이학박사 학위논문

Approaches to the alleviation of the burden of learning for Weakly Supervised Object Localization

(약한 지도학습 기반의 물체 탐지에서의 학습 부담을 줄이기 위한 연구)

2023년 2월

서울대학교 대학원

수 리 과 학 부

구 본 경

Approaches to the alleviation of the burden of learning for Weakly Supervised Object Localization

(약한 지도학습 기반의 물체 탐지에서의 학습 부담을 줄이기 위한 연구)

지도교수 강 명 주

이 논문을 이학박사 학위논문으로 제출함

2022년 10월

서울대학교 대학원

수 리 과 학 부

구 본 경

구 본 경의 이학박사 학위논문을 인준함

2022년 12월

위 원 장 _____ (인)

부 위 원 장 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

Approaches to the alleviation of the burden of learning for Weakly Supervised Object Localization

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Bongyeong Koo

Dissertation Director : Professor Myungjoo Kang

Department of Mathematical Sciences
Seoul National University

February 2023

Abstract

In this thesis, we propose two models for weakly supervised object localization (WSOL). Many existing WSOL models have various burdens of learning, e.g., the nonnegligible cost of hyperparameter search for loss function. Thus, we first propose a model called SFPN to reduce the cost of hyperparameter search for loss function. SFPN enhances the information of the feature maps by exploiting the structure of feature pyramid network. Then these feature maps are engaged in the prediction of the bounding box. This process helps us use only cross-entropy loss as well as improving performance. Furthermore, we propose the second model named A2E Net to enjoy a smaller number of parameters. 'Spatial attention branch' and 'refinement branch' are the constituent parts of A2E Net. Spatial attention branch heightens the spatial information using few parameters. Also, refinement branch is composed of 'attention module' and 'erasing module', and these modules have no trainable parameters. With the output feature map of spatial attention branch, attention module makes the feature map with more accurate information by using a connection between pixels. Also, the most discriminative region is concealed by erasing module to make the network take account of the less discriminative region. Moreover, we boost the performance with multiple sizes of erasing. Finally, we sum up two output feature maps from attention module and erasing module to utilize information from these two modules. Extensive experiments on CUB-200-2011 and ILSVRC show the great performance of SFPN and A2E Net compared to other existing WSOL models.

Key words: Deep Learning, Object Localization, Weakly Supervised Learing
Student Number: 2017-21110

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Various computer vision tasks have benefited from deep learning [14, 15, 39]. These successful applications of deep learning are based on several factors, including task-specific labeling. For example, we need the bounding boxes as well as the class labels for object detection. However, making such labels for a specific task requires quite costly labor.

Weakly supervised learning (WSL) has been introduced to relieve this limitation. WSL aims to use 'weak' ground truth for learning. For example, WSL for semantic segmentation uses only the class labels for learning, while fully supervised methods use class labels and segmentation masks. This example implies that WSL reduces the burden caused by making the ground truth.

In this thesis, we address weakly supervised object localization (WSOL), which finds the class and the bounding box of the object by WSL. Namely, WSOL predicts the class and the bounding box while it uses only the class label for learning. For WSOL, class activation map (CAM) [63] has been one of the most commonly used methods. CAM finds the location that is important for classification by using the weight of the fully connected layer and the output feature map of convolutional neural network (CNN). One can obtain CAM if the classification network is trained because the fully connected layer and CNN are components of the classification network. However, CAM usually concentrates on the most discriminative region, e.g., the face of the bird. This may limit the performance of WSOL because object localization needs the whole region of the object. Thus, there are many WSOL models to

overcome this limitation.

Many existing WSOL models [5, 21, 33, 56, 60] are based on erasing. Here, erasing means removing the most discriminative region. Namely, erasing helps the network be interested in the less discriminative region by restricting the use of information from the most discriminative region. So. using this, ACoL [60] and MEIL [33] introduce another path that receives an erased input feature map and outputs corresponding to this input. These models are trained by using two paths jointly; one is to not use an erased feature map as input, and the other is the path mentioned above. Also, ADL [5], DGDM [56], and InCA [21] pick one architecture between attention mechanism and erasing at random.

On the other hand, there are many existing WSOL models that do not use erasing. For instance, SPG [61] gets the masks to help the learning. Also, DANet [55] makes groups of classes to extract unseen properties and uses the orthogonal relationship to see various areas of the object.

However, many existing WSOL models have a limitation that requires the burden of learning, e.g., the nonnegligible cost of hyperparameter search for loss function. For example, ACoL and MEIL need two loss terms from two paths, respectively. Other models [31, 34, 55, 61] also use many hyperparameters for loss function. The hyperparameter search for loss function often relies on time-consuming methods, e.g., the researcher's empirical knowledge. In particular, this point is likely to be a more larger burden when dealing with a large dataset such as ILSVRC [41].

To relieve the above limitation, we propose a model named "simple feature pyramid network" (SFPN) [22]. It aims to help the feature map enjoy more plentiful information. Since CAM depends on the output feature map of CNN, the feature map with more plentiful information can improve the performance. For this, we utilize many feature maps from CNN, while CAM exploits one feature map from CNN.

More precisely, we get feature maps from FPN [27]. These generated feature maps have high-level semantic information with diverse resolutions. These feature maps then are input feature maps of new blocks, named refinement blocks, which output the same shape of tensors. Next, the output feature maps of refinement blocks are concatenated. Finally, this concatenated feature

map passes through global average pooling (GAP) and classifier. Using many feature maps with heightened information helps to obtain improved performance using only cross-entropy loss for learning. Indeed, SFPN reveals great performance via experiments on CUB-200-2011 [50] dataset and ILSVRC [41] dataset.

A nonnegligible number of learnable weights can also be a burden. In general, resources such as the memory of GPUs are more demanding as the number of learnable weights increases. Indeed, SFPN has a nonnegligible number of learnable weights, although it helps us to get the improvement of the performance. Thus, we propose the second model named "aggregation of attention and erasing" (A2E Net) [23] to reduce the number of additional learnable parameters. A2E Net also still uses only cross-entropy loss for learning.

A2E Net introduces spatial attention branch (SAB) and refinement branch. SAB helps the feature map have heightened spatial information. It uses a transformation that consists of the neural network to get the pixel-wise weight. More precisely, it assigns the higher values to more important pixels for the classification and the lower values to less important pixels. Also, we use channel average pooling (CAP) for the transformation to benefit from a small number of learnable parameters. Since CAP returns the feature map having 1 its number of channels, it is suitable for SAB to accomplish this goal. Then the output feature map of SAB becomes the input feature map of refinement branch.

Refinement branch plays two roles with two modules, where they are named attention module and erasing module, respectively. Attention module in refinement branch makes the information of the input feature map more elaborate by using a connection between pixels. Also, erasing module in refinement branch considers the less discriminative region by erasing. Further, we add output feature maps from these modules to get information from attention module and erasing module more effectively. Thus, using these modules, refinement branch helps the network get a feature map with more elaborate information and concentrate on the whole region of the object.

These two branches have a small number of learnable parameters. More precisely, the transformation consumes all trainable weights in SAB, and the number of such parameters is small. Also, refinement branch has no learnable

parameters. Thus, A2E Net is lightweight. Moreover, through experiments on CUB-200-2011 [50] and ILSVRC [41], A2E Net shows great performance, which still uses only cross-entropy loss for training.

The rest of this thesis is organized as follows. In Chapter 2, we introduce convolutional neural networks (CNNs), which is a basic knowledge in this thesis. In Chapter 3, we describe SFPN in more detail. In Chapter 4, we explain A2E Net in more detail, and finally, we conclude this thesis in Chapter 5.

Chapter 2

Preliminaries

2.1 Convolutional Neural Networks

Image classification has been the important problem in computer vision. It is to predict the class given an image. To this problem, convolutional neural networks (CNNs), which consist of neural networks with convolution operation, have demonstrated great performance. So, we will discuss some CNNs. However, before doing this, we need to understand the convolution operation because the convolution operation is basic for CNNs. Thus, we first see the definition of the convolution operation and then search for some CNNs.

2.1.1 Convolution Operation

A 2D image can be represented as an element of $\mathbb{R}^{C\times H\times W}$, where H and W are the height and the width of the image, respectively, and C is the number of channels. Here, C is 1 if the image is grayscale and is 3 if the image is RGB. In general, we deal with the sets $\mathbb{R}^{C\times H\times W}$, where C, H, and W are any positive integers. Also, the elements of such sets are called a tensor.

In these conditions, we now introduce the convolution operation. The convolution operation is used to extract the feature of the input tensor. For this, it uses the cross-correlation with this input using another tensor called the kernel [12]. Here, there are several factors of cross-correlation, such as padding and stride. These factors have various goals, such as getting the

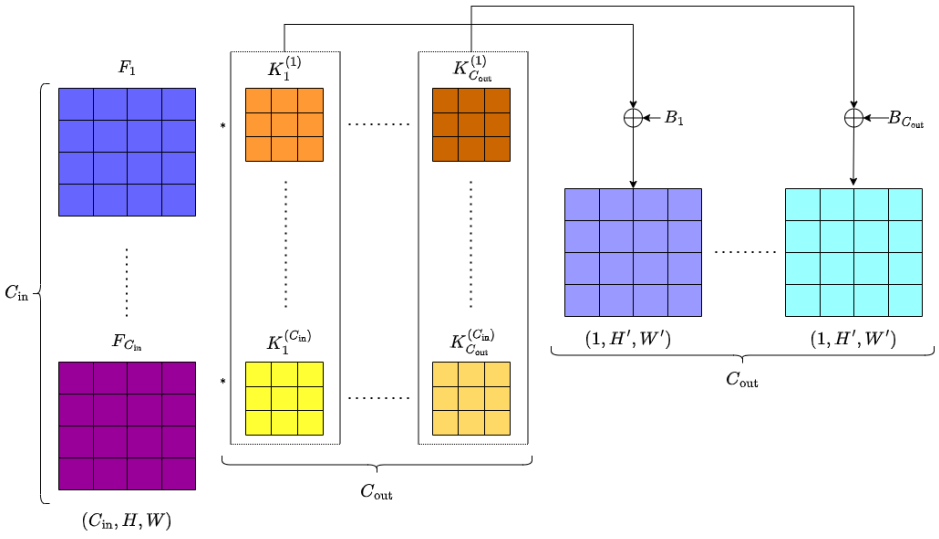


Figure 2.1: Standard Convolution with $H = W = 4$, $k_1 = k_2 = 3$, and $H' = W' = 4$.

desired size of the output. Also, the output of the convolution operation is called the feature map.

With the above discussion, we write the convolution operation. Let $F = [F_1, \cdots, F_{C_{\text{in}}}] \in \mathbb{R}^{C_{\text{in}} \times H \times W}$ be the tensor with $F_i \in \mathbb{R}^{H \times W}$ $(1 \leq i \leq C_{\text{in}})$. Here, C_{in} is the number of channels of F. Also, let $K = [K_1, \cdots, K_{C_{\text{out}}}]$ be the kernel with $K_j \in \mathbb{R}^{C_{\text{in}} \times k_1 \times k_2}$ $(1 \leq j \leq C_{\text{out}})$. Here, (k_1, k_2) is a pair of the height and the width of K, and C_{out} is the number of channels of the output. For the convenience, we write $K_j = [K_j^{(1)}, \cdots, K_j^{(C_{\text{in}})}]$ for each j, where each component of K_j has the shape of (k_1, k_2). Also, we consider the bias $B = [B_1, \cdots, B_{C_{\text{out}}}] \in \mathbb{R}^{C_{\text{out}}}$. Under these settings, we define the convolution operation as follows (see Figure 2.1 for the visual description of the convolution operation):

Definition 1 (Convolution Operation)**.** *The* ***Convolution operation*** *is a function* $\text{Conv}(F, K, B) : \mathbb{R}^{C_{in} \times H \times W} \to \mathbb{R}^{C_{out} \times H' \times W'}$ *that maps* $F \in \mathbb{R}^{C_{in} \times H \times W}$

into $F' \in \mathbb{R}^{C_{out} \times H' \times W'}$ *with*

$$F'_j = B_j + \sum_{i=1}^{C_{in}} K_j^{(i)} * F_i \in \mathbb{R}^{H' \times W'} \tag{2.1}$$

for each $1 \leq j \leq C_{out}$*. Here,* $*$ *is the cross-correlation operation of two tensors.*

Also, there is a modified convolution operation called 'group convolution' to enjoy a smaller number of parameters. In group convolution, C_{in} and C_{out} are split into some groups, and the convolution operation is used for each group independently. In addition, if the number of such groups is equal to C_{in}, group convolution for this case is called 'depthwise convolution'. To write group convolution, we denote $F_{i:j}$ as the slice of the given feature map F from i to j. Then group convolution is defined by

Definition 2 (Group Convolution)**.** *Suppose* g *is a positive integer such that divides both* C_{in} *and* C_{out}*. The* ***group convolution*** *is a function* $\text{Conv}(F, K, B, g) : \mathbb{R}^{C_{in} \times H \times W} \to \mathbb{R}^{C_{out} \times H' \times W'}$ *that maps* $F \in \mathbb{R}^{C_{in} \times H \times W}$ *into* $F' \in \mathbb{R}^{C_{out} \times H' \times W'}$ *where each component of* F' *has the shape of* $(C_{out}/g, H', W')$ *and the* i*th component of* F' *is computed by*

$$\text{Conv}(F_{(i-1)C_{in}/g+1\,:\,i\,\cdot\,C/g}, K_{(i-1)C_{in}/g+1\,:\,i\,\cdot\,C/g}, B_{(i-1)C_{out}/g+1\,:\,i\,\cdot\,C/g}). \tag{2.2}$$

Note that the computational cost is $k_1 \cdot k_2 \cdot H \cdot W \cdot \frac{C_{\text{in}}}{g} C_{\text{out}}$ when we use group convolution while standard convolution uses $k_1 \cdot k_2 \cdot H \cdot W \cdot C_{\text{in}} \cdot C_{\text{out}}$. Thus, group convolution uses a smaller computational cost than standard convolution. In particular, the computational cost is $k_1 \cdot k_2 \cdot H \cdot W \cdot C_{\text{out}}$ for depthwise convolution because $g = C_{\text{in}}$.

2.1.2 Some Convolutional Neural Networks

LeNet [26] conducts image classification by CNN. It uses not only CNN but also subsampling to get the smaller size of the feature map. After that, AlexNet [25] having 5 convolution layers and 3 fully connected layers shows its great performance for ILSVRC dataset [41]. To utilize multiple GPUs,

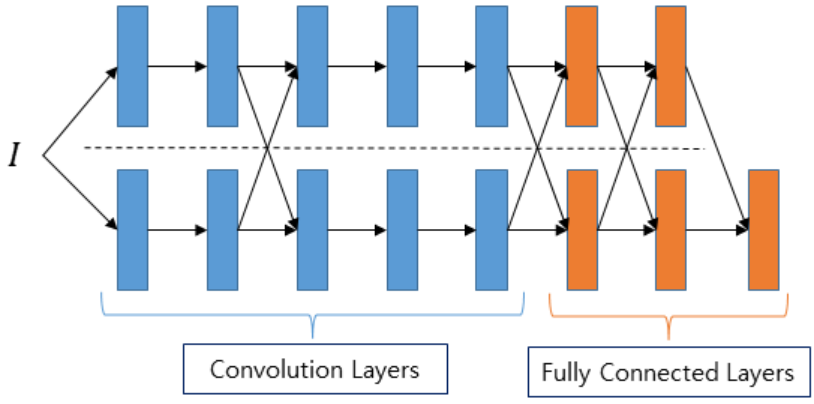


Figure 2.2: Description of AlexNet. The upper part of a dashed line corresponds to one GPU, and the lower part corresponds to another GPU.

AlexNet consists of a parallel structure. Also, AlexNet uses Dropout [45] and a different activation function from the previous works, called Rectified Linear Units (ReLUs), which is defined by

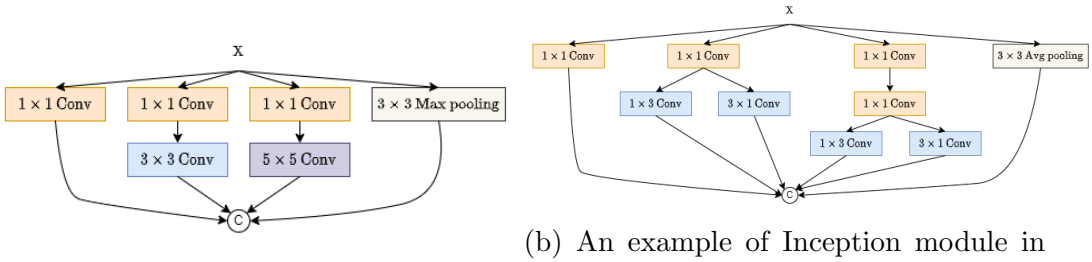
$$\mathrm{ReLU}(x) = \begin{cases} x, & x \geq 0, \\ 0, & x < 0 \end{cases} \tag{2.3}$$

For the visual description of AlexNet, refer to Figure 2.2.

ZFNet [58] has a similar structure to AlexNet except for the parallel structure. Here, ZFNet analyzes the feature maps by visualization. Then this analysis is used to improve the structure of AlexNet. For example, the first convolution layer of AlexNet uses the kernel of the size 11×11 with stride 4. However, the size and the stride decrease to 7×7 and 2 in ZFNet, respectively.

VGG [43] improves performance by using a deeper structure with a smaller size of the kernel. In contrast to the above models, VGG uses multiple 3×3 convolution layers to have a similar receptive field compared to larger kernel sizes. This requires a smaller number of learnable weights and makes use of more ReLU functions compared to using only one layer with a kernel of large size. Also, VGG shows that performance improves as the depth of the model is larger.

GoogleNet [46] also has a deeper structure with a more complex module,



(a) Inception module in GoogleNet

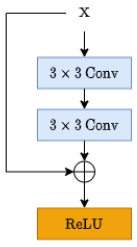
(b) An example of Inception module in Inception V3

Figure 2.3: Comparison between Inception module in GoogleNet and an example of Inception module in Inception V3. Here, X is the input feature map and ⓒ denotes the concatenation.

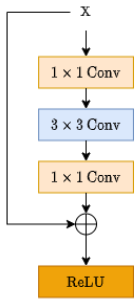
called "Inception module". $k \times k$ convolution layers ($k \in \{1, 3, 5\}$) and pooling layers are used to ingredients of Inception module. Here, 1×1 convolution layer is in front of $k \times k$ convolution layer ($k \in \{3, 5\}$) to decrease the number of channels. Then Inception module outputs the feature map by gathering the information from these layers. Moreover, the "auxiliary classifiers" are introduced to play a role in regularization.

Inception V3 [47] improves GoogleNet by the use of $n \times 1$ convolution layer and $1 \times n$ convolution layer. In other words, Inception V3 splits the $n \times n$ convolution layer into the $n \times 1$ convolution layer and the $1 \times n$ convolution layer to use a smaller number of parameters. With this modification, Inception V3 constructs various Inception modules. For the visual comparison between Inception module in GoogleNet and one example of these modified Inception modules, refer to Figure 2.3.

Also, Inception V3 uses the auxiliary classifier. However, in contrast to the case in GoogleNet, Inception V3 finds that the auxiliary classifier for the earlier layer is not important. Thus, Inception V3 allows only one auxiliary classifier. Furthermore, Inception V3 introduces a regularization called "label smoothing". The previous works use one-hot encoding, which assigns 1 for the index of the ground truth label and 0 for the other indexes. Inception V3 argues that one-hot encoding has two disadvantages due to a large difference between the ground truth and the others: overfitting and decreasing adaptability. Thus, given y as the ground truth label, label smoothing makes a new



(a) Basic Block in ResNet



(b) Bottleneck Block in ResNet

Figure 2.4: Comparison between basic block and bottleneck block in ResNet. X is the input feature map and $\oplus$ denotes the elementwise addition.

label by

$$(1-\epsilon)\delta_{k,y} + \epsilon u(k) \tag{2.4}$$

where k is the position of the classes, $\delta_{k,y}$ is the Kronecker delta, and ϵ and $u(k)$ are the predefined value and distribution, respectively. As seen in Equation 2.4, the indexes which are not the ground truth are also assigned positive values to reduce the gap between the ground truth and the others.

ResNet [15] suggests deeper models which alleviate the degradation problem. The degradation problem means that it is hard for the network to improve further performance as its depth is larger. To preserve the large depth of the network and improve performance, ResNet uses "shortcut connections". Here, shortcut connections mean that when X is the input feature map, the output feature map is given by

$$X + f_\theta(X), \tag{2.5}$$

where f_θ is the neural network with parameter θ. These shortcut connections help construct the deeper networks such as 50 layers, 101 layers, and 152 layers and improve performance as the depth is larger. Figure 2.4 shows the visual examples of shortcut connections.

Although the deeper models show great performance, it is not easy to apply these models to devices that do not support abundant computational

resources. MobileNet V1 [16] is an attempt to use CNN in such devices. Namely, MobileNet V1 aims to make a network with a small computational cost. To this end, the convolution layer that consists of depthwise convolution and 1×1 convolution, which was proposed in [42], is used in MobileNet V1. As we discussed in Section 2.1.1, depthwise convolution outputs the feature map with the same number of channels as the input feature map and requires a smaller computational cost than standard convolution. After using depthwise convolution, 1×1 convolution controls the number of channels. Moreover, MobileNet V1 uses two factors to determine the number of channels or the resolution. So, using these factors, one can create a smaller network.

Chapter 3

SFPN: Simple Feature Pyramid Network for Weakly Supervised Object Localization

3.1 Introduction

Although many existing WSOL methods improve the performance, these methods use loss functions with many hyperparameters. Thus, these models need to find suitable hyperparameters of loss functions. This point can increase the time of learning because the hyperparameter search for loss function often has a wide range of candidates or uses multiple tests to get better performance. Especially when dealing with a large dataset such as ILSVRC [41], this point can be more nonnegligible.

To avoid this issue, we aim to use a simple loss function. For this, we recall CAM [63]. CAM is defined as follows.

Let T be the number of classes. Also, suppose that we have the classification network composed of CNN (ϕ_θ), aggregation operation, and a fully connected layer. Here, aggregation operation means the operation that extracts the information from the output feature map of ϕ_θ. An example of this operation is the global average pooling. Then given an image I, we write $f = \phi_\theta(I)$ with $f = [f_1, \cdots, f_L] \in \mathbb{R}^{L\times H\times W}$, where $f_i \in \mathbb{R}^{H\times W}$ for each $1 \leq i \leq n$. Also, we let $W = [w_{l,t}] \in \mathbb{R}^{L\times T}$ and $b \in \mathbb{R}^T$ be the weight and the bias of the fully

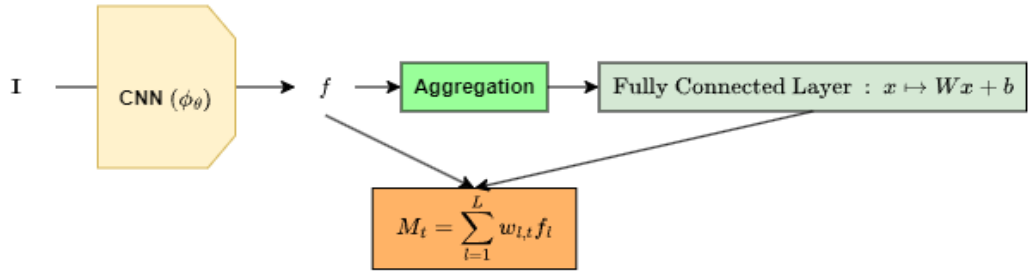


Figure 3.1: The flow of CAM. Here, $f = [f_1, \cdots, f_L] \in \mathbb{R}^{L \times H \times W}$ is the output feature map of CNN and $W = [w_{l,t}] \in \mathbb{R}^{L \times T}$ is the weight of fully connected layer. Also, aggregation means the operation that extracts the information from f (e.g., global average pooling).

connected layer, respectively. Then CAM : $\mathbb{R}^{L \times H \times W} \times \mathbb{R}^{L \times T} \to \mathbb{R}^{T \times H \times W}$ is defined by

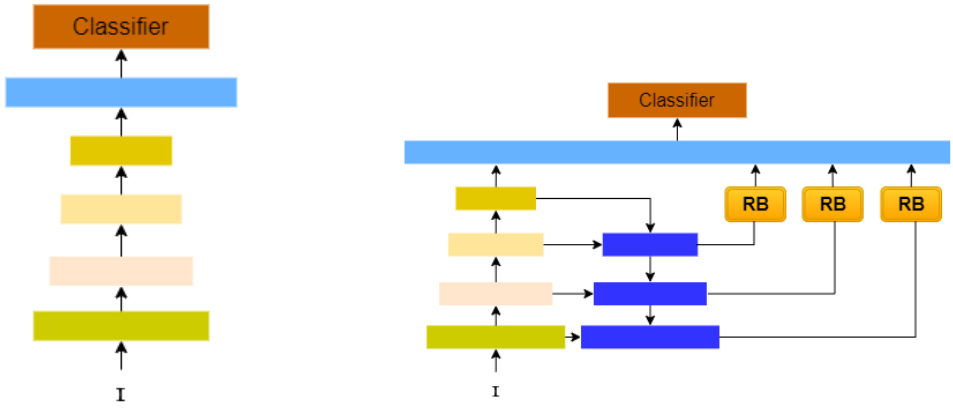
$$\mathrm{CAM}(f, W) = [M_1, \cdots, M_T] \in \mathbb{R}^{T \times H \times W}, \tag{3.1}$$

where $M_t = \sum_{l=1}^{L} w_{l,t} f_l \in \mathbb{R}^{H \times W}$ for each class $1 \leq t \leq T$. This process is visualized in Figure 3.1.

Equation 3.1 for CAM provides the clue for our goal. First, CAM often uses only cross-entropy loss for learning because the classification network is only the required condition. Next, CAM uses the 'last feature map', which is f in Equation 3.1. Thus, the last feature map with more plentiful information can improve the localization, and it may allow still using only cross-entropy loss for learning.

To this end, we utilize the FPN structure. FPN uses multiple tensors with reinforced semantic information. Similarly, we first make multiple tensors with strengthened information and then use these tensors as the last feature map, while CAM uses only one tensor as the last feature map. In the end, we propose SFPN [22] that uses this process. Figure 3.2 describes the difference between the last feature map for standard CNN and the last feature map for SFPN.

The remainder of this chapter is as follows. First, to explain the back-



(a) Standard CNN (b) SFPN

Figure 3.2: Comparison between (a) standard CNN and (b) SFPN. I means the input image, and the sky-blue rectangle denotes the last feature map for CAM. Also, in (b), the blue block denotes the operation that combines feature maps, and RB means the refinement block.

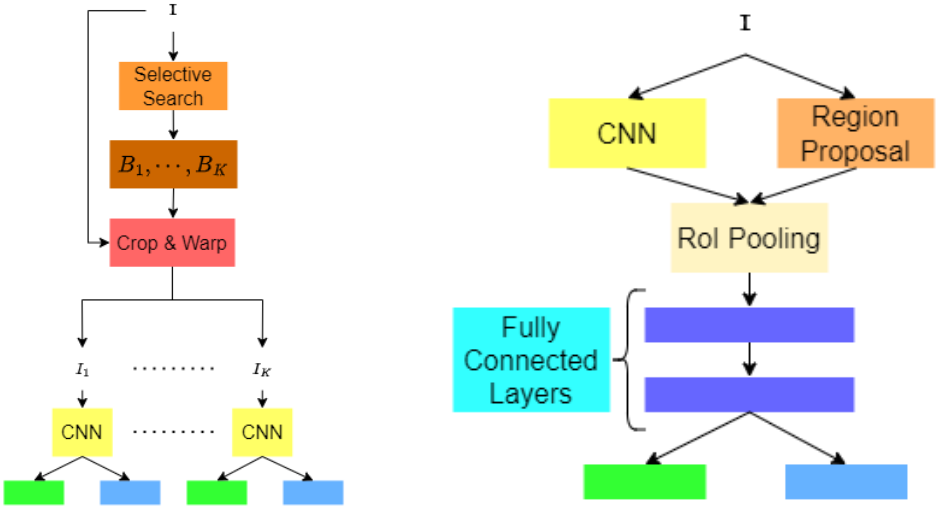
ground for SFPN, we show some object detection models including FPN and existing WSOL models in Section 3.2. In particular, we observe that many existing WSOL models use complex loss functions by describing these models in more detail. Then we show the detail and the results of SFPN in the remaining parts.

3.2 Related Works

3.2.1 Some Object Detection Methods

Object detection is interested in finding the position and the class of each object in an image. Since deep learning has improved the classification, which finds the class of each object in an image, many attempts have applied deep learning techniques to object detection.

R-CNN [11] shows the application of deep learning with region proposal. Here, region proposal plays a role in providing the areas where objects potentially exist. R-CNN uses selective search [49] as region proposal and gets the candidate bounding boxes from this. These bounding boxes make multiple



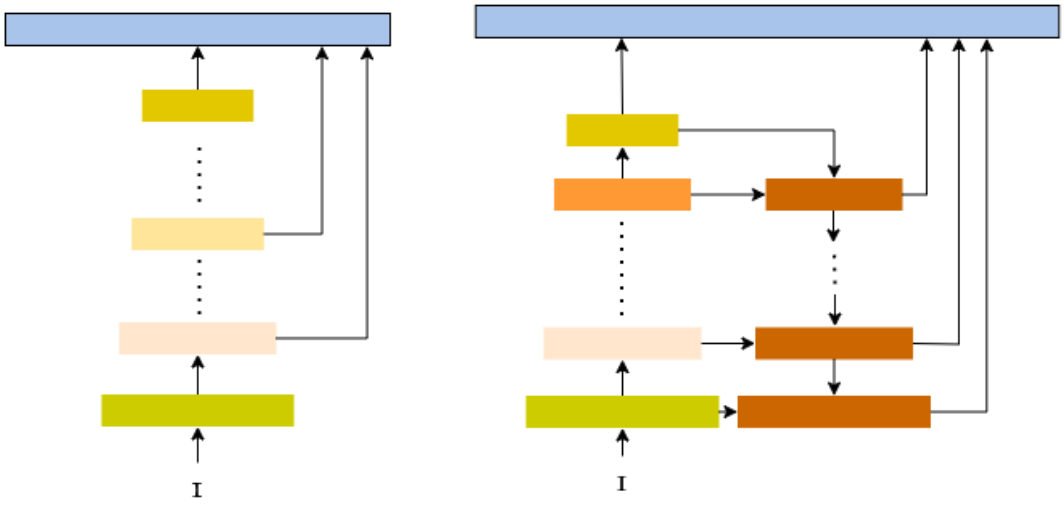
(a) R-CNN (b) Fast R-CNN / Faster R-CNN

Figure 3.3: Descriptions of R-CNN object detectors. I means the input image. Also, the green box and the sky-blue box denote classification and regression for the bounding box, respectively. Here, in (b), region proposal includes selective search for Fast R-CNN and includes RPN for Faster R-CNN.

images by cropping the input image, and these generated images become new input of CNNs.

R-CNN uses the huge computational cost because it applies CNN for each cropped image. Thus, Fast R-CNN [10] does not generate multiple images. Instead, CNN in Fast R-CNN receives only one input image. Also, selective search still gives the candidate bounding boxes in Fast R-CNN. However, in contrast to R-CNN, the bounding boxes from selective search are applied to the output feature map of CNN by adjusting these bounding boxes, called "region of interest (RoI) projection". With RoI projection, the output feature map of CNN gives multiple feature maps corresponding to projected bounding boxes, called "RoI pooling". Here, the resulting feature maps have the same shape because RoI pooling aims to make the input feature map of subsequent fully connected layers.

Fast R-CNN improves the speed by using only one image as the input of CNN. However, region proposal technique applied to Fast R-CNN uses CPU,



(a) SSD (b) FPN

Figure 3.4: Descriptions of SSD and FPN. I means the input image.

which is room for improving speed. Thus, Faster R-CNN [39] replaces CPU-based region proposal with neural networks, called "region proposal network (RPN)". Similar to Fast R-CNN, CNN in Faster R-CNN outputs the feature map from a given image. Then RPN outputs two feature maps with this feature map as the input. Here, RPN uses anchor boxes, which are predefined bounding boxes, for each grid and predicts the bounding box and the presence of the object per anchor box. With the ground truth bounding boxes as supervision, these output feature maps are used to train RPN. After that, RPN provides region proposal, and the remainder is similar to Fast R-CNN. Models from R-CNN to Faster R-CNN are described in Figure 3.3, and the model similar to these models is called the two-stage method.

The above models have shown great object detection. However, these models still have a room for improving speed. To increase the speed, YOLO [38] outputs the bounding boxes and the classes directly. More precisely, it regards the input image as a set of cells and infers the bounding boxes and the classes for each cell. However, the accuracy of YOLO is not high because it has a limit to detecting small objects.

To make the higher accuracy, SSD [29] uses multiple feature maps hav-

ing different shapes. SSD introduces default boxes and assigns these boxes for each feature map. Here, various sizes of feature maps and diverse aspect ratios give default boxes with many shapes. Then SSD predicts the bounding boxes and classes for each default box. Since default boxes have diverse sizes, SSD can deal with various sizes of objects, and so it improves performance. Also, it maintains the increased speed because it outputs the results directly, similar to YOLO.

The model similar to SSD and YOLO is called the one-stage method. These one-stage methods directly output the bounding boxes and the classes, while the two-stage methods first produce region proposals and then estimate the bounding boxes and the classes based on these proposals.

FPN [27] improves the use of multiple feature maps. In CNN, the high-resolution feature map is one of the keys to recognizing small objects. However, SSD uses lower-resolution feature maps because low-level information in the feature map of high-resolution may not help improve object detection. Thus, FPN resolves this dilemma by injecting semantic information of low-resolution feature map into high-resolution feature map. More precisely, since a lower-resolution feature map has better semantic information, FPN gives the higher-resolution feature maps the information of the lower-resolution feature maps by the addition. Hence, this process enhances semantic information of high-resolution feature maps. As a result, the two stage methods, including Faster R-CNN, with FPN show improved performance. For the comparison between SSD and FPN, refer to Figure 3.4.

FPN can also be applied to the one-stage method. RetinaNet [28] produces multiple feature maps from FPN and then outputs the bounding boxes and the classes for each anchor box. Also, it proposes a modified loss function from the cross-entropy loss function, called "focal loss". The focal loss is designed to mitigate class imbalance. In general, easy examples (e.g., background) have a larger proportion than hard examples (e.g., foreground) in an image, and thus, these easy examples can reduce the efficiency of training. The two-stage methods are more easily to address this problem by filtering easy examples using region proposal. However, it is hard for the one-stage methods to deal with this problem because these methods directly predict the outputs without region proposal. So, the focal loss induces the decrease

of the importance of easy examples.

3.2.2 Existing Methods for Weakly Supervised Object Localization

SPG [61] gets a seed for foreground and background and uses this seed to guide learning. To this end, SPG first gets the intermediate feature maps, say F_i ($i \in \{1, 2, 3, 4, 5\}$), where the lower index indicates the resulting feature map of the lower layer. Then using thresholding to these feature maps and interpolation, SPG uses the loss function having the following form:

$$\lambda \mathcal{L}_{\text{cls}}(F_5, y) + \lambda_1 \mathcal{L}_{\text{bce}}(F_1, F_2)) + \lambda_2 \mathcal{L}_{\text{bce}}(F_2, F_4)) + \lambda_3 \mathcal{L}_{\text{bce}}(\text{Fuse}(F_1, F_2), F_3), \tag{3.2}$$

where y is the target, $\mathcal{L}_{\text{cls}}$ is the classification loss function, $\mathcal{L}_{\text{bce}}$ is the binary cross-entropy loss function, "Fuse" denotes the operation for combining two feature maps, and λ_1, λ_2, λ_3, and λ are hyperparameters. Here, we denote the second argument of $\mathcal{L}_{\text{bce}}$ as a mask generated from thresholding, which plays a role in supervision.

Peca-Net [31] is inspired from Piotr et al. [7]. Piotr et al. [7] aim to generate a high-quality saliency map by considering the region related to image classification. More explicitly, they consider two regions; one is the smallest region that makes it possible to categorize the classes properly, and the other is the smallest region that is hard for the network to conduct suitable classification. To this end, for the input image X and the generated mask M from U-Net [40] structure, they make two new images by combining X with M, called $\phi(X, M)$ and $\phi(X, 1 - M)$, respectively. Then they maximize the probability of $\phi(X, M)$ for the class while minimizing the probability of $\phi(X, 1 - M)$ for the class. In addition, removing artifacts in M can further improve performance. So, they use the following loss function to reflect this process:

$$\lambda_1 TV(M) + \lambda_2 AV(M) - \log f_c(\phi(X, M)) + \lambda_3 f_c(\phi(X, 1 - M))^{\lambda_4}, \tag{3.3}$$

where $TV(M) = \sum_{i,j}(M_{i,j} - M_{i,j+1})^2 + \sum_{i,j}(M_{i,j} - M_{i+1,j})^2$ makes M smoother, f_c is the probability for the class c given an image, and $AV(M)$ is the mean of M.

Similarly, Peca-Net makes a mask M from U-Net structure, adding class attention module in the encoder. Then it uses the following loss function to get a more accurate mask:

$$\lambda_1 l_D(1 - M, X) - \lambda_2 l_P(M, X) + \lambda_3 l_{\mathrm{TV}}(M) + \lambda_4 l_A(M), \tag{3.4}$$

where

$$l_D(1 - M, X) = \log p(y = c \mid \phi(X, 1 - M)), \tag{3.5}$$

$$l_P(M, X) = \log p(y = c \mid \phi(X, M)), \tag{3.6}$$

$$l_{\mathrm{TV}}(M) = \sqrt{\sum_{x,y}(M_{x,y} - M_{x,y+1})^2 + \sum_{x,y}(M_{x,y} - M_{x+1,y})^2}, \tag{3.7}$$

$$l_A(M) = \frac{1}{HW}\sum_{x=1}^{H}\sum_{y=1}^{W} M(x, y)^2, \tag{3.8}$$

and λ_1, λ_2, λ_3, and λ_4 are hyperparameters.

DANet [55] groups classes to extract unseen properties. Also, it sees various areas of the object by inducing the orthogonal relationship. More explicitly, DANet first needs the hierarchical classes C_{root}, C_{parent}, and C_{child}. Then it generates the intermediate feature maps from CNN. Here, each generated feature map corresponds to the hierarchical classes, say F_{root}, F_{parent}, and F_{child}. These feature maps are used to predict the corresponding hierarchical class by the average operation and are exploited to compute cosine loss. Thus, the resulting loss function is

$$\lambda_1 \mathcal{L}_{\mathrm{cls}}(F_{\mathrm{root}}, y_{C_{\mathrm{root}}}) + \lambda_2 \mathcal{L}_{\mathrm{cls}}(F_{\mathrm{parent}}, y_{C_{\mathrm{parent}}}) + \lambda_3 \mathcal{L}_{\mathrm{cls}}(F_{\mathrm{child}}, y_{C_{\mathrm{child}}}) + \lambda \cos F, \tag{3.9}$$

where $y_{C_{\mathrm{root}}}$, $y_{C_{\mathrm{parent}}}$, and $y_{C_{\mathrm{child}}}$ are targets corresponding to C_{root}, C_{parent}, and C_{child}, respectively, $\mathcal{L}_{\mathrm{cls}}$ is the loss function for image classification, F comes from the concatenation of F_{root}, F_{parent}, and F_{child}, and λ_1, λ_2, λ_3,

and λ are hyperparameters of the loss function. Since 90° produces the zero value of cosine function, $\cos F$ induces the orthogonal relationship between the channels of the input feature map F.

CSTN [34] uses multiple feature maps from FPN and applies STN [20] in a convolution manner. First, multiple feature maps from FPN help to handle various scales. Also, since STN is suitable for dealing with various geometric versions involving translation, rotation, and scale, it gives more flexible operations. More explicitly, for the input feature map f, convolutional STN gives the parameters of transform $\theta_l \in \mathbb{R}^{2\times 3}$ for each location l of f. Also, to reduce overfitting, CSTN uses two regularization terms: $\mathcal{L}_\theta$ and $\mathcal{L}_{\text{scale}}$.

$\mathcal{L}_\theta$ aims to reduce the perturbation from θ_{ref}. Thus, it has the form of

$$\mathcal{L}_\theta = \sum_{s\in S} \sum_{i=1}^{h_s \times w_s} \|\theta_{\text{ref}} - \theta_i\|^2, \tag{3.10}$$

where S is a set of pyramid levels of FPN, and h_s and w_s are the height and the width of the feature map corresponding to the pyramid level s, respectively. Also, $\mathcal{L}_{\text{scale}}$ is used to encourage the higher-resolution feature map of the network to have a larger value when treating a large object. So, it is of

$$\mathcal{L}_{\text{scale}}(x) = \max\{0, \max_l p(s = s_1, l, c = c^*|x) - \max_l p(s = s_2, l, c = c^*|x)\}, \tag{3.11}$$

where $s_1, s_2 \in S$ and CSTN finally uses

$$\mathcal{L}_{\text{cls}}(x, y) + \lambda\mathcal{L}_\theta + \alpha\mathcal{L}_{\text{scale}}(x), \tag{3.12}$$

where λ and α are the hyperparameters for loss function. SFPN also uses FPN. However, SFPN does not use STN and has simpler loss function that consists of only cross-entropy loss.

There are also data augmentation methods for WSOL. Cutmix [57] and HaS [44] modify the input image. HaS first splits the input image into multiple cells and then removes these cells at random. More precisely, we denote the input image by I. Here, we assume I has a height H and width W. Also, let s be the size of the cell. Then a set $\{I_j\}_{j=1}^{HW/s^2}$ consists of a partition of I, where

$I_j \in \mathbb{R}^{s \times s}$ for each j. Next, HaS generates a binary mask $M := [M_j]_{j=1}^{HW/s^2}$ by

$$M_j \in \mathbb{R}^{s \times s}, \quad M_j \sim \text{Bernoulli}(p) \tag{3.13}$$

for each j, where p is a probability of not removing a cell. Finally, HaS outputs the modified input image $I' := M \odot I$, where the jth cell I'_j of I' is

$$I'_j = M_j \odot I_j \tag{3.14}$$

for each $1 \leq j \leq \frac{HW}{s^2}$. Then I' is the new input image, and this helps the network see the less discriminative region.

Although HaS is useful to see the less discriminative region, HaS can cause the loss of information by setting the values to zero. To overcome this limitation, Cutmix [57] outputs a new input image by dealing with multiple images. Let I_1 and I_2 be two images of a height H and width W. Also, let $y_1 \in \mathbb{R}^T$ and $y_2 \in \mathbb{R}^T$ be targets corresponding to I_1 and I_2, where T is the number of classes, respectively. Then Cutmix computes

$$\begin{aligned} \tilde{I} &= M \odot I_1 + (1 - M) \odot I_2, \\ \tilde{y} &= \lambda y_1 + (1 - \lambda) y_2, \end{aligned} \tag{3.15}$$

where

$$B := \{(i, j) : x_1 \leq i \leq x_2,\ y_1 \leq j \leq y_2\}, \tag{3.16}$$

$$M[i, j] = \begin{cases} 1 & (i, j) \notin B, \\ 0 & (i, j) \in B, \end{cases} \tag{3.17}$$

$$\lambda = 1 - \frac{(x_2 - x_1)(y_2 - y_1)}{HW}. \tag{3.18}$$

Here, (x_1, x_2) ((y_1, y_2)) is a pair of randomly generated x-coordinates (y-coordinates), and λ is the ratio between the area of the truncated region and that of the whole region. Thus, $\tilde{I}$ in Equation 3.15 means that the area within B in I_1 is filled with the corresponding area of I_2. This process reduces the loss of information because the 'hole' of the image is filled with another image.

In addition to the above augmentations, erasing is also useful for WSOL.

ACoL [60] and MEIL [33] use 'additional classifier' to deal with information from the less discriminative region. Let $I \in \mathbb{R}^{3\times H\times W}$ be the input image with a height H and W and let f_θ be a CNN with a parameter θ. Then both ACoL and MEIL have two outputs o_1 and o_2 such that

$$o_1 = c_{\theta_1}(f_\theta(I)), \quad o_2 = c_{\theta_2}(M \odot f_\theta(I)), \tag{3.19}$$

where M is a binary mask which removes the most discriminative region, $\odot$ is the elementwise product, and c_{θ_1} and c_{θ_2} are classifiers with parameter θ_1 and θ_2, respectively. Here, $\theta_1 \neq \theta_2$ for ACoL and $\theta_1 = \theta_2$ for MEIL. Finally, they compute the loss $\mathcal{L}$ by

$$\mathcal{L} = \mathcal{L}_1(o_1, y) + \lambda\mathcal{L}_2(o_2, y), \tag{3.20}$$

where y is a target, $\mathcal{L}_1$ and $\mathcal{L}_2$ are the classification loss functions, and λ is a hyperparameter for $\mathcal{L}$. Note that for MEIL, the number of loss terms of $\mathcal{L}$ can be increased when MEIL uses 3 classifiers. Thus, as seen in Equation 3.20, these approaches need the finding of hyperparameters for the loss function.

ADL [5] introduces another way to exploit erasing, not adding a new classifier. Namely, ADL picks one architecture between attention and erasing at random. More precisely, for the input feature map X, ADL outputs o such that

$$o = w_1 f_1(X) + w_2 f_2(X), \tag{3.21}$$

where f_1 uses attention mechanism, f_2 removes the pixels greater than foreground threshold, and w_1 is a binary random variable with $w_1 + w_2 = 1$.

We can observe that many of the above models use additional hyperparameters of loss functions. This means that these models need the finding of such hyperparameters and, thus, can cause the burden of learning, such as the increased time of learning.

3.3 Proposed Method

ResNet-101 [15] is used as the backbone network of SFPN. First, we split the backbone into four stages (see Table 3.1). For each stage, we assume the ith stage gives the feature map S_i. Next, FPN provides the tensor T_i corresponding to S_i. Then we apply refinement blocks, which will be introduced later in Definition 3. The resulting feature maps of such blocks will be called U_1, U_2, U_3, and U_4. These feature maps are concatenated and passes through GAP and a fully connected layer.

Stage		Output Shape
Stage 1	$[7 \times 7, 64]$ 3×3 maxpool $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$(256, 56, 56)$
Stage 2	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$(512, 28, 28)$
Stage 3	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$(1024, 14, 14)$
Stage 4	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$(2048, 14, 14)$

Table 3.1: Stages for ResNet-101 [15]. Here, the input shape is $(3, 224, 224)$ and the notation for convolution is [kernel $\times$ kernel, channels].

FPN requires feature maps of the same shape because it adds two feature maps. Thus, bilinear interpolation plays a role in making the same height and width, and 1×1 convolution helps two feature maps have the same number of channels. In particular, we make 1×1 convolution output 256 channels. Also, we assume the shape of the input image is $(3, 224, 224)$.

We now explain the proposed method in more detail. To help understand-

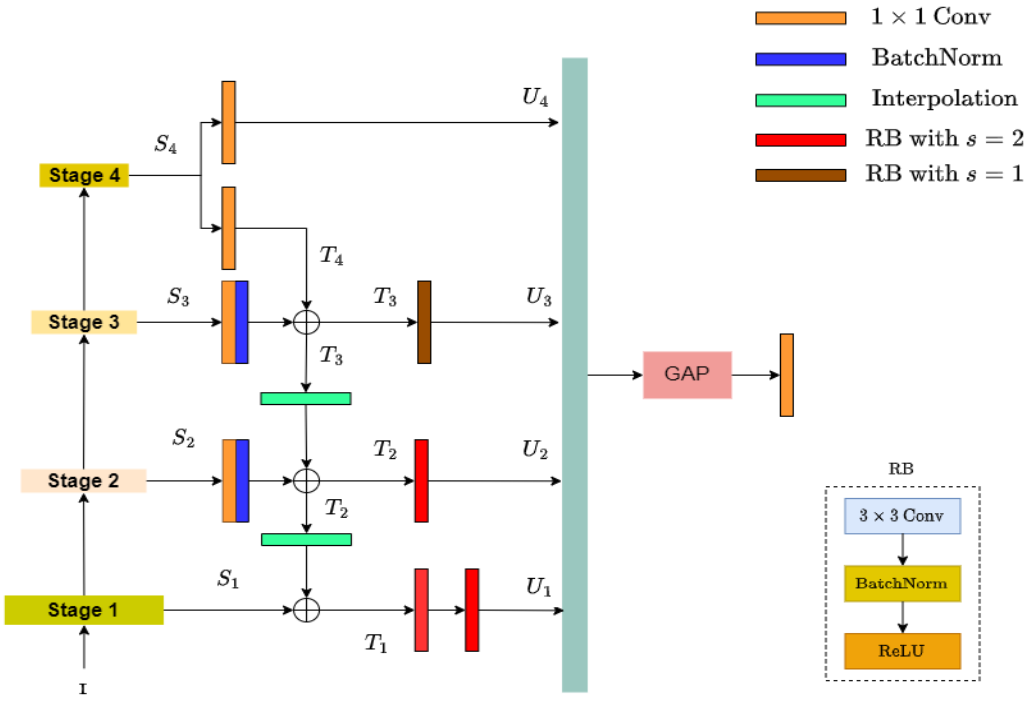


Figure 3.5: Detail of SFPN. Here, RB with s means the refinement block with stride s, GAP is global average pooling, and $\oplus$ denotes elementwise sum.

ing, see Figure 3.5. We start with S_4. To make S_4 have a larger resolution, we change the stride of the last layer for ResNet-101 from 2 to 1. As a result, S_4 has a height of 14 and a width of 14. On the other hand, S_4 has 2048 as the number of channels, while S_3 has 1024 as that of channels. So, 1×1 convolution produces two feature maps that have the same number of channels from S_3 and S_4, respectively. Also, batch normalization (BN) [18] is applied to $\mathrm{Conv}_{1\times1}(S_3)$, where $\mathrm{Conv}_{1\times1}(S_3)$ denotes the output of 1×1 convolution using S_3 its input. Then we have $T_4 \in \mathbb{R}^{256\times14\times14}$ and $T_3 \in \mathbb{R}^{256\times14\times14}$ following the below equation:

$$T_4 = \mathrm{Conv}_{1\times1}(S_4) \in \mathbb{R}^{256\times14\times14}, \tag{3.22}$$

$$T_3 = T_4 + (\mathrm{BN} \circ \mathrm{Conv}_{1\times1})(S_3) \in \mathbb{R}^{256\times14\times14}, \tag{3.23}$$

where $\mathrm{Conv}_{1\times1}$ means 1×1 convolution.

Next, we combine $S_2 \in \mathbb{R}^{512\times28\times28}$ and $T_3 \in \mathbb{R}^{256\times14\times14}$. Since S_2 has the shape of $(512, 28, 28)$, we use bilinear interpolation for T_3. Also, similar to S_3, a layer that consists of 1×1 convolution and BN receives S_2. With this process, T_2 is defined by the following equation:

$$T_2 = (\mathrm{BN} \circ \mathrm{Conv}_{1\times1})(S_2) + \mathrm{Interpolation}(T_3) \in \mathbb{R}^{256\times28\times28}, \tag{3.24}$$

where Interpolation means bilinear interpolation.

Finally, we convert T_2 into a tensor of shape $(256, 56, 56)$ using bilinear interpolation and add this tensor to S_1. Since S_1 has 256 channels, we do not use 1×1 convolution. The resulting tensor is denoted by T_1, and the above process is described in the following equation:

$$T_1 = S_1 + \mathrm{Interpolation}(T_2) \in \mathbb{R}^{256\times56\times56}. \tag{3.25}$$

Now, we have 4 tensors: T_1, T_2, T_3, and T_4. However, we can observe that these tensors have different shapes. For simplicity, we aim to use tensors of the same shape. Thus, we introduce refinement block. Refinement block is a stacked block, where 3×3 convolution, BN, and ReLU are its ingredients. We define refinement block as the following.

Definition 3 (Refinement block)**.**

1. *Let*

$$f_{\theta,s,p} = \mathrm{ReLU} \circ \mathrm{BN} \circ \mathrm{Conv}_{3\times3,s,p}, \tag{3.26}$$

where BN is batch normalization, θ is a collection of learnable parameters, and $\mathrm{Conv}_{3\times3,s,p}$ *a 3×3 convolution having s as its stride and p as its padding. Then we call $f_{\theta,s,p}$ as* ***unit layer****.*

2. *We define* ***refinement block*** *as*

$$\mathrm{Refine}_{\theta,n,s,p} = f_{\theta_1,s_1,p_1} \circ \cdots \circ f_{\theta_n,s_n,p_n}, \tag{3.27}$$

where n denotes the number of unit layers, $\theta = \cup_{i=1}^{n}\theta_i$, $s = \cup_{i=1}^{n}s_i$, and $p = \cup_{i=1}^{n}p_i$.

Note that $\text{Refine}_{\theta,1,s,p} = f_{\theta,s,p}$. With Definition 3, we apply refinement blocks. Here, the final shape we expect is (256,14,14). Thus, we have the output tensors U_i ($i \in \{1,2,3,4\}$) by

$$U_1 = \text{Refine}_{\Theta_1,2,s=\{2,2\},p=\{1,1\}}(T_1) \in \mathbb{R}^{256\times14\times14}, \tag{3.28}$$

$$U_2 = \text{Refine}_{\Theta_2,1,s=\{2\},p=\{1\}}(T_2) \in \mathbb{R}^{256\times14\times14}, \tag{3.29}$$

$$U_3 = \text{Refine}_{\Theta_3,1,s=\{1,1\},p=\{1,1\}}(T_3) \in \mathbb{R}^{256\times14\times14}, \tag{3.30}$$

$$U_4 = \text{Conv}_{1\times1}(S_4) \in \mathbb{R}^{256\times14\times14}. \tag{3.31}$$

After attaining U_1, U_2, U_3, and U_4, we concatenate these tensors and utilize global average pooling layer. With this output feature map of concatenation, the estimated class comes from 1×1 convolution. This process can be shown in the following equation:

$$f = \text{Concat}[U_1, U_2, U_3, U_4] \in \mathbb{R}^{1024\times14\times14}, \tag{3.32}$$

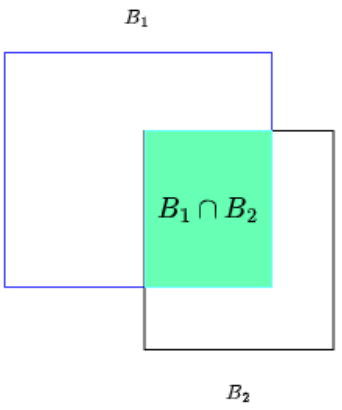
$$w_{\text{classifier}} \circ \text{GAP}(f) \in \mathbb{R}^T, \tag{3.33}$$

where 'Concat' is concatenation, $w_{\text{classifier}}$ is a 1×1 convolution layer that predicts the class, and T is the number of classes. Lastly, f in Equation 3.32 and $w_{\text{classifier}}$ in Equation 3.33 give CAM.

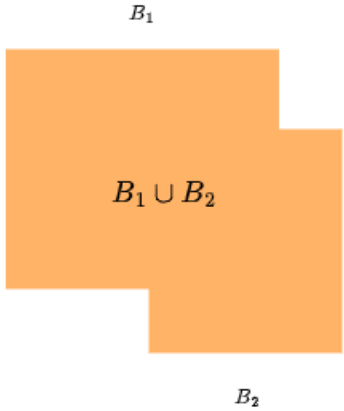
3.4 Experiment

3.4.1 Datasets

To validate the great performance of SFPN, both CUB-200-2011 [50] and ILSVRC [41] are used. CUB-200-2011 is constructed by gathering images of 200 kinds of birds. It divides the images into 5,994 images for training data and 5,794 images for test data. Also, ILSVRC collects images of 1,000 kinds of the object. It has roughly 1.2–1.3 million training images and 50,000 validation images.



(a) Intersection between B_1 and B_2



(b) Union of B_1 and B_2

Figure 3.6: An example of intersection and union of two bounding boxes. For given bounding boxes B_1 and B_2, (a) shows the intersection between these bounding boxes, and (b) shows the union of these bounding boxes.

3.4.2 Evaluation Metrics

In evaluation, we check two factors: the class and the bounding box. First, the predicted class is regarded as right if it is equal to the ground truth class. Next, Intersection over Union (IoU) is used for the bounding box. For a more detailed discussion, we let $B_{x_1,x_2,y_1,y_2} = [x_1, x_2] \times [y_1, y_2] \subseteq \mathbb{R}^2$ for $x_1, x_2, y_1, y_2 \in \mathbb{R}$ and $\mathcal{B} = \{B_{x_1,x_2,y_1,y_2} : x_1, x_2, y_1, y_2 \in \mathbb{R},\ x_1 \leq x_2,\ y_1 \leq y_2\}$. Then we define $\mathrm{IoU} : \mathcal{B} \times \mathcal{B} \to [0, 1]$ is defined by

$$\mathrm{IoU}(B_1, B_2) = \frac{\mathrm{area}(B_1 \cap B_2)}{\mathrm{area}(B_1 \cup B_2)}, \tag{3.34}$$

where $B_1 \in \mathcal{B}$ and $B_2 \in \mathcal{B}$ are bounding boxes and $\mathrm{area} : \mathcal{B} \to \mathbb{R}_{\geq 0}$ outputs the area of given bounding box. For a visual example, refer to Figure 3.6. Thus, the larger IoU between two bounding boxes, the closer these bounding boxes are. Hence, the estimated bounding box B is regarded as right if $\mathrm{IoU}(B, G) \geq 0.5$, where G is the corresponding ground truth bounding box.

With the above argument, we define Top1-cls and Top1-loc as follows. Let $D = \{(x_i, y_i, G_i) : 1 \leq i \leq n\}$ be a dataset, where x_i is an image, y_i is the ground truth class, and G_i is the ground truth bounding box. Also, let y_i' be

the predicted class and B_i be the estimated bounding box from the model for each i. Then Top1-cls and Top1-loc are defined by

$$\text{Top1-cls} = \frac{\sum_{i=1}^{n} \mathbf{1}_{y_i'=y_i}}{n} \times 100, \tag{3.35}$$

$$\text{Top1-loc} = \frac{\sum_{i=1}^{n} \left[\mathbf{1}_{y_i'=y_i} \wedge \mathbf{1}_{\text{IoU}(B_i,G_i)\geq 0.5}\right]}{n} \times 100, \tag{3.36}$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function, and $\wedge$ is the logical AND operator. In other words, Top1-cls shows the proportion of images with rightly estimated class in the whole dataset, and Top1-loc shows the proportion of images with rightly estimated class and bounding box in the whole dataset. Thus, these two metrics will give quantitative values for SFPN.

3.4.3 Implementation Details

The initial weight of SFPN is pretrained on ILSVRC. Also, we optimize SFPN with SGD optimizer of 0.9 momentum, $5e-4$ weight decay, and $5e-3$ initial learning rate. Here, 1×1 convolution layer in the final predictor uses $5e-2$ learning rate, while other layers have a tenfold less learning rate. Also, we utilize only cross-entropy loss. When we train SFPN on CUB-200-2011, we use 32 as batch size. Also, when we train SFPN on ILSVRC, we use 128 as batch size and freeze layers in stage 1–4 and optimize other layers. Also, SFPN is implemented using PyTorch [37].

3.4.4 Result

In this subsection, we draw a comparison between the proposed method and existing WSOL methods. For this comparison, the highest Top1-loc performances reported in their paper are used. Table 3.2–3.3 show the result for this comparison.

In CUB-200-2011, the proposed method shows higher Top1-cls and Top1-loc than methods that use complex loss functions [31, 33, 34, 55, 60, 61]. Moreover, the proposed method shows an increase of 1.12 percentage points in

Method	Top1-loc (%)	Top1-cls (%)
CAM [63]	41.00	63.00
ACoL [60]	45.92	71.90
SPG [61]	46.64	-
CSTN [34]	49.03	78.46
PECA-Net [31]	51.68	-
Cutmix [57]	54.81	-
MEIL [33]	57.46	74.77
DANet [55]	61.10	81.60
ADL [5]	62.29	80.34
SFPN [22]	**63.41**	**81.74**

Table 3.2: Comparison of WSOL performances for CUB-200-2011. The best performance is denoted by bold letters. Also, - means the corresponding paper does not report the result.

Top1-loc and 1.40 percentage points in Top1-cls compared to ADL [5], which is the best method except for the proposed method.

Comparison in ILSVRC also follows this tendency. The proposed method still gains higher performance than methods that use complex loss functions [31, 33, 34, 55, 60, 61]. Furthermore, the proposed method reports the improved performance with 0.10 percentage points increase in Top1-loc and 0.57 percentage points increase in Top1-cls, compared to MEIL [33]. Quantitative comparisons in two datasets validate that the proposed method attains great performance using only cross-entropy loss.

Figure 3.7–3.8 show a qualitative comparison between CAM and the proposed method. The ground truth bounding box is represented by the green bounding box, and the estimated bounding box is described as the red bounding box. The proposed method recognizes a region closer to the whole region of the object compared to CAM. Thus, it estimates the more accurate bounding box.

Method	Top1-loc (%)	Top1-cls (%)
CSTN [34]	42.38	69.48
CAM [63]	43.60	65.00
HaS [44]	45.21	70.70
ACoL [60]	45.83	71.00
Cutmix [57]	47.25	-
DANet [55]	47.53	72.50
PECA-Net [31]	47.93	-
SPG [61]	48.60	-
ADL [5]	48.71	72.83
MEIL [33]	49.48	73.88
SFPN [22]	**49.58**	**74.45**

Table 3.3: Comparison of WSOL performances for ILSVRC. The best performance is denoted by bold letters. Also, - means the corresponding paper does not report the result.

3.4.5 Ablation Study

SFPN aims to help the last feature map have more plentiful information. To see this effect, we examine the feature maps U_1, U_2, U_3, and U_4, which are components of the last feature map. In this study, we use CUB-200-2011. Table 3.4 shows the result of this study. When we use U_4 as the last feature map, the worst Top1-loc is shown. This may come from losing the information because U_4 has 256 as the number of channels and S_4 has 2048 as that of channels. Also, as adding U_3, U_2, and U_1 gradually, we have more improved performance in Top1-loc.

More precisely, adding U_3 (the third row of Table 3.4) gives 4.23 increased percentage points in Top1-loc compared to using only U_4 (the second row of Table 3.4). Also, adding U_2 (the 4th row of Table 3.4) gains 0.3 increased percentage points in Top1-loc compared to using U_3 and U_4 (the third row of Table 3.4). Finally, SFPN (the last row of Table 3.4) gets the best Top1-loc performance with an increase of 1.29 percentage points compared to using U_2, U_3, and U_4 (the 4th row of Table 3.4). This means that enhanced information of the last feature map helps improve WSOL performance.

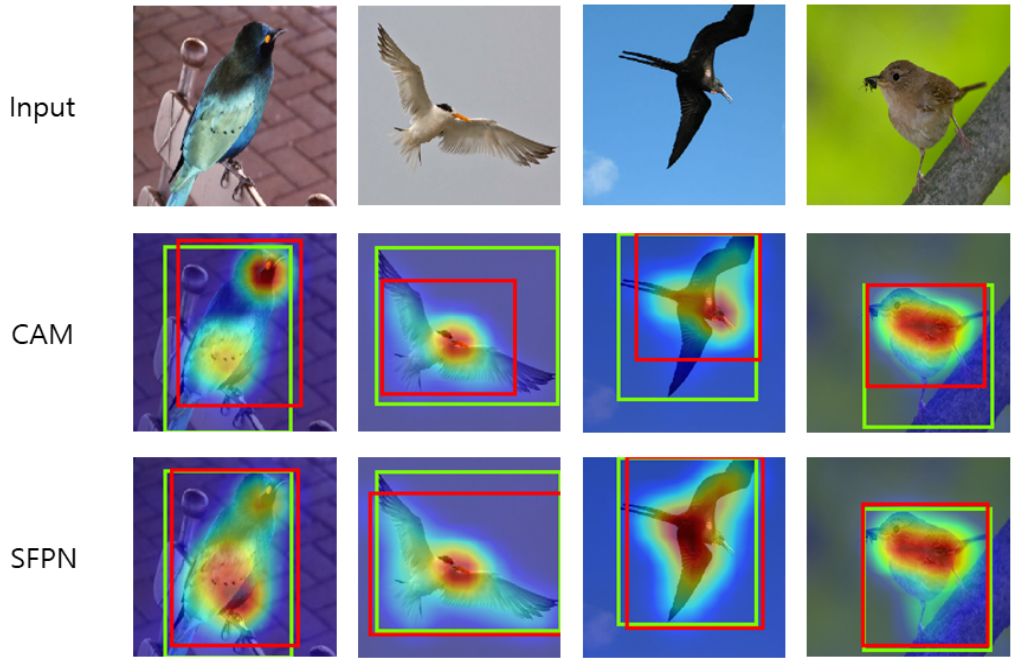


Figure 3.7: Comparison with CAM on CUB-200-2011. The ground truth bounding box is represented by the green bounding box, and the estimated bounding box is described as the red bounding box.

Used feature maps	Top1-loc (%)	Top1-cls (%)
S_4	58.89	**82.24**
U_4	57.59	82.02
U_3, U_4	61.82	81.17
U_2, U_3, U_4	62.12	81.29
U_1, U_2, U_3, U_4 (SFPN [22])	**63.41**	81.74

Table 3.4: Ablation study.

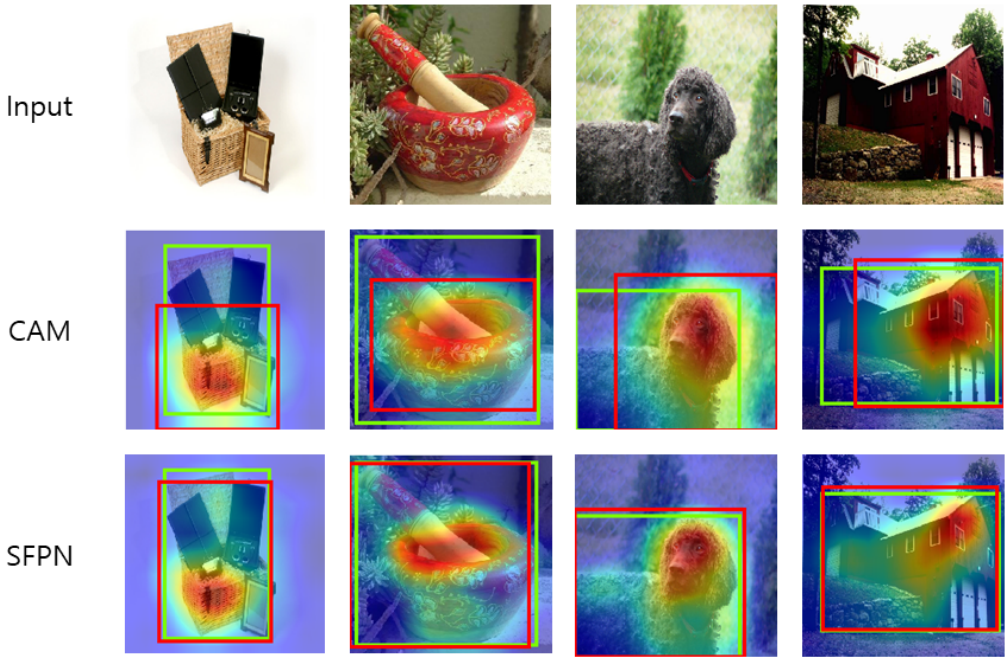


Figure 3.8: Comparison with CAM on ILSVRC. The ground truth bounding box is represented by the green bounding box, and the estimated bounding box is described as the red bounding box.

Chapter 4

A2E Net: Aggregation of Attention and Erasing for Weakly Supervised Object Localization

4.1 Introduction

Although SFPN improves the localization, it needs a nonnegligible number of additional learnable parameters. Indeed, it uses ∼3.8M weights additionally. Since a small number of learnable weights can decrease the burden of learning, we focus on reducing the number of learnable weights in this chapter. Also, we need to prevent the model from having a highly dropped performance caused by the reduction of such parameters.

For this goal, attention mechanism and erasing can be the keys. First, attention mechanism emphasizes the important information, which is inspired from the human visual system [6, 19]. This has demonstrated its effectiveness for many computer vision tasks, including object detection. SE [17] and CBAM [54] are examples for this.

Next, as we discussed in Section 3.2, erasing is one of the most commonly used techniques in WSOL. Thus, one can consider the use of attention mechanism and erasing for WSOL. Indeed, ADL [5], which was introduced in

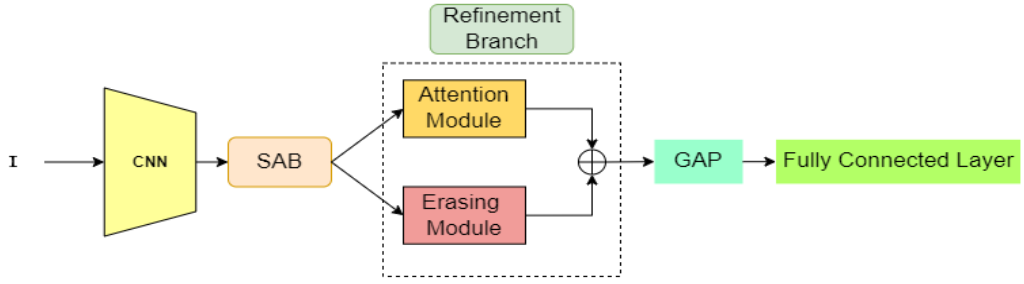


Figure 4.1: The overall architecture of the proposed method. Here, SAB denotes spatial attention branch and GAP is global average pooling. Also, $\oplus$ denotes elementwise sum.

Section 3.2, is an example of realizing this idea and has no trainable weights. More explicitly, ADL picks one architecture from attention mechanism and erasing at random for each iteration. However, such a choice sometimes does not exploit the virtues of attention mechanism and erasing fully, due to the randomness.

Considering the above points, we propose a new model named A2E Net [23] that is lightweight and employs attention mechanism and erasing to maintain the performance. This model is made up of two branches; spatial attention branch and refinement branch. First, spatial attention branch helps the feature map heighten its information. Next, this heightened feature map become the input of refinement branch. Refinement branch makes the information of the input feature map more elaborate and considers the less discriminative region by erasing. The overall flow of A2E Net is described in Figure 4.1.

The rest of this chapter is organized as follows. First, to explain the background for A2E Net, we show some attention mechanisms, some erasing methods, and existing WSOL models in Section 4.2. Then we show the detail and the results of A2E Net in the remaining parts.

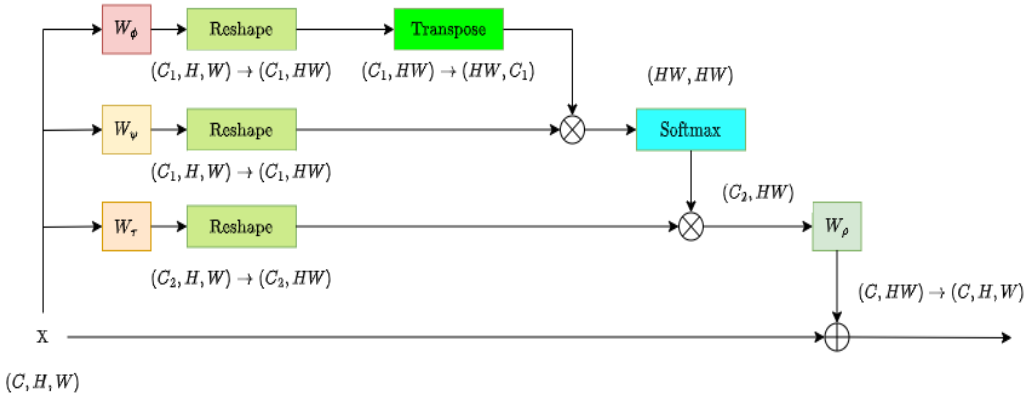


Figure 4.2: An example of NL-network [53]. In this figure, an operation that computes connection between pixels is softmax.

4.2 Related Works

4.2.1 Attention Mechanism

Non-local Neural Network

Non-local algorithm [4] was introduced for image denoising. This algorithm considers similarities between neighborhoods. More precisely, this approach aims to remove noise from a noisy image u by Wu, where I is an index set for pixels, $W = [w_{ij}] \in \mathbb{R}^{|I|\times|I|}$, and $0 \leq w_{ij} \leq 1, \sum_{j\in I} w_{ij} = 1$. In practical, w_{ij} was defined by

$$w_{ij} = \frac{1}{C(i)} \exp\left(-\frac{\|N(u_i) - N(u_j)\|_2^2}{h^2}\right), \tag{4.1}$$

where u_i is the ith pixel of u, $N(u_i)$ is a neighborhood of u_i, h is a parameter, and $C(i)$ is a normalization factor.

Non-local neural network (NL-network) [53] reformulates this algorithm. Namely, NL-network parametrizes Non-local algorithm by neural networks. An application of NL-network to 2D images gives a more detailed description.

Let $X \in \mathbb{R}^{C\times H\times W}$ be the input feature map of NL-network. Here, reshaping sometimes makes it possible to identify $X \in \mathbb{R}^{C\times H\times W}$ as $X \in \mathbb{R}^{C\times HW}$,

where reshaping means rearranging values of a given tensor. Under this identification, let $x_i \in \mathbb{R}^C$ be the ith column (pixel) of X for $1 \leq i \leq HW$. Also, let $W_\phi, W_\psi : \mathbb{R}^{C\times H\times W} \to \mathbb{R}^{C_1\times H\times W}$, $W_\tau : \mathbb{R}^{C\times H\times W} \to \mathbb{R}^{C_2\times H\times W}$, and $W_\rho : \mathbb{R}^{C_2\times H\times W} \to \mathbb{R}^{C\times H\times W}$ be 1×1 convolution layers with learnable parameters. Then $W_\phi X, W_\psi X \in \mathbb{R}^{C_1\times H\times W}$ and $W_\tau X \in \mathbb{R}^{C_2\times H\times W}$. Under reshaping, we get $W_\phi X, W_\psi X \in \mathbb{R}^{C_1\times HW}$ and $W_\tau X \in \mathbb{R}^{C_2\times HW}$.

Next, NL-network uses an operation to $(W_\phi X)^T(W_\psi X) \in \mathbb{R}^{HW\times HW}$ to get the weight for the connection between pixels. Examples of such operations contain softmax and a dot product. Such an operation is applied to $(W_\phi X)^T(W_\psi X)$, called $\mathrm{act}((W_\phi X)^T(W_\psi X))$. Here, when the operation is softmax, we have

$$[\mathrm{softmax}((W_\phi X)^T(W_\psi X))]_{ij} = \frac{\exp\big((W_\phi x_i)^T W_\psi x_j\big)}{\sum_{r=1}^{HW} \exp((W_\phi x_r)^T W_\psi x_j)}. \tag{4.2}$$

Note that with $C(j) := \sum_{r=1}^{HW} \exp\big((W_\phi x_r)^T W_\psi x_j\big)$, we have

$$[\mathrm{softmax}((W_\phi X)^T(W_\psi X))]_{ij} = \frac{1}{C(j)} \exp\big((W_\phi x_i)^T W_\psi x_j\big), \tag{4.3}$$

which corresponds to Equation 4.1. Other operations can also be calculated as similar to softmax.

Then we use matrix multiplication of $\mathrm{act}((W_\phi X)^T(W_\psi X))$ and $W_\tau X$. Finally, W_ρ changes the number of channels of $(W_\tau X)\mathrm{act}((W_\phi X)^T(W_\psi X))$ from C_2 to C, and the output feature map is $X + W_\rho(W_\tau X)\mathrm{act}((W_\phi X)^T(W_\psi X))$, after we regard $W_\rho(W_\tau X)\mathrm{act}((W_\phi X)^T(W_\psi X))$ as a tensor of shape (C, H, W) using reshaping. This process when act = softmax is described in Figure 4.2. The above process can be also used for video data by modifying architectures, e.g., converting 1×1 convolution into $1\times 1\times 1$ convolution.

NL-network shows its great performance for video classification, object detection, instance segmentation, and keypoint detection. Moreover, this network gives inspiration to other studies. For example, SAGAN [59] exploits NL-network for image generation. In SAGAN, NL-network helps GAN [13]

not depend only on the local information. With a similar virtue, NL-network is also applied to image restoration [62] and scene segmentation [8].

Other Networks

As CNN is improving, attention mechanism has provided a way to further boost the performance for various computer vision tasks with CNN. Many attention networks are plugged into CNN and use the combination of transformation and the input feature map. For the convenience of discussion, we let $X \in \mathbb{R}^{C\times H\times W}$ be the input feature map, $\odot$ be the elementwise product, and σ be a sigmoid.

RAN [51] obtains a mask from the network, which is called "soft mask branch". The encoder part of this network includes max pooling and residual network to extract global information. Then the decoder makes a feature map with a resolution higher than that of the output feature map of the encoder by interpolation. Finally, sigmoid makes the range of $[0, 1]$ for the output of the decoder. With "trunk branch" in RAN, RAN has the form of

$$X \mapsto (I + M(X)) \odot T(X), \tag{4.4}$$

where I is the tensor filled with 1, and soft mask branch and trunk branch give $M(X)$ and $T(X)$, respectively. Through this, RAN improves image classification performance and shows its flexibility for various CNNs.

SE [17] constructs a network that formulates the connection between channels. The connection between channels is complex because 2D convolution outputs the feature map by using spatial operation. So, to be more discriminative to the usefulness of information, SE first gets information for each channel and then catches a connection between channels. For the first step, SE uses GAP, namely,

$$\frac{1}{HW}\sum_{i,j} X_{c,i,j} \tag{4.5}$$

for each channel c. Then the neural network composed of fully connected layer, ReLU, and sigmoid is used to construct a network that catches a connection

between channels. Finally, SE has the form of

$$X \mapsto X \odot \underbrace{(\sigma \circ W_2 \circ \mathrm{ReLU} \circ W_1 \circ \mathrm{GAP})(X)}_{\text{transformation}}, \tag{4.6}$$

where r is a reduction factor which controls the number of parameters and $W_1 : \mathbb{R}^C \to \mathbb{R}^{C/r}$ and $W_2 : \mathbb{R}^{C/r} \to \mathbb{R}^C$ are fully connected layers. Also, a range of values of tensors is changed into $[0, 1]$ by σ.

BAM [35] consists of two branches, which are named "channel attention branch" and "spatial attention branch", respectively. Like SE, channel attention branch also catches a connection between channels. So, this branch uses a similar network to SE with a reduction factor r.

Spatial attention branch uses 1×1 convolution layers to control the number of channels and 3×3 dilated convolution layers to allow the larger receptive field. Namely,

$$X \mapsto (\mathrm{Conv}_{1\times1}^{C/r\to C} \circ \mathrm{Conv}_{3\times3,d}^{C/r\to C/r} \circ \mathrm{Conv}_{3\times3,d}^{C/r\to C/r} \circ \mathrm{Conv}_{1\times1}^{C\to C/r})(X), \tag{4.7}$$

where $\mathrm{Conv}_{1\times1}^{A\to B}$ is a 1×1 convolution layer which changes the number of channels from A to B and $\mathrm{Conv}_{3\times3,d}^{A\to B}$ is a 3×3 convolution layer with dilation rate d which changes the number of channels from A to B. Finally, BAM has the form of

$$X \mapsto X \odot \underbrace{(I + \sigma \circ (\mathrm{Channel} + \mathrm{Spatial})(X))}_{\text{transformation}}, \tag{4.8}$$

where I denotes the tensor filled with 1 of the same shape as X, and 'Channel' and 'Spatial' denote channel attention branch and spatial attention branch, respectively.

CBAM [54] also uses two modules, similar to BAM. In other words, two modules, which are named "channel attention module" and "spatial attention module", respectively, are components of CBAM. Channel attention module in CBAM not only uses GAP but also uses global maximum pooling, while SE and BAM only use GAP for channel attention. It first generates two tensors by such pooling layers. Then a neural network similar to SE outputs two tensors corresponding to these tensors, and the summation of these output

tensors gives the output tensor of channel attention module. Namely,

$$X \mapsto \sigma \circ (W_2 \circ \mathrm{ReLU} \circ W_1 \circ \mathrm{GAP} + W_2 \circ \mathrm{ReLU} \circ W_1 \circ \mathrm{GMP})(X), \quad (4.9)$$

where GMP is global maximum pooling, r is a reduction factor, and $W_1 : \mathbb{R}^C \to \mathbb{R}^{C/r}$ and $W_2 : \mathbb{R}^{C/r} \to \mathbb{R}^C$ are fully connected layers.

Spatial attention module also has two pooling operations: CAP and channel maximum pooling. It first makes two tensors by these pooling layers. Then these two tensors are concatenated and passed through a 7×7 convolution layer. Namely,

$$X \mapsto \sigma \circ \mathrm{Conv}_{7\times 7}^{2\to 1} \circ \mathrm{Concat}[\mathrm{CAP}(X), \mathrm{CMP}(X)], \quad (4.10)$$

where CMP is channel maximum pooling, 'Concat' is concatenation, and $\mathrm{Conv}_{7\times 7}^{2\to 1}$ is a 7×7 convolution layer which changes the number of channels from 2 to 1. Finally, CBAM outputs

$$X \mapsto X \odot \underbrace{(\mathrm{Spatial}(\mathrm{Channel}(X) \odot X) \odot \mathrm{Channel}(X))}_{\text{transformation}}, \quad (4.11)$$

where 'Spatial' and 'Channel' are spatial attention module and channel attention module, respectively.

ECA-Net [52] introduces a more efficient algorithm by paying attention to channel attention. It first observes that immediate connection between channels and weights helps the network boost performance. Thus, a reduction factor r is not used for ECA-Net. Also, ECA-Net finds that the relationship between channels is one of the factors that improve performance. From these observations, ECA-Net describes the relationship between channels in the neighborhood by a 1D convolution. Namely,

$$X \mapsto X \odot \underbrace{(\sigma \circ \mathrm{Conv}_{1D,k} \circ \mathrm{GAP}(X))}_{\text{transformation}}, \quad (4.12)$$

where '$\mathrm{Conv}_{1D,k}$' is a 1D convolution having k as its kernel size. Here, k is selected by the number of channels of X.

In the above models, we can observe that many attention mechanisms

have the form of

$$X \mapsto X \odot \text{transformation}(X), \tag{4.13}$$

where 'transformation' usually consists of neural networks. This structure inspires spatial attention branch in A2E Net.

4.2.2 Erasing Methods

Similar to the previous section, we still use the same notations. In other words, let $X \in \mathbb{R}^{C \times H \times W}$ be the input feature map, where C indicates the number of channels, and H and W are the height and the width, respectively. Also, let $\odot$ be the elementwise product.

Dropout [45] is one of the methods that avoid overfitting. This method randomly annihilates neurons in a fully connected layer and pixels in a convolutional feature map. Namely, Dropout outputs $M_p \odot X$, where p is a probability that neuron or pixel will not be removed, M_p is a binary mask with the same shape as X, and

$$M_p \sim \text{Bernoulli}(p). \tag{4.14}$$

Here, Bernoulli distribution for M_p is independently distributed for each neuron or pixel.

Dropout gains the increased performance when applied to a fully connected layer because it generates and learns many networks. However, when applied to a convolutional feature map, its use does not help the improvement of performance. The connection between pixels is one of the main reasons that cause this. More explicitly, pixels in the neighborhood usually have similar information [9, 48].

To increase the effect of erasing for a convolutional feature map, SpatialDropout [48] removes the whole pixels in the randomly chosen channels. Namely, SpatialDropout returns $M_p \odot X$, where p is a probability that pixel will not be removed, and M_p is a binary mask with the same shape as X and

$$M_p[c] \sim \text{Bernouill}(p) \tag{4.15}$$

for each channel c. Here, for each channel c, either $M_p[c] = \mathbf{1} \in \mathbb{R}^{H\times W}$ or $M_p[c] = \mathbf{0} \in \mathbb{R}^{H\times W}$, where $\mathbf{0}$ and $\mathbf{1}$ are the tensor filled with zeros and the tensor filled with ones, respectively.

On the other hand, MaxDrop [36] introduces two approaches. One is to handle the probability of Dropout. It is called stochastic dropout. To give various simulations, stochastic dropout samples the probability of Dropout from the uniform distribution or Gaussian distribution. The other is to focus on the pixel that obtains the maximum value in the channel axis or the position axis. When MaxDrop uses erasing in the channel axis, it first picks the pixels at random and removes the location at which the maximum value is obtained corresponding to these chosen pixels. Namely, MaxDrop with the channel axis gets

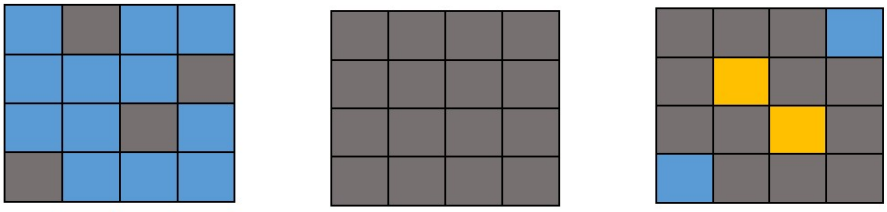
$$i_c := \underset{1\leq c\leq C}{\operatorname{argmax}} X[c,h,w] \Longrightarrow X[i_c,h,w] = 0, \tag{4.16}$$

where (h,w) is the randomly chosen pixel. Similarly, when MaxDrop uses erasing in the position axis, it chooses the channels randomly and erases the pixel that obtains the maximum value. Namely, in this case,

$$i_h, i_w := \underset{h,w}{\operatorname{argmax}} X[c,h,w] \Longrightarrow X[c,i_h,i_w] = 0, \tag{4.17}$$

where c is the randomly chosen channel. This method is similar to erasing module in A2E Net in the perspective of erasing the pixels around the point at which the maximum value is given. However, the erasing module does not choose the channels randomly.

Dropblock [9] provides a different way. From the above discussed property of pixels in the neighborhood [9, 48], Dropblock tries to avoid removing some pixels, leaving other pixels in the neighborhood because this can leave information. Thus, it generates rectangles and removes all pixels in these rectangles. For this, it needs the probability p of not removing the pixel and the size b of these rectangles. Also, it picks the pixels at random to get the centers of these rectangles. More explicitly, it generates a binary mask $M \in \mathbb{R}^{C\times H\times W}$



(a) Dropout (b) SpatialDropout (c) Dropblock

Figure 4.3: Description of Dropout, SpatialDropout, and Dropblock when the number of channels is 1. The blue denotes the pixel that is not removed, and the gray denotes the erased pixel. Also, in (c), the yellow is the center of the rectangle.

with

$$M \sim \text{Bernoulli}(\gamma), \tag{4.18}$$

and

$$\gamma = \frac{1-p}{b^2}\frac{HW}{(H-b+1)(W-b+1)}. \tag{4.19}$$

Then Dropblock regards the zero positions of M as the centers of the rectangles. Next, all pixels in the rectangles with these centers and the size of b have zero values. With this update for M, Dropblock outputs $X \odot M \odot \frac{CHW}{\sum_{i,j} M_{ij}}$. In this process, the distribution of 'keeping' pixels is not determined independently, in contrast to Dropout. Indeed, the distribution corresponding to Dropblock is more regionally defined. Thus, Dropblock becomes a more suitable approach to increase the effect of erasing for a convolutional feature map. To give an easier understanding, the above erasing methods except for MaxDrop are described in Figure 4.3.

Erasing module in A2E Net also removes all pixels in the rectangles, similar to Dropblock. However, we regard only the pixels at which the maximum value is given as the centers of the rectangles. It is because our erasing module aims to remove the most discriminative region. In other words, since the

most discriminative region often contains the pixel of the maximum value, we do not have to regard other pixels as the center of the rectangle. Also, we pick the size of the rectangles in a range of sizes at random to further boost performance, which is also different from Dropblock.

4.2.3 Existing Methods for Weakly Supervised Object Localization

InCA [21] and DGDM [56] use the similar idea to ADL. InCA also picks one architecture between attention and erasing at random. However, InCA removes pixels greater than the foreground threshold and pixels lower than the background threshold to reduce a prediction of a larger region than the region of the object. Also, InCA introduces "contrastive attention loss" and "consistency loss" to help the extension of the attention map and make layers have similar coherence, respectively.

DGDM proposes two architectures with erasing, which have channel direction and spatial direction, respectively. Channel direction in DGDM first applies GAP and leaves channels with high values. Also, some channels with low values are sometimes left randomly. This channel direction aims to construct the connection between channels.

On the other hand, spatial direction in DGDM picks one architecture between attention and erasing at random. Here, erasing in spatial direction removes not only the most discriminative region but also the background, making the rectangles to remove the pixels in the neighborhood. A2E Net also uses rectangles for erasing. However, A2E Net removes the pixels in the rectangles with a randomly selected size, while DGDM uses rectangles with a fixed size.

HCLNet [3] addresses the less discriminative region by an approach inspired from ACoL. It inverts the feature map, instead of using a fixed threshold for erasing. More explicitly, HCLNet uses two classifiers, similar to ACoL. Then for the output feature map f of CNN, one classifier makes CAM as usual, and another classifier takes the feature map $(1 - \mathrm{CAM}) \odot f$ as input. This process produces two feature maps, called f_1 and f_2, which are combined to get the final localization. Also, HCLNet introduces two methods for this

combination: addition strategy and l_1-norm strategy. The addition strategy makes the output feature map F by $F = f_1 + f_2$. The l_1-norm strategy is a more complex method. First, this method gets the feature map M_i ($i \in \{1, 2\}$) by

$$M_i(x, y) = \|f_i(x, y)\|_1 \tag{4.20}$$

for each (x, y). Then for each (x, y), the average operation on its neighborhood is used by

$$\hat{M}_i(x, y) = \frac{\sum_{a=-r}^{r} \sum_{b=-r}^{r} M_i(x + a, y + b)}{(2r + 1)^2}. \tag{4.21}$$

Finally, the normalizing factors w_i come from

$$w_i(x, y) = \frac{\hat{M}_i(x, y)}{\hat{M}_1(x, y) + \hat{M}_2(x, y)}, \tag{4.22}$$

and the resulting feature map is

$$F(x, y) = w_1(x, y) f_1(x, y) + w_2(x, y) f_2(x, y). \tag{4.23}$$

Also, HCLNet has two cross-entropy loss functions corresponding to the two classifiers, similar to ACoL.

Aside from the above models, there are interesting approaches to WSOL. For instance, CSoA [24] uses a "confidence segmentation (ConfSeg) module" to make mask and "co-supervised augmentation (CoAug) module" for regularization. First, in ConfSeg, CSoA provides two CAMs, called S_F and S_L, by adding a new classifier. Here, let H_L and W_L be the height and the width of S_L, respectively. Then a mask is made M from S_L by

$$M_{i,j} = |(S_L^c)_{i,j} - \mu_1|, \tag{4.24}$$

where S_L^c is the cth feature map of S_L and $\mu_1 = \dfrac{\sum_{i,j}(S_L^c)_{i,j}}{H_L \times W_L}$. If $(S_L^c)_{i,j}$ is similar to μ_1, it is difficult to classify this into foreground or background. On the other hand, if $(S_L^c)_{i,j}$ has a large distance from μ_1, then it is potentially in the foreground or background. Thus, M shows the confidence for each pixel.

Based on M, to apply a threshold per batch, M is converted into

$$\hat{M}_{i,j} = \begin{cases} 1, & M_{i,j} > \mu_2, \\ 0, & \text{otherwise}, \end{cases} \tag{4.25}$$

where $\mu_2 = \dfrac{\sum_{i,j} M_{i,j}}{H_L \times W_L}$. In this equation, $\hat{M}$ leaves the pixels with high confidence. On these pixels, ConfSeg module makes the high similarity between S_F^c and S_L^c by minimizing

$$\mathcal{L}_{\text{inner}} := \sum_{i,j} |(S_F^c)_{i,j}^c - (S_L^c)_{i,j}^c| \odot \hat{M}_{i,j}. \tag{4.26}$$

Therefore, the loss function $\mathcal{L}_C$ for ConfSeg module is

$$\mathcal{L}_C := \mathcal{L}_{\text{cls}} + \alpha \mathcal{L}_{\text{inner}}, \tag{4.27}$$

where $\mathcal{L}_{\text{cls}}$ is the classification loss function for two CAMs and $\alpha \in [0, 1]$ is controlled in training phase.

To further boost performance, CSoA uses CoAug. CoAug first makes a large difference between foreground and background. To this end, when I_m is the mth image or feature map with the class c_m and f_θ is the network such as CNN, CoAug produces two feature maps F_m and B_m by

$$F_m = f_\theta(S_L^{c_m} \odot I_m), \quad B_m = f_\theta((1 - S_L^{c_m}) \odot I_m). \tag{4.28}$$

In other words, F_m and B_m correspond to the foreground feature map and background feature map, respectively. Then CoAug maximizes

$$D_m^{\text{cam}} = \|F_m - B_m\|_2, \tag{4.29}$$

and minimizes

$$D_m^{\text{back}} = \|B_m - M_B^m\|_2 \tag{4.30}$$

to reduce the wrong classification for the foreground, where $M_B^m = f_\theta((1 - S_L^{c_m}) \odot \hat{M} \odot I_m)$.

Moreover, CoAug aims to make the high similarity between F_m and F_n when these feature maps have the same class, and wants a large gap between F_m and F_n when these feature maps have different classes. To this end, CoAug defines the following distance

$$D_{m,n} = \|F_m - F_n\|_2 \tag{4.31}$$

and the final loss $\mathcal{L}_D$ by

$$\mathcal{L}_D^{\text{same}} = \sum_{\{m,n \mid c_m \neq c_n\}} \frac{\gamma(D_m^{\text{back}} + D_n^{\text{back}})}{\delta D_{m,n} + \frac{1}{2}(D_m^{\text{cam}} + D_n^{\text{cam}})}, \tag{4.32}$$

$$\mathcal{L}_D^{\text{diff}} = \sum_{\{m,n \mid c_m = c_n\}} \frac{D_{m,n} + \gamma(D_m^{\text{back}} + D_n^{\text{back}})}{\frac{1}{2}(D_m^{\text{cam}} + D_n^{\text{cam}})}, \tag{4.33}$$

$$\mathcal{L}_D = \mathcal{L}_D^{\text{same}} + \mathcal{L}_D^{\text{diff}}. \tag{4.34}$$

Finally, CSoA is trained with $\mathcal{L}_C$ and $\mathcal{L}_D$.

Another interesting approach is EGA [2]. EGA uses adversarial images to get better representation for localization. So, it makes adversarial images corresponding to given input images. To do this, EGA uses the similar method to Madry et al. [32]. Namely, EGA gets adversarial image x by

$$x \leftarrow P[x + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(\theta, x, y))], \tag{4.35}$$

where P is a projection operator such as clipping, θ is the weights for the network, $\mathcal{L}$ is the loss function, y is the class, and

$$\text{sign}(x) = \begin{cases} 1, & x \geq 0, \\ 0, & x = 0, \\ -1, & x < 0 \end{cases} \tag{4.36}$$

Also, EGA considers the values of entropy. While the most discriminative region and the background usually have low entropy values, there are also pixels with high entropy values, which means that these pixels are ambiguous. Thus, EGA has the goal of removing this ambiguity by using

$$\mathcal{L}_{\text{entropy}}(\text{CAM}) = -\sum_{(h,w)} P_{\text{CAM}(h,w)} \log P_{\text{CAM}(h,w)}, \tag{4.37}$$

where the probability at the pixel (h, w) is $P_{\text{CAM}(h,w)}$. Finally, the total loss for EGA is

$$\mathcal{L}_{\text{clean,adv}} + \lambda_{\text{CAM}_{\text{clean}}} \mathcal{L}_{\text{entropy}}(\text{CAM}_{\text{clean}}) + \lambda_{\text{CAM}_{\text{adv}}} \mathcal{L}_{\text{entropy}}(\text{CAM}_{\text{adv}}), \tag{4.38}$$

where $\mathcal{L}_{\text{clean,adv}}$ is the loss function for clean images and adversarial images similar to [32], and $\text{CAM}_{\text{clean}}$ and CAM_{adv} are CAMs from clean image and adversarial image, respectively.

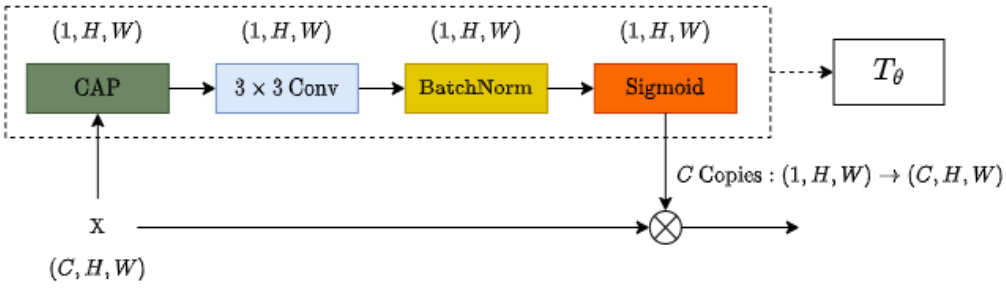


Figure 4.4: Spatial attention branch. CAP means channel average pooling and $\otimes$ is elementwise multiplication.

4.3 Proposed Method

4.3.1 Spatial Attention Branch

As we discussed in Section 4.2, many attention mechanisms make an attention map via a transformation that is composed of neural networks. Then they combine this attention map with the input feature map. This process can be described as follows:

Definition 4 (Attention mechanism)**.** *Let $X \in \mathbb{R}^{C\times H\times W}$ be a tensor.* ***Attention mechanism*** *$T : \mathbb{R}^{C\times H\times W} \to \mathbb{R}^{C\times H\times W}$ is defined by*

$$X \mapsto T(X) = T_\theta(X) \odot X, \tag{4.39}$$

where $T_\theta : \mathbb{R}^{C\times H\times W} \to \mathbb{R}^{C\times H\times W}$ is a transformation that is composed of neural networks with parameter θ and $\odot$ is an operation that combines the input tensor with $T_\theta(X)$.

Inspired by these mechanisms, we construct spatial attention branch (SAB) with a transformation that consists of CAP, 3×3 convolution, batch normalization, and sigmoid. We use our transformation as follows.

Let $X \in \mathbb{R}^{C\times H\times W}$ be the input feature map. Then CAP reduces the number of channels of X to 1 and extracts the representative values for each pixel. Next, we make an attention map by applying 3×3 convolution and batch normalization. After that, sigmoid converts a range of values of this

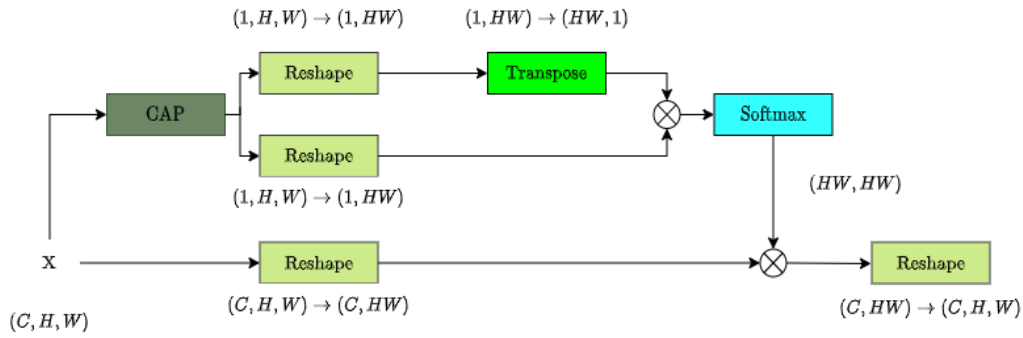


Figure 4.5: Attention module. CAP means channel average pooling and $\otimes$ is the matrix multiplication.

attention map into a range of $[0, 1]$. Note that the output feature map reflects the importance of each pixel. Finally, we multiply X and the output feature map of the transformation. We make C copies of values of this feature map when multiplication since the output feature map of the transformation has the number of channels as 1. The process for spatial attention branch is summarized in Figure 4.4.

Indeed, learnable weight in the transformation of SAB consists of convolution kernel in 3×3 convolution and parameters in batch normalization; thus, SAB has totally $3 \times 3 + 2 = 11$ as its number of learnable parameters. Hence, SAB requires a small number of learnable parameters.

4.3.2 Refinement Branch

We construct refinement branch to make the information of the feature map more elaborate and explore the less discriminative region. This branch is made up of attention module and erasing module.

Attention Module

As we discussed in Section 4.2, NL-network [53] uses the connection between pixels. This approach provides more elaborate information, so it is also applied to WSOL (e.g., InCA [21]). Inspired by this, we construct attention module. Unlike NL-network, attention module uses no additional learnable

parameters, so this module is very lightweight.

Attention module proceeds with the following details. Let $X \in \mathbb{R}^{C\times H\times W}$ be the input feature map. Then we get $\tilde{X} \in \mathbb{R}^{1\times H\times W}$ by using CAP from X. This can be written $\mathrm{CAP}(X) = \tilde{X} = [\tilde{x}_1, \cdots, \tilde{x}_{HW}] \in \mathbb{R}^{1\times HW}$ after reshaping. Then we have a matrix $B := (\tilde{X})^T\tilde{X} \in \mathbb{R}^{HW\times HW}$. Note that

$$B = (\tilde{X})^T\tilde{X} = \begin{bmatrix} \tilde{x}_1 \\ \vdots \\ \tilde{x}_{HW} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 & \cdots & \tilde{x}_{HW} \end{bmatrix} = \begin{bmatrix} \tilde{x}_1\tilde{x}_1 & \cdots & \tilde{x}_1\tilde{x}_{HW} \\ \vdots & \ddots & \vdots \\ \tilde{x}_{HW}\tilde{x}_1 & \cdots & \tilde{x}_{HW}\tilde{x}_{HW} \end{bmatrix} \tag{4.40}$$

and we can know that B represents the connection between pixels from this equation. Next, we use softmax for each column of B, denoted by $\mathrm{softmax}(B)$. Namely, we have

$$\mathrm{softmax}(B) = \begin{bmatrix} | & & | \\ \mathrm{softmax}(B_{:,1}) & \cdots & \mathrm{softmax}(B_{:,HW}) \\ | & & | \end{bmatrix} \tag{4.41}$$

$$= \begin{bmatrix} \dfrac{\exp(\tilde{x}_1\tilde{x}_1)}{\sum_{j=1}^{HW}\exp(\tilde{x}_j\tilde{x}_1)} & \cdots & \dfrac{\exp(\tilde{x}_1\tilde{x}_{HW})}{\sum_{j=1}^{HW}\exp(\tilde{x}_j\tilde{x}_{HW})} \\ \vdots & \ddots & \vdots \\ \dfrac{\exp(\tilde{x}_{HW}\tilde{x}_1)}{\sum_{j=1}^{HW}\exp(\tilde{x}_j\tilde{x}_1)} & \cdots & \dfrac{\exp(\tilde{x}_{HW}\tilde{x}_{HW})}{\sum_{j=1}^{HW}\exp(\tilde{x}_j\tilde{x}_{HW})} \end{bmatrix} = [b_{ij}], \tag{4.42}$$

where $B_{:,c}$ means the cth column of B and $b_{ij} = \dfrac{\exp(\tilde{x}_i\tilde{x}_j)}{\sum_{r=1}^{HW}\exp(\tilde{x}_r\tilde{x}_j)}$. Also, we reshape X into a tensor of shape (C, HW) and write

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,HW} \\ \vdots & \ddots & \vdots \\ x_{C,1} & \cdots & x_{C,HW} \end{bmatrix} \in \mathbb{R}^{C\times HW}. \tag{4.43}$$

Finally, we have $X \cdot \mathrm{softmax}(B) \in \mathbb{R}^{C \times HW}$, and it can be viewed as a tensor of (C, H, W) by reshaping. Here, for each channel c and each pixel j $(1 \leq c \leq C, 1 \leq j \leq HW)$, we have

$$[X \cdot \mathrm{softmax}(B)]_{cj} = \sum_{r=1}^{HW} x_{c,r} b_{rj}. \tag{4.44}$$

In Equation (4.44), when j is fixed, b_{rj} represents the connection to the rth pixel. Thus, $[X \cdot \mathrm{softmax}(B)]_{cj}$ is the weighted sum of the pixel values $x_{c,r}$ and the weight b_{rj}. Hence, attention module produces more elaborate information. The process for attention module is described in Figure 4.5. We summarize the above discussion and represent attention module as follows:

Definition 5 (Attention module)**.** *Let $X \in \mathbb{R}^{C \times H \times W}$. **Attention module** is defined by*

$$X \in \mathbb{R}^{C \times N} \mapsto X \cdot \mathrm{softmax}((CAP(X))^T CAP(X)) \in \mathbb{R}^{C \times N}, \tag{4.45}$$

where $N = HW$, we identify X as a tensor of shape (C, N) by reshaping, and $\mathrm{softmax}(A)$ *is a matrix that uses softmax operation for each column of a matrix $A \in \mathbb{R}^{N \times N}$.*

Erasing Module

As seen in Section 4.2, erasing is a useful tool to make the network recognize the less discriminative regions. To increase the effect of erasing, we focus on the observation that pixels in the neighborhood usually have similar information [9, 48]. Thus, removing pixels without considering this observation may leave pixels that have the information we want to erase.

To avoid this issue, we set a rectangle and remove all pixels in this rectangle. The flow of erasing module is as follows. Let $X \in \mathbb{R}^{C \times H \times W}$ be the input feature map. Then erasing module outputs $M \odot X$, where $M \in \mathbb{R}^{C \times H \times W}$ is a binary mask. When inference, we define a binary mask $M \in \mathbb{R}^{C \times H \times W}$ as the tensor filled with 1. Thus, we do not remove the pixels when inference.

When training, we set a rectangle for each channel. More explicitly, for

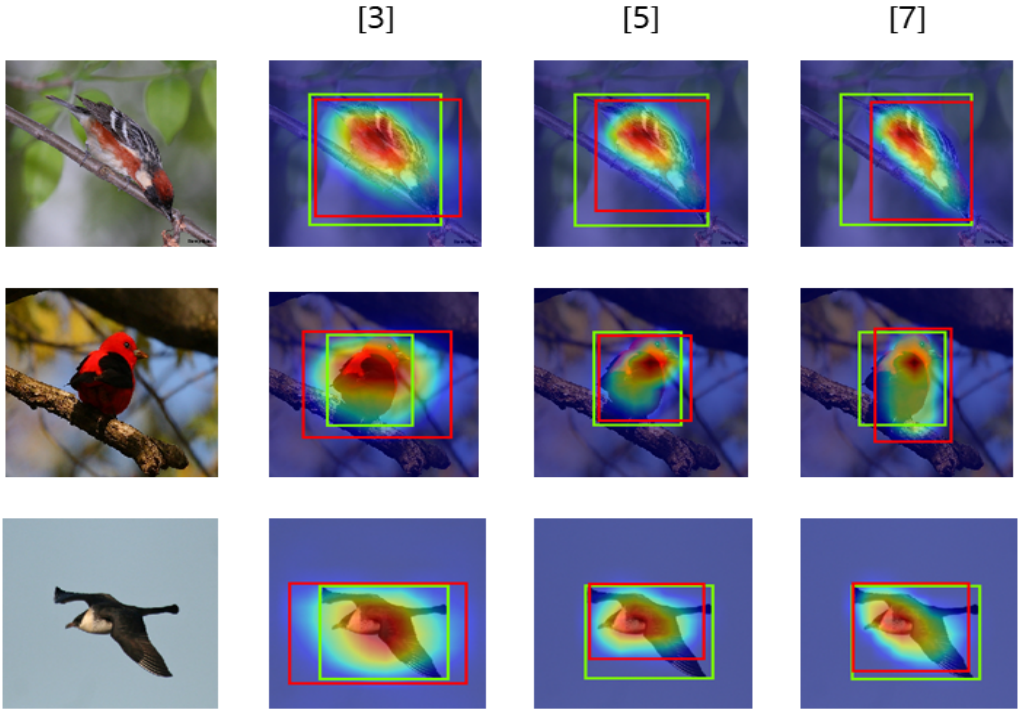


Figure 4.6: Comparison of sizes of rectangles. The ground truth bounding box is represented by the green bounding box, and the estimated bounding box is described as the red bounding box.

fixed channel c, we select the pixel of the maximum value in the cth feature map of X. The center of the rectangle for this channel will be this pixel because this pixel is often in the most discriminative region. For the size of the rectangle, one may use only one size. However, we observe that this use may restrict the increase in performance. For a visual example, see Figure 4.6.

Note that the first column of Figure 4.6 shows the input image, and the predictions of localization are displayed in other columns. For the predictions, we represent the ground truth bounding box and the estimated bounding box by the green bounding box and the red bounding box, respectively. Also, the numbers above the second to the 4th column mean that the used sizes for learning. So, for instance, the second column shows the results from the model using 3 as its size of erasing. With these settings, we first look at the first row

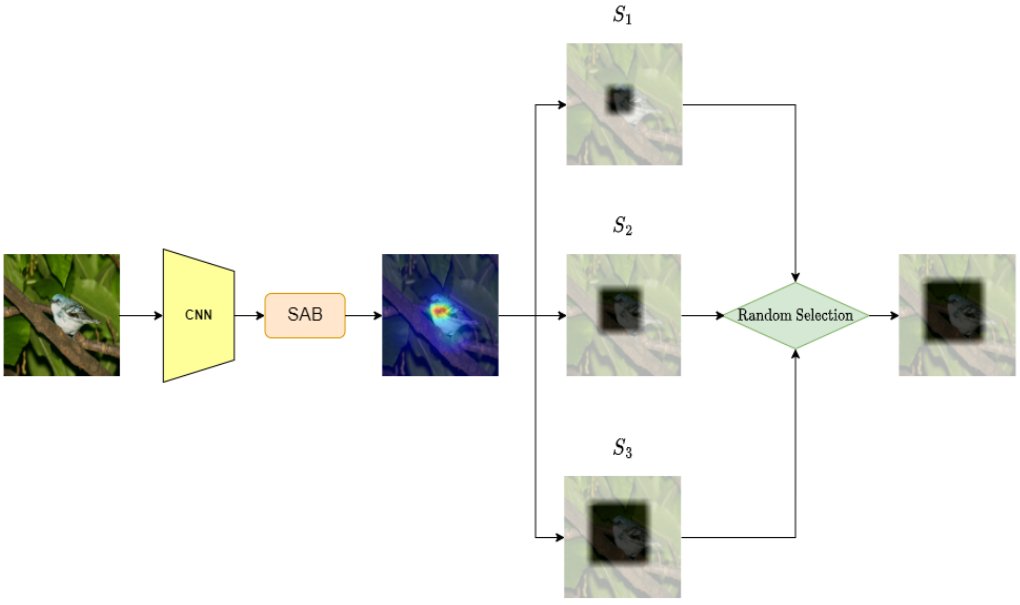


Figure 4.7: The procedure of erasing module for training. SAB denotes spatial attention branch, and S_1, S_2, and S_3 are the candidate sizes.

of Figure 4.6. In the first row, we observe that the size that gives the highest similarity to the ground truth bounding box is 3. However, the other images do not show the same results. In other words, 5 shows the highest similarity to the ground truth bounding box for the second row, and the best-estimated bounding box is obtained when we use 7 for the third row. Thus, we pick one size in a range of sizes at random to give the network a chance to access diverse sizes.

With the motivation above, the remainder flow of erasing module for training is as follows. We let $\{S_i : 1 \leq i \leq m\}$ be a set of sizes of erasing with $S_1 < S_2 < \cdots < S_m$. Also, for each S_i and channel c, we let $\text{Area}(S_i)_c$ be a square with the height S_i. We assume the squares $\text{Area}(S_1)_c, \cdots, \text{Area}(S_m)_c$ have the same center that is in $X[c] \in \mathbb{R}^{H \times W}$ and the probability of choosing S_i is $\frac{1}{m}$. In particular, we select the center of these squares as the pixel that attains the maximum value of $X[c]$. Then we have $\text{Area}(S_1)_c \subseteq \text{Area}(S_2)_c \subseteq \cdots \subseteq \text{Area}(S_m)_c$. Also, due to this relation, we have the following Lemma:

Proposition 1. *For each pixel (i, j), the 'keep' probability $p(S_1, \cdots, S_m)_{c,i,j}$ is*

$$p(S_1, \cdots, S_m)_{c,i,j} = \begin{cases} 0, & \text{if } (i,j) \in \mathit{Area}(S_1)_c, \\ \frac{1}{m}, & \text{if } (i,j) \in \mathit{Area}(S_2)_c - \mathit{Area}(S_1)_c, \\ \vdots & \vdots \\ \frac{m-1}{m}, & \text{if } (i,j) \in \mathit{Area}(S_m)_c - \mathit{Area}(S_{m-1})_c, \\ 1, & \text{otherwise,} \end{cases} \tag{4.46}$$

Here, the 'keep' probability means the probability that the corresponding pixel will not be removed. Then we define a binary mask $M \in \mathbb{R}^{C \times H \times W}$, where

$$M_{c,i,j} \sim \mathit{Bernoulli}(p(S_1, \cdots, S_m)_{c,i,j}). \tag{4.47}$$

Proof. Since $\text{Area}(S_1)_c$ is a subset of all the squares

$$\text{Area}(S_2)_c, \cdots, \text{Area}(S_m)_c, \tag{4.48}$$

$p(S_1, \cdots, S_m)_{c,i,j} = 0$ if (i, j) is in $\text{Area}(S_1)_c$. Also, if (i, j) does not in any squares, then there are no squares that can remove (i, j). Thus, $p(S_1, \cdots, S_m)_{c,i,j} = 1$ for this case. Fix $2 \leq k \leq m$. Suppose $(i, j) \in \text{Area}(S_k)_c - \text{Area}(S_{k-1})_c$. Since $\text{Area}(S_k)_c \subseteq \text{Area}(S_{k+1})_c \subseteq \cdots \subseteq \text{Area}(S_m)_c$, we can remove (i, j) by choosing one of $S_k, \cdots, S_m$. Finally, the uniform distribution over $\{S_1, \cdots, S_m\}$ gives us

$$1 - p(S_1, \cdots, S_m)_{c,i,j} = \frac{m-k+1}{m}, \quad p(S_1, \cdots, S_m)_{c,i,j} = \frac{k-1}{m}. \tag{4.49}$$

□

Finally, with the above binary mask M, the output feature map is $X \odot M$, where $\odot$ is the elementwise multiplication. The flow of erasing module is described in Figure 4.7. Also, from the above discussion, we have the following definition:

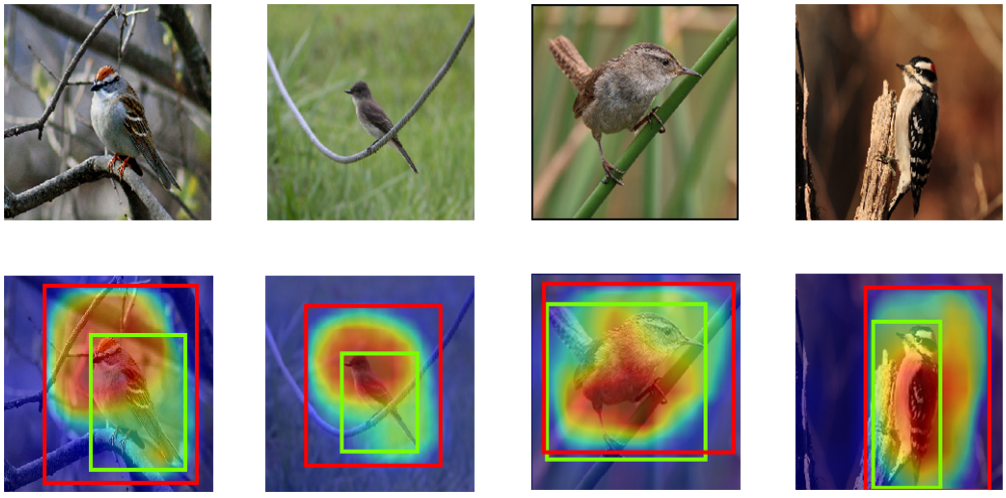


Figure 4.8: Some ADL results correspond to input images. The ground truth bounding box is represented by the green bounding box, and the estimated bounding box is described as the red bounding box.

Definition 6 (Erasing module)**.** *Let $X \in \mathbb{R}^{C\times H\times W}$ be the input feature map.* ***Erasing module*** *is a function that X maps to $M \odot X$, where $\odot$ is the elementwise multiplication and*

$$M = \begin{cases} \mathbf{1}, & \text{inference} \\ [M_{c,i,j}], & \text{training} \end{cases} \in \mathbb{R}^{C\times H\times W}, \tag{4.50}$$

where $p(S_1, \cdots, S_m)_{c,i,j}$ follows Equation 4.46, $M_{c,i,j} \sim$ Bernoulli$(p(S_1, \cdots, S_m)_{c,i,j})$, and $\mathbf{1}$ denotes the tensor filled with 1*.*

Note that if $m = 1$, then it is equivalent to the situation that we use only one size of the rectangle.

Final Step

As we discussed in Section 4.2, models such as ADL [5] have shown that using attention and erasing improves performance. However, since these models pick either attention or erasing at random, they use one piece of information for each iteration. This can limit boosting performance. For instance, if the

influence of erasing increases, it is likely to induce the network to recognize the background as the region of the object. Indeed, for example, ADL sometimes predicts a larger region than the region of the object. For a visual example, see Figure 4.8. Thus, we add two output feature maps from attention module and erasing module to use all information for each iteration. More explicitly, we use

$$X \mapsto w_1 f_1(X) + w_2 f_2(X), \tag{4.51}$$

where $w_1, w_2 > 0$, and f_1 and f_2 are the attention module and the erasing module, respectively. Here, when training, we set $w_1 + w_2 = 1$ and $w_1 = w_2 = 1$ when inference. These w_i's play the role in determining the ratio between attention module and erasing module.

4.4 Experiment

The same datasets and evaluation metrics as in Section 3 are used for our experiment.

4.4.1 Implementation Details

ResNet-50 [15], MobileNet V1 [16], and Inception V3 [47] are used as our backbone networks. The proposed method is inserted between the last layer of the backbone and GAP layer. Also, we switch the stride of the last layer from 2 to 1 when either ResNet-50 or MobileNet V1 is used the backbone network, which results in a feature map with a height of 14. The pretrained weight on ILSVRC [41] is used to train all models. SGD having 0.9 as its momentum and $5e-4$ as its weight decay optimizes all models using cross-entropy loss. Also, similar to SFPN, the tenfold learning rate of other layers is applied to the last fully connected layer. Batch size is 32 for CUB-200-2011 [50] and 128 for ILSVRC [41]. The weights w_i in Equation 4.5 are $w_1 = w_2 = 0.5$ for ResNet-50 and MobileNet V1, and $w_1 = 0.35$, $w_2 = 0.65$ for Inception V3. Also, A2E Net is implemented using PyTorch [37].

Backbone	Method	Top1-loc (%)	Top1-cls (%)
Inception V3 [47]	CAM [63]	43.67	-
	EGA [2]	45.74	72.11
	SPG [61]	46.64	-
	DANet [55]	49.45	71.20
	DGDM [56]	52.62	72.23
	ADL [5]	53.04	74.55
	CSoA [24]	53.94	76.10
	InCA [21]	56.10	64.00
	A2E Net [23]	**61.34**	**79.98**
ResNet-50 [15]	CAM [63]	47.81	80.55
	HCLNet [3]	54.07	**81.77**
	Cutmix [57]	54.81	-
	InCA [21]	56.10	80.40
	DGDM [56]	59.40	76.20
	DANet [55]	61.10	81.60
	ADL* [5]	62.29	80.34
	A2E Net [23]	**67.36**	76.80
MobileNet V1 [16]	CAM [63]	43.70	71.94
	HaS-32 [44]	44.67	66.64
	ADL [5]	47.74	70.43
	A2E Net [23]	**59.58**	**75.03**

Table 4.1: Comparison of WSOL performances for CUB-200-2011. The best performance is denoted by bold letters. Also, * is the result when the corresponding method is applied to ResNet50-SE [17] and - means the corresponding paper does not report the result.

4.4.2 Result

We compare the proposed method with existing WSOL models. Table 4.1–4.2 show the comparisons on CUB-200-2011 and ILSVRC, respectively.

In Table 4.1–4.2, the proposed method gains the best Top1-loc performance for MobileNet V1, ResNet-50, and Inception V3. First, when we use Inception V3, A2E Net shows an increase of 5.24 percentage points in Top1-loc when we compare it with InCA [21] in CUB-200-2011, and gets a higher Top1-

Backbone	Method	Top1-loc (%)	Top1-cls (%)
Inception V3 [47]	CAM [63]	46.29	-
	DANet [55]	47.53	72.50
	SPG [61]	48.60	-
	ADL [5]	48.71	72.83
	InCA [21]	49.30	**76.54**
	MEIL [33]	49.48	73.31
	EGA [2]	49.83	72.58
	CSoA [24]	51.19	71.90
	A2E Net [23]	**51.27**	74.01
ResNet-50 [15]	CAM [63]	46.03	74.57
	Cutmix [57]	47.25	-
	InCA [21]	48.40	71.31
	ADL* [5]	48.53	**75.85**
	DGDM* [56]	48.81	73.50
	A2E Net [23]	**49.48**	74.29
MobileNet V1 [16]	CAM [63]	41.66	68.38
	HaS-32 [44]	41.87	67.48
	ADL [5]	43.01	67.77
	A2E Net [23]	**45.02**	**70.58**

Table 4.2: Comparison of WSOL performances for ILSVRC. The best performance is denoted by bold letters. Also, * is the result when the corresponding method is applied to ResNet50-SE [17] and - means the corresponding paper does not report the result.

loc result than CSoA [24] by 0.08 percentage points when we use ILSVRC. Moreover, comparisons in MobileNet V1 and ResNet-50 show that A2E Net enjoys a negligible number of learnable weights.

When MobileNet V1 is our backbone network, A2E Net gains a higher Top1-loc result than ADL by 11.84 and 2.01 percentage points in CUB-200-2011 and ILSVRC, respectively. Here, since MobilieNet V1 is the lightweight backbone network, it is not suitable for this network to add a large number of learnable weights. Thus, CAM, HaS, and ADL, which have no additional trainable weights, are compared with A2E Net in MobilieNet V1. Because

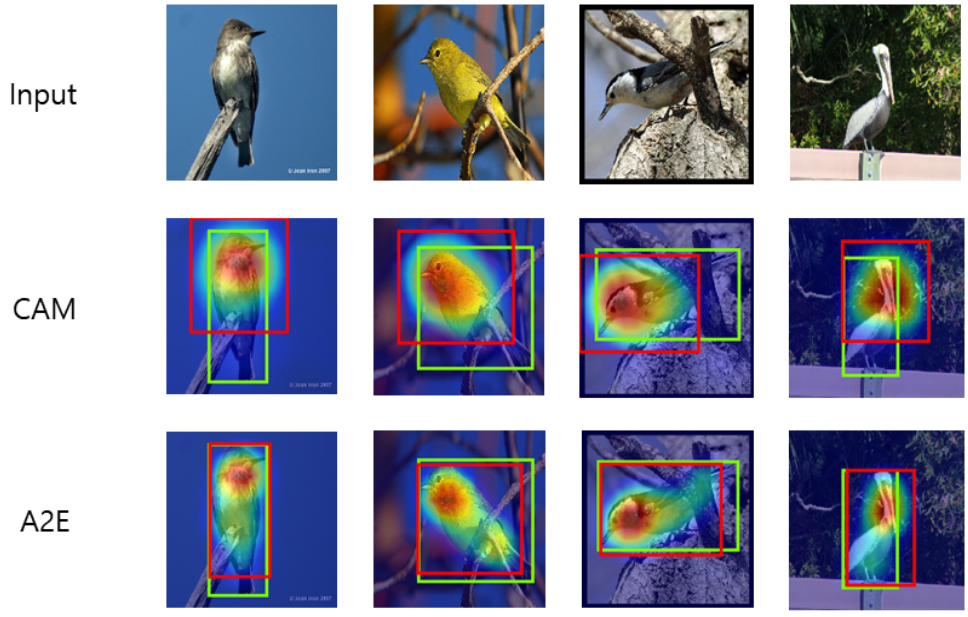


Figure 4.9: Comparison with CAM on CUB-200-2011. The ground truth bounding box is represented by the green bounding box, and the estimated bounding box is described as the red bounding box.

the number of learnable weights of A2E Net is negligible, this comparison shows that A2E Net gains better improvement than CAM, HaS, and ADL, not making MobilieNet V1 overweight.

The comparison in ResNet-50 also shows a similar result. A2E Net gains a higher Top1-loc result than ADL by 5.07 percentage points in CUB-200-2011 and a higher Top1-loc result than DGDM [56] by 0.67 percentage points in ILSVRC. Note that in this comparison, ADL and DGDM use ResNet50-SE [17] as their backbone network. Because ResNet50-SE uses more learnable parameters than ResNet-50, this comparison shows that A2E Net gets better improvement than ADL and DGDM with a smaller number of learnable parameters. In summary, A2E Net shows great performance with a negligible number of learnable weights via comparisons on two datasets.

The qualitative comparison also supports the great performance of the proposed method. Figure 4.9–4.10 show this comparison between CAM and A2E Net. The ground truth bounding box is represented by the green bound-

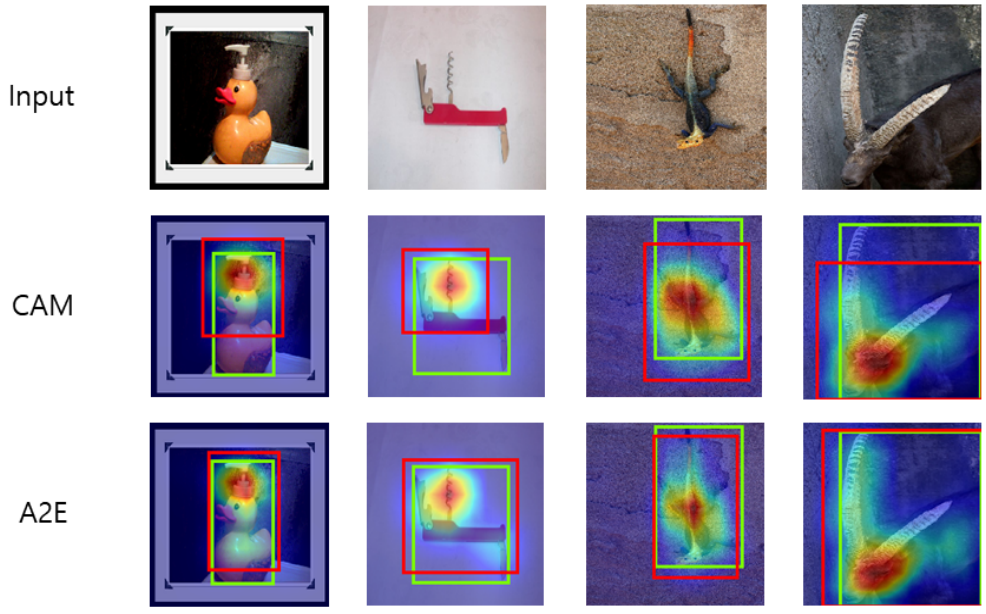


Figure 4.10: Comparison with CAM on ILSVRC. The ground truth bounding box is represented by the green bounding box, and the estimated bounding box is described as the red bounding box.

ing box, and the estimated bounding box is described as the red bounding box. In these figures, A2E Net recognizes a region closer to the whole region of the object compared to CAM. For example, in Figure 4.9, CAM focuses on the face of the bird. Figure 4.10 also shows the similar visualization. This recognition of A2E Net also helps a more precise estimation of the bounding box. Hence, the qualitative comparison shows that A2E Net produces a better prediction of the bounding box.

Based on the above qualitative and quantitative comparison, the proposed method shows its efficiency and great performance.

4.4.3 Ablation Study

We show the validity of the construction of A2E Net by various ablation studies. For this, ResNet-50 is used as the backbone, and CUB-200-2011 is used as the dataset.

Each component

SAB	Refinement Branch		Top1-loc (%)	Top1-cls (%)
	Attention module	Erasing module		
✗	✗	✗	47.81	**80.55**
	✓	✗	51.64	79.39
	✓	✓	64.64	76.08
✓	✗	✗	51.07	79.88
	✓	✗	52.30	77.46
	✓	✓	**67.36**	76.80

Table 4.3: Ablation study for spatial attention branch and refinement branch. Here, SAB means spatial attention branch.

We examine SAB, attention module, and erasing module. For this study, the result is shown in Table 4.3. First, the networks with SAB show higher Top1-loc results than those without SAB. The model with SAB (see the 4th row) increases Top1-loc by 3.26 percentage points compared to the model without any components (see the 1st row). Also, the model with SAB and attention model (see the 5th row) shows a larger Top1-loc performance than the model with attention module (see the 2nd row) by 0.66 percentage points. Finally, A2E Net (see the last row) has the improved Top1-loc performance compared to the model with refinement branch (see the 3rd row) by 2.72 percentage points. In these comparisons, we can know that SAB produces the feature map with preferable information.

Next, we check attention module. When we don't use SAB, the network with attention module gains a higher Top1-loc than the network without this module by 3.83 percentage points. (see the first to the second row) Also, when we use SAB, the network with attention module achieves an increase of 1.23 percentage points in Top1-loc. (see the 4th to the 5th row) These comparisons demonstrate that attention module helps better localization.

Finally, we examine erasing module. Without SAB, the model with erasing module (see the third row) shows higher Top1-loc by 13.00 percentage points compared to the model using only attention module (see the second row).

Also, with SAB, the model with erasing module (see the last row) increases Top1-loc by 15.06 percentage points compared to the model using only attention module (see the 5th row). These margins of increase in Top1-loc are quite higher than the aforementioned results for attention module. Thus, erasing module encourages the pretty improvement in performance.

The range of sizes

The range of size	Top1-loc (%)	Top1-cls (%)
$\{3\}$	57.02	78.39
$\{5\}$	59.15	78.15
$\{7\}$	62.01	77.96
$\{3, 5\}$	61.70	**79.46**
$\{3, 7\}$	63.53	78.51
$\{5, 7\}$	65.55	77.87
$\{3, 5, 7\}$	**67.36**	76.80

Table 4.4: Ablation study for the range of sizes.

We examine the size of erasing. Table 4.4 displays the result for this study. First, the models with two sizes (the 4th to the 6th row) show higher Top1-loc performance than the models with only one size (see the first three rows). For example, for the model using $\{3\}$ as only one size (see the first row), the models using $\{3, 5\}$ and $\{3, 7\}$ as the range of sizes increase Top1-loc by 4.68 percentage points and 6.51 percentage points, respectively.

Further, the model with three sizes (see the last row) achieves a higher Top1-loc than the models with two sizes (see the 4th to the 6th row) by 5.66, 3.83, and 1.81 percentage points, respectively. These results support our hypothesis that using only one size limits the improvement of performance. However, the model using $\{3, 5, 7, 9\}$ as the range of sizes results in 60.77% for Top1-loc and 78.81% for Top1-cls, which is lower Top1-loc than the model with three sizes (see the last row). This says that too many sizes can hurt the performance. Therefore, we need to use a suitable number of sizes and use $\{3, 5, 7\}$ as a range of the sizes.

Combination attention module with erasing module

SAB	Drop rate	Top1-loc (%)	Top1-cls (%)
✗	0.75	43.25	78.37
	0.5	46.17	79.34
	0.25	53.73	**80.32**
	-	64.64	76.08
✓	0.75	54.47	78.01
	0.5	57.35	79.05
	0.25	62.20	79.44
	-	**67.36**	76.80

Table 4.5: Ablation study for combination attention module with erasing module. SAB denotes spatial attention branch.

We examine the validity of our combination of attention module and erasing module. The goal of combining these modules is to use information from attention module and erasing module. For this, one can use the choice of either attention module or erasing module randomly, which is similar to the previous works [5, 21, 56]. To explore a better approach, we compare this random choice with A2E Net. The results for these comparisons are shown in Table 4.5. Drop rate in Table 4.5 is the probability of choosing erasing module.

When we do not use SAB, the first to the third row of Table 4.5 show that as the drop rate decreases, Top1-loc performance increases. Moreover, A2E Net without SAB (see the 4th row) shows the best performance in the case of not using SAB. Indeed, it gets a higher Top1-loc performance by 21.39, 18.47, and 10.91 percentage points, compared to the first to the third row of Table 4.5, respectively.

Otherwise, the 5th to the 7th row of Table 4.5 also display the decreased drop rate increases Top1-loc performance. Also, A2E Net (the last row of Table 4.5) gives the best performance in the case of using SAB, similar to the above case. Furthermore, it has a higher Top1-loc by 12.89, 10.01, and 5.16 percentage points, compared to the 5th to the 7th row of Table 4.5.

The above comparisons for all cases imply two things. First, when we

L_1	L_2	Top1-loc (%)	Top1-cls (%)
-	-	47.81	**80.55**
	NL-network [53]	47.88	80.51
	Attention module	**51.97**	79.03
SE [17]	-	47.60	79.00
	NL-network [53]	48.08	79.03
	Attention module	**51.69**	**79.44**
CBAM [54]	-	50.71	80.08
	NL-network [53]	50.02	**80.17**
	Attention module	**51.45**	79.65
SAB	-	51.07	79.88
	NL-network [53]	49.53	**80.70**
	Attention module	**53.30**	77.56

Table 4.6: Comparison with various attention mechanisms. '-' means that we do not use any attention mechanisms.

choose attention module and erasing module at random, Top1-loc performance is in inverse proportion to the drop rate. This shows that the large influence of erasing can limit boosting performance. Next, the networks with our combination show the best Top1-loc performance in each case. Thus, it is demonstrated that our combination is a better choice.

Comparison with other attention mechanisms

To validate the efficiency of SAB and attention module, we replace these with other attention mechanisms. SE [17] and CBAM [54], and NL-network [53] are used for this replacement. More precisely, either SE or CBAM is used instead of SAB, and NL-network is used instead of attention module. Also, erasing module is not included to focus only on attention mechanisms. With this change, Table 4.6 shows the result. L_1 and L_2 in Table 4.6 represent the locations where SAB and attention module are used, respectively. First, in the case of using NL-network, SAB records the comparable performance in

Model	# of learnable parameters
SAB	11
SE [17]	526,464
CBAM [54]	526,564
Attention module	0
NL-network [53]	8,397,824

Table 4.7: The number of learnable parameters of the attention mechanisms in our ablation study.

Top1-loc compared with SE and CBAM. Moreover, SAB achieves the highest performance in Top1-loc for other cases. Next, attention module represents the highest performance in Top1-loc for all cases. Thus, referring to Table 4.7, these comparisons show the efficiency of SAB and attention module.

Comparison with other erasing techniques

The style of erasing	Top1-loc (%)	Top1-cls (%)
HaS [44]	58.32	74.28
ACoL [60]	57.42	79.55
MEIL [33]	60.61	77.63
ADL [5]	41.91	80.08
InCA [21]	48.33	**80.27**
DGDM [56]	62.75	74.42
A2E Net [23]	**67.36**	76.80

Table 4.8: Comparison with the previous erasing methods.

We also compare our erasing technique with other methods to show the suitability of our erasing method. We divide the previous erasing methods in WSOL into 3 categories. First, we can apply the erasing method to the image directly like HaS [44]. Also, one or more classifiers can be used such as

ACoL [60] and MEIL [33]. Finally, we can pick either attention mechanism or erasing at random like ADL [5], InCA [21], and DGDM [56]. For the comparison with our erasing module, we adjust such methods to A2E Net. Namely, each of such methods replaces our erasing module, and SAB and attention module remain the same. For instance, if we use HaS [44] instead of erasing module, then the input image is erased by HaS, and refinement branch uses the skip connection without erasing module. Table 4.8 gives the corresponding results.

The first row in Table 4.8 says that the direct application of the erasing to the image can be less effective. Next, the use of one or more classifiers also has a lower performance in Top1-loc from the comparison with ACoL and MEIL. Lastly, A2E Net still gives the best performance in Top1-loc compared to the erasing methods in ADL [5], InCA [21], and DGDM [56]. Hence, these comparisons mean that our approach to erasing is more suitable than the previous erasing methods.

Chapter 5

Conclusion

In this thesis, we propose two WSOL models to alleviate the burden of learning. One is SFPN [22]. SFPN aims to remove the need for finding the hyperparameters of loss functions, improving performance. It first generates feature maps with enhanced information by the addition of low-resolution feature maps and high-resolution feature maps. These generated feature maps are involved in the last feature map of CAM and, thus, make the information of the last feature map more plentiful. Due to this, we use only cross-entropy loss for learning. In particular, this point is quite an advantage when we deal with a large dataset such as ILSVRC [41]. Moreover, SFPN demonstrates its great performance through experiments on CUB-200-2011 [50] and ILSVRC [41].

Next, we propose A2E Net [23]. Although SFPN shows great performance, SFPN requires additional learnable parameters to attain feature maps with better information. Because a number of learnable parameters are also a burden of learning, A2E Net aims to reduce the number of such parameters. To satisfy this goal, we construct SAB and refinement branch as ingredients of A2E Net.

Input feature map first passes through SAB. SAB introduces a transformation that informs the importance of each pixel. This transformation extracts the values by using layers, including CAP. Due to CAP, SAB has 11 learnable parameters and computes the representative values for each pixel. Thus, SAB gains better spatial information and is lightweight.

Then refinement branch uses the output feature map of SAB as its input.

This branch is operated by attention module and erasing module. Attention module generates an attention map with more elaborate information by using the connection between pixels. Further, this module does not exploit any trainable neural networks, unlike NL-network. Thus, this module also contributes to making A2E Net lightweight.

Erasing module removes the most discriminative region. Due to the property of pixels in the neighborhood [9, 48], we need to remove all pixels in the neighborhood. To this end, we set the rectangles and remove all pixels in these rectangles. Since the most discriminative region often contains the pixels of the maximum value, these pixels are the center of these rectangles. Also, we pick a size of these rectangles in a range of sizes at random to boost performance.

Finally, we get the final output of refinement branch using the output feature maps from attention module and erasing module. Choosing either attention module or erasing module, which is similar to the previous works [5,21,56], uses one piece of information for each iteration. Thus, it sometimes limits boosting performance. Because of this, we introduce a new method to use both two modules. Hence, we add the output feature maps of attention module and erasing module to use information from these two modules.

Using SAB and refinement branch, A2E Net shows great performance via experiments on CUB-200-2011 [50] and ILSVRC [41]. Moreover, it has lightweight branches, where the number of trainable parameters in each branch is negligible.

From SFPN to A2E Net, we aim to alleviate the burden of learning. An increased burden of learning is usually an obstacle to improving architecture or future research. In particular, this can be a crucial problem when dealing with a large dataset. So, research for reducing the burden of learning will be one of the important directions in the future.

Erasing technique can be the key to this. It is because erasing technique often requires no additional trainable parameters. One can consider the Rademacher Complexity [1] to obtain a better erasing technique. For example, LocalDrop [30] shows a novel regularization method by observing the Rademacher complexities of Dropout [45] and Dropblock [9]. Because Proposition 1 provides a mathematical view of erasing module of A2E Net, we

believe that it will give inspiration for improving the regularization and the localization performance in the future, by combining with mathematical tools such as the Rademacher Complexity.

Bibliography

[1] P. L. Bartlett and S. Mendelson, *Rademacher and gaussian complexities: Risk bounds and structural results*, Journal of Machine Learning Research, 3 (2002), pp. 463–482.

[2] S. N. Benassou, W. Shi, and F. Jiang, *Entropy guided adversarial model for weakly supervised object localization*, Neurocomputing, 429 (2021), pp. 60–68.

[3] S. N. Benassou, W. Shi, F. Jiang, and A. Benzine, *Hierarchical complementary learning for weakly supervised object localization*, Signal Processing: Image Communication, 100 (2022), p. 116520.

[4] A. Buades, B. Coll, and J.-M. Morel, *A non-local algorithm for image denoising*, in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 2, IEEE, 2005, pp. 60–65.

[5] J. Choe and H. Shim, *Attention-based dropout layer for weakly supervised object localization*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2219–2228.

[6] M. Corbetta and G. L. Shulman, *Control of goal-directed and stimulus-driven attention in the brain*, Nature reviews neuroscience, 3 (2002), pp. 201–215.

[7] P. Dabkowski and Y. Gal, *Real time image saliency for black box classifiers*, Advances in neural information processing systems, 30 (2017).

[8] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, *Dual attention network for scene segmentation*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 3146–3154.

[9] G. Ghiasi, T.-Y. Lin, and Q. V. Le, *Dropblock: a regularization method for convolutional networks*, in Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 10750–10760.

[10] R. Girshick, *Fast r-cnn*, in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016. `http://www.deeplearningbook.org`.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, Advances in neural information processing systems, 27 (2014).

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn*, in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.

[15] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, arXiv preprint arXiv:1704.04861, (2017).

[17] J. Hu, L. Shen, and G. Sun, *Squeeze-and-excitation networks*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7132–7141.

[18] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in International conference on machine learning, PMLR, 2015, pp. 448–456.

[19] L. Itti, C. Koch, and E. Niebur, *A model of saliency-based visual attention for rapid scene analysis*, IEEE Transactions on pattern analysis and machine intelligence, 20 (1998), pp. 1254–1259.

[20] M. Jaderberg, K. Simonyan, A. Zisserman, et al., *Spatial transformer networks*, Advances in neural information processing systems, 28 (2015).

[21] M. Ki, Y. Uh, W. Lee, and H. Byun, *In-sample contrastive learning and consistent attention for weakly supervised object localization*, in Proceedings of the Asian Conference on Computer Vision, 2020.

[22] B. Koo, H.-S. Choi, and M. Kang, *Simple feature pyramid network for weakly supervised object localization using multi-scale information*, Multidimensional Systems and Signal Processing, 32 (2021), pp. 1185–1197.

[23] ——, *Aggregation of attention and erasing for weakly supervised object localization*, Image and Vision Computing, 129 (2023), p. 104598.

[24] Z. Kou, G. Cui, S. Wang, W. Zhao, and C. Xu, *Improve cam with auto-adapted segmentation and co-supervised augmentation*, in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 3598–3606.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 25 (2012).

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, 86 (1998), pp. 2278–2324.

[27] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, *Feature pyramid networks for object detection*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2117–2125.

[28] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, *Focal loss for dense object detection*, in Proceedings of the IEEE International Conference on Computer Vision (ICCV), Oct 2017.

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, *Ssd: Single shot multibox detector*, in European conference on computer vision, Springer, 2016, pp. 21–37.

[30] Z. Lu, C. Xu, B. Du, T. Ishida, L. Zhang, and M. Sugiyama, *Localdrop: a hybrid regularization for deep neural networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2021).

[31] L. Luo, C. Yuan, K. Zhang, Y. Jiang, Y. Zhang, and H. Zhang, *Double shot: Preserve and erase based class attention networks for weakly supervised localization (peca-net)*, in 2020 IEEE international conference on multimedia and expo (ICME), IEEE, 2020, pp. 1–6.

[32] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, *Towards deep learning models resistant to adversarial attacks*, in International Conference on Learning Representations, 2018.

[33] J. Mai, M. Yang, and W. Luo, *Erasing integrated learning: A simple yet effective approach for weakly supervised object localization*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 8766–8775.

[34] A. Meethal, M. Pedersoli, S. Belharbi, and E. Granger, *Convolutional stn for weakly supervised object localization*, in 2020 25th

International Conference on Pattern Recognition (ICPR), IEEE, 2021, pp. 10157–10164.

[35] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, *Bam: Bottleneck attention module*, arXiv preprint arXiv:1807.06514, (2018).

[36] S. Park and N. Kwak, *Analysis on the dropout effect in convolutional neural networks*, in Asian conference on computer vision, Springer, 2016, pp. 189–204.

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *Pytorch: An imperative style, high-performance deep learning library*, in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[38] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[39] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, Advances in neural information processing systems, 28 (2015).

[40] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, in International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234–241.

[41] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., *Imagenet large scale visual recognition challenge*, International journal of computer vision, 115 (2015), pp. 211–252.

[42] L. Sifre and S. Mallat, *Rigid-motion scattering for image classification [phd thesis]*, Ecole Polytechnique, (2014).

[43] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, arXiv preprint arXiv:1409.1556, (2014).

[44] K. K. Singh and Y. J. Lee, *Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization*, in 2017 IEEE international conference on computer vision (ICCV), IEEE, 2017, pp. 3544–3553.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, The journal of machine learning research, 15 (2014), pp. 1929–1958.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, *Going deeper with convolutions*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[47] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, *Rethinking the inception architecture for computer vision*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.

[48] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, *Efficient object localization using convolutional networks*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 648–656.

[49] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, *Selective search for object recognition*, International journal of computer vision, 104 (2013), pp. 154–171.

[50] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, *The Caltech-UCSD Birds-200-2011 Dataset*, Tech. Rep. CNS-TR-2011-001, California Institute of Technology, 2011.

[51] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, *Residual attention network for image classification*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 3156–3164.

[52] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, *Eca-net: Efficient channel attention for deep convolutional neural networks*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[53] X. Wang, R. Girshick, A. Gupta, and K. He, *Non-local neural networks*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7794–7803.

[54] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, *Cbam: Convolutional block attention module*, in Proceedings of the European conference on computer vision (ECCV), 2018, pp. 3–19.

[55] H. Xue, C. Liu, F. Wan, J. Jiao, X. Ji, and Q. Ye, *Danet: Divergent activation for weakly supervised object localization*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6589–6598.

[56] J. Yin, S. Zhang, D. Chang, Z. Ma, and J. Guo, *Dual-attention guided dropblock module for weakly supervised object localization*, in 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, 2021, pp. 4229–4236.

[57] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, *Cutmix: Regularization strategy to train strong classifiers with localizable features*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6023–6032.

[58] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks*, in European conference on computer vision, Springer, 2014, pp. 818–833.

[59] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, *Self-attention generative adversarial networks*, in International conference on machine learning, PMLR, 2019, pp. 7354–7363.

[60] X. Zhang, Y. Wei, J. Feng, Y. Yang, and T. S. Huang, *Adversarial complementary learning for weakly supervised object localization*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1325–1334.

[61] X. Zhang, Y. Wei, G. Kang, Y. Yang, and T. Huang, *Self-produced guidance for weakly-supervised object localization*, in Proceedings of the European conference on computer vision (ECCV), 2018, pp. 597–613.

[62] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, *Residual non-local attention networks for image restoration*, arXiv preprint arXiv:1903.10082, (2019).

[63] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, *Learning deep features for discriminative localization*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2921–2929.

국문초록

본 학위논문에서 우리는 약한 지도 기반의 물체탐지를 위한 두 가지 모델을 제안한다. 기존의 많은 약한 지도 기반의 물체탐지를 위한 모델들은 손실함수의 하이퍼파라미터 찾기에 들어가는 비용이 무시하기 어렵다는 등의 한계점이 있다. 그래서 우리는 먼저 이 손실함수의 하이퍼파라미터 찾기에 들어가는 비용을 줄이기 위해서 SFPN이라는 이름을 가진 모델을 제안한다. SFPN은 특징 피라미드 네트워크의 구조를 활용하여 특징 맵들의 정보를 강화시켰다. 이후에 이 특징 맵들은 경계 상자의 예측에 참여한다. 이 과정은 성능 향상뿐만 아니라 오직 교차 엔트로피 함수만을 사용할 수 있게 하는 효과를 가져왔다. 뿐만 아니라 우리는 좀 더 적은 개수의 파라미터를 활용하기 위하여 두 번째 모델인 A2E Net을 제안한다. 이 모델은 공간 집중 분기, 정제 분기로 구성된다. 우선, 공간 집중 분기는 적은 개수의 파라미터를 사용하여 공간 정보를 강화시킨다. 그리고 정제 분기는 집중 모듈과 지우기 모듈로 구성되고, 이 모듈들은 모두 학습 가능한 파라미터가 없다. 공간 집중 분기의 결과를 입력으로 사용하여, 집중 모듈은 픽셀 간의 관계를 고려하여 특징 맵의 정보를 좀 더 정교하게 만든다. 또한, 지우기 모듈은 공간 집중 분기의 출력 특징 맵의 가장 구별되는 영역을 지워서 네트워크가 덜 구별되는 영역도 고려할 수 있도록 한다. 더욱이 지우는 영역의 크기를 다양하게 사용할 수 있게 하여 성능을 더 향상시켰다. 마지막으로, 집중과 지우기에서 나오는 정보를 모두 활용하기 위하여 이 두 모듈의 출력 특징 맵들을 더한다. 이렇게 제안된 SFPN과 A2E Net은 CUB-200-2011과 ILSVRC 에서의 실험을 통해 기존의 약지도 물체 탐지 기법들보다 좋은 성능을 가짐을 보였다.

주요어휘: 딥러닝, 약한 지도 학습, 물체 지역화
학번: 2017-21110