2024.7/2 作成

検証環境は、Windows11(64bit)に Python3.10.6 をインストール済、ログインユーザは「if」としてます。
※ユーザ名「if」部分は適宜各環境のユーザ名として読み替えてください。

（編集部検証係）

==========
第 2 章　1 枚の画像を 1 万枚に！前処理＆データ拡張
==========
以下のようにデータ拡張の為の仮想環境「env-da」、「env-ga」を作成します

■「env-da」仮想環境の準備
●コマンドプロンプトを起動し、以下のコマンドで仮想環境「env-da」を作成、有効化します
C:¥Users¥if>python -m venv env-da
C:¥Users¥if>.¥env-da¥Scripts¥activate
(env-da) C:¥Users¥if>

●「env-da」フォルダに移動し、「user」フォルダを作成、pip をアップデートします
(env-da) C:¥Users¥if>cd env-da
(env-da) C:¥Users¥if¥env-da>md user
(env-da) C:¥Users¥if¥env-da>python.exe -m pip install --upgrade pip
(env-da) C:¥Users¥if¥env-da>pip install --upgrade setuptools

● データ拡張関連ツールのインストール
(env-da) C:¥Users¥if¥env-da>pip install optimum[onnxruntime-gpu]
(env-da) C:¥Users¥if¥env-da>pip install -U rembg[gpu,cli]
(env-da) C:¥Users¥if¥env-da>pip install -U albumentations webuiapi

● 前処理用フォルダ作成
(env-da) C:¥Users¥if¥env-da>cd user
(env-da) C:¥Users¥if¥env-da¥user>md sample_img

■「env-ga」仮想環境の準備
●コマンドプロンプトを起動し、以下のコマンドで仮想環境「env-ga」を作成、有効化します
C:¥Users¥if>python -m venv env-ga
C:¥Users¥if>.¥env-ga¥Scripts¥activate
(env-ga) C:¥Users¥if>cd env-ga

● Stable Diffusion 関連ツールのインストール
(env-ga) C:¥Users¥if¥env-ga>git clone https://github.com/AUTOMATIC1111/stable-diffusion-webui.git
(env-ga) C:¥Users¥if¥env-ga>cd stable-diffusion-webui

● Stable Diffusion のバージョンを 1.9.4（検証時の最新版）から 1.7.0 に戻す
(env-ga) C:¥Users¥if¥env-ga¥stable-diffusion-webui>git checkout cf2772f
※最新版では img2img API の呼出時に問題が出た為、検証済みの 1.7.0 にバージョンを戻します

●「webui-user.bat」の以下の箇所をテキストエディタで修正し、保存（1 回目）
※xformers のインストール
(env-ga) C:¥Users¥if¥env-ga¥stable-diffusion-webui>notepad.exe webui-user.bat
※メモ帳の例
変更前：set COMMANDLINE_ARGS=
変更後：set COMMANDLINE_ARGS=--xformers --reinstall-xformers

●「webui-user.bat」を実行する
(env-ga) C:¥Users¥if¥env-ga¥stable-diffusion-webui>webui-user.bat
※WebUI の起動途中で, 初回実行時に xformers が使えるように関連パッケージなどが自動でインストールされます.

●Stable Diffusion WebUI に拡張機能を導入
・Web ブラウザ上の Stable Diffusion の WebUI のメニューの Extensions タブの内の
vailable タブで「Load from:」ボタンを押して一覧の中から「Dynamic Prompts」を探してインストールします。
・ウェブブラウザを閉じて，env-ga を有効化しているコマンド・プロンプトで Ctlr+C を入

力し、終了します。
・再度「webui-user.bat」を実行し、Extentions タブ内の Installed タブに追加した拡張機能が WebUI 上に表示されることを確認する

●「webui-user.bat」の以下の箇所をテキストエディタで修正し、保存（2 回目）
　※API 有効化と WebUI の無効化
(env-ga) C:¥Users¥if¥env-ga¥stable-diffusion-webui>notepad.exe　webui-user.bat
変更前：set COMMANDLINE_ARGS=--xformers --reinstall-xformers
変更後：set COMMANDLINE_ARGS=--xformers --api –nowebui
※「--reinstall-xformers」は不要ですので削除します。
※Web UI を残したい場合は、「–nowebui」削除します。

●API を有効した状態で「webui-user.bat」を実行します
(env-ga) C:¥Users¥if¥env-ga¥stable-diffusion-webui>webui-user.bat

■ここまでで画像拡張をする準備ができたのでデータ拡張を実行します。

●「env-da」環境の「user」フォルダ（C:¥Users¥if¥env-da¥user)必要なファイルを用意します(1 回目・・・OK クラス)
・ダウンロードデータの「setting.ini」を user フォルダ直下に配置します
・ダウンロードデータの以下の Python プログラムを user フォルダ直下に配置します
annotation.py, annotation.py, dataexpansion_2.py, generate_image.py, generate_prompt.py, generate_prompt.py

●OK クラス用に設定や画像を準備
・「setting.ini」ファイルを開き「[ANNOTATION]」セクションのラベル番号「num_label」を「0」とし保存します
・サンプル画像(本誌例では OK クラスのボルト画像 P.75 図 4(a))を user フォルダ内の「sample_img」フォルダ内に配置します

●「data_exp.bat」を実行（1 回目）
(env-da) C:¥Users¥if¥env-da¥user>data_exp.bat
※今回の検証環境では 30〜40 時間程かかりました

●生成されたデータを「0」ラベルフォルダに移動します
・「env-da」環境の「user」フォルダ内に「0」という名称のフォルダを作成します
・以下のフォルダを「0」に移動します

bbox,labels,prepro_output,sd_input,sd_mask,sd_output,tmp_1,tmp2,tmp_mask1,tmp_mask2

●NG クラス用に設定や画像を準備
・「setting.ini」ファイルを開き「[ANNOTATION]」セクションのラベル番号「num_label」を「1」とし保存します
・サンプル画像(本誌例では NG クラスのボルト画像 P.75 図 4(b))を user フォルダ内の「sample_img」フォルダ内に配置します

●「data_exp.bat」を実行（2 回目）
(env-da) C:¥Users¥if¥env-da¥user>data_exp.bat
※今回の検証環境では 30〜40 時間程かかりました

●生成されたデータを「1」ラベルフォルダに移動します
・「env-da」環境の「user」フォルダ内に「1」という名称のフォルダを作成します
・以下のフォルダを「1」に移動します

bbox,labels,prepro_output,sd_input,sd_mask,sd_output,tmp_1,tmp2,tmp_mask1,tmp_mask2

==========
第 3 章　2 万枚の画像を整理！データセット作成＆評価
==========
以下のようにデータセット作成・評価の為の仮想環境「env-ai」を作成します

■「env-ai」仮想環境の準備
●コマンドプロンプトを起動し、以下のコマンドで仮想環境「env-ai」を作成、有効化します

C:¥Users¥if>python -m venv env-ai
C:¥Users¥if>.¥env-ai¥Scripts¥activate
(env-ai) C:¥Users¥if>

(env-ai) C:¥Users¥if>cd env-ai
(env-ai) C:¥Users¥if¥env-ai>python.exe -m pip install --upgrade pip
(env-ai) C:¥Users¥if¥env-ai>pip install --upgrade setuptools
(env-ai) C:¥Users¥if¥env-ai>pip install imagehash
(env-ai) C:¥Users¥if¥env-ai>md user
(env-ai) C:¥Users¥if¥env-ai>pip install ultralytics
(env-ai) C:¥Users¥if¥env-ai>pip uninstall torch torchvision
(env-ai) C:¥Users¥if¥env-ai>pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118

●「env-da」環境の「user」フォルダ（C:¥Users¥if¥env-ai¥user)必要なファイルを用意します
・ダウンロードデータの「setting.ini」、「evaluation.py」を user フォルダ直下に配置します
・ダウンロードデータの「data.yaml」を user フォルダ内の「datasets」フォルダ直下に配置します

●データ作成用のバッチファイル(setup_datasets.bat）のパス名を適宜修正し保存します
C:¥Users¥[ユーザ名]¥...の箇所を、適宜各環境のユーザ名に書替えて保存してくだい

'''
:: ファイルコピー
copy C:¥Users¥if¥env-da¥user¥0¥sd_output¥* datasets¥0¥images
copy C:¥Users¥if¥env-da¥user¥0¥labels¥* datasets¥0¥labels
copy C:¥Users¥if¥env-da¥user¥1¥sd_output¥* datasets¥1¥images
copy C:¥Users¥if¥env-da¥user¥1¥labels¥* datasets¥1¥labels

'''

●データ作成用のバッチファイル(setup_datasets.bat）を実行します
(env-ai) C:¥Users¥if¥env-ai>cd user

(env-ai) C:¥Users¥if¥env-ai¥user>setup_datasets.bat

●「env-ai」環境の「user」フォルダ内の「datawsets」内のデータを確認します
・「images」フォルダに 2 万枚の画像があること
・「labels」フォルダに 2 万個のテキストファイルがあること

●「datawsets」フォルダ内の「images」、「labels」各データを「test」、「train」、「val」フォルダに振分け
・「images」フォルダ内に「test」、「train」、「val」フォルダを作成
・「lables」フォルダ内に「test」、「train」、「val」フォルダを作成
・P.79 図 10 を参考に「test」に 1000 個, 「train」に 18000 個、「val」に 2000 個ずつそれぞれ振分けます

●データセットを評価するため「evaluation.py」を実行
(env-ai) C:¥Users¥if¥env-ai¥user>cd C:¥Users¥if¥env-da¥user
(env-ai) C:¥Users¥if¥env-da¥user>python evaluation.py
※C:¥Users¥if¥env-da¥user フォルダ内に「0」、「1」とフォルダ分けされている状態で実行します
※similaryty.txt が作成されることを確認します

●GPU デバイスが認識できるか確認します
C:¥Users¥if>nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
省略

●YOLOv8 をコマンドライン実行で動作確認します
(env-ai) C:¥Users¥if¥env-ai¥user>yolo predict model=yolov8n.pt source='https://ultralytics.com/images/zidane.jpg' imgsz=320
Downloading
省略

●「users¥runs¥detect¥predict」フォルダ内に検出結果が作成されていること確認します
C:¥Users¥if¥env-ai¥user¥runs¥detect¥predict
内の「zidane.jpg」ファイルに人物が検出されていることが確認できます

==========
第 4 章　YOLOv8 モデルのトレーニング＆物体検出実験
==========

●これまでの生成画像データを元にトレーニングを実行します
(env-ai) C:¥Users¥if¥env-ai¥user>yolo task=detect mode=train data=datasets¥data.yaml model=yolov8m.pt epochs=30 imgsz=640
Downloading
省略
※検証環境では 3〜4 時間程かかりました

●学習済データが作成されていること確認します
「C:¥Users¥if¥env-ai¥user¥runs¥detect¥train¥weights」フォルダに学習済データ「best.pt」、「last.pt」が作成されていることを確認します

■「env-yl」仮想環境の準備
※記事では高性能 GPU 非搭載のノート PC 環境を使用してますが、ここでは GPU 搭載の同一 PC で CPU 版 PyTorch で検証しました

●コマンドプロンプトを起動し、以下のコマンドで仮想環境「env-yl」を作成、有効化します
C:¥Users¥if>python -m venv env-yl
C:¥Users¥if>.¥env-yl¥Scripts¥activate
(env-yl) C:¥Users¥if>

●「env-yl」フォルダに移動し、「user」フォルダを作成、pip をアップデートします
(env-yl) C:¥Users¥if>cd env-yl
(env-yl) C:¥Users¥if¥env-yl>md user
(env-yl) C:¥Users¥if¥env-yl>python.exe -m pip install --upgrade pip
(env-yl) C:¥Users¥if¥env-yl>pip install --upgrade setuptools

● YOLOv8 のインストール
(env-yl) C:¥Users¥if¥env-yl>pip install ultralytics

※ここで CPU 版 PyTorch がインストールされます

●「env-yl」環境の「user」フォルダ（C:¥Users¥if¥env-yl¥user)必要なファイルを用意します
ダウンロードデータの「predict.py」を user フォルダ直下に配置します

● 学習済データと test2 フォルダを作成
(env-yl) C:¥Users¥if¥env-yl>md user¥runs¥detect¥train¥weights
(env-yl) C:¥Users¥if¥env-yl>md user¥test2

● 学習済データ「best.pt」のコピー
「C:¥Users¥if¥env-ai¥user¥runs¥detect¥train¥weights」 フォルダ内の「best.pt」をコピーし
「env-yl」環境の下記フォルダに配置します
「C:¥Users¥if¥env-yl¥user¥runs¥detect¥train¥weights」

●検出対象の画像データを「test2」フォルダに配置
当検証では「env-ai」環境の「D:¥Users¥if¥env-ai¥user¥datasets¥images¥val」から無作為に 50 枚程
「env-yl」環境の「C:¥Users¥if¥env-yl¥user¥test2」フォルダにコピーしました

●物体検出(predict.bat)を実行
(env-yl) C:¥Users¥if¥env-yl>cd user
(env-yl) C:¥Users¥if¥env-yl¥user>predict.bat

●「runs¥detect¥predict」フォルダの検出結果を確認
「C:¥Users¥if¥env-yl¥user¥runs¥detect¥predict¥crops」フォルダ内に
「OK」、「NG」フォルダ仕分けられたクロッピングされた画像、
「C:¥Users¥if¥env-yl¥user¥runs¥detect¥predict¥labelss」フォルダ内に
ファイル毎の検出結果のテキストファイルが作成されている事を確認