RCNNまとめ

Abstract

- PASCOL VOC dataset で測定される物体検出性能はここ数年停滞していた。

- 当時最も優れた方法は低位の画像特徴と高位のコンテキストを組み合わせたアンサンブルシステム

- この論文では過去のVOC 2012と比較して平均精度の平均(mAP)を30%改善し、53.3%のmAPを達成したシンプルでスケーラブルな検出アルゴリズムを提案する

Abstract

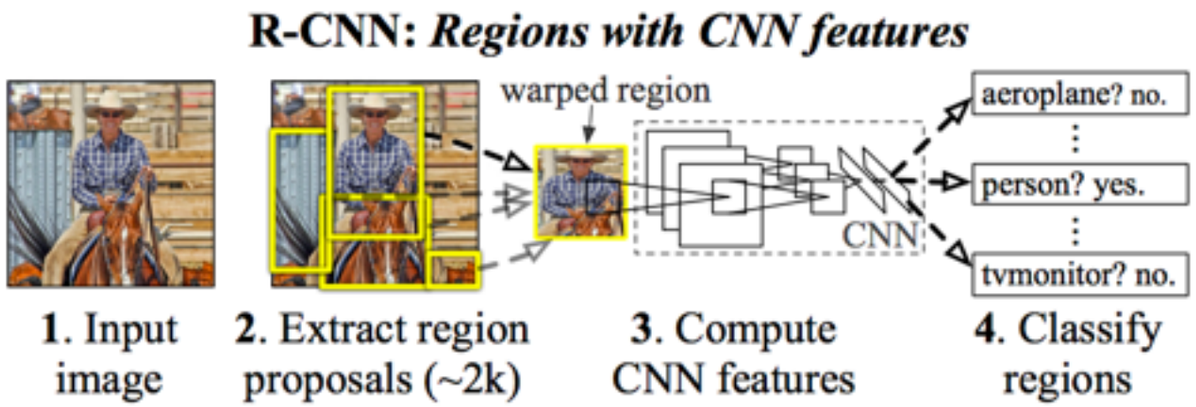
アプローチとしては以下の２つを組み合わせる

- 物体の位置特定とセグメント化を行うためにボトムアップされた候補領域(Region Proposal)を大容量のCNNに適用する
- ドメイン固有のfine-tuningを用いる

領域(Region)とCNNの特徴の組み合わせからこのメソッドを R-CNN と呼ぶ。

Overfeatと呼ばれるスライディングウィンドウ検出器と200クラス分類のILSVRC2013で比較して、RCNNのがOverfeatより大幅に精度を上回った。

検出方法



1. 入力画像から論文で提案されているカテゴリーに依存しない候補領域生成手法(Objectness法、SelectiveSearch法、CMPCなど)を用いて約2000件ほどの候補領域を生成する。

2. 各候補領域をCNNの入力サイズに合うようリサイズし、CNNにかけて特徴を抽出する。

3. 抽出した特徴を用いて、分類するクラスに適した複数のSVMからクラスを識別して物体の位置を特定する。

この論文においては候補生成手法をとしてSelectiveSearchを採用している。

RCNNの設計における必要な３つのモジュール

1. カテゴリーに依存しない候補領域生成器
2. 大容量のCNN(特徴抽出)
3. (CNNで抽出した特徴から)クラス固有の線形SVM

２ においてどのネットワークアーキテクチャを選択するかによってR-CNNの検出性能に大きな影響が出ることがわかっている。

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN T-Net	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
R-CNN T-Net BB	68.1	72.8	56.8	43.0	36.8	66.3	74.2	67.6	34.4	63.5	54.5	61.2	69.1	68.6	58.7	33.4	62.9	51.1	62.5	64.8	58.5
R-CNN O-Net	71.6	73.5	58.1	42.2	39.4	70.7	76.0	74.5	38.7	71.0	56.9	74.5	67.9	69.6	59.3	**35.7**	62.1	64.0	66.5	**71.2**	62.2
R-CNN O-Net BB	**73.4**	**77.0**	**63.4**	**45.4**	**44.6**	**75.1**	**78.1**	**79.8**	**40.5**	**73.7**	**62.2**	**79.4**	**78.1**	**73.1**	**64.2**	35.6	**66.8**	**67.2**	**70.4**	71.1	**66.0**

Krizhevsky氏が提案したネットワークアーキテクチャが O-Net, Simonyan氏とZisserman氏が提案したのが T-Net。(BBは Bounding-Box回帰を用いた場合のもの)
O-NetのR-CNNのがmAPが高く、優れているが計算時間がT-Netと比べて7倍の時間がかかる欠点がある。

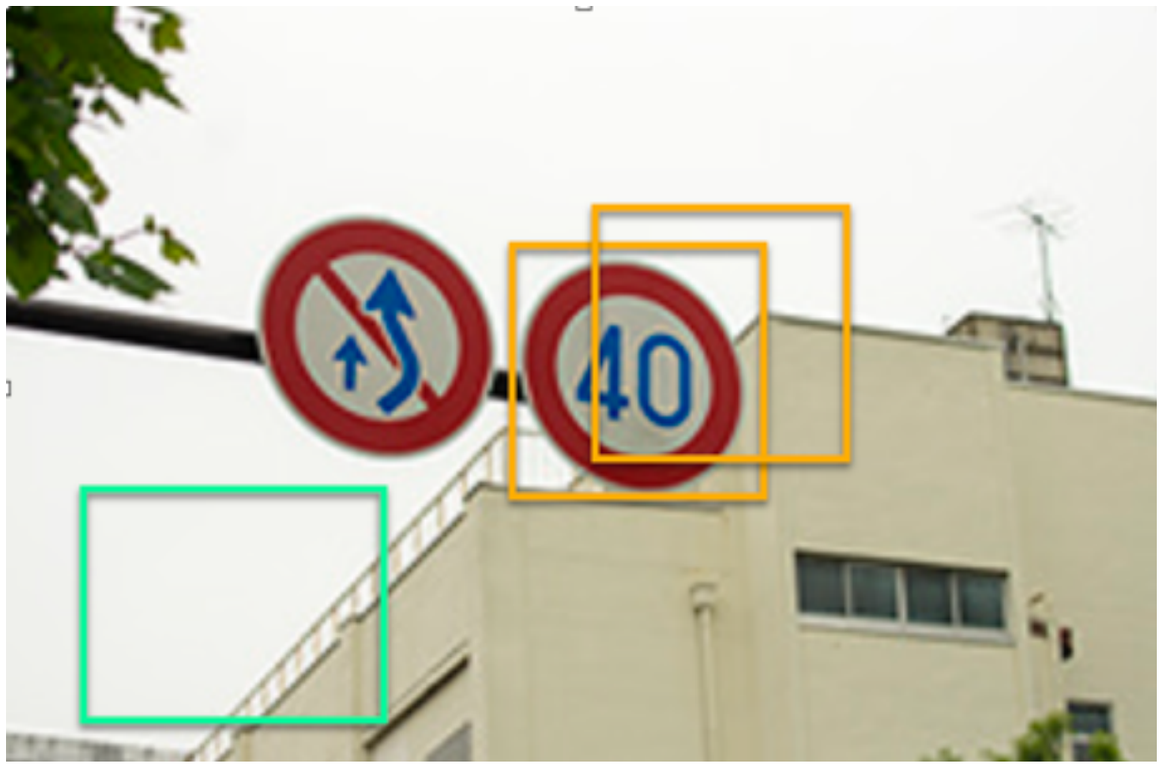
VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN $pool_5$	51.8	60.2	36.4	27.8	23.2	52.8	60.6	49.2	18.3	47.8	44.3	40.8	56.6	58.7	42.4	23.4	46.1	36.7	51.3	55.7	44.2
R-CNN fc_6	59.3	61.8	43.1	34.0	25.1	53.1	60.6	52.8	21.7	47.8	42.7	47.8	52.5	58.5	44.6	25.6	48.3	34.0	53.1	58.0	46.2
R-CNN fc_7	57.6	57.9	38.5	31.8	23.7	51.2	58.9	51.4	20.0	50.5	40.9	46.0	51.6	55.9	43.3	23.3	48.1	35.3	51.0	57.4	44.7
R-CNN FT $pool_5$	58.2	63.3	37.9	27.6	26.1	54.1	66.9	51.4	26.7	55.5	43.4	43.1	57.7	59.0	45.8	28.1	50.8	40.6	53.1	56.4	47.3
R-CNN FT fc_6	63.5	66.0	47.9	37.7	29.9	62.5	70.2	60.2	32.0	57.9	47.0	53.5	60.1	64.2	52.2	31.3	55.0	50.0	57.7	63.0	53.1
R-CNN FT fc_7	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
R-CNN FT fc_7 BB	**68.1**	**72.8**	**56.8**	**43.0**	**36.8**	**66.3**	**74.2**	**67.6**	**34.4**	**63.5**	**54.5**	**61.2**	**69.1**	**68.6**	**58.7**	**33.4**	**62.9**	51.1	**62.5**	**64.8**	**58.5**

Fine-tuningの有無で識別に精度が変わることがわかっている。またILSVRCなどの大規模なデータセットで事前訓練を行い、小規模のデータセット(PASCAL)でドメイン固有のfine-tuningを行うことで大容量のCNNを少ないデータセットで学習することができる。

CNNをILSVRCで事前訓練した２つのR-CNN fc7(fine-tuning 無) と R-CNN FT fc7(fine-tuning有、VOC2007の訓練データでfine-tuning)のmAPを比べると、約10%もの差が出ているのがわかる。

また上の図の 1-3行のmAPの値からfc6(全結合層),fc7(最終層)の値ではなくfc6の前の層pool5の特徴を用いて識別を行ってもかなり良い結果が得られることから、CNNの大部分の表現はDense(全結合) 層よりはConv層から来ているのがわかる。

Non-maximum Suppression

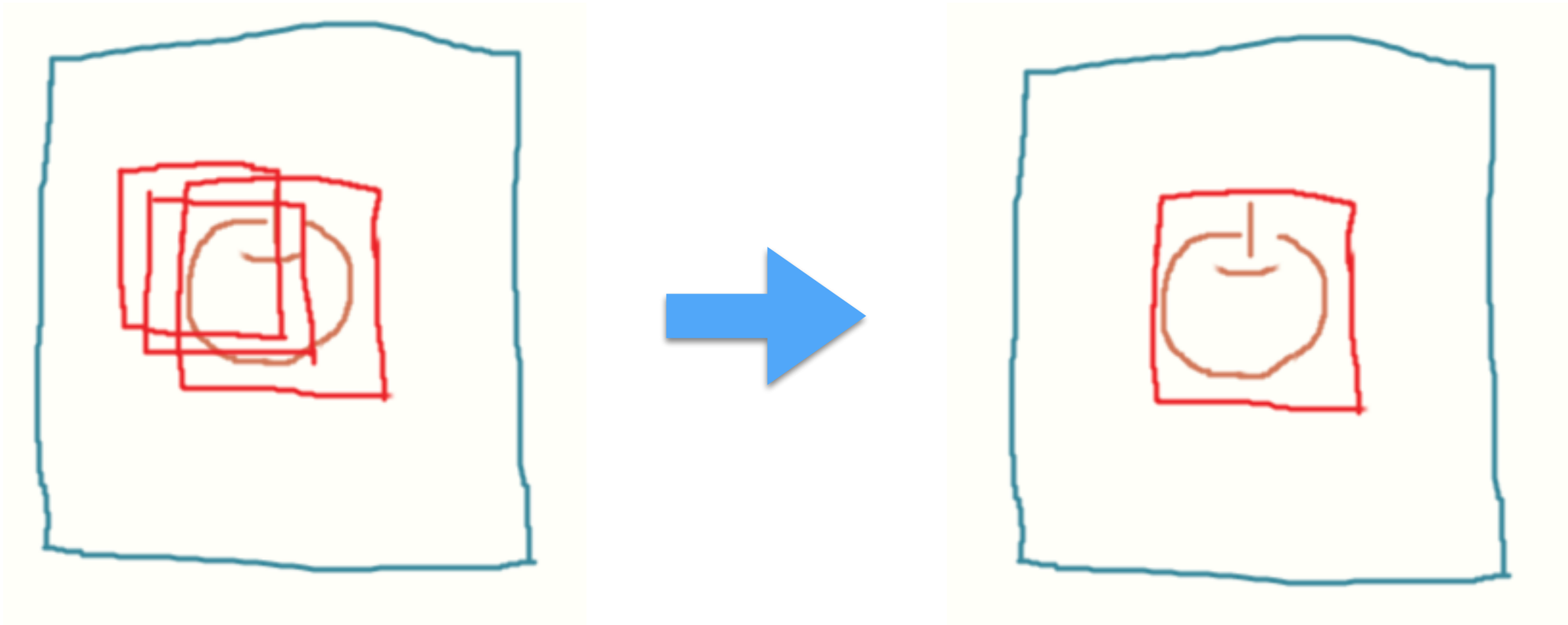


制限速度の標識を検出することを考える。bounding-boxが標識をしっかり囲めばpositiveな例と見なせるが、上の図の緑のように標識と全く関係ない背景の部分(negative)や標識と背景が重なる部分を囲んでしまう結果が出たらどうするか？

この問題には、IoU重複閾値の値を使って解決をする。

- IoU(Intersection-over-Union)とはあるbounding-boxに対して、目的となる領域がどれだけ含まれているかを示す。画像の重なりを示す値。この値が大きければ画像が重なっていることになる。(IoU = 0 -> 全く重なっていない)

- 例としてIoUの閾値を0.3とする。そして右下の領域を基準とする(終点のy座標が一番大きいもの)。そして各領域と右下の領域の重なる部分の面積を求め、その時の領域の面積で割った値が重複の値となる。
(overlap = (重複した面積) / (比べている領域の面積))
ある領域に対するoverlap値が閾値を超えている時(重なりが大きい)はその領域を抑制する。逆に超えていない場合は抑制されない。
この手法を Non-maximum-suppressionと呼ぶ。



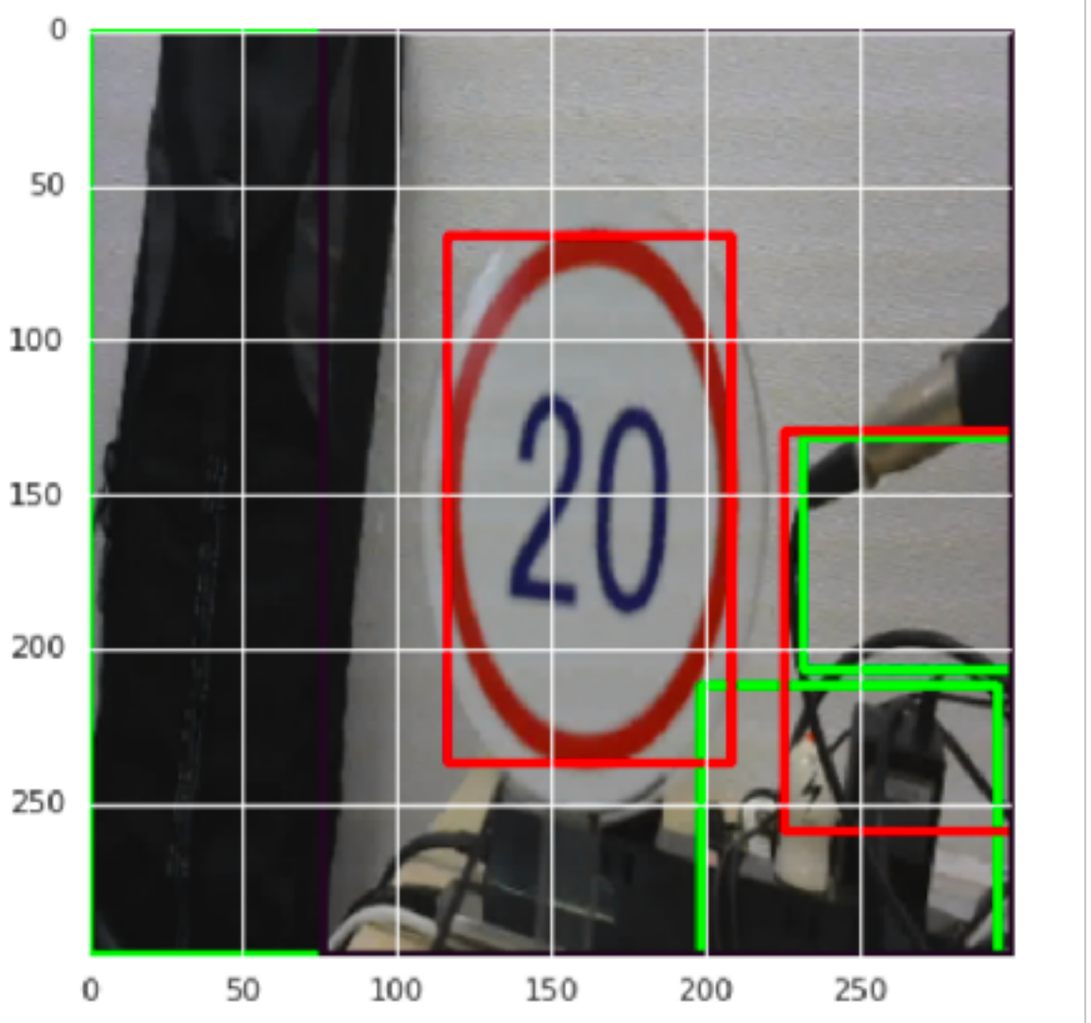
終点のy座標が大きいものを基準として、
overlapの値がIoU閾値0.3以上のものは抑制する。

- この論文のRCNNではIoUの閾値を 0.3 とし、閾値以下の領域をnegativeと定義している。

- 閾値の設定値として{0,0.1,...0.5}まで範囲があるが、このパラメータのを慎重に選択しないとmAPに影響が出ることがわかった。

閾値を0.5に設定するとmAPは5%減少し、逆に0に設定するとmAPは4%減少した。

実行結果



Pythonで実装
学習モデルはTensorflowを使用

Bounding-box regression

- 位置特定の誤差を減らすための方法。これを適用すると誤位置検出を修正し、mAPを3~4%増加することがわかった。

方法として、SelectiveSearchから得た各候補領域をスコア付しクラス固有のSVMにかけた後に新たにクラス固有のbounding-box回帰を用いて検出のための新しいbounding-boxを予測する。

$$\{(P^i, G^i)\}_{i=1,\ldots,N}, \text{where } P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$$

Pは出力、Gは正解の訓練ペア(P,G)の集合があるとする。x,yはboxの中心座標、w,hはboxの幅と高さ。またGのx,y,w,hも同様。

候補boxPを正解boxGに写像するために以下の関数を定義する。

$$d_x(P),\ d_y(P),\ d_w(P),\ \text{and}\ d_h(P).$$

Bounding-box regression

最初の2つdxとdyは領域boxPの中心のスケール不変の変換を指定している。
残りの2つは領域boxPの幅と高さの対数領域での変換を指定している。

これらの関数と以下の変換を適用することで新たに予測した領域が得られる。

$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp(d_w(P))$$
$$\hat{G}_h = P_h \exp(d_h(P)).$$

各d*(* = {x,y,w,h})関数はPool5層の特徴の線形関数としてモデル化され、
ϕ5(P)で表される。従って $d_\star(P) = \mathbf{w}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P)$ (wは学習可能なモデルのパラメータ)
となる。つまりwを最適化することが目的となる。(リッジ回帰で)

$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P^i))^2 + \lambda \left\| \hat{\mathbf{w}}_\star \right\|^2$$

Bounding-box regression

bounding-box回帰を行うにおいて2つ問題がある。

一つ目は正則化が重要であり、wのパラメータ設定においてλ=1000と指定すること。(検証から)

二つ目は(P,G)ペアにおいてPがあまりにもGより離れていてはこの変換法は意味をなさないので、IoUの閾値を0.6に設定してPとGのIoUが閾値以上ならこの変換法を用いること。

テスト時には一回だけbounding-box回帰を適用して新たな検出ウィンドウを予測する。

またbounding-box回帰から得たbounding-box P'で(P',G)として同様に変換を行なってさらに..ということを行うことは可能だが、結果の改善にはならないことがわかっている。

Reference

- https://arxiv.org/pdf/1311.2524.pdf