Jurnal Ilmiah
Teknologi Sistem
Informasi

Jitsi

Vol. 5 (2024) No. 4

https://jurnal-itsi.org/index.php/jitsi; E-ISSN: 2722-4600; ISSN: 2722-4619
DOI: 10.62527/jitsi.5.4.298

Deteksi Objek untuk Produk Retail dengan TensorFlow 2

Ahmad Azzam Alhanafi [#], Arrie Kurniawardhani [#]

[#] *Departemen Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia, Yogyakarta, 55584, Indonesia*
E-mail: azzam.alhanafi[at]uii.ac.id, arrie.kurniawardhani[at]uii.ac.id

A B S T R A C T S

On-shelf availability is a crucial aspect in the retail industry, directly impacting customer satisfaction and sales. Artificial intelligence-based object detection technology can enhance efficiency in monitoring product availability. This study examines the implementation of TensorFlow 2 for detecting retail products on shelves, using the SSD MobileNetV2 FPNLite architecture. Three model variations were developed based on input image sizes: 320x320, 640x640, and 1024x1024. The models were trained using transfer learning with a dataset containing 128 retail product classes. Evaluation results show that the 640x640 model achieved the best performance in terms of the trade-off between precision and speed, with a mAP of 0.72049 and an inference time of 0.283 seconds. The 320x320 model had the fastest inference time of 0.073 seconds, making it suitable for real-time applications. This study offers a solution to improve retail stock management through automatic object detection, aiming to reduce the risk of out-of-stock situations

Manuscript received Oct 26, 2024; revised Oct 29, 2024. accepted Dec 2, 2024 Date of publication Dec 30, 2024. International Journal, JITSI : Jurnal Ilmiah Teknologi Sistem Informasi licensed under a Creative Commons Attribution-Share Alike 4.0 International License

A B S T R A K

Ketersediaan barang di rak (on-shelf availability) merupakan aspek penting dalam industri retail yang berdampak langsung pada kepuasan konsumen dan penjualan. Teknologi deteksi objek berbasis Kecerdasan Buatan dapat meningkatkan efisiensi dalam memantau ketersediaan produk. Penelitian ini mengkaji penerapan TensorFlow 2 untuk deteksi objek produk retail di rak, menggunakan arsitektur SSD MobileNetV2 FPNLite. Tiga variasi model dikembangkan berdasarkan ukuran input gambar, yaitu 320x320, 640x640, dan 1024x1024. Model dilatih menggunakan metode transfer learning dengan dataset yang berisi 128 kelas produk retail. Hasil evaluasi menunjukkan bahwa model dengan ukuran gambar 640x640 memberikan performa terbaik dalam hal trade-off antara presisi dan kecepatan, dengan mAP sebesar 0.72049 dan waktu inferensi 0.283 detik. Model 320x320 memiliki waktu inferensi tercepat sebesar 0.073 detik, menjadikannya cocok untuk aplikasi real-time. Penelitian ini menawarkan solusi untuk meningkatkan pengelolaan stok produk retail dengan deteksi objek otomatis, guna mengurangi risiko out-of-stock.

Keywords / Kata Kunci — ***TensorFlow; Deteksi Objek; Perangkat Bergerak; On-Shelf Availability***

CORRESPONDING AUTHOR

Ahmad Azzam Alhanafi
Departemen Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia, Yogyakarta, 55584, Indonesia
Email: azzam.alhanafi[at]uii.ac.id

1. PENDAHULUAN

Sejak penemuannya di sekitar abad ke-18, revolusi industri telah mengubah gaya hidup manusia dan menciptakan kemajuan pesat di berbagai industri di seluruh dunia [1]. Industri retail merupakan salah satu industri yang terdampak dari adanya revolusi industri ini [2]. Era revolusi industri 4.0 ditandai dengan penerapan

Kecerdasan Buatan secara luas di berbagai sektor industri, termasuk dalam pengambilan keputusan yang canggih berbasis data dalam operasional industri [3]. Oleh karena itu, setiap pelaku industri retail perlu menerapkan Kecerdasan Buatan demi meningkatkan daya saing dalam dunia industri retail modern. Salah satu aspek penting yang ada dalam industri retail adalah ketersediaan barang di rak (on-shelf availability).

Ketersediaan barang di rak merupakan faktor krusial dalam industri retail, karena secara langsung mempengaruhi loyalitas konsumen dan permintaan terhadap produk [4]. Ketika suatu produk tidak tersedia di rak atau out-of-stock, konsumen cenderung beralih ke produk atau ke toko lain, yang dapat mengakibatkan berkurangnya penjualan dan menurunnya kepuasan pelanggan [5]. Oleh karena itu, menjaga on-shelf availability menjadi prioritas utama dalam manajemen rantai pasok retail.

Sebelum adanya penerapan teknologi Kecerdasan Buatan, para pelaku industri retail melakukan pengecekan ketersediaan produk secara manual dengan mendatangi setiap rak yang berisi produk tersebut [6]. Proses ini membutuhkan waktu, tenaga, dan konsentrasi yang tinggi. Untuk menjawab tantangan ini, teknologi Kecerdasan Buatan melalui deteksi objek dapat dimanfaatkan sebagai solusi untuk memantau ketersediaan produk secara otomatis sehingga mengurangi kesalahan manual dan meningkatkan kecepatan respons terhadap situasi out-of-stock.

Deteksi objek merupakan salah satu teknik dalam Visi Komputer yang memungkinkan komputer untuk mengidentifikasi dan melokalisasi objek-objek tertentu dalam gambar atau video. Proses ini tidak hanya mencakup klasifikasi objek, tetapi juga penentuan lokasi objek dalam bentuk bounding box [7]. Dalam pengembangannya, model deteksi objek dapat memanfaatkan framework atau kerangka kerja agar lebih terukur dan efisien. Salah satu framework yang cukup terkenal adalah TensorFlow.

TensorFlow adalah open-source framework Machine Learning yang pertama kali dikembangkan oleh Google pada tahun 2015 dan digunakan untuk mengembangkan dan menerapkan model Deep Learning [8]. TensorFlow mendukung berbagai operasi numerik berbasis grafik dataflow, yang memungkinkan para pengembang dan peneliti untuk membangun dan melatih model jaringan saraf kompleks dengan efisien [9]. TensorFlow juga menyediakan fleksibilitas dalam penerapannya, baik untuk platform perangkat keras seperti CPU maupun GPU, serta mendukung perangkat mobile melalui TensorFlow Lite [10]. TensorFlow 2, versi terbaru dari framework ini, menawarkan peningkatan kemudahan penggunaan dan fleksibilitas, terutama melalui integrasi API Keras, yang memungkinkan pengembangan model secara lebih intuitif. Dengan berbagai API dan model pretrained yang tersedia, TensorFlow menjadi pilihan ideal untuk pengembangan solusi deteksi objek dalam skala industri retail.

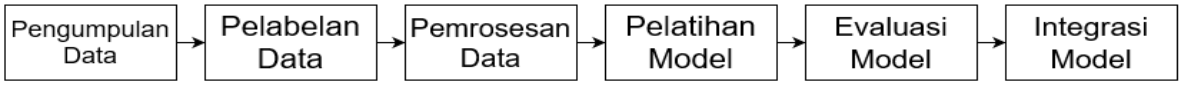
Arsitektur yang digunakan di dalam penelitian ini adalah SSD MobileNetV2 FPNLite. Arsitektur ini memiliki beberapa keunggulan yang relevan untuk aplikasi deteksi objek di industri retail, terutama untuk penggunaan di perangkat bergerak dengan sumber daya terbatas. SSD (Single Shot Multibox Detector) dikenal dengan efisiensinya dalam menghasilkan deteksi objek secara real-time tanpa memerlukan dua tahap deteksi, seperti pada metode Faster R-CNN, sehingga lebih cepat dalam inferensi [11]. Di sisi lain, MobileNetV2 adalah model ringan yang dirancang untuk perangkat dengan sumber daya terbatas, memanfaatkan blok bottleneck dan depthwise separable convolutions untuk mengurangi jumlah parameter dan komputasi yang diperlukan tanpa mengorbankan akurasi secara signifikan [12][13].

FPNLite (Feature Pyramid Network Lite) yang digunakan dalam kombinasi ini menawarkan keunggulan tambahan dengan meningkatkan kemampuan model dalam mendeteksi objek kecil melalui penggabungan informasi dari berbagai tingkatan fitur dalam jaringan. Dengan menggunakan versi "Lite" dari FPN, peningkatan akurasi dapat dicapai tanpa menambah kompleksitas komputasi secara signifikan, menjadikan arsitektur ini ideal untuk aplikasi real-time pada perangkat mobile [14]. Oleh karena itu, SSD MobileNetV2 FPNLite dipilih karena mampu menjaga keseimbangan antara kecepatan inferensi, akurasi, dan efisiensi komputasi yang dibutuhkan dalam sistem deteksi objek otomatis untuk pengelolaan stok produk retail.

Studi ini bertujuan untuk mengevaluasi performa model deteksi objek menggunakan TensorFlow 2 dengan arsitektur SSD MobileNetV2 FPNLite, baik pada komputer maupun perangkat bergerak. Penelitian ini akan mengukur akurasi dan efisiensi model dalam mendeteksi produk retail di rak, serta mengeksplorasi potensi penerapannya untuk meningkatkan efisiensi operasional dalam pengelolaan inventaris produk di masa mendatang

2. METODOLOGI PENELITIAN

Penelitian ini menggunakan kerangka kerja Machine Learning yakni TensorFlow versi 2. Kerangka ini dapat digunakan untuk membangun, melatih, dan mengimplementasikan model Machine Learning [15]. Pada penelitian ini, digunakan arsitektur SSD MobileNetV2 FPNLite yang tersedia di TensorFlow Object Detection API. Alur metodologi penelitian dapat dilihat pada Gambar 1



GAMBAR 1. Alur Metodologi Penelitian

2.1. Pengumpulan Data

Pengumpulan data dilakukan dengan mengambil gambar dari beberapa produk yang berada di rak toko ataupun minimarket. Perangkat yang digunakan untuk pengambilan gambar adalah smartphone. Perangkat tersebut dipilih karena gambar yang dihasilkan dari perangkat tersebut dapat merepresentasikan kondisi asli di lapangan saat proses bisnis berjalan. Produk retail yang dikhususkan pada penelitian ini adalah produk barang konsumen cepat habis atau *fast-moving consumer goods* (FMCG).

FMCG mencakup berbagai produk yang sering dibeli dan dikonsumsi secara rutin oleh konsumen, seperti makanan dan minuman, produk perawatan pribadi, produk kebersihan, dan kebutuhan rumah tangga. Produk-produk tersebut biasa ditemukan di rak toko retail atau supermarket. Untuk mendapatkan model deteksi objek yang baik, diperlukan jumlah dataset yang besar dan juga kualitas gambar yang baik, sesuai dengan gambar yang akan menjadi data uji, sehingga model dapat mendeteksi fitur-fitur yang ada di dalam objek dengan akurat.

2.2. Pelabelan Data

Setelah memiliki banyak gambar yang berisi produk-produk yang akan digunakan untuk data uji dan data pelatihan, gambar-gambar tersebut kemudian diberikan anotasi sehingga objek-objek yang ada dalam gambar tersebut memiliki bounding box dan label tersendiri. Alat yang digunakan untuk melakukan proses anotasi adalah "LabelImg" [16]. LabelImg adalah *open-source tool* yang banyak digunakan untuk anotasi data gambar dalam proyek deteksi objek. Setiap gambar yang telah dianotasi akan menghasilkan file XML yang berguna untuk menyimpan informasi berupa koordinat bounding box dari objek-objek yang ada pada gambar tersebut.

Format anotasi yang digunakan untuk pelatihan model adalah PASCAL VOC [17]. Format anotasi PASCAL VOC adalah standar yang umum digunakan untuk anotasi data dalam proyek deteksi objek. Anotasi ini disimpan dalam file XML dan mendeskripsikan informasi gambar serta objek yang terdapat di dalamnya seperti filename, size, dan koordinat bounding box.

2.3. Pemrosesan Data

Data yang sudah dianotasi kemudian dibagi ke dalam dua subset, yakni data latih dan data uji. Proses ini dikenal dengan istilah data splitting. Proses ini diperlukan untuk menguji performa model pada data yang belum pernah dilihat sebelumnya, sehingga dapat mengevaluasi seberapa baik model bisa melakukan generalisasi terhadap data baru. Setelah dilakukan splitting, data diubah ke dalam format TFRecord [18].

Format TFRecord menggabungkan semua file gambar dan file anotasi ke dalam satu file biner sehingga lebih efisien untuk digunakan dalam pipeline TensorFlow. Setelah itu, pemrosesan data juga dilakukan dengan melakukan konfigurasi terhadap ukuran input gambar, jumlah kelas, dan batch size. Semua konfigurasi tersebut ditulis di dalam file pipeline.config yang tersedia di dalam repositori TensorFlow Object Detection API [19].

2.4. Pelatihan Model

Proses pelatihan memanfaatkan arsitektur SSD MobileNetV2 FPNLite dengan jumlah langkah pelatihan sebesar 40.000 steps dan batch size sebesar 16. Jumlah step 40.000 dipilih karena ukuran dataset yang besar membutuhkan langkah pelatihan yang besar juga namun tidak berlebihan hingga mengakibatkan terjadinya overfitting. Jumlah batch size 16 dipilih karena menawarkan keseimbangan antara stabilitas dan kecepatan konvergensi. Batch size yang lebih besar bisa mempercepat pelatihan, tetapi memerlukan lebih banyak memori. Sementara batch size yang lebih kecil bisa memberikan hasil yang lebih stabil tetapi memakan waktu lebih lama. Jumlah step di angka puluhan ribu dan penggunaan batch size yang menyesuaikan kemampuan memori juga menjadi penggunaan yang umum digunakan di dalam dokumentasi TensorFlow Object Detection API [20].

Untuk mempercepat dan meningkatkan performa dari hasil pelatihan, penulis menggunakan metode *transfer learning* dengan memanfaatkan pre-trained model yang sudah dilatih terhadap dataset COCO 2017 [21]. Pelatihan dilakukan terhadap tiga model dengan variasi ukuran input resizer yang berbeda, yaitu 320x320, 640x640, dan 1024x1024. Variasi ukuran ini digunakan untuk menemukan keseimbangan yang optimal antara kecepatan (*speed*) dan performa (*performance*) model.

2.5. Evaluasi Model

Metrik yang digunakan untuk mengevaluasi performa model adalah *mean Average Precision* (mAP) [17]. Metrik ini memberikan gambaran keseluruhan mengenai precision dan recall [22] model dalam berbagai kategori objek. Evaluasi ini sangat penting untuk memahami performa model deteksi objek yang telah disesuaikan pada dataset baru dan mengidentifikasi aspek yang memerlukan perbaikan.

Untuk menghitung nilai mAP, diperlukan nilai *Average Precision* (AP) dari setiap kelas. AP mengukur area di bawah kurva precision-recall (precision-recall curve) untuk setiap kelas. Precision dan recall dihitung berdasarkan *Intersection over Union* (IoU), yaitu ukuran tumpang tindih antara bounding box prediksi model dengan ground

truth. Jika IoU antara prediksi dan ground truth melebihi ambang batas (biasanya 0.5), maka prediksi tersebut dianggap benar (true positive). Rumus untuk precision dan recall dapat dilihat pada Persamaan (1) dan (2).

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (1)$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (2)$$

Untuk pemahami Persamaan (1) dan (2), terdapat beberapa istilah yang perlu diketahui:

- True Positives adalah jumlah prediksi positif dengan ground truth positif.
- False Positives adalah jumlah prediksi positif dengan ground truth negatif.
- True Negatives adalah jumlah prediksi negatif dengan ground truth negatif
- False Negatives adalah jumlah prediksi negatif dengan ground truth positif.

Precision adalah rasio antara prediksi benar (True Positives) dan total prediksi positif (True Positives dan False Positives. Sedangkan Recall adalah rasio antara prediksi benar (True Positive) dan total objek ground truth dalam gambar (True positives dan False Negatives). Dalam deteksi objek, precision dan recall biasanya diplot dalam grafik untuk membentuk kurva precision-recall. AP diperoleh dengan menghitung area di bawah kurva ini, seperti yang dituliskan dalam Pesamaan (3).

$$\text{AP} = \int_0^1 p(r)\ dr \quad (3)$$

Pada persamaan (3), $p(r)$ adalah precision sebagai fungsi dari recall r. Semakin besar area di bawah kurva, semakin baik kinerja model untuk kelas tersebut. Sedangkan mAP menghitung nilai rata-rata dari *Average Precision* (AP) dari setiap kelas, seperti yang di lihat pada persamaan (4).

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i \quad (4)$$

Di dalam persamaan (4), N adalah representasi dari jumlah kelas di dataset, sendangkan AP_i adalah *Average Precision* untuk kelas ke-i. Nilai mAP merepresentasikan kinerja keseluruhan model dalam mendeteksi berbagai kelas objek, sehingga semakin tinggi nilai mAP, semakin baik kinerja model secara menyeluruh. Dalam evaluasi model deteksi objek, mAP pada nilai ambang IoU tertentu (misalnya, mAP@0.5 untuk IoU $\geq$ 0.5) sering dijadikan standar penilaian.

2.6. Integrasi Model

Setelah proses pelatihan dan evaluasi model selesai, langkah selanjutnya adalah mengintegrasikan model deteksi objek ke dalam aplikasi mobile. Model yang telah dilatih diubah ke format TensorFlow Lite (TFLite) [23], yang dirancang untuk perangkat mobile dengan sumber daya terbatas. Model dalam format TFLite lebih ringan dan efisien sehingga cocok digunakan di perangkat bergerak. Contoh program yang dapat digunakan untuk mengonversi model ke dalam format TFLite dapat dilihat pada kode berikut:

tflite_converter.py

```
import tensorflow as tf

converter = tf.lite.TFLiteConverter.from_saved_model('./training/tflite/saved_model')
converter.allow_custom_ops = True

converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()

with open('./training/tflite/model.tflite', 'wb') as f:
  f.write(tflite_model)
```

3. HASIL DAN PEMBAHASAN

3.1. Pengumpulan Data

Data berupa gambar dikumpulkan dengan mengambil gambar produk secara langsung pada rak toko atau minimarket menggunakan perangkat ponsel. Jumlah gambar yang dikumpulkan adalah sebesar 12473 gambar dengan total 101735 objek dan 128 kelas. Rata-rata objek yang dimiliki untuk setiap gambar adalah 8 objek dan rata-rata variasi kelas untuk setiap gambar adalah sebanyak 2-3 kelas.

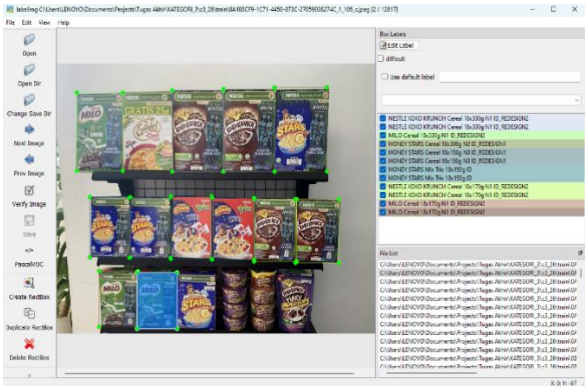
Kriteria gambar yang diambil yakni gambar harus menampilkan informasi spesifik seperti logo, warna, dan ukuran yang dimiliki oleh produk sehingga model dapat membedakan antara satu produk dengan produk lainnya. Selain itu, gambar juga tidak boleh terlalu jauh, terlalu miring, ataupun blur yang dapat menyebabkan model tidak bisa mempelajari fitur yang ada dari objek yang diinginkan. Terakhir, gambar yang diambil harus konsisten

terhadap kriteria pengambilan gambar sehingga model dapat menyimpan informasi dari objek yang diinginkan dengan tepat. Contoh dari gambar yang dikumpulkan dapat dilihat pada Gambar 2.

3.2. Pelabelan Data

Proses pelabelan data dilakukan menggunakan tools bernama "LabelImg". Setiap gambar yang sudah dikumpulkan kemudian diberi anotasi satu persatu dengan menggambar bounding box terhadap setiap objek yang mengandung produk yang akan diberikan label. Proses pelabelan data dapat dilihat pada Gambar 3. Proses ini menghasilkan anotasi berupa file XML dengan format anotasi PASCAL VOC.

GAMBAR 2. Contoh Gambar dari Data Pelatihan

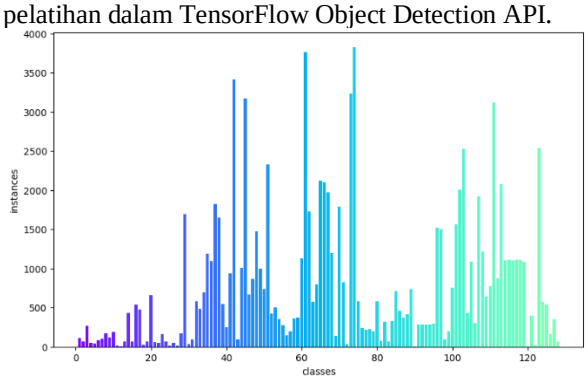


GAMBAR 3. Proses Pelabelan Data dengan LabelImg

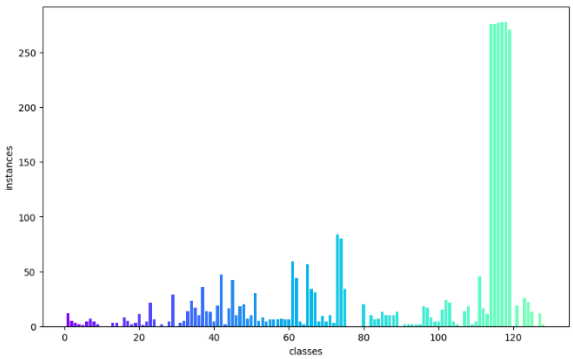
3.3. Permosesan Data

Setelah seluruh gambar berhasil diberi anotasi, dilakukan splitting data sehingga ada data untuk pelatihan dan juga data untuk pengujian. Rasio yang penulis pakai untuk data latih dan data uji adalah sebesar 97:3. Rasio tersebut dipilih karena data yang terkumpul sudah memiliki jumlah yang cukup banyak, yakni sebesar 12473 gambar. Selain itu, mengalokasikan lebih banyak data untuk pelatihan dapat membantu model lebih baik mengenali fitur-fitur produk, terutama jika ada banyak kelas atau kategori produk yang perlu dikenali.

Dari hasil data splitting, didapatkan jumlah gambar dari data pelatihan adalah 12156 gambar dengan jumlah objek sebesar 98634 objek. Distribusi kelas untuk data pelatihan ditunjukkan pada Gambar 4. Sedangkan jumlah gambar dari data uji adalah 317 gambar dengan jumlah objek sebesar 3101 objek. Distribusi kelas untuk data pengujian ditunjukkan pada Gambar 5. Dalam tahap ini, dilakukan pula proses mengubah data yang sebelumnya berupa format gambar dan file XML menjadi format TFRecord agar sesuai dengan format yang diperlukan untuk pelatihan dalam TensorFlow Object Detection API.



GAMBAR 4. Distribusi Kelas dari Data Pelatihan



GAMBAR 5. Distribusi Kelas dari Data Pengujian

3.4. Pelatihan Model

Setelah proses pengumpulan dan pemrosesan data selesai dilakukan, data siap untuk dilakukan pelatihan. Dalam penelitian ini, penulis melakukan tiga kali pelatihan dengan konfigurasi size yang berbeda, yakni 320x320, 640x640, dan 1024x1024. Ketiga model dilatih dengan jumlah step yang sama yakni sebesar 40.000 dan dengan batch size sebesar 16 seperti yang sudah dijelaskan pada subbab 2.4. Model 320x320 dan 640x640 menggunakan satu buah GPU NVIDIA Tesla V100 SXM2 32 GB, sedangkan model 1024x1024 menggunakan 2 buah GPU serupa karena ukuran dari gambar yang besar membutuhkan sumber daya yang besar. Informasi terkait proses pelatihan model dapat dilihat pada Tabel 1.

TABEL 1. Proses Pelatihan Model

Model	Number of Steps	Batch Size	Time per Step (Seconds)	Number of GPUs
SSD MobileNetV2 FPNLite 320x320	40.000	16	0.3	1
SSD MobileNetV2 FPNLite 640x640	40.000	16	2.0	1
SSD MobileNetV2 FPNLite 1024x1024	40.000	16	2.7	2

3.5. Evaluasi Model

Ketiga model yang sudah dilatih selanjutnya dievaluasi dengan data uji yang sudah disiapkan sebelumnya dengan format TFRecord. Data uji terdiri dari 317 gambar dan 3101 objek. Hasil evaluasi dari ketiga model yang sudah dilatih dapat dilihat pada Tabel 2 dan contoh inferensi gambar dapat dilihat pada Gambar 6.

TABEL 2. Hasil Evaluasi Model

Mode l	Localizati on Loss	Classification Loss	Inference Time (Milliseconds)	mAP@0.5IOU
SSD MobileNetV2 FPNLite 320x320	0.27882	0.250912	96.1 ms	0.763618
SSD MobileNetV2 FPNLite 640x640	0.383065	0.248597	216.8 ms	0.79565
SSD MobileNetV2 FPNLite 1024x1024	0.546971	0.254743	539.5 ms	0.808867

Hasil evaluasi menunjukkan bahwa model dengan ukuran gambar 1024x1024 memiliki nilai mAP tertinggi, yaitu 0.808867, diikuti oleh model 640x640 dengan nilai mAP sebesar 0.763618. Model 320x320 memiliki nilai mAP terendah sebesar 0.7204917656. Meskipun model 1024x1024 menghasilkan akurasi yang lebih tinggi, waktu pelatihannya lebih lama dan membutuhkan sumber daya GPU yang lebih besar.

GAMBAR 6. (a) Ground Truth (b) Inferensi Model 320x320 (c) Inferensi Model 640x640 (d) Inferensi Model 1024x1024

Dari hasil di atas, dapat dilihat bahwa terdapat trade-off antara ukuran gambar, kecepatan pelatihan, dan akurasi model. Model 320x320 memberikan kecepatan pelatihan yang jauh lebih cepat dibandingkan model 1024x1024, namun memiliki akurasi yang sedikit lebih rendah. Ini menunjukkan bahwa jika prioritas adalah kecepatan dan sumber daya komputasi terbatas, model 320x320 bisa menjadi pilihan yang lebih efisien. Namun, jika aplikasi memerlukan akurasi yang lebih tinggi, seperti dalam deteksi objek yang lebih rinci, maka model 1024x1024 adalah pilihan yang lebih tepat meskipun membutuhkan lebih banyak waktu dan sumber daya.

3.6. Integrasi Model

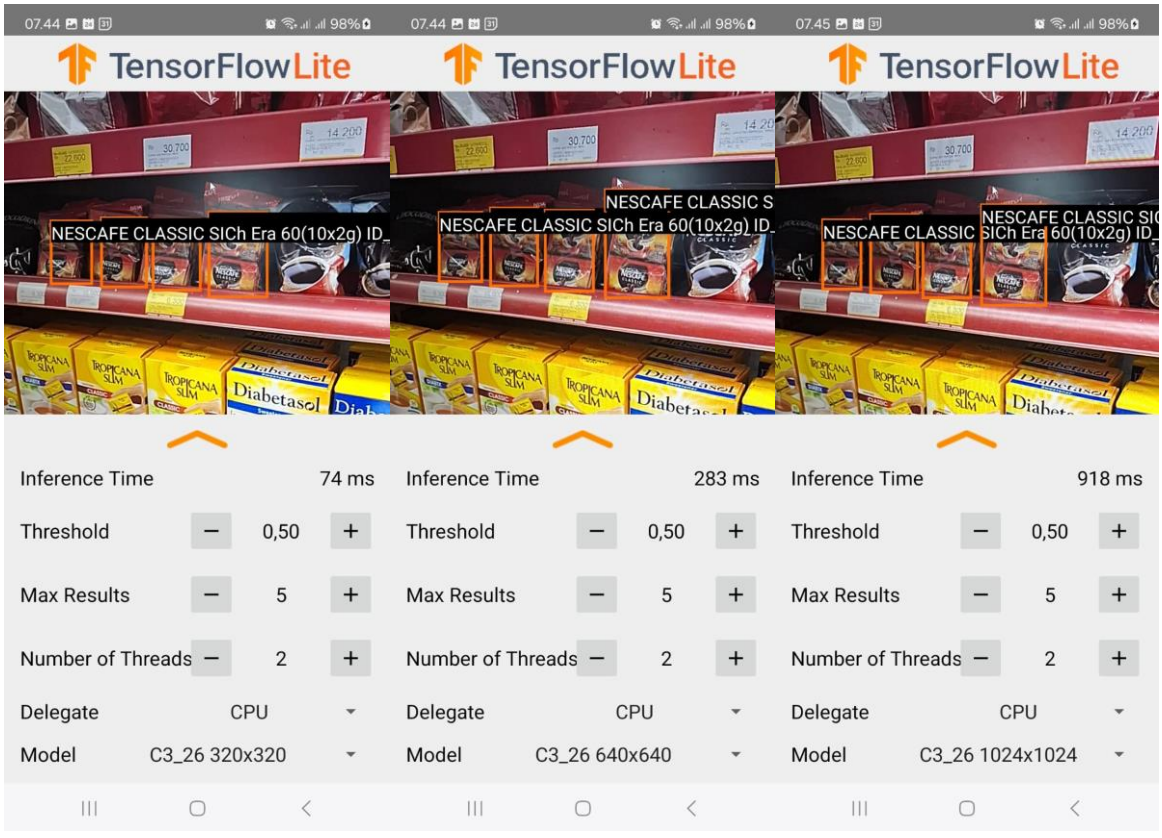
Penelitian ini juga menghitung kemampuan model bekerja pada perangkat bergerak karena sumber daya yang digunakan tentunya juga berbeda. Sebelum diintegrasikan ke dalam perangkat bergerak, model diubah terlebih dahulu ke dalam format TFLite. Integrasi bentuk model ke format TFLite memanfaatkan kuantisasi Dynamic Range Quantization.

Dynamic Range Quantization adalah kuantisasi yang memanfaatkan pengurangan bobot dari model dengan menjaga aktivasi tetap di float32. Kuantisasi ini diharapkan meningkatkan kecepatan dan ukuran dari model namun tetap menjaga akurasi untuk diterapkan pada perangkat dengan sumber daya terbatas seperti perangkat bergerak. Hasil dari pengujian model TFLite dapat dilihat pada Tabel 3 dan contoh inferensi gambar di perangkat bergerak dapat dilihat pada Gambar 7.

TABEL 3. Hasil Evaluasi Model TFLite

Model	Model Size (Megabytes)	Inference Time (Milliseconds)	mAP@0.5IOU
SSD MobileNetV2 FPNLite 320x320	11.9 MB	73 ms	0.6988746936
SSD MobileNetV2 FPNLite 640x640	12.5 MB	283 ms	0.7204917656
SSD MobileNetV2 FPNLite 1024x1024	13.8 MB	918 ms	0.7076417671

Model 320x320 mendapat waktu inferensi yang paling cepat yakni 73 ms, kemudian model 640x640 mendapat waktu inferensi sebesar 283 ms, dan model 1024x1024 mendapat waktu inferensi paling lama, yakni sebesar 918 ms. Perbedaan waktu inferensi disebabkan oleh besar gambar yang diproses. Semakin besar ukuran input gambar, semakin besar pula komputasi yang dibutuhkan untuk menjalankan model.



GAMBAR 7. Contoh Inferensi Gambar di Perangkat Bergerak untuk Setiap Model

Dari segi performa model dalam mendeteksi objek, ketiga model TFLite memiliki kemampuan yang hampir sama namun tidak sebaik model asli yang belum diubah ke dalam format TFLite. Hal tersebut dapat dilihat pada Tabel 2 dan Tabel 3. Perbedaan performa disebabkan oleh kuantisasi yang dilakukan saat mengubah model ke dalam format TFLite mengurangi bobot dari model demi meningkatkan kecepatan inferensi dan mengurangi ukuran model.

Untuk perbandingan ketiga model TFLite, model 640x640 mendapat nilai mAP paling tinggi yakni sebesar 0.7204917656, kemudian model 1024x1024 dengan nilai 0.7076417671, dan yang paling kecil didapatkan oleh model 320x320 dengan nilai sebesar 0.6988746936. Hasil performa yang didapatkan oleh model 640x640 sedikit lebih tinggi dari model 1024x1024 walaupun dengan kecepatan inferensi dan ukuran model yang lebih kecil. Dengan begitu, model 640x640 menunjukkan keseimbangan yang ideal untuk aplikasi mobile yang membutuhkan efisiensi dan akurasi yang baik.

4. KESIMPULAN

Penelitian ini bertujuan untuk mengembangkan dan mengevaluasi model deteksi objek pada produk retail menggunakan TensorFlow 2, dengan fokus pada implementasi di perangkat bergerak. Arsitektur model yang digunakan dalam penelitian ini adalah SSD MobileNetV2 FPNLite dengan tiga variasi ukuran input gambar, yaitu 320x320, 640x640, dan 1024x1024. Hasil penelitian menunjukkan bahwa model dengan ukuran input 640x640 menghasilkan performa terbaik dengan nilai mean Average Precision (mAP) sebesar 0.7204917656, meskipun selisihnya tidak terlalu signifikan dibandingkan dengan model 1024x1024 dan 320x320. Dari segi waktu inferensi, model dengan ukuran input 320x320 memiliki keunggulan signifikan dalam kecepatan, dengan waktu inferensi sebesar 73 ms. Hal ini menjadikan model 320x320 sangat cocok untuk aplikasi yang membutuhkan deteksi objek secara real-time di perangkat bergerak dengan sumber daya komputasi terbatas. Namun, model ini sedikit kalah dalam hal akurasi dibandingkan model 640x640. Penelitian ini dapat memberikan kontribusi bagi pengembangan sistem deteksi objek untuk industri retail, yang bertujuan untuk menjaga ketersediaan barang di rak (on-shelf availability) dan mengurangi risiko out-of-stock. Dengan memanfaatkan model deteksi objek yang cepat dan akurat, sistem ini dapat membantu dalam memantau stok produk secara otomatis dan efisien. Namun, penelitian ini memiliki keterbatasan, antara lain jumlah dataset yang terbatas dan tidak balance. Penelitian lanjutan dapat memperluas cakupan dengan meningkatkan variasi dataset, augmentasi, dan arsitektur lain serta melakukan pengujian pada perangkat dengan spesifikasi yang berbeda untuk melihat dampak performa di lingkungan yang lebih beragam.

REFERENSI

[1] Har, Loh & Rashid, Umi Kartini & Te Chuan, Lee & Yin Xia, Loh. (2022). Revolution of Retail Industry: From Perspective of Retail 1.0 to 4.0. Procedia Computer Science. 200. 10.1016/j.procs.2022.01.362.

[2] Klimchenia, L.. (2024). Retail transformation concepts development under the influence of industrial revolutions. Vestnik Universiteta. 121-129. 10.26425/1816-4277-2024-6-121-129.

[3] Rai, Prakruthi & Nanjundan, Preethi & George, Jossy. (2024). Enhancing Industrial Operations through AI-Driven Decision-Making in the Era of Industry 4.0. 10.1201/9781003432319-3..

[4] Andaur, Juan & Ruz, Gonzalo & Goycoolea, Marcos. (2021). Predicting Out-of-Stock Using Machine Learning: An Application in a Retail Packaged Foods Manufacturing Company. Electronics. 10. 2787. 10.3390/electronics10222787.

[5] Šikić F, Kalafatić Z, Subašić M, Lončarić S. (2024). Enhanced Out-of-Stock Detection in Retail Shelf Images Based on Deep Learning. Sensors; 24(2):693. https://doi.org/10.3390/s2402069.

[6] Afram, G. (2020). A study on how to improve On-shelf availability : With deep learning (Dissertation). Retrieved from https://urn.kb.se/resolve?urn=urn:nbn:se:miun:diva-40006.

[7] al Azzo, Fadwa & Taqi, Arwa & Milanova, Mariofanna. (2018). Human Related-Health Actions Detection using Android Camera based on TensorFlow Object Detection API. International Journal of Advanced Computer Science and Applications. 9. 10.14569/IJACSA.2018.091002.

[8] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Kaiser, L., Kudlur, M., Levenberg, J., & Zheng, X. (2015). TensorFlow : Large-Scale Machine Learning on Heterogeneous Distributed Systems. arXiv:1603.04467v2 [cs.DC]

[9] Vasilakopoulos, G. P., (2024). An Automatic Differentiation Algorithm for Recursive Functions and its Implementation in TensorFlow. Thesis. National and Kapodistrian University of Athens.

[10] Rezaei Nasab, Ali & Dashti, Maedeh & Shahin, Mojtaba & Zahedi, Mansooreh & Khalajzadeh, Hourieh & Arora, Chetan & Liang, Peng. (2024). Fairness Concerns in App Reviews: A Study on AI-based Mobile Apps. 10.48550/arXiv.2401.08097.

[11] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot Multibox Detector. European Conference on Computer Vision (ECCV), 21-37.

[12] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4510-4520.

[13] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861.

[14] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2117-2125.

[15] Goldsborough, P. (2016). A Tour of TensorFlow. ArXiv, abs/1610.01178.

[16] LabelImg, "LabelImg: LabelImg is a graphical image annotation tool and label object bounding boxes in images", Available: https://github.com/heartexlabs/labelImg, accessed on 22 October 2024.

[17] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision, 88(2), 303–338. https://doi.org/10.1007/s11263-009-0275-4..

[18] TensorFlow, "TFRecord and tf.train.Example", Available: https://www.tensorflow.org/tutorials/load_data/tfrecord, accessed on 22 October 2024.

[19] TensorFlow, "TensorFlow Object Detection API," Available: https://github.com/tensorflow/models/tree/master/research/object_detection, accessed on 22 October 2024.

[20] TensorFlow, "Training a TensorFlow Object Detection Model" Available: https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html, accessed on 22 Oktober 2024.

[21] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., & Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. European Conference on Computer Vision (ECCV), 740–755. https://doi.org/10.1007/978-3-319-10602-1_48.

[22] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.

[23] TensorFlow, "TensorFlow Lite", Available: https://www.tensorflow.org/lite, accessed on 23 October 2024.