Computer Vision

Multi-label Image Classification

Group 25
Riajul Islam, Andreas Calonius Kreth, Christine Midtgaard

***Abstract*—Abstract goes here.**

***Index Terms*—Multi-label learning, deep learning, computer vision, multi-label classification, deep learning for MLC.**

I. Introduction

Multi-label classification is the supervised learning problem where an instance may be associated with multiple labels. Image classification is a computer vision task that requires assigning a label or multiple labels to an image. Single-label classification, or multi-class classification, refers to the problem where an image contains only one object to be identified. However, natural images usually contain multiple objects or concepts, highlighting the importance of multi-label classification [1].

In this project we investigate methods aimed to solve two common issues that arise when training a network that assigns multiple labels to an input image: (i) the general multi-label learning issue accurately identifying multiple objects in an image under class imbalance, representing the complexity of real-world images, and (ii) the scenario where the training data underwent sparse supervision.

A. Problem Overview

a) Label Correlation:
b) Class Imbalance:
c) Sparse Supervision:

II. Related Work

Multi-label learning is a well studied problem within computer vision [2].

a) Loss Functions:

b) PU learning: Learning from positive and unlabeled data: a survey by Bekker and Davis [3].

Learning to Classify Texts Using Positive and Unlabeled Data by Li and Liu [4].

c) Partially Observed Labels:

d) Heuristics: Heuristics can be used to reduce the required annotation effort [34, 18], but this runs the risk of increasing error in the labels [2].

III. Background

This section describes relevant background theory related to the project and methods from MLSPL and Query2Label. Beginning with the definition of multi-label learning, followed by deep neural network architectures used to solve multi-label slassification tasks, and, finally, loss functions as they can handle challenges of label imbalance tasks.

A. Multi-Label Learning

Contrary to binary or multi-class classification where each instance is associated with only one label, multi-label classification allows assigning multiple labels to a single input [2]. Given K categories, an input image $x \in \mathcal{X}$ is associated with a binary vector of labels $y = [y_1, ..., y_K]$ from the label space $\mathcal{Y} = \{0, 1\}^K$, where $y_k = 1$ represents that k is present in x and $y_k = 0$ otherwise. The goal is to create a model that outputs the probability of the presence of a category $p = [p_1, ..., p_K]$ [2], [5].

B. Convolutional Neural Networks (CNNs)

In the field of computer vision, Convolutional Neural Networks (CNNs) have been a dominant approach for image analysis tasks since their introduction [6]. CNNs are designed to recognize patterns in images by applying local filters through convolutional layers, preserving the two dimensional input of an image [7]. CNNs consist of three main types of layers: convolutional layers, pooling layers, and a Fully Connected (FC) layer [8]. Convolutional layers extract spatial features through learnable filters, pooling layers reduce dimensionality and summarise information, while the fully connected layer at the end interpret the features to perform the classification.

C. Transformer Architectures

Transformers, introduced by Vaswani et al. [9], are a type of neural network architecture initially developed to model long-range dependencies in sequence data. They achieve this using attention and self-attention mechanisms, which enables the model to have a long-term memory of inputs, and dynamically weigh the importance of each element in the input sequence. Originally, transformer-based models was mainly delevoped for natural language processing tasks, but the introduction of the Vision Transformer (ViT) by Dosovitskiy et al. [10], transformers have been widely used for computer vision tasks.

a) Architecture: The original transformer consists of an encoder-decoder structure. Both encoder and decoder are made of a stack of identical layers, where encoder layers contain a multi-head self-attention mechanism followed by a position-wise feed-forward network. In addition, decoder layers consists of an encoder-decoder attention layer.

The transfomer encoder is of interest, as the authors of [5] use them to extract features and automatically learn label embedding, as described in section IV-B.

b) Attention: The core principle of the transformer architecture is the attention mechanism, which allows the model to attend to all posistions in an input sequence when processing each element. Thus alloowing the model to weigh the relevance of different posistions in a sequence. Given a query and a set of key-value pairs, the attention function computes a weighted sum of the values, where the weights are determined by similarities between the query and the keys.

c) Self-Attention: The transformer model uses self-attention by relating every element in the input sequence to every other element. The attention function is a function that maps a query and a set of key-value pairs to an output. The scaled dot-product self-attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where Q, K, V, are the query, key, and value vectors, and d_k is the dimension of the key vectors and serves as a scaling factor [9].

d) Multi-Head Attention: The transformer applies multiple attention functions in parallel, allowing the model to attend to information from different parts of the sequence. The embedding is split into multiple heads, perform attention for each, and then concatenate them back together, defined as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{2}$$

where each head is computed as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{3}$$

with projection matrices W_i^Q, W_i^K, W_i^V, and W_i^O.

e) Cross-Attention: Contrary to self-attention, where queries, keys, and values are generated from the same input, cross-attention is a mechanisn where queries come from one source, and the keys and values come fron another. This mechanism allows the model to relate elements from one input to relevant parts of another input. The authours of Query2Label [5] make use of this mechanism, where label embeddings (queries) attend to spatial image features (key-value pairs).

f) Feed-Forward Networks and Positional Encoding: Each layer in the Transformer also includes a fully connected feed-forward network applied to each position. To compensate for the lack of order in the input sequence, sinusoidal positional encodings are added to the input embeddings. These encodings allow the model to distinguish the order of elements in the sequence using functions of varying frequencies.

g) Vision Transformers (ViTs): The Vision Transformer (ViT), introduced by Dosovitskiy et al. [10], is a model for image classification that leverages the transformer architecture: the self-attention mechanism of transformers that allows the model to selectively weigh the significance of each part of the input, and positional embeddings that represent the order of tokens in a sequence. In ViTs, images are represented as sequences, where the label function as a learnable token for classification. The input image is divided into a sequence of patches that are flattened and linearly embedded into a vector. The spatial information is preserved by adding positional encodings to the embeddings. The resulting sequence is fed into a transformer decoder identical to that introduced in [9] to model global relations for classification.

D. Loss Functions

Loss functions are a fundamental component in deep learning as they serve as a measure of how much the model predictions deviate from the ground truth, guiding optimization of the network [7]. The goal of optimization is to minimize the loss. In the context of multi-label learning, each instance can belong to multiple classes simumtaneously, and selecting an appropriate loss function is crucial. This section aims to decribe different loss functions employed in multi-label classification, and how they can be used to solve common issues in multi-label learning.

a) Binary Cross-Entropy (BCE): Due to its effectiveness in handling binary decisions for each class, a common choice of loss function for multi-label classification is the The Binary Cross-Entropy (BCE) loss [2], [11], [12]. In binary classification, the goal is to classify data as either positive or negative, represented as 0 or 1. The output of the model is a probability score between 0 and 1, indicating the probability of the input belonging to the positive class [12].

The BCE loss for multi-label classification is given by:

$$\begin{aligned}\mathcal{L}_{\text{BCE}} = -\frac{1}{K}\sum_{i=1}^{K}\big[&y_i \log(p_i) \\ &+(1-y_i)\log(1-p_i)\big]\end{aligned} \tag{4}$$

where K is the number of categories, $y_i \in [0, 1]$ is the binary label, and p_i is the predicted probability.

Due is its popularity in multi-class classification the BCE loss is used as a benchmark for model performance in [2]. However, BCE loss can be sensitive to class imbalance, where some classes appear more often than others. In this case, the model tend to focus on the majority class and perform poorly on the minority class.

b) Loss Functions for the Partially Observed Labels Setting: BCE assumes that each training example indicates the presence or absence of each class, e.i. the example is fully observable. In real-world multi-label datasets, it is often infeasible to annotate all relevant labels for each image, resulting in partially observed labels. In the case where labels are only partially observed, the assumption of full negative labeling in BCE can lead to false negatives during training, degrading model performance [2].

To address this issue, several loss functions have been proposed to handle partial supervision. In this project, we explore the case of minimal supervision: the single positive label setting, where only one relevant class label is observed per image and the rest are unannotated.

Several loss functions have been proposed to mitigate the issue of false negatives, including the Assume Negative loss, Weak Assume Negative loss, and Expected Positive Regularization, all covered in [2]. While these approaches relax the assumptions on the unobserved labels, they still depend on fixed heuristics.

To address these limitations, the authors of [2] propose Regularized Online Label Estimation (ROLE). This method jointly learns the classifier and a label estimator, which maintains soft estimates of the full label vector during training, meaning that ROLE assumes unobserved labels of a probability between 1 and 0. By encouraging consistency between predictions, known labels, and an expected number of positive labels per image, the model can recover many of the unobserved true labels.

c) Regularized Online Label Estimation (ROLE): ROLE introduces a second network to estimate the unobserved labels online during training. These soft label estimates are updated alongside the main classifier using a joint optimization procedure, where the two components reinforce eachother.
The ROLE loss is given by:

$$\mathcal{L}_{ROLE}(\mathbf{F}_B, \tilde{\mathbf{Y}}_B) = \frac{\mathcal{L}'(\mathbf{F}_B)|\tilde{\mathbf{Y}} + \mathcal{L}'(\tilde{\mathbf{Y}}|\mathbf{F}_B)}{2} \tag{5}$$

for a batch B, where $\tilde{\mathbf{Y}} \in [0,1]^{N\times L}$ represent the estimated labels, and $\mathbf{F} \in [0,1]^{N\times L}$ is the matrix of classifier predictions.

IV. Method

In real-world datasets, obtaining full label annotations is practiaclly impossible. This project focuses on finding solutions to the multi-label learning problems:

- The challenge of multi-label learning in scenarios where only a single positive label per image is available durin training (MLSPL).
- Label imbalance (Query2Label).
- Feature localization (Query2Label).

A. Overview of Approach

A brief description of what we did to solve the multi-label classification problem.

B. Query2Label: A Simple Transformer Way to Multi-Label Classification

Query2Label: A Simple Transformer Way to Multi-Label Classification (Query2Label) by Liu et al. [5] is a two-stage framework for multi-label classification. It uses transformer decoders to extract features with multi-head attentions focusing on different parts of an object category and learn label embeddings from data automatically.

C. Multi-Label Learning from Single Positive Labels

MLSPL's solution to a sparsely annotated training dataset is twofold: first, they develop a training methodology using datasets where annotators have only provided a single confirmed positive label per image, with no confirmed negatives. Second, they extend existing multi-label loss functions to handle this challenging scenario and introduce novel variants to constrain the expected number of positive labels.

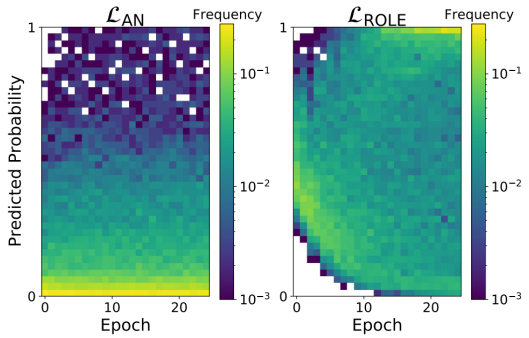

Fig. 1. Figure 2 in MLSPL [2]. The distribution of predicted probabilities for unobserved positives when training with a single positive per image for COCO.

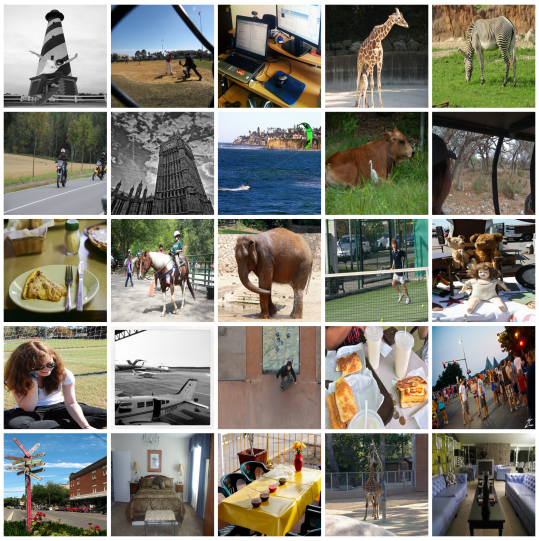

Fig. 2. Examples from the MS-COCO 2014 training set, resized to 448×448 pixels.

V. Experiments

All experiments were conducted on a cluster with access to two NVIDIA GeForce RTX 3060 GPU (12 GB VRAM). The cluster was equipped with Ubuntu 22.04.5 LTS.

A. Dataset

The MS-COCO 2014 [13] dataset is used as a benchmark for evaluation both Query2Label and MLSPL. MS-COCO (Microsoft Common Objects in Context) is a large-scale dataset commonly used for object detection, segmentation, and multi-label image classification. COCO consists of 82,081 training images and 80 classes, and a validation set of 40,137 images.

B. Evaluation Metrics

To asses model performance, we adopt mean Average Precision (mAP), a standard metric widely reported in multi-label classification tasks as it is used to analyze the performance on object detection and segmentation. Both Query2Label and MLSPL report results in terms of mAP.

TABLE I
COMPARISON OF MAP RESULTS FROM OUR EXPERIMENTS AND THE REPORTED RESULTS ON QUERY2LABEL ON THE MS-COCO 2014 DATASET.

Method	Backbone	Resolution	mAP(Ours)	mAP (Paper)
Q2L-R101-448	ResNet-101	448×448	84.9	84.9
Q2L-R101-576	ResNet-101	576×576	86.5	86.5
Q2L-SwinL	Swin-L(22k)	384×384	90.5	90.5
Q2L-CvT	CvT-w24(22k)	384×384	91.3	91.3
Q2L-TresL	TResNetL	448×448	87.3	87.3
Q2L-Tres	TResNetL(22k)	448×448	89.2	89.2

C. Query2Label

All Query2Label experiments were conducted using Python 3.7.3 with PyTorch 1.9.0 and Torchvision 0.10.0, compiled with CUDA 11.1. Due to compatibility issues with the `inplace_abn` dependency required by TResNet backbones, we rebuilt the extension from source using commit `938ffd2` to ensure compatibility with our environment.

Additionally, we encountered a missing dependency for RandAugment, which was not included in the original repository. To address this, we manually added the augmentations.py file from the pytorch-randaugment implementation[1] and modified the import in get_dataset.py to use a local version: from .augmentations import RandAugment. We also replaced the default RandAugment() instantiation with RandAugment(n=2, m=9) to explicitly set the parameters. While a similar implementation exists in the torchvision.transforms module[2], it is incompatible with the version of torchvision used in this project.

a) Data preparation: While the original work includes experiments on five datasets, our reproduction is limited to the MS-COCO 2014 dataset. The model was evaluated on the MS-COCO 2014 dataset. All images were resized to match the input resolution required by each respective backbone. Label annotations for each image were preserved in their multi-label format.

b) Training: Attempts to train Query2Label with the CvT-w24 backbone failed due to memory limitations of the available 12GB GPUs. However, due to time constraints, we did not replicate the original training procedure using our own setup, and only tested if training was possible with the limited memory availability, and instead, we relied on the publicly released pretrained models provided by the authors. The Query2Label framework was evaluated using four backbone configurations: ResNet-101 at resolutions of 448×448 and 576×576, as well as Swin-L(22k) and CvT-w24(22k) at 384×384, both pretrained on the ImageNet-22k [14] dataset. These models were originally trained on the MS-COCO 2014 training set and evaluated in our experiments on the MS-COCO 2014 validation set. We used a batch size of 16 and made no further modifications or fine-tuning.

We tested training the Query2Label model from scratch using the ResNet-101 backbone. Distributed training was enabled to leverage both GPUs. Training was conducted with a batch size of 8, chosen to fit within the memory constraints of the GPUs. The model was optimized using AdamW with a learning rate of 10^{-4}, weight decay of 10^{-2}, and trained for 80 epochs with early stopping enabled. Data augmentation included random cutout with one hole and a cut factor of 0.5. Four data loading workers were used per process. No modifications were made to the original model architecture or loss design.

[1]https://github.com/ildoonet/pytorch-randaugment

[2]https://pytorch.org/vision/main/generated/torchvision.transforms.RandAugment.html

D. Multi-Label Learning from Single Positive Labels

The original implementation was developed with older library versions, specifically PyTorch 1.7.0+cu110, and Torchvision 0.8.1+cu110. In contrast, our reproduction was conducted using PyTorch 2.2.1+cu121, and Torchvision 0.17.1+cu121. The original implementation had unknown versions of Python and CUDA, whereas our experiments were conducted using python 3.11.8 and CUDA 12.4. Despite keeping the code unchanged, and using the original random seed, differences in default behavior between library versions may cause deviations in results. Nevertheless, all code ran without modification, and we verified that training and evaluation completed successfully.

Although two GPUs were available on the system, only a single GPU was utilized during training to maintain consistency and reproducibility of the results.

a) Data preparation: While the original work includes experiments on four datasets, our reproduction is limited to the MS-COCO 2014 dataset. The COCO dataset was converted to a single-positive label format to simulate a weakly supervised setting. Following Cole et al. [2], this was done by starting with the fully annotated multi-label dataset and corrupting it by discarding all but one randomly selected positive label per image. Importantly, the same selected label is consistently used throughout training to ensure fair comparisons across methods. Additionally, 20% of the training data was set aside for validation. Both the validation and test sets remain fully labeled.

To prepare the data, we followed the authors' instructions: first, the standard COCO 2014 training and validation images and annotations were downloaded. Next, we obtained the pre-extracted features provided by the authors. Finally, the `format_coco.py` script was used to produce uniformly formatted image lists and corresponding labels.

b) Training: Results are reported on two training scenarios: (i) linear classification on fixed features and (ii) end-to-end fine-tuning. For each method, a hyperparameter search

was conducted, and results are reported for the configuration achieving the best mean average precision (mAP) on the validation set.

In replicating the experiments, we conducted a similar hyperparameter search, considering learning rates $\{1e{-}2, 1e{-}3, 1e-4, 1e-5\}$ and batch sizes $8, 16$. The linear classifier was trained for 25 epochs, and the fine-tuning phase for 10 epochs. The validation set used was the clean variant, meaning it was fully labeled.

c) Selected Hyperparameters: The final results reported for MS-COCO were obtained using the hyperparameters that achieved the highest validation mAP during the search. For the linear classifier, we used a batch size of 16 and a learning rate of 0.001. For the fine-tuned ResNet-50 model, the learning rate was set to 0.00001, with the same batch size of 16.

VI. Results and Discussion

This section compares results from the experiments to the those of the papers. Discuss why they might not be the same, or describe the similarities. Discuss whether the methods are able to solve the problems.

A. Results from Query2Label

The results from our experiments are compared to those of the paper in table I.

B. Results from Multi-Label Learning from Single Positive Labels

The results from our experiments are compared to those of the paper in table II.

TABLE II
Comparison of mAP results from our experiments and the reported results on Multi-Label Learning from Single Positive Labels on the MS-COCO 2014 Dataset.

Loss	Method	mAP(Ours)	mAP (Paper)
$\mathcal{L}_{ROLE}$	Linear	66.3	66.3
$\mathcal{L}_{ROLE}$	Fine-Tuned	66.9	66.3

VII. Conclusion

Conclusion here.

References

[1] T. Ridnik, G. Sharir, A. Ben-Cohen, E. Ben-Baruch, and A. Noy, "Ml-decoder: Scalable and versatile classification head," 2021. [Online]. Available: https://arxiv.org/abs/2111.12933

[2] E. Cole, O. M. Aodha, T. Lorieul, P. Perona, D. Morris, and N. Jojic, "Multi-label learning from single positive labels," 2021. [Online]. Available: https://arxiv.org/abs/2106.09708

[3] J. Bekker and J. Davis, "Learning from positive and unlabeled data: a survey," *Machine Learning*, vol. 109, no. 4, p. 719–760, Apr. 2020. [Online]. Available: http://dx.doi.org/10.1007/s10994-020-05877-5

[4] X. Li and B. Liu, "Learning to classify texts using positive and unlabeled data." 01 2003, pp. 587–594.

[5] S. Liu, L. Zhang, X. Yang, H. Su, and J. Zhu, "Query2label: A simple transformer way to multi-label classification," 2021. [Online]. Available: https://arxiv.org/abs/2107.10834

[6] Y. Lecun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison of learning algorithms for handwritten digit recognition," 01 1995.

[7] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*. Cambridge University Press, 2023. [Online]. Available: https://d2l.ai

[8] X. L. C. Asawa, *CS231n: Convolutional Neural Networks for Visual Recognition*. Stanford University, 2023. [Online]. Available: http://cs231n.github.io/

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[11] T. Durand, N. Mehrasa, and G. Mori, "Learning a deep convnet for multi-label classification with partial labels," 2019. [Online]. Available: https://arxiv.org/abs/1902.09720

[12] N. Nayan, "Binary cross entropy — machine learning," https://medium.com/@neerajnan/binary-cross-entropy-machine-learning-1c5dee1f2d52, Apr. 2024, medium.

[13] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015. [Online]. Available: https://arxiv.org/abs/1405.0312

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.