

# “*Shrimp-Detector App*” Aplikasi Website untuk Mendeteksi Citra Udang Menggunakan *Custom Model* YOLOv8, Onnxruntime-web API, dan React Js

Rina Noviana dan Muhamad Irfan Maulana

Magister Managemen Sistem Informasi, Universitas Gunadarma  
Jl. Margonda Raya No. 100, Depok 16424, Jawa Barat  
E-mail: rina\_n@staff.gunadarma.ac.id, muhamadirfan45513@gmail.com

## Abstrak

Industri akuakultur atau perikanan di Indonesia telah berkembang pesat dan merupakan sektor yang memiliki peran penting dalam perekonomian dan pasokan pangan nasional di Indonesia. Salah satu komoditas perikanan yang berkembang pesat di Indonesia adalah udang. Teknologi pendeteksian objek dapat membantu para petambak udang dalam mendeteksi udang dan menentukan jumlah udang yang terdeteksi. Dalam penelitian ini, akan dibangun aplikasi website “Shrimp-Detector App” yang dapat mendeteksi udang dan menghitung jumlah udang yang terdeteksi menggunakan YOLOv8, Onnxruntime-web API, dan React Js. Dataset yang digunakan adalah dataset dengan dua kelas, yaitu udang dan bukan\_udang dengan jumlah data citra sebanyak 6630. Proses anotasi citra dilakukan dengan menggunakan roboflow. Proses training dataset dilakukan dengan menggunakan Google Colaboratory, YOLOv8, dan hasil yang didapat disimpan di Google Drive. Pembuatan aplikasi website menggunakan framework javascript yaitu react js. Proses deployment menggunakan tools Netlify sebagai provider CI/CD deployment tools. Pelatihan dilakukan dengan total 300 epoch. Nilai rata-rata precision sebesar 92.57%, sedangkan nilai recall sebesar 88.2%. Akurasi model diperoleh nilai 93.2%. Hasil pengujian yang didapatkan memperoleh tingkat akurasi deteksi udang sebesar 90,9% yang menggambarkan bahwa aplikasi website yang dihasilkan cukup stabil dalam mendeteksi objek secara tepat.

**Kata kunci :** Deteksi citra, Aplikasi Website, YOLOv8, Onnxruntime-web API, React JS

## Pendahuluan

Salah satu komoditas akuakultur [1] yang berkembang sangat pesat di Indonesia saat ini adalah udang. Udang menjadi salah satu komoditas ekspor potensial yang ada di Indonesia [2]. Selain itu, udang juga banyak dibudidayakan oleh para peternak daerah di Indonesia. Berdasarkan data dari Kementerian Kelautan dan Perikanan (KKP) [3], pada tahun 2020, produktivitas budidaya udang di Indonesia berkisar antara 10 – 50 ton/hektar/siklus. Pada tahun 2020 produksi budidaya udang di Indonesia mencapai 911,2 ribu ton, udang juga memberikan kontribusi terhadap total volume ekspor hasil perikanan sebesar 18,95% [4].

Besarnya kontribusi volume ekspor udang tersebut tentunya dapat ditingkatkan dengan penggunaan teknologi informasi dalam bidang perikanan, khususnya penggunaan teknologi computer vision maupun machine learning. Perkembangan teknologi informasi yang sangat pesat menyebabkan

semakin banyaknya perkembangan teknologi di dunia computer vision maupun machine learning. Saat ini, penggunaan teknologi computer vision maupun machine learning digunakan secara masif hampir disetiap kehidupan manusia yang tentunya sangat membantu manusia dalam melakukan pekerjaan maupun kegiatan sehari-hari. Salah satu objek dari computer vision adalah object detection. Object detection merupakan salah satu penerapan dari teknologi deep learning yang merupakan turunan dari machine learning yang menggunakan Artificial Neural Network (ANN) [5], [6], [7]. Teknologi deteksi objek membantu manusia mendeteksi objek [8]. Teknologi ini juga dapat membantu peternak udang dalam mendeteksi udang [9] dan menghitung jumlah udang yang ada. Berbagai algoritma deep learning saat ini dapat digunakan untuk mendeteksi objek [10].

Salah satu yang paling banyak digunakan adalah YOLO (You Only Look Once) [11], [12]. YOLO dikatakan memiliki kecepatan dan akurasi yang pal-

ing cepat dibandingkan dengan algoritma pengamatan objek lainnya [13]. YOLO algorithm merupakan algoritma object detection yang mendeteksi objek secara real-time. YOLO merupakan algoritma object detection yang cukup sederhana karena merupakan single convolutional network yang secara simultan memprediksi multiple bounding box dan kemungkinan kelas untuk bounding box tersebut. YOLO merupakan pre-trained model deep learning. Model terbaru deteksi Objek berbasis YOLO dari Ultralytics, YOLOv8 menawarkan kinerja yang luar biasa. Model ini lebih cepat dan lebih akurat daripada versi sebelumnya, sehingga sangat cocok untuk algoritma deteksi objek [14], [15], [16]. YOLOv8 adalah keluarga terbaru model deteksi objek berbasis YOLO dari Ultralytics yang memberikan kinerja canggih. Memanfaatkan versi YOLO sebelumnya, model YOLOv8 lebih cepat dan lebih akurat sambil menyediakan kerangka kerja terpadu untuk model pelatihan untuk performa deteksi Objek, segmentasi Instance, dan klasifikasi gambar [15].

Penelitian sebelumnya, "Automatic Counting Shrimp Larvae Based You Only Look Once (YOLO)," yang dilakukan oleh Siska Armalivia, Zahir Zainuddin, Andani Achmad, dan Muh. Arief Wicaksono pada tahun 2021, menggunakan algoritma YOLOv3 untuk mengidentifikasi dan menghitung jumlah larva udang yang terdeteksi. Tetapi versi YOLO yang digunakan adalah versi YOLO lama [17]. Keluarga terbaru model Deteksi Objek berbasis YOLO dari Ultralytics, YOLOv8 [15], menawarkan kinerja yang lebih canggih. Model ini lebih cepat dan lebih akurat daripada versi sebelumnya, sehingga sangat cocok untuk digunakan sebagai algoritma deteksi objek [16].

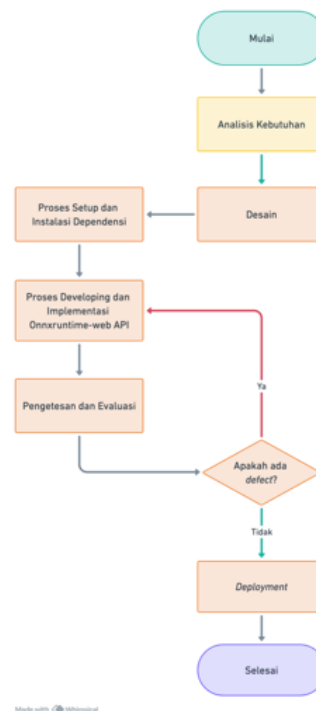
Penelitian ini juga merupakan penelitian lanjutan dari penelitian yang telah dilakukan oleh Muhamad Irfan Maulana dan Rina Noviana pada tahun 2023 yang berjudul "Training Custom Model Deteksi Udang menggunakan YOLOv8" [18]. Pada penelitian tersebut, menghasilkan custom model deteksi udang yang dibangun menggunakan YOLOv8. Dataset yang digunakan adalah dataset sebanyak dua kelas, yaitu udang dan bukan\_udang dengan jumlah data citra sebanyak 6630 data. Proses anotasi citra dilakukan dengan menggunakan roboflow. Proses training dataset dilakukan dengan menggunakan Google Colaboratory, YOLOv8, dan hasil yang didapat disimpan di Google Drive. Pelatihan dilakukan dengan total 300 epoch. Nilai rata-rata precision sebesar 92.57%, sedangkan nilai recall sebesar 88.2%. Akurasi model diperoleh nilai 93.2%. Hasil tersebut menggambarkan bahwa model relatif stabil dalam mengklasifikasikan objek dengan tepat. Custom model deteksi udang yang telah dihasilkan pada penelitian sebelumnya selanjutnya akan diimplementasikan ke dalam aplikasi berbasis website yang akan dibangun menggunakan React Js serta memanfaatkan onnxruntime-web API sebagai backend-nya. Selanjutnya, hasil

dari aplikasi yang dibangun dapat digunakan secara realtime/online.

Berdasarkan penjabaran uraian latar belakang, maka penelitian ini berjudul "Aplikasi Website "Shrimp-Detector App" untuk Deteksi Udang Menggunakan YOLOv8, Onnxruntime-Web API, dan React JS". Aplikasi tersebut akan menerapkan algoritma deep learning YOLOv8 sebagai sarana untuk training custom datasets, onnxruntime-web sebagai sarana untuk menjalankan custom model deep learning secara real-time pada website javascript, serta framework React JS sebagai framework untuk membangun aplikasi tersebut. Aplikasi tersebut dapat mempermudah petambak udang dalam mendeksi udang serta menentukan jumlah udang yang terdeteksi.

## Metode Penelitian

Penelitian ini terdiri dari 6 tahapan yaitu analisis kebutuhan, designing, setup and install dependencies, developing and onnxruntime implementation, testing and evaluating, dan deployment. Alur tahapan penelitian ini terdapat pada Gambar 1.



Gambar 1: Alur tahapan penelitian

### Analisis Kebutuhan

Pada tahap ini, dilakukan analisis kebutuhan sistem untuk melakukan proses pengembangan aplikasi website shrimp-detector. Analisis kebutuhan system ini tidak dapat dilewatkan dalam proses

penelitian karena dapat menentukan hasil custom model yang dihasilkan. Analisis kebutuhan system yang dilakukan mencakup analisis perangkat keras dan perangkat lunak.

Rangkaian tahapan yang dilakukan dalam penelitian ini membutuhkan perangkat keras untuk melakukan proses development. Perangkat keras tersebut digunakan untuk menjalankan instruksi yang diberikan oleh pengguna atau melalui perangkat lunak. Daftar perangkat keras yang digunakan terdapat pada Tabel 1.

Tabel 1: Daftar perangkat keras yang digunakan

No	Perangkat Keras	Keterangan
1	Macbook Pro (13 inci, M1, 2020)	Sebagai perangkat untuk melakukan proses coding menggunakan software Visual Studio Code.
2	Samsung Galaxy A32	Sebagai perangkat untuk menjalankan aplikasi website shrimp-detector yang dihasilkan dan melakukan pengetesan.

Selain perangkat keras, dalam penelitian ini juga menggunakan perangkat lunak untuk melakukan menunjang proses pengembangan aplikasi website shrimp-detector. Daftar perangkat lunak yang digunakan terdapat pada Tabel 2.

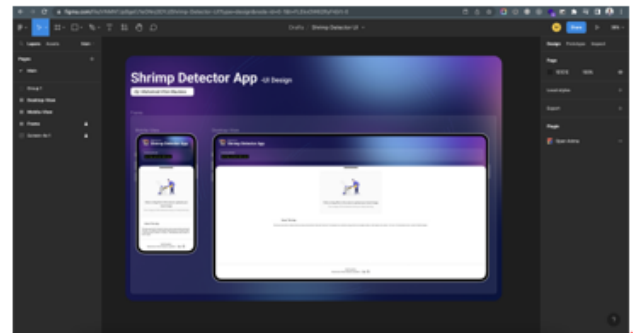
Tabel 2: Daftar perangkat lunak yang digunakan

No	Perangkat Lunak	Keterangan
1	Visual Studio Code untuk Chip Apple Silicon M1	Digunakan untuk pengembangan aplikasi website dalam penelitian
2	Figma	Digunakan untuk proses design user interface dari aplikasi website dalam penelitian ini.
3	Github	Digunakan sebagai software versioning control
4	Netlify	Digunakan sebagai tools untuk melakukan deployment aplikasi website dalam penelitian ini.
5	React Js	Framework javascript yang digunakan untuk membangun website.
6	Ant Design	Library javascript berupa design system yang berisikan komponen React JS untuk membuat UI website interaktif.
7	Onnxruntime-web API	Library javascript untuk menjalankan model ONNX di browser dan di Node.js secara real-time.

## Desain

Tahap desain dilakukan untuk merancang user interface (UI) atau tampilan dari aplikasi website shrimp-detector yang akan dibangun. Penulis menggunakan software figma sebagai alat untuk melakukan proses design [19]. User interface yang digunakan dalam website ini merupakan UI mobile-first, yang artinya website ini akan difokuskan untuk tampilan mobile yang dapat dijalankan pada smartphone.

Penulis mempertimbangkan tampilan mobile-first dikarenakan melihat fakta bahwa sebagian besar petani/petambak udang lebih sering menggunakan perangkat mobile seperti smartphone dibandingkan dengan desktop atau personal computer (PC). Oleh sebab itu, aplikasi website shrimp-detector memiliki fokus tampilan agar terlihat user friendly saat berada pada tampilan mobile. Tampilan user interface aplikasi shrimp-detector terdapat pada Gambar 2.



Gambar 2: Tampilan figma pada saat proses designing

## Proses Setup dan Instalasi Dependensi

Pada tahap ini, dilakukan proses setup react project dan install dependency yang akan dibutuhkan dalam proses developing aplikasi website shrimp-detector menggunakan software visual studio code. Dependency merupakan package pihak ketiga yang dapat digunakan dalam proses pengembangan website [20]. Proses setup website yang dilakukan yaitu menggunakan setup project React JS. Proses ini merupakan proses instalasi react project yang dilakukan menggunakan terminal bawaan visual studio code. Dalam melakukan proses instalasi React JS, penulis menggunakan Create React App (CRA). CRA merupakan salah satu cara untuk membuat aplikasi react modern tanpa melakukan konfigurasi dan hanya dengan satu perintah [21]. Dalam penelitian ini, untuk melakukan instalasi CRA, peneliti menggunakan package manager Yarn. Yarn merupakan system pengemasan (package) software dengan fitur tingkat lanjut yang mampu meningkatkan

alur kerja proses development [22]. Perintah untuk melakukan setup react project terdapat pada Gambar 3.

```

1 # Setup React project using CRA with Yarn
2 yarn create react-app shrimp_detection_yolov8

1 # Install default React Dependency
2 cd shrimp_detection_yolov8
3 yarn install

1 # Run React project
2 yarn start
    
```

Gambar 3: Perintah untuk setup project React Js

Selanjutnya dilakukan proses instalasi dependency-dependency yang dibutuhkan dalam penelitian ini. Dependency tersebut nantinya akan berguna dalam proses development aplikasi website shrimp-detector. Proses penginstalan dependency dapat dilakukan dengan perintah pada Gambar 4.

```

1 # Perintah untuk melakukan instalasi dependency yang dibutuhkan
2 yarn add onnxruntime-web antd boxicons @ant-design/icons @techstark/opencv-js
   typewriter-effect @craco/craco copy-webpack-plugin
    
```

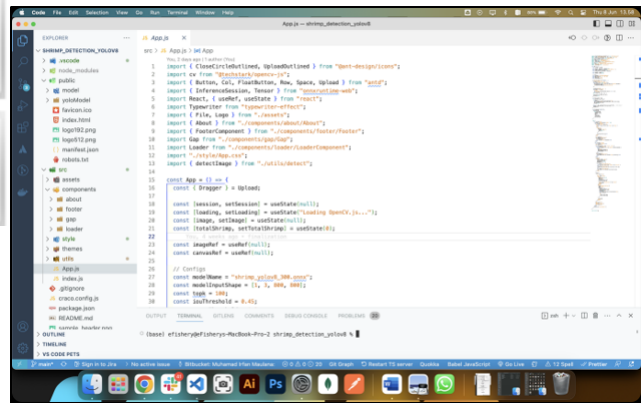
Gambar 4: Perintah untuk melakukan instalasi dependency yang akan digunakan

### Proses Developing dan Implementasi Onnxruntime-web API

Pada tahap ini, mulai dilakukan proses developing atau proses coding untuk membangun aplikasi website shrimp-detector. Proses developing dan implementasi onnxruntime-web dilakukan menggunakan visual studio code pada project react yang telah dibuat sebelumnya. Sebelum masuk ke dalam tahap ini, perlu dipastikan bahwa custom model yang telah dihasilkan pada proses custom training model telah dikonversi ke dalam format model ONNX. Proses developing berfokus kepada pembuatan tampilan website sesuai dengan user-interface yang telah dibuat sebelumnya dengan menggunakan figma. Proses developing tersebut dilakukan pada file App.js.

Pada Gambar 5, dilakukan proses pemrograman untuk membentuk tampilan website. Selain itu, terdapat juga penerapan beberapa fungsi untuk memulai/menjalankan opencv.js serta konfigurasi model ONNX yang akan digunakan. Proses

develop tampilan juga menggunakan beberapa component terpisah yang berada pada folder components. Component tersebut terdiri dari komponen about, footer, gap, dan loader. Selain itu, component yang digunakan juga menggunakan component yang di-import dari framework Ant Design untuk mempercepat proses development tampilan.



Gambar 5: Proses developing pada file App.js

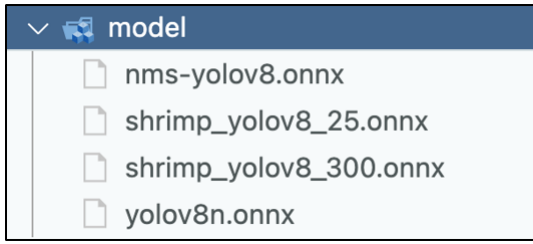
Setelah dilakukan proses development tampilan website, selanjutnya dilakukan proses implementasi custom model YOLOv8 yang telah dihasilkan ke dalam onnxruntime-web API sebagai backend dari aplikasi website yang dibangun. Custom model YOLOv8 yang telah dihasilkan, harus dikonversi ke dalam format ONNX agar dapat digunakan dalam onnxruntime-web API dan dapat berjalan secara real-time pada website berbasis javascript. Proses konversi dilakukan dengan menjalankan perintah dari yolo (ultralitics) menggunakan software google colab pro seperti pada Gambar 6.

```

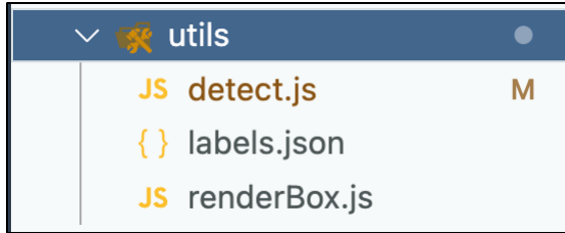
%cd (HOME)
yolo export model=(HOME)/runs/detect/train7/weights/best.pt format=onnx
    
```

Gambar 6: Perintah konversi custom model YOLOv8 ke format ONNX

Setelah dilakukan proses konversi, hasil konversi model ONNX dapat ditemukan pada folder runs/detect/train7/weights dengan nama file best.onnx. Selanjutnya file tersebut dapat digunakan sebagai model untuk melakukan deteksi objek secara real-time pada website javascript menggunakan onnxruntime-web. Model ONNX yang telah dihasilkan perlu dipindahkan kedalam folder public/model yang terdapat pada project aplikasi react yang telah disiapkan sebelumnya seperti pada Gambar 7.



Gambar 7: Folder model ONNX



Gambar 8: Folder utils yang berisi file untuk mengimplementasikan onnxruntime-web

Implementasi model ONNX yang telah didapatkan sebelumnya, dilakukan pada folder utils. Dalam folder utils terdapat tiga buah file, yaitu `detect.js`, `labels.json`, dan juga `renderBox.js` seperti pada Gambar 8. File pertama, yaitu `detect.js` berisi kode program untuk mendeteksi objek. Pada file ini, terdapat beberapa dependency yang digunakan seperti `@techstark/opencv-js` untuk menjalankan `opencv` di javascript, serta `onnxruntime-web` sebagai backend yang menjalankan custom model ONNX secara real-time. Selain itu, terdapat dua function utama, yaitu function untuk melakukan proses pre-processing image sebelum dideteksi serta function untuk melakukan deteksi citra menggunakan `onnxruntime-web`. File `labels.json` digunakan untuk mengkonfigurasi nama kelas yang akan ditampilkan pada bounding-box. File ini berisi Array yang didalamnya berupa nama kelas dengan menggunakan tipe data string. Pada penelitian ini terdapat dua kelas yang digunakan yaitu “not\_shrimp” dan “shrimp”. File `renderBox.js` digunakan untuk melakukan proses pembentukan bounding-box dan label class dari image yang berhasil terdeteksi. Pada file ini terdapat function utama yaitu function `renderBoxes` yang digunakan untuk membentuk bounding-box dan label class. Selain itu, didefinisikan juga kelas `Colors` yang digunakan untuk menampilkan warna dari masing-masing bounding-box secara acak.

### Pengetesan dan Evaluasi

Tahapan testing dan evaluating dilakukan dengan cara menjalankan website yang telah

dibangun secara local dengan alamat website `http://localhost:3000` yang dijalankan pada browser. Testing yang dilakukan adalah berupa self-test yang berfokus kepada pengujian pribadi yang dilakukan oleh peneliti berdasarkan beberapa parameter testing yang telah ditentukan. Daftar parameter testing yang digunakan terdapat pada Tabel 3.

Tabel 3: Daftar Parameter Self-testing yang digunakan

No	Parameter Testing	Acceptance Criteria
1	Tampilan Website	Tampilan website sudah sesuai dengan design UI yang telah dibuat menggunakan figma baik dari segi typografi, penggunaan warna, penggunaan asset berupa gambar, dan penggunaan icon.
2	Tampilan Website	Dapat menampilkan nama custom model ONNX yang digunakan.
3	Upload image file	Area upload image telah sesuai dengan design UI di figma, dapat dilihat dengan jelas, serta dapat diklik.  User dapat memilih file image setelah mengklik area upload.
4	Deteksi image	Image yang telah diupload akan muncul ke dalam website dengan penambahan bounding-box hasil deteksi image.  Bounding-box yang dihasilkan memiliki warna yang berbeda disetiap kelasnya  Label kelas object muncul dengan posisi disebelah kiri atas bounding-box dan memiliki warna background yang sesuai

Setelah dilakukan proses self-testing, selanjutnya akan dilakukan proses evaluasi. Parameter keberhasilan dalam proses self-testing juga dipengaruhi dengan ditemukan atau tidaknya defect atau bug pada aplikasi. Defect adalah kecacatan dalam software karena output tidak sesuai dari hasil yang diharapkan, sedangkan bug adalah kesalahan pada produk atau program software yang tidak berjalan sebagai mana mestinya. Jika hasil testing yang dilakukan telah sesuai dan tidak memiliki defect atau bug, maka tahapan akan dilanjutkan ke tahap berikutnya, yaitu tahap deployment. Akan tetapi, jika hasil testing menunjukkan hasil yang tidak sesuai maupun ditemukan defect atau bug, proses akan kembali ke tahap developing untuk memperbaiki defect yang ditemukan. Dalam proses pengujian, tingkat akurasi juga dapat dihitung menggu-

nakan confusion matrix [23] yang merupakan daftar data kelas yang di setiap kelas, data aktual diklasifikasikan sehingga dapat diamati kategori sampel mana yang mudah bingung dalam jaringan yang dimodifikasi. Dalam confusion matrix, baris mewakili kategori sebenarnya dari gambar uji. Kolom menunjukkan kelas citra uji dibagi dengan jaringan dites yang sebenarnya. Setelah mendapatkan hasil deteksi, kemudian akan dihitung nilai akurasi untuk setiap hasil deteksi yang dihasilkan menggunakan rumus berikut.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Keterangan:

TP : True Positif

TN : True Negatif

FP : False Positif

FN : False Negatif

Accuracy [23] menggambarkan seberapa akurat model dapat mengklasifikasikan dengan benar. Maka, accuracy merupakan rasio prediksi benar (positif dan negatif) dengan keseluruhan data. Dengan kata lain, accuracy merupakan tingkat kedekatan nilai prediksi dengan nilai aktual (sebenarnya). Nilai accuracy dapat diperoleh dengan persamaan (1).

		Actual Values	
		1 (Positive)	0 (Negative)
Predicted Values	1 (Positive)	<p><b>TP</b> (True Positive)</p>	<p><b>FP</b> (False Positive) <small>Type I Error</small></p>
	0 (Negative)	<p><b>FN</b> (False Negative) <small>Type II Error</small></p>	<p><b>TN</b> (True Negative)</p>

Gambar 9: Confusion Matrix

Pada Gambar 9, True Positive merupakan jumlah objek yang terdeteksi, sedangkan True Negative merupakan jumlah objek yang tidak terdeteksi. Untuk mengetahui jumlah aktual objek yang ada dapat diketahui melalui dari penjumlahan total kolom actual positive yaitu True Positive ditambahkan dengan False Negative. Sedangkan jumlah actual objek yang tidak terdeteksi dapat diketahui melalui penjumlahan total kolom actual negative, yaitu False Positive ditambahkan dengan True Negative [18].

### Deployment

Pada tahap ini, dilakukan proses deployment. Deployment adalah proses penerapan atau penem-

patan suatu aplikasi, sistem, solusi, atau teknologi ke dalam lingkungan operasional yang sesungguhnya setelah melalui tahap pengembangan (development). Proses deployment melibatkan penempatan aplikasi atau sistem ke dalam lingkungan produksi atau operasional yang biasanya berbeda dari lingkungan pengembangan atau pengujian [24]. Dengan kata lain, tahap deployment membuat aplikasi website shrimp-detector dapat diakses secara public di internet. Sebelum dilakukan proses deployment, perlu dilakukan proses sinkronisasi github dengan kode program yang ada di visual studio code. Proses sinkronisasi tersebut dilakukan melakukan push kode program ke github. Setelah melakukan sinkronisasi, maka proses deployment dapat dilakukan.

Tahapan proses *deployment* yang dilakukan adalah sebagai berikut.

1. Menambahkan site baru dengan melakukan import existing project yang ada di github
2. Menghubungkan akun github dengan Netlify
3. Memilih repository github yang digunakan dalam penelitian
4. Melakukan konfigurasi pengaturan website yang akan dipublish
5. Menjalankan proses deployment

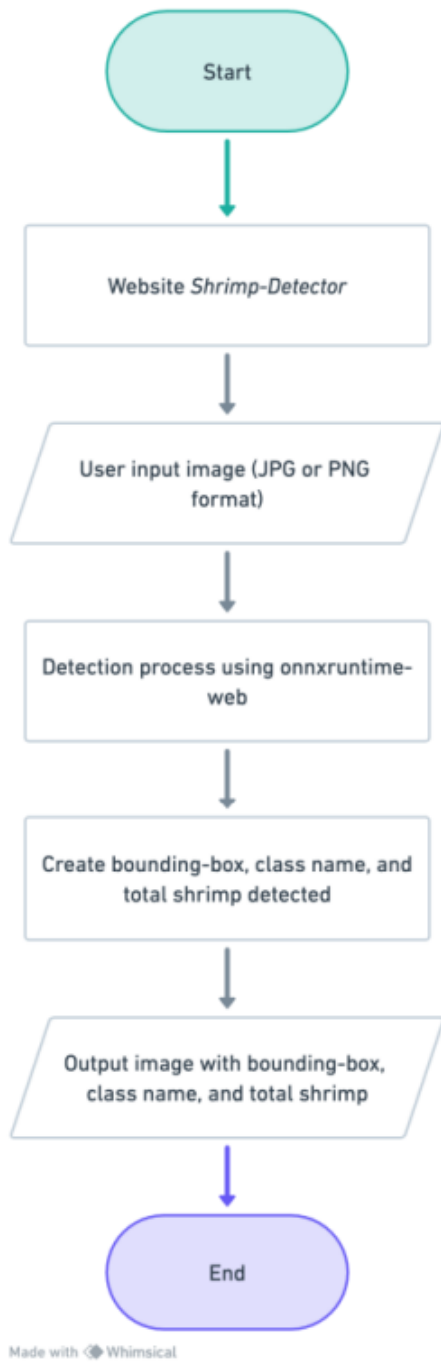
Setelah proses deployment selesai dilakukan, maka aplikasi website shrimp-detector dapat diakses secara public di internet. Alamat url untuk mengakses aplikasi ini yaitu <http://shrimp-detector.netlify.app/>.

### Hasil dan Pembahasan

Pada bagian ini, akan dijelaskan mengenai proses pembuatan aplikasi dilakukan dengan menerapkan custom model ONNX hasil konversi dari format .pt sebelumnya agar dapat dijalankan secara real-time menggunakan onnxruntime-web. Proses pembuatan program dilakukan menggunakan *visual studio code* sebagai *code editor*, lihat Gambar 10.

### Konfigurasi dan Inisialisasi Custom Model ONNC pada file App.js

Setelah sebelumnya memindahkan custom model dalam format ONNX seperti pada Gambar 7, penelitian ini menggunakan custom model ONNX shrimp\_yolov8\_300.onnx yang sebelumnya dikonversi dari hasil custom training model YOLOv8. Model tersebut telah ditraining menggunakan pre-trained model YOLOv8 versi nano yang memiliki ukuran paling kecil sehingga dapat berjalan diperangkat mobile sekalipun. Selain itu, model tersebut juga telah ditraining selama 7,5 jam dengan jumlah epoch sebanyak 300 epoch dengan total datasets sebanyak 6630 data image.



Gambar 10: Alur Proses Coding dan Implementasi Custom Model ONNX

Selanjutnya dilakukan proses konfigurasi awal serta inialisasi custom model ONNX pada file App.js menggunakan visual studio code. Proses konfigurasi dan inialisasi ini dilakukan untuk memastikan custom model ONNX dapat digunakan pada onnxruntime-web yang berperan sebagai backend dalam aplikasi website shrimp-detector. Proses konfigurasi tersebut dilakukan pada file App.js seperti pada Gambar 11.

Potongan kode tersebut terdapat pada file App.jsx. Variabel modelName digunakan untuk mengkonfigurasi nama file custom model ONNX

yang digunakan. Variabel modelInputShape digunakan untuk mengatur input\_shape gambar yang akan dideteksi. Input\_shape umumnya adalah bentuk data input yang diberikan pada model YOLO saat pelatihan. Model tidak dapat mengetahui bentuk data pelatihan. Bentuk tensor (lapisan) lainnya dihitung secara otomatis. Oleh karena itu, kita dapat mengatur model inputShape. Secara umum di YOLO model, model input\_shape terdiri dari: [batch, channels, width, height]. Variable topk digunakan untuk mengatur jumlah maksimum kotak yang akan dipilih per kelas. Variable iouThreshold digunakan untuk mengatur threshold Intersection Over Union (IoU). IoU digunakan untuk mengevaluasi algoritma deteksi objek. Variabel scoreThreshold digunakan untuk menentukan bounding-box mana yang harus dihilangkan berdasarkan score yang ditentukan.

```

shrimp_detection_yolov8 - App.js
1 // Configs
2 const modelName = "shrimp_yolov8_300.onnx";
3 const modelInputShape = [1, 3, 800, 800];
4 const topk = 100;
5 const iouThreshold = 0.45;
6 const scoreThreshold = 0.3;
  
```

Gambar 11: Konfigurasi custom model ONNX pada App.js

```

shrimp_detection_yolov8 - App.js
1 // wait until opencv.js initialized
2 cv["onRuntimeInitialized"] = async () => {
3 // create session
4 setLoading("Loading YOLOv8 model...");
5 const [yolov8, nms] = await Promise.all([
6 InferenceSession.create(`${process.env.PUBLIC_URL}/model/${modelName}`),
7 InferenceSession.create(
8   `${process.env.PUBLIC_URL}/model/nms-yolov8.onnx`
9 ),
10 ]);
11 // warmup main model
12 setLoading("Warming up model...");
13 const tensor = new Tensor(
14   "float32",
15   new Float32Array(modelInputShape.reduce((a, b) => a * b)),
16   modelInputShape
17 );
18 await yolov8.run({ images: tensor });
19
20 setSession({ net: yolov8, nms: nms });
21 setLoading(null);
22 };
  
```

Gambar 12: Instalasi onnxruntime-web

Pada Gambar 12, potongan kode tersebut digunakan untuk melakukan inialisasi onnxruntime-web yang akan menjalankan custom model onnx secara real-time sebagai backendnya. Potongan kode tersebut terdapat pada file App.jsx. Potongan kode tersebut akan membuat sesi onnx agar dapat berjalan secara real-time sebagai backend yang akan mendeteksi gambar yang diinputkan.



## Load Image dan Menjalankan Deteksi

Deteksi Pada tahap ini, dilakukan proses coding agar user dapat melakukan upload gambar yang akan digunakan pada proses deteksi. Selain itu, pada tahap ini, setelah user melakukan proses upload gambar, maka secara otomatis akan menjalankan fungsi deteksi yang terdapat pada file detect.js.



```

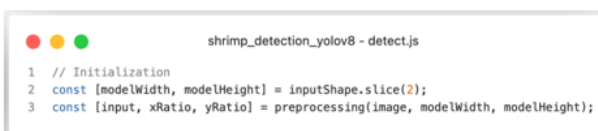
1 // Load image and run detection
2 const onLoadImage = async () => {
3   setTotalShrimp(
4     await detectImage(
5       imageRef.current,
6       canvasRef.current,
7       session,
8       topk,
9       iouThreshold,
10      scoreThreshold,
11      modelInputShape
12    )
13  );
14 };
    
```

Gambar 13: Function onLoadImage untuk menjalankan deteksi

Pada Gambar 13, setelah user memilih image yang akan dideteksi, maka program akan menjalankan fungsi onLoadImage yang terdapat pada file App.jsx. fungsi tersebut akan memanggil fungsi detectImage yang terdapat pada file detect.js. Fungsi tersebut juga mengirimkan beberapa parameter seperti: imageRef.current yang berfungsi untuk menentukan imageRef pada gambar yang dideteksi, canvasRef.current yang berfungsi untuk menentukan canvasRef untuk membentuk bounding-box dan label kelas pada gambar yang dideteksi, session yang berfungsi sebagai sesi onnxruntime-web yang sedang berjalan, nilai topk, nilai iouThreshold, nilai scoreThreshold, serta modelInputShape. Selanjutnya, fungsi detectImage juga akan mengembalikan nilai berupa total shrimp yang berhasil dideteksi.

### Pre-processing Image

Pada tahap ini dilakukan proses coding untuk melakukan pre-processing terhadap gambar yang sudah di upload oleh user. Code fungsi pre-processing ini terdapat pada file detect.js.



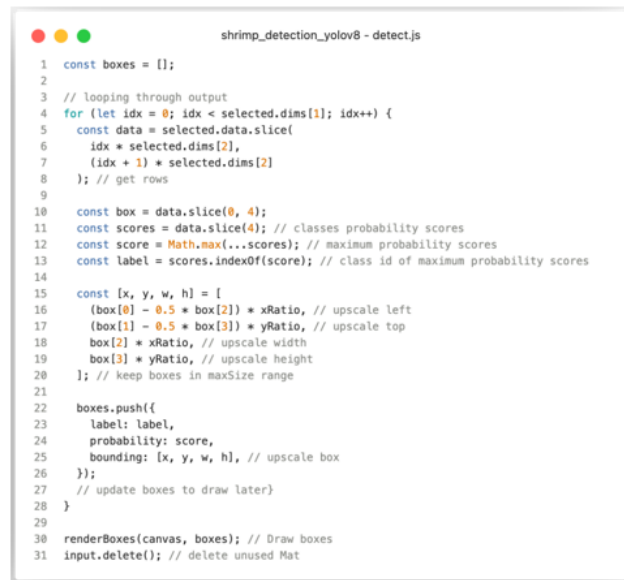
```

1 // Initialization
2 const [modelWidth, modelHeight] = inputShape.slice(2);
3 const [input, xRatio, yRatio] = preprocessing(image, modelWidth, modelHeight);
    
```

Gambar 14: Inisialisasi proses deteksi objek

Gambar 14 merupakan potongan kode program function detectImage yang terdapat pada file detect.js. Potongan kode program tersebut digunakan untuk melakukan inisialisasi proses deteksi objek. Variabel [modelWidth, modelHeight] merupakan variable hasil destruct dari inputShape. Variable tersebut berisi lebar model serta tinggi model yang akan dideteksi dan telah ditentukan sebelumnya pada file App.jsx. Pada line berikutnya akan memanggil function preprocessing untuk melakukan proses preprocessing pada gambar yang akan dideteksi. Pemanggilan function tersebut juga mengirimkan tiga parameter yaitu image, modelWidth, dan modelHeight. Selanjutnya terdapat function preprocessing yang digunakan untuk melakukan pre-processing pada image yang akan dideteksi. Pada function tersebut, dilakukan proses pembentukan image matrix yang akan digunakan untuk mengkonversi gambar ke dalam format BRG. Selanjutnya, akan dilakukan proses mengkonversi gambar agar menjadi berdimensi [n x n]. Setelah itu, gambar yang sudah diproses sebelumnya, akan diubah formatnya menjadi blob image. Function tersebut akan mengembalikan hasil akhir berupa variable input yang berisi matrix gambar, variable xRatio, serta variable yRasio.

## Proses Deteksi menggunakan Onnxruntime-web API



```

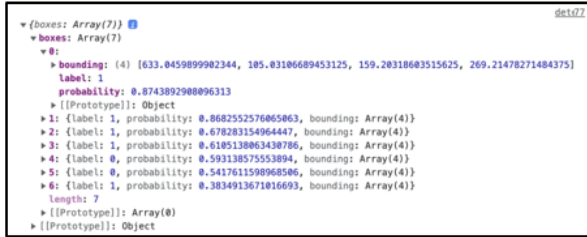
1 const boxes = [];
2
3 // looping through output
4 for (let idx = 0; idx < selected.dims[1]; idx++) {
5   const data = selected.data.slice(
6     idx * selected.dims[2],
7     (idx + 1) * selected.dims[2]
8   ); // get rows
9
10  const box = data.slice(0, 4);
11  const scores = data.slice(4); // classes probability scores
12  const score = Math.max(...scores); // maximum probability scores
13  const label = scores.indexOf(score); // class id of maximum probability scores
14
15  const [x, y, w, h] = [
16    (box[0] - 0.5 * box[2]) * xRatio, // upscale left
17    (box[1] - 0.5 * box[3]) * yRatio, // upscale top
18    box[2] * xRatio, // upscale width
19    box[3] * yRatio, // upscale height
20  ]; // keep boxes in maxSize range
21
22  boxes.push({
23    label: label,
24    probability: score,
25    bounding: [x, y, w, h], // upscale box
26  });
27  // update boxes to draw later
28 }
29
30 renderBoxes(canvas, boxes); // Draw boxes
31 input.delete(); // delete unused Mat
    
```

Gambar 15: Code untuk menentukan koordinat bounding-box dan memanggil fungsi renderBoxes

Pada tahap ini dilakukan proses coding untuk menjalankan sesi onnxruntime-web yang akan mendeteksi gambar yang telah di-upload oleh user. Pada Gambar 15, kode tersebut digunakan untuk menentukan koordinat bounding-box untuk setiap



objek yang dideteksi. Variabel `selected` yang dihasilkan pada proses sebelumnya akan diolah menggunakan perulangan untuk menentukan koordinat bounding-box. Proses tersebut akan menghasilkan label, score, serta koordinat bounding-box yang akan dimasukkan ke dalam variable array `boxes`.



Gambar 16: Contoh hasil label, score, dan koordinat bounding-box pada variable `boxes`

Gambar 16 merupakan contoh output hasil label, score, serta koordinat bounding-box pada variable `boxes`. Koordinat bounding-box tersebut terdiri dari `[x, y, width, height]`. Setelah mendapatkan beberapa data tersebut, selanjutnya akan function `detectImage` akan memanggil function `renderBoxes` yang terdapat pada file `renderBox.js` serta mengirimkan dua parameter yaitu `canvas` yang merupakan `canvas javascript` untuk membentuk bounding-box, serta variable `boxes` yang berisi informasi mengenai bounding-box yang akan dibentuk.

### Membuat *Bounding-box*, Nama Kelas, dan Jumlah Total Uang yang Terdeteksi

Pada tahap ini, dilakukan proses coding untuk membuat bounding-box berdasarkan informasi koordinat bounding-box yang telah didapatkan sebelumnya, membuat label kelas berdasarkan kelas objek yang terdeteksi, serta menentukan jumlah uang yang terdeteksi berdasarkan total dari kelas `shrimp`. Semua proses tersebut terdapat pada file `renderBox.js` pada folder `utils`.

Gambar 17, dilakukan proses pembentukan bounding-box. Proses pembentukan bounding-box dimulai dengan konfigurasi `canvas javascript` sebagai dasar bounding-box serta konfigurasi jenis huruf serta style dari huruf yang akan digunakan untuk label kelas. Selanjutnya, proses pembentukan masing-masing bounding-box dari hasil objek yang terdeteksi dilakukan dengan menggunakan perulangan `forEach` yang dilakukan pada variable `boxes`. Dalam perulangan tersebut terdapat proses penggambaran bounding-box, penggambaran border box, penggambaran background label kelas, serta penggambaran label kelas. Proses perhitungan jumlah uang yang terdeteksi dilakukan dengan

menghitung `length` dari variable `boxes` yang memiliki kelas `“shrimp”`. Setelah itu function `detectImage` akan mengembalikan variable `totalShrimp` yang akan ditampilkan ke dalam tampilan website.



Gambar 17: Code untuk membentuk bounding-box

### Menampilkan Hasil Deteksi

Pada tahap ini dilakukan proses coding untuk menampilkan hasil dari deteksi yang telah dilakukan proses ini akan menampilkan hasil deteksi dengan gambar, bounding-box, label kelas, tingkat akurasi, serta total jumlah uang yang terdeteksi. Code pada tahap ini terdapat pada file `App.js`.

Gambar 18 digunakan untuk menampilkan hasil akhir deteksi objek yang telah dilakukan. Image yang telah diupload oleh user akan ditampilkan menggunakan tag `HTML <img/>` yang juga akan menjalankan function `onLoadImage` Ketika user mengupload image. Selanjutnya, bounding-box hasil deteksi akan ditampilkan dalam tag `HTML <canvas/>`. Total jumlah uang yang terdeteksi sebelumnya juga ditampilkan dengan memanggil value dari variable state `totalShrimp`.

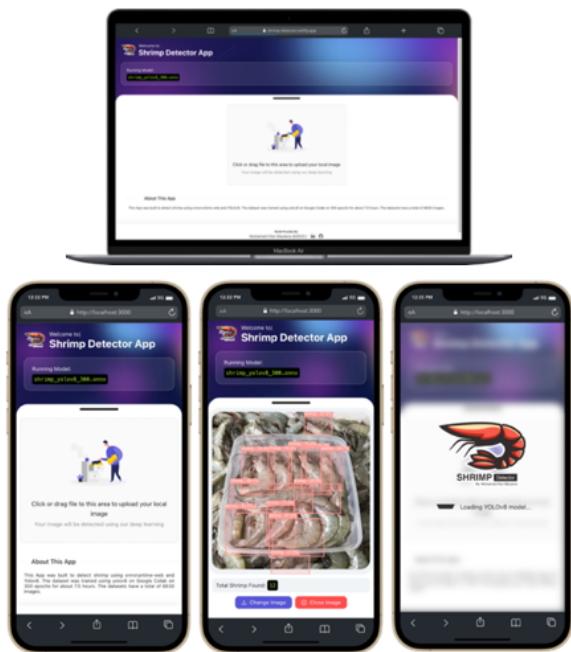
```

shrimp_detection_yolov8 - App.js
1 <div style={{ display: image ? "block" : "none" }}>
2 <div className="content">
3 
4 <canvas
5   id="canvas"
6   width={modelInputShape[2]}
7   height={modelInputShape[3]}
8   ref={canvasRef}
9 />
10 </div>
11 <Gap height={20} />
12 <div className="card-content">
13   Total Shrimp Found: <code className="code">{totalShrimp}</code>
14 </div>
15 </div>
    
```

Gambar 18: Code untuk menampilkan output deteksi objek

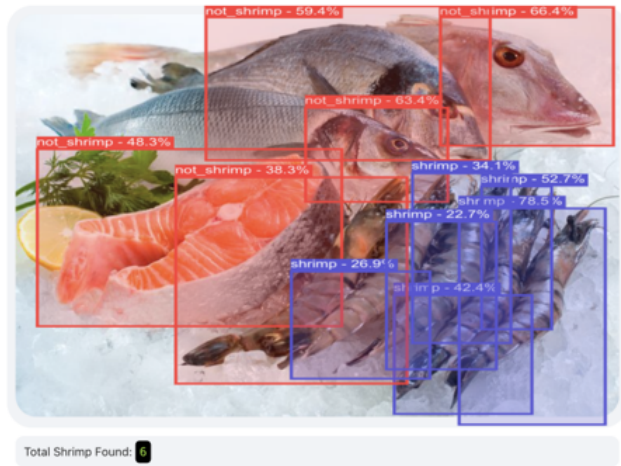
### Proses Pengujian

Pada tahap ini, akan dilakukan proses pengujian terhadap aplikasi website shrimp-detector yang telah dibangun. Proses pengujian dilakukan menggunakan website shrimp-detector yang telah didploy menggunakan software Netlify sebelumnya. Proses pengujian dilakukan dengan proses end-to-end mulai dari user mengakses website sampai dengan hasil deteksi objek muncul. Pengujian dilakukan agar website yang telah dibangun dapat berjalan dengan baik pada saat dijalankan. Aplikasi website shrimp-detector diakses melalui alamat url <https://shrimp-detector.netlify.app/>. Tampilan aplikasi website shrimp-detector terdapat pada Gambar 19.



Gambar 19: Tampilan website Shrimp-Detector pada Desktop dan Mobile

Gambar 20, dapat dilihat hasil pengujian deteksi secara langsung menggunakan aplikasi website “Shrimp Detector App” secara real-time, pada pengujian tersebut secara kasat mata dapat dilihat pada gambar terdapat 5 ekor udang dan sisanya merupakan bukan udang. Deteksi objek yang dilakukan menghasilkan total udang yang terdeteksi adalah sebanyak 5 ekor, Terdapat perbedaan 1 ekor udang yang terdeteksi dibandingkan dengan jumlah udang yang tampak secara kasat mata. Berdasarkan hal tersebut, dapat dilihat bahwa tingkat akurasi yang dihasilkan belum mencapai 100%. Berdasarkan persamaan (1), dapat dihitung tingkat akurasi yang dihasilkan. Hasil deteksi yang didapatkan pada Gambar 20 dapat diubah kedalam bentuk table confusion matrix seperti Gambar 21.



Gambar 20: Tampilan Hasil Deteksi Menggunakan Aplikasi Website “Shrimp Detector App”

		Actual	
		1 (Positive)	0 (Negative)
Predicted	1 (Positive)	6 (True Positive)	0 (False Positive)
	0 (Negative)	1 (False Negative)	4 (True Negative)

Gambar 21: Confussion Matrix dari proses pengujian

Pada Gambar 21, Jumlah udang yang terdeteksi adalah 6 yang disebut sebagai True Positive. Sedangkan jumlah bukan udang yang terdeteksi adalah 4 yang disebut sebagai True Negative. Jumlah aktual udang yang ada dapat dihitung dari penjumlahan total kolom actual positive yaitu sebanyak 6 (TP) + 1 (FN) = 7. Jumlah aktual bukan udang yang ada dapat dihitung dari penjumlahan total kolom actual negative yaitu sebanyak 0 (FP) + 4 (TN) =

4. Selanjutnya akurasi yang dihasilkan dapat dihitung dengan kalkulasi berikut ini.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

$$accuracy = \frac{6 + 4}{6 + 4 + 1 + 0} \times 100\%$$

$$accuracy = \frac{10}{11} \times 100\%$$

$$accuracy = 90,9\%$$

Dari hasil pengujian, maka menghasilkan tingkat akurasi deteksi udang sebesar 90,9%.

## Penutup

Pada penelitian ini, aplikasi website “Shrimp-Detector App” yang menjadi output penelitian berhasil dibangun menggunakan framework React Js sebagai frontednya, Onnxruntime-web API sebagai backendnya, serta custom model shrimp detector ONNX sebagai dasar dari objek deteksi YOLOv8 yang digunakan. Aplikasi website tersebut dapat diakses melalui alamat url <https://shrimp-detector.netlify.app/>. Penelitian ini juga merupakan penelitian lanjutan yang dilakukan oleh Muhamad Irfan Maulana dan Rina Noviana pada tahun 2023 yang berjudul “Training Custom Model Deteksi Udang menggunakan YOLOv8”. Pada penelitian tersebut, menghasilkan custom model deteksi udang yang dibangun menggunakan YOLOv8 yang selanjutnya dikonversi menjadi model ONNX dan digunakan dalam proses development aplikasi website “Shrimp-Detector App”. Aplikasi website tersebut berhasil mendeteksi objek berupa udang dan bukan udang serta menghitung jumlah udang yang terdeteksi. Hasil pengujian yang didapatkan memperoleh tingkat akurasi deteksi udang sebesar 90,9% yang menggambarkan bahwa aplikasi website yang dihasilkan cukup stabil dalam mengklasifikasikan objek secara tepat.

## Daftar Pustaka

- [1] S. Rejeki, R. W. Aryati dan L. L. Widowati, “Pengantar Akuakultur”, Semarang: UNDIP Press Semarang, 2019.
- [2] S. S. Fatimah, S. Marwanti and S. Supardi, “Export Performance of Indonesian Shrimp In the United States During 2009-2017: A Constant Market Share Model Approach”, *Jurnal Sosial Ekonomi Kelautan dan Perikanan*, vol. 15, no. 1, pp. 57–67, doi: <http://dx.doi.org/10.15578/jsekp.v15i1.7677>, Jun. 2020.
- [3] Indarto, “KKP Target Konsumsi Ikan Tahun 2020 Sebesar 56,39 Kg”, *Tabloid Sinar Tani*, 2020.
- [4] Anonim, *Budidaya Udang Vaname di Tambak Milenial*, Balai Perikanan Budidaya Air Payau Situbondo, Situbondo: Kementerian Kelautan dan Perikanan, 2021.
- [5] Lusiana Rahma, Hadi Syaputra, A.Haidar Mirza, and usan Dian Purnamasari, “Objek Deteksi Makanan Khas Palembang Menggunakan Algoritma YOLO(You Only Look Once)”, *Jurnal Nasional Ilmu Komputer*, vol. 2, no. 3, 2021.
- [6] Z. Q. Zhao, P. Zheng, S. T. Xu and X. Wu, “Object Detection with Deep Learning: A Review”, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11. Institute of Electrical and Electronics Engineers Inc., pp. 3212–3232, doi: 10.1109/TNNLS.2018.2876865, Nov. 01, 2019.
- [7] A. R. Pathak, M. Pandey and S. Rautaray, “Application of Deep Learning for Object Detection”, in *Procedia Computer Science*, Elsevier B.V., pp. 1706–1717, doi: 10.1016/j.procs.2018.05.144, 2018 .
- [8] D. Putra dan Westriningsih, “Pengolahan citra digital”, Yogyakarta: ANDI, 2010.
- [9] Supono, “Teknologi Produksi Udang”, Yogyakarta : Plantaxia, 2017.
- [10] Y. Wang, H. Wang and Z. Xin, “Efficient detection model of steel strip surface defects based on YOLO-V7”, *IEEE Access*, Vol. 10, pp:133936 - 133944, doi: 10.1109/ACCESS.2022.3230894, 2022.
- [11] P. Jiang, D. Ergu, F. Liu, Y. Cai and B. Ma, “A Review of Yolo Algorithm Developments”, *Procedia Comput Sci*, vol. 199, Pages 1066-1073, <https://doi.org/10.1016/j.procs.2022.01.135>, 2022.
- [12] N. Buhl, “YOLO models for Object Detection Explained : Evolution, Algorithm, and Applications”, diakses daring pada: <https://encord.com/blog/yolo-object-detection-guide/#h2>, April 2024.
- [13] M. A. bin Zuraimi and F. H. K. Zaman, “Vehicle Detection and Tracking using YOLO and DeepSORT”, 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), pp. 23–29, 2021.
- [14] Z. Wei, M. Chang and Y. Zhong, “Fruit Freshness Detection Based on YOLOv8 and SE attention Mechanism”, *Academic Journal of Science and Technology*, vol. 6, no. 1, 2023.

- [15] J. Solawetz and Francesco, “What is YOLOv8? The Ultimate Guide”, Roboflow, diakses daring pada: <https://blog.roboflow.com/whats-new-in-yolov8/#yolov8-accuracy-improvements>, 2024.
- [16] Ultralytics Team, “Introducing Ultralytics YOLOv8”, Ultralytics, diakses daring pada: <https://ultralytics.com/article/Introducing-Ultralytics-YOLOv8>, 2023
- [17] S. Armalivia, Z. Zainuddin, A. Achmad and Muh. A. Wicaksono, “Automatic Counting Shrimp Larvae Based You Only Look Once (YOLO)”, 2021 International Conference on Artificial Intelligence and Mechatronics Systems (AIMS), pp. 1–4, 2021.
- [18] M. I. Maulana and R. Noviana, “Training Custom Model Deteksi Uang menggunakan YOLOv8”, Jurnal Ilmiah Komputasi, Vol. 22, no. 4, pp. 505–514, doi: 10.32409/jikstik.22.4.3526, Dec. 2023.
- [19] Dianne Eberhardt, “Figma Prototype: What is it and why use it for design?”, <https://designproject.io/blog/figma-prototype>, 2022.
- [20] Anonym, “Difference between dependencies, devDependencies and peerDependencies”, diakses daring pada: <https://www.geeksforgeeks.org/difference-between-dependencies-devdependencies-and-peerdependencies/>, January 2024.
- [21] C. Murphy, “Choose the Best Tool for You: Create React App (CRA) vs. Next.js”, Prismic, diakses daring pada: <https://prismic.io/blog/create-react-app-cra-vs-nextjs>, 2023.
- [22] Beca, “What is Yarn?: What It’s Made From, How You Make It and More”, Contrado, diakses daring pada: <https://www.contrado.co.uk/blog/what-is-yarn/>, Jan 2020.
- [23] K. S. Nugroho, “Confusion Matrix untuk Evaluasi Model pada Supervised Learning”, Medium, diakses daring pada: <https://ksnugroho.medium.com/confusion-matrix-untuk-evaluasi-model-pada-supervised-machine-learning-bc4b1ae9ae3f>, 2019..
- [24] H. I. Abdulmalik, “Deployment: Pengertian, Tujuan, dan Jenis-jenisnya”, diakses daring pada: <https://www.dicoding.com/blog/deployment-pengertian-tujuan-dan-jenis-jenisnya/>, April 2024.