# ディジタル画像処理　ミニレポート

# 「小銭分類器の作成」

08D20080 谷口令

## 1. 開発経緯

後半の授業で物体検出について教わり、とても興味が湧いて実際に使ってみたくなった。そこで身の回りで物体検出が役に立ちそうな場面を考えてみると、丁度毎回の現金支払いのときに溜まっていく小銭の扱いに困っていたことを思い出した。
小銭を 1 枚ずつ数えていく作業はめんどくさく、物体検出を行えば 1 枚の画像から一気に金額を算出できるのではないかと考え、そのためのプログラムを作成することにした。

## 2. 開発

プログラムの開発にあたってはアノテーションをつけた学習用画像の作成とそれを用いた深層学習モデルの学習の 2 段階に分けられた。
まず学習用画像の作成について記す。
当初は各小銭（1,5,10,50,100,500 円）の写真とアノテーションが施された学習用データが既にあるのではないかと思い調べてみたが、検索する範囲では見つけることができなかった。
そこでゲーム制作用のソフトウェアである Unity を使って学習画像を作成することにした。
まず BOOTH で配布されている小銭の 3D モデルをダウンロードし、3D モデリングソフトの blender で一部加工を施した。（図 1）

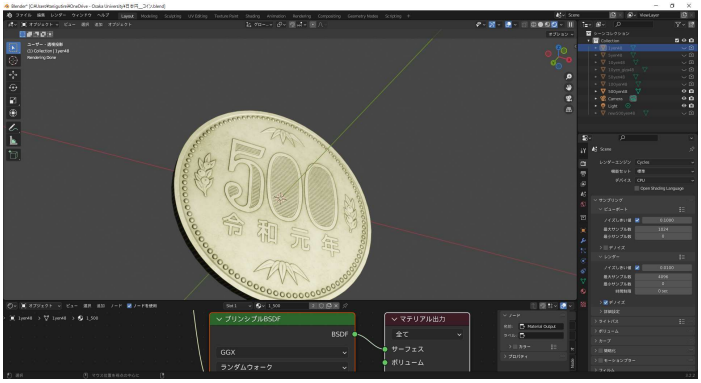


図 1 blender でモデリングしている様子

次に各モデルを Unity 上でプレファブ化し、Perception　Asset を用いてアノテーションを自動でつけるようにした。(図 2)

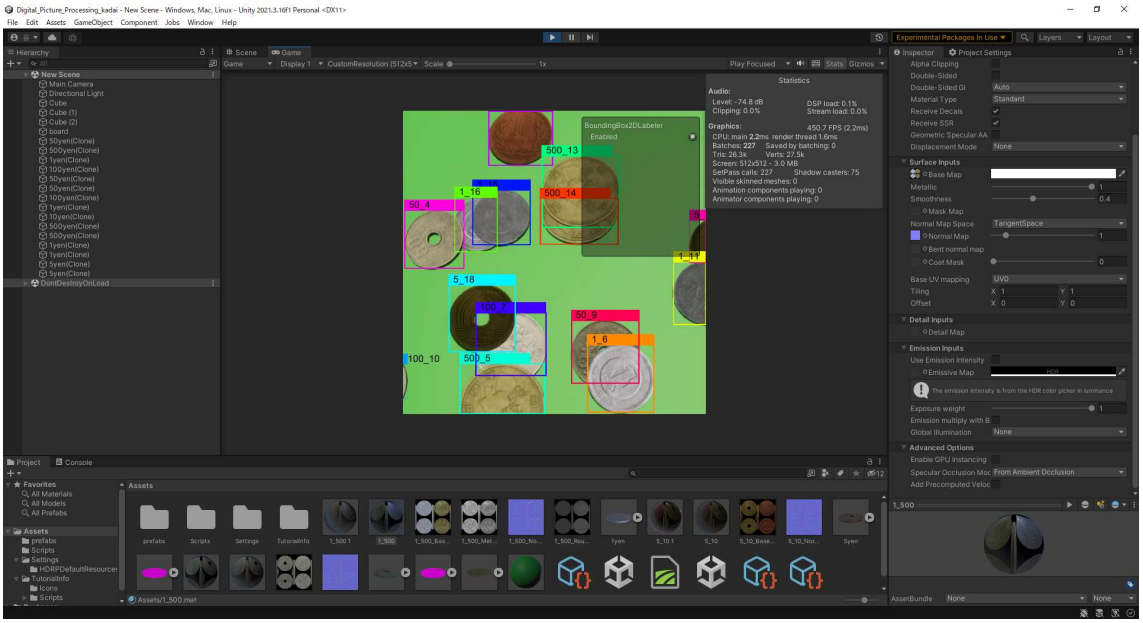


**図 2 Unity 上での学習画像作成を行っている様子**

画像作成手順を以下に示す。

1・カメラをー y 座標を見るように固定する。

2・床（小銭が落ちる場所）を x,z 軸に対して-20~20 度の範囲でランダムに回転させる。（実際の写真では小銭をやや斜めの視点から撮影することが多いため）

4・床の色(RGB＋ アルファ値)をランダムに変える。

3・１０～３０枚の小銭をランダムにインスタンス化する。このとき１～５００円のどの小銭になるかはランダムである。

4・インスタンス化した小銭に Rigidbody と Collision コンポーネントをつけ、重力に従って 2 で作成した床に落とす。

5・Perception を用いてアノテーションをつけ、保存する。(512×512 の大きさ・png で保存)

上記の手順をスクリプトで記述し、for 文を使って 1000 枚の学習用画像とアノテーショを記した json ファイルを作成した。

学習画像を作成する上で工夫した点を以下に示す。

・現実の小銭は汚れていたり傷がついていたりしていることを考慮した。具体的には小銭インスタンス化する際にノーマルマップ・メタリック・スムースネスの値をランダムに変えるようにした。

・実環境で使えるように床の色や角度をランダムに変更した。

・カメラの y 座標をランダムに変更し、学習画像での小銭の大きさが変わるようにした。

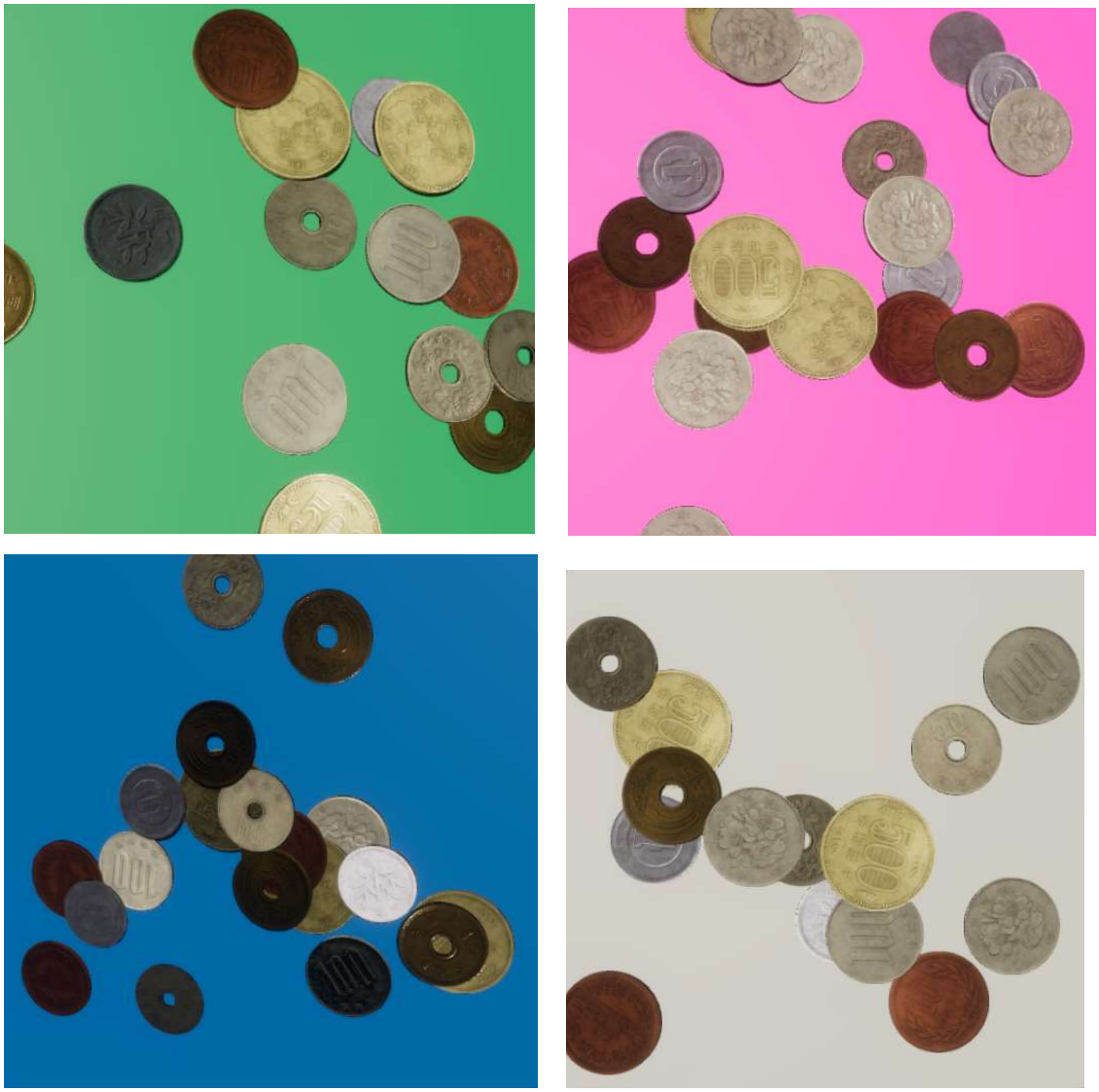

**図 3 学習画像の例**

次に上記で作成した学習画像を用いて、YOLOv3 を用いて転移学習した。
今回使用した YOLOv3 ではアノテーションが txt ファイルである必要があったため json ファイルから変換を行った。
学習においてはミニバッチサイズを 4、イテレーション数を 3000 として RTX3070(VRAM 8GB)用いて学習した。学習時間は約 2 時間であった。
学習における損失の減少を図 4 に示した。

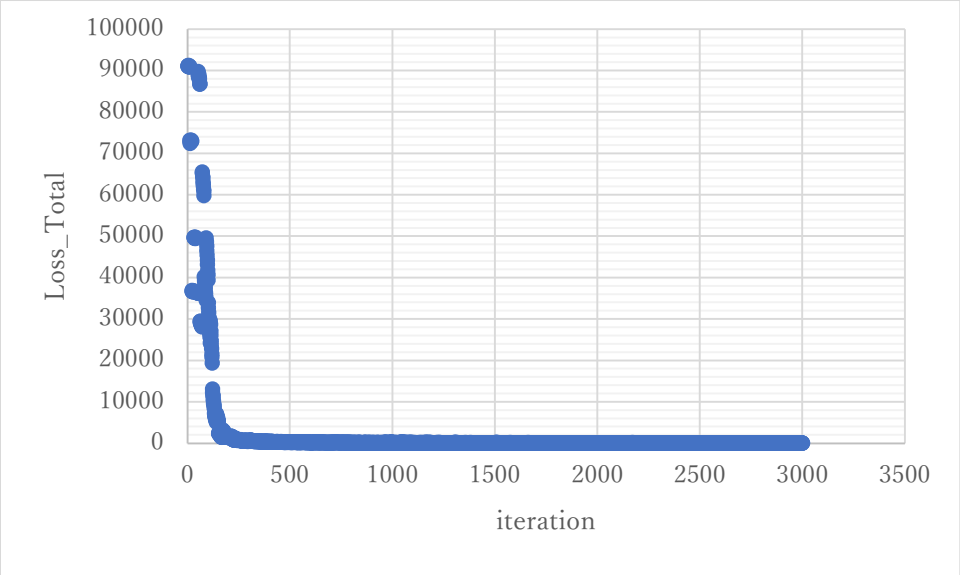


**図 4 学習における損失の減少**

## 3. テスト

まず Unity で作成した画像で、学習では使っていない画像を使ってテストを行った。
数例を試したがほとんどの画像で認識ができていた。
ただ小銭が重なっている画像ではうまく認識ができていなかった。

次にカメラを用いて撮影した画像やネット上で得た画像に学習したモデルを使用してみた。

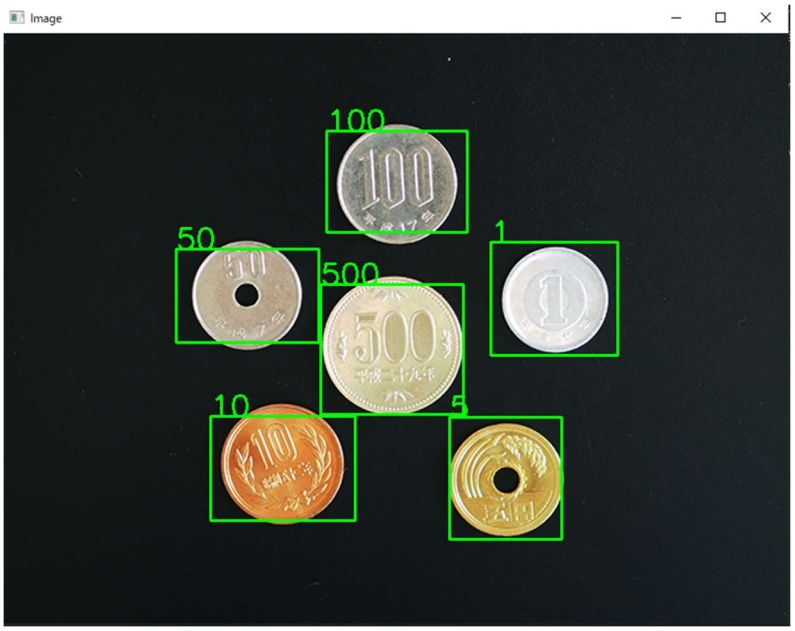


**図 5 実際の画像（ネットから取得）に対してモデルを使用した**

図 5 のような背景がシンプルで各小銭の特徴がよくわかる画像では正しく分類ができることが確認できた。

十分な速さで分類を行えることが確認できたため、次に動画においてリアルタイムでの分類を行えるようにした。具体的には while 文のなかでモデルを使用し、cv2.imshow()を用いて各フレームを出力するようにした。
さらに各金額を合計し、写っている画像に何円分の小銭があるのかを出力するようにした。
比較的高性能な GPU を用いたため 30fps を維持して認識でいることが確認できた。
またこの時 Web カメラの代用として、iVCam を用いてスマートフォンのカメラを使用した。(図 6 )

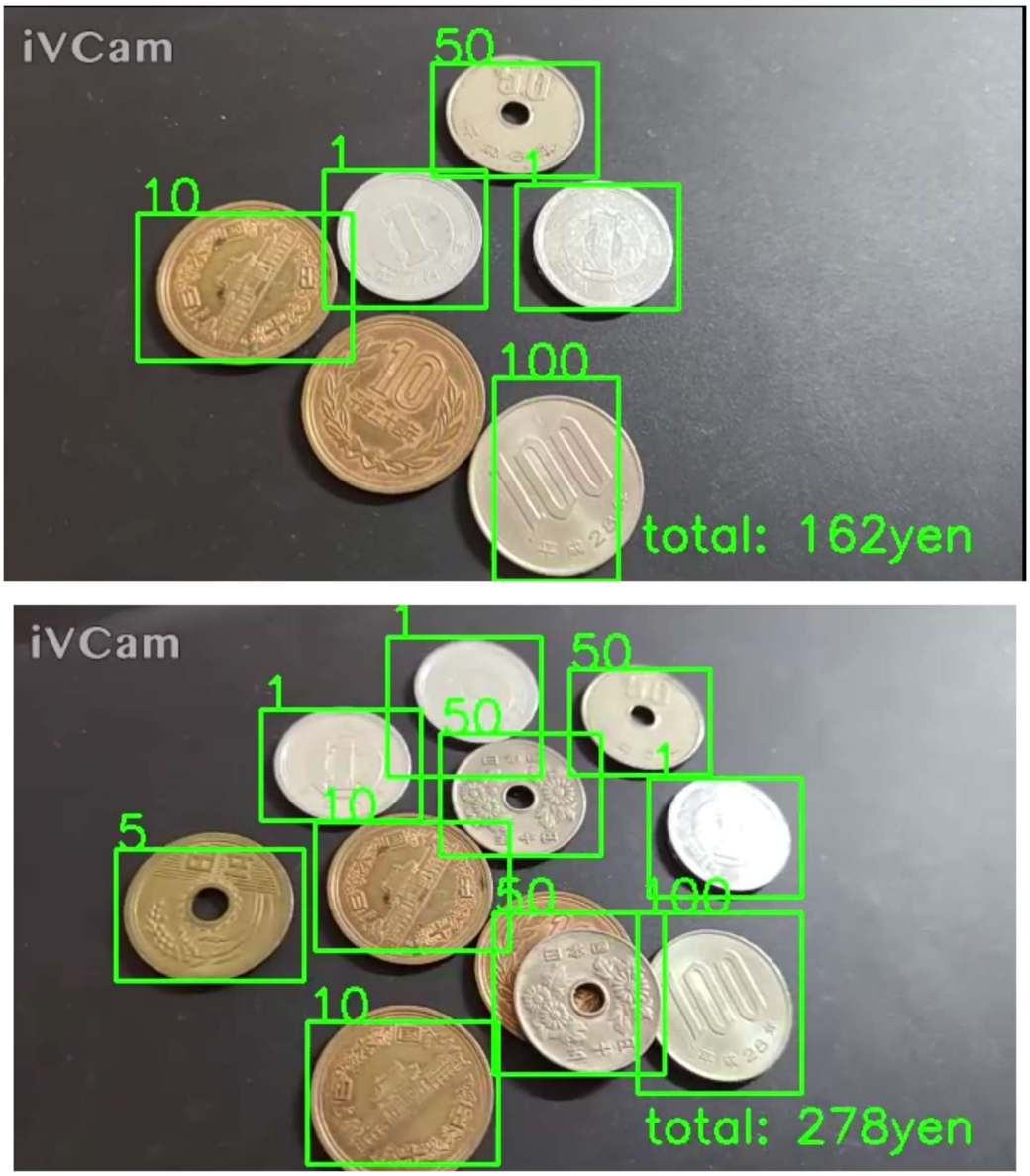


**図 6 モデルをリアルタイムのビデオに適応した例**

ただ重なっている小銭が多くなったり、カメラの角度が変わったりすると認識されなかったり、誤認識されることが多くなった。
これは Unity での学習画像と現実の写真とのズレ・また学習用画像の少なさが原因であると考えられる。
また使用した YOLOv3 のモデルが小さいため、十分な適応が行えなかったことも挙げられる。

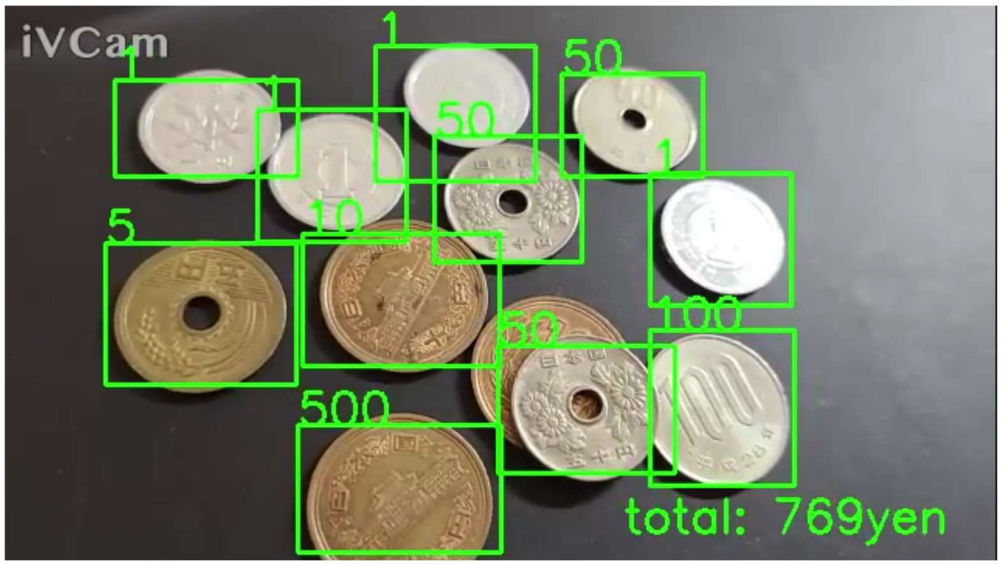


**図 7 多くの小銭があると認識されたなかったり誤って認識されている**

最後に、今回使用したスクリプトを github 上に公開した↓。
https://github.com/TANIGUCHIREI/-Digital_gazousyori_kadai

## 4. あとがき

今回の課題を通して、深層学習モデルの転移学習の手法・また Unity を使った仮想環境での学習用データの作成手順を学べた。
モデルの学習にあたって RTX3070 を使ったのですがそれでも数時間かかった & VRAM 8 GB を全消費したので深層学習はとても学習コストが大きいことが確認できました。
課題とは関係ありませんが、Netflix で AI を利用したアニメーションの PV を見たときに AI 開発者に大倉先生の名前が載っていることを発見して興奮しました。これからも先進的な開発にがんばってください！

## 5. 参考文献

「【ゲーム用素材】日本円硬貨モデルデータ」　BOOTH
https://booth.pm/ja/items/1558400

「YOLOv3 – 自作データセットで学習する方法について」Pystyle
https://pystyle.info/pytorch-yolov3-how-to-train-custom-dataset/

「Unity Perception 入門（2）- ランダム化ツール」　npaka
https://note.com/npaka/n/n7a4f96d44985