

DEVELOPMENT OF DEEP LEARNING MODEL USING CONVOLUTIONAL NEURAL NETWORK FOR FOOD RECOGNITION

การพัฒนาโมเดลการเรียนรู้เชิงลึกโดยใช้โครงข่ายประสาทแบบconvโอลูชันในการ
จำจำประเภทของอาหาร

นักศึกษา

นายธนาภูรพัค ศรีไสว 623040255-1
นายวปรัชญ์ โลมาอันธร 623040486-2

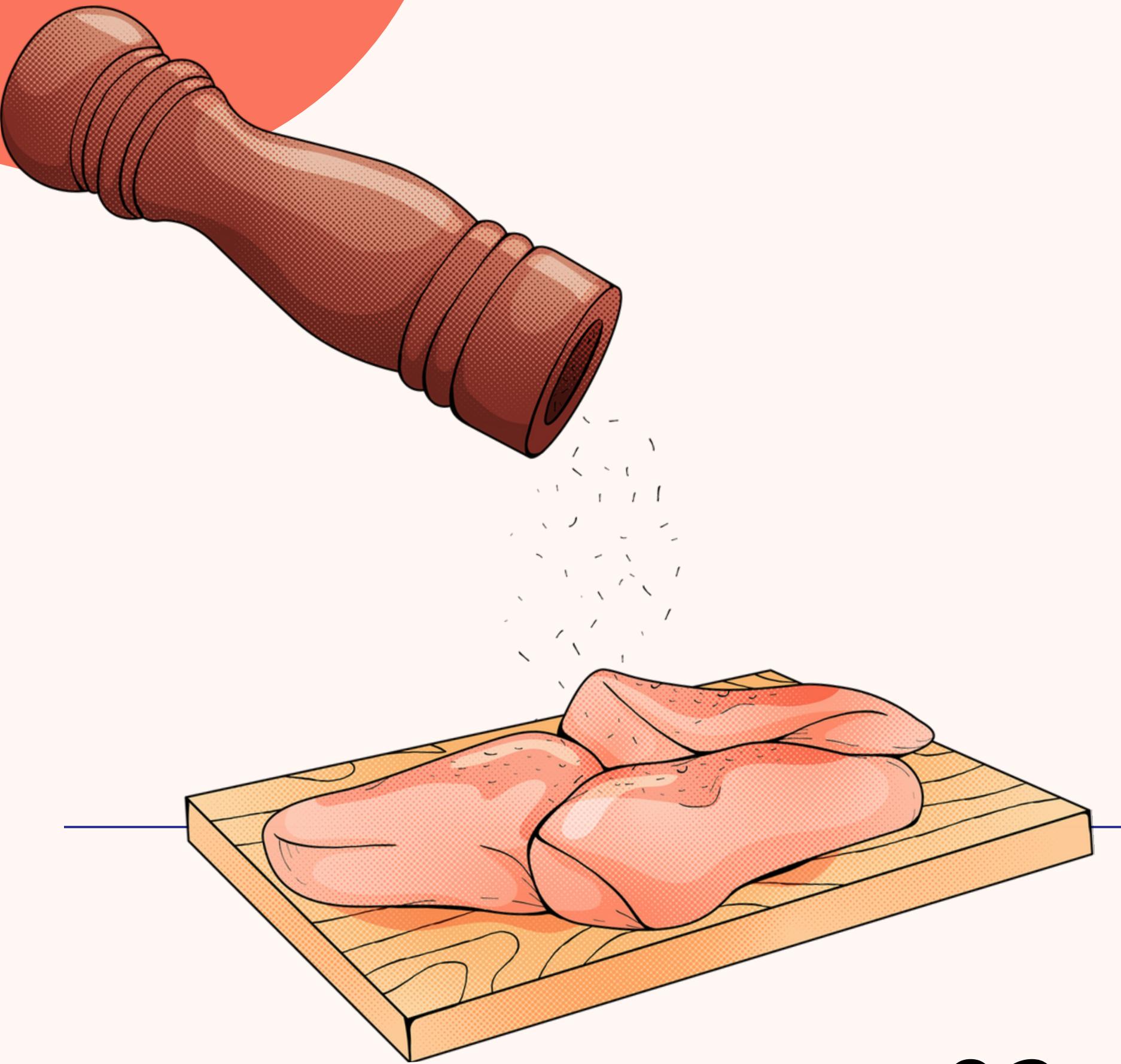
อาจารย์ที่ปรึกษา

อาจารย์กานวิชญ์ หาญพินิจศักดิ์
ผู้ช่วยศาสตราจารย์ ภาณุพงษ์ วนจันทร์
ผู้ช่วยศาสตราจารย์ชวิศ ศรีจันทร์



AGENDAS

- Introduction
- Method
- Result
- Conclusion



Introduction

Motivation

- Nutrition problems in elderly
- Health promotion with technology

Objective

- To design and develop CNN model that estimate food volume
- To design and develop Artificial Intelligence algorithms for food data processing to interpret nutritional values.



Method



Preparing data for
object detection
training

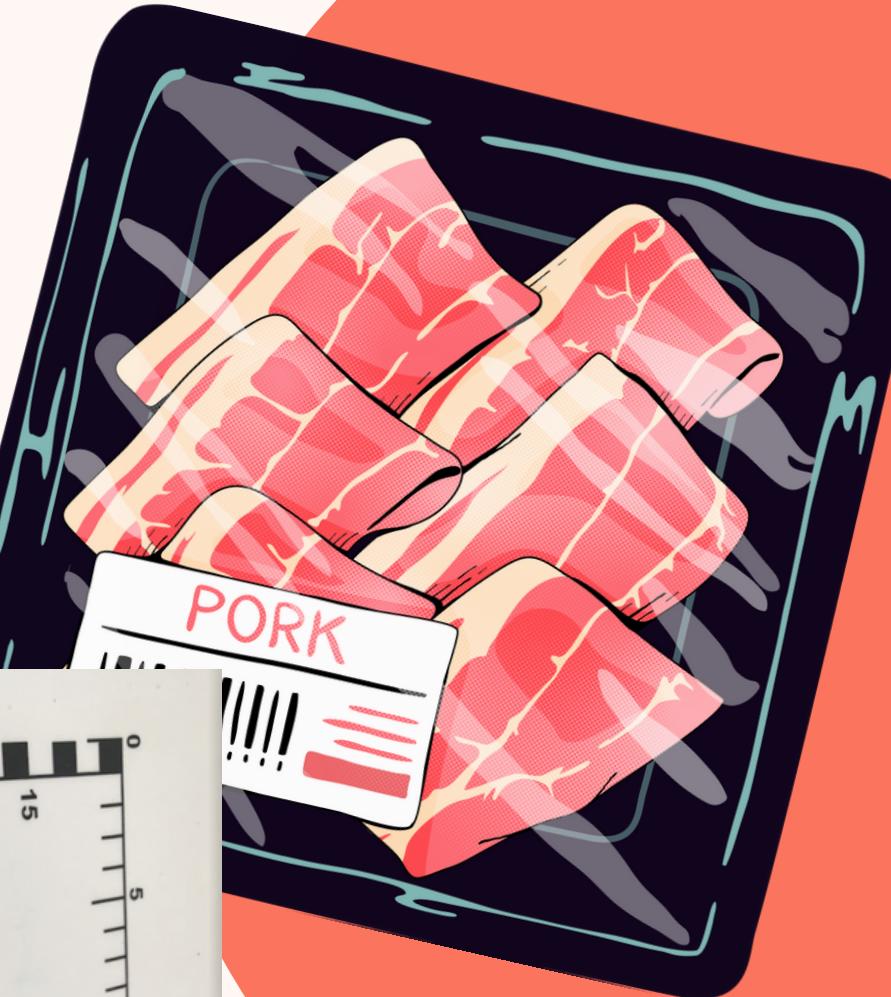
Train object detection
model with
TensorFlow Lite Model
Maker library

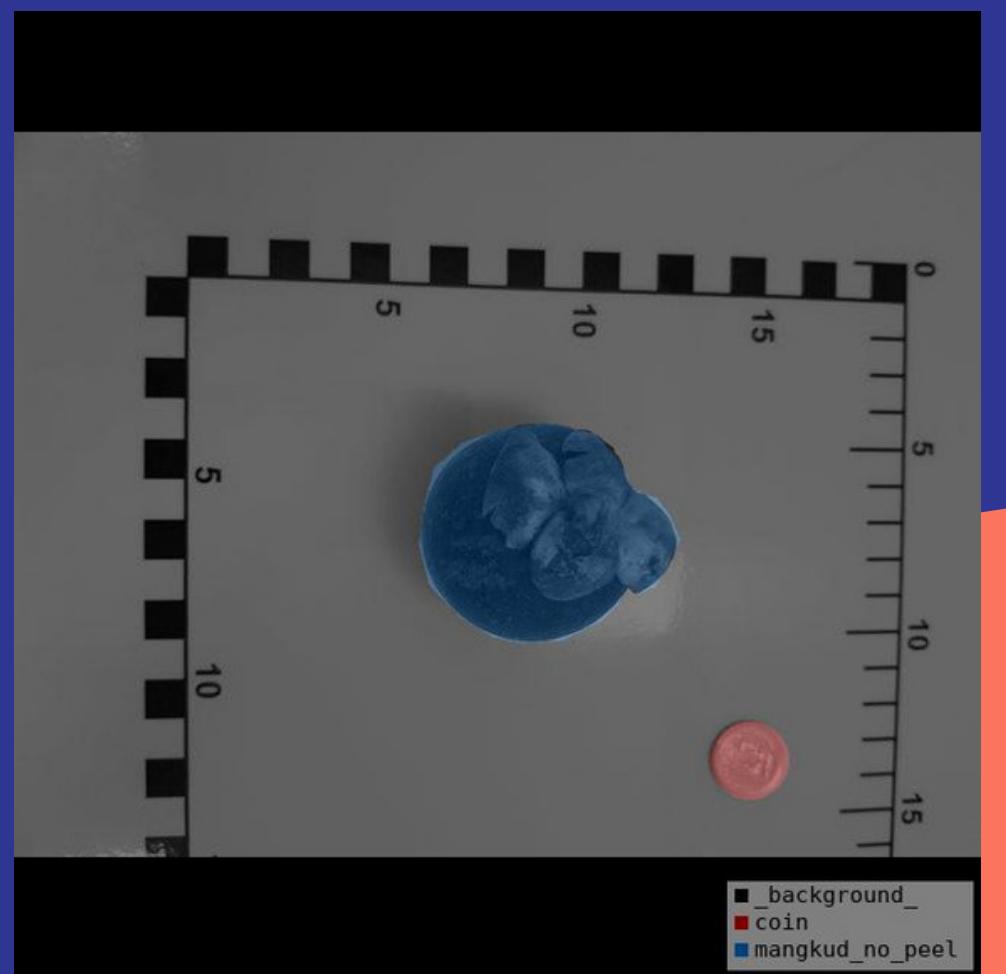
Deploy on mobile

Collecting datasets

Datasets were provided by Department of Nutrition,
Faculty of Public Health, Mahidol University.

Example





```
kl2.py - project - Visual Studio Code [Administrator]
kl2.py kl2.py x
kl2.py
 1 # Python program to read json file
 2 # From https://www.geeksforgeeks.org/read-json-file-using-python/
 3
 4 import glob
 5 import json
 6 import math
 7 import re
 8 from pascal_voc_writer import Writer
 9
10 # path edit in path for all example --> 'D:/Desktop/project/chompoo/'
11 path_for_all = 'D:/Desktop/project/mangkud/'
12 path = path_for_all + '*.json'
13
14 for filename in glob.glob(path):
15     filename = filename.replace('\\', '/')
16     print(filename)
17
18 # Opening JSON file
19 f = open(filename)
20
21 # returns JSON object as a dictionary
22 data = json.load(f)
23
24 bboxa = [] # List for receive json dictionary
25 labelstr = []
26 xval1 = []
27 xval2 = []
28 yval1 = []
29 yval2 = []
30
31 # Iterating through the json list
```

Preparing data for object detection training

Labeling dataset & convert it to Pascal VOC format for training

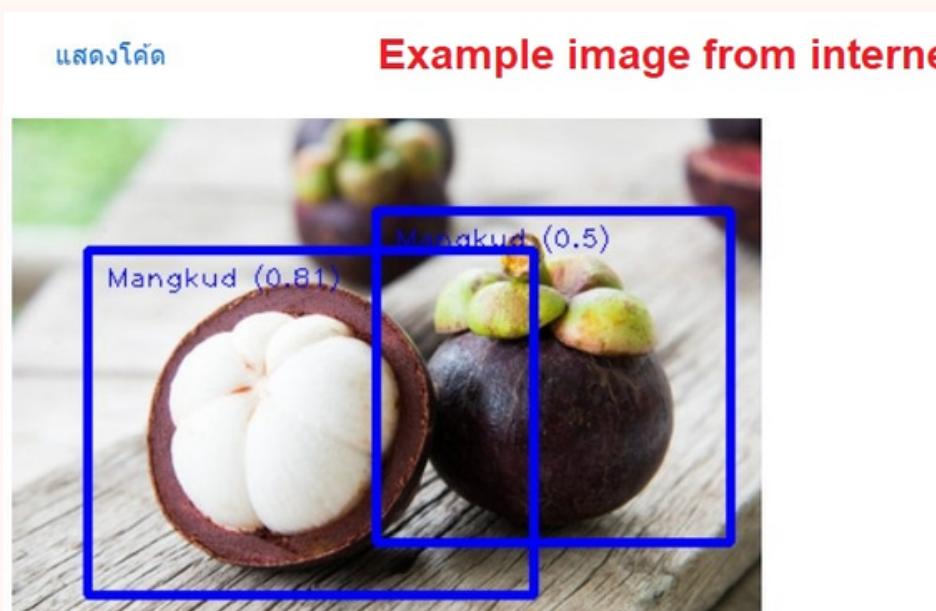
Total class: 7

- Coin
- Farang
- Chompoo
- Longong
- Lumyai
- Mangkud
- Orange

Train object detection model using TensorFlow Lite Model Maker library

Model architecture: efficientdet-lite2

Mean Average Precision (mAP): 0.7548603

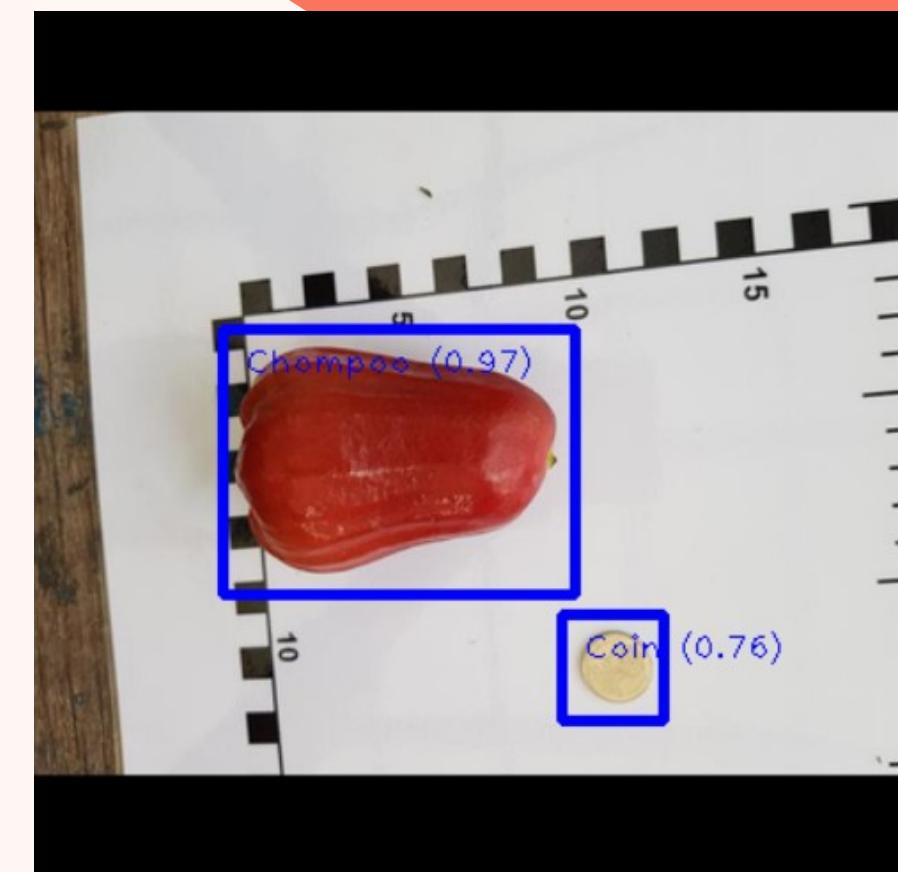
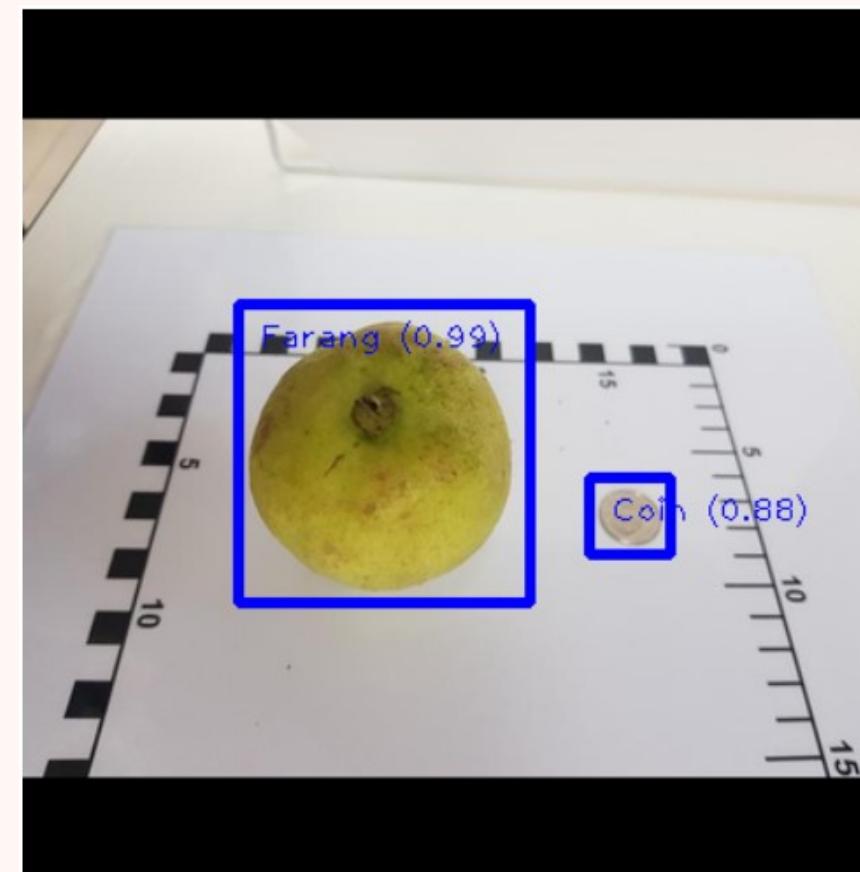
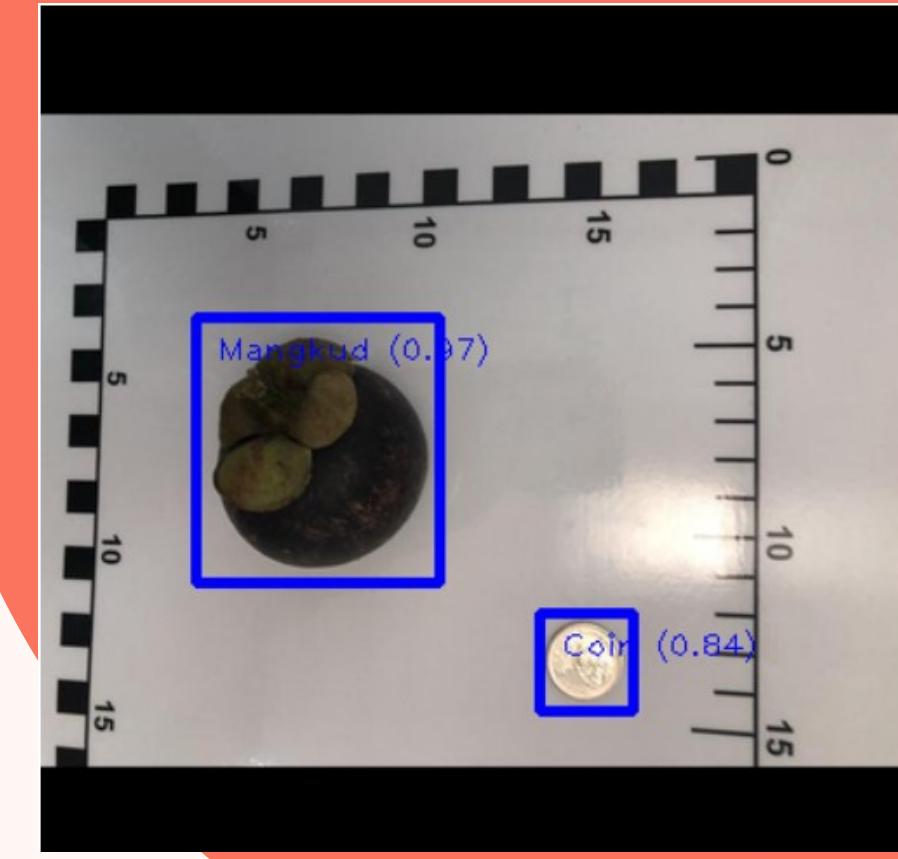
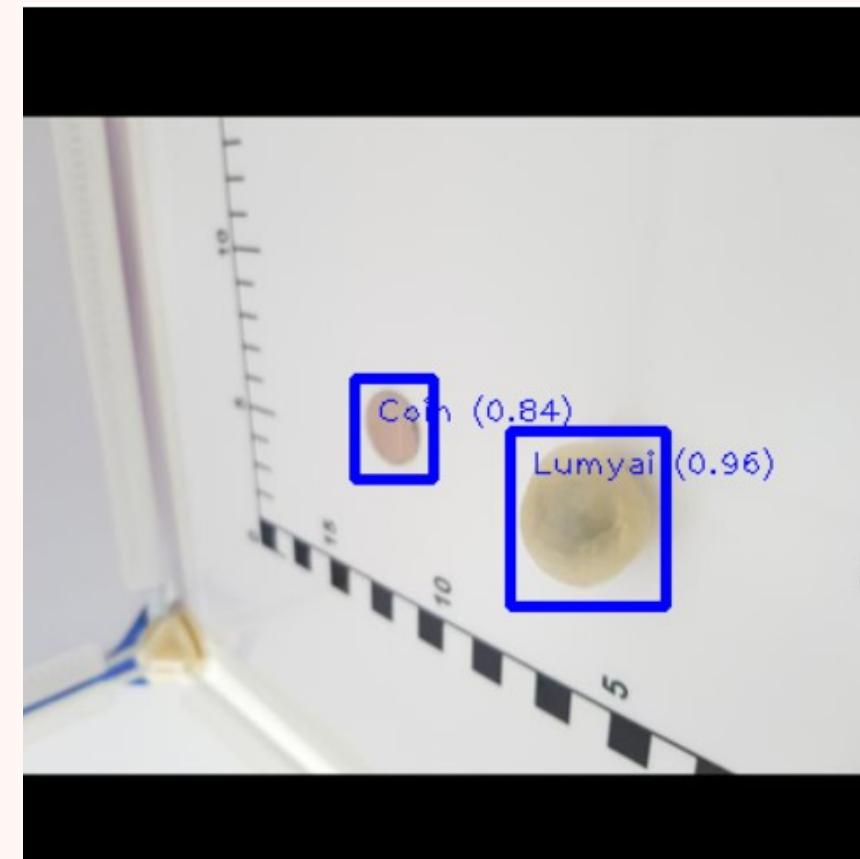


mAP	0.7548603
AP Coin	0.6019163
AP Farang	0.81222296
AP Chom poo	0.7998368
AP Longong	0.7456985
AP Lumyai	0.7570465
AP Mangkud	0.81058097
AP Orange	0.7567197

```
[14] model.evaluate(val_data)
[15] 5/5 [=====]
{'AP': 0.7548603,
'AP50': 0.99427366,
'AP75': 0.9521453,
'APs': -1.0,
'APm': 0.6076802,
'API': 0.74990517,
'ARmax1': 0.7935428,
'ARmax10': 0.8140485,
'ARmax100': 0.8147725,
'ARs': -1.0,
'ARm': 0.65028167,
'ARI': 0.815715,
'AP_Coin': 0.6019163,
'AP_Farang': 0.81222296,
'AP_Chom poo': 0.7998368,
'AP_Longong': 0.7456985,
'AP_Lumyai': 0.7570465,
'AP_Mangkud': 0.81058097,
'AP_Orange': 0.7567197}
```

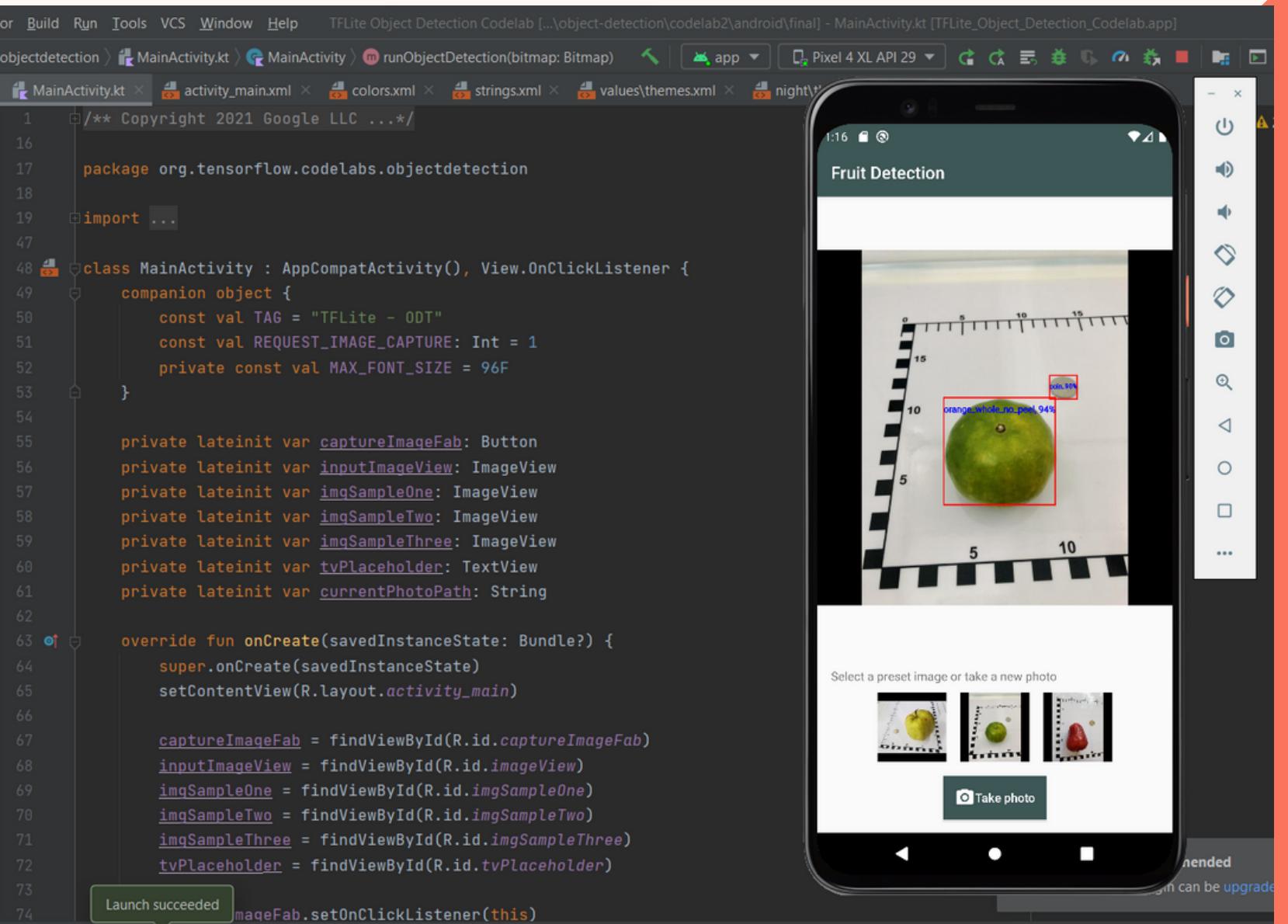
Train object detection model using TensorFlow Lite Model Maker library

Result example



Deploy on mobile

- Developed with Kotlin language on Android Studio.
- Can take a picture and reply with the name of a fruit.



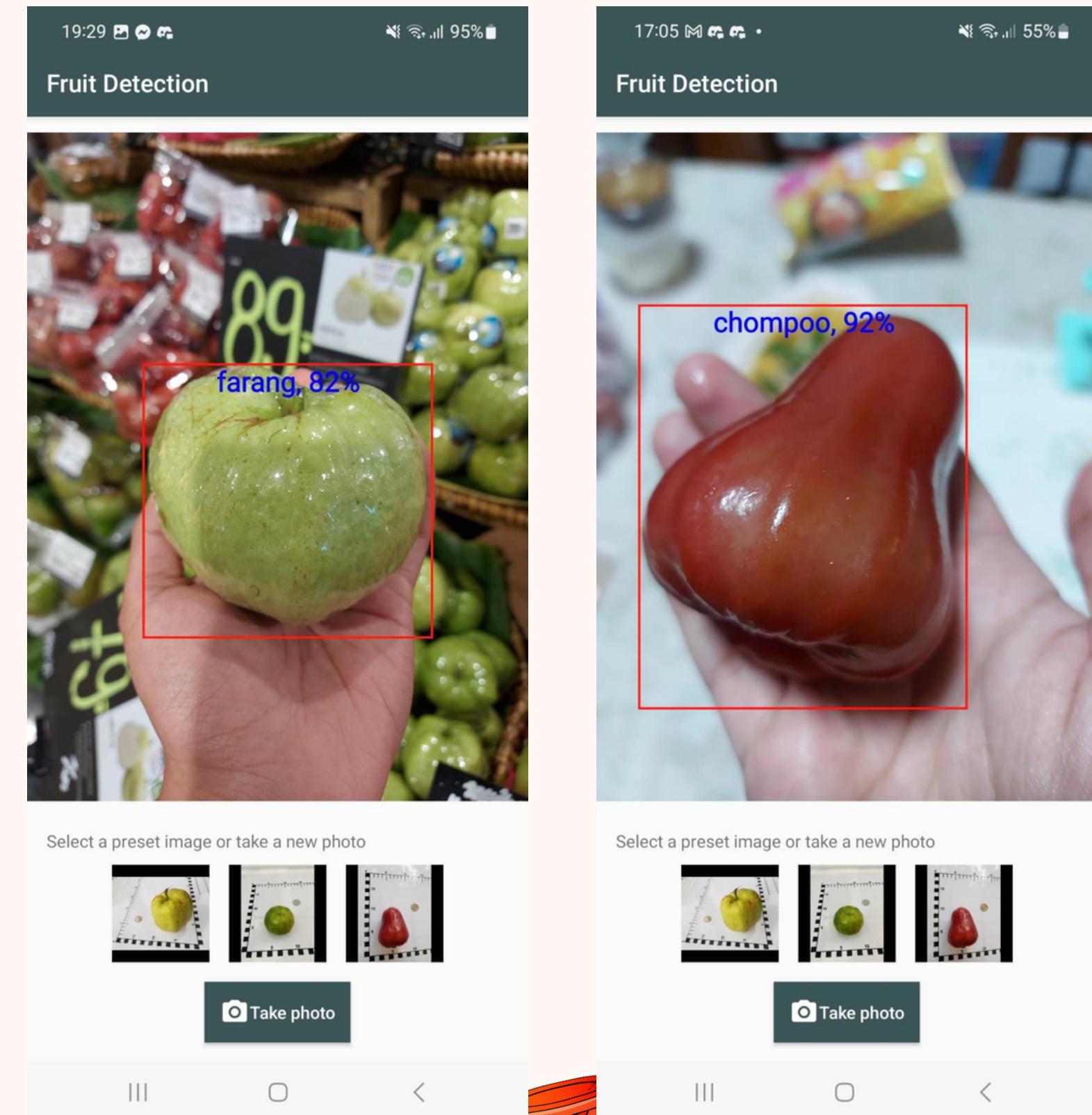
The screenshot shows the Android Studio interface with the code editor open to `MainActivity.kt`. The code is written in Kotlin and implements a fruit detection application. The app's user interface, visible in the emulator, displays a green fruit (an orange) with a bounding box and a confidence score of 94%. Below the image is a camera preview with three sample photos and a "Take photo" button.

```
ktor Build Run Tools VCS Window Help TFLite Object Detection Codelab [...] - MainActivity.kt [TFLite_Object_Detection_Codelab.app]
objectdetection > MainActivity.kt > MainActivity > runObjectDetection(bitmap: Bitmap)
MainActivity.kt x activity_main.xml x colors.xml x strings.xml x values\themes.xml x night\themes.xml
1  /**
2  * Copyright 2021 Google LLC
3  */
4
5 package org.tensorflow.codelabs.objectdetection
6
7 import ...
8
9 class MainActivity : AppCompatActivity(), View.OnClickListener {
10     companion object {
11         const val TAG = "TFLITE - ODT"
12         const val REQUEST_IMAGE_CAPTURE: Int = 1
13         private const val MAX_FONT_SIZE = 96F
14     }
15
16     private lateinit var captureImageFab: Button
17     private lateinit var inputImageView: ImageView
18     private lateinit var imgSampleOne: ImageView
19     private lateinit var imgSampleTwo: ImageView
20     private lateinit var imgSampleThree: ImageView
21     private lateinit var tvPlaceholder: TextView
22     private lateinit var currentPhotoPath: String
23
24     override fun onCreate(savedInstanceState: Bundle?) {
25         super.onCreate(savedInstanceState)
26         setContentView(R.layout.activity_main)
27
28         captureImageFab = findViewById(R.id.captureImageFab)
29         inputImageView = findViewById(R.id.imageView)
30         imgSampleOne = findViewById(R.id.imgSampleOne)
31         imgSampleTwo = findViewById(R.id.imgSampleTwo)
32         imgSampleThree = findViewById(R.id.imgSampleThree)
33         tvPlaceholder = findViewById(R.id.tvPlaceholder)
34
35         captureImageFab.setOnClickListener { imageFab.setOnClickListener(this) }
36     }
37
38     private fun processImage(bitmap: Bitmap) {
39         val result = detect(bitmap)
40
41         if (result != null) {
42             val fruitName = result[0].label
43             val confidence = result[0].score
44
45             val boundingBox = result[0].boundingBox
46             val width = boundingBox.width
47             val height = boundingBox.height
48             val left = boundingBox.left
49             val top = boundingBox.top
50
51             val text = "$fruitName ${confidence * 100} %"
52             val paint = Paint()
53             paint.color = Color.BLUE
54             paint.style = Paint.Style.FILL
55             paint.setTextSize(MAX_FONT_SIZE)
56
57             val textWidth = paint.measureText(text)
58             val textHeight = paint.descent() - paint.ascent()
59
60             val textX = left + (width - textWidth) / 2
61             val textY = top - textHeight
62
63             val rect = Rect(left, top, left + width, top + height)
64             val rectF = RectF(left.toFloat(), top.toFloat(), left + width.toFloat(), top + height.toFloat())
65
66             val canvas = inputImageView.canvas
67             val bitmap = Bitmap.createBitmap(canvas.width, canvas.height, Bitmap.Config.ARGB_8888)
68             val canvas2 = Canvas(bitmap)
69             val paint2 = Paint()
70             paint2.color = Color.RED
71             paint2.setStyle(Paint.Style.STROKE)
72             paint2.setStrokeWidth(2f)
73
74             canvas2.drawBitmap(bitmap, 0f, 0f, paint2)
75             canvas2.drawRect(rectF, paint)
76             canvas2.drawText(text, textX, textY, paint)
77
78             inputImageView.setImageBitmap(bitmap)
79         }
80     }
81
82     private fun detect(bitmap: Bitmap): List<Object>? {
83         return null
84     }
85
86     private fun handleImage(bitmap: Bitmap) {
87         processImage(bitmap)
88     }
89
90     private fun handleImageCapture(result: Uri) {
91         val bitmap = BitmapFactory.decodeFile(result.path)
92         handleImage(bitmap)
93     }
94
95     private fun handleImageSelection(result: Uri) {
96         val bitmap = BitmapFactory.decodeFile(result.path)
97         handleImage(bitmap)
98     }
99
100    private fun handleImageCaptureError(error: String) {
101        Log.e(TAG, "Error capturing image: $error")
102    }
103
104    private fun handleImageSelectionError(error: String) {
105        Log.e(TAG, "Error selecting image: $error")
106    }
107
108    private fun handleImageDecodeError(error: String) {
109        Log.e(TAG, "Error decoding image: $error")
110    }
111
112    private fun handleImageProcessError(error: String) {
113        Log.e(TAG, "Error processing image: $error")
114    }
115
116    private fun handleImageSaveError(error: String) {
117        Log.e(TAG, "Error saving image: $error")
118    }
119
120    private fun handleImageUploadError(error: String) {
121        Log.e(TAG, "Error uploading image: $error")
122    }
123
124    private fun handleImageDeleteError(error: String) {
125        Log.e(TAG, "Error deleting image: $error")
126    }
127
128    private fun handleImageLoadError(error: String) {
129        Log.e(TAG, "Error loading image: $error")
130    }
131
132    private fun handleImageSaveSuccess(path: String) {
133        Log.d(TAG, "Image saved to $path")
134    }
135
136    private fun handleImageUploadSuccess(url: String) {
137        Log.d(TAG, "Image uploaded to $url")
138    }
139
140    private fun handleImageDeleteSuccess(path: String) {
141        Log.d(TAG, "Image deleted from $path")
142    }
143
144    private fun handleImageLoadSuccess(bitmap: Bitmap) {
145        Log.d(TAG, "Image loaded successfully")
146    }
147
148    private fun handleImageSaveError(error: String) {
149        Log.e(TAG, "Error saving image: $error")
150    }
151
152    private fun handleImageUploadError(error: String) {
153        Log.e(TAG, "Error uploading image: $error")
154    }
155
156    private fun handleImageDeleteError(error: String) {
157        Log.e(TAG, "Error deleting image: $error")
158    }
159
160    private fun handleImageLoadError(error: String) {
161        Log.e(TAG, "Error loading image: $error")
162    }
163
164    private fun handleImageSaveSuccess(path: String) {
165        Log.d(TAG, "Image saved to $path")
166    }
167
168    private fun handleImageUploadSuccess(url: String) {
169        Log.d(TAG, "Image uploaded to $url")
170    }
171
172    private fun handleImageDeleteSuccess(path: String) {
173        Log.d(TAG, "Image deleted from $path")
174    }
175
176    private fun handleImageLoadSuccess(bitmap: Bitmap) {
177        Log.d(TAG, "Image loaded successfully")
178    }
179
180    private fun handleImageSaveError(error: String) {
181        Log.e(TAG, "Error saving image: $error")
182    }
183
184    private fun handleImageUploadError(error: String) {
185        Log.e(TAG, "Error uploading image: $error")
186    }
187
188    private fun handleImageDeleteError(error: String) {
189        Log.e(TAG, "Error deleting image: $error")
190    }
191
192    private fun handleImageLoadError(error: String) {
193        Log.e(TAG, "Error loading image: $error")
194    }
195
196    private fun handleImageSaveSuccess(path: String) {
197        Log.d(TAG, "Image saved to $path")
198    }
199
200    private fun handleImageUploadSuccess(url: String) {
201        Log.d(TAG, "Image uploaded to $url")
202    }
203
204    private fun handleImageDeleteSuccess(path: String) {
205        Log.d(TAG, "Image deleted from $path")
206    }
207
208    private fun handleImageLoadSuccess(bitmap: Bitmap) {
209        Log.d(TAG, "Image loaded successfully")
210    }
211
212    private fun handleImageSaveError(error: String) {
213        Log.e(TAG, "Error saving image: $error")
214    }
215
216    private fun handleImageUploadError(error: String) {
217        Log.e(TAG, "Error uploading image: $error")
218    }
219
220    private fun handleImageDeleteError(error: String) {
221        Log.e(TAG, "Error deleting image: $error")
222    }
223
224    private fun handleImageLoadError(error: String) {
225        Log.e(TAG, "Error loading image: $error")
226    }
227
228    private fun handleImageSaveSuccess(path: String) {
229        Log.d(TAG, "Image saved to $path")
230    }
231
232    private fun handleImageUploadSuccess(url: String) {
233        Log.d(TAG, "Image uploaded to $url")
234    }
235
236    private fun handleImageDeleteSuccess(path: String) {
237        Log.d(TAG, "Image deleted from $path")
238    }
239
240    private fun handleImageLoadSuccess(bitmap: Bitmap) {
241        Log.d(TAG, "Image loaded successfully")
242    }
243
244    private fun handleImageSaveError(error: String) {
245        Log.e(TAG, "Error saving image: $error")
246    }
247
248    private fun handleImageUploadError(error: String) {
249        Log.e(TAG, "Error uploading image: $error")
250    }
251
252    private fun handleImageDeleteError(error: String) {
253        Log.e(TAG, "Error deleting image: $error")
254    }
255
256    private fun handleImageLoadError(error: String) {
257        Log.e(TAG, "Error loading image: $error")
258    }
259
260    private fun handleImageSaveSuccess(path: String) {
261        Log.d(TAG, "Image saved to $path")
262    }
263
264    private fun handleImageUploadSuccess(url: String) {
265        Log.d(TAG, "Image uploaded to $url")
266    }
267
268    private fun handleImageDeleteSuccess(path: String) {
269        Log.d(TAG, "Image deleted from $path")
270    }
271
272    private fun handleImageLoadSuccess(bitmap: Bitmap) {
273        Log.d(TAG, "Image loaded successfully")
274    }
275
276    private fun handleImageSaveError(error: String) {
277        Log.e(TAG, "Error saving image: $error")
278    }
279
280    private fun handleImageUploadError(error: String) {
281        Log.e(TAG, "Error uploading image: $error")
282    }
283
284    private fun handleImageDeleteError(error: String) {
285        Log.e(TAG, "Error deleting image: $error")
286    }
287
288    private fun handleImageLoadError(error: String) {
289        Log.e(TAG, "Error loading image: $error")
290    }
291
292    private fun handleImageSaveSuccess(path: String) {
293        Log.d(TAG, "Image saved to $path")
294    }
295
296    private fun handleImageUploadSuccess(url: String) {
297        Log.d(TAG, "Image uploaded to $url")
298    }
299
300    private fun handleImageDeleteSuccess(path: String) {
301        Log.d(TAG, "Image deleted from $path")
302    }
303
304    private fun handleImageLoadSuccess(bitmap: Bitmap) {
305        Log.d(TAG, "Image loaded successfully")
306    }
307
308    private fun handleImageSaveError(error: String) {
309        Log.e(TAG, "Error saving image: $error")
310    }
311
312    private fun handleImageUploadError(error: String) {
313        Log.e(TAG, "Error uploading image: $error")
314    }
315
316    private fun handleImageDeleteError(error: String) {
317        Log.e(TAG, "Error deleting image: $error")
318    }
319
320    private fun handleImageLoadError(error: String) {
321        Log.e(TAG, "Error loading image: $error")
322    }
323
324    private fun handleImageSaveSuccess(path: String) {
325        Log.d(TAG, "Image saved to $path")
326    }
327
328    private fun handleImageUploadSuccess(url: String) {
329        Log.d(TAG, "Image uploaded to $url")
330    }
331
332    private fun handleImageDeleteSuccess(path: String) {
333        Log.d(TAG, "Image deleted from $path")
334    }
335
336    private fun handleImageLoadSuccess(bitmap: Bitmap) {
337        Log.d(TAG, "Image loaded successfully")
338    }
339
340    private fun handleImageSaveError(error: String) {
341        Log.e(TAG, "Error saving image: $error")
342    }
343
344    private fun handleImageUploadError(error: String) {
345        Log.e(TAG, "Error uploading image: $error")
346    }
347
348    private fun handleImageDeleteError(error: String) {
349        Log.e(TAG, "Error deleting image: $error")
350    }
351
352    private fun handleImageLoadError(error: String) {
353        Log.e(TAG, "Error loading image: $error")
354    }
355
356    private fun handleImageSaveSuccess(path: String) {
357        Log.d(TAG, "Image saved to $path")
358    }
359
360    private fun handleImageUploadSuccess(url: String) {
361        Log.d(TAG, "Image uploaded to $url")
362    }
363
364    private fun handleImageDeleteSuccess(path: String) {
365        Log.d(TAG, "Image deleted from $path")
366    }
367
368    private fun handleImageLoadSuccess(bitmap: Bitmap) {
369        Log.d(TAG, "Image loaded successfully")
370    }
371
372    private fun handleImageSaveError(error: String) {
373        Log.e(TAG, "Error saving image: $error")
374    }
375
376    private fun handleImageUploadError(error: String) {
377        Log.e(TAG, "Error uploading image: $error")
378    }
379
380    private fun handleImageDeleteError(error: String) {
381        Log.e(TAG, "Error deleting image: $error")
382    }
383
384    private fun handleImageLoadError(error: String) {
385        Log.e(TAG, "Error loading image: $error")
386    }
387
388    private fun handleImageSaveSuccess(path: String) {
389        Log.d(TAG, "Image saved to $path")
390    }
391
392    private fun handleImageUploadSuccess(url: String) {
393        Log.d(TAG, "Image uploaded to $url")
394    }
395
396    private fun handleImageDeleteSuccess(path: String) {
397        Log.d(TAG, "Image deleted from $path")
398    }
399
400    private fun handleImageLoadSuccess(bitmap: Bitmap) {
401        Log.d(TAG, "Image loaded successfully")
402    }
403
404    private fun handleImageSaveError(error: String) {
405        Log.e(TAG, "Error saving image: $error")
406    }
407
408    private fun handleImageUploadError(error: String) {
409        Log.e(TAG, "Error uploading image: $error")
410    }
411
412    private fun handleImageDeleteError(error: String) {
413        Log.e(TAG, "Error deleting image: $error")
414    }
415
416    private fun handleImageLoadError(error: String) {
417        Log.e(TAG, "Error loading image: $error")
418    }
419
420    private fun handleImageSaveSuccess(path: String) {
421        Log.d(TAG, "Image saved to $path")
422    }
423
424    private fun handleImageUploadSuccess(url: String) {
425        Log.d(TAG, "Image uploaded to $url")
426    }
427
428    private fun handleImageDeleteSuccess(path: String) {
429        Log.d(TAG, "Image deleted from $path")
430    }
431
432    private fun handleImageLoadSuccess(bitmap: Bitmap) {
433        Log.d(TAG, "Image loaded successfully")
434    }
435
436    private fun handleImageSaveError(error: String) {
437        Log.e(TAG, "Error saving image: $error")
438    }
439
440    private fun handleImageUploadError(error: String) {
441        Log.e(TAG, "Error uploading image: $error")
442    }
443
444    private fun handleImageDeleteError(error: String) {
445        Log.e(TAG, "Error deleting image: $error")
446    }
447
448    private fun handleImageLoadError(error: String) {
449        Log.e(TAG, "Error loading image: $error")
450    }
451
452    private fun handleImageSaveSuccess(path: String) {
453        Log.d(TAG, "Image saved to $path")
454    }
455
456    private fun handleImageUploadSuccess(url: String) {
457        Log.d(TAG, "Image uploaded to $url")
458    }
459
460    private fun handleImageDeleteSuccess(path: String) {
461        Log.d(TAG, "Image deleted from $path")
462    }
463
464    private fun handleImageLoadSuccess(bitmap: Bitmap) {
465        Log.d(TAG, "Image loaded successfully")
466    }
467
468    private fun handleImageSaveError(error: String) {
469        Log.e(TAG, "Error saving image: $error")
470    }
471
472    private fun handleImageUploadError(error: String) {
473        Log.e(TAG, "Error uploading image: $error")
474    }
475
476    private fun handleImageDeleteError(error: String) {
477        Log.e(TAG, "Error deleting image: $error")
478    }
479
480    private fun handleImageLoadError(error: String) {
481        Log.e(TAG, "Error loading image: $error")
482    }
483
484    private fun handleImageSaveSuccess(path: String) {
485        Log.d(TAG, "Image saved to $path")
486    }
487
488    private fun handleImageUploadSuccess(url: String) {
489        Log.d(TAG, "Image uploaded to $url")
490    }
491
492    private fun handleImageDeleteSuccess(path: String) {
493        Log.d(TAG, "Image deleted from $path")
494    }
495
496    private fun handleImageLoadSuccess(bitmap: Bitmap) {
497        Log.d(TAG, "Image loaded successfully")
498    }
499
500    private fun handleImageSaveError(error: String) {
501        Log.e(TAG, "Error saving image: $error")
502    }
503
504    private fun handleImageUploadError(error: String) {
505        Log.e(TAG, "Error uploading image: $error")
506    }
507
508    private fun handleImageDeleteError(error: String) {
509        Log.e(TAG, "Error deleting image: $error")
510    }
511
512    private fun handleImageLoadError(error: String) {
513        Log.e(TAG, "Error loading image: $error")
514    }
515
516    private fun handleImageSaveSuccess(path: String) {
517        Log.d(TAG, "Image saved to $path")
518    }
519
520    private fun handleImageUploadSuccess(url: String) {
521        Log.d(TAG, "Image uploaded to $url")
522    }
523
524    private fun handleImageDeleteSuccess(path: String) {
525        Log.d(TAG, "Image deleted from $path")
526    }
527
528    private fun handleImageLoadSuccess(bitmap: Bitmap) {
529        Log.d(TAG, "Image loaded successfully")
530    }
531
532    private fun handleImageSaveError(error: String) {
533        Log.e(TAG, "Error saving image: $error")
534    }
535
536    private fun handleImageUploadError(error: String) {
537        Log.e(TAG, "Error uploading image: $error")
538    }
539
540    private fun handleImageDeleteError(error: String) {
541        Log.e(TAG, "Error deleting image: $error")
542    }
543
544    private fun handleImageLoadError(error: String) {
545        Log.e(TAG, "Error loading image: $error")
546    }
547
548    private fun handleImageSaveSuccess(path: String) {
549        Log.d(TAG, "Image saved to $path")
550    }
551
552    private fun handleImageUploadSuccess(url: String) {
553        Log.d(TAG, "Image uploaded to $url")
554    }
555
556    private fun handleImageDeleteSuccess(path: String) {
557        Log.d(TAG, "Image deleted from $path")
558    }
559
560    private fun handleImageLoadSuccess(bitmap: Bitmap) {
561        Log.d(TAG, "Image loaded successfully")
562    }
563
564    private fun handleImageSaveError(error: String) {
565        Log.e(TAG, "Error saving image: $error")
566    }
567
568    private fun handleImageUploadError(error: String) {
569        Log.e(TAG, "Error uploading image: $error")
570    }
571
572    private fun handleImageDeleteError(error: String) {
573        Log.e(TAG, "Error deleting image: $error")
574    }
575
576    private fun handleImageLoadError(error: String) {
577        Log.e(TAG, "Error loading image: $error")
578    }
579
580    private fun handleImageSaveSuccess(path: String) {
581        Log.d(TAG, "Image saved to $path")
582    }
583
584    private fun handleImageUploadSuccess(url: String) {
585        Log.d(TAG, "Image uploaded to $url")
586    }
587
588    private fun handleImageDeleteSuccess(path: String) {
589        Log.d(TAG, "Image deleted from $path")
590    }
591
592    private fun handleImageLoadSuccess(bitmap: Bitmap) {
593        Log.d(TAG, "Image loaded successfully")
594    }
595
596    private fun handleImageSaveError(error: String) {
597        Log.e(TAG, "Error saving image: $error")
598    }
599
600    private fun handleImageUploadError(error: String) {
601        Log.e(TAG, "Error uploading image: $error")
602    }
603
604    private fun handleImageDeleteError(error: String) {
605        Log.e(TAG, "Error deleting image: $error")
606    }
607
608    private fun handleImageLoadError(error: String) {
609        Log.e(TAG, "Error loading image: $error")
610    }
611
612    private fun handleImageSaveSuccess(path: String) {
613        Log.d(TAG, "Image saved to $path")
614    }
615
616    private fun handleImageUploadSuccess(url: String) {
617        Log.d(TAG, "Image uploaded to $url")
618    }
619
620    private fun handleImageDeleteSuccess(path: String) {
621        Log.d(TAG, "Image deleted from $path")
622    }
623
624    private fun handleImageLoadSuccess(bitmap: Bitmap) {
625        Log.d(TAG, "Image loaded successfully")
626    }
627
628    private fun handleImageSaveError(error: String) {
629        Log.e(TAG, "Error saving image: $error")
630    }
631
632    private fun handleImageUploadError(error: String) {
633        Log.e(TAG, "Error uploading image: $error")
634    }
635
636    private fun handleImageDeleteError(error: String) {
637        Log.e(TAG, "Error deleting image: $error")
638    }
639
640    private fun handleImageLoadError(error: String) {
641        Log.e(TAG, "Error loading image: $error")
642    }
643
644    private fun handleImageSaveSuccess(path: String) {
645        Log.d(TAG, "Image saved to $path")
646    }
647
648    private fun handleImageUploadSuccess(url: String) {
649        Log.d(TAG, "Image uploaded to $url")
650    }
651
652    private fun handleImageDeleteSuccess(path: String) {
653        Log.d(TAG, "Image deleted from $path")
654    }
655
656    private fun handleImageLoadSuccess(bitmap: Bitmap) {
657        Log.d(TAG, "Image loaded successfully")
658    }
659
660    private fun handleImageSaveError(error: String) {
661        Log.e(TAG, "Error saving image: $error")
662    }
663
664    private fun handleImageUploadError(error: String) {
665        Log.e(TAG, "Error uploading image: $error")
666    }
667
668    private fun handleImageDeleteError(error: String) {
669        Log.e(TAG, "Error deleting image: $error")
670    }
671
672    private fun handleImageLoadError(error: String) {
673        Log.e(TAG, "Error loading image: $error")
674    }
675
676    private fun handleImageSaveSuccess(path: String) {
677        Log.d(TAG, "Image saved to $path")
678    }
679
680    private fun handleImageUploadSuccess(url: String) {
681        Log.d(TAG, "Image uploaded to $url")
682    }
683
684    private fun handleImageDeleteSuccess(path: String) {
685        Log.d(TAG, "Image deleted from $path")
686    }
687
688    private fun handleImageLoadSuccess(bitmap: Bitmap) {
689        Log.d(TAG, "Image loaded successfully")
690    }
691
692    private fun handleImageSaveError(error: String) {
693        Log.e(TAG, "Error saving image: $error")
694    }
695
696    private fun handleImageUploadError(error: String) {
697        Log.e(TAG, "Error uploading image: $error")
698    }
699
700    private fun handleImageDeleteError(error: String) {
701        Log.e(TAG, "Error deleting image: $error")
702    }
703
704    private fun handleImageLoadError(error: String) {
705        Log.e(TAG, "Error loading image: $error")
706    }
707
708    private fun handleImageSaveSuccess(path: String) {
709        Log.d(TAG, "Image saved to $path")
710    }
711
712    private fun handleImageUploadSuccess(url: String) {
713        Log.d(TAG, "Image uploaded to $url")
714    }
715
716    private fun handleImageDeleteSuccess(path: String) {
717        Log.d(TAG, "Image deleted from $path")
718    }
719
720    private fun handleImageLoadSuccess(bitmap: Bitmap) {
721        Log.d(TAG, "Image loaded successfully")
722    }
723
724    private fun handleImageSaveError(error: String) {
725        Log.e(TAG, "Error saving image: $error")
726    }
727
728    private fun handleImageUploadError(error: String) {
729        Log.e(TAG, "Error uploading image: $error")
730    }
731
732    private fun handleImageDeleteError(error: String) {
733        Log.e(TAG, "Error deleting image: $error")
734    }
735
736    private fun handleImageLoadError(error: String) {
737        Log.e(TAG, "Error loading image: $error")
738    }
739
740    private fun handleImageSaveSuccess(path: String) {
741        Log.d(TAG, "Image saved to $path")
742    }
743
744    private fun handleImageUploadSuccess(url: String) {
745        Log.d(TAG, "Image uploaded to $url")
746    }
747
748    private fun handleImageDeleteSuccess(path: String) {
749        Log.d(TAG, "Image deleted from $path")
750    }
751
752    private fun handleImageLoadSuccess(bitmap: Bitmap) {
753        Log.d(TAG, "Image loaded successfully")
754    }
755
756    private fun handleImageSaveError(error: String) {
757        Log.e(TAG, "Error saving image: $error")
758    }
759
760    private fun handleImageUploadError(error: String) {
761        Log.e(TAG, "Error uploading image: $error")

```

Deploy on mobile

- More example



Conclusion

Our objective is to develop CNN model for food data processing to interpret nutritional values.

We have trained the object detection model with TensorFlow Lite Model Maker library and deploy it on mobile application

Thank you for your attention

Feel free to ask any question.

