

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 labellmg
 - **Yolo-img-x28_windows.py**
 - labellmg 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

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)
 - `labellmg` 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

labellmg 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`

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>

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

- ~/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/`

- *.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
 - $\text{filters} = (\text{classes} + 5) * 5$

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。

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`

```

150.35.5.167 - shimatani@bspc168:~/darknet$ ./darknet-train.sh bin/darknet19_448.conv.23 all > all2-train.log
[shimatani@bspc168:~/darknet]$ source activate tensorflow
[shimatani@bspc168:~/darknet]$ ./darknet-train.sh bin/darknet19_448.conv.23 all > all2-train.log
layer  filters  size  input  output
0 conv  16  3 x 3 / 1  416 x 416 x 3  -> 416 x 416 x 16 0.150 BF
1 max   2  2 x 2 / 2  416 x 416 x 16  -> 208 x 208 x 16 0.003 BF
2 conv  32  3 x 3 / 1  208 x 208 x 16  -> 208 x 208 x 32 0.399 BF
3 max   2  2 x 2 / 2  208 x 208 x 32  -> 104 x 104 x 32 0.001 BF
4 conv  64  3 x 3 / 1  104 x 104 x 32  -> 104 x 104 x 64 0.399 BF
5 max   2  2 x 2 / 2  104 x 104 x 64  -> 52 x 52 x 64 0.001 BF
6 conv  128 3 x 3 / 1  52 x 52 x 64  -> 52 x 52 x 128 0.399 BF
7 max   2  2 x 2 / 2  52 x 52 x 128  -> 26 x 26 x 128 0.000 BF
8 conv  256 3 x 3 / 1  26 x 26 x 128  -> 26 x 26 x 256 0.399 BF
9 max   2  2 x 2 / 2  26 x 26 x 256  -> 13 x 13 x 256 0.000 BF
10 conv 512 3 x 3 / 1  13 x 13 x 256  -> 13 x 13 x 512 0.399 BF
11 max   2  2 x 2 / 1  13 x 13 x 512  -> 13 x 13 x 512 0.000 BF
12 conv 1024 3 x 3 / 1  13 x 13 x 512  -> 13 x 13 x 1024 1.595 BF
13 conv 1024 3 x 3 / 1  13 x 13 x 1024 -> 13 x 13 x 1024 3.190 BF
14 conv 75  1 x 1 / 1  13 x 13 x 1024 -> 13 x 13 x 75 0.026 BF
15 detection
mask_scale: Using default "1.000000"
Loading weights from bin/darknet19_448.conv.23...Done!
Saving weights to backup/all2-100.weights
Saving weights to backup/all2-200.weights
Saving weights to backup/all2-300.weights
Saving weights to backup/all2-400.weights
Saving weights to backup/all2-500.weights
Saving weights to backup/all2-600.weights
Saving weights to backup/all2-700.weights
Saving weights to backup/all2-800.weights

```

図 学習コマンド実行中画面

```

150.35.5.167 - shimatani@bspc168:~/darknet/backup$ tail -f all2-train.log
2251: 0.971529, 0.809545 avg loss, 0.001000 rate, 0.765797 seconds, 144064 images
Loaded: 0.000021 seconds
Region Avg 100: 0.696346, Class: 0.597148, Obj: 0.090859, No Obj: 0.004734, Avg Recall: 1.000000, count: 17
Region Avg 100: 0.643763, Class: 0.689122, Obj: 0.031449, No Obj: 0.004467, Avg Recall: 0.818182, count: 22
Region Avg 100: 0.588173, Class: 0.858341, Obj: 0.054101, No Obj: 0.005080, Avg Recall: 0.696429, count: 56
Region Avg 100: 0.700285, Class: 0.732185, Obj: 0.073038, No Obj: 0.004801, Avg Recall: 0.920000, count: 25

2252: 0.707487, 0.799340 avg loss, 0.001000 rate, 0.827106 seconds, 144128 images
Loaded: 0.000030 seconds
Region Avg 100: 0.686445, Class: 0.755594, Obj: 0.105247, No Obj: 0.004936, Avg Recall: 0.937500, count: 32
Region Avg 100: 0.664546, Class: 0.606558, Obj: 0.034012, No Obj: 0.004580, Avg Recall: 0.913043, count: 23
Region Avg 100: 0.735676, Class: 0.350637, Obj: 0.063859, No Obj: 0.004661, Avg Recall: 1.000000, count: 16
Region Avg 100: 0.658167, Class: 0.697303, Obj: 0.067938, No Obj: 0.004338, Avg Recall: 0.952381, count: 21

2253: 0.706872, 0.780093 avg loss, 0.001000 rate, 0.855045 seconds, 144192 images
Loaded: 0.000020 seconds
Region Avg 100: 0.612975, Class: 0.756102, Obj: 0.026457, No Obj: 0.004337, Avg Recall: 0.800000, count: 20
Region Avg 100: 0.675854, Class: 0.757072, Obj: 0.100221, No Obj: 0.004726, Avg Recall: 0.900000, count: 30
Region Avg 100: 0.592923, Class: 0.827682, Obj: 0.056507, No Obj: 0.004630, Avg Recall: 0.736842, count: 38
Region Avg 100: 0.635721, Class: 0.470381, Obj: 0.028557, No Obj: 0.004567, Avg Recall: 0.833333, count: 18

2254: 0.882182, 0.799302 avg loss, 0.001000 rate, 0.792722 seconds, 144256 images
Loaded: 0.000016 seconds
Region Avg 100: 0.702080, Class: 0.710381, Obj: 0.071428, No Obj: 0.004333, Avg Recall: 0.944444, count: 18
Region Avg 100: 0.573584, Class: 0.832846, Obj: 0.059479, No Obj: 0.005162, Avg Recall: 0.704545, count: 44
Region Avg 100: 0.637410, Class: 0.744026, Obj: 0.098870, No Obj: 0.005273, Avg Recall: 0.857143, count: 28
Region Avg 100: 0.641078, Class: 0.793400, Obj: 0.077139, No Obj: 0.004612, Avg Recall: 0.821429, count: 28

```

図 ログ収集中画面

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 .weights 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

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

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

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`

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

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

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

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

.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

Android Studio

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

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指定したディレクトリ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)) {
```