令和 5 年度茨城大学大学院理工学研究科情報工学専攻

修士学位論文

CLIP を利用した人工衛星画像の Zero-Shot 物体検出

所属　情報工学専攻

著者　余世豪 (22NM750S)

指導教員　新納浩幸教授

令和 5 年 2 月 2 日 (金)

令和 5 年度茨城大学工学部情報工学科卒業研究

CLIP を利用した人工衛星画像の Zero-Shot 物体検出

著者

余世豪 (22NM750S)

指導教員

新納浩幸教授

論文要旨

本論文は、人工知能分野における物体検出技術、特に人工衛星画像分析のアプリケーションにおける革新的なアルゴリズムに焦点を当てている。まず、R-CNN、YOLO、SSD などの伝統的な物体検出方法を探究し、これらの方法が複雑または大きなサイズの画像を処理する際に持つ限界を指摘している。これらの問題を克服するために、研究では CLIP（Contrastive Language‐Image Pre-training）モデルが導入された。このモデルは、言語と視覚情報を組み合わせることで、訓練サンプルがない状態での物体検出を実現する。研究の焦点は、CLIP モデルのゼロショット（zero-shot）検出能力と、大規模人工衛星画像上での目標認識能力にある。CLIP モデルをファインチューニング（Fine-tuning）することにより、人工衛星画像分類などの分野における能力を効果的に向上させることができる。実験結果は、複数のデータセット上で、ファインチューニングされた CLIP モデルが完全監督の ResNet-50 モデルを上回り、特に細粒度分類タスクで顕著なパフォーマンスを示すことを示している。

CLIP モデルの衛星画像認識の精度を向上させるために、研究では、ラベル付きの衛星画像データセットを使用してモデルをファインチューニングし、衛星画像に特化した認識ツールにすることを目指している。また、論文では、このようなモデルを簡単にファインチューニングする方法が提供されている。

任意のサイズ（225×225 から 10000×10000）の衛星画像に対する物体検出のために、論文では新しい画像処理方法が使用される。この方法は、YOLO モデルに触発され、1 枚の画像を複数の小さなブロックに分割し、それぞれのブロックに CLIP モデルを適用するものた。そして、この方法を用いて、異なるサイズの 10 枚の衛星画像に対するテストが行われている。

Master's Thesis in Scholastic 2023, Major in Computer and Information Sciences, Granduate School of Science and Engineering, Ibaraki University

Zero-Shot Object Detection in Satellite Images with CLIP

Author : YU SHIHAO (22NM750S)

Adviser : Prof. Hiroyuki Shinnou

Abstract

This paper focuses on the technology of object detection in the field of artificial intelligence, specifically innovative algorithms in the application of analyzing satellite imagery. Initially, the study explores traditional object detection methods such as R-CNN, YOLO, and SSD, pointing out their limitations in handling complex or larger-sized images. To overcome these issues, the study introduces the CLIP (Contrastive Language–Image Pre-training) model, which, by integrating language and visual information, is capable of performing object detection without the need for training samples.

The research emphasizes the zero-shot detection capabilities of the CLIP model and its ability to recognize targets in large-scale satellite imagery. By fine-tuning the CLIP model, its performance in tasks like satellite image classification can be significantly improved. Experimental results show that the fine-tuned CLIP model outperforms the fully supervised ResNet-50 model in object detection across multiple datasets, especially in fine-grained classification tasks.

To enhance the accuracy of the CLIP model in satellite image recognition, the study employed a labeled satellite image dataset for fine-tuning, turning it into a specialized recognition tool for satellite imagery. The paper also provides methods for this simple fine-tuning of the model.

For object detection in satellite images of any size (ranging from 225×225 to 10000×10000), the paper introduces a novel image processing method inspired by the YOLO model. This method involves dividing an image into several smaller sections and applying the CLIP model to each section. Additionally, this method was tested on 10 satellite images of various sizes.

目次

第 1 章
序論

人工知能とコンピュータビジョン技術の急速な発展に伴い、物体検出は重要な研究分野になっている。R-CNN、YOLO、SSD などの従来の物体検出方法は、標準的な画像データセットの処理には優れているが、より複雑または動的な環境の画像を処理する際には、モデルの一般化能力と多様なデータへの適応性に制限がある。これに対し、CLIP（Contrastive Language–Image Pre-training）モデルは、言語と視覚情報を組み合わせることで、物体検出分野に新たな可能性をもたらした。

CLIP モデルは、マルチモーダル学習領域で優れた成果を上げるだけでなく、ゼロショット学習（zero-shot learning）にも長けている。この能力により、モデルは訓練中に未だ見たことのないカテゴリを処理し認識することができ、新規性と多様なデータへの適応性を大幅に向上させている。これは従来の物体検出モデルでは実現が困難であり、通常は大量のラベル付きデータを必要とする。また、CLIP モデルのもう一つの特徴は、ファインチューニングへの適応性た。ファインチューニングにより、CLIP は特定のアプリケーションシナリオにより正確に適応し、特定のタスクでのパフォーマンスをさらに向上させることができる。この柔軟性により、CLIP モデルは多様で動的に変化する物体検出タスクに対して、より実用的で効果的た。

本研究は、人工衛星画像における CLIP モデルのゼロショット物体検出への応用を探求することを目的としている。特に、CLIP モデルのマルチモーダル学習能力、ゼロショット認識能力、およびファインチューニング後の大規模衛星画像における目標識別能力に焦点を当

てる。従来の物体検出方法は、大量のラベル付きデータを必要とし、既知のカテゴリの物体にのみ適用可能た。しかし、人工衛星の画像に関しては、画像のサイズが大きく、データが複雑で、ラベリングが困難であるため、従来の物体検出方法はうまく適用できません。深層学習技術の発展に伴い、ゼロショット物体検出は一つの学習方法として登場した。そのため、遠隔センシング画像における物体検出を実現するためのゼロショット学習技術の使用は、理論的および実践的な意義がある。

第 2 章

関連研究

2.1 研究背景

コンピュータビジョンにおいて、画像を理解するためには主に 3 つの側面がある。

・画像分類（Classification）、画像を特定のカテゴリーの情報に構造化することで、事前に決定されたカテゴリー（文字列）やインスタンス ID を使って画像を記述する。応用分野では、顔認識、シーン認識などが分類タスクに分類される。

・物体検出（Detection）は、特定の物体の目標に焦点を当て、その目標のカテゴリー情報と位置情報を同時に取得するタスクた。分類と比較して、検出は画像の前景と背景に対する理解を提供し、背景から興味のある目標を分離し、その目標の記述（カテゴリーと位置）を特定する必要がある。

・分割（Segmentation）には、意味的分割（semantic segmentation）とインスタンス分割（instance segmentation）が含まれる。前者は前背景の分離を拡張し、異なる意味を持つ画像部分を分離することを要求する。後者は検出タスクを拡張し、目標の輪郭を記述すること（検出ボックスよりも精密）を要求する。分割は画像のピクセルレベルの記述であり、各ピクセルにカテゴリー（インスタンス）の意味を与える。これは、無人運転での道路と非道路の分割など、理解の要求が高いシーンに適用される。

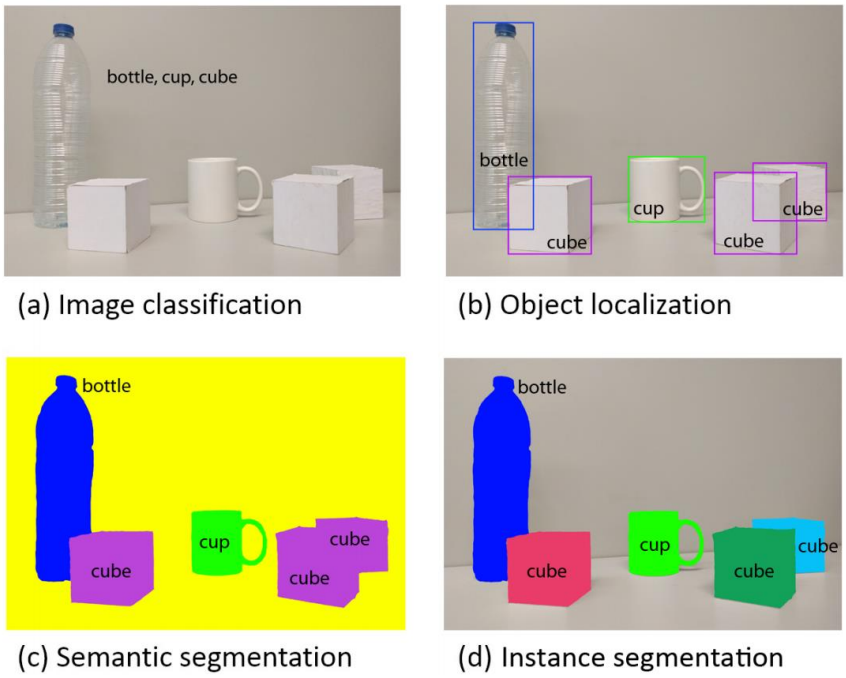

図 1.1：コンピュータビジョンの 3 つのレベル

本論文は主に物体検出（Object Detection）に焦点を当てています。

2.2　物体検出（Object Detection）

2.2.1 物体検出モデル

物体検出（Object Detection）は、コンピュータビジョン分野の重要な研究方向であり、コンピュータが人間のように写真やビデオの中のさまざまな物体を認識し理解することを目的としている。具体的には、物体検出は画像中の物体を認識すること、つまり「何であるか」に加えて、それらの物体の正確な位置を特定すること、つまり「どこにあるか」も含まれている。これは通常、画像内の物体の bounding box（Bounding Box）を識別することによって実現される。

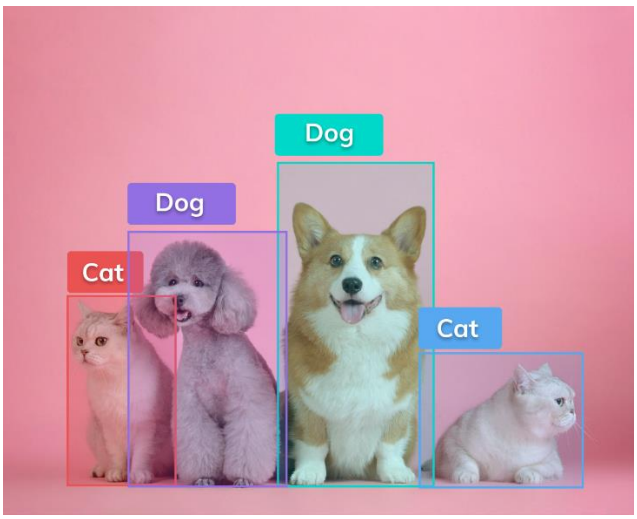

図 1.2：物体検出の例

コンピュータビジョン分野の核心的な任務の一つとして、その発展過程は多くの技術の革新と進化を見せている。初期の物体検出方法は、エッジ検出や領域分割などの伝統的な画像処理技術に大きく依存していた。深層学習の台頭とともに、畳み込みニューラルネットワーク（CNN）を基にした方法がこの分野をリードするようになりた。例えば、R-CNN [2]及びその変種は、特徴を抽出し分類器を使用して異なるオブジェクトを識別することにより、検出の精度を大幅に向上させた。そして、YOLO [3]や SSD のような方法は、一段階の検出アーキテクチャを通じて、速くて効率的な物体検出を実現している。

以下は、これら 4 つの主要な物体検出モデルについての紹介た。

表 1 ：4 種類のモデルの比較

モテル	メリット	デメリット
R-CNN	高精度検出、明確な物体の位置特定	処理速度が遅く、訓練プロセスが煩雑で、大量のストレージスペースを占有する

Fast R-CNN	R-CNN と比較して処理速度が速く、訓練プロセスが簡略化され、より高い検出精度を実現する	外部領域提案アルゴリズムに依存し、速度を制限している
YOLO	非常に速い検出で、リアルタイム処理に適しており、背景の誤検出の可能性を減らす	小物体や群れの中の物体の処理時の精度が低い
SSD	小物体に対して良好な認識能力を持つ高速な検出で、異なるサイズの物体検出に適している	複雑なシーンでの精度が大型モデルに劣る場合がある

2.2.2 Zero Shot モデル

ゼロショット（Zero Shot）[6]物体検出分野では、いくつかの重要な研究とモデルがある。これらのモデルは、通常、深層学習、特に畳み込みニューラルネットワーク（CNN）とグラフニューラルネットワーク（GNN）、そして意味論的埋め込み技術を組み合わせ、訓練過程で一度も見たことのない物体を識別し、位置を特定することを目指している。

・ZSD (Zero-Shot Detection)：これは基本的なゼロショット検出方法で、通常、視覚-意味論的埋め込みを使用して視覚的特徴とカテゴリ記述の間のギャップを埋めるのに依存している。

・DeFRCN (Deformable Fully Convolutional Networks)：この方法は、可変形の畳み込みネットワークを使用し、ゼロショット学習の技術と組み合わせて検出精度を向上させます。

・PL (Prototype Learning)：この方法は、各カテゴリの原型（つまりカテゴリの典型的な代表）を学習し、これらの原型を使用して新しい、未見のカテゴリを識別する。

・CLIP [1] (Contrastive Language‐Image Pre-training)：CLIP はもともと画像とテキストのマッチングのために設計されたが、ゼロショットシナリオでは強力な一般化能力を示し、訓練中に見たことのないカテゴリを識別するのに使用できます。

・ZSD-GAN (Zero-Shot Detection with Generative Adversarial Networks)：この方法は、生成的敵対ネットワーク（GAN）を使用して未見カテゴリの合成サンプルを生成し、これによりモデルがこれらのカテゴリを識別する方法を学習するのを助けます。

2.2.3 Zero Shot の実現方法

ゼロショット（Zero-Shot）物体検出の具体的な実装方法は、視覚的特徴と意味情報を組み合わせることに関連している。モデルは訓練中に一度も見たことのない物体を識別し、位置を特定できるようになる。視覚特徴の抽出、意味的埋め込み、そしてこれら二つの組み合わせを含む。

・視覚的特徴抽出

訓練画像から視覚的特徴を抽出する必要がある。これは通常、事前訓練された深層畳み込みニューラルネットワーク（ResNet、VGG など）を通じて行われる。画像 I に対して、抽出された視覚的特徴は $f(I)$ として表される。

・意味的埋め込み

各目標カテゴリに対して意味的な説明が必要。これらの説明は、カテゴリの属性やテキストの説明などがある。word2vec、GloVe などの埋め込みモデルを通じてベクトル形式に変換される。カテゴリ C に対して、その意味的埋め込みは $s(C)$ として表される。

・視覚と意味の整合

目標は、視覚的特徴と意味的埋め込みを整合させることた。これは通常、マッピング関数 M を介して行われ、この関数の目的は視覚的特徴を意味的埋め込みと同じ空間にマッピングする。このマッピング関数は、以下の目的関数を最適化することにより訓練される：

$$min_M \sum_{(I,C)\in D} \| M\big(f(I)\big) - s(C) \|^2$$

ここで、 D は画像とその対応するカテゴリを含む訓練セットだ。

・ゼロショット検出

テスト時には、新しい画像 I' と一連の未見のカテゴリ $\{C'\}$ が与えられ、モデルはまず画像の視覚的特徴 $f(I')$ を抽出する。次に、マッピング関数 M を通じてこれらの特徴を意味的空間にマッピングし、 $M\big(f(I')\big)$ を得ます。モデルは次に、意味的空間で $M\big(f(I')\big)$ に最も近いカテゴリ埋め込み $s(C')$ を探し、これを識別結果として使用する。

2.3 まとめ

この章では、従来の物体検出モデルについて紹介している。これらの従来の方法は、複雑または動的な環境を処理する際に限界がある。通常、多くのラベル付けされたデータが訓練に必要であり、見たことのないオブジェクトのカテゴリーを処理する際には効果が低いた。これらの制限を克服するために、マルチモーダル学習が徐々に発展してきた。この方法は、視覚データと言語データを組み合わせることで、モデルの汎化能力と適応性を高めている。

この背景の下で、CLIP（Contrastive Language–Image Pre-training）モデルが登場した。CLIP モデルは、その比較学習メカニズムにより画像とテキストデータを組み合わせ、画像コンテンツの理解を向上させている。画像内のさまざまなオブジェクトをよりよく理解し、特に正確な識別のために詳細なテキストの説明が必要なオブジェクトを認識することができた。CLIP はまた、事前訓練モデルでもあり、訓練過程で約 4 億ペアの画像-テキストデータを使用し、さまざまなタイプの画像に対して強力な汎化能力を発揮する。これにより、見たことがない、または訓練データと視覚的に大きく異なる画像においても、効果的な物体検出を行うことができた。モデルは特にゼロサンプルまたは少サンプルの学習シナリオに適している。物体検出タスクでは、モデルの微調整により、大量の特定カテゴリーの訓練データがなくても、CLIP は新しいカテゴリーのオブジェクトを識別し検出することができ、新しいタスクや環境に迅速に適応するのに非常に役立ちます。次の章では、CLIP モデルについて詳しく紹介する。

第 3 章

CLIP

3.1 概要

CLIP [1]（Contrastive Language‐Image Pre-training）モデルは、人工知能分野で革新的な方法であり、視覚理解と言語理解の間のギャップを効果的に埋めている。これは、大規模な画像およびテキストの見出しデータセットを利用して、自然言語の監督から直接視覚的概念を学習することによって達成される。その核心原理は、データセット内で正しい画像とテキストのペアを予測することを通じて学習することた。訓練過程では、一連の画像とテキストが提示され、正しいペアリングを関連付けることを学習する。これは、比較学習方法を通じて実現され、モデルは対応する画像-テキストペアの特徴の類似性を最大化し、非対応ペアの特徴の類似性を最小化するように訓練されている。多様な画像とテキストでの訓練を通じて、CLIP は自然言語で表現される視覚的概念に対する強力な理解を開発した。これにより、モデルは、訓練中に一度も見たことのない画像を、ただテキストの説明に基づいて識別し分類する、ゼロショット（Zero-Shot）学習など、さまざまなタスクを実行する能力を持つようになった。

3.2 CLIP モデルの原理

3.2.1 二層ストリーム アーキテクチャ

CLIP は、二層ストリーム（double-stream）アーキテクチャを採用しており、2 つの主要な

部分で構成されている：画像エンコーダー（Image encoder）とテキストエンコーダー（Text encoder）。これら 2 つのエンコーダーはそれぞれ画像データとテキストデータを処理する。

・画像エンコーダー：通常、事前訓練された畳み込みニューラルネットワーク（例えば ResNet）や視覚 Transformer（ViT）であり、画像の特徴表現を抽出するために使用される。

・テキストエンコーダー：通常、Transformer ベースのモデルであり、テキストデータの特徴表現を抽出するために使用される。

3.2.2 対照学習

「正例」（類似または関連するデータポイント）と複数の「負例」（不類似または関連しないデータポイント）を選択する。CLIP モデルでは、一つの画像とそれに対応する説明テキストが正例を構成し、その画像と他の非対応テキストが複数の負例を構成する。

モデルはデータを特徴空間にマッピングすることを学習し、この空間では、正例間の距離が小さく、負例間の距離が大きくなります。

特徴空間内の点の類似性を計算するために、コサイン類似度（Cosine Similarity）などが使用される。

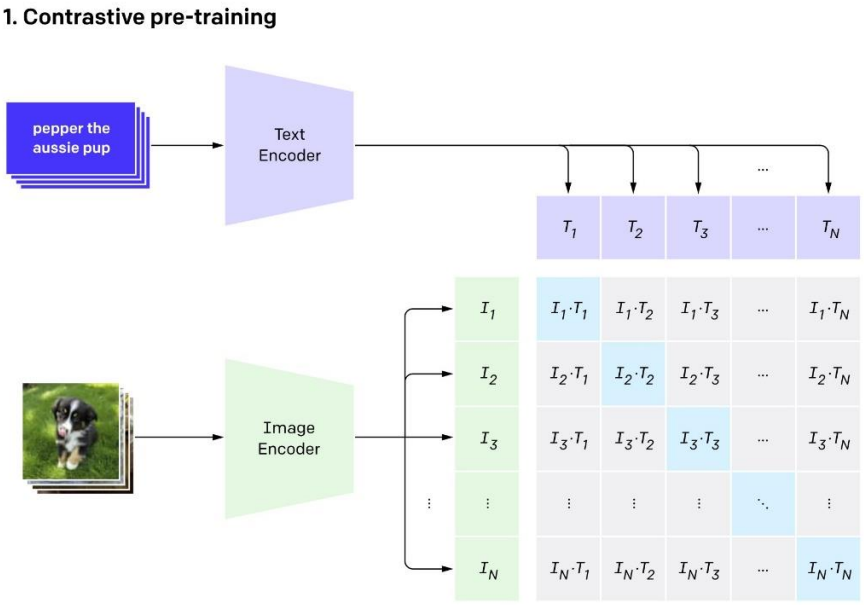

図 3.1：CLIP の対照学習

3.3.3 損失関数

CLIP は通常、対照的な損失（contrastive loss）、例えば対照的学習損失（InfoNCE loss）を使用する。この損失関数の目的は、正例ペアの類似性を最大化し、負例ペアの類似性を最小化することた。N 組の画像とテキストのバッチがあると仮定すると、損失関数は以下のように表現される：

$$L = -\sum_{i=1}^{N} \log \frac{\exp\left(sim(I_i, T_i)/\tau\right)}{\sum_{j=1}^{N} \exp\left(sim(I_i, T_i)/\tau\right)}$$

・ I_i と T_i はそれぞれ第 i 組の画像とテキストの特徴表現た。

・ $sim(.,.)$ は類似性関数で、点積やコサイン類似度（Cosine Similarity）などがある。

・ τ は温度パラメーターで、類似性スコアのスケーリングを制御する。

この損失関数は、モデルが正例ペア（つまり、各画像とそれに対応するテキスト）の類似

性スコアを向上させるように促する。同時に、バッチ内の他のすべてのテキスト（つまり負例）と各画像の類似性スコアを低下させます。この方法により、モデルは関連する画像とテキストを特徴空間内で近づけ、関連しないものを分離することを学習する。

3.3 モデル構造

3.3.1 データ収集

OpenAI 社は、4 億以上の画像-テキストペアからなるデータセット WIT（Web Image Text）を公式に収集した。さまざまなシナリオでのカバレッジを可能な限り高めるために、WIT はまず、英語版ウィキペディアで 100 回以上出現する単語を使って 50 万のクエリを構築し、WordNet を使用して同義語での置換を行いました。データセットのバランスを実現するために、各クエリにつき最大 2 万の検索結果が取得されました。

3.3.2 テキスト画像エンコーダー

CLIP は、基本モデルとして ResNet-50 [4]残差ネットワークを採用し、いくつかの調整を加えている。ブラー・プーリング（Blur Pooling）の導入、ブラー・プーリングの核心は、ダウンサンプリングの前にガウスの低域通過フィルターを加えることた。グローバル・アベレージ・プーリング（Global Average Pooling）をアテンション・プーリングに置き換え、ここでのアテンションは Transformer で紹介されている自己アテンション（Self-Attention）を使用している。

CLIP が採用する Transformer には 5 つのグループがあり、ResNet-50、ResNet-100、EfficientNet のアイディアに基づいて ResNet-50 を 4 倍、16 倍、64 倍に拡大したモデルで、

それぞれ ResNet-50x4、ResNet-50x16、ResNet-50x64 と表される。

画像エンコーダーの別の選択肢は ViT で、ViT では ViT-B/32、ViT-B/16、ViT-L/14 の 3 つのモデルが共同で訓練されました。テキストエンコーダーには、12 層、512 の隠れ層ノード数、および 8 つのヘッドを持つ Transformer が使用されている。

3.3.3 アルゴリズム

CLIP の中核となる考え方は、画像とテキストを同じ特徴空間にマッピングすることた。画像エンコーダーは画像を特徴空間にマッピングするために使用され、テキストエンコーダーはテキストを同じ特徴空間にマッピングするために使用される。

図 3.1 に示すように、モデル訓練中に取得される各バッチは N 個の画像-テキストペアで構成される。これら N 個の画像を画像エンコーダーに送ると、 N 個の画像特徴ベクトル $(I_1, I_2, I_3, \cdots I_N)$ が得られる。同じ、これら N 個のテキストをテキストエンコーダーに送ると、 N 個のテキスト特徴ベクトル $(T_1, T_2, T_3, \cdots T_N)$が得られる。対角線上の画像とテキストのペアのみが一致するため、CLIP の訓練目標は、画像・テキストペアの特徴ベクトル間の類似度を可能な限り高くし、ペアでないものの類似度を可能な限り低くすることた。ここでの類似度の計算にはベクトル内積が使用される。この方法により、CLIP は N 個の正のサンプルと N-N 個の負のサンプルから成る損失関数を構築する。また、異なるエンコーダーから出力される特徴ベクトルの長さが異なるため、CLIP は線形マッピングを使用して、2 つのエンコーダーから生成される特徴ベクトルを統一された長さにマッピングする。計算過程の擬似コードは以下の通る。

```
# マルチモーダルの特徴を抽出
I_f = image_encoder(I) #[n, d_i]
```

```
T_f = text_encoder(T) #[n, d_t]
# マルチモーダルの特徴を空間ベクトルにマッピングする
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)
#コサイン類似度を計算する
logits = np.dot(I_e, T_e.T) * np.exp(t)
# 損失関数を構築する
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t    = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```

3.4 Zero-Shot 学習

CLIP は優れた Zero-Shot [5] [6]学習の性能を有している。Zero-Shot Transfer は以下の図 4 に示されている。この段階では、CLIP の事前訓練済みの Image Encoder と Text Encoder を使用して Zero-Shot Transfer を行いる。例えば、ImageNet-1K の検証セットの画像があり、CLIP の事前訓練済みのモデルがこの分類タスクを完了できることを期待する。しかし、この Image Encoder は直接分類タスクを行うことはできません。そこで、CLIP は以下の Prompt Template モードを採用している。

この画像を CLIP の事前訓練済みの Image Encoder に入力し、特徴 I を得ます。次に、すべてのカテゴリーの単語「cat」、「dog」などをプロンプト「A photo of a {object}」として作成し、このプロンプトを CLIP の事前訓練済みの Text Encoder に入力し、順に特徴 123 を得ます。最後に、どの特徴の余弦類似度が I と最も高いかを見て、その画像がどのカテゴリーに

属するかを判断する。

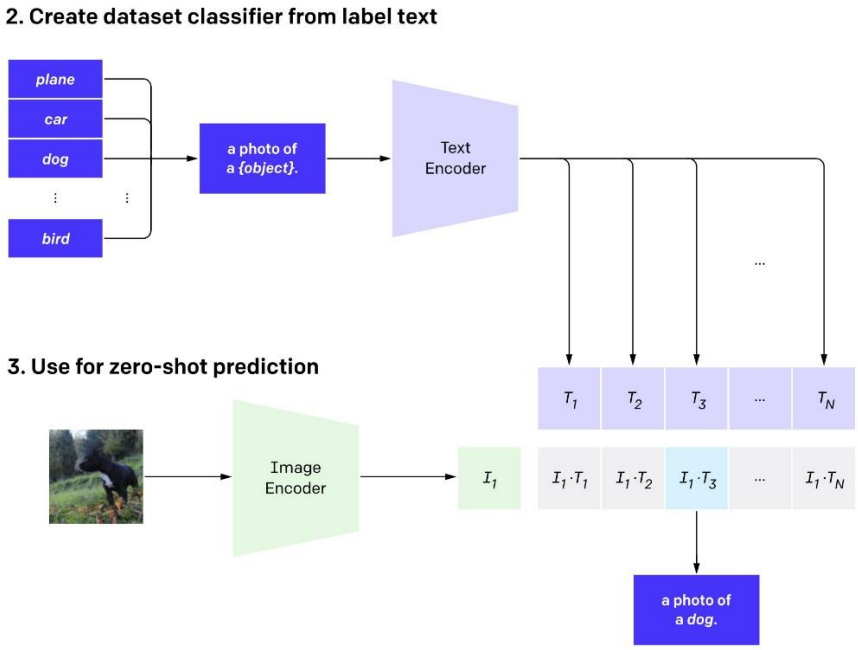

図 3.2：CLIP の Zero-Shot Transfer

3.4.1　Zero-Shot Transfer の実験結果

実験結果は以下の図 3.3 に示されている。比較された合計 27 のデータセットのうち、Zero-Shot CLIP は 16 のデータセットで完全監視の ResNet-50 モデルを上回りました。細粒度分類タスクでは、性能に広範な違いが観察されました。2 つのデータセット（Stanford Cars と Food101）では、Zero-Shot CLIP は ResNet-50 の特徴を用いたロジスティック回帰よりも 20%以上優れていましたが、他の 2 つのデータセット（Flowers102 と FGVCAircraft）では、Zero-Shot CLIP はロジスティック回帰よりも 10%以上劣っている。STL10 では、CLIP は訓練サンプルを一切使用せずに 99.3%の精度を達成した。

同時に、Zero-Shot CLIP は、専門的で複雑、または抽象的なタスクではかなり弱いことも分かります。これには、衛星画像分類（EuroSAT や RESISC45）や、自動運転に関連するタ

スク（ドイツの交通標識認識（GTSRB）や、最も近い車の距離を識別する（KITTI distance）などが含まれます。これらの結果は、より複雑なタスクにおける Zero-Shot CLIP の能力の限界を浮き彫りにしている。

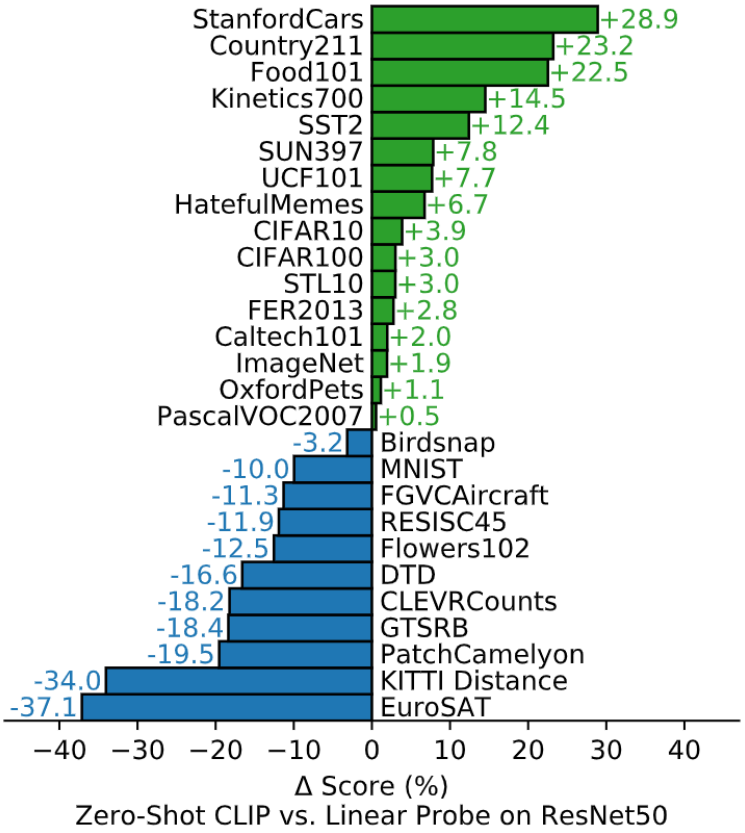

図 3.3：zero-shot CLIP は完全に監督されたベースラインに比べて競争力がある

第 4 章

ファインチューニング(Fine-tuning)

4.1 概要

第 3 章の第 3.4.2 節を通して、Zero-Shot CLIP は、いくつかの専門的で複雑なタスク、例えば衛星画像分類（EuroSAT や RESISC45）などでかなり弱いことが分かる。この章では、CLIP モデルのファインチューニングを通じて、この問題を解決する方法について主に説明する。

ファインチューニング [9]（Fine-tuning：微調整）とは、機械学習におけるファインチューニング（Fine-tuning：微調整）とは、あるデータセットを使って事前学習（Pre-training）した訓練済みモデルの一部もしくは全体を、別のデータセットを使って再トレーニングすることで、新しいタスク向けに機械学習モデルのパラメーターを微調整することである。一般的に、再トレーニングの際の学習率はより小さな値にするため、既に調整済みのパラメーターへの影響もより小さなものとなる。

4.2 人工衛星画像 dataset

モデルのファインチューニングを行うには、まずデータセットを準備する必要がある。私が使用したのは RSICD データセットた。

衛星画像キャプション付けタスク（RSICD）は、Google Earth、Baidu Map、MapABC、Tianditu から収集された 1 万枚以上の遥感画像を含んでいる。画像はさまざまな解像度で

224X224 ピクセルに固定され、遥感画像の総数は 10,921 枚で、画像ごとに 5 つの文での説明がある。このデータセットは衛星キャプション付けのための最大のデータセットた。データセット内のサンプル画像は、高いクラス内多様性と低いクラス間非類似性を持っている。

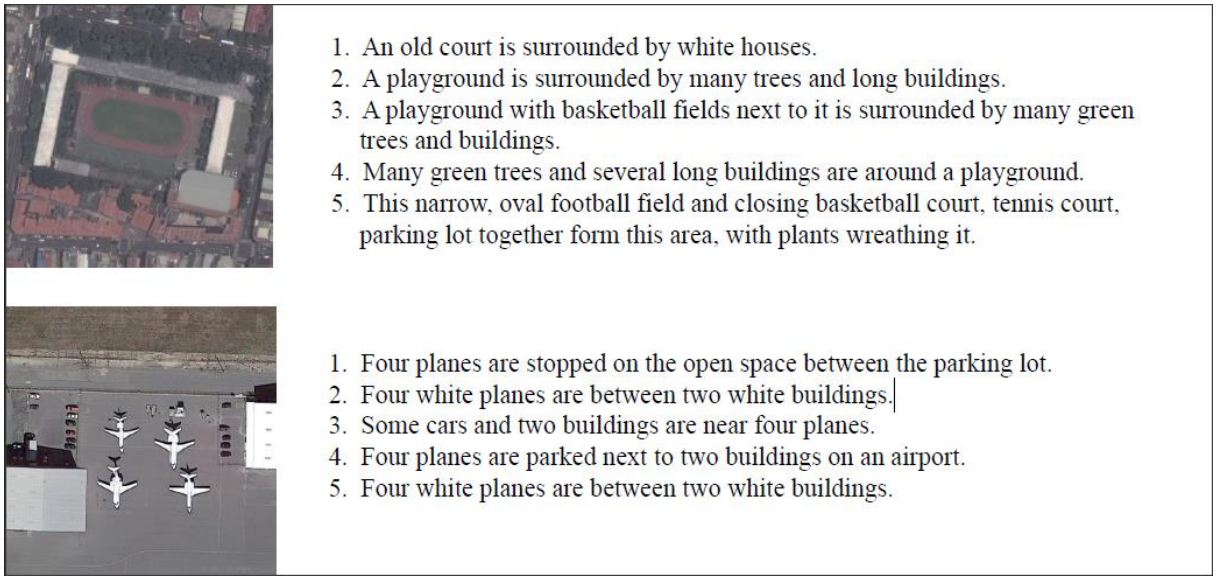

図 4.1：画像データセット

オーバーフィット（Overfitting）を防ぐために、画像の強化が必要です。画像の強化は、Pytorch の Torchvision パッケージ内の組み込み変換を使用してインラインで行われる。使用される変換には、ランダムなクロップ、ランダムなサイズ変更とクロップ、色のジッター、およびランダムな水平および垂直フリップが含まれる。

画像キャプションのテキスト強化は、Marian MT シリーズの翻訳モデルを使用して行われる。これはオフラインで逆翻訳によって完成される。各強化は、異なる言語モデルのペアによる逆翻訳を通じて行われる。

4.3 ファインチューニング

ファインチューニングの過程で、私は CLIP-rsicd [10]の方法を使用しました。公開されているスクリプトを通じて、Colab 上での訓練を再現します。このモデルは、バッチサイズ 1024、線形予熱と減衰を持つ adafactor オプティマイザー、およびピーク学習率 1e-4 で、1 つの TPU-v3-8 上で訓練されています。

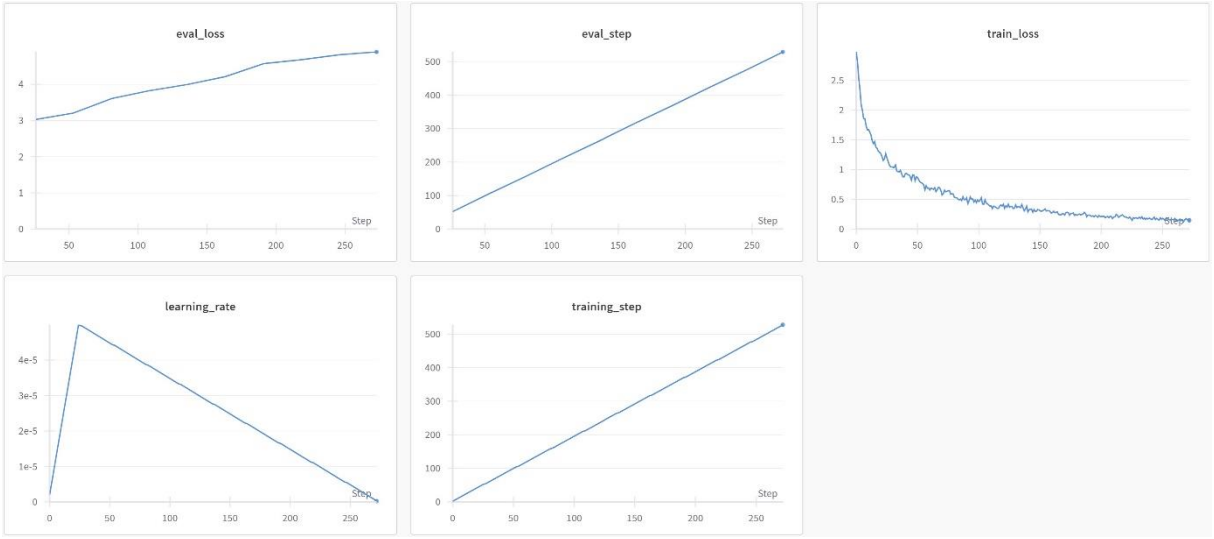

図 4.2：Fine-tuning の流れ

第 5 章

実験

5.1 概要

前の章は CLIP をファインチューニング（fine-tuning）して、その重みを微調整し、モデルを人工衛星画像を専門に処理するツールにする。本章では、任意のスケールの衛星画像の上で CLIP モデルをどのように物体検出に使用するかについて主に説明しています。225×225 の小さなサイズの画像であれ、10000×10000 のような大きなサイズの画像であれ、この方法はそれらに適用可能た。

5.2 検証画像の選び

Google Earth のウェブサイトから大きサイズ衛星画像をダウンロードしてモデルの検証に使用した。10 枚の異なるサイズの地図を検証用の画像として選びした。これらの 10 枚の地図画像は、山や川、森林、空港など、自然景観や人工施設を含んでいる。

図 5.2：テスト画像 1

図 5.3：テスト画像 2

5.2 実験設計

5.2.1 画像処理

図 5.4 に示すように、茨城学校辺りの衛星画像を例に使いました。この衛星画像には野球場、運動場、様々な建物、そしてビーチなどが含まれている。全体の画像サイズは 100MB 以上あるため、画像処理のスピートを速めるために、画像の解像度を下（さ）げた。最終的な画像解像度は 1300×2826 た。

図 5.4：茨城学校辺りの衛星画像

画像オブジェクトを PyTorch テンソルに変換す。torch.size([3, 1300, 2826])

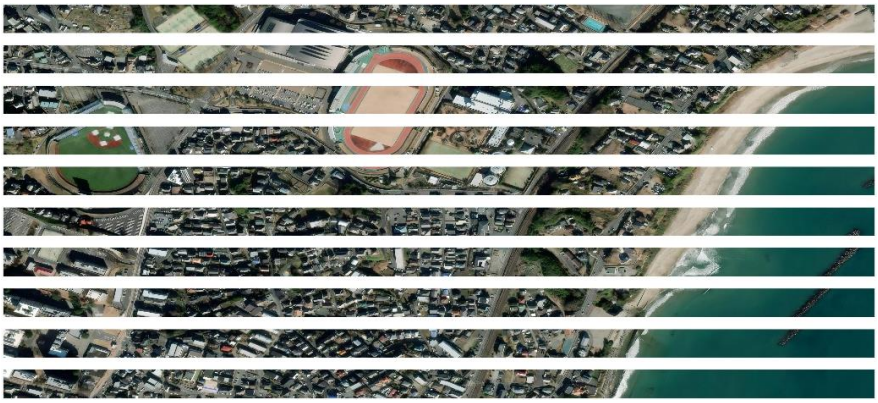

図 5.5：変換の画像 1

図 5.5、画像は 128 画素の高さの帯に分割される。torch.size([1, 10, 2826, 3, 128])

図 5.6：変換の画像２

図 5.6、128 × 128 ピクセルに分割されている。torch.size([1, 10, 22, 3, 128, 128])

次に、ステップサイズが 1 の 4×4 のウィンドウ [8]を使用して、画像の左上から順番にスキャンして、画像をスキャンすると同時に、画像をブロックにして、切り取って保存をする。

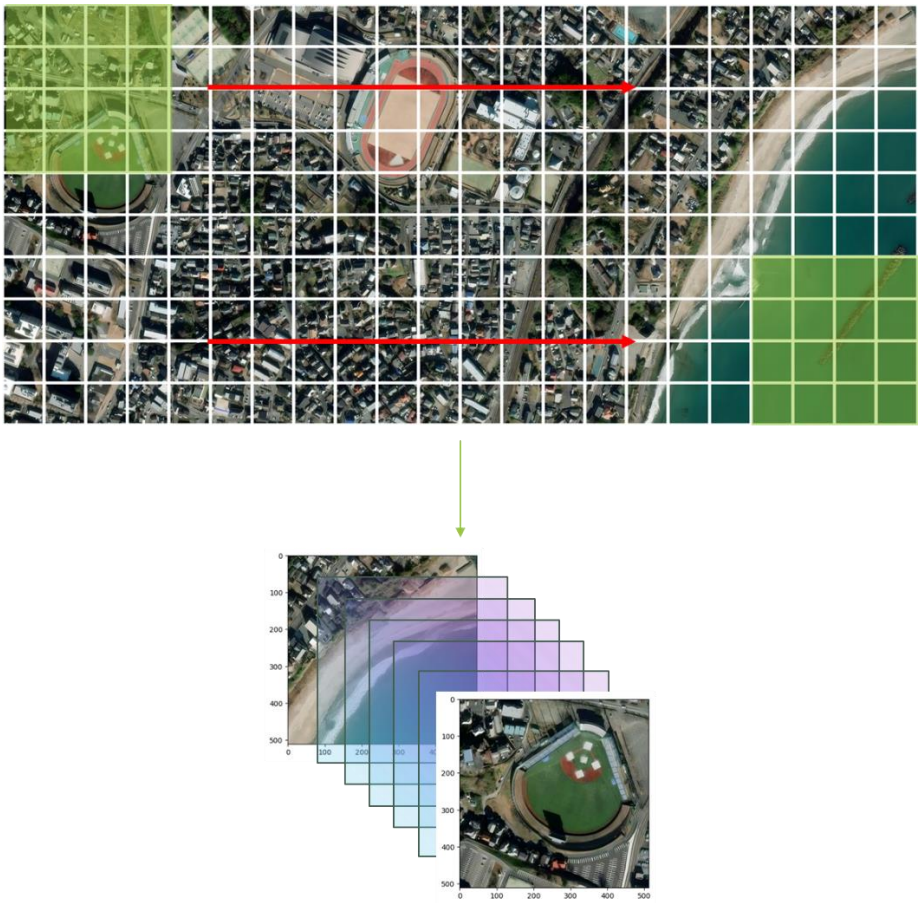

図 5.7：切り取った画像

5.2.2 テキストを用意する

画像処理をした後、テキストの準備をする。テキストは、衛星画像検索で最も一般的な 21 個のキーワードを使用している。図 5.8 に示されているように。すべてのキーワードをプロンプト「A picture of a {keyword}」として作成する。

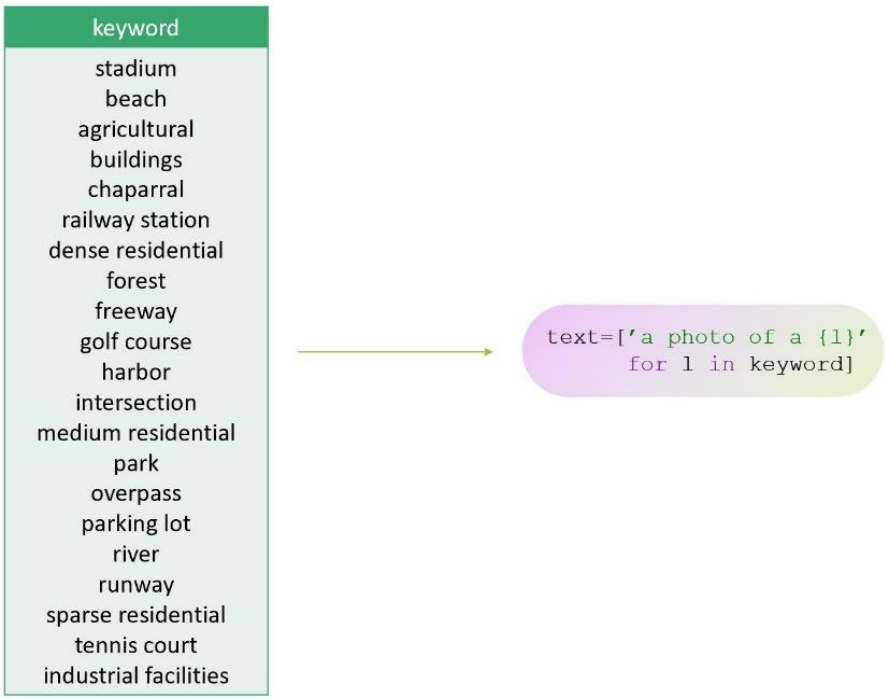

図 5.8：キーワード

5.2.3 類似度計算

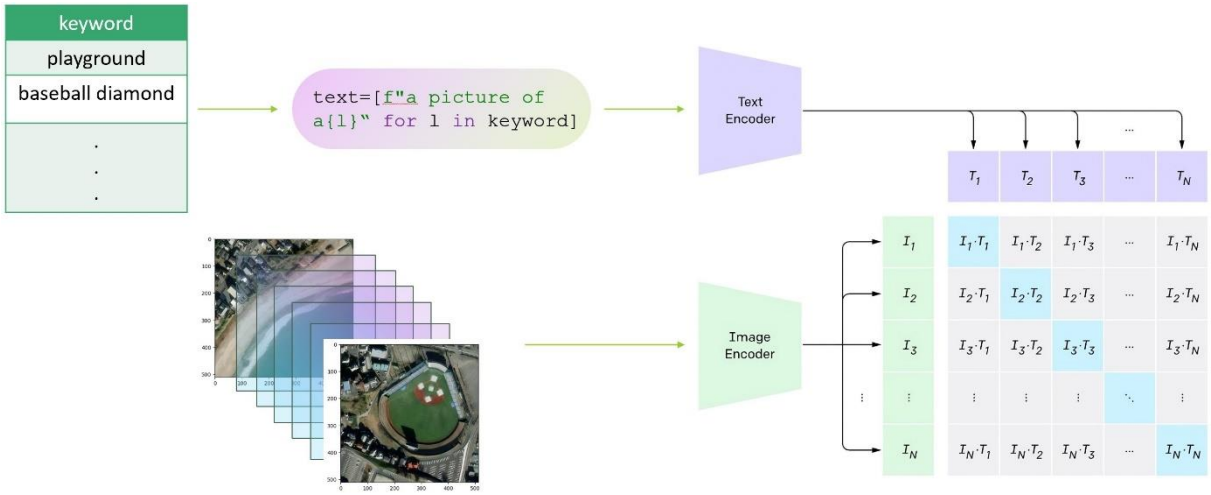

図 5.9：コサイン類似度（Cosine Similarity）を計算する

元の画像サイズが 1300×2826 で、画像ブロックのサイズが 128×128 である場合、4×4 のウィンドウを使用して切り取ると、合計 133 枚の画像が得られる。切り取られた画像は、CLIP モデルに順番に送り込まれて処理される。処理された画像と入力テキストのコサイン類似度を計算する。結果を出力する。結果は、各画像ブロックの類似度の確率値た。コードは付録に添付した。

5.3 実験結果

5.3.1 出力

出力される画像が多すぎるため、ここでは代表性のあるいくつかの画像のみを表示している。出力確率が上位 5 位の画像ブロックと、画像の全体図における位置座標を選択する。

図 5.10：キーワード「beach」に対する出力

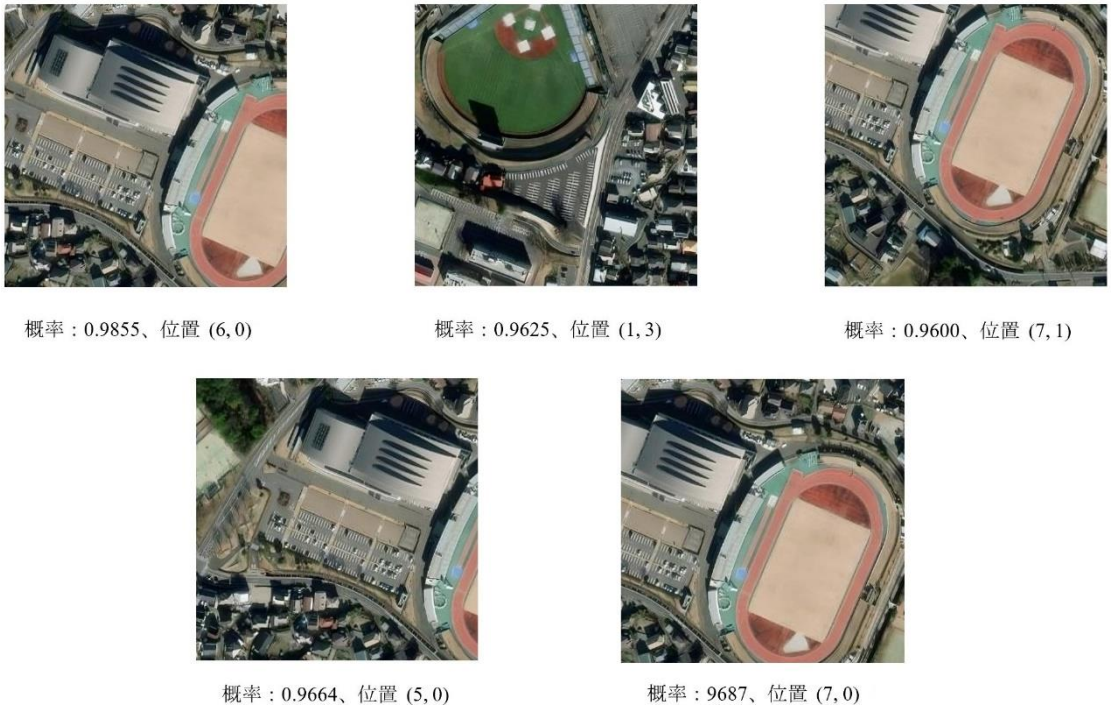

図 5.11：キーワード「stadium」に対する出力

5.3.1 bounding box を作る

出力された確率が上位 5 位（top5）の画像について、画像の座標に基づいて物体検出の bounding box（bounding box）を描画する、その結果は図 5.12 と図 5.13 に示されている。それぞれの 5 枚の画像は、異なる色の bounding box で表示する。

図 5.12：キーワード「beach」に対する bounding box

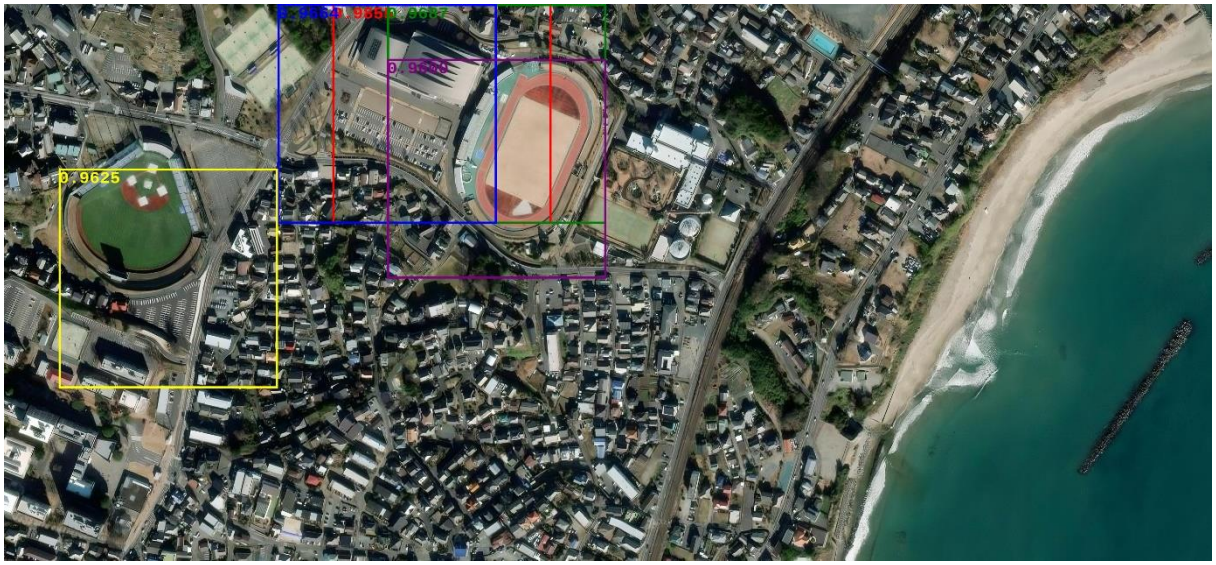

図 5.13：キーワード「stadium」に対する bounding box

以下はテスト画像から、代表的な検査結果。

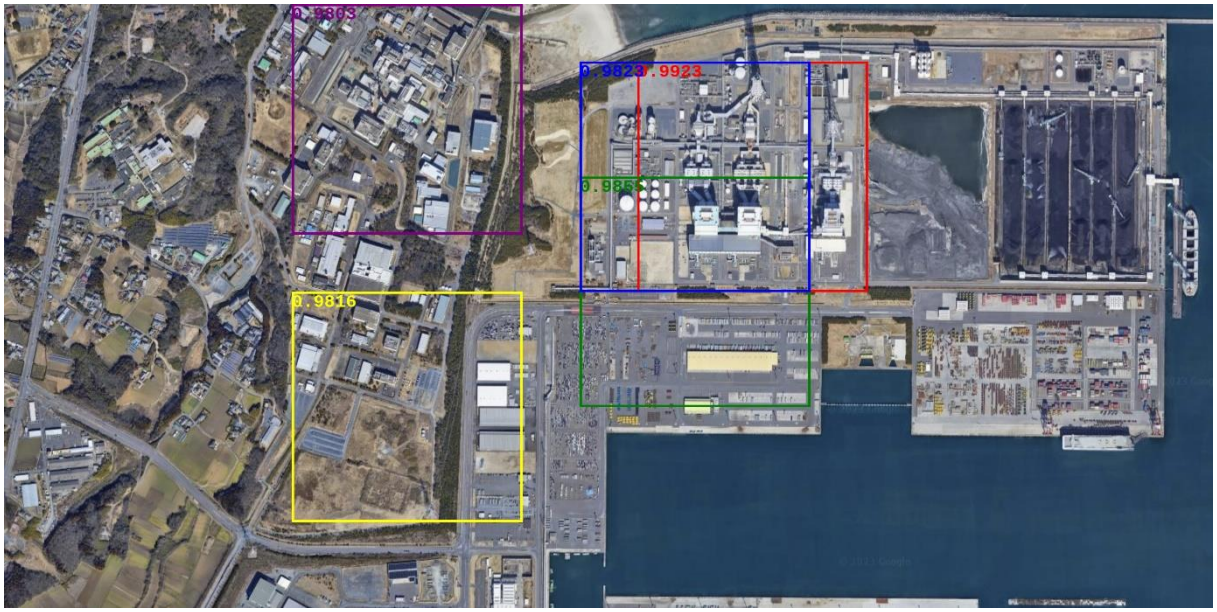

図 5.14：キーワード「industrial facilities」に対する bounding box

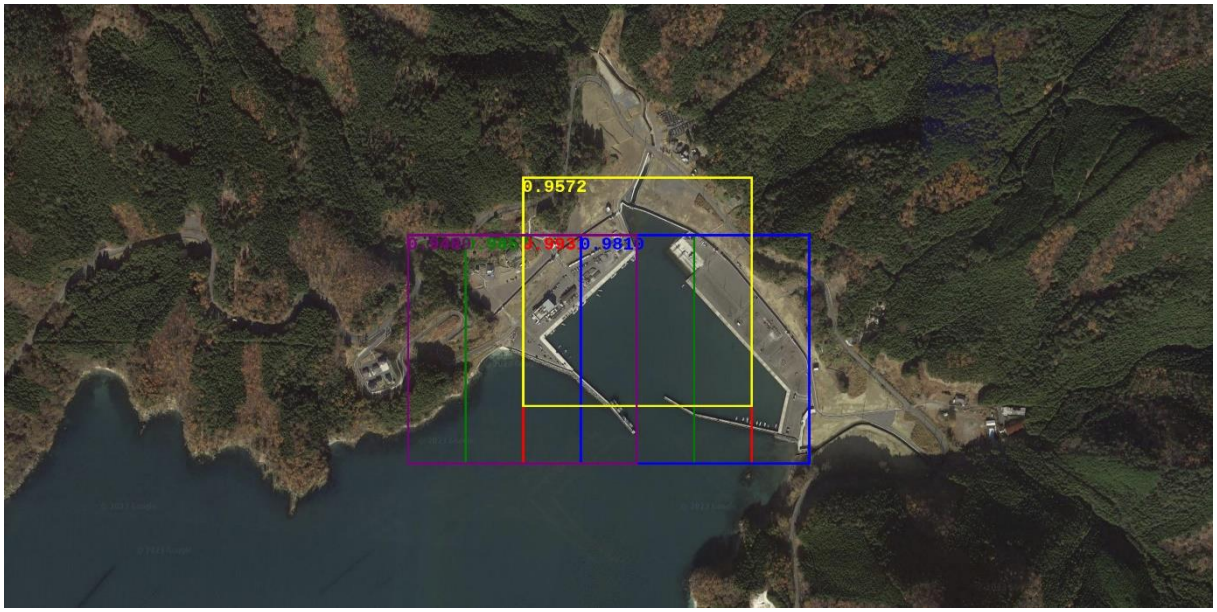

図 5.15：キーワード「harbor」に対する bounding box

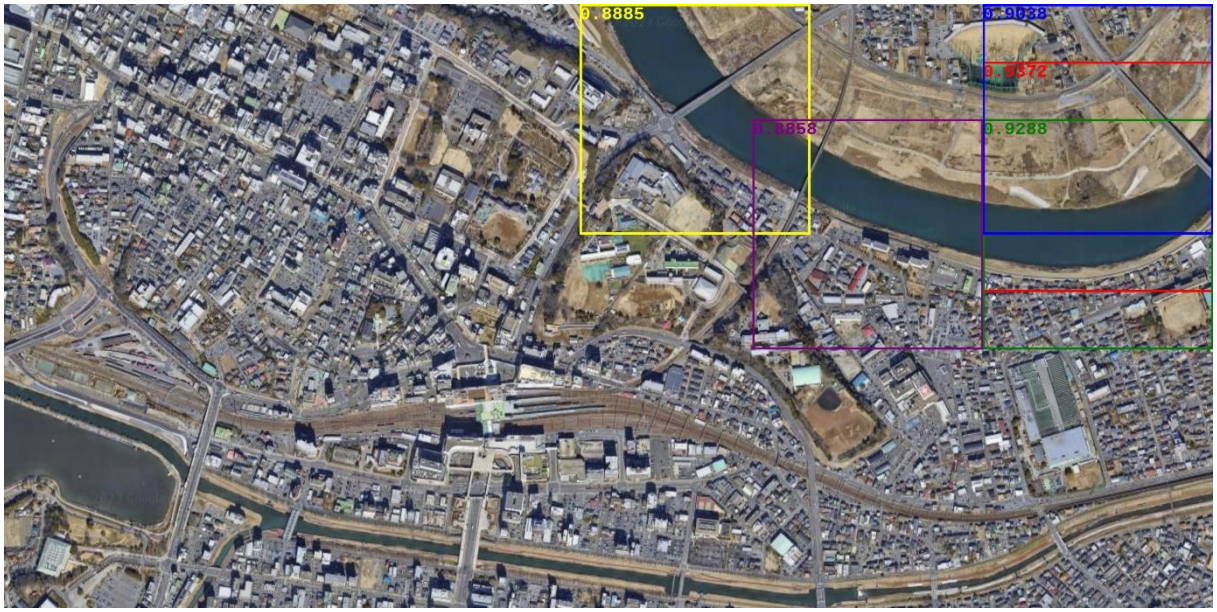

図 5.16：キーワード「river」に対する bounding box

第 6 章

考察

6.1　物体検出結果の改善

研究の方法には欠点もる。図 5.13 と 5.14 に示されているように、画像に複数の目標がある場合、これらの bounding box は散在し、一部の bounding box は中心の検出目標から逸脱することがあります。また、図 5.16 に示されるように、検出目標が比較的小さい場合、bounding box は目標以外の多くのものを含むことになります。次の小節では、これらの欠点を改善する方法についてまとめています。

6.1.1 もっと画像を出力する

元の検出結果（図 5.13 と図 5.14）に表示された画像ブロックは出力確率が上位 5 位でした。もし出力する画像ブロックを増やす、より多くの画像ブロックの bounding box を描画することで、より多くの目標を含めることができる。以下の図に示されている。

図 6.1：確率上位 10 位の物体検出

物体検出の出力範囲を上位 5 位（top5）から上位 10 位（top10）に変更すると、結果の画像にはより多くの目標物が含まれるようになる。

6.1.2 切り取り画像のサイズ

物体検出の結果は図 5.12 と図 5.16 に、ビーチと川を検出する際、目標物が狭いため、4×4 の切り取りウィンドウサイズでの画像ブロックでは、多くの目標外の物体が検出タスクに含

まれている。この状況を改善するには、切り取りウィンドウのサイズと画像ブロックのサイズを変更することで対応できる。切り取りウィンドウを 2×2 のサイズに設定し、画像の最初の処理時に、分割された画像ブロックを 128×128 ピクセルから 64×64 ピクセルに縮小する。これにより、ウィンドウを使用して画像をスキャンすると、より多くの画像ブロックが得られる。CLIP モデルで計算した後、出力される画像ブロックを 50 個まで増やし、図 6.2 と図 6.3 に示されている。

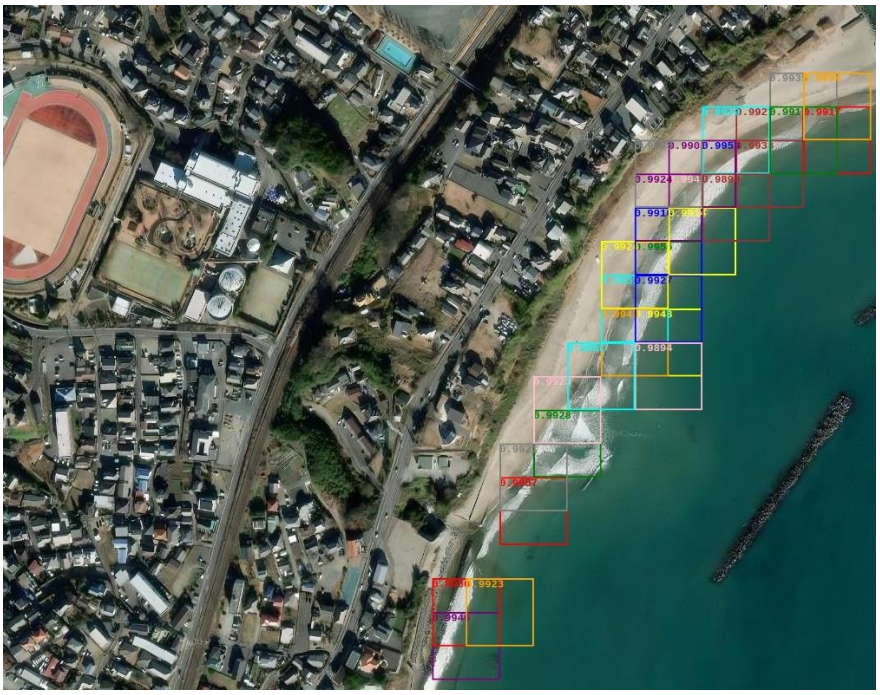

図 6.2：キーワード「beach」の top50

図 6.3：キーワード「river」の top50

衛星画像検出において、一つの画像内に複数の目標物が存在することがあり、また、目標物の大きさも様々。この方法により、描画される bounding box は、検出目標をより多く、より正確に含むことができる。以下の図は、出力された確率上位 100 の画像ブロックに描画された bounding box た。

図 6.4 キーワード「beach」の top100

図 6.5 キーワード「river」の top100

6.2　まとめ

前の小節から分かるように、画像ブロックのサイズを小さくすると、全体の画像がより細かく分割され、裁剪框のサイズを小さくすることで、より多くの画像の細部をスキャンできる。その後、CLIP で認識を行うと、より正確な bounding box を描画する。もし元の画像が非常にクリアで、より多くのピクセルを持つ場合、検出の精度をさらに高めることができる。

しかし、上記の検出結果を改善する方法は、コンピュータのハードウェア性能によって制限された。画像ブロックを減らすことや切り取りウィンドウを小さくすることは、プログラム内のループ計算の回数を大幅に増加させる。図 6.5 で描く 100 個の画像 bounding box

（top100）は、私が実験で使用したコンピュータの性能の限界でした。実際の操作で、画像を 128×128 サイズに分割し、4×4 のウィンドウでスキャンして CLIP に入力するプロセス全体で、プログラムは約 1 分かかりした。切り取りウィンドウを 3×3 に変更するだけで、プログラムの実行時間が 5 分に増加した。したがって、より大きなサイズの画像を処理し、より精密な検出タスクを行う場合は、コンピュータの GPU 性能が非常に高いことが要求される。

第 7 章

結論

この研究で使用された画像処理方法は、任意のサイズの画像における物体検出に適用できる。画像ブロックのサイズと切り取りウィンドウのサイズを設定することで、画像の各部分をスキャンし保存した後、CLIP モデルに入力して認識を行う。さらに、CLIP の多モーダルモデルの強力な画像処理能力を活用し、キーワードを用いて目標を正確に識別し位置を特定する。高い精度を求めない場合は、オリジナルの CLIP モデルを直接使用することもできる。オリジナルの CLIP モデルはすでに非常に優れており、多くの画像を認識するために微調整を行う必要がない。物体検出において、この研究で使用された方法と組み合わせることで、実際にはどんな訓練データセットも不要で、モデルに何の訓練も施さずに物体検出タスクを完了することができる。

画像ブロックのサイズを小さくし、裁剪框のサイズを縮小するなどの方法により、より多くの画像の細部をスキャンし、物体検出の精度を向上させることができる。でも、コンピュータの GPU 性能には高い要求がある。

今後の課題の重点は、画像処理のアルゴリズムを改善して、画像計算にかかる時間を短縮すること。人工知能の発展に伴い、現在、多くの画像処理分野の AI がある。AI モデルを導入して、切り取られた画像ブロックのピクセルを改善することにより、CLIP の検出精度をさらに高めることができる。

謝辞

最後に、本論文を作成するにあたり、指導、助言を頂いた、指導教官の新納浩幸教授に心より感謝申し上げます。また、日常の議論を通して多くの知識、示唆を頂いた新納研究室の皆様に感謝しております。

参考文献

[1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever.Learning Transferable Visual Models From Natural Language Supervision.In International Conference on Machine Learning, pp. 8748–8763. PMLR, 2021.

[2] He, K., Gkioxari, G., Dollár, P., & Girshick, R. B.Mask R-CNN. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5542-5551, 2017.

[3] Redmon, J., Divvala, S., Girshick, R. B., & Farhadi, A.You Only Look Once: Unified, Real-Time Object Detection. Computer Vision and Pattern Recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779-788, 2015.

[4] He, K., Zhang, X., Ren, S., & Sun, J. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778, 2015.

[5] Socher, R., Ganjoo, M., Manning, C. D., & Ng, A. Zero-Shot Learning Through Cross-Modal Transfer. Neural Information Processing Systems, 935-943, 2013.

[6] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. Learning Transferable Visual Models From Natural Language Supervision. International Conference on Machine

Learning, 8748-8763, 2021.

[7] Sarma, S. Zero-Shot Learning for Computer Vision Applications, 2023.

[8] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 2021.

[9] Howard, J., & Ruder, S. Universal Language Model Fine-tuning for Text Classification. Annual Meeting of the Association for Computational Linguistics, 328-339, 2018.

[10] Arampacha, Dev Vidhani, Goutham, Mayank Bhaskar, Sujit Pal. Fine tuning CLIP with Remote Sensing (Satellite) images and captions. Hugging Face Blog, 2021, October 13.

付録

実験は Google Colab の上で行い、画像処理と計算に使用したソースコードは A.1 に示す。bounding box の作るソースコードは A.2 に示す。

A.1 画像処理と類似度計算

```
from PIL import Image
from IPython.display import display
import requests

from transformers import CLIPProcessor, CLIPModel

model = CLIPModel.from_pretrained("flax-community/clip-rsicd")
processor = CLIPProcessor.from_pretrained("flax-community/clip-rsicd")

img_path = '/content/hitachi1-1.jpg'
tile_size = 64
window_size = 4
# 画像を開く
img = Image.open(img_path)
width, height = img.size

# 水平と垂直方向の移動回数を計算
x_steps = (width - window_size * tile_size) // tile_size + 1
y_steps = (height - window_size * tile_size) // tile_size + 1

save_probs = []

# 画像をループで走査
for y_step in range(y_steps):
  for x_step in range(x_steps):
    # ウィンドウの左上の座標を計算
    left = x_step * tile_size
    upper = y_step * tile_size
    right = left + window_size * tile_size
    lower = upper + window_size * tile_size
```

```
31.
32.     # 画像をクロップ
33.     cropped_img = img.crop((left, upper, right, lower))
34.
35.     # キーワードを入力
36.     labels = ["beach", "stadium", "park", "forrest", "river",
37.           "airport", "industrial facilities", "freeway", "harbor", "railway station"]

38.     inputs = processor(text=[f"a photo of a {l}" for l in labels],
39.                        images=cropped_img, return_tensors="pt", padding=True)
40.     outputs = model(**inputs)
41.     logits_per_image = outputs.logits_per_image
42.     # 確率を計算
43.     probs = logits_per_image.softmax(dim=1)
44.
45.     index = labels.index("industrial facilities")
46.
47. # 確率に基づいて並べ替えて取得
48. top_5 = sorted(save_probs, key=lambda x: x[0], reverse=True)[:5]
49. top_10 = sorted(save_probs, key=lambda x: x[0], reverse=True)[:10]
50. top_30 = sorted(save_probs, key=lambda x: x[0], reverse=True)[:30]
51. top_50 = sorted(save_probs, key=lambda x: x[0], reverse=True)[:50]
52. top_100 = sorted(save_probs, key=lambda x: x[0], reverse=True)[:100]
53.
54. # 結果を出力
55. for prob, image, position in top_5:
56.     print(f"確率: {prob:.4f}, 位置: {position}")
57.     display(image)
58.
59.     save_probs.append((probs[0][index].item(), cropped_img, (x_step, y_step)))
```

A.2 bounding box を作る

```
1.  from PIL import Image, ImageDraw, ImageFont
2.
3.  # 元の画像を開く
4.  original_img = Image.open(img_path)
5.  draw = ImageDraw.Draw(original_img)
6.
7.  # 5 種類の色を定義
8.  colors = ["red", "green", "blue", "yellow", "purple"]
```

```
9.
10. # フォントをロード
11. try:
12.     font = ImageFont.truetype("/usr/share/fonts/truetype/liberation/LiberationMono-
    Bold.ttf", 40)
13. except IOError:
14.     font = ImageFont.load_default()
15.
16. # bounding box と確率テキストを描画
17. for (prob, _, position), color in zip(top_5, colors):
18.     x_step, y_step = position
19.     left = x_step * tile_size
20.     upper = y_step * tile_size
21.     right = left + window_size * tile_size
22.     lower = upper + window_size * tile_size
23.
24.     # 矩形ボックスを描画
25.     draw.rectangle([left, upper, right, lower], outline=color, width=5)
26.
27.     # 確率テキストを描画
28.     text = f"{prob:.4f}"
29.     draw.text((left, upper), text, fill=color, font=font)
30.
31. # bounding box と確率が付いた画像を表示
32. display(original_img)
```