Loca

出自

ICCV

題目

Loca

Introduction

背景

1. zero-shot : 算出最多那個 class 的數量
2. CAC 的方法主要差異在於 image representation construction method

動機

1. 過去藉由 refference images 製造 object prototypes 的方法是藉由將 refference images 做 max pooling，這樣會沒有辦法有 shape information

   雖然可以用 nonlinear similarity function 來解決
   但是本篇論文提出更簡單的方法

by explicitly addressing the exemplar shape
and by applying an appropriate object prototype adaptation method

目的

1. object prototype extraction module
2. 本研究分離了 query 與 prototype construct

相關研究

CAC：extracts the image and exemplar object features, 然後 concate 再 regression 生成 density map

CFOCNet : concate 會無法 locate the object，因此引用 Siamese correlation network

Ranjan : test-time Siamese backbone adaptation

Shi : jointly learning the representation + nonlinear similarity metric for improved localization +

test image 中的 self-attention 來讓照片中同種物種多樣性可以降低，更好辨認
You : 在 regression 前有對 image feature 使用 similarity map + learnable similarity metric 讓 exemplar 與 image feature 合成得更好

Liu : vision transformer 來抽 image feature. 用 convolutional encoder 來抽 exemplars feature. 並使用 Cross-attention 來合 image and exemplar features. convolutional decoder regresses the density map.

Lin : one-shot CAC. 用 transformer 來做 image and exemplar features 之間的 correlation

Ranjan and Hoai : 提出 RepRPN-Counter 來做 zero shot CAC. 先使用 region proposal network 並predicts 每個 region 的重複分數，重複分數越高的就是 refference image (exemplars). 再接上 FamNet 生 density map

Hobley and Prisacariu : zero-shot learning. 讓 model 找出最應該被 count 的物體

方法

自製 refference filter (利用原本的整張圖)，不用 max pooling 的方法

【重點】 Loca 的 Refference，只能從圖上取出，不能是一個額外輸入

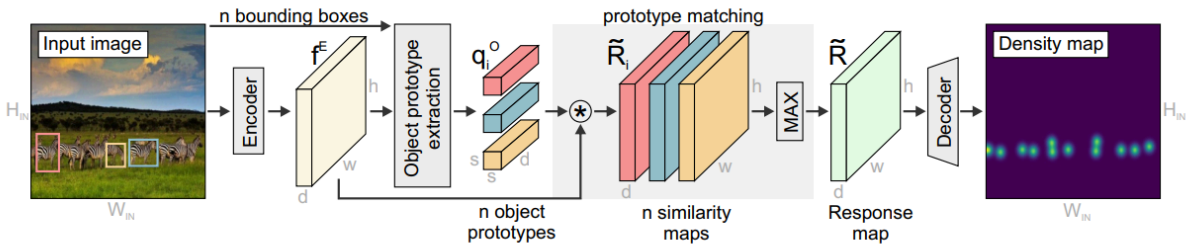


Figure 2. The LOCA architecture. Input image is encoded into features $\mathbf{f}^E$, which are depth-wise correlated (*) by n object queries predicted by the object prototype extraction module. The response map $\tilde{\mathbf{R}}$ is obtained by computing per-element maximum of n similarity maps $\tilde{\mathbf{R}}_i$ and then upsampled by decoder to the final density map.

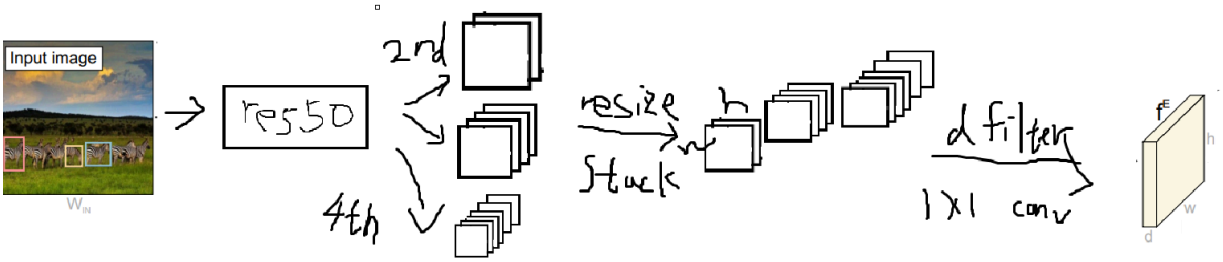
n 個 box 包住 refference image

$$\text{input image } \mathbf{I} \in \mathbb{R}^{H_0 \times W_0 \times 3}$$

$$\text{density map } \boldsymbol{R} \in \mathbb{R}^{H_0 \times W_0}$$

The input image is resized to $H_{IN} \times W_{IN}$

1. input image 先放入 resnet 2, 3, 4 層，可以將 fE 視為特徵圖



2. 從 fE 抽出 refference ，因此可以考慮 objects shape and appearance properties

$$n \text{ object prototypes } \{\mathbf{q}_i^O \in \mathbb{R}^{s \times s \times d}\}_{i=1:n}$$

prototypes 應該包含各種 refference 的樣子

Shape information :

initializing the prototypes with exemplar width and height features

appearance query

fE 中的對應 box 區做用 ROI Pooling 得到 qi

$$\text{queries } \mathbf{q}_i^A \in \mathbb{R}^{s \times s \times d}$$

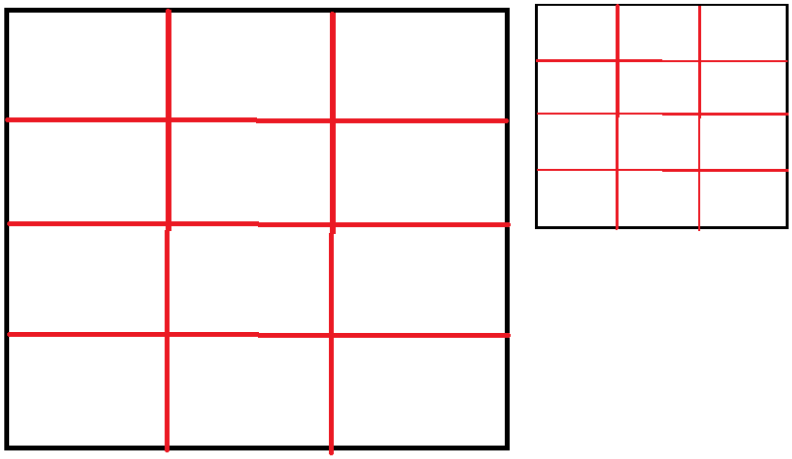
RoI Pooling :

在 Object Detection 的任務中，第一階段 CNN 除了會生成一個 Feature Map 外，也會指名感興趣區域(ROI) 的 Box coordination， ROI Pooling 希望將大小不一的 RoI 轉換成固定大小的 feature

將 ROI resize to Feature Map, 成為 Original'

希望 RoI Pooling 的輸出是一個固定大小的特徵圖

將 Original' 中的 ROI 原區**均勻地劃分成 H×W 個子網格**。每一格做 max pooling 即可放入網路

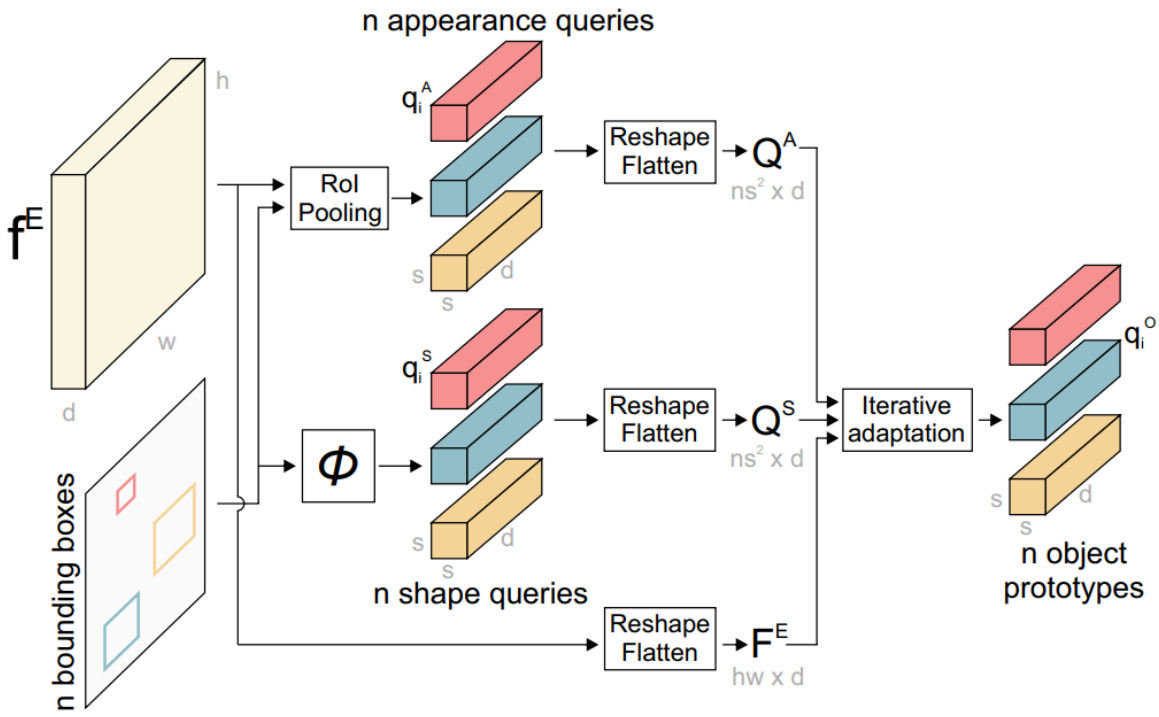


qA 相較起來比較 shape-agnostic ，沒有像 cfo max pooling 掉那麼多東西

shape query : 將向量 (box i 的 weight, box i 的 height) 放入 NN (nonlinear mapping) 成為 s * s * d 維的向量

three-layer feed-forward network $(2 \rightarrow 64 \rightarrow d \rightarrow s^2 d)$

$$\mathbf{q}_i^S = \phi([b_i^w, b_i^h])$$



recursive sequence of cross-attention blocks

找出 AK 對於該 SQ 的混合比例後，將 AV 混和

找出 FK 對於該 SQ′ 的混合比例後，將 FV 混和
以 S, A 為參考將 F 混和

— query 對應— output，所以 QL 屬於 R ns^2d

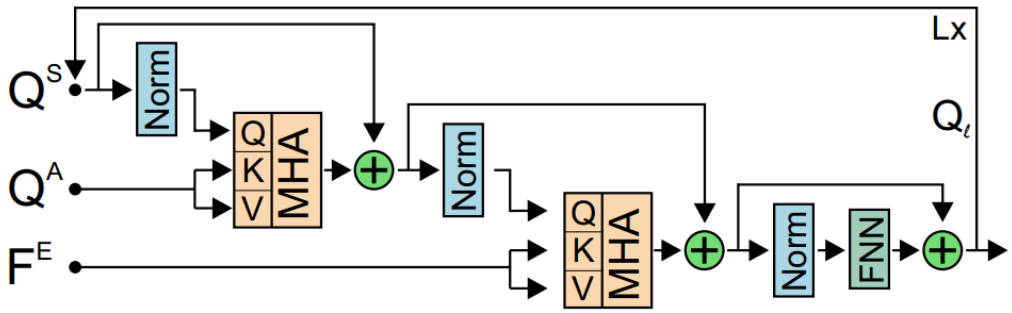
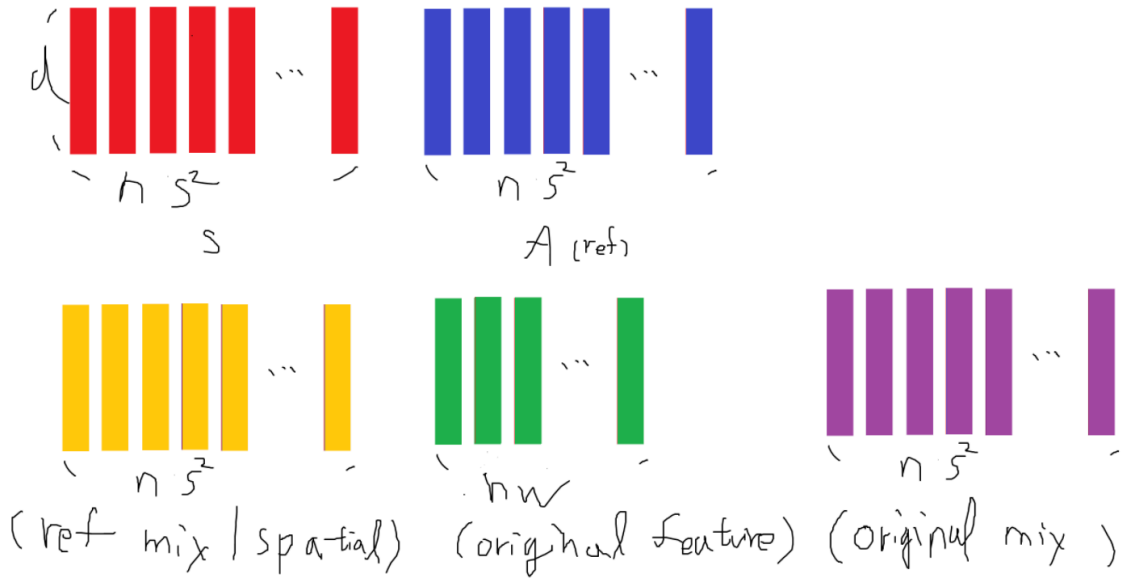


Figure 4. The iterative adaptation module applies attention to gradually generalize prototypes to the object instances indicated by few input exemplars.

將原本的 feature 做重新組合， given on S (spatial 資訊) & A (appearance, shape 資訊)

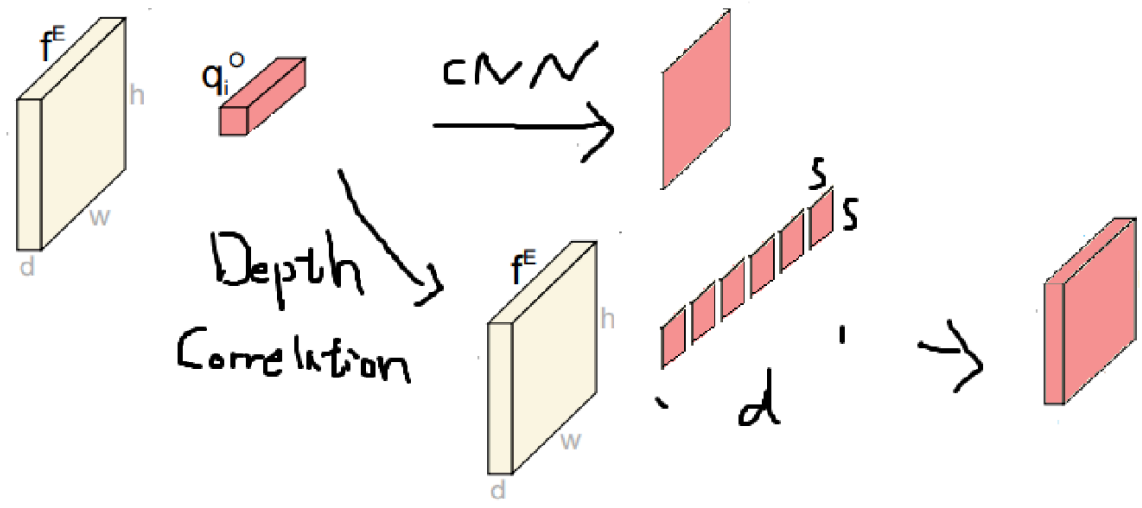


$$\tilde{\boldsymbol{R}}_i = \mathbf{f}^E * \mathbf{q}_i^O, \tag{1}$$

where $(*)$ is a depth-wise correlation. The individual n

因為 qi 是 fE 經過混和後的結果，所以兩個的 d 維度相同，要做 depth-wise conv

depth-wise conv : 每個深度獨立做，不像傳統 conv 加總，因此保證深度有 d



$$\mathcal{L}_{OSE} = \frac{1}{M}||\hat{\boldsymbol{G}} - \boldsymbol{R}||_2^2,$$

$$\mathcal{L}_{AUX} = \frac{1}{M} \sum_{\ell=1}^{L-1} ||\hat{\boldsymbol{G}} - \boldsymbol{R}^{\ell}||_2^2. \tag{6}$$

The final loss is thus $\mathcal{L} = \mathcal{L}_{OSE} + \lambda_{AUX}\mathcal{L}_{AUX}$, where

Experiments

可以學習的地方

問題