SSD300 転移学習　理論と実装

* SSDの概要
* 入力と出力

インプットは画像、アウトプットは画像のどこに物体が存在するのかを示す**バウンディングボックス、**各バウンディングボックスが何の物体なのかを示すラベル情報、その検出に対する信頼度の３つ

* 出力はBBOXと書いたが、正確には8732個のDBOXに対するオフセット情報と信頼度
* SSDによる物体検出の流れ

1. 入力画像を300×300にリサイズ(SSD300の場合)
2. デフォルトボックスというバウンディングボックスの元になるBox を8732個用意
3. 画像をSSDのネットワークに入力してDBOXをどのように変換すれば(物体の位置を表す)BBOXになるのかというオフセット情報と各クラスの物体である信頼度を出力
4. 8732個のDBOXの内、信頼度の高い上位からtop\_k個（SSD300では200個）を取り出す
5. オフセット情報を使用し、DBOXをBBOXに変換
6. BBOX同士の重なりが大きい時(同一物体を検出している)は最も信頼度の高いBBOXのみ残す
7. 信頼度の閾値を設定し、閾値以上の信頼度を持つBBOXのみ出力

* 教師データの作成
* 動画を訓練用と検証用に分割

nap1とnap2を混ぜて（nap1とnap2に意味はないので）２割を検証用に利用(全動画データ数は89)

* 動画をフレームに分割

ファイル名は10588044\_01.jpgのようになっている

**Opencvは日本語のディレクトリを読み込めない！**

* 画像データから訓練用・検証用アノテーションを作成

VOTTの出力形式をPascal VOCにすることでxml形式になる

Cutl+Aでregionを全選択できる

* DatasetとDataloaderの作成
* 訓練・検証の画像データとアノテーションデータへのファイルパスリストを作成
* Xml形式のアノテーションデータをリストに変換
* BBOXの座標を規格化(0~1の範囲に変換)(BBOXの情報を(x座標は)画像の幅、(y座標は)高さで割り算)
* 物体クラス名を数値に置き換える
* 画像とアノテーションの前処理を行うクラスDataTransformを作成
* 画像分類と異なり、データオーギュメンテーションで変形させる際にはBBOXの情報も一緒に変形
* 訓練と推論で異なる動作をする
* Datasetの作成
* 訓練用と検証用の2つのデータセットを作る。
* \_\_getitem\_\_()内で画像読み込み、xml形式のアノテーションをリストに、前処理(データオーギュメンテーション)を行い変換した画像のtensorとBBOXとラベルをセットにしたarrayを取得
* DataLoaderの作成
* Datasetから取り出すアノテーションデータである変数gtのサイズが画像データごとに異なる（画像内の物体数が異なる）のでデータ取り出し関数であるcollate\_fnを別途作る必要がある。
* collate\_fnの出力はtorch.Size([batch\_num, 3, 300, 300])の画像データとtorch.Size([n, 5])がbutch\_num個格納されたアノテーションのリストの２つ。
* ネットワークモデルの作成
* モジュールvgg
* VGG-16モデルと同じ
* nn.Conv2dのinput(入力のチャンネル数)とoutput(出力のチャンネル数)の大小関係はどっちでもいい
* モジュールextras
* 活性化関数Reluは順伝播関数の中で用意するのでextras内で実装しなくていい
* モジュールlocとモジュールconf
* num\_class =3とした(safe, caution,背景)
* padding=1としたことで特徴マップのサイズが入力と出力で同じ。
* L2Norm層
* L2Norm層への入力は(512チャンネル×38×38)のテンソルでありこの38×38=1444個のセルに対してセルごとに各チャンネルの特徴量の2乗を計算し512個足し合わせてルートをとる。この値で各チャンネルの各セルを割り算して正規化をする
* 正規化することでチャンネルごとに特徴量の大きさが大きく異なる状況を修正する。
* さらにL2Norm層ではチャンネルごとに係数を掛け算する。この512個の係数は学習させるパラメータ
* デフォルトボックス(DBOX)の実装
* 各特徴マップの中心座標(source1の場合は38×38=1444個の中心座標)とDBOXのwidthとheight(souce1は中心座標一つに対して4個)を持つtorch.Size([8732, 4])のテンソルを作成
* これらの数値は規格化され0~1の範囲に収まっている。
* 順伝播関数の実装
* Decode関数の実装
* DBOXとSSDモデルから求めたオフセット情報を利用しBBOXの座標情報(cx,cy,w,h)を作成する関数
* さらに(cx,cy,w,h)を(xmin,ymin,xmax,ymax)へと変換
* 推論時に実行されるクラスDetect内で使われる。
* Non-Maximum Supressionを行う関数を実装
* 1つの物体に対しては１つのBBoxのみ残す処理
* BBox同士のかぶっている面積が閾値(0.45)以上の時は同じ物体を検出しているとし、確信度が最も大きいBBoxのみ残す。
* Confの高いTop\_k個だけNon-Maximum Supressionで計算
* 推論時に実行されるクラスDetect内で使われる。
* クラスDetectを実装
* 出力は(batch\_num, 3, 200, 5)のテンソル

3は各クラスのインデックスを表す次元

200は信頼度上の200個のBBoxの何番目かを表す次元

5はBBOXの情報(確信度conf,xmin,ymin,width,height)

* 入力の三要素
* オフセット情報を示すモジュールlocの出力(batch\_num,8732,4)
* 確信度を示すconfモジュールの出力(batch\_num, 8732, 3)
* DBOXの情報(8732, 4)
* クラスDetect内の順伝播関数の流れ

1. Decode関数を用いて、オフセット情報locをBBOXに変換
2. Confが閾値以上のBBOXを取り出す
3. Non-Maximum Supressionを行い、同一物体に対して被っているBBOXを消去

以上の3ステップより物体検出結果となるBBOXが残る

* SSDモデルを実装(これまで作ったvggクラスなどを利用して順伝播関数を実装)
* 学習時の出力はout\_put = (loc, conf, dbox\_list)
* オフセット情報を示すモジュールlocの出力(batch\_num,8732,4)
* 確信度を示すconfモジュールの出力(batch\_num, 8732, 3)
* DBOXの情報(8732, 4)
* 推論時はこのoutputを入力として、Detectの順伝播関数に投入し、最終的に検出されたBBoxの情報である(batch\_num, 3, 200, 5)のテンソルを出力
* クラスを実装するにはpytorchのネットワーク層クラスのnn.Moduleを継承
* これまで作った層のオブジェクト(vgg, extras, L2Norm,loc,conf)をクラスSSDのプロパティーにして順伝播関数を実装
* 推論時と訓練時では順伝播関数は異なる動作をする。

(推論時にはDetectクラスを通す)

* 損失関数の実装
* Jaccard係数を用いたmatch関数の動作
* Match関数は8732個のDBoxから学習データの正解BBoxと近いDBox(正解と物体クラスが一致しており、座標情報も近いDBox)を抽出する
* Jaccard係数はIOUのようなもので訓練データの正解Bboxとのjasccard係数が閾値以上のDBoxをPositive DBoxとし、閾値以下のDBoxをNegative Boxとする。
* Positive Boxには教師データとしてjasccard係数が最も大きい正解BBoxのクラスを与える
* DBOXを正解Bboxに変換させるオフセット値をlocの教師データとする
* Negative Boxには教師データとして正解クラス０（背景クラス）を与える
* Match関数の返り値は各DBoxに対するオフセット値と正解ラベル
* Hard Negative Mining
* Negative DBoxに分類されたDBoxのうち、学習に使用するDBoxの数を絞る操作
* DBoxはラベル予測がうまくいっていないもの(きちんと背景クラスと予測されていない)を優先して選ぶ
* 損失関数
* オフセット情報の予測locは回帰問題であり、誤差関数は誤差の二乗関数に少し工夫を加えたSmooth1Loss関数を利用。

予測locについてはPositive DBoxの予測結果のみを使用

* ラベル予測の損失関数は交差エントロピー誤差関数
* SSDの損失関数クラスMultiBoxLossの実装
* nn.Moduleを継承し、損失関数の計算をforward関数内で行う
* オフセットに対する損失値とラベルに対する損失値を返す