

# Implementasi Algoritma YOLO pada Aplikasi Pendeteksi Senjata Tajam di Android

Christopher Nathanel Liunanda<sup>1</sup>, Silvia Rostianingsih<sup>2</sup>, Anita Nathania Purbowo<sup>3</sup>  
Program Studi Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra  
Surabaya, Indonesia

+6289606072909, +6281654024662, +628185713363  
christo.liunanda@gmail.com<sup>1</sup>, silvia@petra.ac.id<sup>2</sup>, anitaforpetra@gmail.com<sup>3</sup>

## ABSTRAK

Kemajuan computational power di perangkat Smartphone sudah melebihi kebutuhan tradisional handphone. Object detection membutuhkan computational power yang tinggi. Tensorflow Lite dapat digunakan untuk menjalankan model yang dengan cepat dan mudah ke perangkat mobile. Jaringan YOLO dipakai karena kinerja yang lebih cepat dan akurat dari jaringan lain. Object yang akan diidentifikasi adalah senjata tajam pisau dan golok. Senjata tajam ini dipilih karena potensi aplikasi di dunia nyata. Aplikasi dapat digunakan untuk melakukan deteksi senjata bagi taxi online.

Model yang dilatih adalah YOLOv2-tiny, YOLOv3-tiny dan YOLOv3. Transfer Learning dilakukan kepada model-model tersebut agar YOLO dengan Darknet agar dapat mendeteksi senjata tajam yang diinginkan. Kemudian Darknet dikonversi ke Tensorflow Lite. Pengujian model dilakukan dengan melihat beberapa metric akurasi standard seperti precision, recall, mAP, dan average IoU. Model dengan kinerja terbaik akan dipasang di aplikasi Android untuk melakukan deteksi objek senjata tajam.

Hasil pengujian menunjukkan bahwa kinerja model sangat bergantung dari jenis jaringan, jumlah dataset, dan bentuk dataset. Jaringan YOLOv2-tiny menghasilkan rata-rata mAP dan average IoU paling rendah yaitu 55% dan 35%. Akurasi terakhir model Tensorflow Lite Android adalah 72.7% untuk YOLOv3 dan 63.6% untuk YOLOv3-tiny. Jaringan YOLOv3-tiny cocok digunakan untuk deteksi real-time karena waktu inferensi yang cepat (0.9 detik).

**Kata Kunci:** Object Detection, Darkflow, YOLO, Tensorflow, Tensorflow Lite.

## ABSTRACT

*Advances in computing power have exceeded traditional smartphone needs. Object detection requires high computational power. Tensorflow Lite can be used to run models quickly and easily to mobile devices. The YOLO network was used because of faster and more accurate performance than other similar networks. The object to be identified are bladed weapons that are knives and machetes. Bladed weapons are selected because of potential applications in the real world.*

*The trained models are YOLOv2-tiny, YOLOv3-tiny and YOLOv3. Transfer Learning is done to these models with Darknet so that YOLO can detect the desired weapon. Darknet model will be converted to Tensorflow Lite. Model testing is done by looking at some standard accuracy metrics such as precision, recall, mAP, and the average IoU. The model with the best performance will be*

*installed in the Android application to detect bladed weapon objects knives and machetes.*

*The test results show that the performance of the model is very dependent on the type of network, the number of datasets, and the shape of the dataset. YOLOv2-tiny produces the worst result with mAP of 55% and average IoU of 35%. The final accuracy for Tensorflow Lite Android model are 72.7% for YOLOv3 and 63.6% for YOLOv3-tiny. The YOLOv3-tiny network is suitable for real-time detection because of fast inference time (0.9 seconds).*

**Keywords:** Object Detection, Darkflow, YOLO, Tensorflow, Tensorflow Lite..

## 1. INTRODUCTION

Kemajuan computational power di perangkat Smartphone sudah melebihi kebutuhan tradisional handphone. Dengan multi core processors, dedicated GPUs, serta RAM yang mencapai Gigabytes memungkinkan perangkat Smartphone untuk menjalankan task yang menuntut computational power yang lebih tinggi [4].

Deep Learning Object Detection membutuhkan computational power yang tinggi [4] sehingga dibutuhkan sebuah framework yang dapat mengimplementasikan task tersebut dengan optimal. Tensorflow Lite. Tensorflow Lite memungkinkan developer untuk menjalankan model yang dihasilkan dari Tensorflow dengan cepat dan mudah ke perangkat mobile.

You Only Look Once (YOLO) merupakan sebuah metode yang dihasilkan oleh Joseph Redmon untuk melakukan Object Detection. YOLO mampu untuk melakukan object detection secara real time. Jika dibandingkan dengan sistem object detection real time yang lain, YOLO memiliki mAP dan FPS yang lebih tinggi [7]. Dengan dasar model YOLO ini, Transfer Learning dapat dilakukan terhadap model agar model dapat mendeteksi dan mengklasifikasikan objek baru.

Objek baru yang akan diklasifikasikan adalah senjata tajam. Senjata tajam dipilih karena kemudahan untuk mendapatkan dataset dan potensi digunakan aplikasi ini di dunia nyata. Salah satu potensi penggunaan pendeteksi senjata adalah sebagai early warning atau panic button system bagi taxi online. Implementasi aplikasi akan fokus ke perangkat Android. Android memiliki 2.5 milyar pengguna [3] dan memiliki 76% dari market share OS Mobile. Hal inilah yang menjadi dasar alasan pemilihan OS Android sebagai target pembuatan aplikasi.

Sketsa ini akan menghasilkan sebuah aplikasi berbasis Android yang dapat melakukan deteksi dan klasifikasi objek senjata secara real time.

## 2. LANDASAN TEORI

### 2.1 Tinjauan Pustaka

#### 2.1.1 Convolutional Neural Network

Convolutional Neural Network adalah sejenis algoritma Deep Learning yang menerima input gambar, memberikan kepentingan seperti weight dan biases ke bermacam-macam objek dan aspek di dalam gambar agar dapat membedakan antara objek. Convolutional Neural Network tidak membutuhkan preprocessing yang banyak jika dibandingkan dengan algoritma klasifikasi lainnya. Convolutional Neural Network dengan training yang banyak dapat mempelajari filter/karakteristik.

Convolutional Neural Network memiliki Convolution Layer, Pooling Layer, Fully Connected Layer. Convolution Layer bertugas untuk mengambil feature dari input image. Pooling Layer bertugas untuk mengurangi jumlah parameter ketika gambar terlalu besar. Pada Fully Connected Layer, Matrix di flattened menjadi vector lalu di masukkan ke sebuah Fully Connected Layer seperti Neural Network

#### 2.1.2 You Only Look Once (YOLO)

You Only Look Once merupakan object detection network yang dibuat oleh Joseph Redmon di tahun 2016. Cara kerja YOLO cukup sederhana, Hanya sebuah Convolutional Neural Network saja yang akan memprediksi beberapa bounding boxes dan probabilitas kelas tiap box.

YOLO menerima sebuah input image yang dibagi menjadi grid sebesar  $S \times S$  yang dikirimkan ke sebuah neural network untuk membuat bounding box dan class prediction.

Setiap grid cell memprediksi  $B$  bounding box dan confidence score dari tiap kotak. Confidence score inilah yang merefleksikan seberapa tingkat kepercayaan model bahwa objek di dalam kotak berupa objek yang diprediksikan. YOLO menilai confidence sebagai  $\Pr(\text{Object}) * \text{IOU}_{\text{truth}}$  (Intersection of Union)

Tiap bounding box terdiri dari 5 prediksi:  $x, y, w, h$ . Koordinat  $(x, y)$  merepresentasikan pusat dari kotak relatif dengan batas dari grid cell. Lebar (width) dan tinggi (height) adalah prediksi relatif dari seluruh gambar.

Tiap grid cell memprediksi conditional class probabilities  $C$ ,  $\Pr(\text{Class}|\text{Object})$ . Probabilitas ini dikoneksikan ke grid cell yang ada objeknya.

#### 2.1.3 Tensorflow

Tensorflow merupakan library Machine Learning bersifat open source yang di develop oleh Google di tahun 2015. Tensorflow menggunakan data graphs untuk membuat model. Data graphs ini terdiri dari beberapa node yang merepresentasikan sebuah operasi matematik, dan memiliki koneksi antara node yang disebut dengan tensor

Model yang dihasilkan oleh Tensorflow dapat di export dengan format Flatbuffer (.tflite) dan dijalankan di mobile device dengan menggunakan Tensorflow Lite. Tensorflow Lite yang dikenalkan pada tahun 2017 oleh Google ini memiliki performa yang lebih baik dan binary size yang lebih kecil karena kernels yang lebih di optimized, pre-fused activations, serta jumlah dependencies yang lebih kecil.

#### 2.1.4 Object Detection Model Evaluation Metric

Penilaian sebuah model yang melakukan object detection dilakukan dengan melihat perbandingan beberapa metric berbeda.

Metric-metric inilah yang akan digunakan sebagai basis perbandingan performa antara model-model berbeda.

Intersection over Union (IoU) atau nama lainnya Jaccard Index merupakan metric evaluasi yang menghitung persamaan antara bounding box prediksi model dengan ground truth bounding box (bounding box yang dianotasikan ke gambar di test dataset). IoU memiliki nilai antara 0 dengan 1 dimana nilai akan semakin besar jika kedua bounding box semakin dekat posisinya.

Angka IoU ini akan dipakai sebagai sebagai basis melakukan klasifikasi dimana angka IoU diatas treshold tertentu akan dinilai sebagai prediksi positif dan angka IoU dibawah treshold tertentu akan dinilai sebagai prediksi negatif. Prediksi akan dibagi menjadi klasifikasi-klasifikasi berbeda yaitu True Positives (TP), False Negatives (FN), dan False Positives (FP).

Prediksi diklasifikasi sebagai True Positives ketika terdapat object di gambar dan model berhasil memprediksi bounding box dengan IoU diatas treshold tertentu.

Prediksi diklasifikasi sebagai False Positives ketika terdapat object di gambar tetapi model melakukan prediksi bounding box dengan nilai IoU dibawah treshold atau Object tidak ada digambar tetapi model mendeteksi terdapat objek.

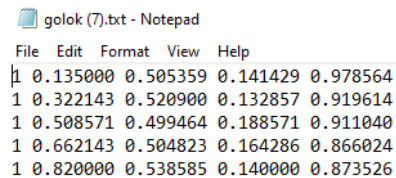
Prediksi diklasifikasi sebagai False Negative ketika terdapat object di gambar dan model gagal melakukan deteksi terdapat objek gambar.

Metric penilaian berikutnya adalah precision, recall, dan F1-score. Precision merupakan probabilitas bounding box prediksi mengikuti ground truth bounding boxes.

Recall merupakan tingkat True Positive. Mengukur probabilitas ground truth dideteksi dengan akurat.

F1-Score merupakan weighted average dari Precision dan Recall.

Recall yang tinggi dengan precision yang rendah berarti semua ground truth boxes berhasil terdeteksi tetapi deteksi tidak benar



```
golok (7).txt - Notepad
File Edit Format View Help
1 0.135000 0.505359 0.141429 0.978564
1 0.322143 0.520900 0.132857 0.919614
1 0.508571 0.499464 0.188571 0.911040
1 0.662143 0.504823 0.164286 0.866024
1 0.820000 0.538585 0.140000 0.873526
```

Gambar 1. Contoh label.txt

(banyak false positives). Recall yang rendah dengan precision yang tinggi berarti tiap kotak hasil prediksi benar tetapi banyak ground truth boxes yang terlewatkan (banyak false negatives). Precision dan Recall yang tinggi menunjukkan bahwa banyak object ground truth yang terdeteksi dengan benar.

AP (Average Precision) merupakan metric yang mencakup precision dan recall dan merangkum kurva Precision-Recall dengan merata-ratakan precision di seluruh nilai recall dari 0 hingga 1. Formula Average Precision adalah

MAP(Mean Average Precision) adalah rata-rata dari Average Precision. Jika dataset memiliki  $N$  class categories, maka mAP berupa rata-rata AP.

## 2.2 Tinjauan Studi

Pada studi pertama [1] ini penelitian menggunakan beberapa model berbeda yaitu network R-CNN, SSD Inception, SSD

Mobilenet, dan Tiny YOLOv2. Model dilatih dengan dataset yang dikumpulkan menggunakan Instagram-scraper. Instagram Scraper adalah aplikasi untuk melakukan scraping terhadap Instagram untuk mendapat gambar. Model dilatih untuk mengenali objek post-it notes. Percobaan menunjukkan bahwa R-CNN terlalu lambat untuk dipakai sebagai real time object detector. SSD Inception memiliki mAP yang tinggi tetapi inference time terlambat. Deep Learning Based Model memiliki peningkatan precision 26.1% dan recall 117.7% jika dibanding dengan Heuristic based.

Percobaan lain[5] menggunakan YOLO network untuk melakukan deteksi defect dari permukaan strip steel dan mendeteksi region yang cacat dari strip steel. Hasil dari penelitian menunjukkan bahwa YOLO network berhasil melakukan deteksi dengan hasil akurasi yang lebih tinggi dari shallow neural network dan SVM classifier

### 3. DESAIN SISTEM

Proses dalam pembuatan sistem meliputi pengolahan dataset, transfer learning, konversi dari model Darknet ke TensorflowLite, desain user interface, dan proses melakukan inferensi pada gambar di aplikasi Android. Model yang dilatih dengan transfer learning adalah model YOLOv3, YOLOv3-tiny, dan YOLOv2-tiny. Model dengan performa terbaik akan dimuat ke aplikasi Android. Proses inferensi tidak membutuhkan internet karena proses hanya mengandalkan kemampuan hardware Android.

#### 3.1 Pengolahan Dataset

Dataset diambil dari beberapa sumber berbeda. Sumber pertama adalah gambar pisau dari penelitian Mاتيolanski[], dataset ini memiliki 9340 contoh negatif (gambar tidak ada objek pisau) dan 3559 contoh positif (gambar ada objek pisau). Dataset kedua berupa screenshot dari action movie yang diambil secara manual dengan screen grabbing tool milik Microsoft. Dataset ketiga berupa hasil scraping gambar di Google Images dan Duckduckgo. Dataset keempat berupa video youtube yang diubah menjadi gambar dengan bantuan library opencv.

Dataset yang dikumpulkan perlu diberi label agar dapat digunakan di proses transfer learning. Format label ini adalah file .txt yang berstruktur satu baris per objek, tiap baris berisi class x\_center dan y\_center width height format, koordinat kotak harus memiliki format xywh yang di normalisasikan, dan angka class dimulai dari 0. Contoh dari isi label ini dapat dilihat di Gambar 1

#### 3.2 Transfer Learning

Proses ini dilakukan menggunakan Darknet yang dikembangkan oleh Joseph Redmon lalu dikembangkan oleh Alexey Bochkovskiy. Proses Transfer Learning di Darknet membutuhkan data file, cfg file, dan pre-trained weight. Data file berisi lokasi gambar yang akan digunakan untuk train dan test, nama class yang di training, dan lokasi penyimpanan weight hasil training. CFG file berisi bentuk jaringan yang digunakan untuk training, pre-trained weight berupa weight yang dilatih untuk mengenali objek baru.

Agar model dapat mendeteksi objek baru, perlu konfigurasi terhadap cfg file. Perubahan cfg file untuk YOLOv3, YOLOv3-tiny, dan YOLOv2-tiny mengikuti proses yang sama tetapi ada beberapa hal spesifik yang berbeda. Hal umum yang diganti di cfg file adalah mengubah batch menjadi 64 dan subdivision menjadi

16, max\_batches diubah menjadi 6000 dan steps menjadi 80% dan 90% dari max\_batches. Hal spesifik yang diubah tergantung dari jenis model. Model YOLOv3 dan YOLOv3-tiny mengubah baris classes di tiap [yolo] layer menjadi jumlah class yang ingin dideteksi. Filter di tiap [convolutional] layer sebelum [yolo] layer juga diganti dengan rumus  $\text{filter} = (\text{classes} + 5) \times 3$ . Pada YOLOv2-tiny, hal spesifik yang diganti adalah mengubah baris class di [region] layer menjadi jumlah class yang ingin dideteksi. Kemudian mengubah filters di layer [convolutional] sebelum [region] dengan rumus  $\text{filters} = (\text{classes} + 5) \times 5$ .

Setelah semua proses itu telah dilakukan, transfer learning dapat dimulai. Hasil dari proses akan disimpan di folder yang tertulis di file data. Darknet akan menyimpan hasil model tiap 1000 iterasi.

#### 3.3 Proses Konversi ke TensorflowLite

Model yang dihasilkan Darknet masih harus dikonversi menjadi format TensorflowLite agar dapat dimuat di aplikasi Android. Proses diawali dengan melakukan perubahan dari .weight (Darkflow) menjadi protobuff (.pb) milik Tensorflow. Perubahan ini dilakukan dengan menggunakan project milik hunglc007. Setelah model berubah menjadi format .pb, maka model dapat dikonversi langsung menjadi flatbuffer .tflite milik Tensorflow Lite. Perubahan ini menggunakan Tensorflow Lite converter yang disediakan bersama dengan library Tensorflow. Model dengan format .tflite inilah yang dimuat di aplikasi Android untuk melakukan inference.

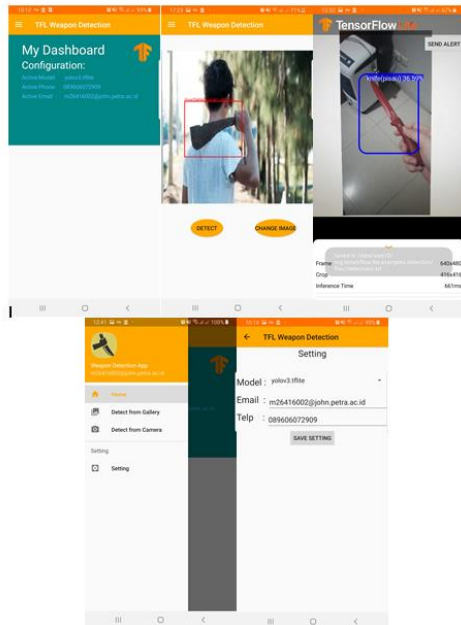
#### 3.4 Proses Inferensi di Android

Proses inferensi di Android dimulai dengan menyimpan model hasil transfer learning ke aplikasi Android. Model kemudian disimpan dalam memory dan akan dipanggil ketika melakukan inferensi. Proses inferensi dilakukan dengan memanggil function yang disediakan oleh library Tensorflow Lite yang bernama runForMultipleInferences. Function ini menerima 2 parameter, sebuah bytearray yang berisi gambar hasil konversi dari bitmap dan sebuah hashmap yang berisi float array yang memiliki bentuk yang sama dengan output model. Hasil dari inferensi akan disimpan ke hashmap yang dipassing ke parameter function. Hashmap ini kemudian diproses untuk mendapatkan posisi xy tengah objek, lebar dan tinggi kotak objek, confidence score adanya objek di kotak itu, dan confidence tiap class di kotak itu. Hasil ini diproses untuk mendapatkan kotak objek, memangkas objek yang *confidence score* di bawah *threshold* dan mengidentifikasi class yang ada pada gambar itu.

Setelah proses sebelumnya dilakukan, hasil masih harus dilakukan *non max suppression* terlebih dahulu untuk memangkas kotak-kotak yang mengacu pada objek yang sama. Proses NMS ini dilakukan dengan mengumpulkan deteksi yang classnya sama, mengambil sebuah acuan yang memiliki tingkat confidence paling tinggi, dan membuang kotak yang memiliki nilai *Intersection of Union* diatas *threshold* tertentu. Jika nilai IoU antara kotak acuan dan kotak lain diatas nilai *threshold* maka kotak lain ini akan dinilai mengacu pada objek yang sama dengan kotak acuan sehingga kotak lain ini perlu dibuang. Proses NMS dilakukan hingga semua class di hasil sudah diproses.

Hasil dari proses inferensi akan diberikan ke activity lain untuk menggambar kotak deteksi objek di layar Android.

### 3.5 User Interface



**Gambar 2. Tampilan Interface Aplikasi**

Gambar 2 menunjukkan *interface* pada aplikasi Weapon Detection. Halaman awal yang akan dibuat adalah halaman dashboard yang berisikan model yang sedang aktif, nomor telpon tujuan, dan alamat email tujuan untuk kepentingan pengiriman alert.

Tersedia sebuah *nav segment* yang bertugas untuk memberi jalan bagi user untuk berpindah dari activity satu ke lain. Pada *nav segment* ini terdapat 4 activity. Activity pertama adalah dashboard, activity kedua adalah Detect from Gallery, activity ketiga adalah Detect from Camera, dan activity keempat adalah setting.

Pada activity kedua terdapat sebuah *imageView* yang berisi gambar yang ingin di deteksi, 2 tombol button bernama *btnDetect* dan *btnChangeImage*. *btnDetect* digunakan untuk melakukan proses inferensi pada gambar di *imageView* sedangkan *btnChangeImage* digunakan untuk mengubah gambar di *imageView* sesuai pilihan user dari gallery.

Activity ketiga memiliki 2 tombol bernama *btnPhoto* dan *btnCamera*. *btnPhoto* memungkinkan user untuk melakukan hal yang sama dengan activity kedua tetapi gambar diambil dengan menggunakan photo camera Android. *btnCamera* akan melakukan deteksi objek secara real-time dengan menggunakan kamera belakang Android. Objek yang dideteksi akan ditampilkan dilayar. User juga dapat menekan tombol di video untuk mengirim alert ke nomor telpon dan alamat email yang sudah dipasang.

Pada activity keempat user dapat menentukan model mana yang akan aktif, nomor telpon tujuan alert, dan email address tujuan alert.

## 4. PENGUJIAN SISTEM

Pengujian akan dilakukan dengan melihat kinerja model Darknet mAP dan average IoU dari 3 iterasi terakhir tiap model yang dilatih dengan 3 dataset berbeda. Iterasi yang menghasilkan nilai

paling baik akan digunakan untuk melakukan testing selanjutnya yaitu testing dengan menggunakan beberapa gambar contoh. Hasil model terbaik akan dipakai untuk dikonversi ke Tensorflow Lite. Kemudian model Tensorflow Lite ini akan dibandingkan lagi dengan gambar yang sama dengan yang diuji di Darkflow untuk melihat perbedaan performa. Model Tensorflow Lite ini kemudian diuji waktu inferensi dengan beberapa perangkat Android berbeda dan dilihat apakah ada perubahan akurasi.

### 4.1 Kinerja Model Darknet.

Pengujian akan melihat mAP dan Average IoU dari 3 iterasi terakhir model. Kinerja model dapat dilihat di Tabel 1, 2, dan 3.

**Tabel 1. Kinerja Model Darknet YOLOv3**

Dataset	Iterasi model	mAP	Average IoU
Dataset 1	4000	62.75%	63.26%
	5000	60.68%	64.52%
	6000	64.92%	65.06%
Dataset 2	4000	43.58%	54.06%
	5000	49.13%	63.00%
	6000	48.69%	64.55%
Dataset 3	4000	59.78%	51.50%
	5000	59.50%	61.28%
	6000	57.16%	64.09%

**Tabel 2. Kinerja Model Darknet YOLOv3-tiny**

Dataset	Iterasi model	mAP	Average IoU
Dataset 1	4000	59.62%	36.21%
	5000	61.37%	47.56%
	6000	60.35%	60.35%
Dataset 2	4000	52.48%	29.71%
	5000	56.49%	35.36%
	6000	57.52%	57.53%
Dataset 3	4000	53.54%	35.07%
	5000	49.52%	30.42%
	6000	58.94%	58.33%

**Tabel 3. Kinerja Model Darknet YOLOv2-tiny**

Dataset	Iterasi model	mAP	Average IoU
Dataset 1	4000	58.59%	38.31%
	5000	55.90%	41.68%
	6000	62.34%	28.46%
Dataset 2	4000	55.44%	30.88%
	5000	57.64%	35.97%
	6000	55.32%	40.00%
Dataset 3	4000	49.44%	32.80%

	5000	53.63%	31.88%
	6000	50.63%	32.37%

Berdasarkan hasil kinerja YOLOv3 dari Tabel 1, YOLOv3 dataset 1 memilih iterasi 6000, YOLOv3 dataset 2 memilih iterasi 5000, dan YOLOv3 dataset 3 memilih iterasi 5000.

Berdasarkan hasil kinerja YOLOv3-tiny dari Tabel 2, YOLOv3-tiny dataset 1 memilih iterasi 6000, YOLOv3-tiny dataset 2 memilih iterasi 6000, dan YOLOv3-tiny memilih iterasi 6000.

Berdasarkan hasil kinerja YOLOv2-tiny dari Tabel 3, YOLOv2-tiny dataset 1 memilih iterasi 4000, YOLOv2-tiny dataset 2 memilih iterasi 5000, dan YOLOv3-tiny memilih iterasi 5000.

## 4.2 Akurasi Deteksi model Darknet pada Gambar Contoh.

Model terbaik dari 4.1 diuji dengan 6 gambar pisau dan 5 gambar golok di situasi yang berbeda. Hasil dari percobaan ini dapat dilihat di Tabel 4

**Tabel 4. Hasil Deteksi menggunakan model Darknet**

Dataset	Jenis Model	Akurasi
Dataset 1	YOLOv3	7/11(63.6%)
	YOLOv3-tiny	9/11(81.8%)
	YOLOv2-tiny	4/11(36.4%)
Dataset 2	YOLOv3	6/11(54.5%)
	YOLOv3-tiny	7/11(63.6%)
	YOLOv2-tiny	4/11(36.4%)
Dataset 3	YOLOv3	9/11(81.8%)
	YOLOv3-tiny	7/11(63.6%)
	YOLOv2-tiny	4/11(36.4%)

Berdasarkan hasil dari Tabel 4 maka model yang digunakan adalah YOLOv3 dengan dataset 3 dan YOLOv3-tiny dengan dataset 1. YOLOv2-tiny tidak digunakan karena hasil performa yang rendah.

## 4.3 Akurasi Deteksi model Tensorflow Lite pada Gambar Contoh.

Model terbaik dari 4.2 akan diuji dengan gambar yang sama dengan sub-bab 4.2. Hasil dari percobaan ini dapat dilihat di Tabel 5

**Tabel 5. Hasil Deteksi menggunakan model Tensorflow Lite**

Jenis Model	Akurasi
YOLOv3	7/11(63.6%)
YOLOv3-tiny	6/11(54.5%)

Terjadi penurunan performa baik bagi YOLOv3 maupun YOLOv3-tiny. YOLOv3 berubah dari 9/11(81.8%) menjadi

7/11(63.6%) sedangkan YOLOv3-tiny berubah dari 9/11(81.8%) menjadi 6/11(54.5%)

## 4.4 Kinerja Model di Android.

Pada segmen ini akan diuji kinerja model ketika berjalan di perangkat Android. Terdapat 2 hal yang dilihat, hal pertama adalah kinerja model di aplikasi dan kecepatan melakukan inferensi. Perangkat yang lambat dapat menyebabkan waktu inferensi menjadi lama sehingga menyebabkan deteksi tidak real-time.

**Tabel 6. Kecepatan waktu inferensi di Android**

Nama Perangkat	Nama Model	Rata-rata waktu Inferensi
Samsung Galaxy S9	YOLOv3	7660ms
	YOLOv3-tiny	764.5ms
Samsung Tab S2	YOLOv3	11211ms
	YOLOv3-tiny	1116.2ms
Samsung Galaxy Note 8	YOLOv3	12641ms
	YOLOv3-tiny	990.2ms

Tabel 7 adalah penilaian akurasi model dari gambar yang dapat dilihat di Tabel 8.


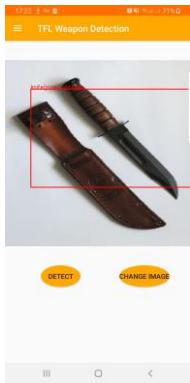
**Tabel 7. Akurasi Model di Android**

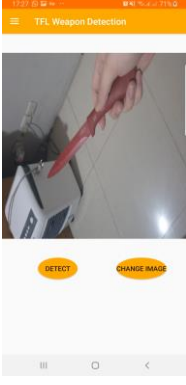
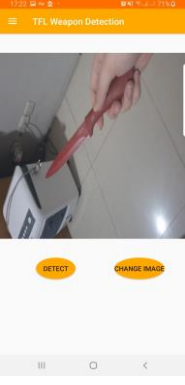
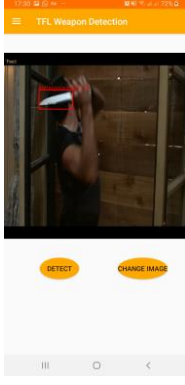
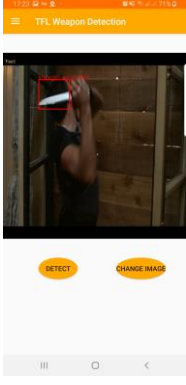
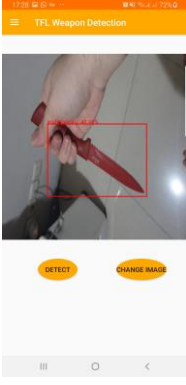
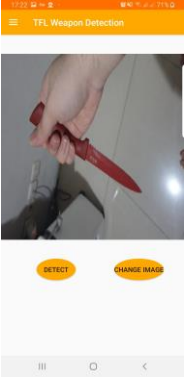






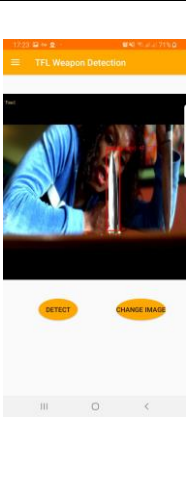
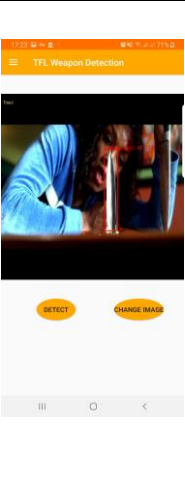
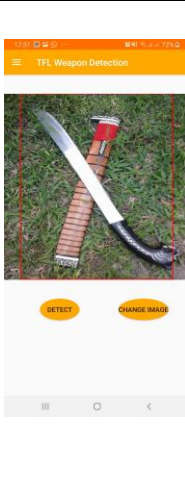

Jenis Model	Akurasi
YOLOv3	8/11 (72.7%)
YOLOv3-tiny	7/11 (63.6%)

Tabel 8 adalah hasil pengenalan Objek di Aplikasi Android. Di tabel ini dapat terlihat hasil melakukan deteksi objek baik menggunakan YOLOv3 maupun menggunakan YOLOv3-tiny di Android.

Pada gambar-gambar ini model akan menggambar sebuah kotak pada gambar dan mengklasifikasikan objek pada gambar ke klasifikasi senjata tajam baik pisau maupun golok.

**Tabel 8. Hasil Pengenalan Objek di Aplikasi Android**

Nama Gambar	YOLOv3	YOLOv3-tiny
Pisau 1		

Nama Gambar	YOLOv3	YOLOv3-tiny	Nama Gambar	YOLOv3	YOLOv3-tiny
Pisau 2			Pisau 6		
Pisau 3			Golok 1		
Pisau 4			Golok 2		
Pisau 5			Golok 3		

Nama Gambar	YOLOv3	YOLOv3-tiny
Golok 4		
Golok 5		

## 5. KESIMPULAN DAN SARAN

Berdasarkan hasil pengamatan dan pengujian pada sistem, dapat disimpulkan beberapa hal sebagai berikut:

- Berdasarkan hasil pengujian dapat disimpulkan bahwa kinerja model YOLOv2-tiny memiliki mAP rata-rata sebesar 53.71% dan average IoU sebesar 52.38%. YOLOv3 yang memiliki rata-rata mAP sebesar 56.24% dan average IoU sebesar 61.26%. YOLOv3-tiny memiliki rata-rata mAP sebesar 56.64% dan average IoU sebesar 43.39%
- Berdasarkan hasil pengujian dapat disimpulkan bahwa penambahan training dataset dengan 900 gambar negatives (gambar yang tidak berisi objek senjata pisau maupun golok) dapat meningkatkan kinerja model bagi YOLOv3 dan YOLOv3-tiny. Peningkatan mAP YOLOv3 dari 49% mAP menjadi 59% mAP sedangkan untuk YOLOv3-tiny terjadi peningkatan dari 57% mAP menjadi 58% mAP
- Berdasarkan hasil pengujian dapat disimpulkan bahwa konversi langsung dari model Darkflow menuju model Tensorflow Lite dapat menyebabkan penurunan performa Model. YOLOv3 yang bisa mendeteksi 9/11 gambar testing (81%) menurun menjadi 8/11(72.7%). Kinerja YOLOv3-tiny lebih juga berkurang dari mendeteksi 9/11 (81.5%) menjadi 7/11 (63.6%).
- Berdasarkan hasil pengujian dapat disimpulkan bahwa model YOLOv3-tiny cocok untuk dijalankan di perangkat

Android karena waktu inferensi yang pendek. Rata-rata waktu inferensi bagi YOLOv3-tiny adalah 956ms atau 0.9 detik sedangkan model YOLOv3 memiliki rata-rata waktu inferensi 10504ms atau 10 detik

- Berdasarkan hasil pengujian dapat disimpulkan bahwa model masih rentan melakukan deteksi salah ke objek yang mirip senjata tetapi bukan senjata, rentan gagal deteksi terdapat objek blur, dan objek senjata yang memiliki refleksi cahaya yang tinggi

Dengan adanya kesimpulan, ada beberapa hal yang dapat dijadikan sebagai saran dalam pengembangan untuk selanjutnya antara lain:

- Penambahan dataset untuk training model. Gambar golok yang jelas dan dipegang orang cukup susah untuk ditemukan, sehingga terdapat kesulitan mencari dataset golok.
- Pelatihan model dapat ditambah iterasinya agar menghasilkan model yang lebih baik.
- Konversi dari Darkflow menuju Tensorflow Lite dapat dikembangkan lagi, agar tidak ada penurunan kinerja model

## 6. DAFTAR PUSTAKA

- Alsing, Oscar. "Mobile Object Detection using TensorFlow Lite and Transfer Learning." 2018.
- Bochkovskiy, A. (n.d.). AlexeyAB (Alexey) • GitHub. Retrieved July 13, 2020, from <https://github.com/AlexeyAB>
- Alsing, Oscar. "Mobile Object Detection using TensorFlow Lite and Transfer Learning." 2018.
- Brandom, R. (2019, May 7). There are now 2.5 billion active Android devices. Retrieved November 27, 2019, from <https://www.theverge.com/2019/5/7/18528297/google-io-2019-android-devices-play-store-total-number-statistic-keynote>
- Ignatov, Andrey, et al. "AI Benchmark: Running Deep Neural Networks on Android Smartphones." Lecture Notes in Computer Science Computer Vision – ECCV 2018 Workshops, 2019, pp. 288–314., doi:10.1007/978-3-030-11021-5\_19
- Li, Jianguyun, et al. "Real-Time Detection of Steel Strip Surface Defects Based on Improved YOLO Detection Network." IFAC-PapersOnLine, vol. 51, no. 21, 2018, pp. 76–81., doi:10.1016/j.ifacol.2018.09.412
- Matiolański, A., Maksimova, A., & Dziech, A. (2015). CCTV object detection with fuzzy classification and image enhancement. Multimedia Tools and Applications, 75(17), 10513-10528. doi:10.1007/s11042-015-2697-z
- Redmon, Joseph, et al. "You Only Look Once: Unified, Real-Time Object Detection." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, doi:10.1109/cvpr.2016.91.
- Yosinki, Jason, et al. "How Transferable Are Features in Deep Neural Networks?" Advances in Neural Information Processing Systems 27, 20