Combining Metric Learning and Attention Heads For Accurate and Efficient Multilabel Image Classification

Kirill Prokofiev$^{[0000-0001-9619-0248]1}$ and
Vladislav Sovrasov$^{[0000-0001-6525-2602]2}$

[1] Intel, Munich, Germany kirill.prokofiev@intel.com,
[2] Intel, Munich, Germany vladislav.sovrasov@intel.com

Abstract. Multi-label image classification allows predicting a set of labels from a given image. Unlike multiclass classification, where only one label per image is assigned, such setup is applicable for a broader range of applications. In this work we revisit two popular approaches to multilabel classification: transformer-based heads and labels relations information graph processing branches. Although transformer-based heads are considered to achieve better results than graph-based branches, we argue that with the proper training strategy graph-based methods can demonstrate just a small accuracy drop, while spending less computational resources on inference. In our training strategy, instead of Asymmetric Loss (ASL), which is the de-facto standard for multilabel classification, we introduce its modification acting in the angle space. It implicitly learns a proxy feature vector on the unit hypersphere for each class, providing a better discrimination ability, than binary cross entropy loss does on unnormalized features. With the proposed loss and training strategy, we obtain SOTA results among single modality methods on widespread multilabel classification benchmarks such as MS-COCO, PASCAL-VOC, NUS-Wide and Visual Genome 500. Source code of our method is available as a part of the OpenVINO™ Training Extensions[3].

Keywords: Multilabel image classification · deep learning · lightweight models · graph attention

1 Introduction

Starting from the impressive AlexNet [19] breakthrough on the ImageNet benchmark [6], deep-learning era has drastically changed approaches to almost every computer vision task. Throughout this process multiclass classification problem was a polygon for developing new architectures [14, 16, 30] and learning paradigms [2, 13, 17]. At the same time, multilabel classification has been developing not so intensively, although the presence of several labels on one image is more natural, than having one hard label. Due to lack of specialized multilabel datasets, researchers turned general object detection datasets such as

[3] https://github.com/openvinotoolkit/deep-object-reid/tree/multilabel

MS-COCO [22] and PASCAL VOC [9] into challenging multilabel classification benchmarks by removing bounding boxes from the data annotation and leveraging only their class labels.

Conventionally, multilabel classification task is transformed into a set of binary classification tasks, which are solved by optimizing a BCE-like loss function. This formulation leads to the following two key challenges: positive-negative imbalance in each binary subtask and modeling the inter-label dependencies. We tackle the imbalance challenge by applying a metric learning loss inspired by SphereFace2 [34] loss. By its nature, this loss performs hard sample mining by normalizing features and scaling logits. Combining the metric learning approach with ASL [1] focal weighting provides even stronger mechanism for mining hard positives and discarding easy negatives.

Implicit label co-occurrence modeling is also beneficial for multilabel classification [3,23,27]. Baseline classification architectures consist of a backbone (CNN or transformer), global pooling layer and linear layer with binary classification losses attached to each logit. This architecture can be augmented by a set of binary classifiers generated by stacked GCNs or graph attention layers, which process a graph built over vector representations of labels [3,35]. Applying graph attention layers enables utilizing labels co-occurrences to generate more discriminative classifiers. We incorporate the output of the graph attention subnetwork in a slightly different manner, which allow us to combine this approach with transformer-based classification heads.

The main contributions of this paper are as follows:

- We proposed a modification of ML-GCN that adds graph attention operations [32] and performs graph and CNN features fusion in a more conventional way than generating a set of binary classifiers in the graph branch.
- We demonstrated that using a proper training strategy, one can decrease the performance gap between transformer-based heads and label co-occurrence modeling via graph attention.
- We studied the importance of per class confidence thresholds tuning for adapting models to real-world applications.
- We first applied the metric learning paradigm to multilabel classification task and proposed a modified version of angular margin binary loss [34], which adds ASL [1] mechanism to it.
- We verified the effectiveness of our loss and overall training strategy with comprehensive experiments on widespread multilabel classification benchmarks: PASCAL VOC, MS-COCO, Visual Genome [18] and NUS-WIDE [4].

2 Related Work

Historically multilabel classification was attracting less attention than the multiclass scenario, but nonetheless there is stills a great progress on that field. Notable progress was achieved by developing advanced loss functions [1], label co-occurrence modeling [3,35], designing advanced classification heads [23,27,36] and discovering architectures taking into account spatial distribution of objects via exploring attentional regions [11, 33].

Each single-class classification subtask in multilabel scenario suffers from hard positives-negatives imbalance. More classes the training dataset contains, more negatives we have for in each of the single-class subtasks, because an image typically contains a tiny fraction of the vast number of all classes. A modified asymmetric loss [1], that down-weights and hard-thresholds easy negative samples, showed impressive results, reaching state-of-the-art results on multiple popular multi-label datasets without any sophisticated architecture tricks. These results indicate that a proper choice of a loss function is crucial for multilabel classification performance.

Another promising direction is designing class-specific classifiers instead of using a fully connected layer on top of a single feature vector produced by a backbone network. This approach also doesn't introduce additional training steps and marginally increases model complexity. Authors of [36] propose a drop-in replacement of global average pooling layer that generates class-specific features for every category. Leveraging compact transformer heads for generating such features [23, 27] turned out to be even more effective. This approach assumes pooling class-specific features by employing learnable embedding queries.

Taking into account spatial a distribution of objects or label relationships based on prior knowledge requires data pre-processing and additional assumptions [3, 35] or sophisticated model architecture [11, 33]. For instance, [3, 35] represents labels by word embeddings, then a directed graph is built over these label representations, where each node denotes a label. Then stacked GCNs are learned over this graph to obtain a set of object classifiers. The method relies on an ability to represent labels as words, which is not always possible. Spatial distribution modeling requires placing a RCNN-like [12] module inside the model [11, 33], which drastically increases complexity of the training pipeline.

3 Method

In this section, we describe the overall training pipeline and details of our approach. We aimed not only to achieve competitive results, but also to make the training more friendly to the end user and adaptive to data. Thus, following principles described in [25], we use lightweight model architectures, hyperparameters optimization and early stopping.

3.1 Model architecture

We chose EfficientNetV2 [31] and TResNet [26] as base architectures for performing multilabel image classification. Namely, we conducted all the experiments on TResNet-L, EfficientNetV2 small and large. On top of these backbones we used several different features aggregation approaches.

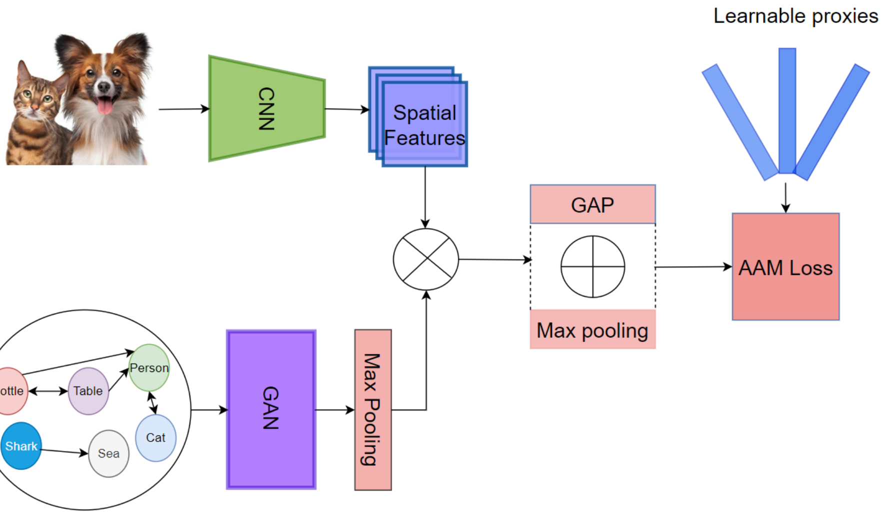


Fig. 1: Re-weighting scheme for incorporating labels semantic and labels co-occurrence information.

3.2 Transformer multilabel classification head

As a representor of transformer-based feature aggregation methods we use the ML-Decoder [27] head. It provides up to K feature vectors (where K is the number of classes) as a model output instead of a single class-agnostic vector when using a standard global average pooling (GAP) head. Let's denote $x \in \mathbb{R}^{C \times H \times W}$ as a model input, then the model F with parameters W produces a downscaled multi-channel featuremap $f = F_W(x) \in \mathbb{R}^{S \times \frac{H}{d} \times \frac{W}{d}}$, where S is the number of output channels, d is the spatial downscale factor. That featuremap is then passed to the ML-Decoder head: $v = MLD(f) \in \mathbb{R}^{M \times L}$, where M is the embedding dimension, $L \leq K$ is the number of groups in decoder. Finally, vectors v are projected to K class logits by a fully-connected (if $L = K$) or group fully-connected projection (if $L < K$) as it is described in [27]. In our experiments we set $L = \min(100, K)$. Also we normalize the arguments of all dot products in projections in case if we need to operate with features on the unit hypersphere.

3.3 Graph attention multilabel branch

The original structure of the graph processing branch from [3] supposes generating classifiers right in this branch and then applying them directly to the features generated by a backbone. This approach is incompatible with the transformer-based head or any other processing of the raw spatial features f, like CSRA [36]. To alleviate this limitation, we propose the architecture showed in Figure 1.

First, we generate the label correlation matrix $Z \in \mathbb{R}^{K \times K}$ in the same way as in [3]. Together with the word embeddings $G \in \mathbb{R}^{K \times N}$, obtained with GLOVE

[24] model, where $N = 300$ is word embeddings dimension, we utilize Z as an input of the Graph Attention Network [32]. Unlike [35], we use estimations of conditional probabilities to build Z, rather that fully rely on GLOVE and compute cosine similarities.

We process the input with the graph attention layers and obtain output $h \in \mathbb{R}^{S \times K}$. Then we derive the most influential features through the max pooling operation and receive the weights $w \in \mathbb{R}^S$ for further re-weighting of the CNN spatial features: $\tilde{f} = w \bigodot f$. Next, we apply global average pooling and max pooling operations to $\tilde{f}$ in parallel, sum the results and obtain final latent embedding $\tilde{v} \in \mathbb{R}^S$. The embedding $\tilde{v}$ is finally passed to the binary classifiers. Instead of applying a simple spatial pooling, we can pass the weighted features $\tilde{f}$ to the ML-Decoder or any other features processing module.

The main advantage of using the graph attention branch for re-weighting or classification over transformer head is a tiny computational and model complexity overhead at the inference stage. Since graph attention branch has the same input for any image, we can compute the result of its execution only once, before starting the inference of the resulting model. At the same time, the graph attention branch requires a vector representation of labels. Such representations can be generated by a text-to-vec model in case we have a meaningful descriptions for all labels (even single word ones). This condition doesn't always hold: some dataset could have untitled labels. How to generate representations for labels in that case is still an open question.

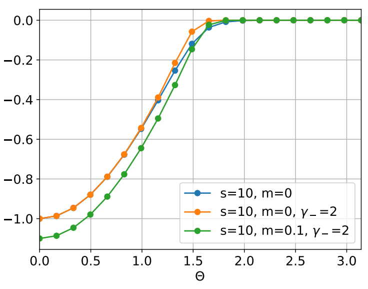
3.4 Angular margin binary classification

Recently Asymmetric loss [1] has become a standard loss option for performing multi-label classification. By design it penalizes each logit with a modified binary cross-entropy (BCE) loss. Asymmetric handling of positives and negatives allows ASL to down-weight the negative part of the loss to tackle the positives-negatives imbalance problem. But this approach leaves a room for improvement from the model's discriminative ability perspective.

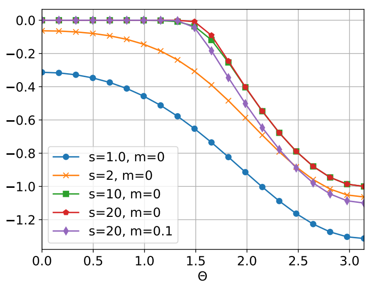
Angular margin losses are known for generating more discriminative classification features than the cross-entropy loss, which is a must-have property for recognition tasks [7, 29, 34]. We propose joining paradigms from [1] and [34] to build even stronger loss for multilabel classification. Denote the result of the dot product between the normalized class embedding v_j and the j-th binary classifier W_j as $cos\Theta_j$. Then, for a training sample x and corresponding embeddings set v we formulate our asymmetric angular margin loss (AAM) as:

$$L_{AAM}(v, y) = -\sum_{j=1}^{K} L_j(cos\Theta_j, y)$$

$$L_j(cos\Theta_j, y) = \frac{k}{s} y p_-^{\gamma^-} \log p_+ + \frac{1-k}{s}(1-y) p_+^{\gamma^+} \log p_-, \tag{1}$$



(a) Loss for p_-



(b) Loss for p_+

Fig. 2: Plots of positive and negative parts of AAM with varying hyperparameters

$$p_+ = \sigma(s(cos\Theta_j - m)),$$
$$p_- = \sigma(-s(cos\Theta_j + m)),$$

where s is a scale parameter, m is an angular margin, k is negative-positive weighting coefficient, γ^+ and γ^- are weighting parameters from ASL. Despite on a large number of hyperparameters, some of them could be safely fixed (like γ^+ and γ^- from ASL). The effect of varying s saturates when increasing s (see Figure 2b), and if the suitable value of this parameter is large enough, we don't need to tune it precisely. Also values of m should be close to 0, because it duplicates to some extent the effect of s and γ and can even bring undesirable increase of the negative part of AAM (see Figure 2a). Detailed analysis of the hyperparameters is provided in Section 4.6.

3.5 Details of the training strategy

As in our former work [25], we aim to make the training pipeline for multilabel classification reliable, fast and adaptive to dataset, so we use the following components:

- SAM [10] optimizer with no bias decay [15] is a default optimizer;
- EMA weights averaging for an additional protection from overfitting;
- Initial learning rate estimation process from [25];
- OneCycle [28] learning rate scheduler;
- Early stopping heuristic: if the best result on the validation subset hasn't been improved during 5 epochs, and evaluation results stay below the EMA averaged sequence of the previous best results, the training process stops;
- Random flip, pre-defined Randaugment [5] strategy and Cutout [8] data augmentations.

4 Experiments

In this section, we conduct a comparison of our approach with the state-of-the-art solutions on popular multilabel benchmarks. Besides the mAP metric, we present the GFLOPs for each method to consider the speed/accuracy trade-off. Also, we show the results of the ablation study to reveal the importance of each training pipeline component.

4.1 Datasets

To compare performance of different methods we picked up widespread datasets for multilabel classification, that are listed in Table 1. According to the table, all of them have a noticeable positives-negatives imbalance on each image: average number of presented labels per images is significantly lower than the number of classes.

Also, we use a precisely annotated subset of OpenImages V6 [20] for pretraining purposes. We join the original training and testing parts into one train subset and use the original validation subset for testing.

Table 1: Image classification datasets which were used for training.

Dataset	# of classes	# of Images		Avg # of
		Train	Validation	labels per image
Pascal VOC 2007 [9]	20	5011	4952	1.58
MS-COCO 2014 [22]	80	117266	4952	2.92
NUS-WIDE [4]	81	119103	50720	2.43
Visual Genome [18]	500	82904	10000	13.61
OpenImages V6 [20]	601	1866950	41151	5.23

4.2 Evaluation protocol

We adopt commonly used metrics for evaluation of multilabel classification models: mean average precision (mAP) over all categories, overall precision (OP), recall (OR), F1-measure (OF1) and per-category precision (CP), recall (CR), F1-measure (CF1). We use the mAP as a main metric, others are provided when an advanced comparison of approaches is conducted. In every operation where the confidence thresholding is required, threshold 0.5 is substituted. The exact formulas of the mentioned metrics can be found in [23].

4.3 Pretraining

Although the multilabel classification task is similar to multiclass classification, a multilabel model should pay attention to several objects on an image, instead of concentrating its attention at one object. Thus, additional task specific pretraining a on large-scale dataset looks beneficial to the standard ImageNet pretraining, and it was shown in recent works [27]. In this work we utilize OpenImages V6 for pretraining. According to our experiments, the precisely-annotated subset containing 1.8M images is enough to get a substantial improvement over the ImageNet weights.

To obtain pretrained weights on OpenImages for our models, we use the ML-Decoder head, 224×224 input resolution, ASL loss with $\gamma^+ = 0$ and $\gamma^- = 7$, learning rate 0.001, and 50 epochs of training with OneCycle scheduler.

For TResNet-L we use the weights provided with source code in [27].

4.4 Comparison with the state-of-the-art

In all cases where we use EfficientNet-V2-s as a backbone we also use our training strategy from Section 3.5 for a fair comparison. For ASL loss [1] we set $lr = 0.0001$, $\gamma^- = 4$, $\gamma^+ = 0$ as suggested in the original paper [1]. We share the following hyperparameters across all the experiments: $m = 0.0$, $k = 0.7$, EMA decay factor equals to 0.9997, $\rho = 0.05$ for the SAM optimizer. Other hyperperameters for particular datasets were found empirically or via coarse grid search.

In Table 2 results on MS-COCO are presented. For this dataset we set $s = 23$, $lr = 0.007$, $\gamma^- = 1$, $\gamma^+ = 0$ (see Sec. 4.6). With our AAM loss, we can achieve a state-of-the-art result using TResNet-L as a backbone. At the same time, combination of EfficientNetV2-s with ML-Decoder and AAM loss outperforms TResNet-L with ASL, while consuming 3.5x less FLOPS. GCN/GAN branches performs slightly worse than ML-Decoder, but still improves results over EfficientNetV2-s + ASL with a marginal computational cost at inference.

Table 2: Comparison with the state-of-the-art on MS-COCO dataset.

Method	Backbone	Input resolution	mAP	GFLOPs
ASL [1]	TResNet-L	448x448	88.40	43.50
Q2L [23]	TResNet-L	448x448	89.20	60.40
GATN [35]	ResNeXt-101	448x448	89.30	36.00
Q2L [23]	TResNet-L	640x640	90.30	119.69
ML-Decoder [27]	TResNet-L	448x448	90.00	36.15
ML-Decoder [27]	TResNet-L	640x640	91.10	73.42
ASL*	EfficientNet-V2-s	448x448	87.05	10.83
ML-GCN*	EfficientNet-V2-s	448x448	87.50	10.83
Q2L*	EfficientNet-V2-s	448x448	87.35	16.25
ML-Decoder*	EfficientNet-V2-s	448x448	88.25	12.28
GAN re-weighting (ours)	EfficientNet-V2-s	448x448	87.70	**10.83**
GAN re-weighting (ours)	TResNet-L	448x448	89.95	35.20
ML-Decoder + AAM (ours)	EfficientNet-V2-s	448x448	88.75	12.28
ML-Decoder + AAM (ours)	EfficientNet-V2-L	448x448	90.10	49.92
ML-Decoder + AAM (ours)	TResNet-L	448x448	**90.30**	36.15
ML-Decoder + AAM (ours)	TResNet-L	640x640	**91.30**	73.42

*Trained by us using our training strategy

Results on Pascal-VOC can be found in Table 3. We set $s = 17$, $lr = 0.005$, $\gamma^- = 2$, $\gamma^+ = 1$ to train our models on this dataset. Our modification of the GAN branch outperforms ML-Decoder when using EfficientNet-V2-s, while AAM loss gives a small performance boost and allows achieving SOTA with TResNet-L. Also, on Pascal-VOC EfficientNet-V2-s with all of considered additional graph branches or heads demonstrates a great speed/accuracy trade-off outperforming TResNet-L with ASL.

Table 3: Comparison with the state-of-the-art on Pascal-VOC dataset.

Method	Backbone	Input resolution	mAP	GFLOPs
ASL [1]	TResNet-L	448x448	94.60	43.50
Q2L [23]	TResNet-L	448x448	96.10	57.43
GATN [35]	ResNeXt-101	448x448	96.30	36.00
ML-Decoder [27]	TResNet-L	448x448	96.60	35.78
ASL*	EfficientNet-V2-s	448x448	94.24	10.83
Q2L*	EfficientNet-V2-s	448x448	94.94	15.40
ML-GCN*	EfficientNet-V2-s	448x448	95.25	10.83
ML-Decoder*	EfficientNet-V2-s	448x448	95.54	12.00
GAN re-weighting (ours)	EfficientNet-V2-s	448x448	**96.00**	**10.83**
GAN re-weighting (ours)	TResNet-L	448x448	96.67	35.20
ML-Decoder + AAM (ours)	EfficientNet-V2-s	448x448	95.86	12.00
ML-Decoder + AAM (ours)	EfficientNet-V2-L	448x448	96.05	49.92
ML-Decoder + AAM (ours)	TResNet-L	448x448	**96.70**	35.78

*Trained by us using our training strategy

Tables 4 and 5 show results on NUS and VG500 datasets. For NUS dataset we set $s = 23$, $lr = 0.009$, $\gamma^- = 2$, $\gamma^+ = 1$. For VG500 the hyperparameters are $s = 25$, $lr = 0.005$, $\gamma^- = 1$, $\gamma^+ = 0$. ML-Decoder clearly outperforms GCN and GAN branches on NUS-WIDE dataset. Also on NUS EfficientNet-V2-s with the AAM loss and GAN or ML-decoder head performs significantly better than the original implementations of Q2L and ASL with TResNet-L.

On VG500 we don't provide results of applying GCN or GAN branches, because this dataset has unnamed labels, so we can not apply a text-to-vec model to generate representations of graph nodes. Applying AAM with ML-Decoder on VG500 together with increased resolution allows achieving SOTA performance on this dataset as well.

As a result of the experiments, we can conclude that the combination of ML-Decoder with AAM loss gives an optimal performance on all of the considered datasets, while use the of the GAN-based branch could lead to better inference speed at the price of a small accuracy drop.

4.5 Confidence threshold tuning

Conventionally, threshold a value equals to 0.5 is used for calculating OP, OR, OF1 and CP, CR, CF1. But to apply a classification model in the wild under

Table 4: Comparison with the state-of-the-art on NUS-WIDE.

Method	Backbone	Input resolution	mAP	GFLOPs
GATN [35]	ResNeXt-101	448x448	59.80	36.00
ASL [1]	TResNet-L	448x448	65.20	43.50
Q2L [23]	TResNet-L	448x448	66.30	60.40
ML-GCN*	EfficientNet-V2-s	448x448	66.30	10.83
ASL*	EfficientNet-V2-s	448x448	65.20	10.83
Q2L*	EfficientNet-V2-s	448x448	65.79	16.25
ML-Decoder*	EfficientNet-V2-s	448x448	67.07	12.28
GAN re-weighting (ours)	EfficientNet-V2-s	448x448	66.85	**10.83**
GAN re-weighting (ours)	TResNet-L	448x448	68.10	35.20
ML-Decoder + AAM (ours)	EfficientNet-V2-s	448x448	67.60	12.28
ML-Decoder + AAM (ours)	TResNet-L	448x448	**68.30**	36.16

*Trained by us using our training strategy

Table 5: Comparison with the state-of-the-art on VG500.

Method	Backbone	Input resolution	mAP	GFLOPs
C-Tran [21]	ResNet101	576x576	38.40	-
Q2L [23]	TResNet-L	512x512	42.50	119.37
ASL*	EfficientNet-V2-s	576x576	38.84	17.90
Q2L*	EfficientNet-V2-s	576x576	40.35	32.81
ML-Decoder*	EfficientNet-V2-s	576x576	41.20	20.16
ML-Decoder + AAM (ours)	EfficientNet-V2-s	576x576	42.00	20.16
ML-Decoder + AAM (ours)	TResNet-L	576x576	**43.10**	59.63

*Trained by us using our training strategy

the variety of input data and presented classes, one need to estimate per-class confidence thresholds. When we don't have an access to the target real-world data, we can at least take into account per-class threshold variance, which arises because of training data distribution and loss function modifications. To do that, we propose finding per class thresholds by maximizing F1 score on each class via grid search on the train subset. Then we can evaluate the obtained thresholds on the validation subset and check if the precision-recall balance was improved.

In Table 6 the results of thresholds tuning for EfficientNetV2-s model trained with AAM loss on several datasets are shown. Precision-recall balance was improved by the mentioned procedure under train-validation distribution shift conditions for all of the considered datasets excepting Visual Genome 500. There thresholds tuning on train slightly decreased both OF1 and CF1 scores. This fact indicates that VG500 has a big distribution gap between the train and validation subsets, which the model can't handle (mAP $< 50\%$). Thus, if the trained model has low accuracy, estimating confidence threshold on the training subset may not work as expected.

Table 6: Results of confidence thresholds calibration for EfficientNetV2-s+MLD model trained with AAM.

Dataset	OF1	OF1-adaptive	CF1	CF1-adaptive
Pascal-VOC	91.14	91.94	92.60	93.21
COCO	82.86	84.39	85.03	86.06
NUS-WIDE	71.88	75.52	72.90	75.04
VG500	56.04	54.09	55.15	53.07

4.6 Ablation study

Contribution of algorithm's components To demonstrate the impact of each component on the whole pipeline we add them to a baseline one by one. As a baseline we take EfficientNetV2-s backbone and ASL loss with SGD optimizer. We set all the hyperparameters of the ASL loss and learning rate as in [1]. We use the training strategy described in Section 3.5 for all the experiments.

In Table 7 we can see that each component brings an improvement, except adding the GAN branch. ML-Decoder has enough capacity to learn labels correlation information, so further hints, that provides the GAN branch, don't improve the result. Also, we can see that tuning of the γ parameters is beneficial for the AAM loss, but the metric learning approach itself brings an improvement even without it.

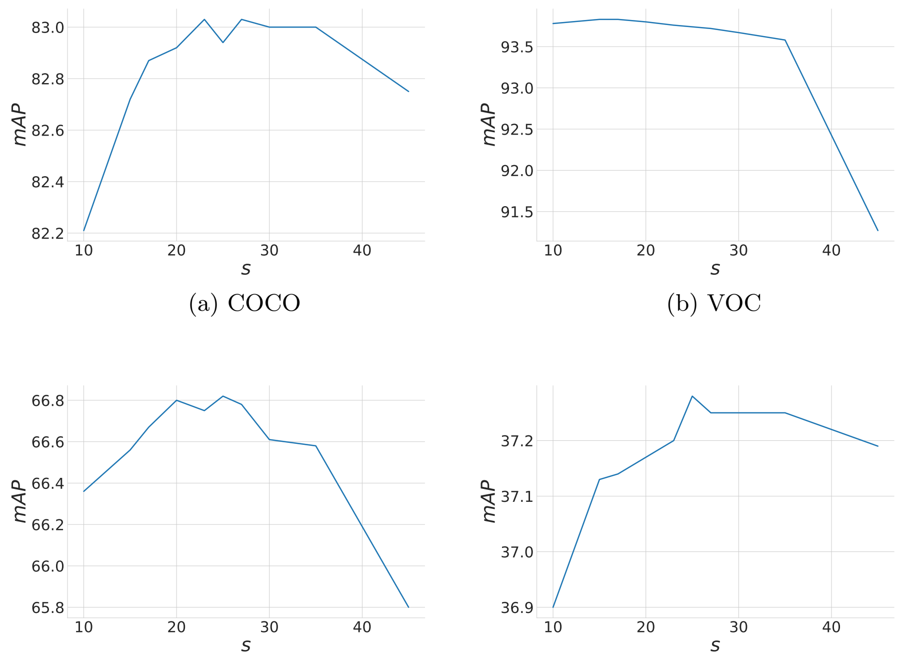
Table 7: Algorithm's components contribution.

Step	VOC-2007	COCO	NUS-WIDE	VG500
baseline	93.58	85.90	63.85	37.85
+ SAM	94.00	86.75	65.20	38.50
+ OI pretraining	94.24	87.05	65.54	38.84
+ MLD	95.54	88.30	67.07	41.20
+ AM loss*	95.80	88.60	67.30	41.90
+ AAM loss	95.86	88.75	67.60	42.00
+ GAN branch	95.85	88.70	67.20	-

* Angular Margin loss with $\gamma+ = \gamma- = 0$

Hyperparameters impact Figure 3 shows the scale parameter s impact on the training pipeline. We fix margin parameter at 0.0 value. We use our full training pipeline with AAM loss and ML-Decoder on 224x224 resolution for faster training. We can conclude that the value of s is correlated with the number of classes. For a small number of classes, the optimal value lies in the range 10-20. Otherwise, the 20-30 range will be a good choice.

Figure 4 shows margin influence from AAM loss. We see that this parameter brings unnecessary complexity to the choice of extra hyperparameter. We can exclude this parameter and simplify applying of the loss function.



(a) COCO (b) VOC

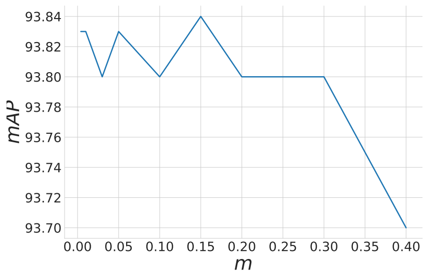
(c) NUS-WIDE (d) VG500

Fig. 3: The performance curves of EfficientNetV2-s+MLD+AAM model with different values of the s parameter. Margin parameter in AAM loss is set to zero. Training resolution is 224x224.

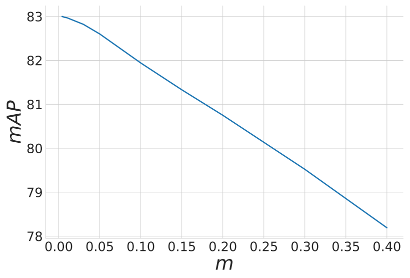
Table 8 shows the impact of the different asymmetry parameters values γ^+ and γ^-. As the authors of ASL [1] state in their work, a larger γ^- is required to handle a larger positive-negative imbalance. They also set γ^+ to 0 for all experiments, but we observe that there are datasets where non-zero γ^+ in AAM loss could bring an improvement as well.

We also notice that in the case of AAM loss, we need just 0-2 values ranges, as we have additional global loss scale parameter $\frac{1-k}{s}$ in (1).

Graph attention branch and transformer head comparison We want to compare our proposed new re-weighting scheme and the GCN method. If we refer to ML-GCN work [3], we see that the training strategy the authors chose for their method is rather simple. We argued that if we add a modern bag of training tricks to the method, the accuracy potentially could increase. We train backbone with ML-GCN head as described in [3], but with our training strategy as described in Section 3.5. Table 9 shows that GAN with a re-weighting scheme is more sophisticated in incorporating inter-label correlation. At the same time, the global attention ability of the transformer is sufficient for seeking internal



(a) VOC



(b) COCO

Fig. 4: The performance curves of EfficientNetV2-s+MLD+AAM model with different values of the m parameter. We fixed other hyperparameters for each dataset for all experiments. Training resolution is 224x224.

Table 8: Asymmetry influence on the training pipeline.

Gamma	VOC-2007	COCO	NUS-WIDE	VG500
$\gamma^- = 0, \gamma^+ = 0$	95.80	88.60	67.20	41.90
$\gamma^- = 1, \gamma^+ = 0$	95.80	**88.75**	67.30	**42.00**
$\gamma^- = 2, \gamma^+ = 0$	95.77	88.68	67.47	41.90
$\gamma^- = 3, \gamma^+ = 0$	95.78	88.62	67.27	41.80
$\gamma^- = 2, \gamma^+ = 1$	**95.86**	88.64	**67.60**	41.95
$\gamma^- = 3, \gamma^+ = 1$	95.80	88.52	**67.60**	41.60

dependencies. Even if we combine two methods and add to the model apriori information about the conditional probability of appearing labels, we will not obtain better quality.

Table 9: Comparison of GAN vs GCN method and ML Decoder.

Step	VOC-2007	COCO	NUS-WIDE
ML-GCN EfficientNetV2-s	95.25	87.50	66.30
GAN re-weighting EfficientNetV2-s	**96.00**	87.70	66.85
EfficientNetV2-s + MLD	95.86	**88.75**	**67.60**
GAN re-weighting EfficientNetV2-s + MLD	95.85	88.70	67.20

The next point is to check the ability of the transformer and GAN model to handle case of the high level of label noise. A high level of label noise can be achieved by simply downsizing the image resolution. Most of the small objects (especially in COCO and NUS-WIDE datasets) will disappear because of resize artifacts.

Table 10 shows the obtained results on all data with resolution 224x224. We also present the results of the model without both ML Decoder and GAN. We can conclude, that while the GAN branch is sufficient to help the ordinary model to better obtain global information on label distribution and co-occurrence. The transformer-based head can derive this information implicitly from the CNN features and in a more efficient way. Again, adding GAN as an additional branch doesn't improve the results.

Table 10: Comparison of graph-based method and transformer based in low resolution setup.

Step	VOC-2007	COCO	NUS-WIDE
EfficientNetV2-s	92.85	82.25	65.60
GAN re-weighting EfficientNetV2-s	93.15	82.55	65.87
EfficientNetV2-s + MLD	**93.77**	**83.03**	**66.74**
GAN re-weighting EfficientNetV2-s + MLD	93.71	**83.03**	66.60

5 Conclusion

In this work, we revisited two popular approaches to multilabel classification: transformer-based heads and labels graph branches. We refined the performance of these approaches by applying our training strategy with the modern bug of tricks and introducing a novel loss for multilabel classification called AAM. The loss combines properties of the ASL loss and metric learning approach and allows achieving competitive results on popular multilabel benchmarks.

References

1. Baruch, E.B., Ridnik, T., Zamir, N., Noy, A., Friedman, I., Protter, M., Zelnik-Manor, L.: Asymmetric loss for multi-label classification. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 82–91 (2021)
2. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709 (2020)
3. Chen, Z.M., Wei, X.S., Wang, P., Guo, Y.: Multi-label image recognition with graph convolutional networks. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 5172–5181 (2019)
4. Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: Nus-wide: a real-world web image database from national university of singapore. In: CIVR '09 (2009)
5. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.: Randaugment: Practical automated data augmentation with a reduced search space. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 18613–18624. Curran Associates, Inc. (2020)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)

7. Deng, J., Guo, J., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. ArXiv (2018)
8. Devries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. ArXiv **abs/1708.04552** (2017)
9. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision **88**, 303–338 (2009)
10. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. ArXiv **abs/2010.01412** (2020)
11. Gao, B.B., Zhou, H.Y.: Learning to discover multi-class attentional regions for multi-label image recognition. IEEE Transactions on Image Processing **30**, 5920–5932 (2021)
12. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition pp. 580–587 (2014)
13. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. arXiv preprint arXiv:1911.05722 (2019)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770–778 (2016)
15. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 558–567
16. Howard, A.G., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H.: Searching for mobilenetv3. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 1314–1324 (2019)
17. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 18661–18673. Curran Associates, Inc. (2020)
18. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., Bernstein, M.S., Fei-Fei, L.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. International Journal of Computer Vision **123**, 32–73 (2016)
19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012)
20. Kuznetsova, A., Rom, H., Alldrin, N.G., Uijlings, J.R.R., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Duerig, T., Ferrari, V.: Dataset v 4 unified image classification , object detection , and visual relationship detection at scale (2018)
21. Lanchantin, J., Wang, T., Ordonez, V., Qi, Y.: General multi-label image classification with transformers. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 16473–16483 (2021)
22. Lin, T.Y., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)
23. Liu, S., Zhang, L., Yang, X., Su, H., Zhu, J.: Query2label: A simple transformer way to multi-label classification. ArXiv **abs/2107.10834** (2021)

24. Makarenkov, V., Shapira, B., Rokach, L.: Language models with pre-trained (glove) word embeddings. arXiv: Computation and Language (2016)
25. Prokofiev, K., Sovrasov, V.: Towards efficient and data agnostic image classification training pipeline for embedded systems. In: Image Analysis and Processing – ICIAP 2022: 21st International Conference, Lecce, Italy, May 23–27, 2022, Proceedings, Part I. p. 476–488. Springer-Verlag, Berlin, Heidelberg (2022)
26. Ridnik, T., Lawen, H., Noy, A., Friedman, I.: Tresnet: High performance gpu-dedicated architecture. 2021 IEEE Winter Conference on Applications of Computer Vision (WACV) pp. 1399–1408 (2021)
27. Ridnik, T., Sharir, G., Ben-Cohen, A., Ben-Baruch, E., Noy, A.: Ml-decoder: Scalable and versatile classification head (2021)
28. Smith, L.N.: A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. ArXiv **abs/1803.09820** (2018)
29. Sovrasov, V., Sidnev, D.: Building computationally efficient and well-generalizing person re-identification models with metric learning. 2020 25th International Conference on Pattern Recognition (ICPR) pp. 639–646 (2021)
30. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. ArXiv **abs/1905.11946** (2019)
31. Tan, M., Le, Q.V.: Efficientnetv2: Smaller models and faster training. ArXiv **abs/2104.00298** (2021)
32. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. International Conference on Learning Representations (2018)
33. Wang, Z., Chen, T., Li, G., Xu, R., Lin, L.: Multi-label image recognition by recurrently discovering attentional regions. 2017 IEEE International Conference on Computer Vision (ICCV) pp. 464–472 (2017)
34. Wen, Y., Liu, W., Weller, A., Raj, B., Singh, R.: Sphereface2: Binary classification is all you need for deep face recognition. ArXiv **abs/2108.01513** (2021)
35. Yuan, J., Chen, S., Zhang, Y., Shi, Z., Geng, X., Fan, J., Rui, Y.: Graph attention transformer network for multi-label image classification. ArXiv **abs/2203.04049** (2022)
36. Zhu, K., Wu, J.: Residual attention: A simple but effective method for multi-label recognition (2021)