ごみ分別
人工知能スマホアプリ作成手順
（概略）

2018/10/23

嶋谷

作業の流れ

赤字は使用するスクリプト（**赤太字は作成したスクリプト**）

- データセット準備 p.3
  - 画像収集
  - 画像水増し Ralpha
  - 画像アノテーション labelImg
    - **Yolo-img-x28_windows.py**
  - labelImg 分類結果一括修正 p.4
    - **replace_classid.sh**
  - データセットのチェック p.5
    - **filecheck.py**
  - 学習用とテスト用画像振分け p.6
    - process.py

- 設定ファイル編集 p.7-8
  - 手作業
- darknet 学習 p.9
  - **darknet-train.sh**

- ログの確認 p.10-12
  - Loss（損失）の収集
  - Loss（損失）のグラフ作成
    - **darknet-log_grep.sh**
    - 手作業
- mAP計測 p.13
  - **darknet-map.sh**
- mAP計測結果編集 p.14
  - **darknet-map_grep.sh**
- mAPグラフ作成 p.15-16
  - 手作業

- darkflow モデル変換 p.17-18
  - **darkflow-flow.sh**

- .weightsファイル削除 p.19
  - **darknet-rm_100weights.sh**
  - スマホ p.20
  - 参考 p.21

前処理

データセットの準備

- ごみの撮影
  - ごみを撮影し、撮影した画像ファイル（*.jpg）をごみの名前が付いたフォルダーに保存する
- リサイズ～水増し
  - Ralphaを使って画像をリサイズ、水増しする
  - Yolo-img-x28_windows.py を使えばこれらの作業は不要
- バウンディング・ボックスの作成（アノテーション）
  - labelImg を使って、アノテーションを行う
- 画像の集約
  - 学習を開始する前に画像データを集約する
  - 学習は、すべての画像をまとめて学習するため、各担当が撮影した画像ファイルとアノテーションファイルを1つのフォルダーに集約する
  - 具体的には、すべてのフォルダーから学習する画像ファイル（画像名.jpg）とアノテーションファイル（画像名.txt）を、~/darknet/data/all フォルダーにコピーする

前処理

labelImg 分類結果一括修正
replace_classid.sh

- Purpose:
  - アノテーション時に分類番号を間違えたとき、生成したテキストファイルの先頭1文字（間違っている分類番号）を正しい分類番号に置き換える
  - 但し、分類番号は、0～9まで
- Usage:
  - $ ./replace_classid.sh <フォルダ名> <ファイル名> <変更後の分類番号>
  - 例
  - $ ./replace_classid.sh cd 2

前処理

データセットのチェック
filecheck.py

- データセットは、画像ファイル（*.jpg）とアノテーションファイル（*.txt）のセットが必要
- 学習中に、片方が無い場合、エラーが出て手戻りになるので、学習前に、filecheck.pyを使って、画像ファイル（*.jpg）とアノテーションファイル（*.txt）のペアの存在をチェックしておく
- filecheck.py を実行する
  - $ python filecheck.py <フォルダ名>

前処理

学習用画像とテスト用画像を振り分ける

- process.py を使って、学習用画像、テスト用画像それぞれのリストを作成する
- process.py ファイルの7行目 path_dataを画像の保存先フォルダー名に変更する
  - （例）
  - path_data = 'data/all/‘
- process.py を実行する
  - $ python process.py
- 結果、test.txtとtrain.txtができる
  - （例）
  - test.txt： 440行、train.txt： 3960行、合計： 4402個

学習

darknet設定ファイルの作成/編集　(1/2)

- ~/darknet/cfg にある流用元フォルダをコピーして新しいフォルダを作成する
  - 例
  - $ cd ~/darknet/cfg
  - $ cp all all2

- *.data を編集する
  - 例：　分類数=10の場合の all2.data
  - classes = 10
  - train = data/all2/train.txt
  - valid = data/all2/test.txt
  - names = cfg/all2/all2.names
  - backup = backup/

学習

darknet設定ファイルの作成/編集　(2/2)

- *.names を編集する
  - 例：　all2.names
  - 分類名を列挙する

- *.cfg を編集する （tiny-yolo の場合）
  - 例：　all2.cfg
  - 125行目：　classes = 10
  - 119行目：　filters= 75
    - filters = (classes + 5) * 5

学習

darknet 学習
darknet-train.sh

- Purpose:
  - darknet(YOLO)が提供している事前学習済み重み係数モデル（darknet19_448.conv.23.weights）*1を使って学習する
- Usage:
  - $ ./darknet-train.sh <事前学習済みモデルファイル名> <作成するモデル名>
  - 例
  - $ ./darknet-train.sh bin/darknet19_448.conv.23 all > all2.log

- ＊１ 学習済みの重み係数モデルを用いて、繰り返し新たな学習を行う転移学習方法が、かば焼きや串カツに使われる秘伝のたれの再利用方法に似ている。ここから、この方法により学習を重ねた重み係数モデルを秘伝のたれと呼び、この方法による転移学習方法を秘伝のたれ学習法と名付けたが、評価の結果、繰り返し学習されるごみが過学習になる懸念があったため、この方法ではなく、darknetが提供するモデルを利用することにした。

学習

Loss（損失）の収集

- 学習中のLossをグラフ化するために、画面出力をログファイルに落とす
- 下記コマンドで、ログをファイルに落とすと画面がスクロールしなくなるので、ターミナルをもう一つ開いて、tail コマンドで表示する
  - $ ./darknet-train.sh bin/darknet19_448.conv.23 all > all2-train.log
  - $ tail –f all2-train.log

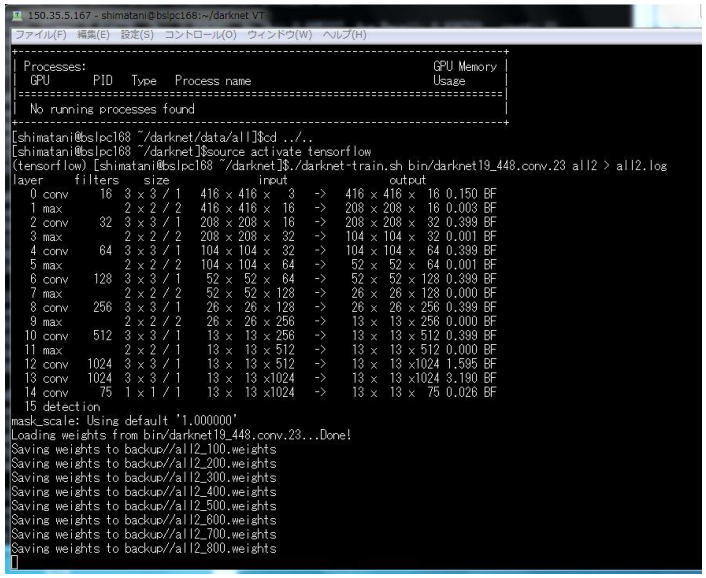

図　学習コマンド実行中画面

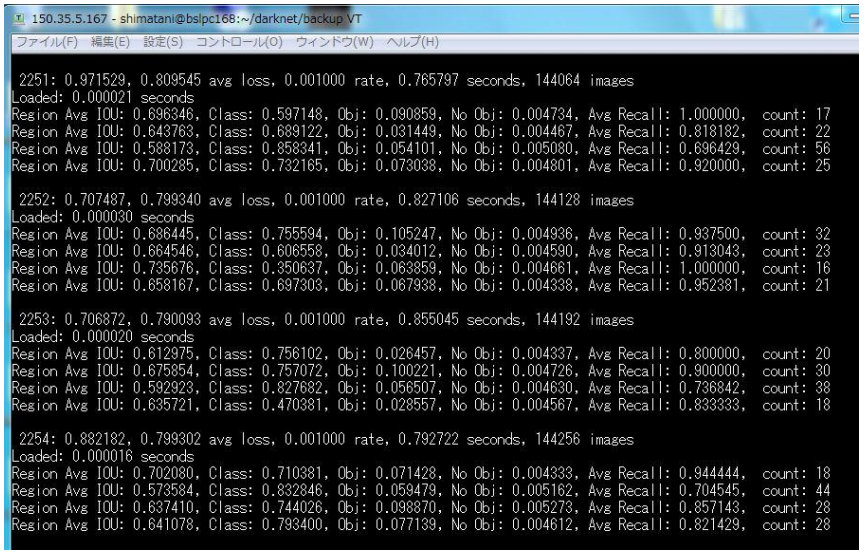

図　ログ収集中画面

モデル検証

Lossの収集～グラフ作成
darknet-log_grep.sh

- Purpose:
  - Logファイルから、loss を含む行を抜き出す
- Usage:
  - $ ./darknet-log_grep.sh <生成された .weighs 保存フォルダ名>
  - 例
  - ./darknet-log_grep.sh all2
- Function:
  - ファイル backup/all2-train.log を開き、all2-train-A.txt を出力する

モデル検証

Lossグラフ作成

- Purpose:
  - loss計測結果をグラフ化する
  - エクセルでグラフ化する
- Process:
  - all2-train-A.txt をエクセルで開く
  - 元データの形式：
    - ◎カンマやタブの区切り文字・・・ を選択
  - 次へ
    - □カンマ をチェック
    - □スペース をチェック
    - □その他 をチェックして、”：” を入力
    - □連続した区切り文字は1文字として扱う をチェック

モデル検証

mAP 計測
darknet-map.sh

- Purpose:
  - .weightsファイルからmAPを計測する
  - 計測結果は、mAP 計測結果編集スクリプト darknet-map_grep.sh を使って見える化する
- Usage:
  - $ ./darknet-map.sh <モデル名>
  - 例
  - $ ./darknet-map.sh pp4

モデル検証

mAP 計測結果編集
darknet-map_grep.sh

- Purpose:
  - mapファイルから必要な行だけを抜き出す
  - darknetが出力するファイルに、CRLFが含まれるので、これをLFに置き換えている
- Usage:
  - $ ./darknet-map_grep.sh <モデル名>
  - 例
  - $ ./darknet-map_grep.sh pp4

モデル検証

mAPグラフ作成　(1/2)

- Purpose:
  - エクセルを使って、mAP計測結果をグラフ化する

- Process:
  - *-map_result-B.txt をエクセルで編集しグラフ化する
  - 元データの形式：
    - ◎カンマやタブの区切り文字・・・　を選択
  - 次へ
    - □カンマ　に、チェック
    - □その他に、= 入力

モデル検証

mAPグラフ作成　(2/2)

- ファイルを開く
  - 例
  - A列をall2_*でフィルタリング
  - B列をコピーして、A列の最終行以下に貼り付け
  - A列をmean_averageでフィルタリング
  - B列をコピーして、B列の最終行以下（A列貼り付けの右）に貼り付け
  - C列をprecisionでフィルタリング
  - D列をコピーして、C列の最終行以下（B列貼り付けの右）に貼り付け
  - F列（recall）をコピーして、D列の最終行以下（C列貼り付けの右）に貼り付け
  - H列（F1-score）をコピーして、E列の最終行以下（D列貼り付けの右）に貼り付け

モデル変換

darkflow モデル変換
darkflow-flow.sh

- Purpose:
  - .weightsファイルを.pbファイルに変換する
- Usage:
  - $ ./darkflow-flow.sh <モデル名>
  - 例
  - $ ./darkflow-flow.sh all2
- Function:
  - darkflowフォルダにプロジェクトフォルダを作成
  - *_final.weights ファイルを、darkflow/binフォルダにコピー
  - ファイル名を、yolov2-tiny-voc.weights に変更
  - darknetの.cfgファイルをdarkflowにコピー
  - darknetの.namesファイルをdarkflowのlabels.txtにコピー
  - .weightsモデルを.pb形式に変換
  - ファイル名を、tiny-yolo-voc-graph.pbに変更

モデル変換

darkflow モデル変換
darkflow-flow.sh

- darkflow/built_graph/all2フォルダを作成し、pbモデルをここに移動

- 手作業
  - WinSPを使って、tiny-yolo-voc-graph.pbモデルを、G:¥Android¥Project¥garbage¥assets¥tiny-yolo-voc-graph.pb にダウンロードする

後処理

.weightsファイル削除
darknet-rm_100weights.sh

- Purpose:
  - ディスク容量削減のために、100回ごとに生成した .weightsファイルを削除する
  - 1000回ごとの .weightsファイルは削除しない
- Usage:
  - $ ./darknet-rm_100weights.sh <モデル名>
  - 例
  - $ ./darknet-rm_100weights.sh pp4

スマホ

Android Studio

- Android Studio
  - 同期
  - ビルド
  - Build Variant をdebugにし、Build～Buils Apk(s)
    - G:¥Android¥Project¥garbage¥gradleBuild¥outputs¥apk¥debug¥garbage-debug.apk ができる
  - Build Variant をreleaseにし、Build～Buils Apk(s)
    - G:¥Android¥Project¥garbage¥gradleBuild¥outputs¥apk¥release¥garbage-release.apk ができる
  - Build Variant をreleaseにし、Build～Generate Signed Apk
    - G:¥Android¥Project¥garbage¥release¥garbage-release.apk ができる

- スマホをPCにつないで、Run

参考

weights ファイル保存間隔の変更方法

- http://demura.net/misc/14592.html
- 3. ネットワークのウェイトを保存する間隔を変更
- ~/src/darknet/examples/detector.cでは、ネットワークのウェイトを保存する間隔は138行目で次のようになっている
- つまり、学習回数が1000回未満のときは100回毎に保存し、それ以降は10,000回毎に保存する
- なお、ウェイトはデータ設定ファイルでしたディレクトリbackupに保存される
- if (i%10000==0 || (i < 1000 && i%100 == 0)) {　// これを10,000回までは1000回毎にも保存したければ次のように変更する
- if (i%10000==0 || (i <= 1000 && i%100 == 0) || (i <=10000 && i % 1000 ==0)) {