Abstract

This project delved into the issue of Single Positive Multi-Label Learning (SPML), a common challenge across various real-world domains such as image tagging and bioinformatics. To address this challenge, we proposed an innovative model that combines the BoostCAM algorithm with the EM [3]loss function. By incorporating an entropy maximization loss function, our model demonstrated greater robustness in handling negative labels and showed promising results across a range of standard datasets. Additionally, we attempted to apply the semi-supervised learning method Semireward to SPML. Although the results did not meet expectations due to the inherent complexity of multi-label learning, this attempt provided new avenues for future research directions. Later in the project, we explored a strategy that combines the SAM (Segment Anything Model) with the S2C (From SAM to CAM)[8] method, effectively improving the model's classification accuracy by first performing image segmentation followed by classification. While this approach may have some controversies in practical applications, it theoretically shows great potential in handling multi-label learning problems and provides valuable experience and insights for subsequent research. Our research not only advances the field of multi-label learning but also lays the foundation for technological progress in related areas.

Keywords: SPML, Image classification, Image segmentation, Semi-supervised learning, Multi label learning

目　录

Chapter1 Introduction

1.1 Background

In the field of Multi-Label Learning (MLL), the problem of Single-Positive Multi-Label Learning (SPML) presents a unique challenge. In real-world scenarios, many data objects such as images, texts, or audio files are often associated with multiple labels. However, in many cases, only one positive label is provided during the annotation process, while other potentially relevant labels are ignored. This phenomenon is especially common in large-scale datasets like ImageNet [10], where each image is typically labeled with only one category, even though the average number of categories per image is actually 1.22.

The core challenge of the SPML problem [1] is how to learn and predict all relevant labels based on limited annotated information. Since most images or other data objects contain multiple objects or categories, an effective method is needed to handle this complexity. Furthermore, collecting data with complete labels is not only time-consuming and expensive but also prone to errors, especially when images contain numerous categories, some of which may only be visible in a small part of the image.

In practical applications, methods for problem transformation and algorithm adaptation in multi-label learning are continuously evolving. For instance, the Binary Relevance method decomposes the multi-label problem into several binary classification problems, while Classifier Chains consider the interdependencies among labels in a chain-like manner. Overall, SPML emphasizes how to effectively train models capable of predicting multiple labels when the training data is incompletely annotated. With the advancement of deep learning techniques, researchers are exploring how to leverage deep neural network models and machine learning methods to solve multi-label learning problems and improve model performance and generalization ability. These studies not only advance the field of multi-label learning but also lay the foundation for technical progress in related fields.

1.2 Related Work

Single-Positive Multi-Label Learning (SPML) has emerged as a popular research direction in recent years within the fields of computer vision and machine learning. The main challenge of SPML lies in inferring the other potential labels of a sample based on only one positive label. This minimal annotation approach not only significantly reduces the cost of manual labeling but also offers an efficient learning mechanism for tasks with high label complexity, such as object recognition in images and multi-label text classification. However, SPML faces several challenges, particularly in addressing the issues of highly imbalanced positive and negative labels, label sparsity, and noisy labels, for which researchers have proposed various strategies.

Early studies, such as the ROLE algorithm proposed by Elijah Cole et al. in 2021, effectively addressed the asymmetry between positive and negative labels by introducing label count regularization and an Entropy-Maximization (EM) algorithm. ROLE improves the model's overall predictive ability by estimating the expected values of unknown labels and incorporating them into the learning process. Subsequent researchers, like Ning Xu et al. with the SMILE algorithm, further integrated the Unbiased Risk Estimator, refining pseudo-labeling strategies to enhance the inference of label dependencies in SPML models.

However, methods like ROLE and SMILE still face limitations in practical applications. Firstly, they rely on traditional pseudo-label generation strategies, which are vulnerable to noisy labels and may cause biases when handling unlabeled data. Secondly, the pseudo-label generation often depends on label prediction confidence, which can lead to overfitting to positive labels while neglecting the influence of negative labels. To address this issue, Biao Liu et al. proposed the MIME algorithm in 2022, which leverages Mutual Information between labels to enhance the accuracy of label inference by considering label correlations. Additionally, MIME introduces pseudo-label consistency regularization to ensure more robust performance across multiple tasks.

Another important study is the Large Loss Matters (LL) algorithm proposed by Kim et al. in 2022, which highlights the impact of large-loss samples on model training in weakly supervised multi-label learning. Kim et al. argued that traditional SPML methods often overlook large-loss samples when handling unknown labels, preventing the model from effectively learning from these samples with potential information. Therefore, the LL algorithm incorporates a specialized strategy for handling large-loss samples, including their loss in the model optimization process, thereby enhancing the model's robustness and inference accuracy. This strategy is particularly effective for datasets with a small number of labels or uneven label distributions.

In the further development of SPML, Xie et al. proposed the Label-Aware Global Consistency (LAC) algorithm in 2022. This method introduces label consistency regularization and pseudo-label consistency regularization, significantly improving the model's performance in fine-grained classification tasks. LAC utilizes the Self-Attention and Cross-Attention mechanisms of Transformers to strengthen the modeling of complex dependencies among labels. Compared with traditional SPML methods, LAC demonstrates better scalability and precision when handling high-dimensional label data. Moreover, LAC's global consistency regularization helps maintain high stability and robustness in the label inference process.

In a recent study published in 2024, Julio Arroyo et al. introduced Vision-Language Pseudo-Labels for Single-Positive Multi-Label Learning (VLSPL), marking a significant extension in the SPML field. VLSPL integrates Vision-Language Pseudo-Labels, combining visual and textual information to infer single positive labels. This cross-modal pseudo-label generation method captures semantic information from images more accurately, thus enhancing the SPML model' s understanding of labels. Specifically, VLSPL uses contextual clues in visual information (e.g., spatial relationships among objects) and textual descriptions to generate high-quality pseudo-labels, complementing the single positive label. This approach not only improves classification accuracy but also addresses the issue of semantic information loss caused by single-label annotation in traditional SPML methods.

1.3 Common Methods for SPML

The common methods in Single-Positive Multi-Label Learning (SPML) mainly focus on pseudo-label generation, label consistency regularization, modeling mutual information between labels, and the design of novel loss functions. These methods play a crucial role in improving model performance and enhancing the model' s capability to infer unknown labels.

1.3.1 Pseudo-Label Generation

Pseudo-labeling is a core technique in SPML. Early studies like ROLE, SMILE[16], and MIME[15] relied heavily on pseudo-label generation to fill in the missing label information in the dataset. The ROLE algorithm employs label count regularization, enabling the model to predict potential multiple labels for each sample. The SMILE algorithm incorporates an unbiased risk estimator to correct pseudo-labels, reducing the noise introduced by pseudo-labeling. In contrast, MIME further leverages mutual information between labels to generate more consistent and correlated pseudo-labels, thereby improving the model' s prediction accuracy. Additionally, VLSPL' s cross-modal pseudo-label generation method combines visual and language information, significantly enhancing the quality of pseudo-labels and the ability to capture semantic information.

1.3.2 Label Consistency Regularization

Label consistency regularization is an essential technique for improving the model' s understanding of label relationships. In the LAC algorithm, label consistency regularization introduces pseudo-label consistency, enabling the model to maintain better global consistency between labels during inference. This mechanism effectively reduces label interference and enhances the model' s performance in fine-grained classification tasks. The label consistency regularization of LAC is especially suitable for high-dimensional label data, such as CUB and CityScapes datasets.

1.3.3 Handling Large Losses

The Large Loss Matters (LL) algorithm proposed by Kim et al. emphasizes the importance of large-loss samples in SPML tasks. Traditional SPML methods often neglect these difficult-to-classify samples, leading to suboptimal model performance in complex scenarios. LL introduces a specific strategy for handling large-loss samples, incorporating them into the model' s optimization process, thereby improving the model' s robustness and accuracy. This approach is particularly effective for datasets with a small number of labels or imbalanced sample distributions and has broad application prospects.

1.3.4 Mutual Information Modeling

The MIME algorithm enhances the model' s ability to capture label dependencies by introducing mutual information between labels. Mutual information is a statistical measure of the dependency between two random variables. MIME leverages this information to strengthen the model' s understanding of the relationships between different labels, thereby achieving better predictive performance in multi-label classification tasks. Compared to other pseudo-label generation methods, MIME shows significant advantages in datasets with strong label correlations.

1.3.5 Cross-Modal Pseudo-Label Generation

VLSPL is a pseudo-label generation method based on the combination of visual and language information. It generates high-quality pseudo-labels by utilizing contextual information from both image and text descriptions. Unlike traditional pseudo-labeling methods that rely solely on visual information, VLSPL integrates the strengths of language and visual models, enabling the model to better capture semantic information in the image, thus improving label prediction accuracy. This method is particularly well-suited for tasks requiring detailed interpretation of image content, such as scene recognition and complex object classification.

1.4 Research Objectives and Main Research Content

1.4.1 Research Objectives

The main objective of Single-Positive Multi-Label Learning (SPML) is to effectively address the practical issues of high labeling costs and asymmetry between positive and negative labels. In traditional multi-label learning tasks, models typically rely on a large number of positive and negative labels to learn complex relationships between multiple categories. However, obtaining comprehensive annotations for each sample (i.e., annotating all positive and negative labels) is time-consuming and labor-intensive, especially for large-scale visual tasks like image recognition, object detection, and scene understanding. Therefore, how to infer all potential labels of a sample using partial or minimal annotations (such as a single positive label) has become a key focus for researchers.

The core goal of SPML is to predict the remaining possible labels for each sample by providing only one positive label. Compared to traditional multi-label learning, SPML significantly reduces the annotation requirements for training data, which is particularly important for building large-scale multi-label datasets. For example, in object recognition tasks, annotating only one object in an image allows the model to automatically infer other objects in the image, greatly reducing labeling costs.

Another key research objective of SPML is to tackle the challenges posed by label imbalance and missing labels. In multi-label tasks, label distributions are often highly imbalanced, with some labels appearing frequently while others are rare. Additionally, in real-world applications, the annotated positive labels are often partially random and may even contain biases, which can mislead the model during training. Thus, ensuring the model’s robustness and effectively inferring unknown labels under highly asymmetric label distributions is a crucial issue in SPML research.

To achieve these objectives, a major focus of SPML research is on designing efficient pseudo-label generation strategies. In traditional multi-label learning, models typically use annotated positive and negative labels to guide training. However, in SPML, where only a single positive label is provided, the model must rely on pseudo-label generation to supplement the missing labels. The quality of pseudo-labels is critical for the model’s inference performance, especially in the presence of noise or uncertainty. Current research has proposed various innovative methods to improve the accuracy of pseudo-label generation, aiming to increase the reliability and stability of pseudo-labels by introducing techniques like semi-supervised learning and mutual information modeling.

Additionally, SPML research aims to design new loss functions to balance the extreme asymmetry between positive and negative labels. Traditional loss functions used in multi-label learning are often unsuitable for SPML tasks because the number of negative labels far exceeds that of positive labels, and the information from positive labels is limited. In this context, designing loss functions tailored for SPML, which can maintain sensitivity to negative labels even when only a few positive labels are available, is another key research focus. This not only provides a new research direction for multi-label learning but also offers practical solutions for large-scale, multi-label tasks in real-world applications.

Chapter2 Model Design and Improvement

2.1 Combining BoostCAM with EM Algorithm

2.1.1 Model Motivation

One of the main challenges in Single Positive Multi-Label Learning (SPML) is handling the extreme asymmetry between positive and negative labels. In most cases, the model can only learn from a single positive label per sample, while the other labels remain unknown or unannotated. The Entropy Maximization (EM) algorithm, by maximizing the entropy of the data, has demonstrated good inference capabilities when dealing with unknown labels, especially in the scenarios of pseudo-label generation and label absence. However, the EM algorithm relies heavily on probabilistic estimation and can be vulnerable to high-noise data. This issue is particularly evident when the label distribution is imbalanced, potentially leading to a decline in prediction accuracy.

To enhance the performance of the EM algorithm under imbalanced positive and negative labels, we introduce the BoostLU[11] activation function. BoostLU is an enhanced activation function based on ReLU, which amplifies the activation values of positive labels, thereby increasing the model's sensitivity to positive labels. The advantage of BoostLU lies in its ability to strengthen the model's learning capacity for difficult-to-distinguish labels, particularly during the pseudo-label generation process, providing the model with better discriminative power. Thus, combining BoostLU with the EM algorithm effectively improves the model' s robustness in handling imbalanced and noisy labels.

2.1.2 Model Design

The core of the proposed method is applying the BoostLU activation function in the model' s positive label inference process, while integrating it with the loss function of the EM algorithm to optimize the model' s learning process. Specifically, BoostLU contributes to the model in the following ways:

1. Positive Label Enhancement:BoostLU amplifies the activation values of positive labels non-linearly, allowing the model to capture their features more accurately during processing. It modifies the traditional ReLU activation function, which is particularly beneficial for fine-grained classification tasks, enhancing the model' s ability to recognize weak signal labels.

2. Smoothing of Negative Labels: For negative labels, the EM algorithm adopts a special gradient design where negative labels are not directly involved in training. Instead, they are processed through smoothing to reduce their interference with the model. This approach is similar to the ”Large Loss Matters” strategy, where large-loss samples are handled specifically. This technique helps the model avoid excessive reliance on negative labels during training, thus mitigating the noise impact from pseudo-labels.

In terms of loss function design, we integrate the BoostLU activation function with the EM loss function. The activation values from BoostLU adjust the model's predictions to be more inclined towards positive labels. Meanwhile, the loss function of the EM algorithm aims to maximize the entropy of the model' s predictions, effectively reducing the risk of overfitting to negative labels and ensuring a gradual optimization of label inference accuracy throughout the training process.

2.1.3 Theoretical Analysis

Here, we conduct a gradient analysis using the Assume Negative (AN) model(1) as a simple baseline. In this model, L_+ and L_- are defined as shown in (2). The key characteristic of this approach is that when dealing with a single positive label, the remaining labels are directly assumed to be negative. This straightforward label inference mechanism results in an extremely asymmetric label distribution, where the model's gradient updates often rely on a single positive label, and all unannotated labels are treated as negative by the model. The difference between this scenario and the fully-labeled Binary Cross-Entropy (BCE) loss is shown in equation (3), where g_i represents the model's output logit, and $\sigma\left(g_i\right)=\frac{1}{1+e^{-g_i}}$:

$$\mathcal{L}_{AN}=\frac{1}{C}\left[\sum_{i\in\mathcal{I}^p}\mathcal{L}_{+}+\sum_{i\in\mathcal{I}^n\cup\mathcal{I}^\phi}\mathcal{L}_{-}\right] \tag{1}$$

$$\mathcal{L}_{+}=-\log\left(\sigma\left(g_i\right)\right),\mathcal{L}_{-}=-\log\left(1-\sigma\left(g_i\right)\right) \tag{2}$$

$$\begin{aligned}&\frac{1}{C}\left[\sum_{i\in\mathcal{I}^p}\frac{\partial\mathcal{L}_{+}}{\partial g_i}+\sum_{i\in\mathcal{I}^{fn}}\frac{\partial\mathcal{L}_{-}}{\partial g_i}+\sum_{i\in\mathcal{I}^{tn}}\frac{\partial\mathcal{L}_{-}}{\partial g_i}\right]-\frac{1}{C}\left[\sum_{i\in\mathcal{I}^p}\frac{\partial\mathcal{L}_{+}}{\partial g_i}+\sum_{i\in\mathcal{I}^{fn}}\frac{\partial\mathcal{L}_{+}}{\partial g_i}+\sum_{i\in\mathcal{I}^{tn}}\frac{\partial\mathcal{L}_{-}}{\partial g_i}\right]\\&=\frac{1}{C}\left(\sum_{i\in\mathcal{I}^{fn}}\frac{\partial\mathcal{L}_{-}}{\partial g_i}-\sum_{i\in\mathcal{I}^{fn}}\frac{\partial\mathcal{L}_{+}}{\partial g_i}\right)=\frac{\left|\mathcal{I}^{fn}\right|}{C}\end{aligned} \tag{3}$$

From the formula, it is not difficult to see that the gradient difference between fully

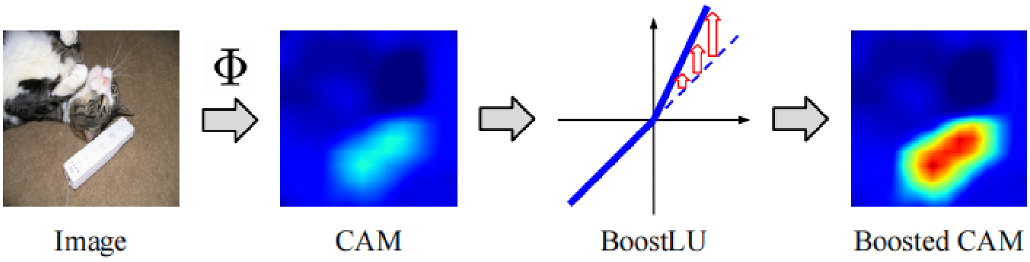


Figure 2.1: BoostLU Visualization

labeled classification and single positive label classification lies in the categories that are classified as false negatives. Next, we analyze the gradient of the EM (Entropy Maximization) algorithm. The EM loss function4, where$\epsilon$is a hyperparameter. Taking the partial derivative of L$\mathcal{L}$, gives us Equation 6

$$\mathcal{L}_{EM}\left(f^{(n)}, y^{(n)}\right) = -\frac{1}{C}\sum_{c=1}^{C}\left[1_{\left[y_c^{(n)}=1\right]}\log\left(f_c^{(n)}\right) + 1_{\left[y_c^{(n)}=0\right]}\epsilon H\left(f_c^{(n)}\right)\right] \tag{4}$$

$$H\left(f_c^{(n)}\right) = -\left[f_c^{(n)}\log\left(f_c^{(n)}\right) + \left(1-f_c^{(n)}\right)\log\left(1-f_c^{(n)}\right)\right] \tag{5}$$

$$\begin{cases} \dfrac{\partial\mathcal{L}_{+}}{\partial g} = \dfrac{\partial\mathcal{L}_{+}}{\partial p}\dfrac{\partial p}{\partial g} = \dfrac{-e^{-g}}{1+e^{-g}}, & y_c^{(n)} = 1, \\ \dfrac{\partial\mathcal{L}_{\varnothing}}{\partial g} = \dfrac{\partial\mathcal{L}_{\varnothing}}{\partial p}\dfrac{\partial p}{\partial g} = \dfrac{-\epsilon e^{-g}\log e^{-g}}{\left(1+e^{-g}\right)^2}, & y_c^{(n)} = 0, \end{cases} \tag{6}$$

Similarly, the visualization of the activation function for BoostLU is shown in Figure2.1Taking the gradient of the BoostLU activation function7, yeilds8

$$\mathrm{BoostLU}(x) = \max(x, \alpha x). \tag{7}$$

$$\begin{cases} \dfrac{\partial\mathcal{L}_{\varnothing}}{\partial g_i} = \dfrac{\partial\mathcal{L}_{+}}{\partial p}\dfrac{\partial p}{\partial g} = \alpha\left[\sigma\left(g_i\right)-1\right], & g_i > 0, \\ \dfrac{\partial\mathcal{L}_{\varnothing}}{\partial g_i} = \dfrac{\partial\mathcal{L}_{+}}{\partial p}\dfrac{\partial p}{\partial g} = \sigma\left(g_i\right)-1, & g_i < 0, \end{cases} \tag{8}$$

In summary, when we combine the EM loss function with the BoostLU activation function, by the chain rule, we obtain the following gradient results9,10, This formula actually explains that the EM loss function effectively encourages the gradients of unlabeled labels to become zero, while BoostLU further incentivizes the situation where$logit > 0$ Since the two are orthogonal, they can be used in combination.

$$\begin{cases} \dfrac{\partial \mathcal{L}_+}{\partial g} = \dfrac{\partial \mathcal{L}_+}{\partial p}\dfrac{\partial p}{\partial g} = \dfrac{-\alpha e^{-g}}{1+e^{-g}}, & y_c^{(n)} = 1, \\ \dfrac{\partial \mathcal{L}_\varnothing}{\partial g} = \dfrac{\partial \mathcal{L}_\varnothing}{\partial p}\dfrac{\partial p}{\partial g} = \dfrac{-\epsilon\alpha e^{-g}\log e^{-g}}{(1+e^{-g})^2}, & y_c^{(n)} = 0, \end{cases} \quad g_i > 0 \tag{9}$$

$$\begin{cases} \dfrac{\partial \mathcal{L}_+}{\partial g} = \dfrac{\partial \mathcal{L}_+}{\partial p}\dfrac{\partial p}{\partial g} = \dfrac{-e^{-g}}{1+e^{-g}}, & y_c^{(n)} = 1, \\ \dfrac{\partial \mathcal{L}_\varnothing}{\partial g} = \dfrac{\partial \mathcal{L}_\varnothing}{\partial p}\dfrac{\partial p}{\partial g} = \dfrac{-\epsilon e^{-g}\log e^{-g}}{(1+e^{-g})^2}, & y_c^{(n)} = 0, \end{cases} \quad g_i < 0 \tag{10}$$

2.2 Application of Semireward in SPML

2.2.1 Model Motivation

In Single Positive Multi-Label Learning (SPML), the model can only learn from a single positive label provided for each sample, while the other labels remain unannotated or unknown. Thus, generating high-quality pseudo-labels to aid the model in inferring the remaining labels becomes a critical issue. Semireward is a reward mechanism designed for Semi-Supervised Learning (SSL) tasks, which assigns reward scores to evaluate the quality of pseudo-labels. This mechanism helps to filter out high-quality pseudo-labels, preventing model bias caused by incorrect pseudo-labels.

In SPML, traditional pseudo-label generation methods often rely on confidence scores or simple strategies based on gradients and loss functions. These methods typically perform poorly in complex multi-label tasks, as the model might exhibit excessive confidence for certain labels, introducing noisy labels. To address these issues, we apply the Semireward method to SPML, aiming to improve the accuracy of pseudo-labels through the reward mechanism and reduce the impact of incorrect labels.

2.2.2 Model Design

The Semireward method introduces a Rewarder to evaluate the quality of pseudo-labels and filter out high-quality ones based on their reward scores. The design involves several key steps, with the model structure shown in Figure 2.2 and the training process illustrated in Figure 2.3.

1. Pseudo-label Generation: In the first stage, Semireward uses a generator model to create pseudo-labels. This generator model is pretrained on the labeled dataset and then further trained on a combination of randomly sampled labeled data and selected unlabeled data. The generator' s purpose is to produce high-quality pseudo-labels for training the student model. To mitigate confirmation bias, Semireward employs a subsampling strategy, which does not solely rely on high-confidence predictions of the model but instead selects more reliable pseudo-labels through subsampling.

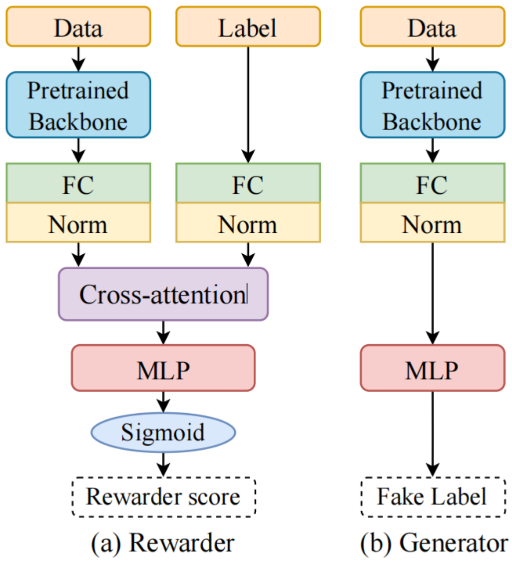
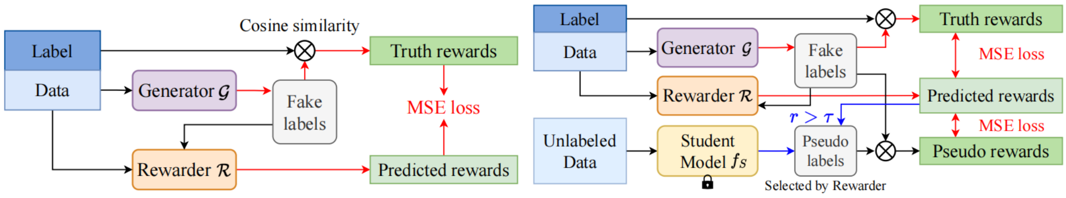


Figure 2.2: Semireward Model

In the second stage, Semireward trains a Rewarder network responsible for predicting reward scores based on the cosine similarity between pseudo-labels and true labels. The task of the Rewarder network is to assess the quality of pseudo-labels and filter out those most likely to be close to the true labels. This stage allows the Rewarder network to effectively learn the reward scores without interfering with the training of the student model.

2. Rewarder Design: The Rewarder assigns reward scores based on the quality of the pseudo-labels. These scores are calculated using cosine similarity between pseudo-labels and true labels. Cosine similarity is a smooth and monotonically increasing metric, effectively assessing the reliability of pseudo-labels. Specifically, the Rewarder processes input data and pseudo-labels through a cross-attention module and a Multi-Layer Perceptron (MLP) module, ultimately outputting a reward score for each pseudo-label. We made multi-label improvements, including introducing C label embedding layers to output different prediction values for each label, modifying the Cross Entropy (CE) loss function, and enhancing the Softmax activation mechanism to better fit weakly supervised multi-label problems.

3. Two-stage Training: The training of Semireward is divided into two stages. In the first stage, the Rewarder is pretrained using generated ”fake labels” to gradually learn how to accurately evaluate the quality of pseudo-labels. Simultaneously, the existing



(a) Stage 1: Pre-training with labeled data (b) Stage 2: Semi-supervised training with $\mathcal{D}_R$

Figure 2.3: Two-stage Training Overview

labeled data is used for model training. In the second stage, the Rewarder assists in model training by selecting high-quality pseudo-labels and dynamically adjusting the selection threshold, ensuring a continuous improvement in pseudo-label quality.

2.2.3 Theoretical Analysis

In Semireward, the design of the reward score is based on the similarity between pseudo-labels and true labels, enabling a more comprehensive and accurate evaluation of pseudo-label quality. Unlike traditional confidence scoring, the reward score considers not only the confidence of the pseudo-labels but also incorporates the semantic information of the data. This approach allows the model to evaluate pseudo-label quality more thoroughly, avoiding noise introduced by solely relying on confidence scores. Additionally, the Rewarder captures the correlation between input data and pseudo-labels through the cross-attention module. This module helps the model better understand complex dependencies in the data, especially in multi-label learning tasks where strong associations often exist between labels. By introducing the attention mechanism, the Rewarder can more accurately assess the semantic similarity between different labels, thus assigning more reasonable reward scores for pseudo-labels.

Theoretically, the design of Semireward reduces the model's dependency on incorrect pseudo-labels, thereby lowering the confirmation bias caused by inaccurate pseudo-labels. Confirmation bias refers to the model gradually falling into erroneous predictions during training due to reliance on low-quality pseudo-labels. Semireward effectively mitigates this issue by filtering high-quality pseudo-labels based on reward scores. However, the method faces certain limitations in SPML tasks. Firstly, SPML tasks exhibit strong label dependencies, with complex correlations often existing among multiple labels. Although the Rewarder can capture some of these dependencies, ensuring the quality of generated pseudo-labels remains challenging when dealing with highly dependent label sets. Secondly, since SPML relies solely on a single positive label, it further increases the difficulty of pseudo-label generation. Although the reward mechanism improves the selection

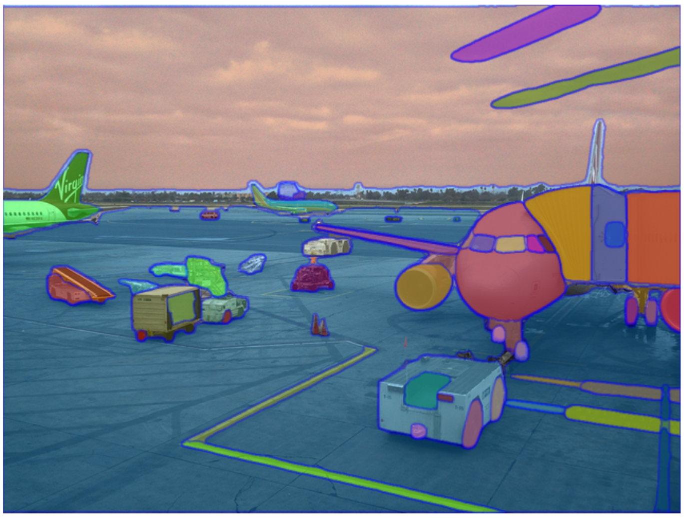


Figure 2.4: Segmentation results of the SAM model, using the MS-COCO dataset

quality of pseudo-labels to some extent, its effectiveness is still limited in scenarios with strong label dependencies.

2.3 Segmentation and Classification Method Combining SAM and CAM

2.3.1 Model Motivation

In Single Positive Multi-Label Learning (SPML), the model infers additional labels based solely on a single positive label per sample, facing challenges of label scarcity and complex label dependencies. To improve the quality of pseudo-label generation, the Segment Anything Model (SAM) offers powerful image segmentation capabilities, as shown in Figure 2.4. By segmenting an image into multiple regions, SAM helps the model better understand the image content, facilitating more accurate label inference. Building upon the idea of the S2C (From SAM to CAM) framework [8], we aim to further enhance pseudo-label accuracy by combining segmentation with Class Activation Map (CAM) to guide label inference.

S2C is a weakly supervised semantic segmentation framework that leverages the segmentation results of SAM to enhance the classifier's label inference accuracy. Unlike naive approaches that simply use SAM [7], in S2C, SAM is not only used to predict segmented regions for each object but also combined with activation maps to better capture the label dependencies. This method is particularly well-suited for SPML tasks, as it fully utilizes the annotated positive labels and additional image information to

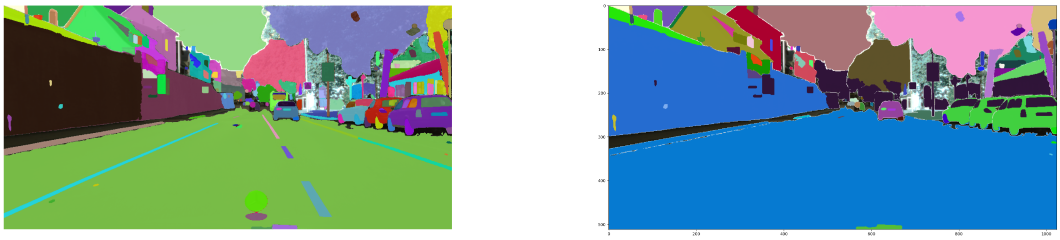


Figure 2.5: Original SAM segmentation output (left) and result after clustering (right), using the Cityscapes dataset

generate high-quality pseudo-labels. By integrating SAM' s segmentation output with the semantic information from CAM, S2C significantly improves the quality of pseudo-label generation and reduces the impact of noisy labels.

2.3.2 Model Design

We apply the Segment-Anything approach to SPML tasks, as shown in Figure 2.6:

1. Image Segmentation and Feature Extraction: First, we use SAM to segment the input image into multiple semantic regions (segments). These regions represent potential objects or scenes in the image. This step helps the model focus label inference on the segmented regions rather than the entire image. Unlike traditional pseudo-label generation methods, SAM' s segmentation helps the model concentrate on specific regions, thereby improving pseudo-label accuracy. However, we observed that SAM's segmentation tends to be overly detailed on SPML data, making it difficult to meet the task's needs even after extensive parameter tuning. To address this, we introduced a hierarchical clustering approach, merging similar or overlapping segments, resulting in improved performance, as shown in Figure 2.5. This not only enhances model inference efficiency but also has a positive impact on classification.

2. Combining CAM with Segmentation Results: Alongside generating segmentation results, we use the CAM module in S2C to produce activation maps of the image. These activation maps indicate which regions are associated with the positive label. However, CAM often focuses on the most salient feature regions, limiting label coverage. To address this issue, S2C combines SAM-generated segments with CAM by using local peaks extracted from activation maps as point prompts. These prompts guide SAM to produce more precise class-specific segmentation results.

3. Contrastive Learning and Pseudo-Label Generation: S2C introduces a contrastive learning mechanism called SAM-Segment Contrastive Learning (SSC). In this module, SAM's segmentation results are used to generate prototype features for each segmented region. These prototype features guide the model in learning feature similarities within different segments. Specifically, the model leverages contrastive learning to make features within the same segment more similar while keeping features from different segments distinct. This process helps the model capture label dependencies more effectively. Additionally, in the later training stages, the model introduces a CAM-based Prompt Module (CPM) loss, using the brightest areas of the activation map as point prompts inputted into the SAM model. However, we found that CPM is less effective in SPML tasks, likely due to noisy pseudo-labels causing incorrect prompts.

4. Filtering and Integrating Pseudo-Labels: During pseudo-label generation, the segmentation results from SAM are combined with the activation regions from CAM. The model evaluates each pseudo-label' s reliability using a stability score and an activation score. These scores are combined to filter high-quality pseudo-labels for subsequent training, using them as self-supervised signals to further optimize the model' s performance.

By integrating SPML with the S2C framework, the model effectively utilizes contextual information and spatial relationships between objects in the image when inferring unannotated labels, significantly improving the accuracy of pseudo-label generation.

2.3.3 Theoretical Analysis

The strength of the Segment Anything Model (SAM) lies in its ability to provide rich contextual information and segmentation regions for SPML tasks. This information helps the model infer multiple labels in an image more effectively. In multi-label tasks, spatial and semantic dependencies among different objects often serve as critical clues for inferring unlabeled tags. By incorporating SAM, the model can leverage the spatial relationships between segmented regions, making it especially effective in tasks with strong label dependencies.

For instance, when a bicycle and a rider are located in different areas of an image, traditional pseudo-label generation methods may struggle to capture the relationship between these labels. With SAM's segmentation, the model can separate different object regions and, through the contrastive learning mechanism in S2C, generate higher-quality pseudo-labels. Moreover, S2C combines the semantic information from CAM with SAM's segmentation output, allowing the model to consider features from different regions com-

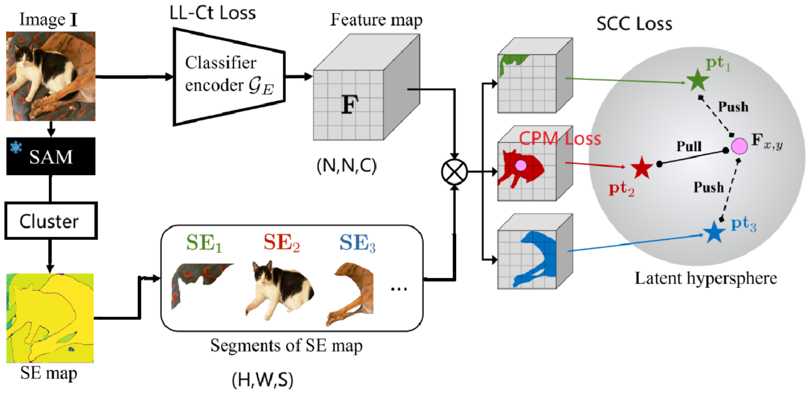


Figure 2.6: Model structure of S2C for SPML

prehensively when inferring labels, thereby improving the quality of pseudo-label generation.

From a theoretical standpoint, the application of S2C in SPML offers several key advantages:

1. Capturing Label Dependencies: In multi-label learning, labels often exhibit mutual dependencies. For example, an image containing a car is likely to also contain roads or buildings. Through SAM' s segmentation, the model can leverage spatial dependencies and use them as the basis for pseudo-label generation. Coupled with the contrastive learning in S2C, the model gains an enhanced understanding of these complex dependencies.

2. Semantic Consistency of Segmented Regions: The segmented regions produced by SAM typically have high semantic consistency, meaning that pixels within the same region usually belong to the same category. Compared to global feature inference, generating pseudo-labels based on segmented regions can reduce label confusion. For instance, in an image containing a cat and a dog, SAM can separate the cat and dog into distinct segments, reducing noise in the pseudo-label generation process.

3. Introduction of Contrastive Learning: S2C leverages contrastive learning to make features within the segmented regions more consistent, while maintaining separation from features in other regions. This mechanism helps the model better distinguish similar objects, thereby improving the accuracy of pseudo-label generation. The theoretical basis for this lies in the fact that the prototype features of segmented regions can serve as the foundation for generating pseudo-labels, and contrastive learning enhances the distinction among these features.

4. Utilization of Self-Supervised Signals: By combining scores from SAM and CAM, S2C can generate reliable pseudo-labels that are not only used for current task training

but also serve as self-supervised signals to further optimize the model. The introduction of self-supervision effectively reduces model bias caused by low-quality pseudo-labels and enhances the model' s robustness.

Chapter3 Experimental Results and Analysis

3.1 Experimental Results

The detailed numerical results of the experiments are shown in Table 3.1. Below is an analysis of each method:

1. AN (Assume Negative): The AN method assumes that the unlabeled tags are negative, which can reduce label noise when the data is sparse. However, this negative assumption can introduce incorrect predictions, leading to degraded performance. As shown in the table, AN performs poorly across all datasets, particularly on CUB (18.37%) and NUS (42.75%), indicating that simply assuming unlabeled tags as negative is insufficient to handle complex label dependencies in multi-label learning tasks.

2. WAN (Weak Assume Negative): Compared to AN, WAN is a relaxed version that incorporates more flexibility when assuming unlabeled tags as negative. WAN performs better than AN on multiple datasets, such as VOC (86.90%) and Mirflickr (72.19%). However, WAN still exhibits significant limitations in handling complex label relationships, especially on the CUB dataset (16.46%). This suggests that while soft-labeling improves performance, it may still fail on fine-grained or sparse-label datasets.

3. ROLE (Regularized Online Label Estimation): The ROLE method enhances the label inference capability of the model through regularized online label estimation, focusing on utilizing unlabeled tag information. As seen in the table, ROLE shows moderate performance on COCO (66.31%), NUS (49.33%), and CUB (19.20%), with stable results on VOC and Mirflickr datasets. While ROLE improves the model’s label prediction capability, its regularization strategy may be insufficient to capture complex label dependencies in multi-label learning.

4. EM (Entropy Maximization): Entropy Maximization (EM) estimates the likelihood of unlabeled tags by maximizing the model’s uncertainty (entropy), enhancing the model’s robustness in multi-label tasks. According to the table, EM performs well on COCO (71.02%), NUS (47.47%), and Mirflickr (76.15

5. BoostLU + LL_R: BoostLU combined with LL_R (Large Loss Rejection), which filters out excessively high loss values, often indicative of incorrect label predictions, performs exceptionally well across several datasets, particularly on VOC (89.31%) and vg-500 (25.89%). This method improves label inference by optimizing the activation function and introducing regularization, making it effective in complex multi-label scenarios. Notably, BoostLU + LL_R also exhibits robust performance on COCO (72.69%)

and NUS (49.50%), indicating its applicability to tasks of varying complexity.

6. BoostLU + EM: The combination of BoostLU and EM further enhances the model' s performance in handling complex label tasks, showing superior results on datasets like COCO (72.65%), CUB (21.38%), and vg-500 (23.15%). BoostLU optimizes the activation function, while EM' s inference capability on unlabeled tags significantly improves the model' s robustness and accuracy on sparse-label datasets.

7. LAC (Label-Aware Consistency) [14]: LAC improves the model' s multi-label prediction by introducing label consistency regularization and label-wise embedding. The table shows that LAC performs well on COCO (72.90%), NUS (49.84%), and CUB (23.41%), especially in handling complex label relationships. LAC's advantage lies in its ability to better capture label dependencies and ensure consistent predictions, making it highly adaptable in complex multi-label learning tasks.

8. VLPL (Vision-Language Partial-Label Learning) [2]: VLPL utilizes the CLIP model and combines vision-language representations [6] to generate pseudo-labels for partially labeled data. The table indicates that VLPL achieves excellent performance on the CUB dataset (89.05%), surpassing other methods, demonstrating the effectiveness of vision-language models in capturing the correlation between image content and labels. However, on datasets like COCO (71.49%) and NUS (49.30%), the performance is slightly lower, possibly due to the weaker association between fine-grained labels and vision-language representations in these cases.

9. S2C + llct (From SAM to CAM) [13]: S2C integrates the Segment Anything Model (SAM) with Class Activation Maps (CAM), achieving optimal or near-optimal performance across multiple datasets. For example, it reaches 99.3% accuracy on the VOC dataset and 75.4% on COCO. This method leverages a segment-first, classify-later approach, effectively utilizing the contextual information within the image, significantly enhancing performance in complex multi-label tasks. It shows superior results particularly on highly annotated datasets like COCO and VOC.

3.2 Analysis of Results: BoostCAM Combined with EM

Experimental results on various datasets demonstrate that the combination of BoostLU and EM significantly improves label inference accuracy in SPML tasks. In particular, on complex datasets such as MS-COCO and Pascal VOC, the combined model outperforms traditional EM and BoostCAM methods. This improvement is mainly attributed to the enhancement of positive labels by the BoostLU activation function, enabling the model to better capture dependencies between labels.

Dataset	**VOC**	**COCO**	**NUS**	**CUB**	**vg-500**	**Mirflickr**	**Cityscape**
Avg_label	**1.46**	**2.94**	**1.89**	**31.40**	**13.62**	**3.71**	**17.88**
BCE	89.69	76.44	52.18	30.48	30.47	80.93	62.88
AN	85.56	64.36	42.75	18.37	20.80	69.82	55.14
WAN	86.90	65.91	45.66	16.46	17.51	72.19	55.43
ROLE	88.69	66.31	49.33	19.20	19.62	71.51	56.93
EM	89.09	71.02	47.47	21.44	21.55	76.15	**57.03**
EM+APL	89.17	70.94	47.80	20.78	21.91	**76.21**	56.90
LAC	88.81	**72.90**	49.84	**23.41**	15.58	75.28	55.36
BoostLU+LL_R	**89.31**	72.69	49.50	18.41	**25.89**	72.61	56.10
VLPL	89.05	71.49	49.30	24.12	nan	nan	nan
BoostLU+EM	89.14	72.65	49.42	21.38	23.15	76.05	56.78
SemiReward	72.53	nan	nan	13.26	nan	nan	52.18
S2C+llct	99.3	75.4	nan	nan	nan	nan	55.6

Table 3.1: Comparison of model performance on different datasets

When handling datasets with sparse and noisy labels, the non-linear amplification mechanism of BoostLU stabilizes the learning of positive labels, reducing the interference from noise during pseudo-label generation. Additionally, experiments show that combining BoostLU with the EM loss function accelerates model convergence when processing unlabeled samples, leading to significant improvements in inference accuracy. Our model exhibits superior robustness across multiple datasets in multi-label tasks, providing an effective improvement path for SPML.

3.3 Analysis of Semireward Application in SPML

Results on several experimental datasets (e.g., Cityscapes, CUB-200) indicate that the introduction of the Semireward mechanism did not lead to significant improvement in overall label inference capability. Compared to traditional pseudo-label generation methods, the reward mechanism offers some assistance during the early training stages, reducing interference from noisy labels. However, its advantages diminish when faced with complex label dependencies. The limitations of this effect are mainly reflected in the following aspects:

1. Label Dependency Issue: In multi-label tasks, there is often strong dependency between labels; for instance, the presence of one object is frequently accompanied by others (e.g., a bicycle and a rider). In such cases, a single positive label may not provide sufficient semantic information for the model, leading to significant biases in pseudo-label generation.

2. Limitations of Reward Evaluator: Although the reward mechanism can filter pseudo-labels based on confidence, the complexity of pseudo-labels in multi-label scenarios makes it challenging for the evaluator to accurately assess their quality. Particularly

Figure 3.7: S2C segmentation for the ”Person” category (left) and ”Aeroplane” category (right), images from Pascal VOC

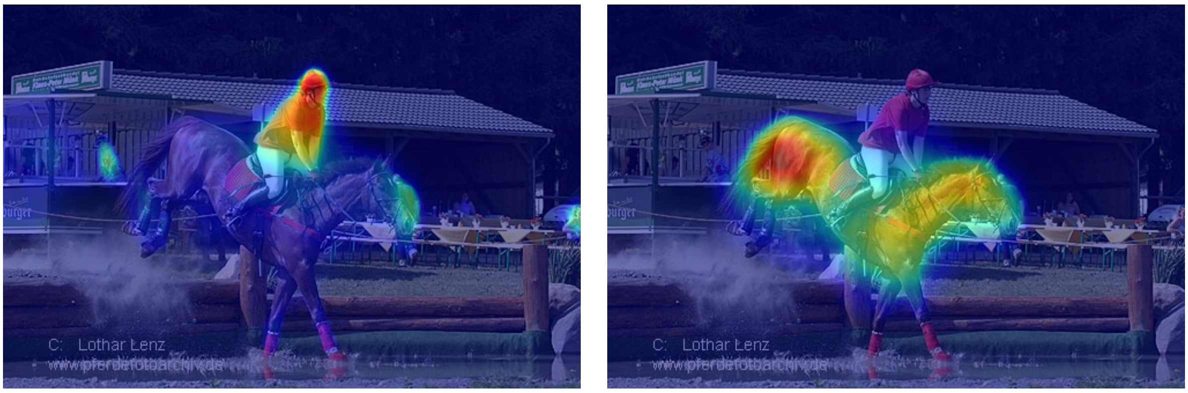


Figure 3.8: S2C segmentation for the ”Person” category (left) and ”Horse” category (right), images from Pascal VOC

when there is strong dependency between labels, the evaluator may struggle to make effective judgments based solely on the confidence of individual labels.

3. Label Imbalance: Label imbalance in SPML further exacerbates the difficulty of generating reliable pseudo-labels. For samples with rare labels, the reward mechanism may fail to provide adequate positive feedback, resulting in a decline in the quality of pseudo-labels for these labels.

3.4 Analysis of Segmentation Model Application in SPML

Experimental results on various datasets demonstrate that the combination of SAM and S2C significantly enhances label inference accuracy and the quality of pseudo-label generation in SPML tasks, as shown in Figures 3.8 and 3.7. Specifically, using images segmented by SAM, the pseudo-label generation process of the model becomes more precise. In complex scenes, the segmentation information provided by SAM helps the model better understand the distribution of objects in the image.

Moreover, experiments show that the contrastive learning mechanism of the S2C framework effectively reduces noise in pseudo-label generation, improving the model's performance in multi-label tasks. When dealing with datasets characterized by strong label dependencies but a low average number of categories per image (e.g., MS-COCO and Pascal VOC), the combination of SAM and S2C can more accurately capture the spatial relationships between labels, reducing label confusion. The introduction of the S2C framework into SPML provides an effective method for pseudo-label generation for single positive label multi-label learning. By incorporating segmentation information, contrastive learning, and self-supervised signals, S2C not only improves the quality of pseudo-labels but also significantly enhances the model's understanding of label dependencies. This approach excels in multi-label tasks, particularly when handling complex image scenes, and offers higher label inference accuracy.

However, the SAM model has a large number of parameters, requiring substantial computational resources and slower processing speed. Additionally, image quality can impact the effectiveness of SAM's segmentation. For blurry or noisy images, SAM may fail to produce accurate segmentation results. The model also struggles with scene features not well learned during training, a common issue with such large models, necessitating careful fine-tuning.

References

[1] Elijah Cole, Oisin Mac Aodha, Titouan Lorieul, Pietro Perona, Dan Morris, and Nebojsa Jojic. Multi-label learning from single positive labels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 933–942, 2021.

[2] Xing, Xin and Xiong, Zhexiao and Stylianou, Abby and Sastry, Srikumar and Gong, Liyu and Jacobs, Nathan. Vision-Language Pseudo-Labels for Single-Positive Multi-Label Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2024

[3] Donghao Zhou, Pengfei Chen, Qiong Wang, Guangyong Chen, and Pheng-Ann Heng. Acknowledging the unknown for multi-label learning with single positive labels. In European Conference on Computer Vision, pages 423–440. Springer, 2022

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html

[5] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In Proceedings of the ACM international conference on image and video retrieval, pages 1–9, 2009

[6] Sunan He, Taian Guo, Tao Dai, Ruizhi Qiao, Xiujun Shu, Bo Ren, and Shu-Tao Xia. Open-vocabulary multi-label classification via multi-modal knowledge transfer. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 808–816, 2023

[7] Tianle Chen, Zheda Mai, Ruiwen Li, and Wei-lun Chao. Segment anything model (sam) enhanced pseudo labels for weakly supervised semantic segmentation. arXiv preprint arXiv:2305.05803, 2023.

[8] Hyeokjun Kweon and Kuk-Jin Yoon. From SAM to CAMs: Exploring Segment Anything Model for Weakly Supervised Semantic Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 19499-19509, 2024

[9] Kirillov, Alexander and Mintun, Eric and Ravi, Nikhila and Mao, Hanzi and Rolland, Chloe and Gustafson, Laura and Xiao, Tete and Whitehead, Spencer and Berg, Alexander C. and Lo, Wan-Yen and Dollr, Piotr and Girshick, Ross. Segment

Anything. arXiv:2304.02643, 2023

[10] Sangdoo Yun, Seong Joon Oh, Byeongho Heo, Dongyoon Han, Junsuk Choe, and Sanghyuk Chun. Re-labeling imagenet: from single to multi-labels, from global to localized labels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2340–2350, 2021.

[11] Kim, Youngwook and Kim, Jae Myung and Jeong, Jieun and Schmid, Cordelia and Akata, Zeynep and Lee, Jungwoo. Bridging the Gap Between Model Explanations in Partially Annotated Multi-Label Classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3408-3417, 2023.

[12] Siyuan Li and Weiyang Jin and Zedong Wang and Fang Wu and Zicheng Liu and Cheng Tan and Stan Z. Li. SemiReward: A General Reward Model for Semi-supervised Learning. In International Conference on Learning Representations, 2024.

[13] Kim, Youngwook and Kim, Jae Myung and Akata, Zeynep and Lee, Jungwoo. Large Loss Matters in Weakly Supervised Multi-Label Classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 14156-14165, 2022.

[14] Ming-Kun Xie and Jia-Hao Xiao and Sheng-Jun Huang. Label-Aware Global Consistency for Multi-Label Learning with Single Positive Labels. In Advances in Neural Information Processing Systems, 2022.

[15] Biao Liu, Ning Xu, Jiaqi Lv, Xin Geng. Label-Aware Global Consistency for Multi-Label Learning with Single Positive Labels. In Proceedings of the 40 th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[16] Ning Xu, Congyu Qiao, Jiaqi Lv, Xin Geng1, Min-Ling Zhang. One Positive Label is Sufficient: Single-Positive Multi-Label Learning with Label Enhancement. In 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

[17] Hao Guo and Song Wang. Long-tailed multi-label visual recognition by collaborative training on uniform and rebalanced samplings. In CVPR, pages 15089–15098, 2021.

[18] Thibaut Durand, Nazanin Mehrasa, and Greg Mori. Learning a deep convnet for multi-label classification with partial labels. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 647–657, 2019.

[19] Oisin Mac Aodha, Elijah Cole, and Pietro Perona. Presence-only geographical priors for fine-grained image classification. In IEEE International Conference on Computer Vision, pages 9596–9606, 2019.

Acknowledgments

I would like to express my sincere gratitude to my advisor, Professor Jia Yuheng, for providing me with substantial help and constructive suggestions throughout every stage of this project. He has also shown great care in all aspects of my academic and personal life, especially during times when I felt confused or down. His extensive knowledge, rigorous academic attitude, and the relatively relaxed student-teacher relationship have had a profound influence on my learning and research.

I am also grateful to my graduate mentors, including Jianhong Chen, Jiahao Jiang, and Xiaoyu Wang, who have each contributed significantly to this project in different aspects.

I appreciate the contributions of my project team members, who played key roles in tasks such as document writing, paper reading, and code debugging.

I would also like to thank the Southeast University PALM Lab (Key Laboratory of New Generation Artificial Intelligence Technology and Cross-Application of the Ministry of Education). The projects I was responsible for there are closely related to this work and have helped me hone my skills in various areas.

Dataset Introduction

Pascal VOC Dataset[4]

The Pascal VOC dataset is a benchmark dataset in the field of computer vision, widely used for object detection and image segmentation tasks. It originated from the EU-funded PASCAL2 Network of Excellence on Pattern Analysis, Statistical Modelling, and Computational Learning project. The Pascal VOC Challenge was held annually from 2005 to 2012. Although the competition has ended, the dataset remains a significant resource for computer vision research.

Pascal VOC includes multiple versions, with the most commonly used being Pascal VOC 2007 and Pascal VOC 2012. This paper uses VOC2012. These versions offer extensive image data and detailed annotations, including object bounding boxes, class labels, and pixel-level semantic segmentation masks. The dataset structure is well-organized and typically includes the following parts: - JPEGImages : Stores all the training and testing images in JPEG format.
- Annotations : Stores the annotation files for each image in XML format, including object class, bounding box coordinates, truncation status, occlusion status, and difficulty of recognition.
- ImageSets : Contains index files for different tasks, such as 'train.txt', 'val.txt', and 'test.txt', which list the images for the training, validation, and testing sets.
- SegmentationClass : Stores pixel-level annotations for semantic segmentation, indicating the class of each pixel.
- SegmentationObject : Stores annotations for instance segmentation, identifying the contours and classes of individual objects.

The Pascal VOC dataset plays a crucial role in the development of object detection and image segmentation algorithms. It provides researchers with abundant image resources and annotations, fostering the advancement of related algorithms. Many popular object detection and image segmentation models, such as R-CNN, Fast R-CNN, YOLO, and SSD, have been trained and evaluated using the Pascal VOC dataset. Additionally, the dataset is widely used in tasks such as image classification, object recognition, and tracking.

MS-COCO Dataset

The COCO (Common Objects in Context) dataset, initiated by Microsoft, is a large-scale image recognition project aimed at advancing the field of computer vision. Since its release in 2014, it has become one of the most popular and authoritative datasets in the field. The COCO dataset contains over 330,000 images of everyday scenes, covering 80 different object categories such as people, cars, and animals, as well as 91 different material categories like sky, grass, and buildings. Each image comes with detailed annotations, including object bounding boxes, segmentation masks, and even keypoint annotations like joints of a person. Additionally, the COCO dataset offers image captions, where each image has five descriptive sentences, enriching the dataset's diversity. The dataset is divided into a training set, validation set, and test set, containing 118,000, 5,000, and 20,000 images, respectively.

The COCO dataset is valued not only for its large scale and rich annotations but also for providing standardized evaluation metrics, such as mean Average Precision (mAP) and mean Average Recall (mAR), allowing researchers and developers to accurately measure and compare model performance. These features make COCO a benchmark dataset for tasks like object detection, multi-label image classification, instance segmentation, keypoint detection, and image captioning.

NUS-wide Dataset[5]

The NUS-WIDE dataset, or "a real-world web image database from National University of Singapore," was created by the Media Search Lab at the National University of Singapore. It is a multi-label web image dataset aimed at enhancing image annotation and retrieval methods. The NUS-WIDE dataset contains 269,648 images, covering 5,018 specific labels that encompass diverse content ranging from everyday life to natural landscapes.

In addition to its extensive image resources, NUS-WIDE provides six types of low-level features, including a 64-D color histogram, a 144-D color correlogram, a 73-D edge histogram, a 128-D wavelet texture, a 225-D block color moment based on a 5×5 fixed grid partition, and a 500-D bag-of-words representation using SIFT descriptors. These features offer valuable information for image processing and analysis. The dataset also defines 81 concepts that cover both objects and scenes, providing rich annotations for various visual research tasks.

Moreover, the NUS-WIDE dataset includes image URLs collected from Flickr, along with metadata such as geographic and EXIF information, which can be used for more in-depth image analysis. The dataset is well-structured, making it easy for researchers to download and use. Researchers can leverage this dataset to explore tasks like image retrieval, multi-label classification, and cross-modal retrieval, and use traditional k-NN algorithms to learn from the labels, providing baseline results for web image annotation. Benchmark results indicate that it is possible to learn effective models from sufficiently large image datasets, facilitating general image retrieval.

CUB-200 Dataset

The CUB-200-2011 dataset, short for Caltech-UCSD Birds-200-2011, is a bird image classification dataset jointly created by Caltech and UCSD. It is an extended version of the original CUB-200 dataset, approximately doubling the number of images per class and adding additional localization annotations to provide more detailed fine-grained information.

The CUB-200-2011 dataset features: - **Diverse Species**: It contains 200 different bird species, with approximately 30 images per species, totaling 11,788 high-resolution images. - **Large Data Size**: It consists of 5,994 training images and 5,794 testing images. - **Fine-Grained Annotations**: In addition to class labels, each image is annotated with 15 key points (e.g., beak, eyes, wings), 312 binary attributes (e.g., color, shape), and bounding boxes for the birds. We follow previous studies by using the 312 categories as labels for multi-label tasks.

The CUB-200-2011 dataset is widely used in computer vision research, especially for tasks such as fine-grained classification, object detection, and domain adaptation. Due to its fine-grained features and challenging nature, CUB-200-2011 serves as an important benchmark for evaluating model performance. Moreover, this dataset can be utilized to explore new methods in image processing and pattern recognition, making significant contributions to the advancement of related technologies.

VG-500 Dataset

The VG-500 dataset is a subset of the Visual Genome dataset, containing 500 common categories curated from the larger Visual Genome dataset. Visual Genome (VG), introduced by Fei-Fei Li's team at Stanford University in 2016, is a large-scale image dataset focused on advancing research in high-level semantic understanding of images.

The VG-500 dataset inherits the characteristics of Visual Genome, with each image containing rich semantic information. The annotation process is meticulous: annotators first provide descriptions of the image, then mark the bounding boxes, objects, attributes, and relationships based on the descriptions. This procedure ensures accurate annotations and focuses on the main subjects of the images.

The statistical distribution of the Visual Genome dataset shows a long-tail pattern in object and relationship categories, where a small number of categories account for the majority of annotations. To address this, Fei-Fei Li's team later introduced the VG150 dataset, containing only the 150 most frequent object categories and 50 most frequent relationships. However, even in VG150, the issue of biased relationship annotations persists.

The VG-500 dataset, as a smaller-scale version, offers an easily manageable and usable resource while retaining the richness and complexity of the Visual Genome dataset. It is suitable for a variety of vision tasks, such as multi-label image classification, object detection, image captioning, and visual question answering.

Code

BoostLU+EM Core Code

```
def loss_EM(batch, P, Z):
    # unpack:
    preds = batch['preds']
    observed_labels = batch['label_vec_obs']
    true_labels = batch['label_vec_true'].to(Z['device'])

    # input validation:
    assert torch.min(observed_labels) >= 0

    loss_mtx = torch.zeros_like(preds)

    loss_mtx[observed_labels == 1] = neg_log(preds[observed_labels == 1])
    loss_mtx[observed_labels == 0] = -P['alpha'] * (
            preds[observed_labels == 0] * neg_log(preds[observed_labels == 0]) +
            (1 - preds[observed_labels == 0]) * neg_log(1 - preds[observed_labels == 0])
        )

    return loss_mtx, None


def loss_EM_APL(batch, P, Z):
    # unpack:
    preds = batch['preds']
    observed_labels = batch['label_vec_obs']

    # input validation:
    assert torch.min(observed_labels) >= -1

    loss_mtx = torch.zeros_like(preds)

    loss_mtx[observed_labels == 1] = neg_log(preds[observed_labels == 1])
    loss_mtx[observed_labels == 0] = -P['alpha'] * (
            preds[observed_labels == 0] * neg_log(preds[observed_labels == 0]) +
            (1 - preds[observed_labels == 0]) * neg_log(1 - preds[observed_labels == 0])
        )

    soft_label = -observed_labels[observed_labels < 0]
    loss_mtx[observed_labels < 0] = P['beta'] * (
            soft_label * neg_log(preds[observed_labels < 0]) +
            (1 - soft_label) * neg_log(1 - preds[observed_labels < 0])
        )
    return loss_mtx, None

class GlobalAvgPool2d(torch.nn.Module):
    def __init__(self):
```

```
        super(GlobalAvgPool2d, self).__init__()

    def forward(self, feature_map):
        return torch.nn.functional.adaptive_avg_pool2d(feature_map, ...
            1).squeeze(-1).squeeze(-1)

class ImageClassifier(torch.nn.Module):
    def __init__(self, P, model_feature_extractor=None, model_linear_classifier=None):
        super(ImageClassifier, self).__init__()

        self.arch = P['arch']
        if self.arch == 'resnet50':
            print('feature extractor: imagenet pretrained')
            feature_extractor = resnet50(pretrained=P['use_pretrained'])
        feature_extractor = torch.nn.Sequential(*list(feature_extractor.children())[:-2])

        if P['freeze_feature_extractor']:
            for param in feature_extractor.parameters():
                param.requires_grad = False
        else:
            for param in feature_extractor.parameters():
                param.requires_grad = True

        self.feature_extractor = feature_extractor
        self.avgpool = GlobalAvgPool2d()
        self.onebyone_conv = torch.nn.Conv2d(P['feat_dim'], P['num_classes'], 1)
        self.alpha = P['alpha_b']

        # print(P['image_ids'])

    def unfreeze_feature_extractor(self):
        for param in self.feature_extractor.parameters():
            param.requires_grad = True

    def forward(self, x):
        feats = self.feature_extractor(x)
        CAMa = self.onebyone_conv(feats)
        CAMa = torch.where(CAMa > 0, CAMa * self.alpha, CAMa)  # BoostLU operation
        logits = torch.nn.functional.adaptive_avg_pool2d(CAMa, 1).squeeze(-1).squeeze(-1)
        # print(feats.size())
        # print(CAMa.size())
        # print(logits.size())
        return logits

class MultilabelModel(torch.nn.Module):
    def __init__(self, P, feature_extractor, linear_classifier, observed_label_matrix, ...
        estimated_labels=None):
        super(MultilabelModel, self).__init__()
        self.f = ImageClassifier(P, feature_extractor, linear_classifier)
        # self.g = LabelEstimator(P, observed_label_matrix, estimated_labels)

    def forward(self, batch):
```

```
        f_logits = self.f(batch['image'])
        # print(f_logits.size())
        # g_preds = self.g(batch['idx'])
        return f_logits
```

S2C for SPML core code

```

class LLCtLoss(nn.Module):
    def __init__(self):
        super(LLCtLoss, self).__init__()

    def forward(self, epoch, pred, target):
        batch_size = int(pred.size(0))
        num_classes = int(pred.size(1))
        loss_matrix = F.binary_cross_entropy_with_logits(pred, target, reduction='none')
        corrected_loss_matrix = F.binary_cross_entropy_with_logits(pred, ...
            torch.logical_not(target).float(), reduction='none')
        P_△ = 0.01
        P_clean_rate = 1- P_△ * epoch
        unobserved_mask = (target == 0)
        correction_idx = []
        if epoch≤1:
            final_loss_matrix = loss_matrix
        else:
            k = math.ceil(batch_size * num_classes * (1-P_clean_rate))
            #k=1
            unobserved_loss = unobserved_mask.bool() * loss_matrix
            topk = torch.topk(unobserved_loss.flatten(), k)
            topk_lossvalue = topk.values[-1]
            correction_idx = torch.where(unobserved_loss > topk_lossvalue)
            final_loss_matrix = torch.where(unobserved_loss < topk_lossvalue, ...
                loss_matrix, corrected_loss_matrix)
        main_loss = final_loss_matrix.mean()
        return main_loss, correction_idx

class model_WSSS():

    def __init__(self, args, logger=None, writer=None):

        self.args = args
        self.categories = args.categories
        # Common things
        self.phase = 'train'
        self.dev = 'cuda'
        self.bce = nn.BCEWithLogitsLoss()
        self.llct = LLCtLoss()
        self.L1 = nn.L1Loss()
        self.bs = args.batch_size
        if logger is not None:
```

```
            self.logger = logger
        if writer is not None:
            self.writer = writer

        # Attributes
        self.C = args.C  # Number of classes - VOC : 20
        self.D = args.D  # Feature dimension - Default : 256
        self.W = args.W  # Weight for each term in loss - Default : [1, 1, 1]

        self.T = args.T
        self.th_multi = args.th_multi # Default: 0.5
        self.size_sam = 1024

        # Model attributes
        self.net_names = ['net_main']
        self.base_names = ['cls', 'ssc', 'cpm']
        self.loss_names = ['loss_' + bn for bn in self.base_names]
        self.acc_names = ['acc_' + bn for bn in self.base_names]

        self.nets = []
        self.opts = []

        # Evaluation-related
        self.running_loss = [0] * len(self.loss_names)
        self.right_count = [0] * len(self.acc_names)
        self.wrong_count = [0] * len(self.acc_names)
        self.accs = [0] * len(self.acc_names)
        self.count = 0
        self.num_count = 0

        # Define networks
        self.net_main = resnet38d.Net_CAM(C=self.C, D=self.D)
        sam_path = './pretrained/sam_vit_h.pth'
        self.net_sam = sam_model_registry['vit_h'](checkpoint=sam_path)

        # Initialize networks with ImageNet pretrained weight
        self.net_main.load_state_dict(resnet38d.convert_mxnet_to_torch('./pretrained/resnet_38d.params'),
            strict=False)

    # Load networks
    def load_model(self, epo, ckpt_path):
        epo_str = str(epo).zfill(3)
        self.net_main.load_state_dict(torch.load(ckpt_path + '/' + epo_str + ...
            'net_main.pth'), strict=True)
        if not self.args.debug:
            self.net_main = torch.nn.DataParallel(self.net_main.to(self.dev))

    # Do forward/backward propagation and call optimizer to update the networks
    def update(self, epo, iter):

        # Tensor dimensions
        B = self.img.shape[0]
```

```
        H = self.img.shape[2]
        W = self.img.shape[3]
        C = self.C
        D = self.D

        self.net_sam.eval()
        use_cpm = epo>self.args.sstart-1

        ##################################################################### CPM branch ...
            ###################################################################
        if use_cpm:

            # Obtain MS-CAM
            with torch.no_grad():
                self.net_main.eval()
                img_05 = F.interpolate(self.img, scale_factor=0.5, mode='bilinear', ...
                    align_corners=True)
                img_10 = self.img
                img_15 = F.interpolate(self.img, scale_factor=1.5, mode='bilinear', ...
                    align_corners=True)
                img_20 = F.interpolate(self.img, scale_factor=2.0, mode='bilinear', ...
                    align_corners=True)

                img_ms = [img_05, img_10, img_15, img_20]

                for k, img in enumerate(img_ms):
                    out = self.net_main(img)
                    cam_temp = F.relu(F.interpolate(out['cam'], size=(H,W), ...
                        mode='bilinear', align_corners=False))
                    cam_temp *= self.label.view(B,C,1,1)

                    if k==0:
                        cam_ms = cam_temp
                    else:
                        cam_ms += cam_temp

                cam_max = F.adaptive_max_pool2d(cam_ms, (1, 1))
                cam_ms = cam_ms / (cam_max + 1e-5) # (B,C,H,W)

            # Sample points from the MS-CAM
            with torch.no_grad():

                img_sam = F.interpolate(denorm(self.img)*255, (self.size_sam, ...
                    self.size_sam), mode='bilinear', align_corners=True)
                img_sam = img_sam.to(torch.uint8)


                # For efficient inference, we get embedding first
                features_sam = self.net_sam(run_encoder_only=True,
                                            transformed_image=img_sam,
                                            original_image_size=(H,W))
                del img_sam
```

```python

        ################################################## Sample local peaks ##################################################
        points_all = {}
        for i in range(B):
            points_img = {}
            for ct in self.label[i].nonzero(as_tuple=False)[:,0]:
                ct = ct.item()

                cam_target = cam_ms[i,ct]

                # Global maximum
                cam_target_f = cam_target.view(-1)
                argmax_indices = torch.argmax(cam_target_f)
                coord_w = argmax_indices // W
                coord_h = argmax_indices % W
                peak_max = torch.cat((coord_w.view(1,1),coord_h.view(1,1)), dim=-1) # (1,2)
                peak_max = peak_max.cpu().detach().numpy()

                # Local maximums
                cam_target_np = cam_target.cpu().detach().numpy()

                cam_filtered = ndi.maximum_filter(cam_target_np, size=3, mode='constant')
                peaks_temp = peak_local_max(cam_filtered, min_distance=50)
                peaks_valid = peaks_temp[cam_target_np[peaks_temp[:,0],peaks_temp[:,1]]>self.th_multi]

                # Aggregate all the peaks
                peaks = np.concatenate((peak_max, peaks_valid[1:]),axis=0) # (NP,2)

                points = np.flip(peaks,axis=(-1)) * self.size_sam / H
                points = torch.from_numpy(points).cuda()
                points_img[ct] = points

            points_all[i] = points_img

        ################################################## Get masks using SAM ##################################################

        sam_conf = -1e5*torch.ones_like(cam_ms)

        for i in range(B):
            for k in points_all[i].keys():
                points = points_all[i][k].unsqueeze(0)
                points_label = torch.ones_like(points[:,:,0])

                output_sam = self.net_sam(run_decoder_only=True,
                                          features_sam=features_sam[i].unsqueeze(0),
                                          original_image_size=(H,W),
```

```
                                        point_coords=points,
                                        point_labels=points_label)

                        mask = output_sam[0] # (1,3,H,W)
                        conf = output_sam[2] # (1,3,H,W)

                        idx_max_sam = 2 # Empirically, 2 is the best.

                        target_mask = mask[0,idx_max_sam]
                        target_conf = conf[0,idx_max_sam].unsqueeze(0).unsqueeze(0)
                        target_conf = F.interpolate(target_conf, (H,W), mode='bilinear', ...
                            align_corners=False)[0,0]

                        # Confidence-based aggregation
                        sam_conf[i,k][target_mask] = target_conf[target_mask] * ...
                            cam_ms[i,k][target_mask].mean() # scalar

                temp = sam_conf.max(dim=1)
                pgt_sam = temp[1]
                pgt_score = temp[0]

                pgt_sam[pgt_score<0] = 20
                pgt_score[pgt_score<0] = 0

        ############################################################ Main branch ...
            ############################################################


        self.net_main.train()
        self.opt_main.zero_grad()

        loss = 0

        out_main = self.net_main(self.img)
        feat_main = out_main['feat']
        cam_main = out_main['cam']
        pred_main = out_main['pred']

        cam_main = F.relu(cam_main)
        cam_max = F.adaptive_max_pool2d(cam_main, (1, 1))
        cam_main = cam_main / (cam_max + 1e-5)
        cam_main = F.interpolate(cam_main, size=(H,W), mode='bilinear', ...
            align_corners=False) * self.label.view(B,C,1,1)

        ############################################################ Loss functions ...
            ############################################################


        # mere classification loss
        #self.loss_cls = 0.1 * self.W[0] * self.bce(pred_main, self.label)
        losscls, idx = self.llct(epo, pred_main, self.label)
        self.loss_cls = self.W[0] * losscls
```

```
        loss += self.loss_cls

        # SAM-Segment Contrasting (SSC)
        feat_main = F.interpolate(feat_main, size=(H,W), mode='bilinear', ...
            align_corners=False)
        feat_main = F.normalize(feat_main, dim=1)
        feat_main_ = feat_main.view(B,D,-1) # (B,D,HW)
        index_ = self.se.view(B,1,-1).long() # (B,1,HW)

        pt = torch_scatter.scatter_mean(feat_main_.detach(), index_) # (B,D,N)
        pt = F.normalize(pt, dim=1)
        index_ = index_.squeeze(1)
        pred_ssc = torch.bmm(pt.permute(0,2,1), feat_main_) # (B,N,HW)

        self.loss_ssc = F.cross_entropy(pred_ssc*self.T, index_, ignore_index=0)
        if not torch.isnan(self.loss_ssc):
            loss += self.loss_ssc
        else:
            print("loss_ssc is NaN!")
            self.loss_ssc = torch.zeros_like(self.loss_cls)

        # CAM-based Prompting Module (CPM)
        if use_cpm:
            cam_bg = 1-cam_main.max(dim=1,keepdims=True)[0]
            cam_main = torch.cat((cam_main, cam_bg), dim=1)
            self.loss_cpm = F.cross_entropy(cam_main, pgt_sam, ignore_index=255)

            if not torch.isnan(self.loss_cpm):
                loss += self.loss_cpm
            else:
                print("loss_cpm is NaN!")
                self.loss_cpm = torch.zeros_like(self.loss_cls)
        else:
            self.loss_cpm = torch.zeros_like(self.loss_cls)

        loss.backward()
        self.opt_main.step()

    # Initialization for msf-infer
    def infer_init(self):
        n_gpus = torch.cuda.device_count()
        self.net_main_replicas = torch.nn.parallel.replicate(self.net_main.module, ...
            list(range(n_gpus)))

    # (Multi-Thread) Infer MSF-CAM and save image/cam_dict/crf_dict
    def infer_multi(self, epo, val_path, dict_path, crf_path, vis=False, dict=False, ...
        crf=False, writer=None):

        if self.phase != 'eval':
            self.set_phase('eval')

        epo_str = str(epo).zfill(3)
```

```
        gt = self.label[0].cpu().detach().numpy()
        self.gt_cls = np.nonzero(gt)[0]

        _, _, H, W = self.img[2].shape
        n_gpus = torch.cuda.device_count()

        def _work(i, img):
            with torch.no_grad():
                with torch.cuda.device(i % n_gpus):
                    out = self.net_main_replicas[i % n_gpus](img.cuda())
                    cam = out['cam']
                    pred = out['pred']
                    cam = F.interpolate(cam, (H, W), mode='bilinear', ...
                        align_corners=False)[0]
                    cam = F.relu(cam)

                    cam = cam.cpu().numpy()
                    cam *= self.label.clone().cpu().view(35, 1, 1).numpy()

                    if i % 2 == 1:
                        cam = np.flip(cam, axis=-1)
                    return cam, pred

        #thread_pool = pyutils.BatchThreader(_work, list(enumerate(self.img)), ...
            batch_size=2, prefetch_size=0, processes=2)

        cam_list = []
        pred_list = []
        for i, img in enumerate(self.img):
            cam, pred = _work(i, img)
            cam_list.append(cam)
            pred_list.append(pred.cpu())

        cam = np.sum(cam_list, axis=0)
        cam_max = np.max(cam, (1, 2), keepdims=True)
        norm_cam = cam / (cam_max + 1e-5)

        stacked_preds = torch.stack(pred_list)
        mean_pred = torch.mean(stacked_preds, axis=0)
        pred_list = mean_pred.cpu().numpy()

        self.cam_dict = {}
        for i in range(35):
            if self.label[0, i] > 1e-5:
                self.cam_dict[i] = norm_cam[i]

        with open('pred_y.txt', 'a') as f:
            np.savetxt(f, pred_list, newline='\n')
        with open('true_y.txt', 'a') as f:
            np.savetxt(f, self.label.cpu(), newline='\n')
        if vis and (self.name == "val/lindau/lindau_000016_000019_leftImg8bit.png" or ...
            self.name == "val/munster/munster_000172_000019_leftImg8bit.png"):
```

```
            img_np = denorm(self.img[2][0]).cpu().detach().numpy()
            for c in self.gt_cls:
                temp = cam_on_image(img_np, norm_cam[c])
                names = self.name.split('/')[2]
                names = names[0:-4]
                temp_path = osp.join(val_path, epo_str + '_' + names+ '_cam_' + ...
                    self.categories[c] + '.png')
                plt.imsave(temp_path, np.transpose(temp, (1,2,0)))
                if writer is not None:
                    writer.add_image(self.name+'/'+self.categories[c], temp, epo)

        if dict:
            names = self.name.split('/')[2]
            names = names[0:-4]
            np.save(osp.join(dict_path, names + '.npy'), self.cam_dict)
```