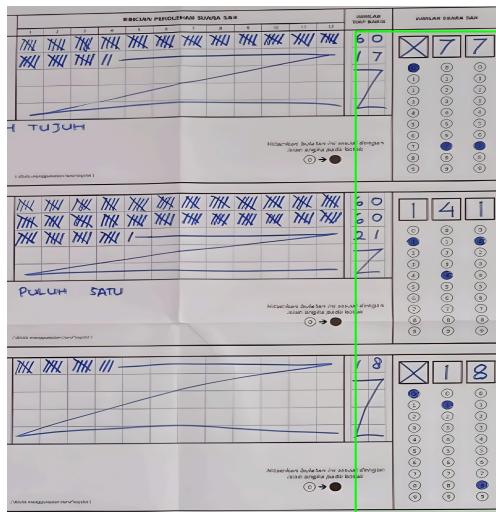
pada tingkat *Recall* tertentu dengan nilai  $IoU = 50$  dan  $IoU = 50 - 95$ . Berikut lengkapnya hasil evaluasi menggunakan data *train*.

Class	Images	Instances	Precision	Recall	mAP 50	mAP 50-95
all	1500	13500	0.997	0.997	0.994	0.841
0	1500	3885	0.993	0.998	0.99	0.842
1	1500	2349	0.997	0.996	0.994	0.851
2	1500	1095	0.993	0.995	0.993	0.838
3	1500	861	0.999	1	0.995	0.832
4	1500	813	0.995	0.996	0.995	0.845
5	1500	777	1	0.998	0.995	0.843
6	1500	780	0.992	0.996	0.995	0.842
7	1500	852	1	0.994	0.995	0.835
8	1500	771	1	1	0.995	0.846
9	1500	780	0.995	1	0.994	0.845
x	1500	537	1	1	0.995	0.831

**Tabel 1. Hasil Evaluasi Model Menggunakan Data Train**

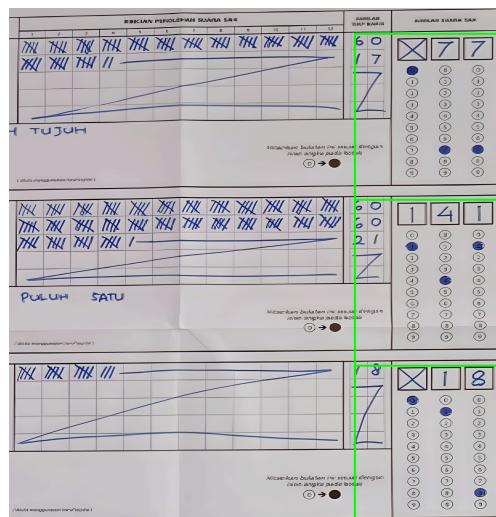
Walaupun hasil ini baik tetapi belum dapat dipastikan model tersebut memiliki performa yang akurat sebab hal ini wajar karena evaluasi tersebut menggunakan data *train* atau data yang sudah “terlihat” model untuk proses *training* dan harus diuji keakuratannya pada data yang belum pernah “terlihat” oleh model.

Untuk melakukan prediksi suara menggunakan data *test* gambar terlebih dahulu disesuaikan dengan melakukan *filter area* yang hanya memfokuskan bagian kanan (30% *width*) untuk menghindari adanya deteksi yang tidak diinginkan.



#### **Gambar X. Filter Area pada Gambar**

Selanjutnya adalah membagi menjadi 3 region untuk mendapatkan jumlah suara masing masing paslon menggunakan persamaan (4), (5), (6).



## Gambar X. Pembagian 3 Region

Dengan begini kami dapat mengambil informasi yang terdeteksi dari masing-masing paslon.

Setelah melakukan penyesuaian untuk memprediksi suara dari ketiga paslon selanjutnya adalah memprediksi menggunakan semua data *test*, kami melakukan evaluasi manual terhadap data *test* dan ternyata masih ada beberapa kesalahan prediksi yang disebabkan oleh data yang menyimpang (anomali) antara lain seperti terbalik yang diatasi dengan persamaan (7) dan (8) lalu data yang tidak

ada mark yang diatasi dengan mengasumsikan model memprediksi kelas tersebut 0 (sesuai label data train) dan menambahkan 1, tapi ternyata model belum sepenuhnya memprediksi dengan tepat untuk semua data *test* oleh karena itu dilakukan *hyperparameter tuning* untuk mencari parameter yang paling cocok untuk data *test*, dan didapat *hyperparameter* terbaik dengan kombinasi sebagai berikut:

*Confidence Score Threshold* = 0.4

*Intersection Over Union (IoU) Threshold* = 0.5

Augmentasi waktu pengujian = *True*

*Agnostic Non-Maximum Suppression (NMS)* = *True*

Nilai *Confidence Score Threshold* di angka 0.4 yang semula (default) 0.25 menganggap kelas yang valid dengan ketentuan lebih ketat yang nilai *confidence* nya harus  $\geq 0.4$ , lalu nilai *IoU Threshold* di angka 0.5 yang semula 0.75 menganggap kelas yang sama dengan bounding box yang tidak overlap dengan ketentuan lebih ketat juga yang nilai *IoU* nya harus  $\leq 0.5$ . Sedangkan untuk nilai Augmentasi waktu pengujian yang bernilai *True* dapat meningkatkan keakuratan prediksi dengan mengaktifkan augmentasi data, dan untuk *Agnostic Non-Maximum Suppression (NMS)* yang bernilai *True* berguna untuk mengatasi 2 kelas yang berbeda dengan bounding box yang overlap berdasarkan *IoU Threshold*. Dengan kombinasi *hyperparameter* tersebut hasil prediksi suara dari ketiga paslon sepenuhnya benar dengan nilai MAPE mencapai 0,00000 atau tidak ada prediksi yang salah.

#### IV. Kesimpulan

Berdasarkan analisis prediksi yang telah dilakukan, dapat disimpulkan bahwa metode *Optical Mark Recognition* (OMR) menggunakan YOLOv8 dapat digunakan untuk melakukan otomatisasi pembacaan dan penghitungan surat suara dalam pemilihan umum secara akurat dan efektif. Pada *processing* menggunakan Real-ESRGAN terbukti dapat meningkatkan resolusi gambar pada *data train* dan *data test* sehingga model lebih mudah untuk mengekstraksi data dari formulir kertas suara pemilu. Meski terdapat beberapa data anomali seperti mark yang terlewat atau gambar yang terbalik, pendekatan ini mampu

menangani situasi tersebut dengan asumsi tertentu dan teknik rotasi gambar. Setelah dilakukan optimasi parameter, diperoleh konfigurasi terbaik dengan nilai conf=4, iou=0.5, augment=True, dan agnostic\_nms=True. Model yang dihasilkan berhasil mencapai nilai *Mean Absolute Percentage Error* (MAPE) yang sangat baik, yaitu 0,00000. Ini menunjukkan bahwa metode ini dapat memberikan akurasi yang sangat tinggi dalam pembacaan data pada surat suara pemilihan umum.

Implementasi OMR menggunakan YOLOv8 dalam pembacaan data surat suara berpotensi untuk meningkatkan akurasi, efisiensi dan transparansi pada pemilihan umum. Dengan mengotomatisasi proses ini, resiko *human error* dapat diminimalisir, waktu menjadi lebih efisien, dan hasil perhitungan dapat diverifikasi dengan mudah. Transparansi data hasil pemilu ini juga dapat meningkatkan kepercayaan publik terhadap integritas pemilihan umum yang menjadi salah satu pilar penting sistem demokrasi di Indonesia.

## V. Daftar Pustaka

- Bodla, N., Singh, B., Chellappa, R., & Davis, L. S. (2017). *Soft-NMS -- Improving Object Detection With One Line of Code*. <http://arxiv.org/abs/1704.04503>
- Dona Marleny, F. (n.d.). *Pengolahan Citra Digital Menggunakan Python*. <https://www.researchgate.net/publication/358220979>
- Foorthuis, R. (2021). On the nature and types of anomalies: a review of deviations in data. In *International Journal of Data Science and Analytics* (Vol. 12, Issue 4, pp. 297–331). Springer Science and Business Media Deutschland GmbH. <https://doi.org/10.1007/s41060-021-00265-1>
- Kumar, H., Chauhan, H., & Mittal, S. (n.d.). *THINK INDIA JOURNAL Analysis of OMR Sheet Using Machine Learning Model*.
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (n.d.). *You Only Look Once: Unified, Real-Time Object Detection*. <http://pjreddie.com/yolo/>

Reis, D., Kupec, J., Hong, J., & Daoudi, A. (2023). *Real-Time Flying Object Detection with YOLOv8*. <http://arxiv.org/abs/2305.09972>

Song, J., Yi, H., Xu, W., Li, X., Li, B., & Liu, Y. (2023). ESRGAN-DP: Enhanced super-resolution generative adversarial network with adaptive dual perceptual loss. *Heliyon*, 9(4). <https://doi.org/10.1016/j.heliyon.2023.e15134>

Wahyuni, E. D., Arifyanti, A. A., & Kustyani, M. (2019). *Exploratory Data Analysis dalam Konteks Klasifikasi Data Mining*. 263–269. <http://journal.itny.ac.id/index.php/ReTII>

Wang, X., Xie, L., Dong, C., & Shan, Y. (2021). *Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data*. <http://arxiv.org/abs/2107.10833>

Roboflow. (2024). *Apa yang Baru di YOLOv8?* Retrieved from <https://blog.roboflow.com/whats-new-in-yolov8/> (Diakses pada 30 April 2024)

Tao, X. (2024). *Real-ESRGAN*. Retrieved from <https://github.com/xinntao/Real-ESRGAN> (Diakses pada 30 April 2024)

towardsdatascience. (2023). *Non-Maximum Suppression (NMS)*. Retrieved from <https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c> (Diakses pada 30 April 2024)

Ultralytics. (2024). *Ultralytics: Object Detection*. Retrieved from <https://docs.ultralytics.com/modes/predict/#inference-arguments> (Diakses pada 30 April 2024)

## VI. Lampiran

[Masterpiece Tim WekWek](#)