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

Kirill Prokofiev [a] and Vladislav Sovrasov [b]

Intel, Munich, Germany

{*kirill.prokofiev, vladislav.sovrasov*}*@intel.com*

Keywords: Multilabel Image Classification, Deep Learning, Lightweight Models, Graph Attention.

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 a 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 metric learning modification. In each binary classification sub-problem it operates with L_2 normalized feature vectors coming from a backbone and enforces angles between the normalized representations of positive and negative samples to be as large as possible. This results in 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[*].

1 INTRODUCTION

Starting from the impressive AlexNet (Krizhevsky et al., 2012) breakthrough on the ImageNet benchmark (Deng et al., 2009), the 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 (He et al., 2016; Howard et al., 2019; Tan and Le, 2019) and learning paradigms (Chen et al., 2020; Khosla et al., 2020; He et al., 2019a). 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 a lack of specialized multilabel datasets, researchers turned general object detection datasets such as MS-COCO (Lin et al., 2014) and PASCAL VOC (Everingham et al., 2009) into challenging multilabel classification benchmarks by removing bounding boxes from the

data annotation and leveraging only their class labels.

Despite the recent progress in solving the mentioned benchmarks, the latest works focus mainly on the resulting model accuracy, not taking into account computational complexity (Liu et al., 2021) or the use of outdated training techniques (Chen et al., 2019), introducing promising model architectures at the same time.

In this work, we are revisiting the latest approaches to multilabel classification, to propose a lightweight solution suitable for real-time applications and also improve the performance-accuracy tradeoff of the existing models.

The main contributions of this paper are as follows:

- We proposed a modification of ML-GCN (Chen et al., 2019) that adds graph attention operations (Veličković et al., 2018) 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 oc-

---

[a] https://orcid.org/0000-0001-9619-0248

[b] https://orcid.org/0000-0001-6525-2602

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

currence modeling via graph attention.

- We first applied the metric learning paradigm to multilabel a classification task and proposed a modified version of angular margin binary loss (Wen et al., 2021), which adds an ASL (Baruch et al., 2021) 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 (Krishna et al., 2016) and NUS-WIDE (Chua et al., 2009).

2 RELATED WORK

Historically, multilabel classification was attracting less attention than the multiclass scenario, but nonetheless there is still great progress in that field. Notable progress was achieved by developing advanced loss functions (Baruch et al., 2021), label co occurrence modeling (Chen et al., 2019; Yuan et al., 2022), designing advanced classification heads (Liu et al., 2021; Ridnik et al., 2021b; Zhu and Wu, 2021) and discovering architectures taking into account spatial distribution of objects via exploring attentional regions (Wang et al., 2017; Gao and Zhou, 2021).

Conventionally, a multilabel classification task is transformed into a set of binary classification tasks, which are solved by optimizing a binary cross-entropy loss function. Each single-class classification subtask suffers from a hard positives-negatives imbalance. The more classes the training dataset contains, the more negatives we have 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 (Baruch et al., 2021), 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 (Zhu and Wu, 2021) propose a drop-in replacement of the global average pooling layer that generates class-specific features for every category. Leveraging compact transformer heads for generating such features (Liu et al.,

2021; Ridnik et al., 2021b) turned out to be even more effective. This approach assumes pooling class-specific features by employing learnable embedding queries.

Considering the distribution of objects locations, or statistical label relationships requires data preprocessing and additional assumptions (Chen et al., 2019; Yuan et al., 2022) or sophisticated model architecture (Wang et al., 2017; Gao and Zhou, 2021). For instance, (Chen et al., 2019; Yuan et al., 2022) 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 (Girshick et al., 2014) module inside the model (Wang et al., 2017; Gao and Zhou, 2021), which drastically increases the 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 the principles described in (Prokofiev and Sovrasov, 2022), we use lightweight model architectures, hyperparameters optimization and early stopping.

3.1 Model Architecture

We chose EfficientNetV2 (Tan and Le, 2021) and TResNet (Ridnik et al., 2021a) 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 two different features aggregation approaches and compared their effectiveness and performance.

3.2 Transformer Multilabel Classification Head

As a representer of transformer-based feature aggregation methods, we use the ML-Decoder (Ridnik et al., 2021b) 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
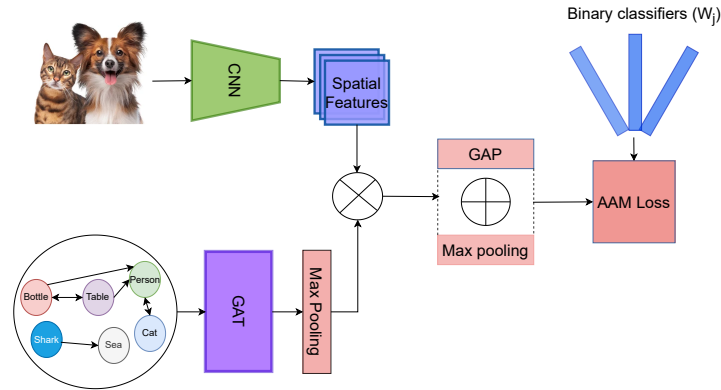
Figure 1: Overall diagram of the proposed GAT-based feature reweighting approach. Features obtained from the label relations graph by applying GAT are used to reweight channels in the CNN spatial features. Later, a pooling operation is applied to the reweighted features to obtain a vector representation. Finally, the resulting vectors are fed into binary classifiers W_j and optimized with the AAM loss, which is introduced in this work.

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 (Ridnik et al., 2021b). In our experiments we set $L = \min(100, K)$. Also, we L_2 normalize the arguments of all dot products in projections in case if we need to attach a metric learning loss to the ML-Decoder head.

3.3 Graph Attention Multilabel Branch

The original structure of the graph processing branch from (Chen et al., 2019) 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 (Zhu and Wu, 2021). To alleviate this limitation, we propose the architecture shown in Figure 1.

First, we generate the label correlation matrix $Z \in \mathbb{R}^{K \times K}$ in the same way as in (Chen et al., 2019). Together with the word embeddings $G \in \mathbb{R}^{K \times N}$, obtained with GLOVE (Makarenkov et al., 2016) model, where $N = 300$ is word embeddings dimension, we utilize Z as an input of the Graph Attention Network (Veličković et al., 2018). Unlike (Yuan et al., 2022), we use estimations of conditional probabilities to build Z, rather than fully relying on GLOVE and

calculating 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 reweighting of the CNN spatial features: $\tilde{f} = w \odot f$. Next, we apply global average pooling and max-pooling operations to $\tilde{f}$ in parallel, sum the results and obtain the 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 (GAT) branch for features re-weighting over the transformer head is a tiny computational and model complexity overhead at the inference stage. Since the GAT 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 GAT requires a vector representation of labels. Such representations can be generated by a text-to-vec model in case we have a meaningful description for all labels (even single word ones). This condition doesn't always hold: some datasets 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 (Baruch et al., 2021) has become a standard loss option for performing multi-label classification. By design it penalizes each logit with a modified binary cross-entropy 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 (Deng et al., 2018; Wen et al., 2021; Sovrasov and Sidnev, 2021).

We propose joining paradigms from (Baruch et al., 2021) and (Wen et al., 2021) to build even stronger loss for multilabel classification. Denote the result of the dot product between the normalized class embedding v_j produced by ML-Decoder ($v_j = \tilde{v}$, $j = \overline{1, K}$ if we use a backbone alone or with an optional GAT-based head) 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:

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

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

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

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 the 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 4.5.

3.5 Details of the Training Strategy

As in our former work (Prokofiev and Sovrasov, 2022), we aim to make the training pipeline for multilabel classification reliable, fast and adaptive to dataset, so we use the following components:

- SAM (Foret et al., 2020) optimizer with no bias decay (He et al., 2019b) is a default optimizer;
- EMA weights averaging for an additional protection from overfitting;
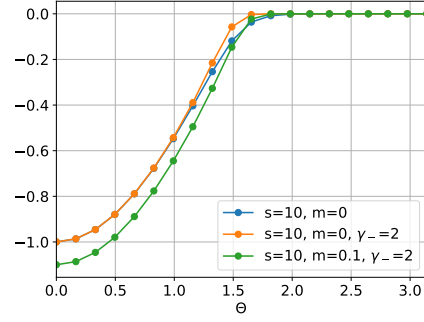
(a) Loss for p_-
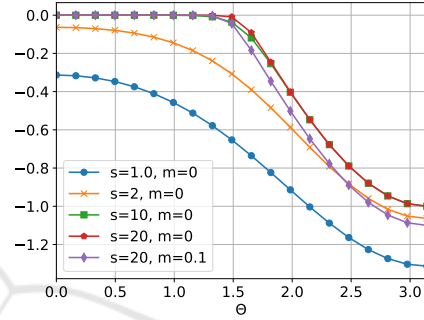
(b) Loss for p_+

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

- Initial learning rate estimation process from (Prokofiev and Sovrasov, 2022);
- OneCycle (Smith, 2018) 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 (Cubuk et al., 2020) strategy and Cutout (Devries and Taylor, 2017) 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 the 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: the average number of presented labels per image is significantly lower than the number of classes.

Also, we use a precisely annotated subset of OpenImages V6 (Kuznetsova et al., 2018) for pretraining purposes. We join the original training and testing parts into one training subset and use the original validation subset for testing.

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 confidence thresholding is required, threshold 0.5 is substituted. The exact formulas of the mentioned metrics can be found in (Liu et al., 2021).

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 on one object. Thus, additional task-specific pre-training on a large-scale dataset looks beneficial to standard ImageNet pretraining, and it was shown in recent works (Ridnik et al., 2021b). 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 the source code in (Ridnik et al., 2021b).

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 (Baruch et al., 2021) we set $lr = 0.0001$, $\gamma^- = 4$, $\gamma^+ = 0$ as suggested in the original paper (Baruch et al., 2021). 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 hyperparameters 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 $s = 23$, $lr = 0.007$, $\gamma^- = 1$, $\gamma^+ = 0$. With our AAM loss, we can achieve a state-of-the-art result using TResNet-L as a backbone. At the same time, the combination of EfficientNetV2-s with ML-Decoder and AAM loss outperforms TResNet-L with ASL, while consuming 3.5x less FLOPS. GCN/GAT branches perform slightly worse than ML-Decoder, but still improves results over EfficientNetV2-s + ASL with a marginal computational cost at inference.

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 GAT 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 the considered additional graph branches or heads demonstrates a great speed/accuracy trade-off outperforming TResNet-L with ASL.

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 GAT branches on NUS-WIDE dataset. Also on NUS EfficientNet-V2-s with the AAM loss and GAT 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 GAT 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 using the GAT-based branch could

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 (Everingham et al., 2009)	20	5011	4952	1.58
MS-COCO 2014 (Lin et al., 2014)	80	117266	4952	2.92
NUS-WIDE (Chua et al., 2009)	81	119103	50720	2.43
Visual Genome 500 (Krishna et al., 2016)	500	82904	10000	13.61
OpenImages V6 (Kuznetsova et al., 2018)	601	1866950	41151	5.23

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

Method	Backbone	Input resolution	mAP	GFLOPs
ASL (Baruch et al., 2021)	TResNet-L	448x448	88.40	43.50
Q2L (Liu et al., 2021)	TResNet-L	448x448	89.20	60.40
GATN (Yuan et al., 2022)	ResNeXt-101	448x448	89.30	36.00
Q2L (Liu et al., 2021)	TResNet-L	640x640	90.30	119.69
ML-Decoder (Ridnik et al., 2021b)	TResNet-L	448x448	90.00	36.15
ML-Decoder (Ridnik et al., 2021b)	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
GAT re-weighting (ours)	EfficientNet-V2-s	448x448	87.70	**10.83**
GAT 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

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

Method	Backbone	Input resolution	mAP	GFLOPs
ASL (Baruch et al., 2021)	TResNet-L	448x448	94.60	43.50
Q2L (Liu et al., 2021)	TResNet-L	448x448	96.10	57.43
GATN (Yuan et al., 2022)	ResNeXt-101	448x448	96.30	36.00
ML-Decoder (Ridnik et al., 2021b)	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
GAT re-weighting (ours)	EfficientNet-V2-s	448x448	**96.00**	**10.83**
GAT 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

lead to better inference speed at the price of a small accuracy drop. In particular, EfficientNetV2-s model with the GAT-based head or ML-Decoder on top outperforms T-ResNet-L with ASL, while consuming at least 3.5x less FLOPS. These results enable real-time applications of high-accuracy multilabel classification models on edge devices with a small computa-tional budget. At the same time, applying AAM loss, ML-Decoder and carefully designed training strategy allows achieving SOTA-level accuracy with TResNet-L.

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

Method	Backbone	Input resolution	mAP	GFLOPs
GATN (Yuan et al., 2022)	ResNeXt-101	448x448	59.80	36.00
ASL (Baruch et al., 2021)	TResNet-L	448x448	65.20	43.50
Q2L (Liu et al., 2021)	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
GAT re-weighting (ours)	EfficientNet-V2-s	448x448	66.85	**10.83**
GAT 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 (Lanchantin et al., 2021)	ResNet101	576x576	38.40	-
Q2L (Liu et al., 2021)	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

4.5 Ablation Study

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 hyper parameters of the ASL loss and learning rate as in (Baruch et al., 2021). We use the training strategy described in Section 3.5 for all the experiments.

In Table 6 we can see that each component brings an improvement, except adding the GAT branch. ML Decoder has enough capacity to learn labels correlation information, so further cues, that provide the GAT 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. Finally, adding the GAT branch to ML-Decoder doesn't increase the accuracy, indicating that additional information, coming from GAT, gives no new cues to ML-Decoder.

5 CONCLUSION

In this work, we revisited two popular approaches to multilabel classification: transformer-based heads

Table 6: Algorithm's components contribution.

Method	P-VOC	COCO	N-WIDE	VG500
baseline	93.58	85.90	63.85	37.85
+ SAM	94.00	86.75	65.20	38.50
+ OI wghts	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	95.86	88.75	67.60	42.00
+ GAT	95.85	88.70	67.20	-

* AAM with $\gamma+ = \gamma- = 0$

and labels graph branches. We refined the performance of these approaches by applying our training strategy with the modern bag 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. Although we demonstrated that graph branches perform very close to transformer-based heads, there is one major drawback of the graph-based method: it relies on labels' representations provided by a language model. The direction of future work could be developing an approach which would build a label relations graph relying on representations extracted directly from images, not involving potentially meaningless label names.

REFERENCES

Baruch, E. B., Ridnik, T., Zamir, N., Noy, A., Friedman, I., Protter, M., and Zelnik-Manor, L. (2021). Asymmetric loss for multi-label classification. *ICCV*.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*.

Chen, Z.-M., Wei, X.-S., Wang, P., and Guo, Y. (2019). Multi-label image recognition with graph convolutional networks. *CVPR*.

Chua, T.-S., Tang, J., Hong, R., Li, H., Luo, Z., and Zheng, Y. (2009). Nus-wide: a real-world web image database from national university of singapore. In *CIVR*.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Advances in Neural Information Processing Systems*, volume 33, pages 18613–18624.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Deng, J., Guo, J., and Zafeiriou, S. (2018). Arcface: Additive angular margin loss for deep face recognition. *ArXiv*.

Devries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *ArXiv*, abs/1708.04552.

Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J. M., and Zisserman, A. (2009). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. (2020). Sharpness-aware minimization for efficiently improving generalization. *ArXiv*, abs/2010.01412.

Gao, B.-B. and Zhou, H.-Y. (2021). Learning to discover multi-class attentional regions for multi-label image recognition. *IEEE Transactions on Image Processing*, 30:5920–5932.

Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2019a). Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *CVPR*.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., and Li, M. (2019b). Bag of tricks for image classification with convolutional neural networks. In *CVPR*.

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., and Adam, H. (2019). Searching for mobilenetv3. *ICCV*.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673.

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., and Fei-Fei, L. (2016). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

Kuznetsova, A., Rom, H., Alldrin, N. G., Uijlings, J. R. R., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Duerig, T., and Ferrari, V. (2018). Dataset v 4 unified image classification , object detection , and visual relationship detection at scale.

Lanchantin, J., Wang, T., Ordonez, V., and Qi, Y. (2021). General multi-label image classification with transformers. *CVPR*.

Lin, T.-Y., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*.

Liu, S., Zhang, L., Yang, X., Su, H., and Zhu, J. (2021). Query2label: A simple transformer way to multi-label classification. *ArXiv*, abs/2107.10834.

Makarenkov, V., Shapira, B., and Rokach, L. (2016). Language models with pre-trained (glove) word embeddings. *arXiv: Computation and Language*.

Prokofiev, K. and Sovrasov, V. (2022). Towards efficient and data agnostic image classification training pipeline for embedded systems. In *ICIAP*.

Ridnik, T., Lawen, H., Noy, A., and Friedman, I. (2021a). Tresnet: High performance gpu-dedicated architecture. *WACV*.

Ridnik, T., Sharir, G., Ben-Cohen, A., Ben-Baruch, E., and Noy, A. (2021b). Ml-decoder: Scalable and versatile classification head.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *ArXiv*, abs/1803.09820.

Sovrasov, V. and Sidnev, D. (2021). Building computationally efficient and well-generalizing person re-identification models with metric learning. *ICPR*.

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *ArXiv*, abs/1905.11946.

Tan, M. and Le, Q. V. (2021). Efficientnetv2: Smaller models and faster training. *ArXiv*, abs/2104.00298.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph Attention Networks. *ICLR*.

Wang, Z., Chen, T., Li, G., Xu, R., and Lin, L. (2017). Multi-label image recognition by recurrently discovering attentional regions. *ICCV*.

Wen, Y., Liu, W., Weller, A., Raj, B., and Singh, R. (2021). Sphereface2: Binary classification is all you need for deep face recognition. *ArXiv*, abs/2108.01513.

Yuan, J., Chen, S., Zhang, Y., Shi, Z., Geng, X., Fan, J., and Rui, Y. (2022). Graph attention transformer

network for multi-label image classification. *ArXiv*, abs/2203.04049.

Zhu, K. and Wu, J. (2021). Residual attention: A simple but effective method for multi-label recognition.