令和 2 年度茨城大学工学部情報工学科卒業研究論文

Faster R-CNN を用いた one-click supervision

所属　　情報工学科

著者　　平野　友基 (17T4068A)

指導教員　　新納浩幸教授

令和 3 年 2 月 5 日 (金)

令和 2 年度茨城大学工学部情報工学科卒業研究論文

FasterR-CNN を用いた one-clicksupervision

著者

平野　友基 (17T4068A)

指導教員

新納浩幸教授

論文要旨

物体検出を行うタスクでは様々な学習モデルが提案されているが，中でも Faster R-CNN は良い性能を示している．

オブジェクトクラスを検出するには，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．本稿では，detectron2 を用いた Faster R-CNN による one-clicksupervision の提案．これを風船の物体検出を学習するカスタムデータセットを用いて行い，one-click supervision を用いて学習したモデルとカスタムデータセットのみを使用して学習したモデルの精度を比較した．

この結果，モデルの精度はカスタムデータセットを用いた場合には及ばないが，学習の効果は見られた．これに基づき，提案手法では精度の向上は見られるが，通常の方法でアノテーションされたデータに精度を近づけるには改善が必要であると結論付けた．

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 [1] [2]．中でも [3] 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 [4] 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．[4] 内での学習モデルには Fast R-CNN [2] が使用されている．

また，プログラミング言語の Python では物体検出用にいくつかのライブラリが公開されており，本実験では Detectron2 を使用した．

Detectron2 とは，Facebook AI が開発した，PyTorch ベースの物体検出のライブラリである．これは，他のライブラリと比較して少ないコードで Bounding box や Instance Segmentation 等を生成する物体検出を実装することができる．

本研究では，この click supervision を detectron2 を用いて導入した．さらに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 Faster R-CNN 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ものである．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 2.3) 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．

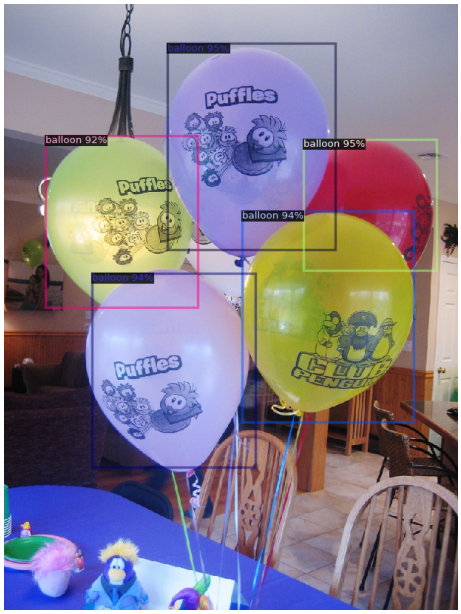


図 2.1: 物体検出のイメージ

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.

物体検出のモデルは大きく R-CNN 系とその他に分けられる.

R-CNN 系では最初に物体候補が提案され，次にこれらの候補を分類/回帰に送るというパイプラインを構築することで物体検出を行っている．しかし，領域提案の処理で時間を要し，リアルタイム検出への応用には速度が課題となっていた.

YOLO,SSD はボックスの領域を提案する処理をなくして，画像に直接ボックスやグリッドをあてはめることでボックスと物体の位置の差分を計算する方法を採用している．これによってネットワークの処理がシンプルになったため，R-CNN 系よりも速度が速い.

本研究では，学習に速度を必要としないため Faster R-CNN を使用した.

2.2　Faster R-CNN

Faster R-CNN は，物体検出の手法の一つである．R-CNN，Fast R-CNN からモデルが改善され，Faster R-CNN が派生した．改善の結果，以前のモデルよりも学習が高速になり，精度も向上したモデルである.

2.2.1　R-CNN の派生

R-CNN から Faster R-CNN の派生までの流れを説明する．

R-CNN

R-CNN はディープラーニングを用いた物体検出の先駆け的な存在である．学習の流れを (図 2.4) に示す．検出までの手順を説明する．

1. 入力された画像の中から，物体が写っている領域の候補（Region Proposal）を最大 2000 個まで抽出する

2.Region Proposal の特徴量を CNN を用いて計算する

3.Region Proposal ごとの分類を行う．

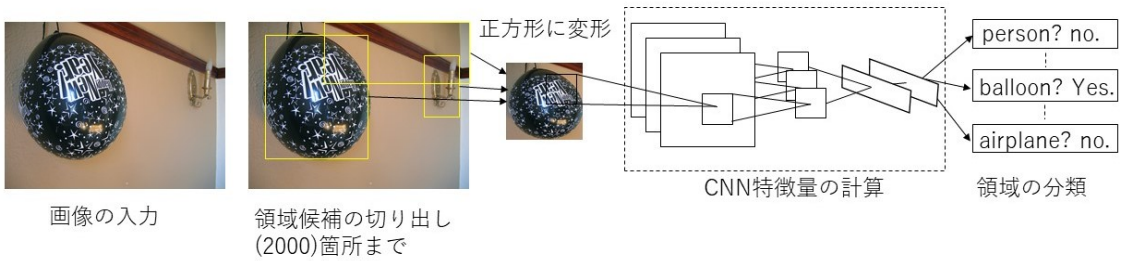


図 2.2: R-CNN の流れ

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，region proposal 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．

Fast R-CNN

Fast R-CNN は R-CNN の高速化を達成したモデルであり，RegionProposal の抽出した特徴領域を全結合層に与えることで，システム全体を一度に学習できるようにした．これによって，毎回 CNN を走らせる必要はなく，RegionProposal 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，画像認識毎に CNN 層も走らせていた従来の 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 2.5) に示す．

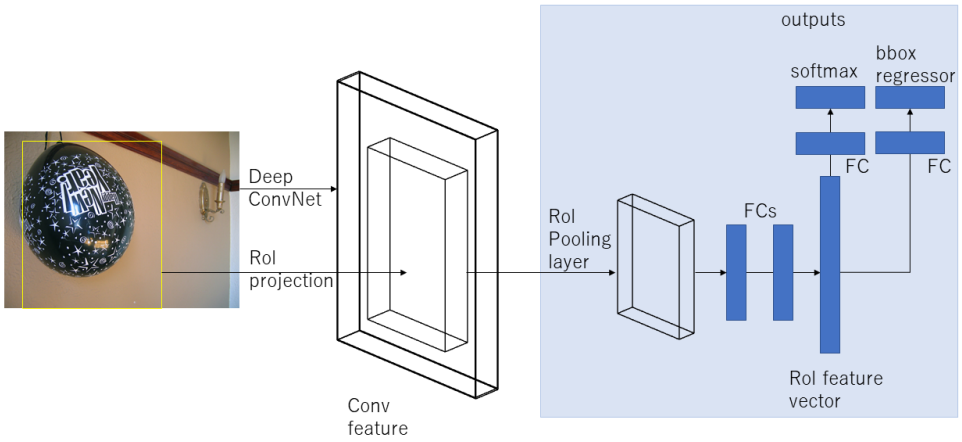


図 2.3: Fast R-CNN の流れ

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．

RoI Pooling

RoI Pooling(図 2.6) 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．

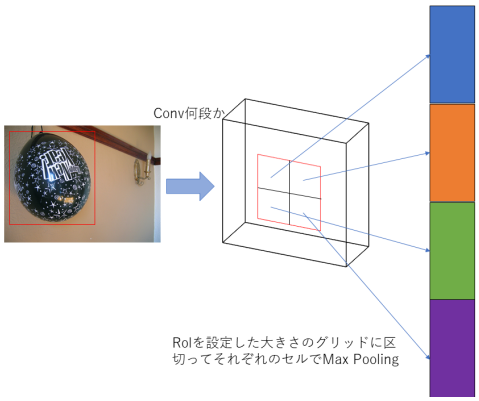


図 2.4: RoI Pooling の流れ

Multi-task Loss

Fast R-CNN で使用した損失関数 Multi-task Loss(図 2.7) 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．

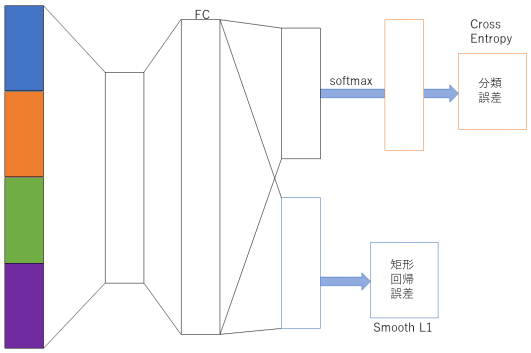


図 2.5: Multi-task Loss の流れ

Faster R-CNN

Fast R-CNN までの Region Proposal 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．

RPN(Region Proposal Network)

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．

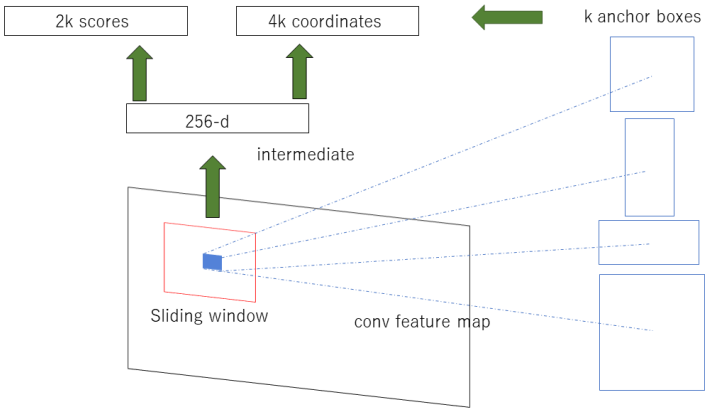


図 2.6: RPN の学習と anchor boxes の生成

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．

1.ImageNet で学習済みの畳み込み層を使って RPN を学習する．

2. 学習済み RPN が出力する矩形を使って畳み込み層のパラメータを更新する．

3. 畳み込み層を fix して RPN を学習し直す．

2.3　click supervision

[4] 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) [6] 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 2.1) 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 2.2) 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．

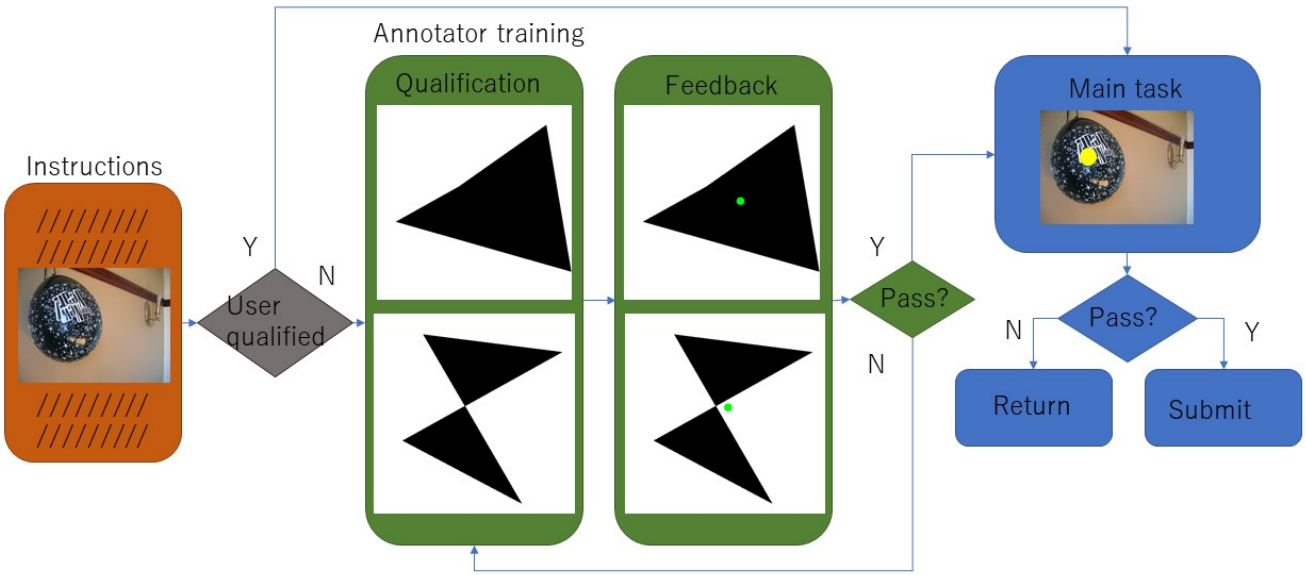
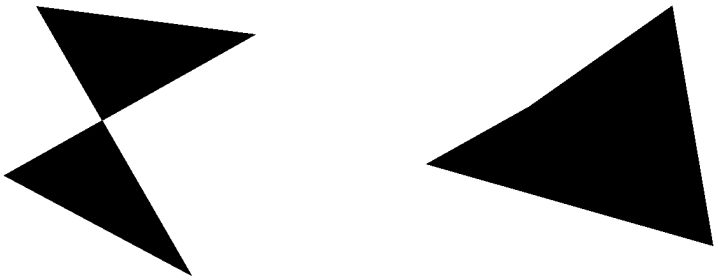


図 2.7: click annotation をクラウドソーシングで行う Workflow



[1]　　[2]

図 2.8: 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 2.1) に示す計算で求める．p は提案された領域である.

$$S_{ap}(p) = \frac{1}{2} \cdot A(p) + \frac{1}{2} \cdot O(p) \tag{2.1}$$

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 2.2) 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.

$$S_{bc}(p; c, \sigma_{bc}) = e^{-\frac{||cp - c||^2}{2\ \sigma_{bc}^2}} \tag{2.2}$$

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 2.3) に示す．

$$S_{ap}(p)\cdot S_{bc}(p; c,\ \sigma_{bc}) \tag{2.3}$$

- 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 Faster R-CNN で行った．

第 3 章

実験

3.1 提案手法

本研究では Detectron を用いて one-click supervision を実装した．また，学習モデルに Faster R-CNN を使用した．また，領域の提案と学習の両方を Faster R-CNN で実装した．

3.2 実験データ

本実験で使用したデータセットは，以下のサイトで公開されているクラスが balloon のみのデータセットである The balloon segmentation dataset を使用した．また，モデルは Detectron2 の Model Zoo で利用できる COCO データセットで学習済みのモデルを使い Balloon segmentation dataset を用いて学習を行った．なお，COCO データセットは全 80 クラスの大規模なデータセットであるが，Balloon のデータは含まれていない．

`https://github.com/matterport/Mask_RCNN/releases/download/v2.1/balloon_dataset.zip`

データの内訳は訓練データ 61 枚，テストデータ 13 枚である．

3.3 実験方法

Detectron で one-click supervision を用いて，COCO データセットで学習済みのモデルを使い Balloon segmentation dataset の転移学習を行う．まず，A.1 に示すプログラムで annotator のトレーニングを行う．

トレーニングの内容は，(図 2.2) に示すような生成された画像の中心をクリックする．この作業を 20 回行い誤差の平均が 20px 以下になったらクリアとする．この時，クリックした点と中心点の誤差の標準偏差を求める．これを one-click supervision のソースコード内の (式 2.1) における σ_{bc} とする．

次に，A.2 に示すプログラムで，表示されたデータセット中の画像の物体の中心をクリックし，クリックデータを収集する．

これをもとに A.3 に示すプログラムで，COCO の学習済みの Faster R-CNN のモデルを click annotation された balloon dataset によるモデルの学習と評価を行う．train 用の画像は balloon segmentation dataset 内の train 用の画像 61 枚にそれぞれクリックデータをもとにしたバウンディングボックスの定位を行い，学習用のデータセットとする．これを 1 セットとし，エポック数は 300 とした．また，このプログラムの実装は Detectron2 として以下のリンクに公開されている Detectron2 Beginner's Tutorial を参考にした．

`https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5#scrollTo=QHnVupBBn9eR`

また，データセットのアノテーションデータをもとに学習を行うプログラム A.4 で学習したモデルの評価を行う．

これらの手順をもとに，one-click supervision の精度の向上も試みた，内容としては，300 エポックごとにクリックデータと学習モデルをもとに再びバウンディングボックスの定位を行う．また，定位を決定する式を (式 4.1) とし，スコアを二乗することでその重要度を高め，モデルの精度を見た．

3.4　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．

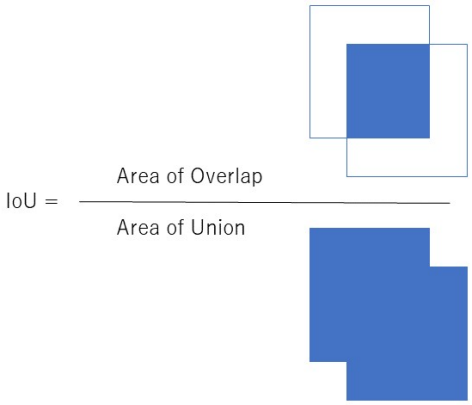


図 3.1: IoU の計算

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．

$Precision = \frac{\text{正しく検出した数}}{\text{全ての検出した数}}$　　$Recall = \frac{\text{正しく検出した数}}{\text{全ての実際のボックス数}}$　　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 3.1) に示す．

この表をもとに P/R 曲線を求める．(図 3.2) にこれを示す．

次に，(図 3.2) 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 3.1) に表すと，

$$\int_0^1 p(r)dr \tag{3.1}$$

となる．(式 3.1) 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 3.2) によって凹凸を均す．結果を (図 3.3) に示す．

$$Pinterp(r) = \max_{\tilde{r}:\tilde{r}\geq r} p(\tilde{r}) \tag{3.2}$$

この (式 3.2) に r=(0,0.1,0.2・・・,1) と 0.1 刻みで 11 個の点を取り，縦×横の掛け算で計算を求める．結果を (図 3.4) に示す．

表 3.1: 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
:	:	:	:	:

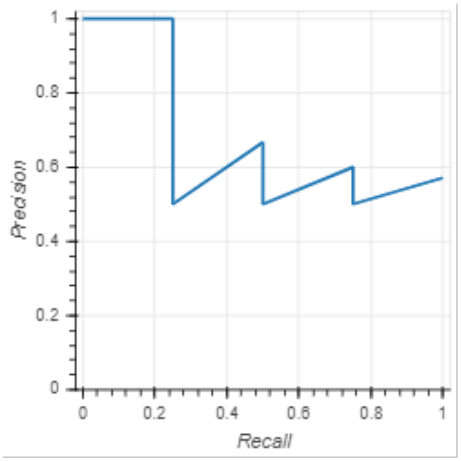


図 3.2: P/R 曲線のグラフイメージ

これが AP(Average Precision) となる，これをクラスごとに算出し，平均を mAP とする．本実験ではクラスが balloon1 つのみであるが，これをもとにモデルの評価を行った．

また，本研究では，COCOEvaluator 関数を用いてモデルの評価を行った．この関数では，複数の条件下での AP の導出結果が導出されている，それらの説明を (表 3.2) に示す．

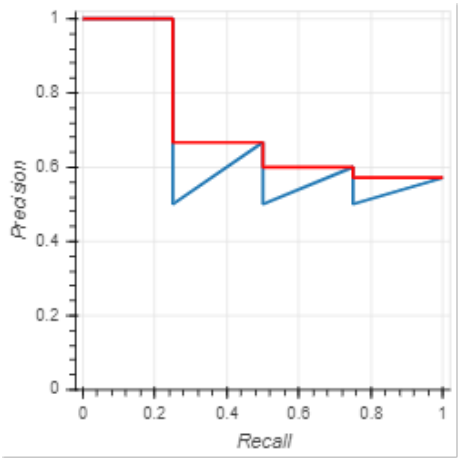


図 3.3: 簡略化された P/R 曲線

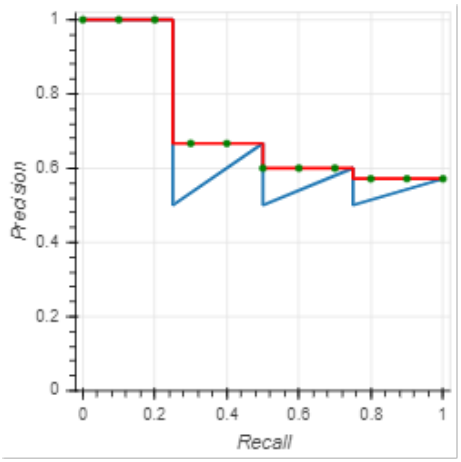


図 3.4: 11 個の点が打たれた P/R 曲線

本実験では，これらを評価の指標として使用した．

表 3.2: 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

3.5　実験結果

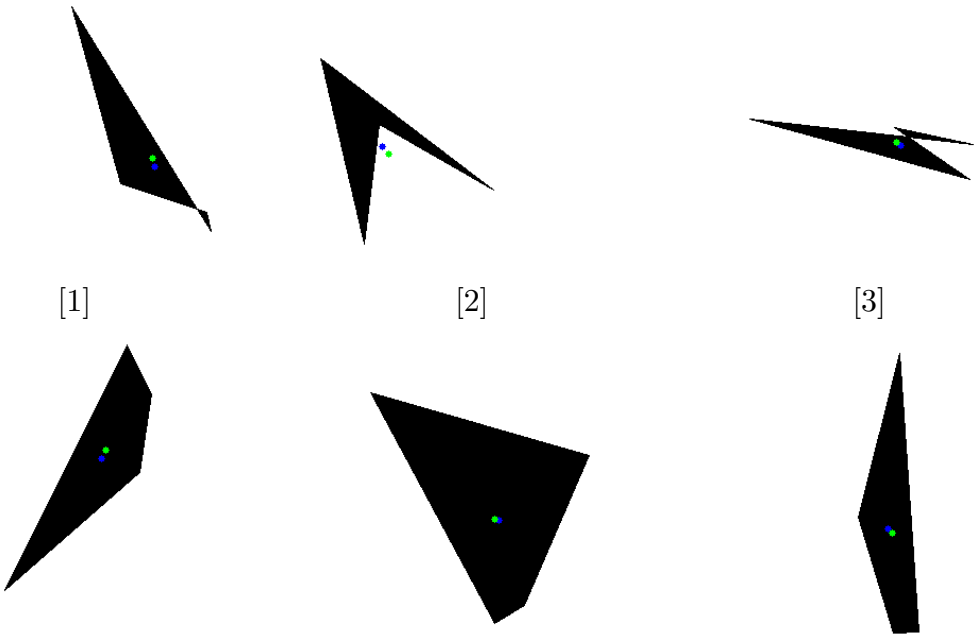
annotator training の実験結果を (図 3.5) と (表 3.3) に示す．画像内の緑色の点が黒い図形の中心点で青い点が実際にクリックした点である．また，(表 3.3) は誤差とその平均値，標準偏差である．

これより，誤差の標準偏差は $\sigma_b c = 12.524101764198502$ となる．これを式本研究では，(式 2,1) 内の $\sigma_b c$ に使用した．今回のアノテーターは私のみなので，balloon データセットの学習に使うコードでは変数を直接書き換えた．

次に実際に使用するデータセットによる one-click supervision を行った．実行結果を (図 3.6) に示す．

青い点がクリックした点で，緑のボックスがクリックデータをもとに生成されたバウンディングボックスを再現したものである．比較的正確にアノテーションデータを作成できた画像もあるが，図 3.6 内の画像，annotated5.png や annotated6.png ではクリックした点から大きく離れた位置にバウンディングボックスが生成された．

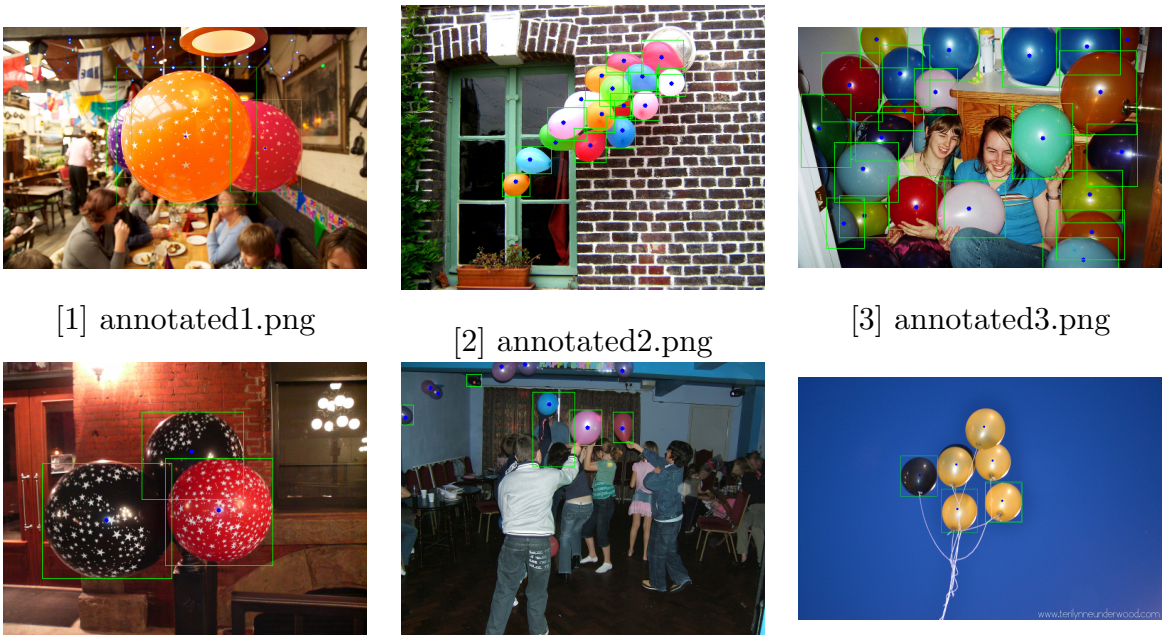
このクリックデータによりアノテーションされた Balloon segmentation dataset　をもとに，COCO データセットにより学習済みの Faster R-CNN の転移学習を行った．結果を (表 3.4) に示す．また，データセット内のアノテーションデータを元に行った学習結果を (表 3.5) に示す．これより，データセット内のアノテーションデータには及ばないが，one-click supervision による学習の成果が見られることが分かった．また，(式 2.3) の $S_{ap}(p)\cdot S_{bc}(p; c, \sigma_{bc})$ を $S_{ap}(p)$ を二乗することでスコアによる値の変化を大きくした



[1] [2] [3]

[4] [5] [6]

図 3.5: annotator training の実行例



[1] annotated1.png [2] annotated2.png [3] annotated3.png

[4] annotated4.png [5] annotated5.png [6] annotated6.png

図 3.6: One-click supervision の実行例

表 3.3: 誤差とその標準偏差

誤差	8.00390529679106	9.95301461869719
	9.905806378079475	5.50567888638631
	4.650268809434569	5.315072906367325
	8.721381771256205	32.31098884280702
	4.123105625617661	5.533985905294664
	6.373774391990981	14.502155012273176
	14.75211849193193	23.85634087616959
	15.988276955319481	8.514693182963201
	12.816005617976296	5.408326913195984
	6.9731628404906765	4.47213595499958
誤差の平均値	10.38400996390212	
誤差の標準偏差	12.524101764198502	

ときの結果を求めた．結果を (表 3.6) に示す．結果に大きな変化は見られなかった．

次に，one-click supervision によって一定回数学習を行った後，その学習済みモデルを元に再びバウンディングボックスの定位を行った．結果を (表 3.7) に示す．すると，物体検出の精度は大幅に低下した．データセット内のアノテーションをもとに学習を行ったがこれも同様に物体検出の精度は大幅に低下した．結果を (表 3.8) に示す．

表 3.4: one-click supervision による学習

イテレーション数	AP AP50 AP75 APs APm APl
300	62.5 79.9 78.9 1.1 46.2 79.0
600	56.5 76.3 72.4 2.8 42.4 73.0
900	69.0 83.3 77.7 5.3 53.7 84.4
1200	71.4 85.4 82.5 5.1 57.5 85.9
1500	69.9 85.4 84.0 6.8 56.3 83.6
1800	72.2 85.4 82.8 4.5 58.1 86.5
2100	71.3 85.5 82.7 4.5 59.7 84.6
2400	72.0 85.2 82.4 5.0 58.3 86.5
2700	71.7 84.9 82.1 5.0 57.0 86.8
3000	70.6 85.5 82.6 12.3 56.4 84.7

表 3.5: データセットによる学習

イテレーション数	AP AP50 AP75 APs APm APl
300	62.1 84.4 73.1 10.6 54.9 73.5
600	77.5 90.6 85.8 17.2 61.7 90.4
900	77.4 91.4 90.6 18.6 63.7 88.6
1200	78.7 91.5 88.1 33.4 64.1 90.5
1500	80.7 92.3 88.3 34.0 65.1 92.5
1800	78.6 91.4 87.3 27.9 62.0 92.1
2100	79.2 91.4 87.2 27.7 63.3 92.6
2400	80.3 91.4 87.1 27.9 64.2 93.4
2700	78.9 91.5 87.0 28.0 64.7 91.6
3000	79.7 91.5 87.1 31.4 67.3 91.6

表 3.6: $S_{ap}(p)$ を二乗したときの学習

イテレーション数	AP AP50 AP75 APs APm APl
300	66.4 83.4 76.8 3.6 52.9 82.5
600	60.7 82.1 70.7 3.5 53.6 73.5
900	69.9 84.2 81.8 4.2 53.9 85.2
1200	70.7 84.7 81.5 3.0 53.4 86.2
1500	70.4 84.3 83.1 3.4 54.8 85.4
1800	68.7 83.7 82.6 2.5 51.9 84.0
2100	69.5 83.5 82.8 2.5 52.9 85.1
2400	71.3 83.7 82.9 3.4 52.8 88.0
2700	71.0 85.3 84.5 14.4 53.2 85.6
3000	69.4 84.3 83.5 14.4 52.0 84.3

表 3.7: one-click supervision による学習済みモデルでバウンディングボックスの再定位

イテレーション数	AP AP50 AP75 APs APm APl
学習済みモデル	70.6 85.5 82.6 12.3 56.4 84.7
300	45.1 68.0 49.6 11.4 55.6 58.0
600	16.1 51.9 1.4 2.2 16.0 21.1
900	15.6 46.9 3.8 0.5 12.4 21.5

表 3.8: データセットによる学習済みモデルでバウンディングボックスの再定位

イテレーション数	AP AP50 AP75 APs APm APl
学習前	80.5 92.4 86.6 19.1 66.3 92.8
300	67.5 88.1 76.9 24.4 55.5 80.5
600	35.2 64.6 33.0 6.0 30.3 43.4
900	30.8 63.1 11.0 1.2 23.0 38.3

第 4 章

考察

one-click supervision によるバウンディングボックスの定位時の (図 3.6) 内の画像 annotated5.png や annotated6.png ではクリックした点から大きく離れた位置にバウンディングボックスが定位された．またこのクリックした点から大きく離れたボックスは他の複数の点をデータとして与えても同じ位置に発生する傾向にあった (図 4.1)．これより，この大きく離れた位置に定位するボックスは物体尤度のスコアの値 (S_{ap}) が大きく,(式 2.3)S_{ap}·S_{bc} において，S_{bc} の値の影響をほとんど受けなかったからだと考えた．ただ，このような確度の高いボックスはある風船のボックスとしては良い結果を示していた．これより，モデルの精度が悪いのではないと考え，スコア関数の見直しを考えた．結果として，S_{ap} を二乗することで，領域候補の尤度の優先度を下げつつ尤度の差によるスコアの開きを大きくする方法を考えた．

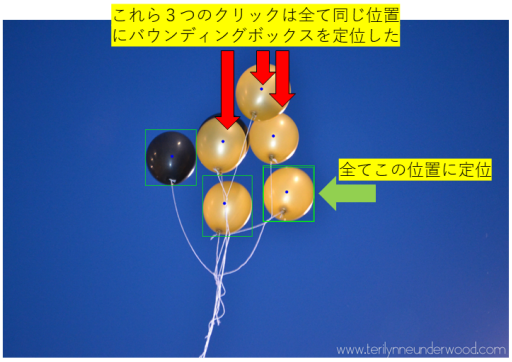


図 4.1: バウンディングボックスが正しく定位されなかった例

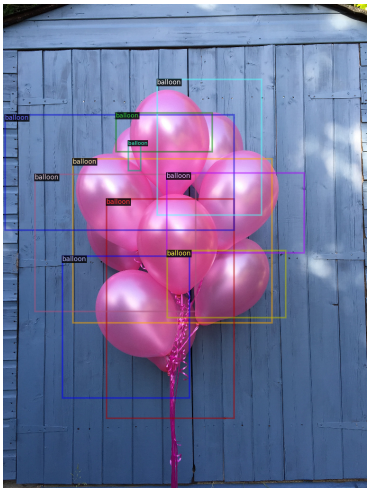
クリックデータをもとにした学習では，one-click supervision を Detectron2 を用いて実装した．結果はデータセットにあった教師データほどではないが学習の成果が見られ

たため，Faster R-CNN による one-click supervision は有効であるといえる．次に，学習の二つの方法でモデルの精度向上を試みた．まず，領域候補のスコアを (式 4.1) とすることで学習の精度の向上を図ったものであるが．結果として，精度に変化は見られなかった．また，$S_{ap}/2$ をスコアとしてみたり，σ_{bc} を 2 倍したりと式の調整を行ったが物体検出の精度に変化は見られなかった．これより，式の変更により one-click supervision の精度向上を試みるには何か新しい指標を取り入れるか別の視点での改善が必要であると考えられる．

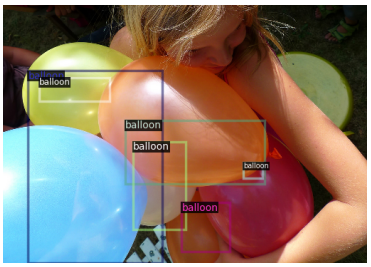
$$S_{ap}(p)^2 \cdot S_{bc}(p; c, \sigma_{bc}) \tag{4.1}$$

次に，balloon dataset を学習させたモデルによって，訓練データのバウンディングボックスの再定位を行ったが，精度が大きく下がる結果となった．学習後のデータの方が正確に物体の位置を検出できるはずなのでクリックデータをもとにした学習によって精度が落ちたと考え，データセット内のアノテーションをもとに学習したものでも同様にバウンディングボックスの再定位を行ってみたが，これも同様に精度が大幅に下がった．クリックデータと学習済みモデルを元にバウンディングボックスを再定位した画像 (図 4.2)，ある画像のクリックした座標と領域候補の中心座標 (表 4.1)，S_{ap} のスコアの例 (表 4.2)，をそれぞれ示す．(表 4.1) より，クリックした座標と近い位置に多くの物体候補の中心が来ていることが分かる．また，(表 4.2) より，どれも値が小さく差も少ないため，(式 2.3)$S_{ap} \cdot S_{bc}$ における S_bc の占める割合が大きくなり，結果としてクリックした点と中心の座標の近い領域が選択されたのだと考えた．また，データセット学習前の画像内で提案された領域のスコアの例も見てみた．これを (表 4.3) に示す．これはスコアの値は (表 4.2) より大きいものが多く，値のばらつきがあった．これより，学習の結果，全体的なスコアが下がり，物体の周辺に多くの領域が提案されるようになり，結果としてバウンディングボックスの再定位がうまくいかなくなったのだと考えた．

[1]



[2]



[3]

図 4.2: 学習済みモデルとクリックデータによる再定位の様子

表 4.1: クリックの座標と学習済みモデルによる領域候補の中心の例

クリックした座標	[764.0, 356.0]
提案された領域の中心	[767.5751953125, 387.76220703125]
	[767.4793090820312, 387.6131896972656]
	[759.986572265625, 394.29254150390625]
	[145.46592712402344, 675.8079223632812]
	[473.4599304199219, 733.7346801757812]
	[677.51025390625, 584.4248046875]
	[171.94485473632812, 264.83245849609375]
	[241.77516174316406, 773.17041015625]
	[149.52032470703125, 793.8538818359375]
	[232.23907470703125, 767.191650390625]
	[682.5278930664062, 831.7732543945312]
	[221.67617797851562, 796.3654174804688]
	[753.8954467773438, 377.502197265625]
	[94.735595703125, 327.2052917480469]

表 4.2: データセット学習後の画像内で提案された領域のスコアの例

1 枚の画像内で提案された領域のスコア
[0.05750284 0.05503758 0.04730576 0.04685047 0.04525393 0.04506718
0.04413376 0.04405423 0.04368179 0.04272439 0.04261538 0.04234602
0.04209882 0.04206597 0.04200458 0.04142088 0.04128059 0.04110219
0.04092795 0.04062471 0.04046457 0.04021546 0.04009205 0.03996186
0.03962237 0.03933227 0.03929323 0.03912786 0.03881278 0.03877638
0.03873719 0.03858089 0.03849057 0.03829236 0.03779565 0.03774681
0.03767374 0.03766021 0.03756976 0.03747676 0.03745938 0.0372408
0.03710535 0.03710351 0.0368304 0.03682677 0.03682232 0.03681206
0.03666718 0.03656283 0.03637851 0.03634549 0.03628229 0.03605225
0.03587141 0.03584847 0.03582688 0.03582363 0.03577318 0.03570791
0.03557035 0.03556325 0.03549081 0.03543304 0.03541135 0.03520732
0.03499231 0.03498841 0.03488104 0.034706 0.03458041 0.03455111
0.03453941 0.03451467 0.03450404 0.03447058 0.03446143 0.03437898
0.03436363 0.03426138 0.0341511 0.03414787 0.03414631 0.03389453
0.03387089 0.03384266 0.03382152 0.03380146 0.03378239 0.03374479
0.03373432 0.03370759 0.03369196 0.03355319 0.03351654 0.03351258
0.03349325 0.03344404 0.03338905 0.03330942]

表 4.3: データセット学習前の画像内で提案された領域のスコアの例

1 枚の画像内で提案された領域のスコア
[0.9981007 0.45543352 0.38916886 0.18551028 0.18215384 0.16502753
0.15673214 0.13743496 0.1246085 0.10877094 0.08154009 0.0788823
0.06184978 0.05918268 0.04802212 0.04542884 0.03795803 0.03744774
0.03272698 0.03271998 0.03184075 0.03049509 0.02968801 0.0279434
0.02646975 0.02531818 0.02471585 0.0205195 0.01973778 0.01936754
0.01935125 0.01887713 0.01789567 0.01764324 0.01702741 0.01646081
0.01608655 0.01605045 0.01589993 0.01581717 0.01379204 0.01375336
0.01359896 0.0130925 0.01218759 0.01164441 0.01148109 0.01134769
0.01117732 0.01044613 0.01014282 0.01003407 0.0096504 0.00953372
0.00913083 0.0090321 0.00881118 0.00869004 0.00858188 0.00815262
0.00812034 0.0080918 0.00784174 0.00762392 0.00759635 0.00717416
0.00712933 0.00709273 0.00708383 0.00651473 0.00625164 0.00616298
0.00592609 0.00582832 0.00558461 0.00556392 0.0054501 0.00543759
0.005338 0.00513707 0.00511498 0.00494604 0.0049135 0.00488702
0.00482075 0.00477711 0.00468038 0.00466953 0.00463351 0.00463218
0.00452793 0.00445344 0.00437055 0.00436128 0.00419178 0.00410537
0.00410011 0.00405773 0.00400262 0.00399828]

第 5 章

結論

本研究では，Detectron2 と Faster R-CNN を用いた one-click supervision による学習を行った．またバウンディングボックスの定位に工夫を施し，精度の向上を試みた．結果としてと Faster R-CNN を用いた one-click supervision による学習の成果は見られたが，精度の向上のために行った工夫では成果が見られなかった．精度の向上のためには，別の視点からのアプローチが必要である．

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支援をいただきました．お礼申し上げます．
ありがとうございました．

参考文献

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, UC Berkeley Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)

[2] Ross Girshick, Microsoft Research Fast R-CNN

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

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

[5] Bogdan Alexe, Thomas Deselaers, Vittorio Ferrari Computer Vision Laboratory, ETH Zurich What is an object ?

[6] Boris Babenko, Ming-Hsuan Yang, Serge Belongie Visual Tracking with Online Multiple Instance Learning

[7] 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).

付録

A Faster R-CNN による one-click supervision を実行するソースコード

このソースコードは公開されている Detectron2 Beginner’s Tutorial を改良して作成した．また言語は全て Python で書かれており，A.1 以外は PyTorch ベースの物体検出のライブラリである Detectron2 を用いて実行される．annotator のトレーニングを行うソースコードを A.1 に，アノテーターによるクリックデータの取得を行うのソースコードを A.2 に，one-click supervision を用いた学習を行うソースコードを A.3 に，データセットを用いた学習を行うソースコードを A.4 示す．

ソースコード A.1: antr.py

```
import os
import numpy as np
import cv2
import random
name_num=1

# 白イメージの生成
r_list=[]
for i in range(20):
    img = np.full((300,300, 3), 255, dtype=np.uint8)
    a = np.random.randint(0,300,size=[4,2])
    #pts = pts.reshape((-1,1,2))
    cv2.fillPoly(img, [a], (0, 0, 0))
    class mouseParam:
        def __init__(self, input_img_name):
            #マウス入力用のパラメータ
            self.mouseEvent = {"x":None, "y":None, "event":None, "
```

```
            flags":None}
18          #マウス入力の設定
19          cv2.setMouseCallback(input_img_name, self.__CallBackFunc,
                None)
20
21      #コールバック関数
22      def __CallBackFunc(self, eventType, x, y, flags, userdata):
23
24          self.mouseEvent["x"] = x
25          self.mouseEvent["y"] = y
26          self.mouseEvent["event"] = eventType
27          self.mouseEvent["flags"] = flags
28
29      #マウス入力用のパラメータを返すための関数
30      def getData(self):
31          return self.mouseEvent
32
33      #マウスイベントを返す関数
34      def getEvent(self):
35          return self.mouseEvent["event"]
36
37      #マウスフラグを返す関数
38      def getFlags(self):
39          return self.mouseEvent["flags"]
40
41      #の座標を返す関数 x
42      def getX(self):
43          return self.mouseEvent["x"]
44
45      #の座標を返す関数 y
46      def getY(self):
47          return self.mouseEvent["y"]
48
49      #との座標を返す関数 xy
50      def getPos(self):
51          return (self.mouseEvent["x"], self.mouseEvent["y"])
52
53
54  if __name__ == "__main__":
55      #表示する名 Window
56      window_name = "input␣window"
```

```
57
58          #画像の表示
59          c=(np.mean(a, axis=0))
60          cv2.imshow(window_name, img)
61          #コールバックの設定
62          mouseData = mouseParam(window_name)
63
64          while 1:
65              cv2.waitKey(20)
66              #左クリックがあったら表示
67              if mouseData.getEvent() == cv2.EVENT_LBUTTONDOWN:
68                  b=[mouseData.getX(),mouseData.getY()]
69                  c=(np.mean(a, axis=0))
70                  u=b-c
71                  result=np.linalg.norm(u)
72                  print("正解との距離\クリックした点 n\中心 n")
73                  print(result)
74                  print(b)
75                  print(c)
76                  r_list.append(result)
77                  #def Sbc(mouseData.getPos(),outputs,)
78                  cv2.destroyAllWindows()
79                  break;
80      cv2.circle(img, (mouseData.getX(),mouseData.getY()), 3, (255, 0,
            0), thickness=-1)
81      cv2.circle(img, (int(c[0]),int(c[1])), 3, (0, 255, 0), thickness
            =-1)
82      cv2.imshow(window_name, img)
83      name = str(name_num)+".png"
84      name_num=name_num+1
85      print(name)
86      cv2.imwrite(name,img)
87      cv2.waitKey(0)
88      cv2.destroyAllWindows()
89
90
91  print("誤差のリスト")
92  print(r_list)
93  print("誤差の平均値")
94  print(np.mean(r_list))
95  score=np.sqrt(np.mean(np.power(r_list,2)))
```

```
96 print("δ bc:")
97 print(score)
98 if(np.mean(r_list)<=20):
99     print("clear")
```

ソースコード A.2: annotation.py

```
1  import os
2  import numpy as np
3  import json
4  import cv2
5  import random
6  import math
7  import torch
8  from detectron2 import model_zoo
9  from detectron2.engine import DefaultTrainer
10 from detectron2.config import get_cfg
11 from detectron2.utils.visualizer import Visualizer
12 from detectron2.structures import BoxMode
13 from detectron2.data import DatasetCatalog, MetadataCatalog
14 from detectron2.engine import DefaultPredictor
15 from detectron2.data import MetadataCatalog
16 import detectron2
17 from detectron2.utils.logger import setup_logger
18 from detectron2.evaluation import COCOEvaluator, inference_on_dataset
19 from detectron2.data import build_detection_test_loader
20
21 setup_logger()
22
23 from detectron2.utils.visualizer import ColorMode
24
25 def get_balloon_dicts(img_dir):
26     json_file = os.path.join(img_dir, "via_region_data.json")
27     cfg = get_cfg()
28     cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/
           faster_rcnn_R_50_FPN_3x.yaml"))
29
30     # （閾値）を設定します．この閾値より確度の高いもののみ出力されます．
           threshold
31     cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.00
32
33     # 今回利用する Faster R-のトレーニング済みファイルを読み込みます．CNN
```

```
34      cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/
            faster_rcnn_R_50_FPN_3x.yaml") # Let training initialize from
            model zoo
35      # 推論を実行します.
36      predictor = DefaultPredictor(cfg)
37
38      with open(json_file) as f:
39          imgs_anns = json.load(f)
40
41      dataset_dicts = []
42      for idx, v in enumerate(imgs_anns.values()):
43          record = {}
44
45          filename = os.path.join(img_dir, v["filename"])
46          height, width = cv2.imread(filename).shape[:2]
47
48          record["file_name"] = filename
49          record["image_id"] = idx
50          record["height"] = height
51          record["width"] = width
52
53          img = cv2.imread(filename)
54          outputs = predictor(img)
55          cv2.destroyAllWindows()
56
57
58          boxes=outputs['instances'].pred_boxes.tensor.cpu().numpy()
59          scores=outputs["instances"].scores.to('cpu').detach().numpy().
                copy()
60
61          objs = []
62          str_filename = img_dir+"/"+v["filename"]
63          objs = annotation(str_filename,filename,boxes,scores)
64          record["annotations"] = objs
65          dataset_dicts.append(record)
66
67      return dataset_dicts
68
69  class mouseParam:
70      def __init__(self, input_img_name):
71          #マウス入力用のパラメータ
```

```
72          self.mouseEvent = {"x":None, "y":None, "event":None, "flags":
                None}
73          #マウス入力の設定
74          cv2.setMouseCallback(input_img_name, self.__CallBackFunc, None)
75
76      #コールバック関数
77      def __CallBackFunc(self, eventType, x, y, flags, userdata):
78
79          self.mouseEvent["x"] = x
80          self.mouseEvent["y"] = y
81          self.mouseEvent["event"] = eventType
82          self.mouseEvent["flags"] = flags
83
84      #マウス入力用のパラメータを返すための関数
85      def getData(self):
86          return self.mouseEvent
87
88      #マウスイベントを返す関数
89      def getEvent(self):
90          return self.mouseEvent["event"]
91
92      #マウスフラグを返す関数
93      def getFlags(self):
94          return self.mouseEvent["flags"]
95
96      #の座標を返す関数 x
97      def getX(self):
98          return self.mouseEvent["x"]
99
100     #の座標を返す関数 y
101     def getY(self):
102         return self.mouseEvent["y"]
103
104     #との座標を返す関数 xy
105     def getPos(self):
106         return (self.mouseEvent["x"], self.mouseEvent["y"])
107
108
109 def annotation(str_filename,filename,boxes,scores):
110     read = cv2.imread(filename)
111     #画像サイズを小さくする
```

```
112     height = read.shape[0]
113     width = read.shape[1]
114     read = cv2.resize(read , (int(width*0.5), int(height*0.5)))
115     objs = []
116     click_data=""
117     #アノテーターは私のみであるため直接を代入 dbc
118     dbc = 12.524101764198502
119     #表示する名 Window
120     window_name = str_filename
121
122     flag=0
123     while flag<1:
124         #画像の表示
125         cv2.imshow(window_name, read)
126
127         #コールバックの設定
128         mouseData = mouseParam(window_name)
129
130         while 1:
131             cv2.waitKey(20)
132             #左クリックがあったら表示
133             if mouseData.getEvent() == cv2.EVENT_LBUTTONDOWN:
134                 cp=[mouseData.getX()*2,mouseData.getY()*2]
135                 click_data=click_data+str(mouseData.getX()*2)+"␣"+str(
                        mouseData.getY()*2)+"␣"
136                 list=[]
137                 for i in range(len(scores)):
138                     c=[(boxes[i][0]+boxes[i][2])/2 ,(boxes[i][1]+
                            boxes[i][3])/2]
139                     u=[cp[0]-c[0],cp[1]-c[1]]
140                     distance=np.linalg.norm(u)
141                     list.append((scores[i]) * (math.exp( -((distance
                            ** 2) / (2*(dbc ** 2)) ) )))
142
143                 l_index = list.index(max(list))
144                 cv2.destroyAllWindows()
145
146                 i2=[]
147                 for num in range(4):
148                         i2.append(int(boxes[l_index][num]/2))
149                 cv2.rectangle(read, (i2[0],i2[1]),(i2[2],i2[3]), (0,
```

```
                    255, 0))
150                 cv2.circle(read, (mouseData.getX(),mouseData.getY()),
                        3, (255, 0, 0), thickness=-1)
151                 obj = {
152                     "bbox": [boxes[l_index][0],boxes[l_index][1],boxes
                            [l_index][2],boxes[l_index][3]],
153                     "bbox_mode": BoxMode.XYXY_ABS,
154                     "category_id": 0,
155                     "iscrowd": 0
156                 }
157                 objs.append(obj)
158                 break;
159             #右クリックがあったら終了
160             elif mouseData.getEvent() == cv2.EVENT_RBUTTONDOWN:
161                 f = open('click_data.txt', 'a')
162                 #print(str_filename)
163                 f.write(str_filename+"\n")
164                 f.write(click_data+"\n")
165                 f.close
166                 flag=1
167                 cv2.destroyAllWindows()
168                 break;
169     return objs
170
171
172 if __name__ == '__main__':
173
174     d="train"
175     get_balloon_dicts("balloon/" + d)
```

ソースコード A.3: train.py

```
1 import os
2 import numpy as np
3 import json
4 import cv2
5 import random
6 import math
7 import torch
8 from detectron2 import model_zoo
9 from detectron2.engine import DefaultTrainer
```

```
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.structures import BoxMode
from detectron2.data import DatasetCatalog, MetadataCatalog
from detectron2.engine import DefaultPredictor
from detectron2.data import MetadataCatalog
import detectron2
from detectron2.utils.logger import setup_logger
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
from detectron2.data import build_detection_test_loader

setup_logger()

from detectron2.utils.visualizer import ColorMode

def get_annos_data(img_dir):
    json_file = os.path.join(img_dir, "via_region_data.json")
    with open(json_file) as f:
        imgs_anns = json.load(f)

    dataset_dicts = []
    for idx, v in enumerate(imgs_anns.values()):
        record = {}

        filename = os.path.join(img_dir, v["filename"])
        height, width = cv2.imread(filename).shape[:2]

        record["file_name"] = filename
        record["image_id"] = idx
        record["height"] = height
        record["width"] = width


        annos = v["regions"]
        objs = []
        for _, anno in annos.items():
            assert not anno["region_attributes"]
            anno = anno["shape_attributes"]
            px = anno["all_points_x"]
            py = anno["all_points_y"]

```

```
51              obj = {
52                  "bbox": [np.min(px), np.min(py), np.max(px), np.max(
                        py)],
53                  "bbox_mode": BoxMode.XYXY_ABS,
54                  "category_id": 0,
55                  "iscrowd": 0
56              }
57              objs.append(obj)
58          record["annotations"] = objs
59          dataset_dicts.append(record)
60      return dataset_dicts
61
62  def annotation_from_file(img_dir):
63      #read = cv2.imread(filename)
64      objs = []
65      #アノテーターは私のみであるため直接を代入 dbc
66      dbc = 12.524101764198502
67
68      cfg = get_cfg()
69      cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/
            faster_rcnn_R_50_FPN_3x.yaml"))
70      #（閾値）を設定します．この閾値より確度の高いもののみ出力されます．
            threshold
71      cfg.DATASETS.TRAIN = (dataset_name,)
72      cfg.DATASETS.TEST = ()
73      cfg.DATALOADER.NUM_WORKERS = 2
74      cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.00
75      # トレーニング済みファイルを読み込みます．
76      cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/
            faster_rcnn_R_50_FPN_3x.yaml")
77      #cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth
            ")
78      # 推論を実行します．
79      predictor = DefaultPredictor(cfg)
80
81
82      dataset_dicts = []
83      line_num=1
84      f = open('click_data.txt','r')
85      line = f.readline()
86      while line:
```

```
87          if line_num%2 == 1:
88              idx=line_num/2
89              filestr = line.strip()
90              record = {}
91
92              filename = os.path.join(filestr)
93              height, width = cv2.imread(filename).shape[:2]
94
95              record["file_name"] = filename
96              record["image_id"] = idx
97              record["height"] = height
98              record["width"] = width
99
100             img = cv2.imread(filename)
101             outputs = predictor(img)
102
103             boxes=outputs['instances'].pred_boxes.tensor.cpu().numpy()
104             scores=outputs["instances"].scores.to('cpu').detach().
                    numpy().copy()
105         else:
106             clicks = [float(i) for i in line.split()]
107             objs=[]
108             for p_num in range(0,len(clicks),2):
109                 point_x = clicks[p_num]
110                 point_y = clicks[p_num + 1]
111                 cp=[point_x,point_y]
112
113                 list=[]
114                 #画像の数ループ
115                 for i in range(len(scores)):
116                     c=[(boxes[i][0]+boxes[i][2])/2 ,(boxes[i][1]+
                            boxes[i][3])/2]
117                     u=[cp[0]-c[0],cp[1]-c[1]]
118                     distance=np.linalg.norm(u)
119                     list.append((scores[i]) / 2 * (math.exp( -((
                            distance ** 2) / (4*(dbc ** 2)) ) )))
120
121
122                 l_index = list.index(max(list))
123                 obj = {
124                         "bbox": [boxes[l_index][0],boxes[l_index][1],
```

```
                                boxes[l_index][2],boxes[l_index][3]],
125                         "bbox_mode": BoxMode.XYXY_ABS,
126                         "category_id": 0,
127                         "iscrowd": 0
128                     }
129                 objs.append(obj)
130             record["annotations"] = objs
131             dataset_dicts.append(record)
132         line_num=line_num+1
133         line = f.readline()
134     f.close
135     return dataset_dicts
136
137
138 def verify_dataset(dists_name):
139     dataset_dicts = annotation_from_file(dicts_name)
140     i=4
141     for d in random.sample(dataset_dicts,3):
142         i=i+1
143         img = cv2.imread(d["file_name"])
144         visualizer = Visualizer(img[:, :, ::-1], metadata=
                balloon_metadata, scale=0.5)
145         vis = visualizer.draw_dataset_dict(d)
146         cv2.imshow(dicts_name, vis.get_image()[:, :, ::-1])
147         #cv2.imwrite(str(i)+'madeBB.png', vis.get_image()[:, :, ::-1])
148         cv2.waitKey(2000) # wait 2000ms
149         cv2.destroyAllWindows()
150
151
152 def train_dataset(dataset_name,i):
153     cfg = get_cfg()
154
155     cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/
            faster_rcnn_R_50_FPN_3x.yaml"))
156
157     cfg.DATASETS.TRAIN = (dataset_name,)
158     cfg.DATASETS.TEST = ()
159     cfg.DATALOADER.NUM_WORKERS = 2
160     #cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-
            InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml") # Let
            training initialize from model zoo
```

```
161     #cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/
            faster_rcnn_R_50_FPN_3x.yaml")
162     if i == 1:
163         cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-
                Detection/faster_rcnn_R_50_FPN_3x.yaml") # Let training
                initialize from model zoo
164     else:
165         cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.
                pth")
166     cfg.SOLVER.IMS_PER_BATCH = 2
167     cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.00
168     cfg.SOLVER.BASE_LR = 0.00025 # pick a good LR
169     cfg.SOLVER.MAX_ITER = 300 # 300 iterations seems good enough for
            this toy dataset; you may need to train longer for a practical
            dataset
170     cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128 # faster, and good
            enough for this toy dataset (default: 512)
171     cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1 # only has one class (ballon)
172
173     os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
174     trainer = DefaultTrainer(cfg)
175     trainer.resume_or_load(resume=False)
176     trainer.train()
177
178     # Evaluation
179     evaluator = COCOEvaluator(val_dataset_name, cfg, False, output_dir
            ="./output/")
180     val_loader = build_detection_test_loader(cfg, val_dataset_name)
181     result = inference_on_dataset(trainer.model, val_loader, evaluator
            )
182     w = str(round(result['bbox']['AP'], 1))+ "␣" + str(round(result['
            bbox']['AP50'], 1)) + "␣" + str(round(result['bbox']['AP75'],
             1)) + "␣"
183     w = w + str(round(result['bbox']['APs'], 1))+ "␣" + str(round(
            result['bbox']['APm'], 1)) + "␣" + str(round(result['bbox']['
            APl'], 1)) + "\n"
184     f = open('balloon_inference.txt', 'a')
185     f.write(w)
186     f.close
187
188 if __name__ == '__main__':
```

```
189     # Prepare the data set
190     dataset_name = "balloon_train"
191     dicts_name = "balloon/train"
192     train_dataset_name = "balloon_train"
193     val_dataset_name = "balloon_val"
194     val_dicts_name = "balloon/val"
195
196     d="train"
197     DatasetCatalog.register("balloon_" + d, lambda d=d:
            annotation_from_file("balloon/" + d))
198     MetadataCatalog.get("balloon_" + d).set(thing_classes=["balloon"])
199     d="val"
200     DatasetCatalog.register("balloon_" + d, lambda d=d: get_annos_data
            ("balloon/" + d))
201     MetadataCatalog.get("balloon_" + d).set(thing_classes=["balloon"])
202     balloon_metadata = MetadataCatalog.get(dataset_name)
203
204     # Verify the data set
205     verify_dataset(dicts_name)
206
207     #train_dataset(dataset_name,1)
208         # train
209     for i in range(1,4):
210         train_dataset(dataset_name,i)
```

ソースコード A.4: test.py

```
1  import os
2  import numpy as np
3  import json
4  import cv2
5  import random
6  from detectron2 import model_zoo
7  from detectron2.engine import DefaultTrainer
8  from detectron2.config import get_cfg
9  from detectron2.utils.visualizer import Visualizer
10 from detectron2.structures import BoxMode
11 from detectron2.data import DatasetCatalog, MetadataCatalog
12 from detectron2.utils.visualizer import ColorMode
13 from detectron2.engine import DefaultPredictor
14 from detectron2.evaluation import COCOEvaluator, inference_on_dataset
```

```
15 from detectron2.data import build_detection_test_loader
16
17 def get_balloon_dicts(img_dir):
18     json_file = os.path.join(img_dir, "via_region_data.json")
19     with open(json_file) as f:
20         imgs_anns = json.load(f)
21
22     dataset_dicts = []
23     for idx, v in enumerate(imgs_anns.values()):
24         record = {}
25
26         filename = os.path.join(img_dir, v["filename"])
27         height, width = cv2.imread(filename).shape[:2]
28
29         record["file_name"] = filename
30         record["image_id"] = idx
31         record["height"] = height
32         record["width"] = width
33
34         annos = v["regions"]
35         objs = []
36         for _, anno in annos.items():
37             assert not anno["region_attributes"]
38             anno = anno["shape_attributes"]
39             px = anno["all_points_x"]
40             py = anno["all_points_y"]
41
42             obj = {
43                 "bbox": [np.min(px), np.min(py), np.max(px), np.max(
                       py)],
44                 "bbox_mode": BoxMode.XYXY_ABS,
45                 "category_id": 0,
46                 "iscrowd": 0
47             }
48             objs.append(obj)
49         record["annotations"] = objs
50         dataset_dicts.append(record)
51     return dataset_dicts
52
53 def verify_dataset(dists_name):
54     dataset_dicts = get_balloon_dicts(dicts_name)
```

```
55      for d in random.sample(dataset_dicts,3):
56          img = cv2.imread(d["file_name"])
57          visualizer = Visualizer(img[:, :, ::-1], metadata=
                balloon_metadata, scale=0.5)
58          vis = visualizer.draw_dataset_dict(d)
59          cv2.imshow(dicts_name, vis.get_image()[:, :, ::-1])
60          cv2.waitKey(2000) # wait 2000ms
61          cv2.destroyAllWindows()
62
63  def train_dataset(dataset_name,i):
64      cfg = get_cfg()
65      cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/
            faster_rcnn_R_50_FPN_3x.yaml"))
66      #cfg.merge_from_file(model_zoo.get_config_file("COCO-
            InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
67      cfg.DATASETS.TRAIN = (dataset_name,)
68      cfg.DATASETS.TEST = ()
69      cfg.DATALOADER.NUM_WORKERS = 2
70      if i == 1:
71          cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-
                Detection/faster_rcnn_R_50_FPN_3x.yaml") # Let training
                initialize from model zoo
72      else:
73          cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.
                pth")
74      #cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-
            InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml") # Let
            training initialize from model zoo
75      cfg.SOLVER.IMS_PER_BATCH = 2
76      cfg.SOLVER.BASE_LR = 0.00025 # pick a good LR
77      cfg.SOLVER.MAX_ITER = 300 # 300 iterations seems good enough for
            this toy dataset; you may need to train longer for a practical
            dataset
78      cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128 # faster, and good
            enough for this toy dataset (default: 512)
79      cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1 # only has one class (ballon)
80
81      os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
82      trainer = DefaultTrainer(cfg)
83      trainer.resume_or_load(resume=False)
84      trainer.train()
```

```
85
86      cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth"
            ) #
87
88      cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7 # set the testing
            threshold for this model
89
90
91      # Evaluation
92      evaluator = COCOEvaluator(val_dataset_name, cfg, False, output_dir
            ="./output/")
93      val_loader = build_detection_test_loader(cfg, val_dataset_name)
94      result = inference_on_dataset(trainer.model, val_loader, evaluator
            )
95      w = str(round(result['bbox']['AP'], 1))+ "␣" + str(round(result['
            bbox']['AP50'], 1)) + "␣" + str(round(result['bbox']['AP75'],
             1)) + "␣"
96      w = w + str(round(result['bbox']['APs'], 1))+ "␣" + str(round(
            result['bbox']['APm'], 1)) + "␣" + str(round(result['bbox']['
            APl'], 1)) + "\n"
97      #f = open('test2_inference.txt', 'a')
98      f = open('balloon_inference.txt', 'a')
99      f.write(w)
100     f.close
101
102 if __name__ == '__main__':
103     # Prepare the data set
104     dataset_name = "balloon_train"
105     dicts_name = "balloon/train"
106     train_dataset_name = "balloon_train"
107     val_dataset_name = "balloon_val"
108     val_dicts_name = "balloon/val"
109
110     for d in ["train", "val"]:
111         print("balloon/" + d)
112         DatasetCatalog.register("balloon_" + d, lambda d=d:
                get_balloon_dicts("balloon/" + d))
113         MetadataCatalog.get("balloon_" + d).set(thing_classes=["
                balloon"])
114     balloon_metadata = MetadataCatalog.get(dataset_name)
115
```

```
116     # Verify the data set
117     #verify_dataset(dicts_name)
118
119     # train
120     for i in range(1,4):
121         train_dataset(dataset_name,i)
```