

養蜂のための外敵検出カメラシステムの開発

寺田充樹^{†1} 佐藤証¹

概要： 養蜂場に大きな被害を与えるミツバチの襲来を検知するため、機械学習による物体認識アルゴリズムを実装したカメラシステムを開発した。養蜂場で撮影したスズメバチの動画とインターネットで収集した画像を用いて学習モデルを生成し、マイコンボード Raspberry Pi にアルゴリズムを実装してその精度と速度の評価を行った。SSD アルゴリズムを用いたサンプル動画によるテストでは、平均で 92.9% の検出精度が得られた。誤検知も発生しているが、そのときの信頼度は最大でも 62% であったため、検出の閾値をこれよりも高く設定することで誤検知をなくすことが可能である。Raspberry Pi 4 Model B に TensorFlow Lite を実装し、Wi-Fi 接続した 10 台のカメラでスズメバチの映像を転送したときの処理性能は 1 台当たり 1 フレーム/秒となり、本システムの十分な実用性が示された。

1. はじめに

本研究では養蜂場のミツバチを集団で襲い壊滅させるスズメバチに早期対応するため、機械学習によりその襲来を検知するカメラシステムを開発する。筆者らは IoT 技術を活用し、巣箱の温湿度や重量などのデータを遠隔で 24 時間遠隔モニタするスマート養蜂の研究を進めている。しかし大きな動画データを扱う場合、草原や山間部の養蜂では安定したネットワークの確保が難しく、また都市養蜂を行うビルの屋上は電波環境が悪いことも珍しくない。したがって、カメラの映像をサーバに送信して物体認識を行うのではなく、ローカルで処理するいわゆるエッジコンピューティングの実現が重要となる。そこで、Raspberry Pi のローカルサーバに複数カメラを接続してスズメバチを検出し、その精度および速度性能を評価する。

2. 機械学習用データセットの作成

代表的な 3 つの画像認識アルゴリズム Faster R-CNN (Regional Convolution Neural Network)[1], YOLO (You Only Look Once) v3[2], SSD (Single Shot multibox Detector)[3] に

対して、スズメバチ検出のための学習を行った。学習に用いたデータには、千葉県の養蜂場で一つの養蜂箱を正面、斜め、横の 3 台のカメラで 10 日間撮影した動画(図 1)からスズメバチの画像をキャプチャしたものと、インターネットで収集した画像を用いた。学習には図 2 に示した方法で、3 種類のデータセット 1~3 を作成した。データセット 1 は動画から生



図 1 スズメバチの動画

成した 120 枚と、ネットの収集画像 380 枚の、計 500 枚のラベル付きの画像である。動画の 120 枚は正面から撮影したスズメバチが映っているフレームをそのままのサイズでキャプチャした。画像が大きいと学習に長時間を要するため、各画像を 320×320 ピクセルに縮小し、縦横比が 1:1 でないものはスペースを白で塗りつぶしている。画像中でスズメバチが映っている部分を長方形で指定するアノテーション作業には labelImg[4]を用い、長方形のデータを Pascal VOC 形式で xml ファイルに出力した。アノテーション作業はデータセット 2 と 3 も同様である。

データセット 2 はデータセット 1 の 500 枚に、横と斜めのキャプチャ画像 500 枚を加えた 1,000 枚の画像である。



図 2 データセットの作成

¹ 電気通信大学

^{†1} 現在、ソフトバンク株式会社

横と斜めの動画中のスズメバチは正面に比べてかなり小さいため、スズメバチが映っている部分を正方形に大きくトリミングをした後で320×320に縮小した。

データセット3はデータセット2に、動画から生成した画像400枚を加えた1,400枚である。元画像はトリミングせずにそのまま縮小し、アノテーションを行なった。

3. スズメバチ検出精度の評価

3つの画像認識アルゴリズムに対し、データセット1を用いてスズメバチの検出の精度を評価した。なお、YOLOv3はIntel Core i7パソコン上でImageAIライブラリ[5]を用い、Faster R-CNNとSSDはGPGPU GeForce GTX 980Tiを搭載したCore i5パソコン上でTensorFlowライブラリ[6]を用いて実行した。

検出精度の評価には、表1の6つの動画を用いた。正1と2は図1に示した正面からの動画、斜1と2は同様に図1の斜めから、横1と2は横から撮影したものである。奇数番号の動画はスズメバチが1匹で、巣箱へ出入りするミツバチが少なく、偶数番号の動画はスズメバチが2匹で、巣箱へ出入りするミツバチが多い。全ての動画は1フレーム/秒で、「スズメバチ数」は各動画のフレームに映っていたスズメバチの総数である。

表1 精度評価用動画

動画	解像度	フレーム数 (枚)	スズメバチ数 (匹)
正1	1,280×720	54	42
正2		391	580
斜1	1,920×1,080	129	48
斜2		116	127
横1	1,280×720	16	11
横2		121	135

まず、データセット1の500枚の画像をトレーニング用400枚とテスト用に100枚に分け、ImageAIの公式ページ[5]に公開のYOLOv3用の事前学習モデル pretrained-yolov3.h5、TensorFlowが提供するFast R-CNNとSSD用事前学習モデルの ssd_mobilenet_v2_quantized_coco と faster_rcnn_inception_v2_coco から転移学習を行った。図3に各アルゴリズムの学習回数と損失値の関係を示す。YOLOv3のバッチサイズとエポック数は、公式ページに従って4と200に設定した。学習開始から約32時間経過した23step以降、2日間学習を続けても損失値が更新されな

かったので学習を打ち切り、損失値が最小3.602の23step時の学習データを用いた。Faster R-CNNのバッチサイズは転移学習モデルで設定されている1を指定し、0.2秒/stepの速度で92,133step学習させた。SSDのバッチサイズはGPUのメモリ制約から6とし、約2秒/stepの速度で学習を進め、200,000step時の学習データをTensorFlow Lite[7]用に変換した。

各学習モデルによる精度評価の結果を表2にまとめる。YOLOv3のスズメバチの検出率は全体の平均で88.9%と高いものの、ミツバチや巣箱の木目等をスズメバチと誤検出することが多発した。Faster R-CNNは正面からの動画の検出率は80%を超え誤検出も少ないが、斜めや横の検出率が低く、平均では61.0%であった。誤検出はカメラの近くを通ったミツバチをスズメバチと誤検出していた。SSDでは正面からの検出率は約65%とFaster R-CNNよりも高かったが、斜めと横では一匹も検出することができなかった。

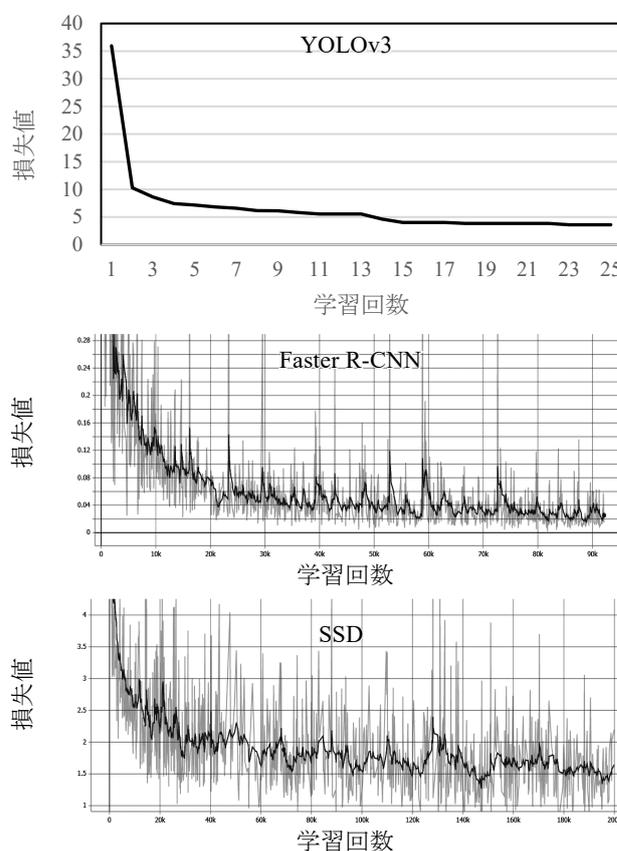


図3 各アルゴリズムの学習回数と損失値

表2 3つのアルゴリズムの検出精度の比較

動画	YOLOv3				Faster R-CNN				SSD			
	目視 (匹)	検出数 (匹)	検出率 (%)	誤検出 (回)	目視 (匹)	検出数 (匹)	検出率 (%)	誤検出 (回)	目視 (匹)	検出数 (匹)	検出率 (%)	誤検出 (回)
正1	42	40	95.2	0	42	33	78.6	0	42	20	7.1	0
正2	580	560	96.6	63	580	477	82.2	11	580	282	68.8	0
斜1	48	39	91.2	49	48	6	12.5	0	48	0	0.0	0
斜2	127	98	77.2	116	127	29	22.8	0	127	0	0.0	0
横1	11	8	72.8	0	11	0	0.0	0	11	0	0.0	0
横2	135	84	62.2	13	135	30	22.2	1	135	0	0.0	0
合計	943	829	88.9	241	943	575	61.0	12	943	302	42.6	0

表3 データセット1~3での学習によるSSDの検出精度の比較

動画	データセット1				データセット2				データセット3			
	目視 (匹)	検出数 (匹)	検出率 (%)	誤検出 (回)	目視 (匹)	検出数 (匹)	検出率 (%)	誤検出 (回)	目視 (匹)	検出数 (匹)	検出率 (%)	誤検出 (回)
正1	42	20	7.1	0	42	25	59.5	0	42	35	83.3	30
正2	580	282	68.8	63	580	498	85.9	3	580	559	96.4	5
斜1	48	0	0.0	49	48	2	4.2	0	48	42	87.5	0
斜2	127	0	0.0	116	127	1	0.8	0	127	110	86.6	0
横1	11	0	0.0	0	11	0	0.0	0	11	10	90.9	0
横2	135	0	0.0	13	135	0	0.0	0	135	120	88.9	0
合計	943	302	42.6	241	943	526	55.8	3	943	876	92.9	35

4. SSDの検出精度の向上

上記学習モデルを用い、養蜂場でエッジサーバーとして利用する Raspberry Pi 4 Model B+ メモリ 2GB 上に ImageAI, TensorFlow, TensorFlow Lite を実装して、それぞれ YOLOv3, Faster R-CNN, SSD を動かした。その結果、解像度 1,280×720 の 10 秒の動画中の 2 匹のスズメバチの検出を、SSD は 1 フレームを 0.29 秒(=1/3.5)で処理できたが、YOLOv3 は約 6 秒、Faster R-CNN は約 9 秒も要し、リアルタイムでの運用が困難であることがわかった。また、全てのフレームでスズメバチを検出することよりも、誤った警報が頻繁に利用者に届かないように誤検出をゼロにしたいという観点において、SSD は YOLOv3 や Faster R-CNN に対してある意味優れている言うこともできる。しかしながら、斜めと横の検出率がゼロでは、カメラの設置場所が変わった時にまったく検出できない可能性が懸念される。そこで、以下では SSD の検出率向上のため、画像データを増やしたデータセット 2 とデータセット 3 を用いた学習を進める。なお、評価にはこれまでと同じ 6 つの動画を用いた。

まず、データセット 2 の 1,000 枚の画像をトレーニング用 800 枚とテスト用 200 枚に分け、バッチサイズを 6 に設定して学習させ、86,970step 時の学習データを TensorFlow Lite 用に変換して精度評価を行った。さらに、データセット 3 の 1,400 枚の画像をトレーニング用 1,120 枚とテスト用 280 枚に分けて学習を進め、89,621step 時の学習データでも精度評価を行った。それらを表 2 に示したデータセット 1 の SSD の結果と合わせて表 3 に示す。

データセット 1~3 の順に学習を進めた結果、正面からの動画の正 1 と 2 に対しては、それぞれ 7.1%→59.5%→83.3%、68.8%→85.9%→96.4%と精度が大幅に向上している。データセット 3 では、斜めと横からの検出率も 80%以上に向上し、平均では 92.9%となった。しかしその反面、正面からの誤検出が計 35 回も生じてしまった。巣箱の一部をスズメバチと誤った例を図 4 に示す。このような誤検出時の信頼度は最大 62%で 50%以下が 27 回、ミツバチをスズメバチと誤ったときの信頼度は最大でも 37%であった。つまりスズメバチ検出の閾値を 80%程度と高く設定すれば、少なくともテストデータでは誤検知をなくすることができる。またスズメバチは巣箱の周りに数分以上滞在するので、正し

く検知されている場合、それは時間的・空間的に連続しているはずである。先述のように養蜂現場での利用では誤検出をゼロにすることが重要となる。したがって、学習データを増やすことに加え、適切な閾値の設定方法の検討や、連続した検知の判断等を行うことで、誤検知をゼロにすることが可能であると考えられる。



図4 データセット3での学習によるSSDの誤検出

5. 複数カメラによる検出速度性能

Raspberry Pi に複数台のカメラを接続したときの、SSD の処理速度を評価する。表 4 に Raspberry Pi 4 Model B 2GB メモリ(以下 RP4 B)、下位モデルの Raspberry Pi Model 3 B+(以下 RP3 B+)、Raspberry Pi Zero W (以下 RPZ W)の 1 秒に処理できるフレーム数と、その逆数である 1 フレームあたりの処理時間を示す。RP4 B は 1 フレームの処理に 0.29 秒しかかからないのに対して、RPZ W は 17 秒もかかっており、リアルタイム処理には向かないことがわかる。RP3 B+ は 1 秒を切っており、実用範囲内であるが、RP4 B の価格が RP3 B+ と同等に下がってきたため、低消費電力の RP3 B+ を使用したいといった場合など、その用途は限定的である。

表4 Raspberry Pi の速度比較

	RP4 B	RP3 B+	RPZ W
フレーム/秒	3.5	1.3	0.059
秒/フレーム	0.29	0.77	16.95

次に、図 5 の 32 ビットマイコンを実装した実装したカメラ ESP32-CAM (製品版の ESP32-WROOM-32S を ESP32-WROOM-32E に張替えている) を図 6 のように 1~10 台並べ、Raspberry Pi に Wi-Fi して処理性能を調べた。カメラはスズメバチの写った写真を撮影し、800×600 ピクセルの映像を rtsp (real time streaming protocol) プロトコルで RP4 B ま

たは RP3 B+ に転送した。また、Google の USB タイプの AI アクセラレータ Coral USB Accelerator [8] を挿した RP4 B も利用した。表 5 に接続したカメラ台数、1 秒間の平均処理フレーム数 (fps)、カメラ 1 台当たりの平均処理フレーム数 (fps/台)、そして CPU 使用率を示すまた図 7 はカメラの接続数と、全体の fps である。



図 5 ESP32-WROOM-32E を実装したカメラモジュール

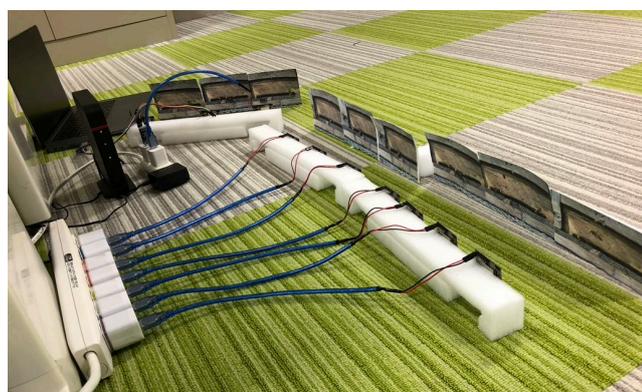


図 6 実験の様子

表 5 Raspberry Pi の処理性能の比較

接続台数	RP4 B			RP3 B			Coral Accel.	
	fps	fps/台	CPU (%)	fps	fps/台	CPU (%)	fps	fps/台
1	2.51	2.51	18.6	1.48	1.48	25.6	8.33	8.33
2	4.46	2.43	40.2	2.46	1.23	49.4	7.12	3.56
3	6.93	2.31	40.9	3.63	1.21	70.8	6.63	2.21
4	8.56	2.14	71.0	4.68	1.17	93.1	7.04	1.76
5	9.45	1.89	78.5	4.90	0.98	100	8.05	1.61
6	9.00	1.50	82.0	4.86	0.81	100	9.36	1.56
7	10.29	1.47	91.6	4.97	0.71	100	9.66	1.38
8	10.32	1.29	100	-	-	-	10.96	1.37
9	10.44	1.16	100	-	-	-	10.35	1.15
10	10.05	1.05	100	-	-	-	12.20	1.22

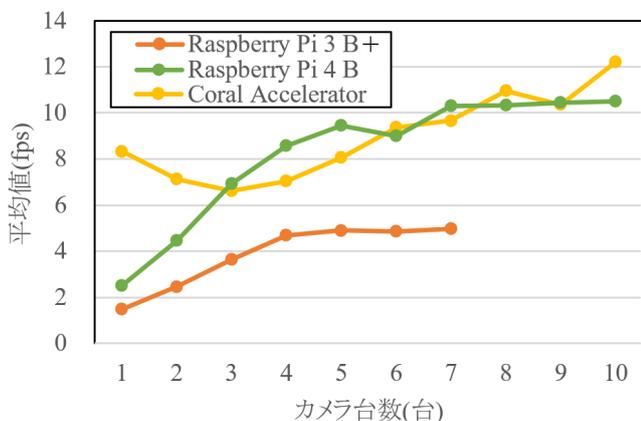


図 7 Raspberry Pi の処理性能の比較

RP4 B と RP3 B+ は共に 4 コアなので、カメラの接続が 4 台までは 1 台当たりの性能の低下はわずかで、台数が増えるにしたがって全体の fps が大きく向上している。RP3 B+ は 5 台目で CPU 使用率が 100% に達して処理速度が頭打ちとなり、8 台接続しようとするとうりフリーズしてしまった。それに対して RP4 B は 8 台目で CPU 使用率が 100% となり速度が頭打ちとなっているが、10 台接続しても止まることはなかった。またその時の 1 台当たりの処理速度は 1.05fps であった。先に述べたようにスズメバチは巣箱の周りに数分間は滞在するため、その間に数十~数百枚の画像に対して検出処理が行えることになる。したがって、1 秒間に 1 フレームの処理は十分な性能であり、10 台以上のカメラを接続することも実用上問題ないと考えられる。

Coral USB Accelerator の使用では、カメラが 1~2 台では RP4 B よりも高い性能が得られているものの、それ以上ではほぼ五角である。Coral USB Accelerator は RP4 B よりも高価であり、また単体では動作せずに RP4 B が別途必要であることを考えると、スズメバチの検出で利用するメリットはないと言える。

以上から、1 台のマイコンボード Raspberry Pi 4 model B だけで、10 台のカメラによるスズメバチの検出に実用上十分な性能を有られることが示された。

6. むすび

3 つの物体認識アルゴリズム YOLOv3, Faster R-CNN, SSD を用いてスズメバチ検出の精度と Raspberry Pi での処理速度評価を行い、性能の高かった SSD に対してデータを増やして学習を進めた。Raspberry Pi 4 model B 上に TensorFlow Lite ライブラリを用いて SSD を実装し、カメラ 10 台の映像を処理させたとき、各カメラで 1 秒間に 1 フレームの処理が可能であった。スズメバチは巣箱に短くとも数分間留まるため、リアルタイムでの運用に十分な性能と言える。

テスト動画による SSD の検出率は平均で 92.9% と高いが、多少の誤検知が生じていた。しかし誤検知時の信頼度は最大 62% と低いため閾値を 80% 程度に設定したり、検出が連続していたときだけ正しい判定とすることで誤検出の問題は解決可能である。

今回は一ヶ所の養蜂場で撮影した動画で評価を行ったが、実用化に向けたより一層の検出精度向上のため、複数ヶ所での動画撮影を進めている。またさほど深刻ではないが野鳥やカマキリ等もミツバチを捕食するため、スズメバチ以外の外敵も検出できるよう学習を行っていく。また、新女王バチの誕生により群が 2 分して一方が巣を離れる分蜂は養蜂にとって最大の問題であり、カメラ映像や他のセンサのデータの活用により、そのような緊急事態をいち早く察知、あるいは予測できるシステムの開発も今後の課題である。

参考文献

- [1] S. Ren, K. He, R. Girshick, and J. Sun: Faster R-CNN: Towards real-time object detection with region proposal networks, Advances in Neural Information Processing Systems (NIPS 2015). Jun. 2015.
- [2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi: You only look once: Unified, real-time object detection, Proc IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2106), pp. 789-798, Jun. 2016.
- [3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg: SSD: Single Shot MultiBox Detector, Proc. European Conference on Computer Vision (ECCV 2016), pp.21-37, Sep. 2016.
- [4] tzutalin/labelImg, <<https://github.com/tzutalin/labelImg/>> (参照 2021-05-01).
- [5] M. Olafenwa: Official English Documentation for ImageAI! — ImageAI 2.1.5, <<https://imageai.readthedocs.io/en/latest/>> (参照 2021-05-01).
- [6] TensorFlow, <<https://www.tensorflow.org/>> (参照 2021-05-01).
- [7] TensorFlow Lite <<https://www.tensorflow.org/lite?hl=ja>> (参照 2021-05-01).
- [8] Coral: USB accelerator, <<https://coral.ai/products/accelerator>> (参照 2021-5-1).

本研究の一部は、令和2年度日本中央競馬会畜産振興事業の支援を受けて実施された。