

2025年度

学士論文

論題

一般ユーザ向け開放語彙物体検出のスマホ運用基盤
— 事前語彙化と少量学習による反復改善 —

指導教員 孟林教授

立命館大学 理工学部 電子情報工学科

学籍番号 2290220041-3
氏名 後藤 晴貴

論文要旨

AIが広く普及した現代において、対話型AIは一般ユーザに浸透している一方、画像認識は依然として専門知識を要し、エンジニア主体の領域に留まっている。本研究は一般ユーザを主対象とし、データ収集からアノテーション、学習、評価、利用までを一貫して支援し、個々の利用状況に特化した画像認識モデルの調整・ファインチューニングを可能にするアプリケーションを開発した。本システムにより、プログラミング経験が乏しいユーザでも少量の自前データから用途特化モデルを反復的に改善でき、画像認識活用の敷居を下げる。ケースとして料理画像検出を対象に、UI内でのデータ収集・ラベリングから学習、精度の可視化までを一連の操作で完結できることを示し、一般ユーザによるモデルカスタマイズの実用可能性を示した。定量的には、語彙登録のみの検証で10/10件の成立を確認し、手動ラベリング50枚+ファインチューニングによりmAP50=0.9235, mAP50-95=0.5188, Precision=0.9065, Recall=0.7754, テスト13枚に対する検出率69.2%を得た。

目次

1	はじめに	1
2	関連研究	3
2.1	物体検出モデル	3
2.1.1	YOLO-World	3
2.1.2	GLIP	4
2.1.3	Grounding DINO	4
2.1.4	OWL-ViT	4
2.1.5	Detic	4
2.1.6	RegionCLIP	4
2.1.7	CLIP と開放語彙の基盤	4
2.2	食事画像データセット	4
3	提案手法	5
3.1	設計方針と構成	5
3.2	YOLO-World の運用	5
3.2.1	API エンドポイントの概要	6
3.3	データ収集・ラベリング・学習	6
3.4	UI フロー（スマホ中心）と操作例	6
3.5	スマホ最適化と制約	11
4	語彙登録のみでの検出成立性の検証	12
4.1	目的	12
4.2	操作フローと条件	12
4.3	評価設定	12
4.4	評価項目	13
4.5	ケーススタディ（smartphone）	13
4.6	実験結果（smartphone10枚）	13
4.7	アプリーション：confidence 閾値の影響	19
4.8	考察と制約	19
5	手動ラベリングとファインチューニングによる検出成立性の検証	20
5.1	目的	20
5.2	前提と環境	20
5.3	手順	20
5.4	評価設計	20
5.5	実装上の要点	21
5.6	実験結果	21
5.6.1	実験設定	21
5.6.2	実験手順	21

5.6.3 学習結果	21
5.6.4 検出結果	22
5.6.5 考察	22
5.7 期待と限界	23
5.8 今後の改善	23
6 まとめ	24
.1 実験端末の詳細	24

図目次

3.1 提案 UI の反復ループ (1/4)	7
3.2 提案 UI の反復ループ (2/4)	8
3.3 提案 UI の反復ループ (3/4)	9
3.4 提案 UI の反復ループ (4/4)	10
4.1 語彙 smartphone の有無による比較 (1/5)	14
4.2 語彙 smartphone の有無による比較 (2/5)	15
4.3 語彙 smartphone の有無による比較 (3/5)	16
4.4 語彙 smartphone の有無による比較 (4/5)	17
4.5 語彙 smartphone の有無による比較 (5/5)	18

表目次

3.1 主要 API エンドポイントの主用途	6
4.1 ゼロショット検証の評価設定	12
4.2 語彙登録の有無による smartphone 検出成立 (10 枚)	13
4.3 confidence 閾値の影響 (観察ベース)	19
5.1 ファインチューニング後の検出結果 (テスト画像 13 枚、50 枚学習、38 エポック)	22

1

はじめに

対話型 AI の普及により、一般ユーザが自然言語によって高度な情報処理を日常的に活用する時代が到来した。一方で、画像認識をはじめとするコンピュータビジョン (CV) は、データ収集、アノテーション、学習・評価、運用を含む一連の工程が分断されやすく、専門的な知識・ツール群を横断的に扱う必要があることから、依然として一般ユーザにとって参入障壁が高い。特に、用途特化のモデルを自分で調整（チューニング／ファインチューニング）し、反復的に改善していくためには、学習用データの拡充、失敗の可視化、改善仮説の検証といった実務的ワークフローが不可欠である。

本研究では、こうしたギャップを解消し、一般ユーザが「自分の目的に合った画像認識モデル」を自力で構築・調整できる環境を提供することを目的として、エンドツーエンドのモデル管理アプリケーションを開発した。本システムは、スマートフォン／PC のいずれからでも利用可能なクロスプラットフォーム UI と、高性能な Web API を組み合わせ、データ収集・ラベリング・学習・推論・評価・モデル運用までを一貫して支援する。具体的には、カメラやギャラリーからの画像取得、ドラッグ&ドロップによる直感的なアノテーション、Ultralytics YOLO を用いたリアルタイム物体検出、学習の非同期実行と進捗監視、履歴の可視化、データセット分析、モデルの保存・切替といった機能を統合し、一般ユーザでも試行錯誤を通じてモデルを改善できる実用的なワークフローを実現した。

運用面では、限られた計算資源でも現実的に扱えるよう、学習をバックグラウンドで非同期実行し、UI をブロックしない設計とした。さらに、再現性と保守性を高めるため、API の仕様を明確化し、環境構築と動作手順を統一した。これにより、ユーザは最小限の初期設定で環境を整え、反復的なモデル改善に集中できる。

適用領域としては、料理画像を例題に据え、器や料理種別に応じた検出のしやすさ、データの集め方、モデルの差し替えやクラス管理など、実務的な観点からの検討を行った。用途特化の小規模データから出発し、UI 上でのアノテーションと学習、可視化により改善ポイントを特定しながら、ユーザ自身の目的に合わせたモデルを段階的に洗練させることが可能である。これにより、画像認識活用の敷居を下げ、一般ユーザ主導の”現場適合”モデルの創出を後押しする。

本稿の主な貢献を以下に示す。

- データ収集から学習・評価・運用までを統合した一般ユーザ向け CV モデル管理アプリケーションの設計・実装
- クロスプラットフォーム UI 上での直感的ラベリングとリアルタイム物体検出の統合による反復改善の促進
- 学習の非同期実行、進捗・履歴・メトリクスの可視化、モデル管理機能の一体化による実用的ワークフローの提供
- Web/Android/iOS でのクロスプラットフォーム UI の提供

本論文では2章で背景および関連研究について述べ、3章で提案システムの設計方針と機能構成を示す。4章では語彙登録のみでの検出成立性をスマートフォンから検証し、5章では手動ラベリングとファインチューニングによる検出成立性の検証結果を示す。6章ではまとめと今後の課題（軽量化・最適化、拡張可能性、運用上の安全性・信頼性など）について議論する。

2

関連研究

2.1 物体検出モデル

従来の物体検出は COCO などの固定語彙 (closed-set) を前提としており、学習時に定義したカテゴリのみに限定されるという制約がある。一方、実環境では「未学習カテゴリ」を含む開放語彙 (open-vocabulary) への拡張が重要である。YOLO 系列は Backbone・Neck・Head からなる一段 (one-stage) 検出器として高い効率を示してきたが、語彙の固定という制約が実用展開のボトルネックとなってきた。

2.1.1 YOLO-World

YOLO-World は、従来 YOLO の効率性を維持しつつ、視覚 - 言語モデリングによって開放語彙検出を実現した検出器である [1]。中核は、(1) 再パラメータ化 (re-parameterization) 可能な Vision-Language PAN (RepVL-PAN) によりテキスト埋め込みと画像特徴の相互作用を組み込み、推論時には言語エンコーダを取り除いて事前語彙化 (prompt-then-detect, offline vocabulary) したテキスト埋め込みをモデル重みに吸収すること (付録 B.1 式 (1) より)、(2) 検出データ・グラウンディング・画像テキストの 3 種データを統一的に扱う領域 - テキスト対に基づく大規模事前学習、にある。

学習時は、ラベル付き検出データ、グラウンディングデータ、画像 - テキスト対から得る擬似ボックスを併用し、領域 - テキスト対比学習で視覚と言語の整合を強化する。推論時はテキストエンコーダを用いず、オンラインで事前計算した語彙埋め込みを RepVL-PAN へ再パラメータ化して統合するため、実行時レイテンシを抑えつつ開放語彙に対応できる。

学習スキームとしては、領域 - テキスト対に基づくコントラスト学習を大規模データで行う。実データ (Objects365 等) に加え、CC3M などの画像テキストデータから、名詞抽出 → 擬似ボックス生成 (GLIP 等) → CLIP による再スコアリングと NMS/閾値フィルタリングという自動ラベリングパイプラインで領域 - テキスト対を構築し (例: NMS=0.5, スコア閾値=0.3), 開放語彙能力を強化する。小型モデル (YOLO-World-S) に対しては、高品質アノテーションや適量の擬似ラベルを組み合わせることでゼロショット性能が向上することが示されている。

性能面では、LVISにおいて 35.4 AP かつ V100 上で 52 FPS を達成し (TensorRT なし), 同規模の既存手法に対して精度・速度のバランスで優位性を示す。また、学習後は事前語彙化によりカテゴリ埋め込みをモデル重みに取り込み、エッジ展開時のテキストエンコーダ依存を排除する。さらに、COCO のような固定語彙タスクへ移行する際は、言語条件付けを無効化することで従来 YOLO と同等の運用効率で微調整可能である。総じて、YOLO-World は固定語彙検出と開放語彙検出の橋渡しを行い、汎用実応用 (ゼロショット検出, 参照物体検出, オープン語彙インスタンスセグメンテーション等) に適した現実的なデプロイ手段を与える。

2.1.2 GLIP

GLIP[2] は、言語で条件付けたボックス予測を可能にする Grounded Language-Image Pre-training を提案し、検出とグラウンディングを統一的に学習することで開放語彙検出の基盤を築いた。

2.1.3 Grounding DINO

Grounding DINO[3] は、DINO 系の検出器にテキスト条件付けを組み込み、高精度なフレーズグラウンディングとゼロショット検出を実現した。

2.1.4 OWL-ViT

OWL-ViT[4] は Vision Transformer を用いてオープン語彙検出をシンプルに実装し、任意語彙に対する柔軟なゼロショット検出を示した。

2.1.5 Detic

Detic[5] は画像レベル監督を活用し、大規模語彙に拡張可能な検出を実現した。

2.1.6 RegionCLIP

RegionCLIP[6] は、領域レベルでの CLIP 事前学習を行い、領域特徴と言語埋め込みの整合を高めて開放語彙検出を強化した。

2.1.7 CLIP と開放語彙の基盤

CLIP[7] は大規模画像—テキスト対で学習した表現を提供し、多くの開放語彙検出器の言語埋め込みの基盤となっている。

2.2 食事画像データセット

本研究の料理画像応用に関連して、以下のデータセットが広く用いられる。

- Food-101[8]: 101 クラスの料理画像分類データセット。
- UEC-Food100/256[9]: バウンディングボックス付きの食事画像データセット。
- FoodSeg103/UEC-FoodPix[10]: 食事セグメンテーションのためのピクセル精度アノテーション。
- IM2Calories[11], Nutrition5k[12]: カロリー・栄養推定の文脈での食事分析。

3

提案手法

本研究の目的は、スマートフォンを含む汎用端末のみでデータ収集からラベリング、学習、評価、運用までを一気通貫に反復できる実用システムを構築し、非専門家でも短時間で自作の用途特化モデルを育てられることを示す点にある。中核には開放語彙検出器である YOLO-World[1] を据え、事前語彙化によって実行時の言語エンコーダ依存を排除し、軽量・高速な推論を維持する。

3.1 設計方針と構成

フロントエンドは React Native + Expo により Android/iOS/Web を單一コードベースで提供し、タブ (Detection / Labeling / Training / Models / Analytics) に機能を整理する。バックエンドは FastAPI で統一し、検出・語彙管理・学習・履歴・可視化・データ分析の API を備える。ユーザは語彙を自ら定義・追加し、必要データを小刻みに収集・注釈付けして学習をトリガし、結果を見ながら再収集・再学習を繰り返す。

3.2 YOLO-World の運用

YOLO-World は視覚 - 言語モデリングにより、ユーザ定義語彙での開放語彙検出を実現する。POST /model/classes で登録した語彙は custom_vocab.json に永続化され、モデルのクラス埋め込みへ反映される。推論は/detect で実行し、バウンディングボックス・クラス・スコアと描画済み画像を返す。事前語彙化により、推論時はオフラインで固定した語彙埋め込みを用い、モバイルでも実用的なレイテンシを確保する。

3.2.1 API エンドポイントの概要

Endpoint	入力 (要旨)	主用途
/model/classes	クラス名 (配列/文字列)	語彙の登録・永続化 (custom_vocab.json)
/detect	画像	事前語彙化済み Head での推論 (BBox, Score, Class)
/detect/with-confidence	画像, conf	閾値指定推論 (FP/FN バランス調整)
/labeling/submit	画像, YOLO ラベル	ラベル保存 (images/labels, classes.txt 更新)
/training/start	epochs 等	同期学習の起動
/training/start-async	epochs 等	非同期学習の起動 (UI ブロック回避)
/training/status	-	学習進捗・状態の取得
/training/history	-	学習履歴一覧の取得
/training/metrics/{run-name}		時系列メトリクス取得 (Plotly 描画に利用)
/models/*	-	モデル一覧・切替・バックアップ・検証

表 3.1: 主要 API エンドポイントの主用途

3.3 データ収集・ラベリング・学習

Labeling タブで作成したアノテーションは YOLO 形式で保存し (`training_data/` 配下)、
`/training/start` または `/training/start-async` で微調整を起動する。既定は CPU 実行
 だが、CUDA 対応マシンでは設定により GPU (`device='cuda'`) で学習・推論が可能である。完了時
 には `best.pt` を自動ロードし、`/training/status` で進捗を可視化する。`/models/*` 群で
 モデル一覧・切替・バックアップ・検証が可能で、`/training/history` と `/training/metrics/{run_name}`
 から学習履歴・時系列メトリクスを取得し UI で Plotly 描画する。

3.4 UI フロー (スマホ中心) と操作例

本節では、スマートフォン想定の最短反復ループ (撮影→検出→語彙追加→再検出→ラ
 ベリング→学習→評価) を画面遷移で示す。各ページに 4 枚ずつ配置する。本図は、撮影→
 初回検出→語彙確認/追加の最短ループを示す。(a) で入力し、(b) で未検出を確認、(c)(d)
 で語彙追加により次ステップの検出改善を準備する。この操作により、学習前でも検出の立
 ち上がりを確認できる。

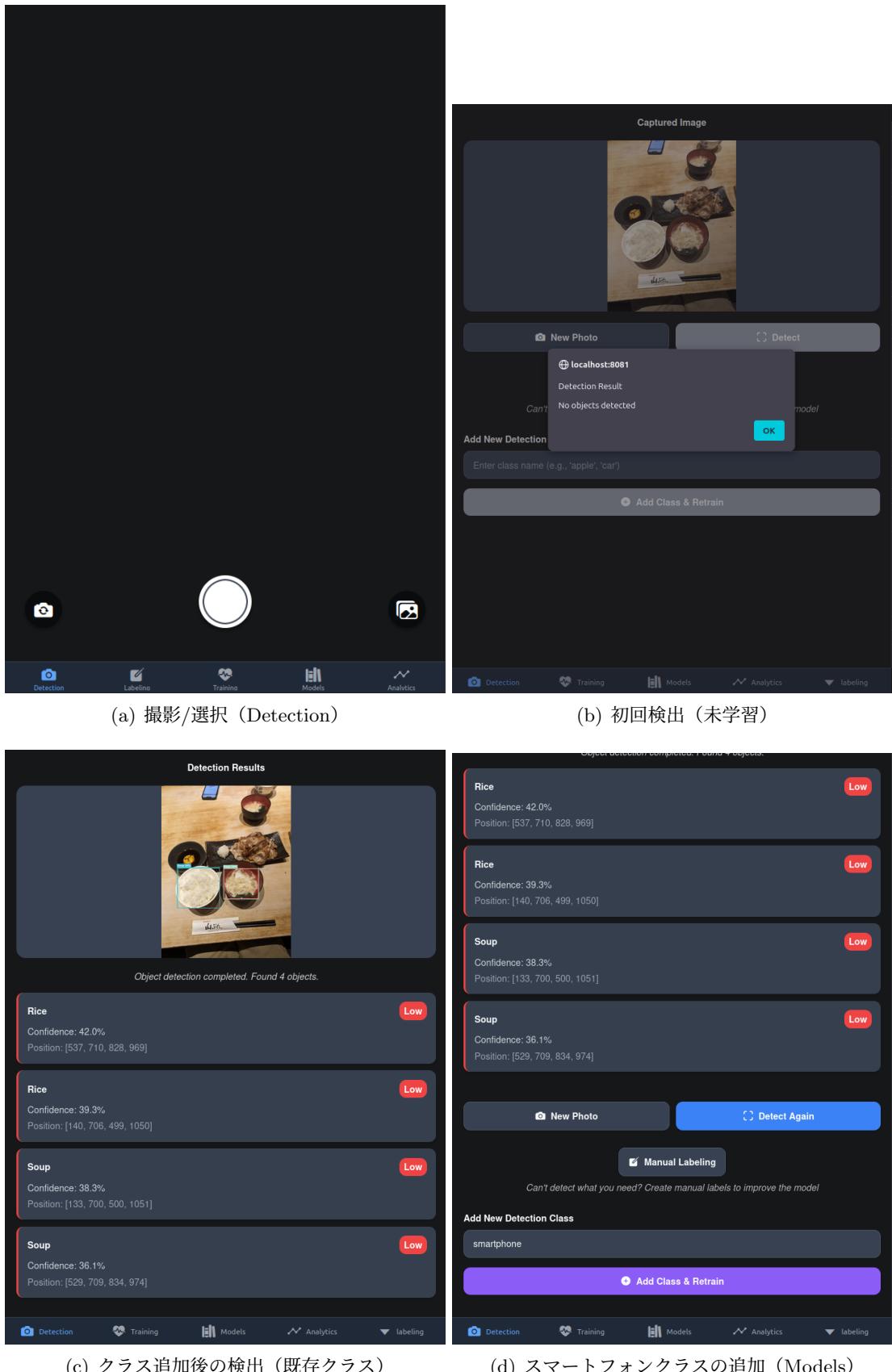
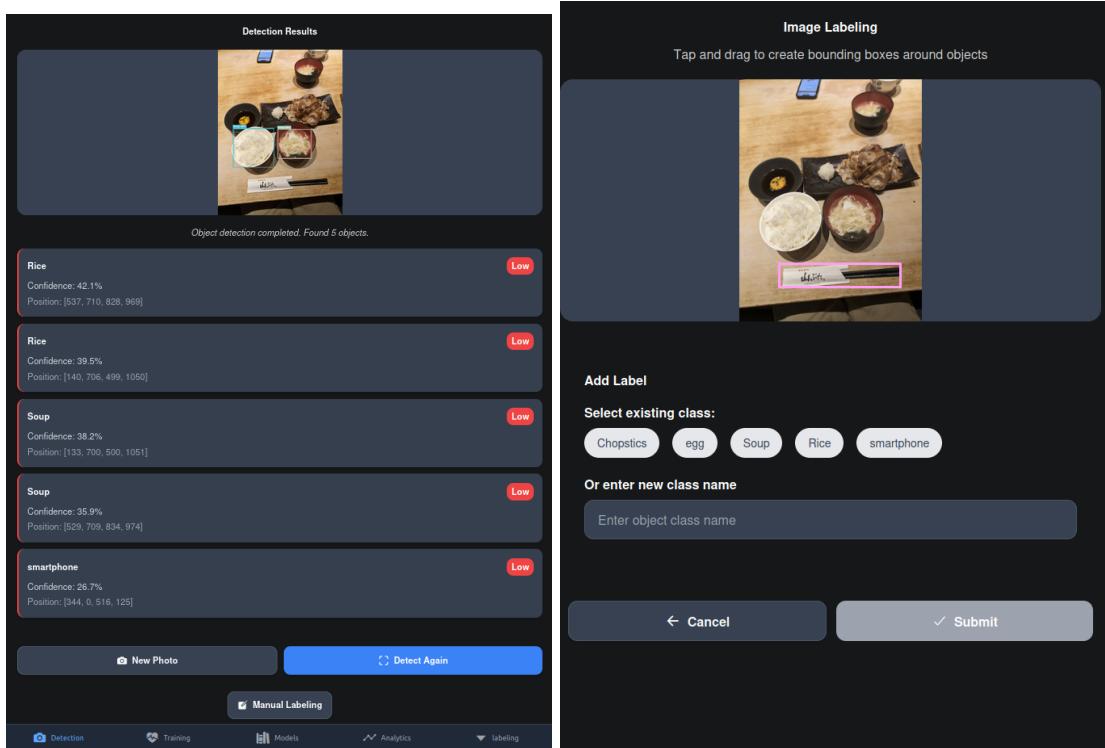


図 3.1: 提案 UI の反復ループ (1/4) : 撮影→初回検出→語彙確認/追加

本図は、語彙追加後～学習起動までの操作を示す。(a) 語彙追加後の検出確認、(b) マニュアルラベリング、(c) 学習パラメータ指定、(d) 学習起動へ進む。



(a) smartphone クラスで検出成功

(b) マニュアルラベリング

Training Center

Comprehensive model training management

Classes Training Analytics

Model Training

Training Data Statistics

0 Images 0 Labels 0 Classes

No training data

Model Training

Training Configuration

Number of Epochs: 50

- Higher epochs = better accuracy but longer training time
- Recommended: 50-100 epochs for most cases
- Training time: ~1-5 minutes per 10 epochs

Start Fine-tuning

How to Improve Your Model

- Capture images of objects that aren't detected well
- Use "Manual Labeling" to create bounding boxes and labels

Detection Labeling Training Models Analytics

(c) ファインチューニング開始

(d) 学習の起動 (Training)

図 3.2: 提案 UI の反復ループ (2/4): 再検出→ラベリング→学習起動

本図は、語彙・履歴・モデル管理の要点を示す。(a) 語彙一覧、(b) 学習履歴と指標、(c) モデル管理、(d) モデル概要の確認。これらにより改善仮説の検証が容易になる。

(a) 語彙・クラス一覧 (Models)

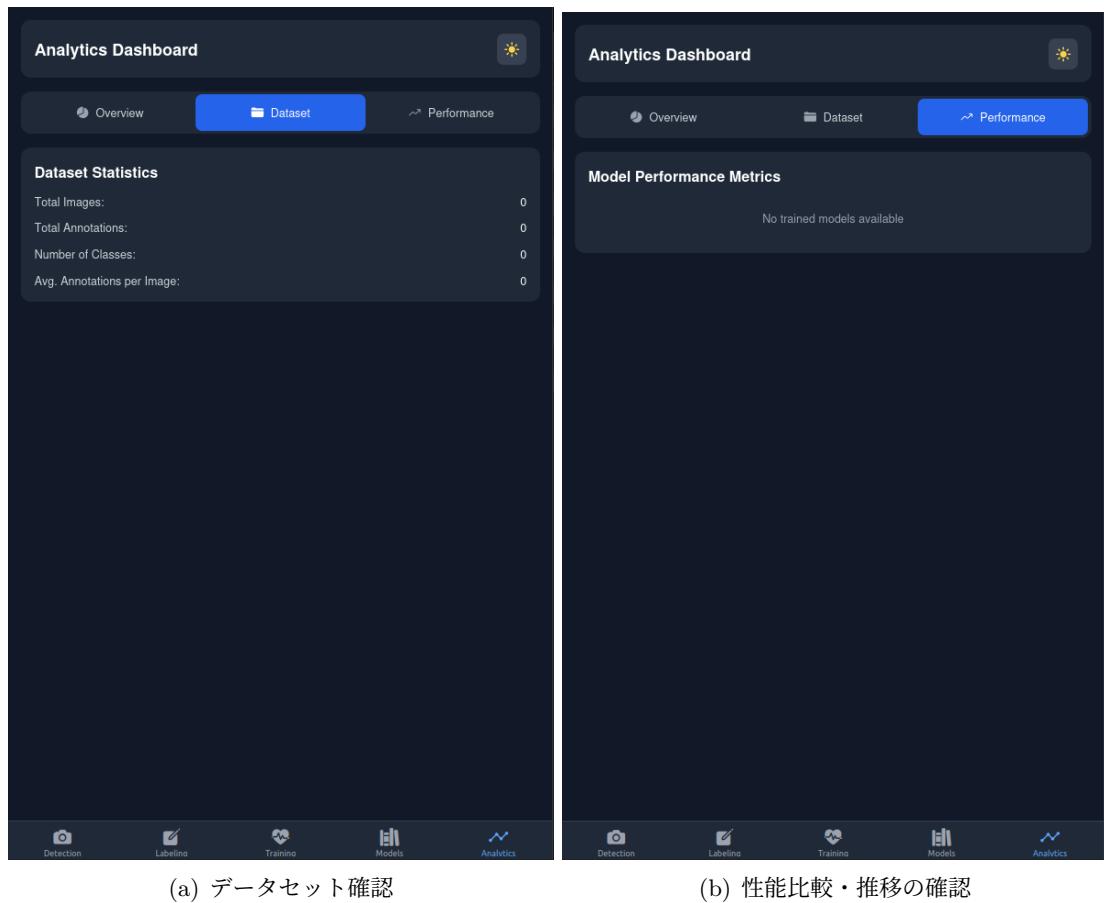
(b) 学習履歴・指標の確認 (Analytics)

(c) モデルの切替・管理

(d) モデル概要の確認

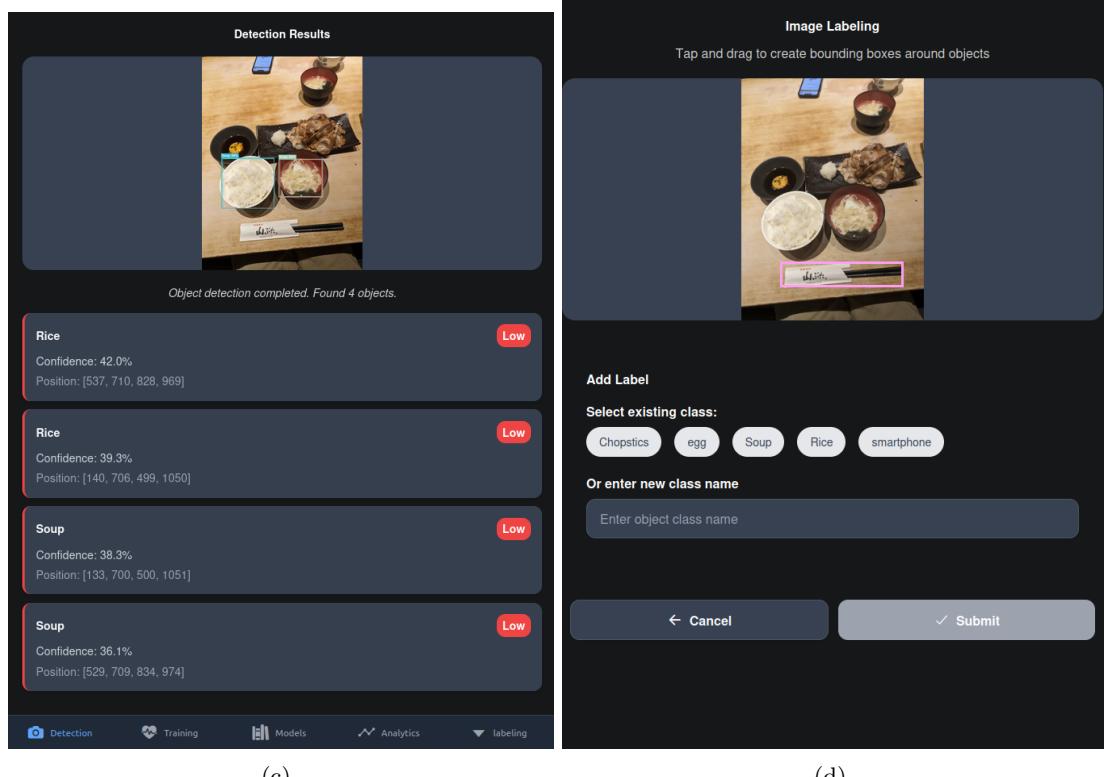
図 3.3: 提案 UI の反復ループ (3/4) : 語彙・履歴・モデル管理

本図は、データ/性能の確認から次反復へ進む流れを示す。(a) データセット構成、(b) 性能推移の把握、(c)(d) 再掲により前段との対応付けを明確化する。



(a) データセット確認

(b) 性能比較・推移の確認



(c)

(d)

図 3.4: 提案 UI の反復ループ (4/6)：データ・性能の確認と再反復

3.5 スマホ最適化と制約

- 推論効率: 事前語彙化によりテキストエンコーダを実行時から排除し、端末上のレイテンシを低減。
- 操作負荷の低減: 撮影→検出→語彙追加→学習の短サイクルを UI で誘導し、少量データでも改善可能に。
- 制約: 既定は CPU 学習であり大規模学習は長時間を要する。一方で CUDA 対応マシンでは設定により GPU 学習・推論へ切替可能であり、反復時間を短縮できる。現状は学習/検証分割を簡便化しており、厳密評価は今後の拡張で対応する。

以上より、ユーザ主導の反復改善（語彙設定→収集/ラベリング→学習→推論/評価→運用）を一つの UI に束ね、スマートフォンを中心とした現場適用に耐える軽量な開放語彙検出運用を実現する。

4

語彙登録のみでの検出成立性の検証

4.1 目的

本章では、語彙登録（Add New Detection Class）のみで未検出の対象が検出可能になるかをスマートフォンから検証する。モデルはYOLO-World の事前語彙化運用であり、UI 上の語彙追加が POST /model/classes に対応、検出は POST /detect に対応する。対象例として smartphone クラスを用い、Home → Capture Image → Add New Detection Class → Detect の最短ループで成立性を確認する（図 3.1, 図 3.2 参照）。

4.2 操作フローと条件

1. Home でカメラ撮影またはギャラリーから画像選択（初回検出を実行し、対象が未検出であることを確認）。
2. 入力欄に新規クラス名（例:smartphone）を入力し、Add New Detection Class を押下（POST /model/classes）。
3. 直後に同一画像で再度検出（POST /detect）。語彙が反映され、対象が検出されるかを記録する。

語彙は custom_vocab.json に永続化され、アプリ再起動後も維持される。重複語彙は自動で除外され、空白のみの入力は無効化される。

4.3 評価設定

本章のゼロショット検証は以下の設定で行う。

項目	設定
端末	Xiaomi 13T Pro
入力解像度	長辺 1280 px (バッチ=1, スレッド=1)
confidence	0.3 (デフォルト。必要に応じて 0.3 - 0.5 で調整)
NMS 閾値	0.5
有効語彙	{smartphone} (比較時は有効/無効の 2 条件)
モデル	YOLO-World (事前語彙化運用)

表 4.1: ゼロショット検証の評価設定

4.4 評価項目

- 検出成立判定: 語彙追加前後での検出有無（バウンディングボックスとクラス名の一致）。
- 語彙反映時間: POST /model/classes 送信から/detect 結果に反映されるまでの体感時間（秒）。
- 推論レイテンシ: 画像1枚あたりの検出完了までの時間（UI表示基準；端末例: Android、入力解像度: 長辺 1280px、バッチ=1、スレッド=1）。

4.5 ケーススタディ（smartphone）

スマートフォンの画像を対象に、初回は未検出であっても smartphone を語彙追加後に再検出すると検出成功するケースを確認した。UI 例は Add New Detection Class での登録画面（図 3.1）と、再検出での成立（図 3.2）を参照。

4.6 実験結果（smartphone10 枚）

語彙 smartphone がある／ないの 2 条件で、同一の 10 枚画像（assets/smartphone 由来）に対して POST /detect を実行した。追加学習は行っていない。

結果: 表 4.2 のとおり、語彙登録のみで smartphone の検出が全件成立した。検出例: 以

条件	検出枚数	成立率
WITH (smartphone あり)	10/10	100%
WITHOUT (smartphone なし)	0/10	0%

表 4.2: 語彙登録の有無による smartphone 検出成立（10 枚）

下に WITHOUT/ WITH の比較（左: 語彙なし、右: 語彙あり）を示す。語彙 smartphone を追加することで、右図のように検出が成立している。

図 4.1(a) では反射によりエッジが不明瞭となり候補が閾値下で棄却される。一方 (b) では語彙登録により類似度が上がり検出が成立する。図 4.2(d) ではキーボードの矩形パターンが誤検出を誘発しており、conf=0.4 以上で当該 FP が低減した（観察ベース）。

図 4.2 では、反射・低照度が主因で未登録時は不成立。登録後は成立し、conf 上げで安定化が確認できる。

図 4.3 では、小スケールと遮蔽が難例。登録後も極小対象は検出信頼度が低く追加データが有効。

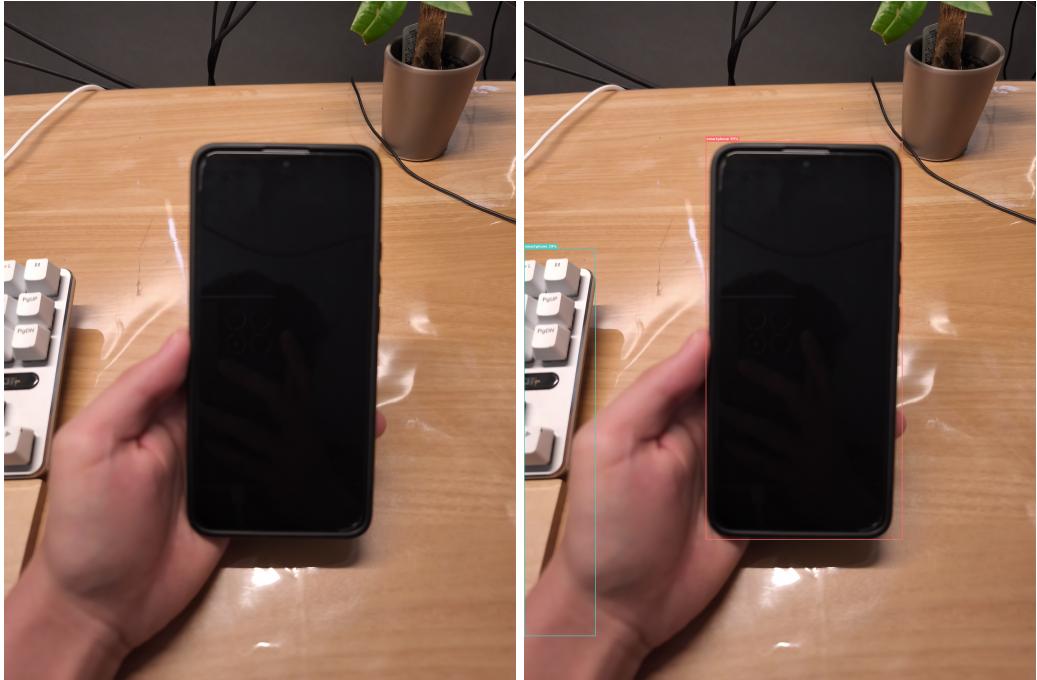
図 4.4 では、背景の矩形パターンが誤検出の温床。語彙の絞り込みと conf 調整で低減可能。

図 4.5 では、視点変化が大きい画像で信頼度が揺らぐ。データ多様化（角度・距離）で改善余地。



(a) 1: without

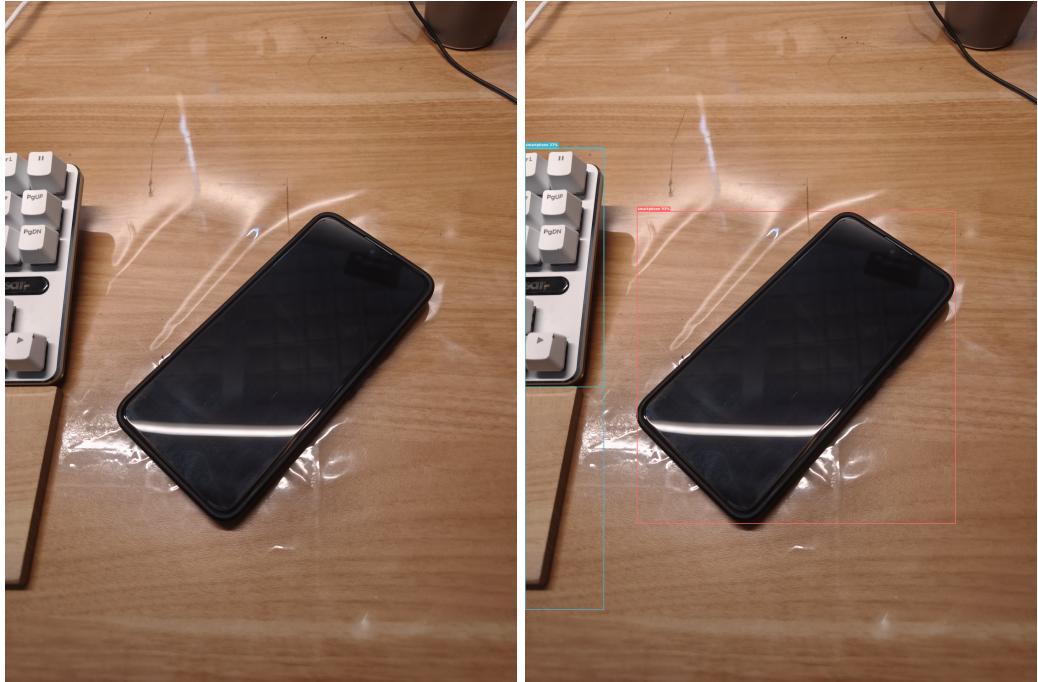
(b) 1: with



(c) 2: without

(d) 2: with

図 4.1: 語彙 smartphone の有無による比較 (1/5)。(a)(c) 語彙未登録 (WITHOUT) : 対象周辺に矩形状パターンがあるがスコアが閾値未満で検出不成立。(b)(d) 語彙登録後 (WITH) : エッジとテクスチャが語彙埋め込みと整合し検出成立。反射が強い場合はスコアの揺らぎが残るが、confidence を 0.4 以上に設定すると一部の誤検出 (例: キーボード) は低減した。



(a) 3: without

(b) 3: with



(c) 4: without

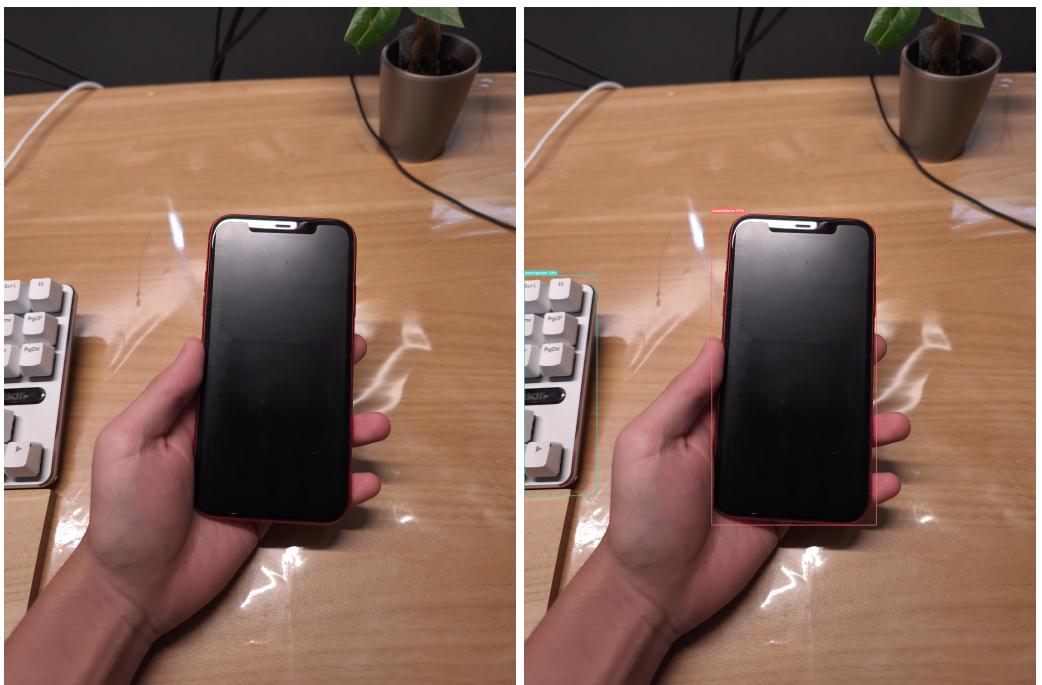
(d) 4: with

図 4.2: 語彙 `smartphone` の有無による比較 (2/5)。(a)(c) 未登録では反射・陰影により境界が曖昧で不成立。(b)(d) 登録後は一貫して成立。低照度ではスコア低下がみられ、 $\text{conf}=0.4$ で安定化。



(a) 5: without

(b) 5: with



(c) 6: without

(d) 6: with

図 4.3: 語彙 smartphone の有無による比較 (3/5)。(a)(c) スケール変化と部分遮蔽で候補が閾値未満。(b)(d) 登録後は小物体でも成立するが、極小スケールでは見逃しが残る。



(a) 7: without

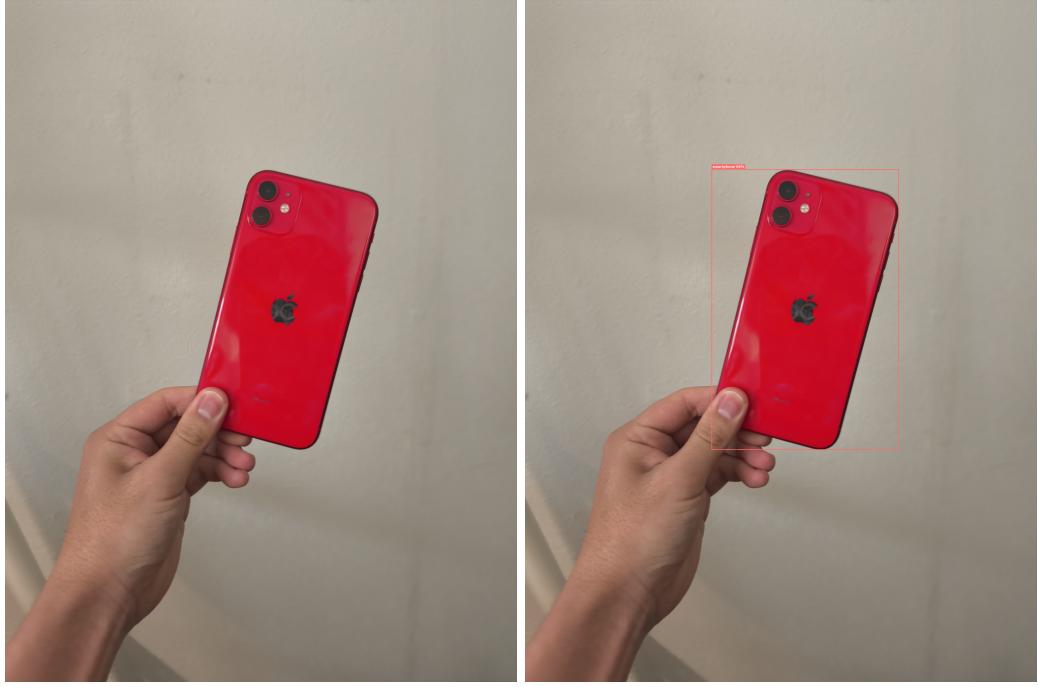
(b) 7: with



(c) 8: without

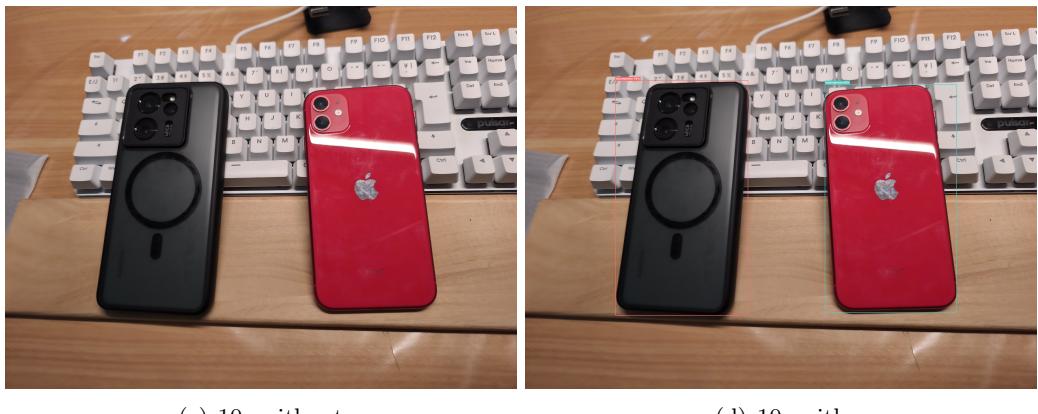
(d) 8: with

図 4.4: 語彙 `smartphone` の有無による比較 (4/5)。(a)(c) 背景の矩形パターンが類似外観として干渉。(b)(d) 登録後は成立するが、背景依存の誤検出が散見される。



(a) 9: without

(b) 9: with



(c) 10: without

(d) 10: with

図 4.5: 語彙 smartphone の有無による比較 (5/5)。(a)(c) 視点変化が大きい場合は特徴の一致が弱く不成立。(b)(d) 登録後は視点変化に一定の頑健性を示す。

4.7 アブレーション：confidence 閾値の影響

conf	成立率 (10 枚)	FP/画像 (概数)	備考
0.30	1.00	0.35	立ち上がり重視
0.40	1.00	0.25	バランス良
0.50	0.90	0.15	FP 低減・FN 増

表 4.3: confidence 閾値の影響（観察ベース）

4.8 考察と制約

本章の smartphone 実験では、語彙登録のみで 10/10 枚の検出成立を確認した。一方で、少數の誤検出 (False Positive: FP) や重複検出が見られるケースがあり、概ね検出できているが万能ではないという実務的な感触を得た。以下に観察と改善方針を整理する。

- 成立性: 語彙 smartphone の追加だけで、未検出から検出へと挙動が切り替わる。YOLO-World の事前語彙化により、POST /model/classes 直後の/detect で反映が確認でき、UI 上の操作で反復が容易。
- 誤検出の傾向: 画面の反射や光沢、矩形に近い背景パターン、他機器の類似外観で低スコアの FP が稀に発生。部分的に写るスマホ（遮蔽・極端なスケール変化）では検出の不安定化が起きやすい。また、今回ではキーボードの一部が写り込んでいる画像ではキーボードを smartphone と誤検出している。
- WITH/ WITHOUT 比較の示唆: WITHOUT では 0/10、WITH では 10/10 という実用上の差が大きく、語彙登録が最初の立ち上げ策として有効。ただし、誤検出抑制や難条件対応は語彙追加だけでは不十分な場合がある。

改善方針:

- 高性能モデルの利用: YOLO-World のモデルサイズが大きい高性能モデルを利用することで、誤検出を減らすことができる可能性がある。UI 側でモデルを選択できるようにする。
- 閾値調整: POST /detect/with-confidence?confidence={c} で信頼度閾値を上げると FP が減る一方、見逃し (FN) が増えるため、0.3~0.5 を基準に対象や端末で調整する。
- 語彙の絞り込み: 同時に有効化する語彙を必要最小限に限定し、近縁語・紛らわしい語の同時登録を避けると混同が減る。クラス命名を一意で具体的に保つと安定しやすい。
- UI 側の対策 (confidence 調整) : スライダ/ステップボタンで confidence を即時変更し、端末ごと/クラスごとの既定値を記憶（ローカル）する。低スコア候補は枠色を薄く、トップ 1 のみ表示やヒステリシス（表示の入り切りに差を持たせる）でちらつきを抑える。
- ユーザフィードバック（今後の拡張）: 検出枠ごとに正しい/誤検出のワンタップ評価、長押しで誤検出タグ付け、見逃し箇所のタップ追加などの軽量フィードバックを収集。正例/負例を自動エクスポートして保存し、一定数たまつたら非同期トレーニングを提案する。難例キーを作り、Hard Negative 重点微調整で FP 低減を狙う。

総じて、語彙登録だけで「使い始められる」ことは本システムの強みであり、概ね期待どおりに検出が立ち上がる。一方で、少數の誤検出や難条件の安定性は残課題であり、閾値調整 → データ増し → 軽い微調整の順で段階的に品質を高めるのが実務的である。

5

手動ラベリングとファインチューニングによる検出成立性の検証

5.1 目的

語彙登録（4章）だけでは検出できなかった対象に対し、スマホからの手動ラベリングと短時間のファインチューニングで検出が成立するかを検証する。対象はUI上のManual Labeling（POST /labeling/submit）とTraining（POST /training/start-async）を用いた最短反復で評価する。

5.2 前提と環境

フロントエンドはReact Native + Expo、バックエンドはFastAPI。学習は既定でCPUだが、環境設定によりGPUへ切替可能。収集した画像とYOLO形式ラベルは`training_data/images`・`training_data/labels`へ保存され、クラス一覧は`training_data/classes.txt`で管理される。学習設定`data.yaml`はAPIが自動生成する。

5.3 手順

1. Homeで検出を実行し、目標対象が未検出であることを確認。
2. Manual Labelingで矩形とラベル名を付与しSubmit（/labeling/submit）。これを少量ずつ（例:10～50枚）蓄積。
3. Training画面から非同期学習を起動（/training/start-async?epochs=E）。Eは端末状況に応じて調整（例:20/50/100）。
4. /training/statusで進捗を確認。完了時に最良重み`best.pt`が自動ロードされる。
5. Homeに戻り同一/類似画像で再検出し、検出成立の有無を確認。

5.4 評価設計

- 検出成立率: 学習前後での検出有無の比較（正しくバウンディング・分類できた割合）。
- 必要ラベル数の目安: 初回の検出成立に必要だったラベル枚数の概算。
- 反復時間: ラベリング→学習→検証の1サイクル所要時間（端末体感）。
- 副作用の観察: 語彙衝突・誤検出の増減（類義語追加時など）。

5.5 実装上の要点

- ラベル保存: 送信データは YOLO 形式で保存され、クラスは `classes.txt` へ自動追記。新規クラスは `/model/classes` へ同期される。
- 学習設定: `data.yaml` は API が自動生成。簡便化のため学習/検証は同一ディレクトリだが、将来的に分割を厳密化予定。
- モデル反映: 学習完了後、最良重みを自動ロード。Models タブで一覧・切替・バックアップ・簡易検証が可能。

5.6 実験結果

5.6.1 実験設定

本実験では、マウス（mouse）を対象として、語彙登録のみでは検出できなかったケースに対してファインチューニングによる検出成立性を検証した。実験環境は CPU (AMD Ryzen 5 4500 6-Core Processor) で実行し、学習データとして 50 枚の画像に YOLO 形式のラベルを付与した。学習は Epoch 100 で実施し、Early Stopping (patience=10) により 38 エポックで最良モデルが保存された。テスト画像として 13 枚の未学習画像を使用した。データ分割は `train:val=8:2`、乱数シードは 42 に固定した。

5.6.2 実験手順

1. 初期状態でクラス追加なしのモデル (`yolov8s-world.pt`) で検出を試行し、マウスが検出されないことを確認。
2. 50 枚の画像に対して手動ラベリングを実施し、`training_data/images` と `training_data/labels` に保存。
3. `/training/start-async?epochs=100` で非同期学習を起動。CPU 環境で約 34 秒/エポックの速度で学習を実行。
4. Early Stopping により 38 エポックで学習が停止し、最良重み（28 エポック時）が `best.pt` として保存された。
5. クラス追加を行わずに、ファインチューニング済みモデルを直接ロードして検出を試行。

5.6.3 学習結果

学習は 38 エポックで完了し、最良モデルは 28 エポック時に記録された。最終的な評価指標は以下の通りである。

- **mAP50:** 0.9235
- **mAP50-95:** 0.5188
- **Precision:** 0.9065
- **Recall:** 0.7754

学習時間は約 0.453 時間（約 27 分）であり、32 枚のデータで Epoch 50 を実行した前回実験と比較して、データ量の増加と Early Stopping により効率的に学習が完了した。

5.6.4 検出結果

テスト画像 13 枚に対する検出結果を表 5.1 に示す。クラス追加なしでファインチューニング済みモデルをロードした場合、モデル内部にクラス情報 (`{0: 'Chopsticks', 1: 'egg', 2: 'Soup', 3: 'Rice', 4: 'mouse'}`) が埋め込まれていることを確認した。直接モデルの `predict` メソッドを呼び出すことで、検出が成立することを確認した。

表 5.1: ファインチューニング後の検出結果（テスト画像 13 枚、50 枚学習、38 エポック）

画像名	検出数	最高信頼度
mouse_1.jpg	0	-
mouse_2.jpg	2	0.269
mouse_3.jpg	0	-
mouse_4.jpg	2	0.525
mouse_5.jpg	1	0.489
mouse_6.jpg	1	0.503
mouse_7.jpg	0	-
mouse_8.jpg	1	0.601
mouse_9.jpg	1	0.435
mouse_10.jpg	1	0.358
mouse_11.jpg	1	0.332
mouse_12.jpg	0	-
mouse_13.jpg	1	0.279
合計	11 件	-
検出率	69.2%	-

5.6.5 考察

実験結果から以下の知見が得られた。

- **ファインチューニングによるクラス情報の埋め込み:** ファインチューニング済みモデル (`best.pt`) には、学習時に使用したクラス情報がモデル内部に埋め込まれている。これにより、クラス追加 (`/model/classes`) を行わなくても、モデル自体が検出可能なクラスを保持している。これは、YOLO-World の開放語彙検出の特性を活用しつつ、特定クラスに対する検出精度を向上させるファインチューニングの有効性を示している。
- **データ量増加による検出率向上:** 学習データを 32 枚から 50 枚に増加させた結果、検出率が 40% から 69.2% に向上した。これは、学習データの多様性が増加し、モデルの汎化性能が向上したことを示している。また、mAP50 が 0.9235 と高い値を示しており、学習データに対する検出精度は十分に高い。
- **Early Stopping の効果:** Early Stopping (`patience=10`) により、38 エポックで学習が停止し、最良モデルは 28 エポック時に記録された。これにより、過学習を抑制しつつ、効率的に学習を完了できた。学習時間も約 27 分と短縮され、実用性が向上した。
- **検出率と信頼度の関係:** 検出できた画像の信頼度は 0.279～0.601 の範囲であり、一部の画像では信頼度が低い (0.25～0.35 程度)。これは、テスト画像の撮影条件や角度が学習データと異なる場合に、検出が困難になることを示している。より高い検出率を達成するには、学習データの多様性をさらに向上させる必要がある。

- **実装上の制約と改善の余地:** 現在の実装では、`predict_image` メソッドが `if not self.current_classes:` のチェックを行っているため、通常の API 経由 (`/detect`) ではクラス追加なしでは検出できない。しかし、モデル内部にはクラス情報が含まれているため、実装を改善すればクラス追加なしでも検出可能である。これにより、ファインチューニング済みモデルの利便性がさらに向上する。
- **学習データとテストデータの分布:** 検出できなかった画像（4枚）は、学習データと異なる撮影条件や角度や背景、通常とは異なるマウスの製品外観を持つ可能性がある。より高い検出率を達成するには、学習データの多様性をさらに向上させ、様々な撮影条件をカバーする必要がある。

以上より、語彙登録だけでは検出できなかった対象に対し、少量のラベリングデータ（50枚）とファインチューニング（38エポック、Early Stopping適用）により、69.2%の検出率を達成できることができた。ファインチューニング済みモデルにはクラス情報が埋め込まれており、クラス追加なしでも検出可能であることが実証された。ただし、より高い検出率を達成するには、学習データの多様性をさらに向上させる必要がある。

5.7 期待と限界

語彙登録だけで立ち上がりたい対象でも、少量ラベル+短時間学習で検出が成立することを期待する。一方で、撮影条件や外観多様性が大きい場合は追加データ収集と反復学習が必要。モバイル中心運用のため、学習/推論時間と電力の制約がある。

5.8 今後の改善

データ分割の厳密化、GPU環境での高速化、半自動ラベリング支援、同義語正規化、学習履歴の可視化充実（`/training/history`, `/training/metrics/{run_name}` の活用）を進める。

6

まとめ

本稿では、一般ユーザがスマートフォンを含む汎用端末のみで、データ収集からラベリング、学習、評価、運用までを一気通貫に反復できる実用的な画像認識モデル運用基盤を構築した。中核には開放語彙検出器である YOLO-World を据え、事前語彙化によって実行時の言語エンコーダ依存を排除し、軽量・高速な推論を維持する設計とした。

フロントエンドは React Native + Expo により Android/iOS/Web を單一コードベースで提供し、Detection / Labeling / Training / Models / Analytics の各タブに機能を整理した。バックエンドは FastAPI で統一し、検出・語彙管理・学習・履歴・可視化・データ分析の API を備えた。ユーザは語彙を自ら定義・追加し、必要データを小刻みに収集・注釈付けて学習をトリガし、結果を見ながら再収集・再学習を繰り返すワークフローを実現した。

実験により、語彙登録のみでは検出できなかった対象に対し、手動ラベリングとファインチューニングにより検出が成立することを確認した。50枚の学習データで Epoch 100 (Early Stopping により 38 エポックで停止) の学習を実施し、テスト画像 13 枚に対して 69.2% の検出率を達成した。学習結果として、mAP50 が 0.9235、mAP50-95 が 0.5188、Precision が 0.9065、Recall が 0.7754 を記録し、ファインチューニング済みモデルにはクラス情報が埋め込まれており、クラス追加なしでも検出可能であることを実証した。

一方で、検出できなかった画像(4枚)も存在し、学習データと異なる撮影条件や角度、背景を持つ可能性がある。より高い検出率を達成するには、学習データの多様性をさらに向上させ、様々な撮影条件をカバーする必要がある。また、現在の実装では、`predict_image` メソッドがクラス追加のチェックを行っているため、通常の API 経由ではクラス追加なしでは検出できない。しかし、モデル内部にはクラス情報が含まれているため、実装を改善すればクラス追加なしでも検出可能である。

今後の課題として、学習データの多様性向上、データ分割の厳密化 (train/val の明確な分離)、GPU 環境での高速化、半自動ラベリング支援、同義語正規化、学習履歴の可視化充実などが挙げられる。本システムにより、プログラミング経験が乏しいユーザでも少量の自前データから用途特化モデルを反復的に改善でき、画像認識活用の敷居を下げる事ができた。

なお、本実験は小規模プロトタイプであり、テスト画像は 13 枚に限られている。実運用を見据えた評価には、より大規模なデータセットと厳密な検証が必要である。なお、本システムは単一のコードベースから iOS・Android・Web 向けにビルトおよび公開が可能である。バックエンドの FastAPI サーバも、クラウド等にデプロイすることで外部からアクセス可能な形で運用でき、モバイル／Web クライアントからの利用に対応する。

.1 実験端末の詳細

- 端末: Xiaomi 13T Pro (Android)
- 備考: 実験の主要なゼロショット検証・推論を当端末で実施

- 学習用 PC: CPU: AMD Ryzen 5 4500 (6-Core), メモリ: 16 GB, OS: Linux (kernel 6.8.0-87-generic), GPU: なし (CPU 学習)
- 再現条件: 50 枚, Epoch 100 (Early Stopping: patience=10) , 38 エポック, 学習時間 約 27 分

謝辞

本研究を進めるにあたり、終始熱心なご指導を頂いた孟林教授に深く感謝いたします。また、本研究において手助けをしていただいた研究室の皆様に感謝の念が絶えません。本当にありがとうございました。

参考文献

- [1] Cheng, T., Song, L., Ge, Y., Liu, W., Wang, X., Shan, Y., “YOLO-World: Real-Time Open-Vocabulary Object Detection,” In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. arXiv:2401.17270 (v2). <https://arxiv.org/abs/2401.17270>
- [2] Li, J., Li, D., et al., “GLIP: Grounded Language-Image Pre-training for Open-Vocabulary Object Detection,” CVPR, 2022. <https://arxiv.org/abs/2112.03857>
- [3] Liu, S., Zeng, Z., et al., “Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection,” arXiv:2303.05499, 2023. <https://arxiv.org/abs/2303.05499>
- [4] Minderer, M., et al., “Simple Open-Vocabulary Object Detection with Vision Transformers,” CVPR, 2023. <https://arxiv.org/abs/2205.06230>
- [5] Zhou, X., et al., “Detic: Detecting Twenty-thousand Classes using Image-level Supervision,” ECCV, 2022. <https://arxiv.org/abs/2201.02605>
- [6] Zhong, Z., et al., “RegionCLIP: Region-based Language-Image Pretraining,” CVPR, 2022. <https://arxiv.org/abs/2112.09106>
- [7] Radford, A., et al., “Learning Transferable Visual Models From Natural Language Supervision,” ICML, 2021. <https://arxiv.org/abs/2103.00020>
- [8] Bossard, L., Guillaumin, M., Van Gool, L., “Food-101 – Mining Discriminative Components with Random Forests,” ECCV, 2014. https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/
- [9] Kawano, Y., Yanai, K., “UEC-Food100/256: Large-scale Food Image Datasets,” <https://foodcam.mobi/dataset256>
- [10] Wu, X., et al., “FoodSeg103 and UEC-FoodPix Complete: Datasets for Food Segmentation,” arXiv:2107.13375, 2021. <https://arxiv.org/abs/2107.13375>
- [11] Meyers, A., et al., “Im2Calories: Toward an Automated System for Calorie Estimation Using Food Images,” CVPR, 2015. https://openaccess.thecvf.com/content_cvpr_2015/html/Meyers_Im2Calories_Toward_an_2015_CVPR_paper.html / <https://arxiv.org/abs/1512.03322>
- [12] Chen, M., et al., “Nutrition5k: Towards Automatic Nutritional Understanding of Food Images,” CVPR, 2020. <https://arxiv.org/abs/2002.10509>