令和 4 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

DETR を用いた物体検出の One-Click Supervision

所属　　情報工学専攻

著者　　平野友基 (21NM750L)

指導教員　　新納浩幸教授

令和 5 年 2 月 3 日 (水)

令和 4 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

DETR を用いた物体検出の One-Click Supervision

著者

平野友基 (21NM750L)

指導教員

新納浩幸教授

論文要旨

本稿では，[7] で click supervison というアノテーション手法内で MIL が使用されているのに対し，この領域の提案を DETR に置き換えることで，学習の簡易化を試みた．また，その時の精度を見た．

オブジェクトを検出するには，通常バウンディングボックスにより注釈されたオブジェクトを持つ大規模なセットが必要である．しかし，手作業でバウンディングボックスを描くのは非常に大変な作業である．そこで中心をクリックすることでアノテーションの時間を大幅に短縮する click supervision という手法が提案された．

先行研究では，物体の中心点と MIL をもとに教師データの作成をこなっている．学習の結果，Bounding Box を手作業で作成したときと近い精度を達成し，アノテータの作業を 9 から 18 倍短縮した．

本研究は物体の中心点に学習済みの DETR の予測結果を適用することで，教師データの作成を試みた．

DETR は自然言語処理などで用いられていた注意機構をネットワーク内に初めて取り入れた物体検出モデルである．DETR における transformer の注意機構は CNN から入力された画像の特徴量から物体を予測するが，このとき，他の予測内容を考慮するという特性から，似たような候補が多く生成されるといった問題が起こらずに，同じような物体領域候補の中から重複を取り除く non-maximal suppression という処理が必要無くなった．この DETR の領域の提案をを用いることで，教師データが作成できると予想した．

実験内容は学習済みのクラスでは教師データがうまく作成されることは自明であったため，教師データのクラスには学習済みの DETR のクラスが含まれないものにした．実験の結果，Boundingbox による教師データに近いものが得られるといったことが期待されたが，想定した結果を得ることはできず，Bounding Box を利用したときのの精度よりも劣る結果が得られた．

Master's Thesis in Scholastic 2022, Major in Computer and Information Sciences,
Granduate School of Science and Engineering, Ibaraki University

One-Click Supervision for Object Detection using DETR

Auther

Tomoki Hirano (21NM750L)

Adviser

Prof. Hiroyuki Shinnou

Abstract

In this paper, we attempted to simplify learning by replacing the region proposal with DETR, whereas MIL is used within the annotation method called click supervison in [7].We also looked at the accuracy at that time.
Detecting objects usually requires a large set with objects annotated by bounding boxes.However, manually drawing bounding boxes is a very difficult task. A technique called "click supervision" was proposed to reduce annotation time by clicking the center of the object. In the previous study, the annotator created the teacher data based on the object's center point and MIL. The training results achieved accuracy close to that of manual creation of the Bounding Box and reduced the annotator's workload by a factor of 9 to 18. This study attempted to generate supervised data by applying the learned DETR predictions to the object's center point.DETR is the first object detection model that incorporates the attention mechanism used in natural language processing into the network.The transformer's attention mechanism in DETR predicts objects from image features input from a CNN. This eliminates the need for non-maximal suppression, which removes duplicates from similar object region candidates, without causing problems such as the generation of many similar candidates due to the nature of considering other predictions.We expected that the proposed DETR regions could be used to generate teacher data.The class of the teacher data was set to not include the class of the learned DETR.The expected results, which were close to those obtained with the Bounding Box, were not achieved, and were less accurate than those obtained when using the Bounding Box.

目次

第 1 章

序論

物体検出は入力された画像から物体の位置とカテゴリーを検出するタスクである．物体検出はオブジェクトクラス検出器によって行われ，これを訓練することで，物体検出の精度を高める．物体検出モデルには様々な物が提案されている [4] [5]．中でも，DETR [3] というモデルは優れた性能を示している．しかし，物体検出において，オブジェクトクラス検出器を訓練するためには，画像にバウンディングボックスと呼ばれる対象を囲む矩形の情報を与える必要があり，これを一からすべて手作業で行うには時間がかかる．そこで物体の中心をクリックするのみで行うことのできる click supervision [7] という手法が提案された．

click supervision とは，物体の中心点をクリックすることで，画像にアノテーションデータを与えるもので，通常のアノテーションデータの作成の場合と同等の精度で 9 から 18 倍速く作成することができる．[7] 内での学習モデルには Fast R-CNN [5] が使用されている．

DETR(End-to-End Object Detection with Transformers) は 2020 年 5 月に Facebook の研究チームによって公開された物体検出モデルであり，初めて Transformer を採用した物体検出モデルである．

本研究では，この click supervision において，物体検出器を作成する過程で，物体候補の提案と，学習部分の学習モデルを統一して DETR を用いて学習を行った．

第 2 章

関連研究

2.1　物体検出

物体検出は画像の中の物体の位置を推定して，物体のクラスを分類するタスクである．物体を取り囲むボックスをバウンディングボックスという．（図 2.1）の場合，複数の風船が，バウンディングボックスで囲まれて，ボックスの左上にクラス分類と確信度が表示されている．

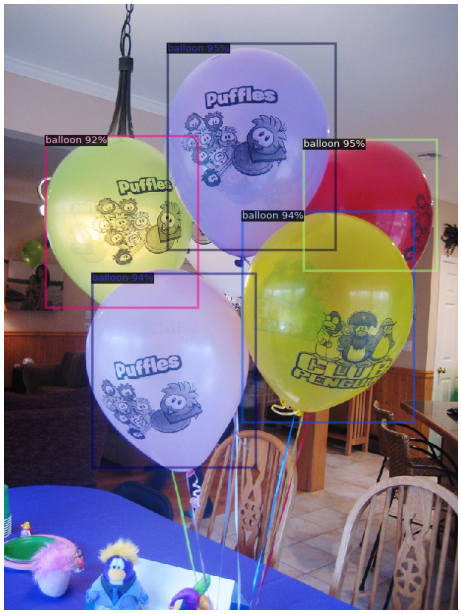


図 2.1: 物体検出のイメージ

物体検出には R-CNN, Faster R-CNN, YOLO, SSD などの複数モデルがあり，それぞれ特徴がある．物体検出の各種モデルを表 2.1 に示す．

表 2.1: 物体検出の各種モデル

モデル名	説明
R-CNN	入力画像から物体の領域候補を探し，CNN で特徴を取り出し，その特徴を使い SVM でクラス分類をし，回帰によりバウンディングボックスを作成.
Fast R-CNN	R-CNN では分類と回帰の損失関数が分かれていたが，Fast R-CNN では損失関数が統合され，一度に学習が可能になった.
Faster R-CNN	物体の領域候補の提案が CNN になり，速度が大きく向上した.
YOLO	画像を 7 × 7 のグリッドに分割し，グリッドごとに位置推定とクラス分類を実行する.
SSD	画像に 8732 個のデフォルトボックスを敷き詰めて，デフォルトボックスごとに位置推定とクラス分類を実行.

2.2　R-CNN

R-CNN はディープラーニングを用いた物体検出の初期のモデルである．学習の流れを (図 2.2) に示す．検出までの手順は，入力された画像の中から，物体が写っている領域の候補を最大 2000 個まで抽出，領域提案の特徴量を CNN を用いて計算，領域提案ごとの分類を行う．と言った流れになっている．

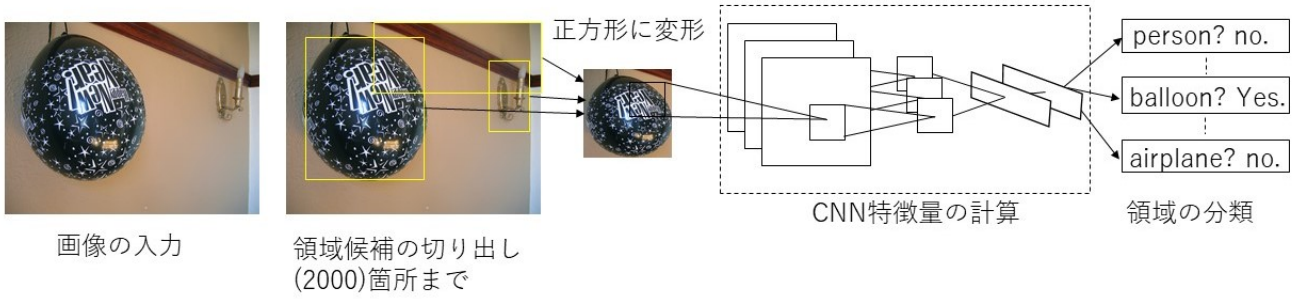


図 2.2: R-CNN のアーキテクチャ

これにより R-CNN 以前のモデルよりも高精度な物体検出が可能になった．しかし，それぞれの項目ごとに学習をする必要があり，学習に非常に時間がかかっていた．また，

領域提案自体は精度が低く領域候補の数だけ CNN を計算するため，演算量も大きいという欠点があった．

2.3　Fast R-CNN

物体検出アルゴリズムでの一つで，R-CNN などの以前の物体検出アルゴリズムでは学習が多段になっていて複雑であり，検出に時間がかかっていた．これの原因として，領域候補の個数分だけ CNN を計算するため，冗長な計算がたくさん発生していた．これに対して Fast R-CNN では，Edge Boxes などのアルゴリズムを用いて領域の提案を行ったあと，各領域から抽出した特徴領域をプーリングしている．また，学習時には最下層まで誤差逆伝播する．これによって，毎回 CNN を走らせる必要はなく，領域提案の抽出した特徴領域を切り出し，全結合層に与えるだけでよくなったため，従来の R-CNN と比べ，大幅な高速化を達成した．学習の一連の流れを (図 2.3) に示す．

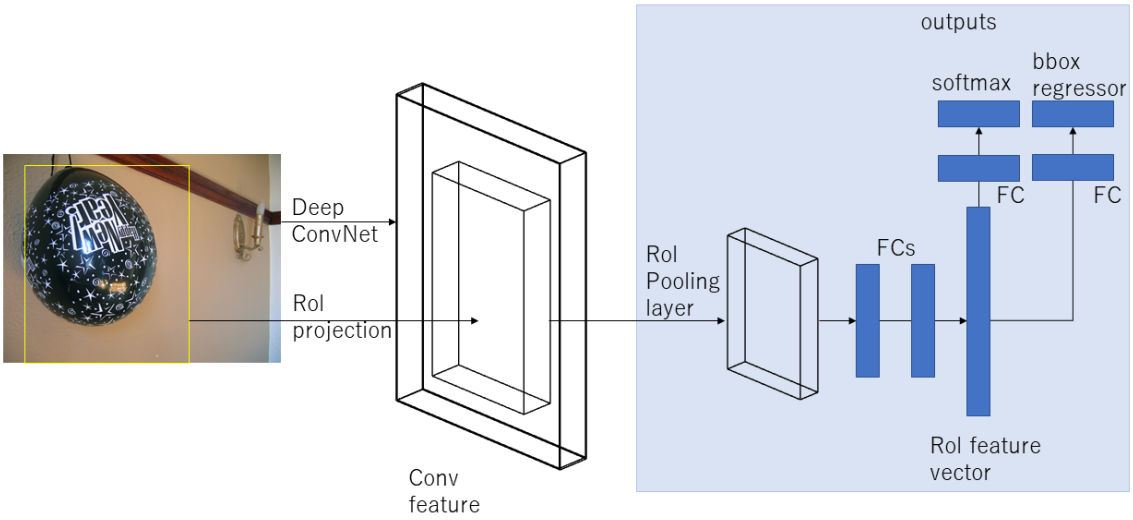


図 2.3: Fast R-CNN のアーキテクチャ

検出までの手順を説明する．

まず，入力画像を Image Net 学習済みモデルの conv 層までを使って任意サイズの featuremap を計算する．次に，Selective Search などで求めた領域候補 (RoI) を feature map 上に射影し，Feature map 上で射影された RoI を RoI Pooling．その後，何段か FC

層を挟んだ後，物体カテゴリの分類問題と矩形回帰問題を同時に解く．学習時には，最下層まで誤差逆伝播をする．

このような流れで学習を行う．画像認識毎に CNN 層も走らせなくてもよくなったことによる恩恵として，RegionProposal が n 回あったとすると，演算量は，

- R-CNN : CNN*n+FC*n
- Fast R-CNN : CNN*1+FC*n

となっている．n の最大値が 2000 であるため，CNN の演算回数を最大 1/2000 に出来るようになった．

また Fast R-CNN は Multi-task loss という学習技術を提案しており．BoundingBox とクラス分類のネットワークを同時に学習をすることに成功している．結果として Fast R-CNN は R-CNN に対し，推論速度と学習速度を大きく向上させた．

2.3.1　Rol Pooling

Rol Pooling(図 2.4) はアスペクト比の違いを考慮して Max Pooling を行う処理で，これにより，領域を固定サイズ化して出力する．

2.3.2　Multi-task Loss

Fast R-CNN で使用した損失関数 Multi-task Loss(図 2.5) について説明する．これは，物体の分類と Bounding Box の回帰という 2 つのタスク (Multi-task) を学習するために，この 2 のタスクの推定誤差を同時に考慮した損失関数 (loss function) である．

2.4　Faster R-CNN

Fast R-CNN のあとに Microsoft が発表した物体検出アルゴリズムで，Deep Learning による End-to-End の学習を可能にした最初のモデルである．SPPnet や Fast R-CNN などでボトルネックとなっていた領域提案と多段階の学習を検出ネットワークと全画像の畳み込みの特徴を共有することで領域提案を可能にした RPN を導入することで実行時間を短縮した．RPN は，各位置におけるオブジェクトの境界とオブジェクトスコアを同時に予測する完全畳み込みネットワークである．RPN は高品質な領域提案を生成する

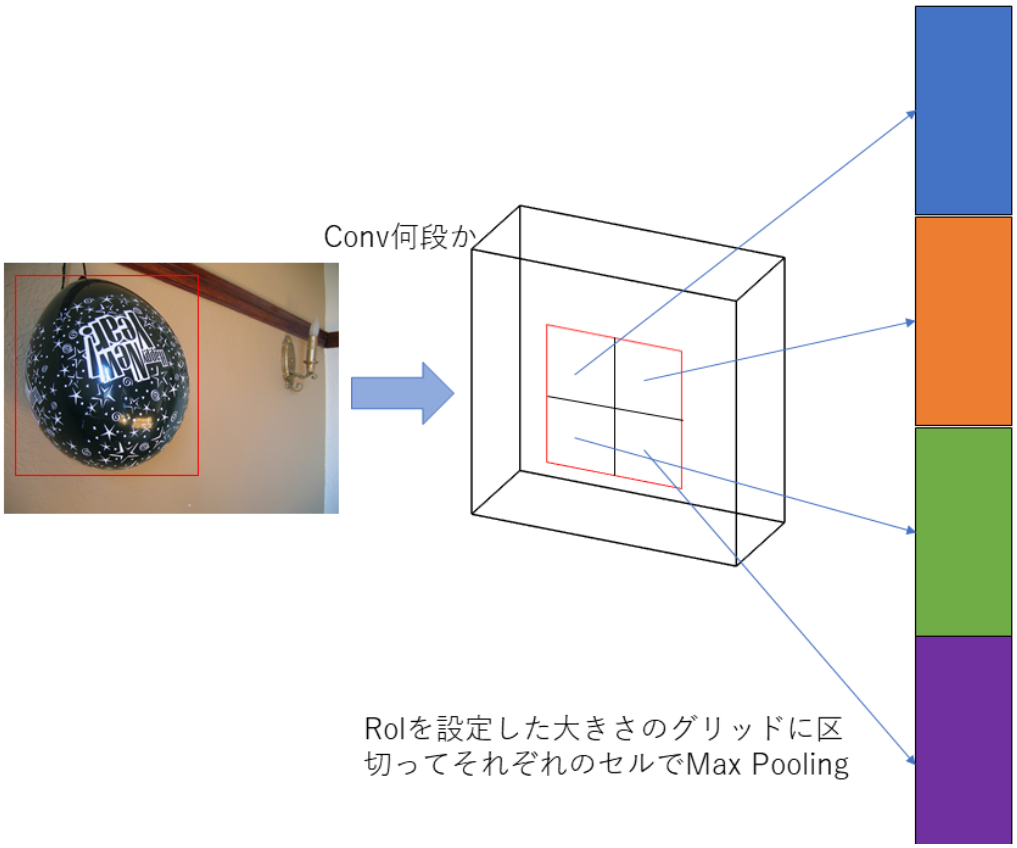


図 2.4: RoI Pooling の構造

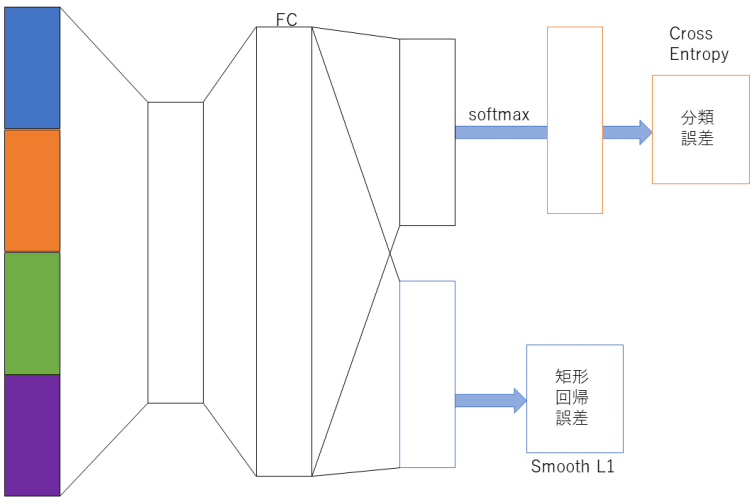


図 2.5: Multi-task Loss の構造

ために End-to-End で学習され，Faster-RCNN によって検出のために使用される．このとき，RPN と Fast R-CNN の畳み込み機能を共有することでネットワークを一つに統合している．

Fast R-CNN までの領域提案には従来技術である Selective Search を使用しておりこの部分の処理が全体の内の多くの時間を占めていた．具体的には，Fast R-CNN では一枚の画像の処理に 2.3 秒かかるが，そのうち 86 %が RegionProposal に費やされ，ボトルネックとなっていた．

そこで, Faster R-CNN は RegionProposal も Region Proposal Network というニューラルネットワークに置き換えて物体検出モデルを全て DNN にし，End-to-End で学習することによって高速化を実現した．

全体図は（図 2.6）のようになっている．

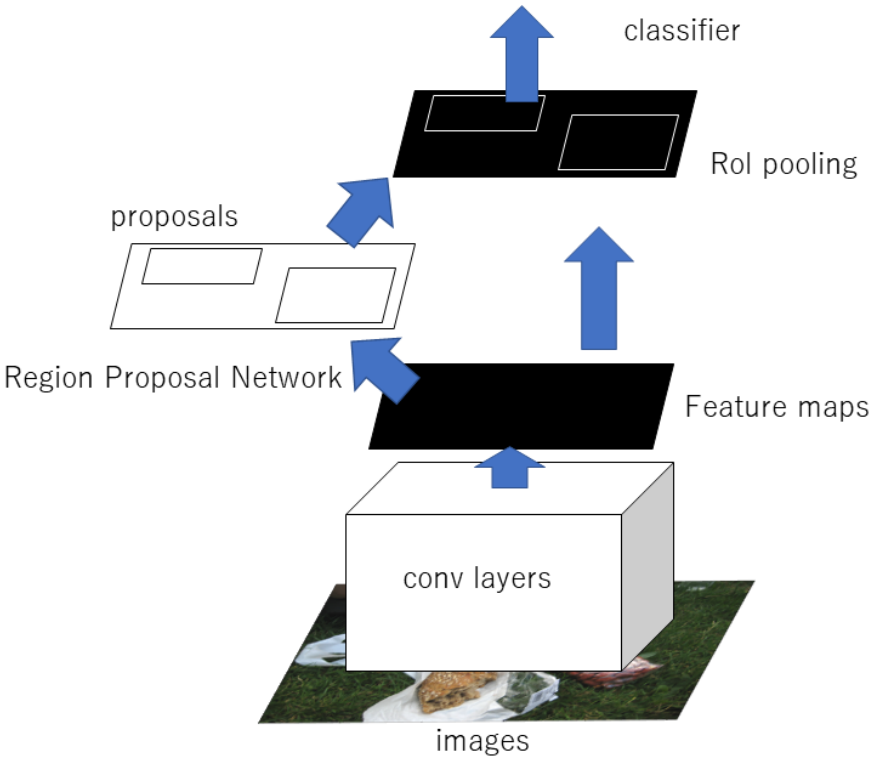


図 2.6: Faster R-CNN の構造

2.4.1　RPN（Region Proposal Network）

RPN は物体の場所と矩形を検出する機械学習モデルである．分類は Faster R-CNN の別の箇所で行われる．RPN の学習は入力画像から特徴マップの入力，Anchor boxes を設定する．Anchor boxes と Ground truth を比較し，RPN の教師データを作成するといった手順で行われる．これによって，入力画像と教師データでの教師あり学習を行う．

物体領域候補の検出を行う RPN (Region Proposal Network) は，画像全体から CNN で抽出した特徴マップに対して，候補領域の Bounding Box と，その領域の物体らしさ (Objectness) を表すスコアを出力する．

RPN は小さなニューラルネットワークで，特徴マップ上を n × n サイズのスライディング window で走査し，各々の n × n サイズの注目領域をネットワークの入力とする．そして，各スライディング window 位置に対して k 個の候補領域を推定する．また，この推定のために RPN は，物体かどうか（objectness）を分類する cls layer と，Bounding Box の回帰を行う reg layer に分岐する．

cls layer には，k 個の各候補領域がオブジェクトか，背景かの確率を推定した 2k 個のスコアが出力される．reg layer には，k 個の Bounding Box の座標・サイズ (x 座標，y 座標，幅，高さ) を表す 4k 個の出力がある．

Anchor

スライディング window に対し，Anchor と呼ばれる k 個の検出矩形パターンを用意する．Anchor はスライディング window の中心を基準に設定される．論文では，アスペクト比の違う 3 種類の矩形をさらにスケール違いで 3 種類用意し，k=9 としている．特徴マップのサイズを W × H(2,400 以下) とすると，RPN は合計 W × H × k 個の物体候補の領域が生成される．これらを RPN で判別して，物体である可能性が高いものは，Fast R-CNN と同様に RoI プーリング以降のネットワークへと進み，物体の分類が行われる．

また，Faster R-CNN では畳み込み層を共有するだけでは学習するパラメータが相互に依存してしまうため，段階的に学習する方法が採られている．

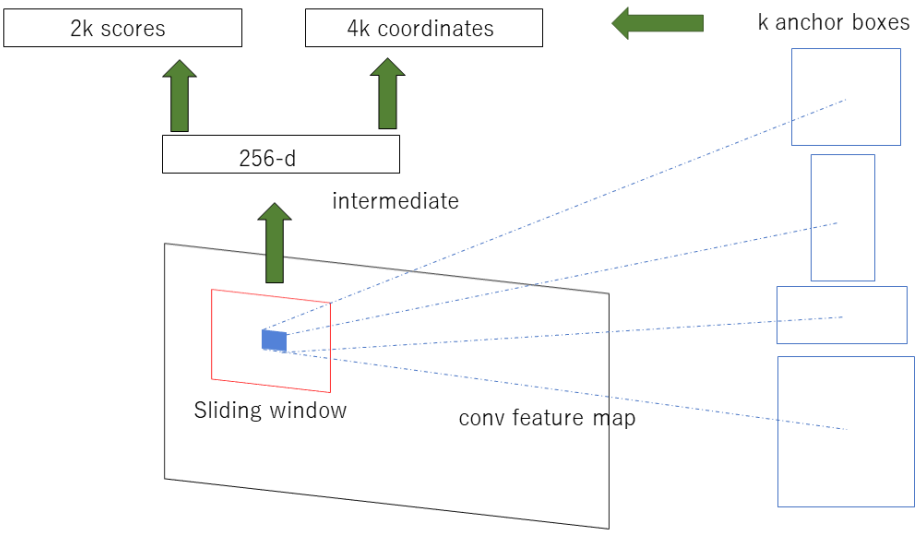


図 2.7: RPN の学習と anchor boxes の生成

2.5　transformer

2017 年に発表された”Attention is All You Need”という自然言語処理に関する論文内で提唱された深層学習のモデルであり，主に翻訳などの文章を他の文章に変換するモデルで使われる．エンコーダやデコーダにそれまでの主流であった CNN や RNN を用いずに Attention というモデルのみで，早い処理速度を持ちながら，高い精度を達成した．特徴として，再帰や畳み込みがない，並列化がしやすい，Transformer の汎用性が高いといったものが挙げられる．従来のモデルの欠点として，訓練時に並列で処理を行うことができないというものがあり計算量は文章の長さに従って $O(N)$ または $O(\log x)$ で計算量が増え，長文での依存関係が掴みづらかった．Transformer は CNN や逐次的な RNN を使わずに Attention のみを用いることで並列化を簡単にし，計算量を $O(1)$ に抑えた．これによって，訓練の速度が早くなっている．Transformer の構造を（図 2.8）に示す．（図 2.8）の構造は，左がエンコーダ，右がデコーダを表している．

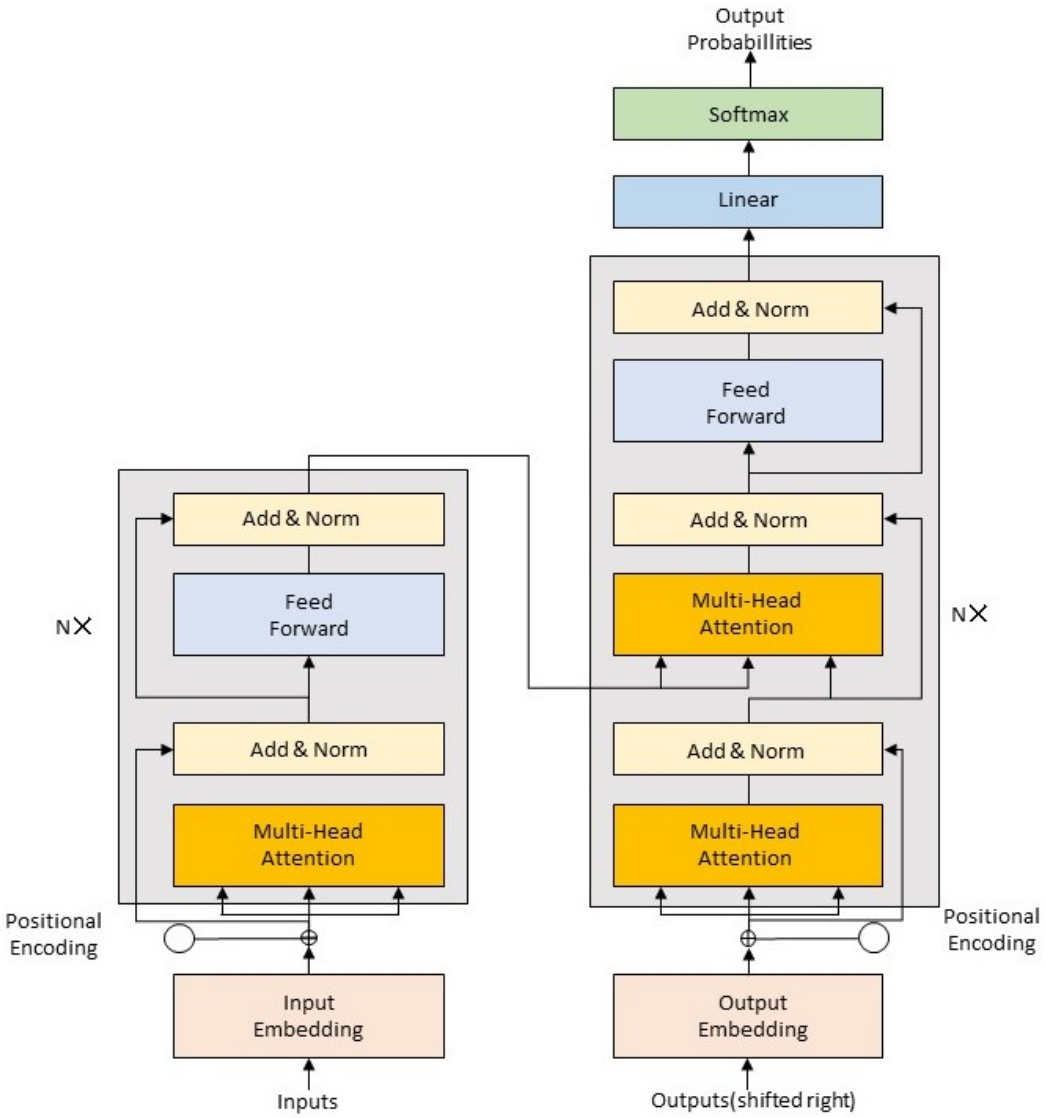


図 2.8: Transformer のアーキテクチャ

2.5.1　encorder

（図 2.8）中に n ×とあるが，ここでは n 回同じ処理を行う．通常 N=6 で構成され，各層は Multi-Head Attention 層と Position-wise 全結合層の 2 つの層で構成される．

2.5.2　decoder

encoder と同様に N=6 からなる．各層はエンコーダの層の間にエンコーダからの出力を受け取る Multi-Head Attenition が追加されている．

2.5.3　Attention

文中の単語のどれに注目すれば良いのかを表すスコアで，例えば it のような指示後に対して，文章中のどの単語に注目すればよいかといったスコアを表す．Attention は Query,Key,Value の 3 つのベクトルで計算され，各単語がそれぞれのベクトルを持っている．計算は V を Q，K を使った加重和で計算される．これによって単語の潜在表現を表す．

2.6　DETR

トランスフォーマーは, 特に言語モデリングや機械翻訳などの自然言語処理タスクにおいて, 連続したデータを扱う問題に広く適用されており, 音声認識, 記号数学, 強化学習などの多様なタスクにも拡張されてきた．DETR は transformer を用いて作成された初めての物体検出である．DETR は transformer を用いていない以前のアーキテクチャとは異なり，大幅に単純化，合理化された．これにより，Bounding box のデフォルト設定，NMS の閾値の事前設定といったものが不要になった．また，性能は最適化された Faster R-CNN の最先端手法の性能と一致している．DETR のアーキテクチャを（図 2.9）に示す．

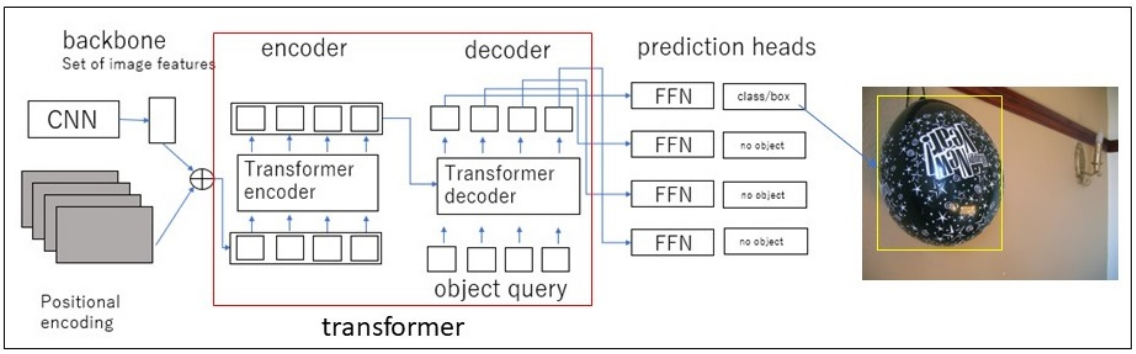


図 2.9: DETR のアーキテクチャ

DETR の主な構成要素は，backbone,transformer,FFN からなる．それぞれの役割を説明する．

2.6.1　backbone

インプット画像に対して CNN での畳み込みを行い d 次元の特徴マップに変換する．また，Transformer の input にするために次元削減も行う． これによって，d 個の特徴マップが生成される．

2.6.2　positional encoding

Transformer への入力の際に reshape を行い，位置情報が失われてしまっている．そのため，ここでの処理で位置情報を付与する．

2.6.3　Encoder-Decoder

backbone では CNN による画像特徴量の抽出, 畳み込みで特徴量の次元を削減, 縦横をまとめて一次元に変形, 特徴量マップの作成を行う．この特徴量から attention 機構を用いて各物体の位置や種類の情報へと変換し, 事前に決められた個数 N の物体を予測する．N は DETR [3] 内では 100 となっている．最後に，prediction heads で Feed Forward Network を用いて Transformer の出力を物体の位置座標，クラスラベルにデコードしている．DETR は，transformer を用いることによって精度を維持しながら並列計算による高速な処理を可能にした．

また, DETR の工夫として, Hungarian Loss,Parallel Decoding,Generalized IoU Loss と言ったものが挙げられる．

2.6.4　Hungarian Loss

学習において，複数の予測されたオブジェクトを Ground truth に対して対応付け，スコアリングする必要がある．これは二部マッチング問題といい．これを解く手段として，Hungarian Loss が用いられている．これは，Ground truth の集合 y,DETR の 推論結果 $\hat{y}$ をもとに最適なマッチングパターン $\hat{\sigma}$ におけるロスを計算するものである．L_{box} は IoU に距離の要素を追加した Generalized IoU ロスと回帰ロスの和である．計算を（式 2.1）に示す．

$$\pounds_box(b_i, \hat{b}_{\sigma\ (i)}) = \lambda_{iou}\pounds_{iou}(b_i, \hat{b}_{\sigma_{(i)}}) + \lambda_{L1} \parallel b_i - \hat{b}_{\sigma(i)} \parallel_1 \tag{2.1}$$

クラスのロスと Lbox とのロスの和を L_{match} とする．計算を（式 2.2）に示す．

$$\pounds_{match}(y_i, y_{\hat{\sigma(i)}}) = -1_{\{c_i \neq \emptyset\}}\hat{p}_{\sigma(i))}(c_i) + 1_{\{c_i \neq \emptyset\}}\pounds_{box}(b_i, \hat{b}_{\sigma(i)}) \tag{2.2}$$

この L_{match} を予測と正解ラベルのすべてで行い，対応関係を得る．

$$\hat{\sigma} = \overset{argmmin}{\sigma \in S_N} \sum_{i}^{N} \pounds_{match}(y_i, \hat{y}_{\sigma(i)}) \tag{2.3}$$

$\hat{\sigma}$ によって得た物体の対応関係から予測と正解ラベルのマッチングを行う．計算を（式 2.4）に示す．

$$\pounds_{Hungarian(y,\hat{y})} = \sum_{i=1}^{N}[-\log \hat{p}_{\hat{\sigma(i)}} + 1_{\{c_i \neq \emptyset\}}\pounds_{box}(b_i, \hat{b}_{\hat{\sigma}}(i))] \tag{2.4}$$

2.7　MIL

機械学習では教師あり学習と呼ばれる教師データを元に学習を行うものに対して，弱教師あり学習と呼ばれるものが存在する．これは教師データとして，矩形で囲ったものではなく，画像内に目的の物体が写っているかというようなヒントを与えるものである．MIL はこれにあたり，多数の negative のなかに最低でも一つ以上の正解を含む positive bags と，正解を含まない negative bags という複数の集合をもとに正のインスタンスを求めるというものである．先行研究 [7] ではこれを用いて物体の領域候補の提案を行っている．

$$labelvariable y \epsilon \{0, 1\}, instance x \epsilon R, KbagX = \{x_1, x_2 ..., x_k\} \tag{2.5}$$

（図 2.10）に概要図を示す．instance には個々のラベルが存在している．しかし，実装ではこれを知る必要はなく，分割する前のラベルがわかれば良い．

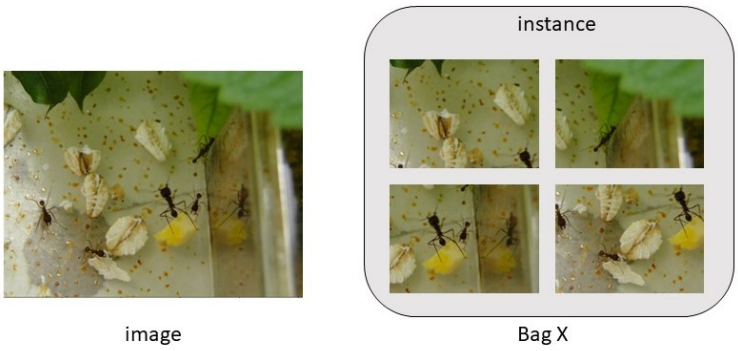


図 2.10: MIL の例

2.8　click supervision

[7] における研究では，中心をクリックすることでアノテーションを行う click supervision が提案されている．これは，物体の中心点 (center-click annotation) に MIL(Multiple Instance Learning) [9] を適用することでバウンディングボックスの定位を決定する手法で，これによってアノテーション時間を大幅に短縮した．

結果として，この手法はわずかなアノテーションの追加作業で弱教師化技術で生成された検出器よりも優れた性能を発揮し，手動で描画されたバウンディングボックスから学習された検出器に近い学習結果が得られた．

click supervision では，クラウドソーシングを用いて大量のクリックデータを収集し，それをもとにモデルの学習を行う．そこで作業結果のクオリティを担保するための workflow が用意されている．これを（図 2.11）に示す．

workflow の流れを説明する．instruction で click annotation についての説明を行い，次に Annotator training を行う，これは（図 2.12）に示されているような画像に対して，黒い部分の面積の中心をアノテーションしてもらい，実際の中心とのずれのフィードバックを受け取る．これを中心との誤差が 20px 以下になるまで繰り返す．

また，ここで求めた中心との誤差の標準偏差 σ_{bc} を実際の作業時に使用する．

このテストをクリアしたら，実際の作業へと移行する．Annotationg images では，特定の枚数を単位として，アノテーターに物体の中心のクリックをしてもらう．ここでは，Quality を担保するために，もともと Ground Truth を持っているデータをバッチごとにランダムに混ぜて精度計測を行う．この時，精度が一定に満たなかった人のデータは

受け付けない．例えば，バッチを 20 枚とし，その中に 2 枚 Ground Truth を持っているデータをランダムに含ませ，このデータの精度が一定の基準を超えていれば 18 枚分のクリックデータを受け取る．

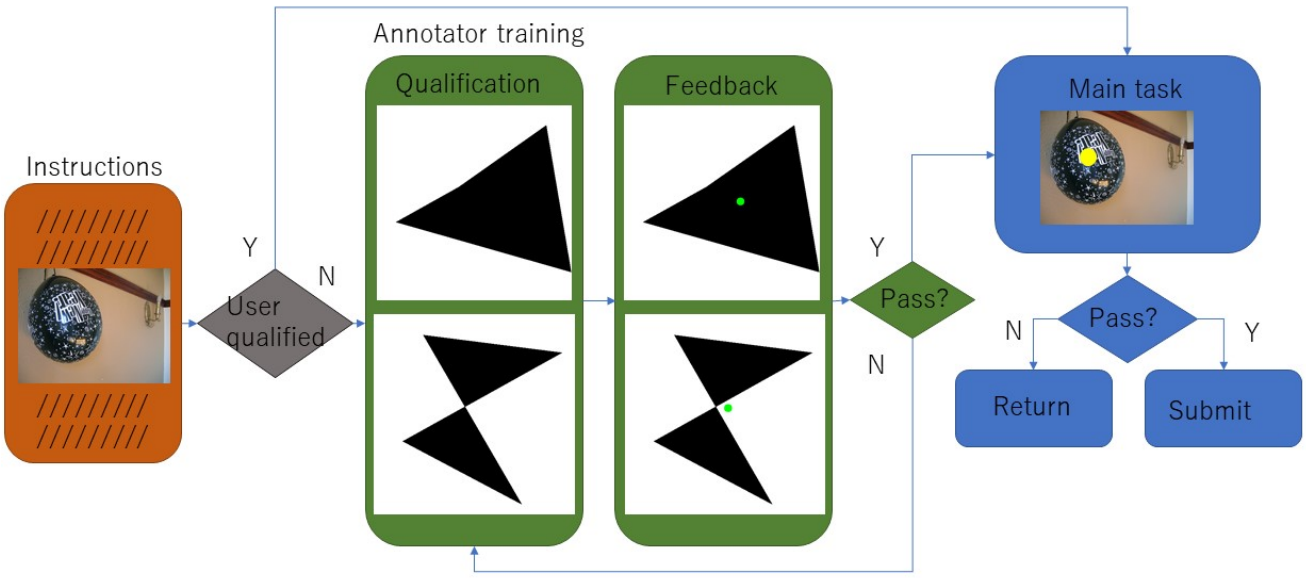
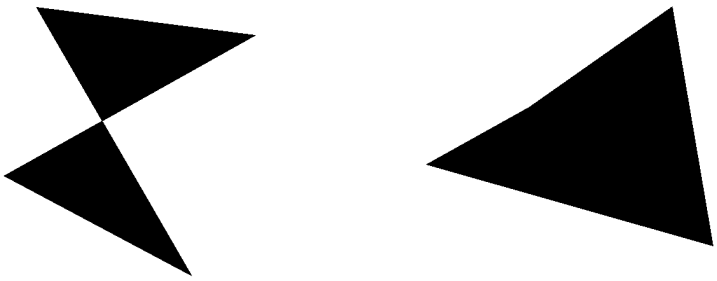


図 2.11: click annotation をクラウドソーシングで行う Workflow



[1]　　[2]

図 2.12: annotator training 用に生成された画像

click supervision でアノテータによって与えられたデータをもとに, Multiple Instance Learning による物体検出器の作成を行う.

Multiple Instance Learning (MIL) は，弱教師あり学習の一種で，1 枚の大きい画像から複数枚の画像 (パッチ) を切り出して bag を作り，これを bag 内に少なくとも 1 つポジティブなインスタンスが存在する bag とそうでない bag に分け，学習するといったものである．この MIL を用いて物体検出器の作成を行う.

そのため，Pre-train された AlexNet CNN と SVM を使って下記の 2 ステップを交互に回す.

- re-localization　識別機 A を使い物体候補を探す．この時，物体候補 p のスコアは識別機 A と Objectness[2] を用いた物体候補らしさ O を使い (式 2.6) に示す計算で求める．p は提案された領域である.

$$S_{ap}(p) = \frac{1}{2} \cdot A(p) + \frac{1}{2} \cdot O(p) \tag{2.6}$$

また，クリックに最も近いオブジェクトを選択するだけでは正確なバウンディングボックスは得られない．そこで，クリックした点 c と annotator training 時に求めた σ_{bc} を用いて，スコア関数 Sbc によって中心尤度を求める．これを (式 2.7) に示す．p は提案された領域，c_p はその中心点，c はクリックした点である．σ_{bc} は c_p が c から離れるにつれて値が小さくなるよう制御するものである.

$$S_{bc}(p; c, \sigma_{bc}) = e^{-\frac{||c_p - c||^2}{2\ \sigma_{bc}^2}} \tag{2.7}$$

S_{ap} と S_{bc} を積としてバウンディングボックスを定位する．これを (式 2.8) に示す.

$$S_{ap}(p) \cdot S_{bc}(p; c, \sigma_{bc}) \tag{2.8}$$

- re-trainning

(1) で示した位置を Positive として，識別機 A を SVM にて学習させる.

一定回数イテレーションを回した後，re-trainning で識別機 A を Fast R-CNN にして再学習する．

しかし本研究では，これらの一連の学習を DETR で行った．

第 3 章

DETR を用いた物体検出の One-click supervision

3.1 動機

[7] では，物体検出器の作成に MIL と物体の中心をクリックさせたポイントのデータを用いていたが，この MIL では．一枚の画像内から複数の画像を切り出し，bag を作り，この bag 内に posotive な instance が含まれるかどうかという 2 値分類を行い，この精度を高めることによって領域の提案をうまくできるようにするものである．私はこの部分に物体検出で使われている領域の提案を用いることで同じような振る舞いができると考えた．これができることで，精度の向上が著しい物体検出の精度を利用でき，また，学習済みのモデルの利用などが簡単であるため学習の簡易化ができると考えた．そこでまずは Faster R-CNN ないの RPN という領域提案を行う部分を用いた実験を行った [11]．しかし，この RPN で作成した教師データを用いて実験を行ったところ，MIL を使用したときの精度ど比較した時と比べて，精度は低くなる結果となった．原因の一つとして，領域の提案が物体の周辺にかなり重なっていて，これを元に教師データを作成することで，検出が簡単な物体であっても教師データの精度があまり高くならないことが考えられた．[11] の研究で，領域の候補が右下に集中したことによって教師データのバウンディングボックスが右下に集中している例を（図 3.1）に示す．（図 3.1）は，上部のクリックデータ周辺にはボックスがなく，右下の位置にボックスが重なっているのがわかる．このように物体周辺に領域候補が生成されないことによって，悪い教師データが作成され，学習の精度が上がらなかった事が考えられる．次に，近年注目を集めてい

図 3.1: Faster R-CNN を用いた One-Click Supervision の一例

る transformer をもちいた物体検出の手法のうちの一つである DETR を利用してみることを考えた．この DETR で使われている Transformer では，Attention 機構が使われている．この Attention 機構を用いることによって，広範囲の関係性を見ながら領域の提案を行うため，faster r-cnn よりも精度の高い教師データを作成することができると考えた．そこで本研究では，MIL を DETR の領域の提案に置き換えて教師データの作成を行ったときに学習の様子を見た．

3.2　手法の説明

[7] では MIL とクリックデータを用いて教師データの作成を行っていた．本稿では MIL による領域提案の代わりに DETR のを用いて物体検出の one-click supervision を実装した．また，仮想的なクリックデータとして Open Image Dataset の教師データによる中心点を用いた．この予測結果の中心点と，仮想クリックデータの中心点との距離が最も近いものをそれぞれ教師データとした．この時作成する教師データのクラスはは，学習済みのクラスには無いものを選択した．これは学習済みのクラスによって教師データを作成することは明らかであると考えたからである．

第 4 章

実験

4.1　実験データ

本実験で使用したデータセットは，Open Images Dataset という Google によって提供されるデータセットの中から Ant,Bee,Fish,Flower,Balloon の画像を各 100 枚，合計 500 枚の画像を使用した．本実験では COCO データセットを学習済みの DETR を使用したが学習した画像のクラスにはこれらは含まれていない．

4.2　実験方法

Open Image Dataset の教師データから中心データを取り出しこれをクリックデータと想定する．これと DETR の予測結果から教師データの作成を行う．その後，作成した教師データと Open Image Dataset の教師データでそれぞれ DETR の Fine-tuning を行い，学習結果の比較を行った．予測結果からの教師データの選択は中心点から距離が最も近いものを選択した．

4.3　評価指標

物体検出の精度の評価指標に mAP(Mean Average Precision) がある．これについて説明する．　mAP とは，AP の平均 (mean) であり，物体検出における mAP は，クラスごとにおける AP の平均値である．これについて説明する．　まず，物体検出にあける正誤の判定は，IoU(Intersection over Union) を用いて行う．これは，検出したバウンディングボックスと正解のバウンディングボックスの重なりを表す指標で，以下の図

のように求められる．AreaofOverlap は 2 つのボックスの共通している部分の面積で，AreaofUnion は全体の面積である．

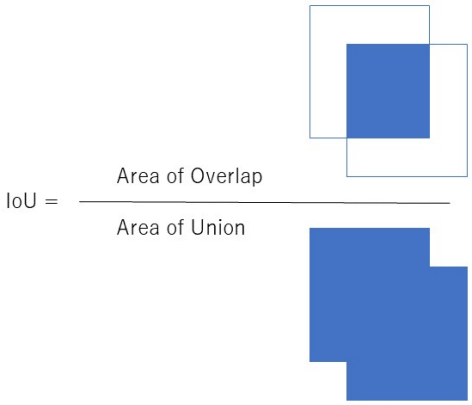


図 4.1: IoU の計算

これにより，二つの領域の重なりを 0 から 1 の値で表すことができる．本研究における評価ではこれに閾値を設け，超えたものを正解 (positive) としている．次に，P/R 曲線について説明する．

P/R 曲線とは，Precision/Recall 曲線と呼ばれるもので，Precision と Recall は次の（式 4.1）で表される．

$$Precision = \frac{\text{正しく検出した数}}{\text{全ての検出した数}} \quad Recall = \frac{\text{正しく検出した数}}{\text{全ての実際のボックス数}} \tag{4.1}$$

P/R 曲線の導出では，まず全てのテストデータから検出したボックスをクラス別に確信度の高い順にソートする．次にボックスの正誤を IoU で判定する．さらにこれらの Precision と Recal を算出する．導出例を (表 4.1) に示す．

この表をもとに P/R 曲線を求める．(図 4.2) にこれを示す．

次に，(図 4.2) のグラフを積分して AP を求める．

AP とは，Average(平均) Precision(適合率) のことで，(式 4.2) に表すと，

$$\int_0^1 p(r)dr \tag{4.2}$$

となる．(式 4.2) 内にある関数 $p(r)$ は Precision/Recall 曲線である．

実際にはこの積分を簡単にするため，(式 4.3) によって凹凸を均す．結果を（図 4.3）に示す．

表 4.1: 物体検出の各種モデル

順位	確信度 (%)	正誤	Precision	Recall
1	100	○	1/1=1	1/10=0.1
2	99	○	2/2=1	2/10=0.2
3	96	×	2/3=0.67	2/10=0.2
4	92	×	2/4=0.5	2/10=0.2
5	89	×	2/5=0.4	2/10=0.2
6	88	○	3/6=0.5	3/10=0.3
7	80	○	4/7=0.57	4/10=0.4
8	70	○	5/8=0.63	5/10=0.5
:	:	:	:	:

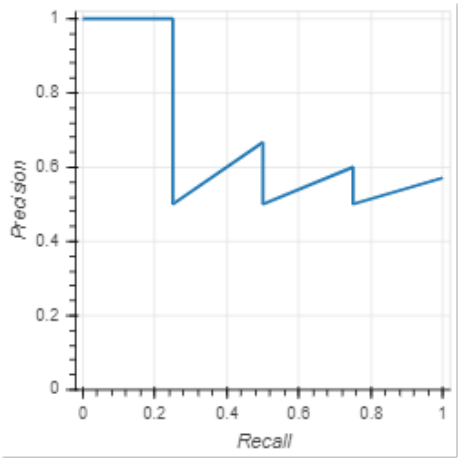


図 4.2: P/R 曲線のグラフイメージ

$$Pinterp(r) = \max_{\tilde{r}:\tilde{r}\geq r} p(\tilde{r}) \tag{4.3}$$

この (式 4.3) に r=(0,0.1,0.2・・・,1) と 0.1 刻みで 11 個の点を取り，縦×横の掛け

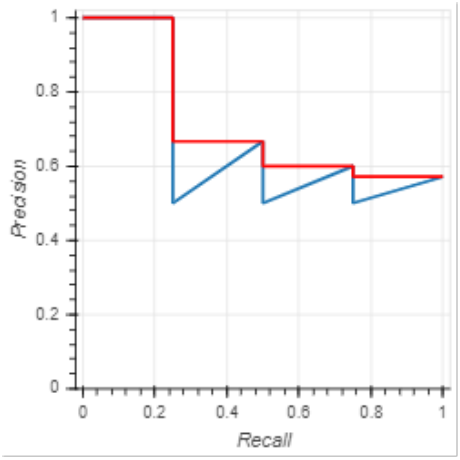


図 4.3: 簡略化された P/R 曲線

算で計算を求める．結果を（図 4.4）に示す．

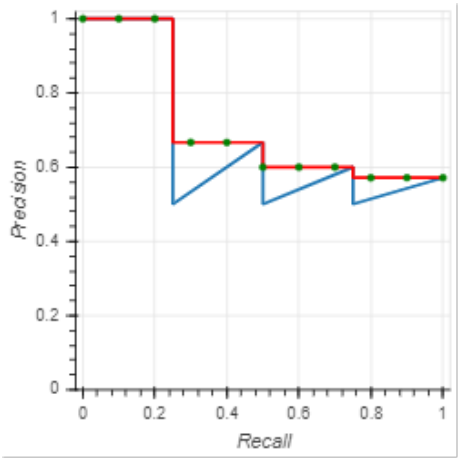


図 4.4: 11 個の点が打たれた P/R 曲線

また，実験の評価指標を（表 4.2）に示す．

表 4.2: mAP の評価指標

指標	IoU の閾値	説明
AP	0.50:0.95	IoU の閾値を 0.50 : 0.95 をとしたときの mAP
AP50	0.50	IoU の閾値を 0.50 をとしたときの mAP
AP75	0.75	IoU の閾値を 0.75 をとしたときの mAP
APs	0.50:0.95	対象の領域を 32^2px 以下に限定したときの mAP
APm	0.50:0.95	対象の領域を 32^2px から 96^2px に限定したときの mAP
APl	0.50:0.95	対象の領域を 96^2px 以上に限定したときの mAP

4.4　実験

学習済みの DETR をと Open Image Dataset からダウンロードした画像と教師データの中心点を用いて click supervison による bounding box の作成を行う．実験に使用した画像は ant,ablloon,bee,fish,flower 5 つのクラスそれぞれ 100 枚からなる合計 500 枚のデータセットであり，これらのクラスは学習済みの DETR のクラスには含まれない．ダウンロードした画像とその教師データを描画した様子を（図 4.5）に示す．

この，Open Image Dataset の画像内の物体検出を DETR を用いて行った．また，検出時の閾値を 0 に設定することによってすべての領域の候補を表示している．結果を（図 4.6）に示す．

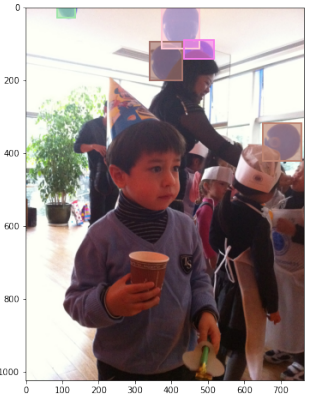
[1]

[2]

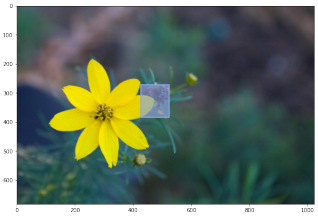
[3]

[4]

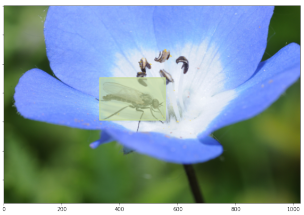
[5]



[6]

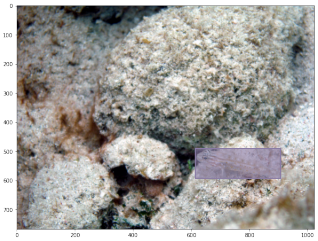


[7]

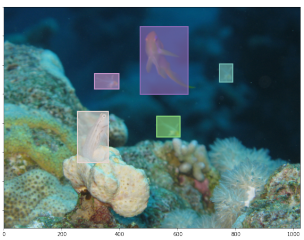


[8]

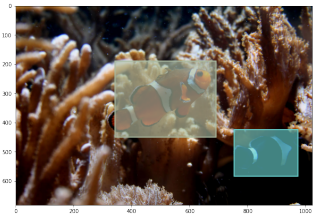
[9]



[10]



[11]



[12]

[13]

[14]

[15]

図 4.5: Open Image Dataset とその教師データ

表 4.3: Open Image Dataset の AP

AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
0.137	0.259	0.110	0.022	0.020	0.252

表 4.4: 作成した教師データの AP

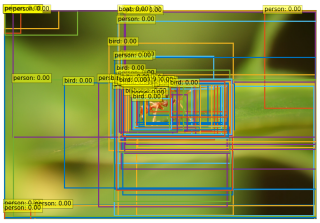
AP	AP_{50}	AP_{70}	AP_S	AP_M	AP_L
0.075	0.149	0.065	0.004	0.024	0.136

教師データの中心をクリックデータと見立てて，この点と中心が近い予測結果を教師データとした．これによって作成した教師データを（図 4.7）に示す．

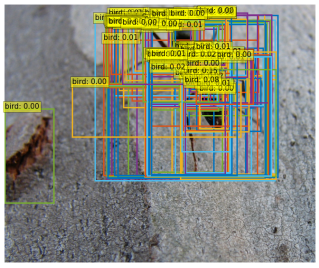
Open Image Dataset からダウンロードした教師データと click supervision を用いて作成された教師データを用いてそれぞれ DETR の fine-tuning を行った．Open Image Dataset による学習結果を（図 4.8）,click supervision による学習結果を（図 4.9）に示す．

Open Image Dataset の教師データによる学習を行い，イテレーション 50 回時点での各条件での Avarage Precisionl を（表 4.3）, 作成した教師データによる学習を行い，イテレーション 50 回時点での各条件での Avarage Precision を（表 4.4）にそれぞれ示す．

[1]

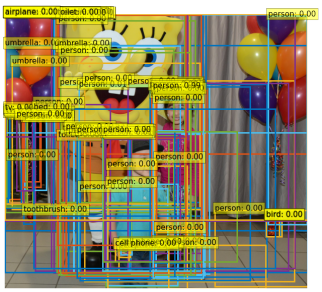


[2]

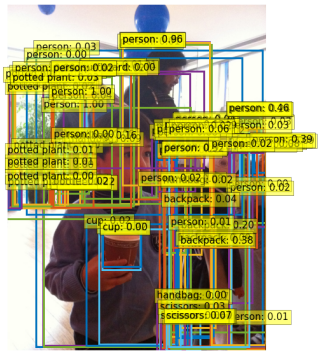


[3]

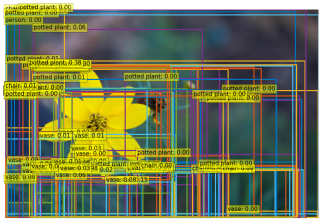
[4]



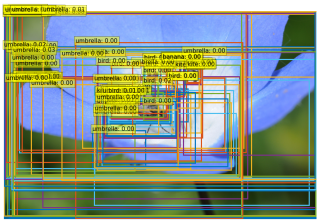
[5]



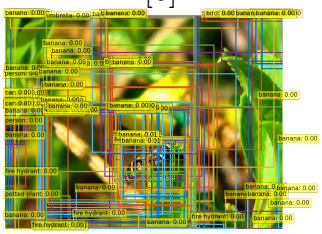
[6]



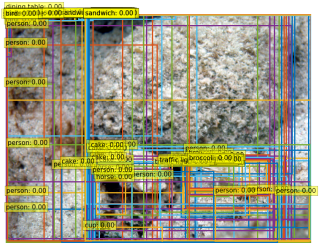
[7]



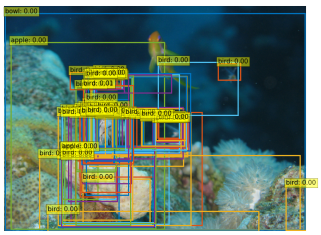
[8]



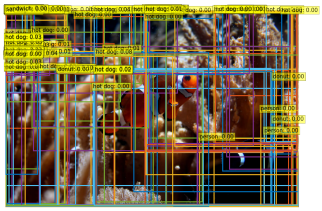
[9]



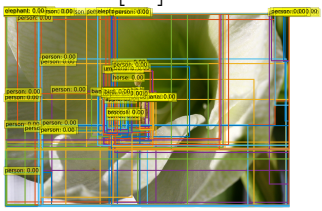
[10]



[11]



[12]

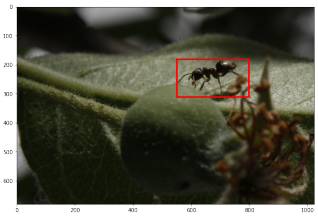


[13]

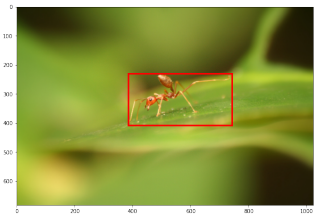
[14]

[15]

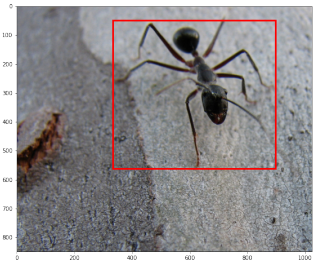
図 4.6: Open Image Dataset とその教師データ



[1]



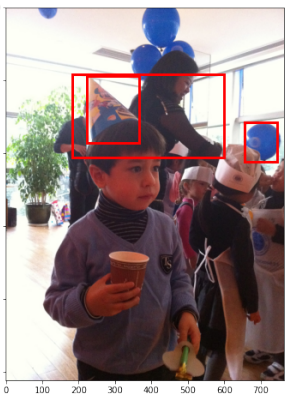
[2]



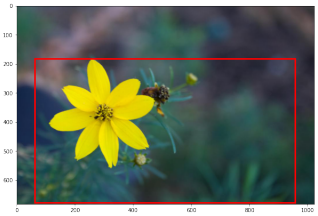
[3]

[4]

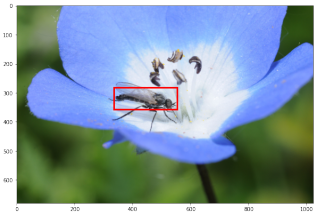
[5]



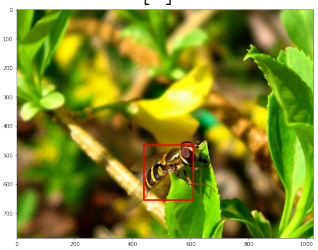
[6]



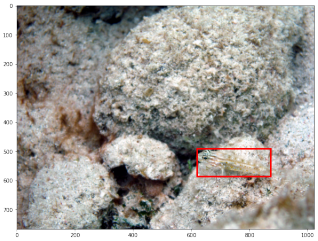
[7]



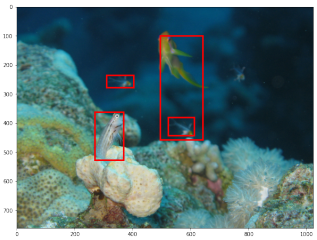
[8]



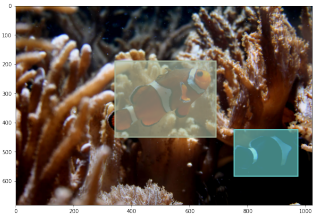
[9]



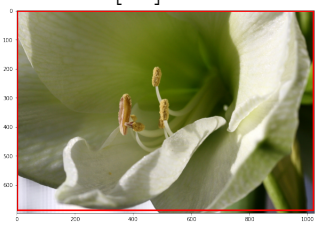
[10]



[11]

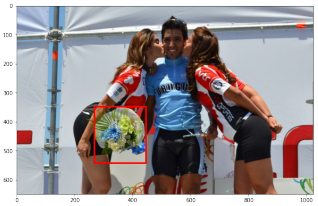


[12]



[13]

[14]



[15]

図 4.7: Open Image Dataset とその教師データ

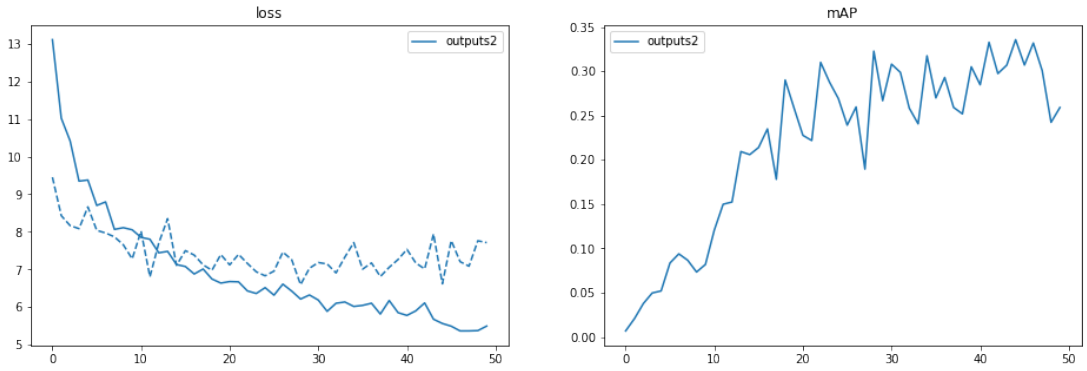


図 4.8: Open Image Dataset の教師データによる学習結果

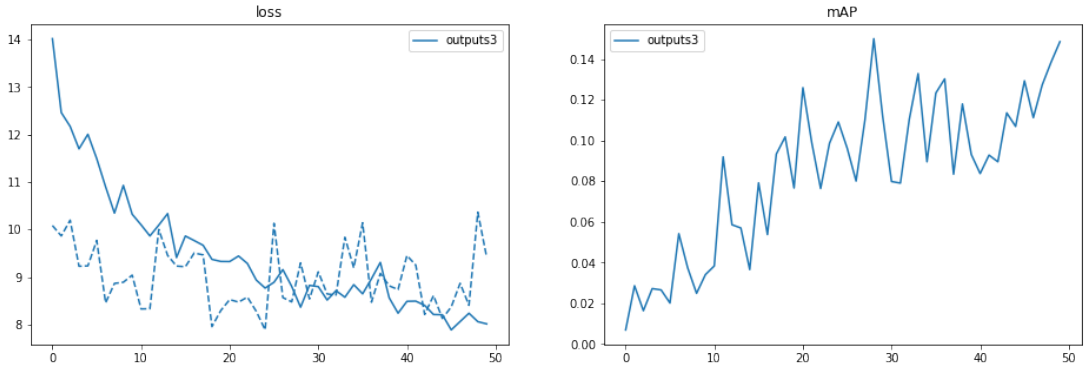
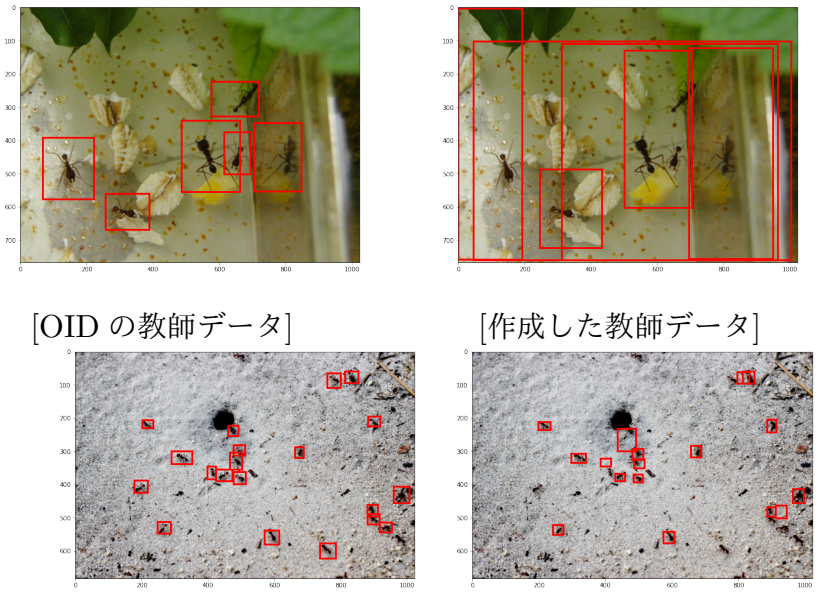


図 4.9: 作成した教師データによる学習結果

第 5 章

考察

Open Image Dataset の教師データと作成した教師データの mAP を比較した時，作成したものの精度は低かった．これの原因は，教師データ作成がうまくいかなかったからであると考えられる．例を（図 5.1）に示す．



[OID の教師データ]　　[作成した教師データ]

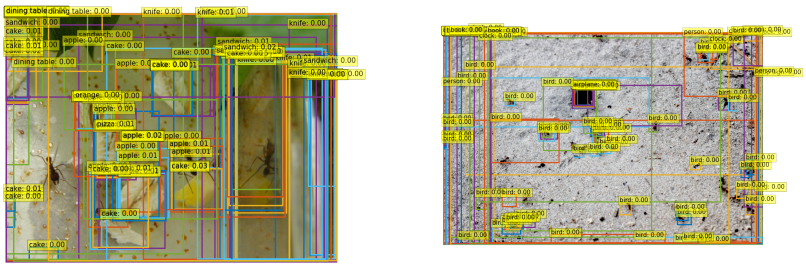
[OID の教師データ]　　[作成した教師データ]

図 5.1: 教師データの比較

左の画像が OID の教師データで右の画像が作成した教師データである，上の比較画像では画像のサイズがかなりずれている．また，下の比較画像では右下あたりの蟻が検出

されずに何もない部分にボックスが生成されてしまっているのがわかる．次に，これらの画像がどのような予測に基づいているのかを見ていく．

それぞれの DETR での閾値が 0 のときの検出結果を見る．検出結果を（図 5.2）に示す．



[1]　　[2]

図 5.2: DETR の検出結果

（図 5.2）の [1] は，アリをうまく検出できているボックスがほとんどなく，蟻の付近では実際の蟻よりも大きな領域候補が提案されていることがわかる．次に，[2] では，アリが実際にいるところには物体が検出されていない部分が多くあった．これらのように背景が複雑であったり，物体そのものの検出が難しい画像の場合，教師データの作成がうまくなされなかった．また，DETR 内での attention 機構による他の領域提案を考慮した領域の提案により，物体の付近に領域候補が集中するといった点の改善が予想されたが，（図 5.2）の [1] のように，画像によっては特定の箇所に領域の候補が集中している様子も見られた．

作成した教師データと Open Image Dataset を比較したときに精度が低くなってしまった原因として，これらのような，検出がうまくできていない領域候補の提案を元に教師データを作成してしまったため，Open Image Dateset のものほど精度が上がらなかったと考えられる．

また，DETR の領域提案を見ていると，このような元の教師データの各ボックスごとにどのようなボックスに置き換わったのかを見ていく．学習結果がうまくいかなかった要因として，教師データの近くに DETR に用領域候補の生成が行われなかったこと，砂場に小さいアリがいくつかいるような検出の難しい画像では大きく異なるボックスが生

成されていること，中心点との距離の指標のみでは物体のサイズが考慮されないことを原因としてあげる．蟻の足の部分がボックスとして検出されていなかった点も上げ

Faster R-CNN のときに感じた予測結果が近い箇所に集中するような点は DETR の attention 機構の特性のためか見られなかった．

第 6 章

結論

本研究では，DETR を用いた物体検出の one-click supervision を行った．この時．既にある教師データの中心をクリックと見立てて DETR の領域候補との距離を元に教師データの作成を行った．結果として mAP の向上が見られ学習の成果は見られたが，手作業によって作成したバウンディングボックスと比較すると精度は劣るものとなった．

謝辞

本論文の執筆にあたり，指導教官の新納浩幸教授には，研究の着想と調査において多くのご指導をいただきました．深く感謝申し上げます．また，所属する新納ゼミのみなさまには論文執筆，研究内容について多くのご支援とご助言をいただきました．お礼申し上げます．ありがとうございました．

参考文献

[1] Dim P. Papadopoulos, Jasper R. R. Uijlings, Frank Keller1 Vittorio Ferrari, University of Edinburgh, Google Research Training object class detectors with click supervision.(CVF),(2017)

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, Attention Is All You Need,(NIPS),(2017)

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, Sergey Zagoruyko, Paris Dauphine University,Facebook AI, End-to-end object detection with Transformers,(ECVA),(2020)

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, UC Berkeley:Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5),Conference on Computer Vision and Pattern Recognition (CVPR),(2014).

[5] Ross Girshick, Microsoft Research:Fast R-CNN,International Conference of Computer Vision(ICCV),(2015).

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun:Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,Neural Information Processing Systems 28(NIPS),(2015).

[7] Dim P. Papadopoulos, Jasper R. R. Uijlings, Frank Keller1 Vittorio Ferrari, University of Edinburgh, Google Research:Training object class detectors with click supervision,Conference on Computer Vision and Pattern Recognition (CVPR),(2017).

[8] Bogdan Alexe, Thomas Deselaers, Vittorio Ferrari Computer Vision Laboratory,

ETH Zurich:What is an object ?,Conference on Computer Vision and Pattern Recognition (CVPR),(2010).

[9] Boris Babenko, Ming-Hsuan Yang, Serge Belongie:Visual Tracking with Online Multiple Instance Learning,Conference on Computer Vision and Pattern Recognition (CVPR),(2009).

[10] 宮本　圭一郎, 大川　洋平, 毛利　拓也『PyTorch ニューラルネットワーク実装ハンドブック』秀和システム,(2019).

[11] Tomoki Hirano, Hiroyuki Shinnou:One-click supervision using Faster R-CNN (SIG Technical Reports),(2021).

付録

A　DETR を用いた物体検出の One-click supervision のコード

実験は Google Colaboratory 上で行い，ディレクトリの構造は以下のようになっている．/content/drive/MyDrive/data detr Open Image Dataset から教師データをダウンロードするソースコードを A.1 に示す.

ソースコード A.1: data.ipynb

```
1  from google.colab import drive
2  drive.mount('/content/drive')
3
4  !git clone https://github.com/EscVM/OIDv4_ToolKit.git
5  !pip install urllib3==1.25.11 folium==0.2.1
6  !pip install -r OIDv4_ToolKit/requirements.txt
7
8  !python OIDv4_ToolKit/main.py downloader -y --classes Ant Bee Fish
       Flower Balloon --type_csv train --limit 100
9  !python OIDv4_ToolKit/main.py downloader -y --classes Ant Bee Fish
       Flower Balloon --type_csv validation --limit 100
10
11 import os, json, glob
12
13 def OID2JSON(OIDFiles, saveName, subset):
14   """
15 ␣␣アノテーションを OpenImage␣format(txt)から COCO␣format(json)に変換
16
17 ␣␣Parameters
18 ␣␣----------
19 ␣␣OIDFiles␣:␣string
20 ␣␣␣␣のフォルダパス OpenImageDataset
21 ␣␣saveName␣:␣string
```

```
␣␣␣␣保存ファイル名（json）
␣␣subset␣:␣string
␣␣␣␣変換したい。、、のいずれかを指定。type_csvtrainvalidationtest
␣␣"""
  attrDict = dict()
  # 要素の設定 categories
  attrDict['categories'] = []
  categories = sorted(os.listdir(os.path.join(OIDFiles, 'Dataset',
      subset)))
  for i in range(len(categories)):
    attrDict['categories'].append({'supercategory': 'none', 'id': i, '
        name': categories[i]})

  images = list()
  annotations = list()
  filenames = list()
  image_id = 1
  anno_id = 1
  for category in attrDict['categories']:
    for jpg_file in glob.glob(os.path.join(OIDFiles, 'Dataset', subset
        , category['name'], '*.jpg')):
      filename = os.path.splitext(os.path.basename(jpg_file))[0]
      # カテゴリ全体で同じファイル名が存在する場合、とをリネーム
          imageanno
      if filename in filenames:
        rename_filename = filename + '_' + str(image_id)
        os.rename(jpg_file, os.path.join(OIDFiles, 'Dataset', subset,
            category['name'], rename_filename + '.jpg'))
        os.rename(os.path.join(OIDFiles, 'Dataset', subset, category['
            name'], 'Label', filename + '.txt'),
                  os.path.join(OIDFiles, 'Dataset', subset, category['
                      name'], 'Label', rename_filename + '.txt'))
        filename = rename_filename
      filenames.append(filename)
      # 要素の設定 images
      # ※ではとを使わないので、DETRheightwidth'none'を設定
      image = {'file_name': filename + '.jpg', 'height': 'none', '
          width': 'none', 'id': image_id}
      images.append(image)
      # 要素の設定 annotations
      anno_path = os.path.join(OIDFiles, 'Dataset', subset, category['
```

```
            name'], 'Label', filename + '.txt')
55        with open(anno_path) as f:
56          for line in f:
57            splitline = line.split('␣')
58            # カテゴリが要素に存在しないバウンディングボックスは使わない
                  categories
59            if splitline[0] in [d.get('name') for d in attrDict['
                  categories']]:
60              # の座標は OpenImage(xmin, ymin, xmax, ymax)、の座標は
                    COCO(x, y, width, height)
61              x1 = int(float(splitline[1]))
62              y1 = int(float(splitline[2]))
63              x2 = int(float(splitline[3])) - x1
64              y2 = int(float(splitline[4])) - y1
65              # はピクセル数 area(float)
66              area = float(x2 * y2)
67              # は segmentation(x1, y1, x2, y2, ...)と順番に定義
68              segmentation = [[x1, y1, x1, (y1+y2), (x1+x2), (y1+y2),
                    (x1+x2), y1]]
69              annotation = {'iscrowd': 0, 'image_id': image_id, 'bbox':
                    [x1, y1, x2, y2], 'area': area,
70                            'category_id': category['id'], 'ignore': 0,
                                  'id': anno_id, 'segmentation':
                                  segmentation}
71              anno_id += 1
72              annotations.append(annotation)
73        image_id = image_id + 1
74
75    attrDict['images'] = images
76    attrDict['annotations'] = annotations
77    attrDict['type'] = 'instances'
78    jsonString = json.dumps(attrDict)
79
80    with open(saveName, 'w') as f:
81      f.write(jsonString)
82
83  OID2JSON('/content/OID', 'custom_train.json', 'train')
84  OID2JSON('/content/OID', 'custom_val.json', 'validation')
85
86  import shutil
87
88  source_train_paths = glob.glob(os.path.join('/content/OID/Dataset', '
```

```
    train', '**/'))
source_val_paths = glob.glob(os.path.join('/content/OID/Dataset', '
    validation', '**/'))
train_path = '/content/data/custom/train2017/'
val_path = '/content/data/custom/val2017/'
convert_anno_path = '/content/data/custom/annotations/'

# ディレクトリ作成
os.makedirs(train_path, exist_ok=True)
os.makedirs(val_path, exist_ok=True)
os.makedirs(convert_anno_path, exist_ok=True)
# 移動 train
for source_train_path in source_train_paths:
  for img_path in glob.glob(os.path.join(source_train_path, '*.jpg')):
    shutil.move(img_path, train_path)
# 移動 val
for source_val_path in source_val_paths:
  for img_path in glob.glob(os.path.join(source_val_path, '*.jpg')):
    shutil.move(img_path, val_path)
# 移動 anno
shutil.move('/content/custom_train.json', convert_anno_path)
shutil.move('/content/custom_val.json', convert_anno_path)
```

DETR による click supervision と学習を行うコードを A.2 に示す.

ソースコード A.2: supervision.ipynb

```
from google.colab import drive
drive.mount('/content/drive')

!git clone https://github.com/EscVM/OIDv4_ToolKit.git
!pip install urllib3==1.25.11 folium==0.2.1
!pip install -r OIDv4_ToolKit/requirements.txt

import shutil
import os, json, glob
#学習がエラーのときこれを実行すると良い
!pip install torch==1.8.0+cu111 torchvision==0.9.0+cu111 torchaudio
    ==0.8.0 -f https://download.pytorch.org/whl/torch_stable.html
import torch, torchvision
import torchvision.transforms as T
import matplotlib.pyplot as plt
```

```
from PIL import Image
import requests
print(torch.__version__) # 1.8.0
print(torchvision.__version__) # 0.9.0
print(torch.cuda.is_available()) # True
torch.set_grad_enabled(False);
%cd /content/drive/MyDrive%cd detr/

# 学習済みモデルの取得
checkpoint = torch.hub.load_state_dict_from_url(
    url='https://dl.fbaipublicfiles.com/detr/detr-r50-e632da11.pth',
    map_location='cpu',
    check_hash=True
)

# 分類ヘッドの削除
del checkpoint['model']['class_embed.weight']
del checkpoint['model']['class_embed.bias']

# 保存
torch.save(checkpoint, 'detr-r50_no-class-head.pth')

# 可視化用クラスラベル
oid_labels = [
  'Ant',
  'Balloon',
  'Bee',
  'Fish',
  'Flower',
]
coco_labels = [
    'N/A', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus'
        ,
    'train', 'truck', 'boat', 'traffic␣light', 'fire␣hydrant', 'N/A',
    'stop␣sign', 'parking␣meter', 'bench', 'bird', 'cat', 'dog', '
        horse',
    'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'N/A', '
        backpack',
    'umbrella', 'N/A', 'N/A', 'handbag', 'tie', 'suitcase', 'frisbee',
         'skis',
    'snowboard', 'sports␣ball', 'kite', 'baseball␣bat', 'baseball␣glove
```

```
        ',
52      'skateboard', 'surfboard', 'tennis␣racket', 'bottle', 'N/A', 'wine
        ␣glass',
53      'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple', '
        sandwich',
54      'orange', 'broccoli', 'carrot', 'hot␣dog', 'pizza', 'donut', 'cake
        ',
55      'chair', 'couch', 'potted␣plant', 'bed', 'N/A', 'dining␣table', 'N
        /A',
56      'N/A', 'toilet', 'N/A', 'tv', 'laptop', 'mouse', 'remote', '
        keyboard',
57      'cell␣phone', 'microwave', 'oven', 'toaster', 'sink', '
        refrigerator', 'N/A',
58      'book', 'clock', 'vase', 'scissors', 'teddy␣bear', 'hair␣drier',
59      'toothbrush'
60  ]
61  # 可視化用 COLOR
62  COLORS = [[0.000, 0.447, 0.741], [0.850, 0.325, 0.098], [0.929,
        0.694, 0.125],
63            [0.494, 0.184, 0.556], [0.466, 0.674, 0.188], [0.301,
                0.745, 0.933]]
64
65  # 標準的なの PyTorchmean-入力画像の正規化 std
66  transform = T.Compose([
67      T.Resize(800),
68      T.ToTensor(),
69      T.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
70  ])
71
72  def box_cxcywh_to_xyxy(x):
73      """
74      (center_x, center_y, width, height)から (xmin, ymin, xmax, ymax)に座
            標変換
75      """
76      # unbind(1)で次元を削除 Tensor
77      # (center_x, center_y, width, height)*N →
            (center_x*N, center_y*N, width*N, height*N)
78      x_c, y_c, w, h = x.unbind(1)
79      b = [(x_c - 0.5 * w), (y_c - 0.5 * h), (x_c + 0.5 * w), (y_c +
             0.5 * h)]
80      # (center_x, center_y, width, height)*N の形に戻す
81      return torch.stack(b, dim=1)
```

```
82
83 def rescale_bboxes(out_bbox, size):
84     """バウンディングボックスのリスケール
85
86     """
87     img_w, img_h = size
88     b = box_cxcywh_to_xyxy(out_bbox)
89     # バウンディングボックスの～から元画像の大きさにリスケール [01]
90     b = b * torch.tensor([img_w, img_h, img_w, img_h], dtype=torch.
           float32)
91     return b
92
93 def filter_bboxes_from_outputs(outputs, threshold=0.7):
94   #print(outputs)
95   # 閾値以上の信頼度を持つ予測値のみを保持
96   probas = outputs['pred_logits'].softmax(-1)[0, :, :-1]
97   keep = probas.max(-1).values > threshold
98   probas_to_keep = probas[keep]
99   # [0, のボックスを画像のスケールに変換 1]
100  bboxes_scaled = rescale_bboxes(outputs['pred_boxes'][0, keep], im.
         size)
101  return probas_to_keep, bboxes_scaled
102
103 # 結果の表示
104 def plot_finetuned_results(pil_img, prob=None, boxes=None, labels=None
       ):
105  plt.figure(figsize=(16, 10))
106  plt.imshow(pil_img)
107  ax = plt.gca()
108  colors = COLORS * 100
109  if prob is not None and boxes is not None:
110    for p, (xmin, ymin, xmax, ymax), c in zip(prob, boxes.tolist(),
           colors):
111      ax.add_patch(plt.Rectangle((xmin, ymin), xmax-xmin, ymax-ymin,
112                                 fill=False, color=c, linewidth=3))
113      cl = p.argmax()
114      #print(labels, p)
115      text = f'{labels[cl]}: {p[cl]:0.2f}'
116      ax.text(xmin, ymin, text, fontsize=15,
117              bbox=dict(facecolor='yellow', alpha=0.5))
118  plt.axis('off')
```

```
119   plt.show()
120
121 # 物体検出
122 def run_worflow(my_image, my_model, labels, threshold=0.7):
123   # mean-入力画像の正規化 std バッチサイズ ( : 1)
124   img = transform(my_image).unsqueeze(0)
125   # モデルに反映
126   outputs = my_model(img)
127
128   probas_to_keep, bboxes_scaled = filter_bboxes_from_outputs(outputs,
          threshold=threshold)
129   plot_finetuned_results(my_image, probas_to_keep, bboxes_scaled,
          labels)
130
131   #print(torch.mean(bboxes_scaled,axis=1))
132   #print(probas_to_keep)
133
134   #閾値を超えたを boxreturn
135   #return zip(probas_to_keep,bboxes_scaled)
136   #予想するすべてのボックスを return
137   return bboxes_scaled
138
139 # 物体検出結果を表示しない ()
140 def run_makedata(my_image, my_model, labels, threshold=0.7):
141   # mean-入力画像の正規化 std バッチサイズ ( : 1)
142   img = transform(my_image).unsqueeze(0)
143   # モデルに反映
144   outputs = my_model(img)
145
146   probas_to_keep, bboxes_scaled = filter_bboxes_from_outputs(outputs,
          threshold=threshold)
147   return bboxes_scaled
148
149 #モデルのロード
150 original_model = torch.hub.load('facebookresearch/detr', '
        detr_resnet50_dc5', pretrained=True)
151
152 %matplotlib inline
153 import pycocotools.coco as coco
154 from pycocotools.coco import COCO
155 import numpy as np
```

```
156 import skimage.io as io
157 import matplotlib.pyplot as plt
158 import pylab
159 pylab.rcParams['figure.figsize'] = (10.0, 8.0)
160
161 #すべての画像から教師データを作成
162
163 dataDir='/content/drive/MyDrive/data/custom2/'
164 dataType='train2017'
165 annFile='{}annotations/custom_train.json'.format(dataDir)
166 # initialize COCO api for instance annotations
167 coco=COCO(annFile)
168 catIds = coco.getCatIds(catNms=['balloon']);
169
170 attrDict = dict()
171
172 # 要素の設定 categories
173 attrDict['categories'] = []
174 cats = coco.loadCats(coco.getCatIds())
175 attrDict = dict()
176 attrDict['categories'] = cats
177 images = []
178 annotations = []
179
180 #アノテーションされた教師データを用いて click supervision
181 imgIds = coco.getImgIds();
182 for img_id in imgIds:
183   img = coco.loadImgs(img_id)[0]
184   img_name = '%s/%s/%s'%(dataDir, dataType, img['file_name'])
185   print('Image␣name:␣{}'.format(img_name))
186   im = Image.open(img_name)
187   im = im.convert("RGB")
188   preds=run_makedata(im, original_model, coco_labels, 0.0)
189   annIds = coco.getAnnIds(imgIds=img['id'], catIds=catIds)
190   anns = coco.loadAnns(annIds)
191   print(anns)
192   I = io.imread(img_name)
193   for p in anns:
194     #最も適した予測を選択する
195     #距離の近いものを計算
196     a = np.array(p['bbox'][0]+p['bbox'][0]+p['bbox'][2])/2,(p['bbox'
```

```
            ][1]+p['bbox'][1]+p['bbox'][3])/2
197     b1 = ((preds[:,0]+preds[:,2])/2)
198     b2 = ((preds[:,1]+preds[:,3])/2)
199     c=[]
200     c.append(b1)
201     c.append(b2)
202     c=torch.stack(c,dim=1)
203     a2=torch.tensor(a)
204     idx = ((a2-c)**2).sum(axis=1).argmin()
205     #の値を変更する anns
206     data=[]
207     data=preds[idx].tolist()
208     data[2]=data[2]-data[0]
209     data[3]=data[3]-data[1]
210     for i in range(len(data)):
211       data[i]=(int(data[i]))
212     p['bbox']=data
213     ax = plt.gca()
214     xmin, ymin, xmax, ymax = preds[idx].tolist()
215     #c = colors
216     ax.add_patch(plt.Rectangle((xmin, ymin), xmax-xmin, ymax-ymin,
217                                     fill=False, ec='r', linewidth=3))
218   #に書き込む jsonfile
219
220   #回答の表示
221   images.append(img)
222   annotations.append(anns)
223   print('anns')
224   print(anns)
225   print('images')
226   print(img)
227 attrDict['images'] = images
228 flat_list = [item for l in annotations for item in l]
229 attrDict['annotations'] = flat_list
230 attrDict['type'] = 'instances'
231 jsonString = json.dumps(attrDict)
232 print(images)
233 print(annotations)
234 print('instances')
235 print(attrDict['categories'])
236 print(attrDict)
```

```
path = '/content/drive/MyDrive/data/custom3/annotations/'
os.makedirs(path, exist_ok=True)
with open(path+'custom_train.json', 'w') as f:
  f.write(jsonString)

import shutil

shutil.copyfile('/content/drive/My␣Drive/data/custom2/annotations/
    custom_val.json', '/content/drive/My␣Drive/data/custom2/supervision
    /custom_val.json')

num_classes = 5

%cd /content/drive/MyDrive/detr/
os.makedirs('outputs2', exist_ok=True)

# 学習
!python main.py \
    --dataset_file "custom" \
    --coco_path "/content/drive/My␣Drive/data/custom2/" \
    --output_dir "outputs2" \
    --resume "detr-r50_no-class-head.pth" \
    --num_classes $num_classes \
    --epochs 50

from util.plot_utils import plot_logs
from pathlib import Path

%cd /content/drive/MyDrive/detr/
log_directory = [Path('/content/drive/MyDrive/detr/outputs2')]

# 実線... トレーニング結果 (train_loss)
# 破線... 検証結果 (val_loss)
fields_of_interest = (
    'loss',
    'mAP',
)
plot_logs(log_directory, fields_of_interest)

num_classes = 5

```

```
finetuned_model = torch.hub.load('facebookresearch/detr',
                        'detr_resnet50',
                        pretrained=False,
                        num_classes=num_classes)
checkpoint = torch.load('/content/drive/MyDrive/detr/outputs2/
    checkpoint.pth',
                         map_location='cpu')
finetuned_model.load_state_dict(checkpoint['model'], strict=False)
finetuned_model.eval()

original_model = torch.hub.load('facebookresearch/detr', '
    detr_resnet50_dc5', pretrained=True)
original_model.eval()

num_classes = 5

%cd /content/drive/MyDrive/detr/
os.makedirs('outputs3', exist_ok=True)

# 学習
!python main.py \
    --dataset_file "custom" \
    --coco_path "/content/drive/My␣Drive/data/custom3/" \
    --output_dir "outputs3" \
    --resume "detr-r50_no-class-head.pth" \
    --num_classes $num_classes \
    --epochs 50

from util.plot_utils import plot_logs
from pathlib import Path

%cd /content/drive/MyDrive/detr/
log_directory = [Path('/content/drive/MyDrive/detr/outputs3')]

# 実線... トレーニング結果 (train_loss)
# 破線... 検証結果 (val_loss)
fields_of_interest = (
    'loss',
    'mAP',
)
plot_logs(log_directory, fields_of_interest)
```