

ML-Decoder: Scalable and Versatile Classification Head

Tal Ridnik* Gilad Sharir*
Avi Ben-Cohen Emanuel Ben-Baruch Asaf Noy

DAMO Academy, Alibaba Group

Abstract

In this paper, we introduce *ML-Decoder*, a new attention-based classification head. *ML-Decoder* predicts the existence of class labels via queries, and enables better utilization of spatial data compared to global average pooling. By redesigning the decoder architecture, and using a novel group-decoding scheme, *ML-Decoder* is highly efficient, and can scale well to thousands of classes. Compared to using a larger backbone, *ML-Decoder* consistently provides a better speed-accuracy trade-off. *ML-Decoder* is also versatile - it can be used as a drop-in replacement for various classification heads, and generalize to unseen classes when operated with word queries. Novel query augmentations further improve its generalization ability. Using *ML-Decoder*, we achieve state-of-the-art results on several classification tasks: on MS-COCO multi-label, we reach 91.4% mAP; on NUS-WIDE zero-shot, we reach 31.1% ZSL mAP; and on ImageNet single-label, we reach with vanilla ResNet50 backbone a new top score of 80.7%, without extra data or distillation. Public code is available at: https://github.com/Alibaba-MIIL/ML_Decoder

1. Introduction

Image classification is a vital computer-vision task, that requires assigning a label or multiple labels to an image, according to the objects present in it. With *single-label classification* [40, 46], we assume that the image contains only one object, hence we can apply a softmax operation on the output logits. However, natural images usually contain multiple objects and concepts, highlighting the importance of *multi-label classification* [36, 44], where we predict each class separately and independently, in a similar fashion to multi-task problems [5, 34]. Notable success in the field of multi-label classification was reported by exploiting label correlation via graph neural networks [7, 8], and improving loss functions, pretrain methods and backbones [1, 2, 32, 33].

*Equal contribution

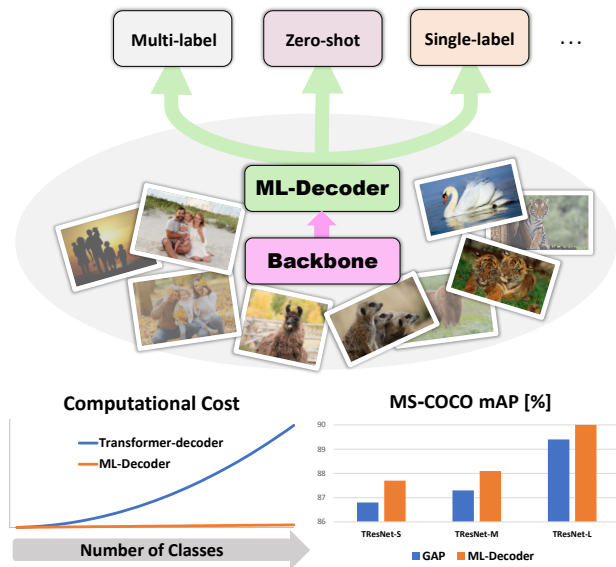


Figure 1. **Our proposed classification head.** ML-Decoder is versatile, and provides a unified solution for several classification tasks, with state-of-the-art results. Unlike transformer-decoder, it is also scalable, and can handle efficiently thousands of classes.

In a regime of *extreme classification* [27, 47], we need to predict the existence of a large number of classes (usually thousands or more), forcing our model and training scheme to be efficient and scalable. Multi-label *zero-shot learning* (ZSL) [38, 43] is an extension of multi-label classification, where during inference the network tries to recognize unseen labels, i.e., labels from additional categories that were not used during training. This is usually done by sharing knowledge between the seen classes (that were used for training) and the unseen classes via a text model [15, 30].

Classification networks usually contain a backbone, and a classification head [16, 33, 35]. The backbone outputs a spatial embedding tensor, and the classification head transforms the spatial embeddings into prediction logits. In single-label classification, this is commonly done by global-average-pooling (GAP), followed by a fully connected

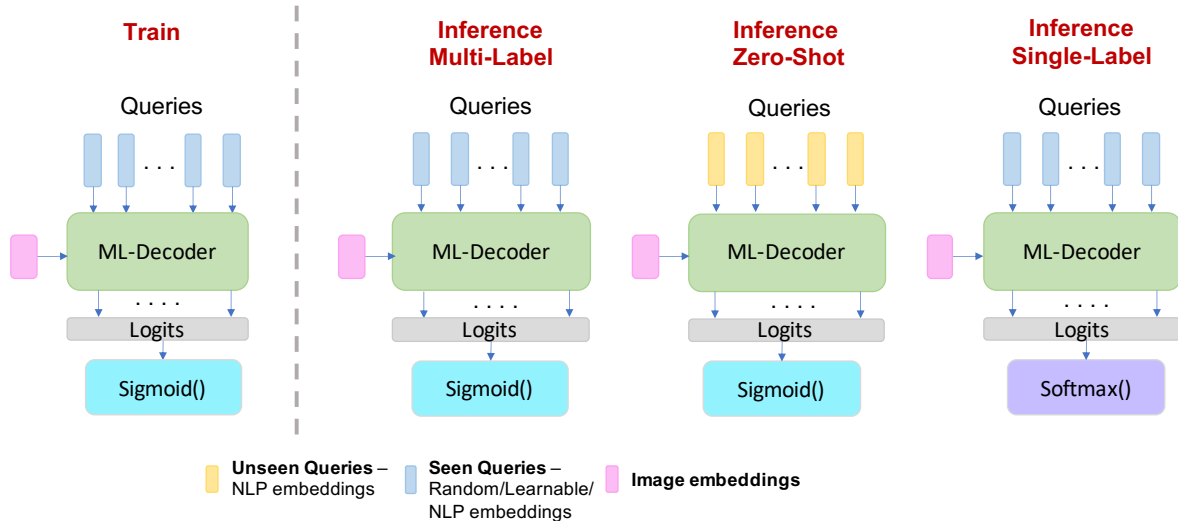


Figure 2. **Versatility** - ML-Decoder module is applicable to various classification tasks, such as multi-label, zero-shot, and single-label.

layer [14]. GAP-based heads are also used also for multi-label classification [8, 11, 42]. However, the need to identify several objects, with different locations and sizes, can make the usage of average pooling sub-optimal. Recently, several works proposed attention-based heads for multi-label classification. [13] offered a two-stream attention framework to recognize multi-category objects from global image to local regions. [49] suggested simple spatial attention scores, and then combined them with class-agnostic average pooling features. [25] presented a pooling transformer with learnable queries for multi-label classification, achieving top results.

GAP-based classification heads are simple and efficient, and scale well with the number of classes, since they have a fixed spatial pooling cost. However, they provide sub-optimal results, and are not directly applicable to ZSL. Attention-based classification heads do improve results, but are often costly, even for datasets with a small number of classes, and practically infeasible to extreme classification scenarios. They also have no natural extension to ZSL.

In this paper, we introduce a new classification head, called *ML-Decoder*, that provides a unified solution for single-label, multi-label, and zero-shot classification, with state-of-the-art results (see Figure 1). ML-Decoder design is based on the original transformer-decoder [37], with two major modifications, that significantly improve its scalability and efficiency. First, it reduces the quadratic dependence of the decoder in the number of input queries to a linear one, by removing the redundant self-attention block. Second, ML-Decoder uses a novel *group-decoding* scheme, where instead of assigning a query per class, it uses a fixed number of queries, that are interpolated to the final number of classes via a new architectural block called *group fully-connected*. Using group-decoding, ML-Decoder also

enjoys a fixed spatial pooling cost, and scales well to thousands of classes.

ML-Decoder is flexible and efficient. It can be trained equally well with learnable or fixed queries, and can use different queries during training and inference (see Figure 2). These key features make ML-Decoder suitable for ZSL tasks. When we assign a query per class and train ML-decoder with word queries, it generalizes well to unseen queries, and significantly improves previous state-of-the-art ZSL results. We also show that the group-decoding scheme can be extended to the ZSL scenario, and introduce novel query augmentations during training to further encourage generalization.

The paper’s contributions can be summarized as follows:

- We propose a new classification head called ML-Decoder, which provides a *unified solution* for multi-label, zero-shot, and single-label classification, with state-of-the-art results.
- ML-Decoder can be used as a drop-in replacement for global average pooling. It is *simple* and *efficient*, and provides improved speed-accuracy trade-off compared to larger backbones, or other attention-based heads.
- ML-Decoder novel design makes it *scalable* to classification with thousands of classes. Complementary query-augmentation technique improves its *generalizability* to unseen classes as well.
- We verify the effectiveness of ML-Decoder with comprehensive experiments on commonly-used classification datasets: MS-COCO, Open Images, NUS-WIDE, PASCAL-VOC, and ImageNet.

2. Method

In this section, we will first review the baseline classification heads. Then we will present our novel ML-Decoder, discuss its advantages, and show its applicability to several computer-vision tasks, such as multi-label, ZSL, and single-label classification.

2.1. Baseline Classification Heads

A typical classification network is comprised of a backbone, and a classification head. The network’s backbone outputs a spatial embedding tensor, $E \in \mathbb{R}^{H \times W \times D}$, and the classification head transforms the spatial embeddings tensor into N logits, $\{l_n\}_{n=1}^N$, where N is the number of classes. There are two baseline approaches for processing the spatial embeddings: GAP-based, and attention-based.

GAP-based: with a GAP-based classification head, we first reduce the spatial embeddings to a one-dimensional vector via simple global averaging operation on the spatial dimensions, outputting a vector $\mathbf{z} \in \mathbb{R}^{D \times 1}$. Then, a fully connected layer transforms the embedding vector into N output logits: $\mathbf{l} = W\mathbf{z}$, where $W \in \mathbb{R}^{N \times D}$ is a learnable linear projections matrix. GAP is commonly used for single-label classification tasks [16, 33, 35], and has some generalizations, for example [22, 31]. GAP was also adopted as a baseline approach for mutli-label classification [2, 26, 39]

Attention-based: Unlike single-label classification, in multi-label classification several objects can appear in the image, in different locations and sizes. Several works [13, 25, 49] have noticed that the GAP operation, which eliminates the spatial dimension via simple averaging, can be sub-optimal for identifying multiple objects with different sizes. Instead they suggested using attention-based classification heads, which enable more elaborate usage of the spatial data, with improved results.

2.2. Recap - Attention and Transformer-Decoder

Among the attention-based classification heads proposed, a simple approach based on a transformer-decoder, similar to the one used by DETR for object detection [4], has achieved top results on multi-label classification [25].

A transformer-decoder unit relies on the multi-head attention module, introduced in [37]. A multi-head attention has three inputs: Q, K, V . If we define the attention operation to be:

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

A multi-head module output is:

$$\text{MultiHeadAttn}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

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

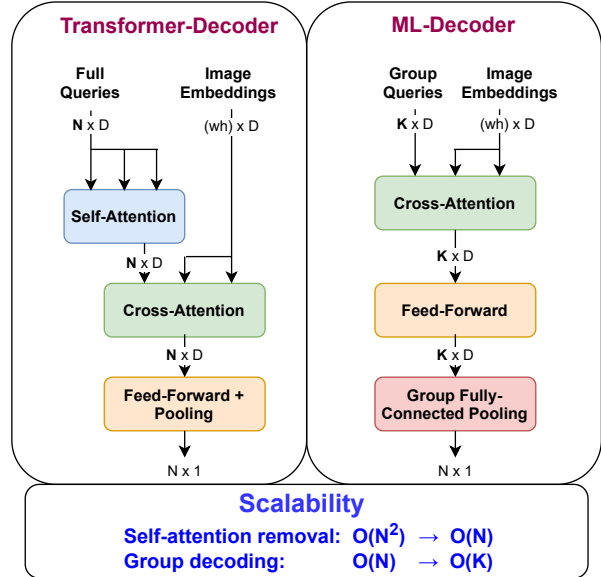


Figure 3. **Scalability** - baseline transformer-decoder vs. our proposed ML-Decoder. N - number of classes, K - number of group queries, D - tokens length. Removing the redundant self-attention block relaxes the quadratic dependence in the number of queries to a linear one, while retraining the same expressivity. When using group queries with fixed number of queries $K < N$, ML-Decoder becomes fully scalable, with spatial pooling cost independent in the number of classes.

W_i^Q, W_i^K, W_i^V, W^O are learnable projections matrices.

An illustration of a transformer-decoder classification head is given in Figure 3 (left side). The transformer-decoder has two inputs: The spatial embedding tensor, E , and a set of N learnable queries, Q , one for each class. The transformer-decoder processes the inputs via four consecutive stages called self-attention, cross-attention, feed-forward and token-pool:

$$\begin{aligned} \text{self-attn:} & \quad Q_1 \leftarrow \text{MultiHeadAttn}(Q, Q, Q) \\ \text{cross-attn:} & \quad Q_2 \leftarrow \text{MultiHeadAttn}(Q_1, E, E) \\ \text{feed-forward:} & \quad Q_3 \leftarrow \text{FF}(Q_2) \\ \text{token-pool:} & \quad \text{Logits} \leftarrow \text{Pool}(Q_3) \end{aligned} \quad (2)$$

FF is a feed-forward fully connected layer, as defined in [37]. The token pooling stage is a simple pooling on the token embeddings’ dimension D , to produce N output logits.

2.3. ML-Decoder

2.3.1 Motivation

On multi-label datasets with small number of classes, such as MS-COCO [24] and Pascal-VOC [12] (80 and 20 classes respectively), transformer-decoder classification

head works well, and achieves state-of-the-art results [25], with small additional computational overhead. However, it suffers from a critical drawback - the computational cost is quadratic with the number of classes. Hence, for datasets with large number of classes, such as Open Images [21] (9600 classes), using transformer-decoder is practically infeasible in terms of computational cost, as we will show in Section 3.2. For real-world applications, a large number of classes is imperative to provide a complete and comprehensive description of an input image. Hence, a more scalable and efficient attention-based classification head is needed. In addition, transformer-decoder as a classification head is suitable for multi-label classification only. A more general attention-based head, that can also address other tasks such as single-label and ZSL, will be beneficial.

2.3.2 ML-Decoder Design

We will now describe our proposed classification head, ML-Decoder. Illustration of ML-Decoder flow is given in Figure 3 (right-side). Compared to transformer-decoder, ML-Decoder includes the following modifications:

(1) Self-attention removal: We start by observing that during inference, the self-attention module of transformer-decoder provides a *fixed* transformation on the input queries. However, as the queries are entering the cross-attention module, they are subjected to a projection layer, before going through the attention operation (Eq. 1). In practice, the projection layer can transform the queries to any desired output, making the self-attention module redundant. Hence, we can remove the self-attention layer, while still maintaining the same expressivity of the classification head, and without degrading the results. We will validate this empirically in Section 3.1. By removing the self-attention, we avoid a costly module, and relax the quadratic dependence of ML-Decoder in the number of input queries to a linear one, making it more practical and efficient.

(2) Group-decoding: In an extreme classification scenario, even linear dependency of the classification head with the number of classes can be costly. We want to break this coupling, and make the cross-attention module, and the feed-forward layer after it, independent of the number of classes, same as GAP operation. To this end, instead of assigning a query per class, we use as inputs a fixed number of group queries, K (see Figure 3).

After the feed-forward layer, we transform the group queries into output logits via a novel layer called *group fully-connected*. This layer performs simultaneously two tasks - (1) expand each group query to $\frac{N}{K}$ outputs; (2) pool the embeddings' dimension. If we define the *group-factor* to be $g = \frac{N}{K}$, group fully-connected generates an output

logit L_i with the following operation:

$$L_i = (W_k \cdot Q_k)_j$$

$$\text{where: } k = i \text{ div } g, \quad j = i \text{ mod } g \quad (3)$$

$Q_k \in \mathbb{R}^D$ is the k^{th} query, and $W_k \in \mathbb{R}^{g \times D}$ is the k^{th} learnable projection matrix. An illustration of group fully-connected layer is given in Figure 4, and a pseudo-code appears in appendix L.

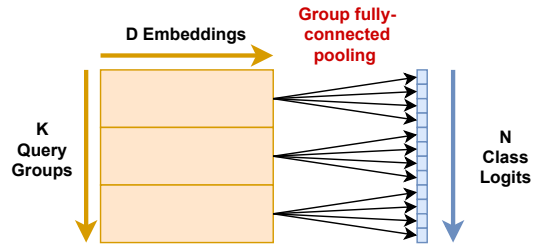


Figure 4. **Scheme of a group fully-connected layer** (with $g = 4$).

The full flow of ML-Decoder with group-decoding is depicted in Eq. 4, where G_q are the input group queries:

$$\begin{aligned} \text{cross-attn: } & G_{q_1} \leftarrow \text{MultiHeadAttn}(G_q, E, E) \\ \text{feed-forward: } & G_{q_2} \leftarrow \text{FF}(G_{q_1}) \\ \text{group FC: } & \text{Logits} \leftarrow \text{Group-FC}(G_{q_2}) \end{aligned} \quad (4)$$

Some additional observations and insights into the group-decoding scheme:

- With full-decoding ($g = 1$), each query checks the existence of a single class. With group-decoding, each query checks the existence of several classes. We chose to divide the classes into groups in a random manner. Clustering the classes via semantic proximity is an alternative, but will require a cumbersome clustering process, with extra hyper-parameters that might need tuning per dataset. In Section 3.2 we will show that random group clustering is enough to provide results comparable to a full-decoding scheme.
- In terms of flops, the group fully-connected layer is equivalent to a fully-connected layer in a GAP-based head ($N \times D$ multiplications). Both are linearly dependent in the number classes, but in practice they have a small computational overhead, even for thousands of classes. In terms of memory consumption, performing the two tasks of group fully-connected together, in a single operation, is more efficient than doing them consecutively, since there is no need to store large intermediate maps.
- The only component in ML-Decoder that depends on the input image size is the cross-attention module. We can think of the cross-attention layer as doing *spatial pooling*, similar to GAP. With group-decoding, ML-Decoder has fixed spatial pooling cost, independent of N .

(3) Non-learnable queries: [25] argued that transformer-decoder for multi-label classification achieves top results only with learnable queries. However, we observe that the queries are always fed into a multi-head attention layer, that applies a learnable projection on them (Eq. 1). Hence, setting the queries weights as learnable is redundant - a learnable projection can transform any fixed-value query to any value obtained by a learnable query. We will validate this empirically in Section 3.1, showing that the same accuracies are obtained when training ML-Decoder with learnable or fixed queries. In addition to simplifying the training process, using fixed queries will enable us to do ZSL.

2.3.3 ML-Decoder for ZSL

Next, we will present the adaptations needed in order to use ML-Decoder in a multi-label ZSL scenario, and discuss key features of ML-Decoder that make it suitable for the task. We will also show that the group-decoding scheme can be extended to ZSL, and present novel query augmentations that further improve ML-Decoder generalizability.

NLP-based queries: We begin by presenting a version of ML-Decoder for ZSL with a full-decoding scheme (each label has a corresponding query). As discussed in the previous section, the input queries can be either learnable or fixed. For ZSL, we use fixed NLP-based queries - for each label, a word embedding vector is extracted using a language model, and set as the input query. We also use shared projection matrix in the group fully-connected layer (setting $W_k = W$ in Eq. 3). With NLP-based queries and a shared projection matrix, semantic information can propagate from the *seen* (training) classes to the *unseen* (test) classes during inference, enabling generalization.

ML-Decoder features: ML-Decoder contains several favorable features which make it well suited for ZSL. Firstly, its attention mechanism is based on dot-product similarity between vectors (Eq. 1). Since NLP word embeddings preserve this dot-product semantic similarity [15], the unseen labels are more likely to be matched with the most similar keys and values within the decoder. In addition, ML-Decoder with a shared projection matrix allows a variable number of input queries, and is not sensitive to the order of queries. This is beneficial since in ZSL we perform training and testing on different sets of classes, and therefore different sets of queries. For ZSL we train exclusively on the seen labels, and perform inference on the unseen classes, while for Generalized ZSL (GZSL), we perform inference on the union of the unseen and seen sets of labels.

Group-decoding: Group-decoding (with $K < N$) requires modifications in order to work in a ZSL setting.

In appendix B we thoroughly detail our variant of group-decoding for ZSL.

Query augmentations With NLP-based queries, ML-Decoder naturally extends to the task of ZSL. Yet, we want to apply dedicated training tricks that further improve its generalizability. It is a common practice in computer-vision to apply augmentations on the input images to prevent overfitting, and improve generalizability to new unseen images. Similarly, we introduce *query augmentations* to encourage generalization to unseen class queries. The first augmentation is *random-query*, which adds additional random-valued queries to the set of input queries, and assigns a positive ground truth label signifying “random” for these added queries. The second augmentation is *query-noise*, where we add each batch a small random noise to the input queries. See Figure 8 in the appendix for illustration of the augmentations. In Section 3.3 we will show that query augmentations encourage the model to recognize novel query vectors which it hasn’t encountered before, and improve ZSL scores. We also tried query-cutout augmentation, where random parts of the queries are deleted each batch. However, this technique was not beneficial in our experiments.

2.3.4 ML-Decoder for Single-label Classification

Most attention-based classification heads previously proposed were purposed for multi-label classification ([13, 25, 49], for example), and this was also the primary task we focused on in our work. However, our design of ML-Decoder enables it to be used as a drop-in replacement for GAP-based heads on other computer-vision tasks, such as single-label classification, as shown in Figure 2.

The main motivation for attention-based heads in multi-label classification was the need to identify several objects, with different locations and sizes [25]. We will show in Section 4.3 that the benefit from ML-Decoder is more general, and fully applies also to single-label problems, where the image usually contains a single object.

3. Experimental Study

In this section, we will bring ablation tests and inner comparisons for our proposed ML-Decoder classification head. First, we will test ML-Decoder with different types of input queries. Then we will compare ML-Decoder to other classification heads, such as transformer-decoder and GAP-based. Finally, we will provide an ablation study for ZSL with augmentation queries and group-decoding.

3.1. Comparing Query Types

As discussed in Section 2.3.2, due to the linear projection in the attention module, ML-Decoder retains the same expressivity when it uses learnable or fixed queries. In Table

7 in the appendix we compare results for ML-Decoder with different types of queries, on MS-COCO multi-label dataset (see appendix C for full training details on MS-COCO).

Indeed we see that learnable, fixed random, and fixed NLP-based word queries all lead to the same accuracy, 88.1% mAP, as expected. For reducing the number of learned parameters, we shall use fixed queries.

3.2. Comparing Different Classification Heads

In Table 1 we compare MS-COCO results for training with different classification heads.

Classification Head	Num of Classes	Num of Queries	Flops [G]	mAP [%]
GAP	80	—	23.0	87.0
Transformer-Decoder	80	80	24.1	88.1
ML-Decoder	80	20	23.6	88.0
ML-Decoder	80	80	23.9	88.1

Table 1. Comparison of multi-label MS-COCO mAP score for different classification heads. Architecture - TResNet-M [33].

From Table 1 we learn the following observations:

- ML-Decoder (and transformer-decoder) provide a significant improvement (more than 1% mAP), compared to GAP-based classification head.
- When using the same number of input queries (80), transformer-decoder and ML-Decoder reach the same accuracy, demonstrating that indeed the self-attention module provides a redundant transformation (see Section 2.3.2), and removing it in ML-Decoder reduces the computational cost without impacting the results.
- Using group decoding with a ratio of $\frac{N}{K} = 4$ has a minimal impact on the results - a reduction of only 0.1% mAP. However, since MS-COCO dataset has a small number of classes, the additional flops from an attention-based classification head are minimal, so reducing the number of queries is not essential in this case.

In Table 2 we repeat the same comparison on Open Images multi-label dataset, which has significantly more classes - 9600 instead of 80. Full training details on Open Images are given in appendix H.

On Open Images, using transformer-decoder classification head is not feasible - due to the large number of classes, the additional computational cost is very high, and even with a batch size of 1 and input resolution of 224, our training is out-of-memory (see Table 9 for full specifications). In contrast, ML-Decoder with group-decoding increases the flops count by only 10% – 20%, while significantly improving the mAP score on this challenging extreme classification dataset, compared to GAP. We also see from Table 2 that group decoding with a ratio of $\frac{N}{K} = 48$ already

Classification Head	Num of Classes	Num of Queries	Flops [G]	mAP [%]
GAP	9600	—	5.8	86.0
Transformer Decoder	9600	9600	178.6	NA
ML-Decoder	9600	100	6.3	86.7
ML-Decoder	9600	200	6.7	86.8
ML-Decoder	9600	400	7.6	86.8

Table 2. Comparison of Open Images mAP score for different classification heads. Architecture - TResNet-M.

gives the full score benefit, and further increasing the ratio to $\frac{N}{K} = 96$ reduces the score by only 0.1%.

In Figure 5 we compare on MS-COCO the mAP score vs. flops for GAP-based and ML-Decoder classification head, with three different architectures - TResNet-S, TResNet-M, TResNet-L (equivalent in runtime to ResNet34, ResNet50 and ResNet101 [33]).

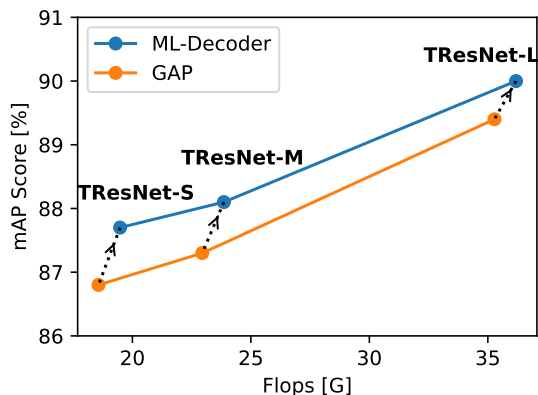


Figure 5. mAP score vs. Flops comparison, on MS-COCO dataset, for different classification heads. For ML-Decoder, we used $K = N = 80$.

We see from Figure 5 that using ML-decoder provides a better flops-accuracy trade-off compared to using GAP-based classification head with a larger backbone.

In addition to the flops-accuracy measurements, in Table 9 in the appendix we provide full speed-accuracy comparisons of different classification heads, measuring inference speed, training speed, maximal batch size and flops. This table can aid in future comparisons of our work for different speed-accuracy metrics.

3.3. Zero-shot Learning

This section presents an ablation study of ML-Decoder for ZSL, on NUS-WIDE dataset [9]. NUS-WIDE is the most widely used benchmark for the multi-label ZSL task, and therefore we focus on this dataset. It is comprised of

925 seen labels, and 81 unseen labels. Full training and dataset details are given in appendix F.

In Table 3 we compare the different types of query augmentations presented in Section 2.3.3.

Augmentation Type	mAP [%] (ZSL)
None	29.9
Additive noise	30.6
Random-query	30.7
Both	31.1

Table 3. Comparison of NUS-WIDE ZSL mAP scores for ML-Decoder with different query augmentations.

It is evident from the table that both random-query and additive noise contribute to the model’s ability to generalize to unseen classes. When applying both of them, we see an increase of 1.2% in the mAP score of the unseen classes. We also tested the impact of query augmentations on the seen classes. Both on MS-COCO and on NUS-WIDE, adding query augmentations had no impact on the seen classes’ mAP score (88.1% on MS-COCO, 22.7% on NUS-WIDE). This is in agreement with our results in Section 3.1, where learnable and fixed queries provided the same results on the seen classes. The benefit from using query augmentation is by better generalization to the unseen classes.

In Table 4 we compare group-decoder to full-decoder for the ZSL task. The modifications to the group-decoding scheme for ZSL are described in appendix B.

Classification Head	Num of Classes	Num of Queries	mAP [%] (ZSL)
ML-Decoder	1006	1006	31.1
ML-Decoder	1006	100	28.7

Table 4. Comparison of NUS-WIDE ZSL mAP scores for ML-Decoder with different number of input queries.

We see from the Table 4 that the group-decoding scheme works well also for the ZSL scenario, with a small decrease in the mAP scores compared to full-decoding.

4. Results

In this section, we will evaluate our ML-Decoder-based solution on popular multi-label, ZSL, and single-label classification datasets, and compare results to known state-of-the-art techniques.

4.1. Multi-label Classification

4.1.1 MS-COCO

MS-COCO [24] is a commonly-used dataset to evaluate multi-label image classification. It contains 122, 218 im-

ages from 80 different categories, divided to a training set of 82, 081 images and a validation set of 40, 137 images.

In Table 5 we compare ML-Decoder results to top known solutions from the literature. Full training details appear in appendix C.

Method	Backbone	Input Resolution	mAP [%]
ML-GCN [8]	ResNet101	448x448	83.0
KSSNET [26]	ResNet101	448x448	83.7
SSGRL [6]	ResNet101	576x576	83.8
MS-CMA [45]	ResNet101	448x448	83.8
ASL [2]	ResNet101	448x448	85.0
ASL [2]	TResNet-L	448x448	88.4
Q2L [25]	TResNet-L	448x448	89.2
ML-Decoder	ResNet101	448x448	87.1
ML-Decoder	TResNet-L	448x448	90.0

Table 5. State-of-the-art comparison on MS-COCO dataset. TResNet-L has equivalent runtime to ResNet101, with design tricks that lead to improved results [33].

Since previous works did not always report their computational cost, and released a reproducible code, we cannot provide a full speed-accuracy comparison to them. Still, we see that with ML-Decoder we improve results on MS-COCO dataset, with minimal additional computational cost compared to plain GAP-based solution (see also Table 1). We hope that future works will use our results as a baseline for a more complete comparison, including computational cost and accuracy. For that matter, in Table 10 in the appendix we report our results and flops count for different input resolutions. Note that with input resolution of 640, we reach on MS-COCO with TResNet-L and TResNet-XL new state-of-the-art results of 91.1% and 91.4%, respectively.

4.1.2 Additional Multi-label Datasets

Pascal-VOC: In Table 11 in the appendix we present results on another popular multi-label dataset - Pascal-VOC [12]. With ML-Decoder we reach new state-of-the-art result on Pascal-VOC - 96.6% mAP.

Open Images: In Table 12 in the appendix we present results on an extreme classification multi-label dataset - Open Images [21], which contains 9600 classes. Also on this dataset ML-Decoder outperforms previous methods, achieving 86.8% mAP. Notice that due the large number of classes, some attention-based methods are not feasible for this dataset. Hence, no results for them are available.

4.2. Zero-Shot Learning

In Table 6 we present a SotA comparison on NUS-WIDE multi-label zero-shot dataset [9]. Similar to previous works, we use F1 score at top-K predictions and mAP as evaluation

metrics. mAP is measured both on ZSL (unseen classes only) and GZSL (seen+unseen classes). Full training details appear in appendix F.

Method	Task	mAP	F1 (K = 3)	F1 (K = 5)
CONSE [29]	ZSL	9.4	21.6	20.2
	GZSL	2.1	7.0	8.1
Fast0Tag [48]	ZSL	15.1	27.8	26.4
	GZSL	3.7	11.5	13.5
Attention per Label [18]	ZSL	10.4	25.8	23.6
	GZSL	3.7	10.9	13.2
Attention per Cluster [17]	ZSL	12.9	24.6	22.9
	GZSL	2.6	6.4	7.7
LESA [17]	ZSL	19.4	31.6	28.7
	GZSL	5.6	14.4	16.8
BiAM [28]	ZSL	26.3	33.1	30.7
	GZSL	9.3	16.1	19.0
SDL [3]	ZSL	25.9	30.5	27.8
	GZSL	12.1	18.5	21.0
ML-Decoder	ZSL	31.1	34.1	30.8
	GZSL	19.9	23.3	26.1

Table 6. **State-of-the-art comparison for multi-label ZSL and GZSL tasks on NUS-WIDE dataset.**

We see from Table 6 that our approach significantly outperforms the previous top solution by 4.8% mAP (ZSL), setting a new SotA in this task.

Note that previous methods were mostly aimed at optimizing the task of zero-shot, at the expense of the seen classes (ZSL vs. GZSL trade-off). SDL [3], for example, proposed to use several principal embedding vectors, and trained them using a tailored loss function for ZSL. In contrast to previous methods, ML-Decoder offers a simple unified solution for plain and zero-shot classification, and achieves top results for both ZSL and GZSL. ML-Decoder sets a new SotA score also on the GZSL task, outperforming SDL with a significant improvement (from 12.1% to 19.9%). This clearly demonstrates that ML-Decoder generalizes well to unseen classes, while maintaining high accuracy on the seen classes.

4.3. Single-label Classification

To test ML-Decoder effectiveness for single-label classification, we used ImageNet dataset [10], with the high-quality training code suggested in [41] (A2 configuration). Comparison of various ResNet architectures, with different classification heads, appears in Figure 6. As can be seen, when replacing the baseline GAP-based head with ML-Decoder, we significantly improve the models’ accuracy. ML-Decoder also provides a better speed-accuracy trade-off compared to using GAP with a larger backbone.

Notice that we used ML-Decoder without introducing any change or tuning any hyper-parameter in the training configuration. The fact that ML-Decoder can serve as a

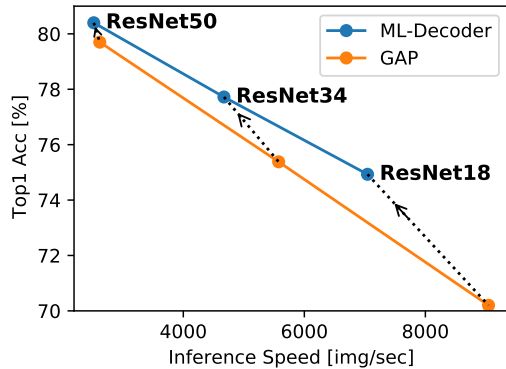


Figure 6. **Speed-accuracy comparison for different classification heads on ImageNet dataset.** For ML-Decoder, we used group-decoding (100 groups). Training configuration - A2 [41].

drop-in replacement for GAP-based classification head, in a highly-optimized single-label training setting (baseline ResNet50 achieves 79.7%), and still provides an additional boost, demonstrate its effectiveness and versatility. Also, note that following [41], our training configuration for ImageNet is using a multi-label loss (sigmoid with BCE loss instead of softmax and CE loss). That’s the default mode we present in Figure 2. In Table 13 in the appendix we validate that also with softmax, ML-Decoder provides a benefit compared to plain GAP.

When increasing the number of training epochs to 600 (A1 configuration in [41]), with ML-Decoder and vanilla ResNet50 backbone we reach 80.7% accuracy. To the best of our knowledge, this is the top results so far achieved with ResNet50 (without extra data or distillation).

4.3.1 Additional Results

In appendix J we bring additional results for single-label classification, showing that with ML-Decoder we achieve top results on two other prominent single-label datasets - CIFAR-100 [20] and Stanford-Cars [19].

5. Conclusions and Future Work

In this paper, we introduced ML-Decoder, a new attention-based classification head. By removing the redundant self-attention layer and using a novel group-decoding scheme, ML-Decoder scales well to thousands of classes, and provides a better speed-accuracy trade-off than using a larger backbone. ML-Decoder can work equally well with fixed or random queries, and use different queries during training and inference. With word-based queries and novel query augmentations, ML-Decoder also generalizes well to unseen classes. Extensive experimental analysis shows that ML-Decoder outperforms GAP-based heads on

several classification tasks, such as multi-label, single-label and zero-shot, and achieves new state-of-the-art results.

Our future work will focus on extending the usage of ML-Decoder to other computer-vision tasks that include classification, such as object detection and video recognition. We will also explore using the group-decoding scheme more generally, for processing any spatial embedding tensor, and will try to apply it to other fields and tasks, such as segmentation, pose estimation and NLP-related problems.

References

- [1] Emanuel Ben-Baruch, Tal Ridnik, Itamar Friedman, Avi Ben-Cohen, Nadav Zamir, Asaf Noy, and Lihi Zelnik-Manor. Multi-label classification with partial annotations using class-aware selective loss, 2021. **1**
- [2] Emanuel Ben-Baruch, Tal Ridnik, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. *arXiv preprint arXiv:2009.14119*, 2020. **1, 3, 7, 11, 12, 13**
- [3] Avi Ben-Cohen, Nadav Zamir, Emanuel Ben Baruch, Itamar Friedman, and Lihi Zelnik-Manor. Semantic diversity learning for zero-shot multi-label classification. *arXiv preprint arXiv:2105.05926*, 2021. **8, 12**
- [4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. **3**
- [5] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997. **1**
- [6] Tianshui Chen, Muxin Xu, Xiaolu Hui, Hefeng Wu, and Liang Lin. Learning semantic-specific graph representation for multi-label image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 522–531, 2019. **7, 12**
- [7] Zhao-Min Chen, Xiu-Shen Wei, Xin Jin, and Yanwen Guo. Multi-label image recognition with joint class-aware map disentangling and label correlation embedding. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 622–627. IEEE, 2019. **1**
- [8] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5177–5186, 2019. **1, 2, 7, 12**
- [9] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, pages 1–9, 2009. **6, 7**
- [10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. **8**
- [11] Thibaut Durand, Nazanin Mehrasa, and Greg Mori. Learning a deep convnet for multi-label classification with partial labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 647–657, 2019. **2**
- [12] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge 2007 (voc2007) results. *arxiv*, 2007. **3, 7**
- [13] Bin-Bin Gao and Hong-Yu Zhou. Learning to discover multi-class attentional regions for multi-label image recognition. *IEEE Transactions on Image Processing*, 30:5920–5932, 2021. **2, 3, 5**
- [14] Arna Ghosh, Biswarup Bhattacharya, and Somnath Basu Roy Chowdhury. Adgap: Advanced global average pooling. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. **2**
- [15] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.’s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014. **1, 5**
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. **1, 3**
- [17] Dat Huynh and Ehsan Elhamifar. A shared multi-attention framework for multi-label zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8776–8786, 2020. **8**
- [18] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear attention networks. *arXiv preprint arXiv:1805.07932*, 2018. **8**
- [19] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013. **8, 13**
- [20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. **8, 13**
- [21] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Mallocci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018. **4, 7, 13**
- [22] Chen-Yu Lee, Patrick Gallagher, and Zhuowen Tu. Generalizing pooling functions in cnns: Mixed, gated, and tree. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):863–875, 2017. **3**
- [23] Peng Li, Peng Chen, Yonghong Xie, and Dezheng Zhang. Bi-modal learning with channel-wise attention for multi-label image classification. *IEEE Access*, 8:9965–9977, 2020. **12**
- [24] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. **3, 7**
- [25] Shilong Liu, Lei Zhang, Xiao Yang, Hang Su, and Jun Zhu. Query2label: A simple transformer way to multi-label classification. *arXiv preprint arXiv:2107.10834*, 2021. **2, 3, 4, 5, 7, 12**
- [26] Yongcheng Liu, Lu Sheng, Jing Shao, Junjie Yan, Shiming Xiang, and Chunhong Pan. Multi-label image classification via knowledge distillation from weakly-supervised de-

- tection. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 700–708, 2018. 3, 7
- [27] Tharun Medini, Qixuan Huang, Yiqiu Wang, Vijai Mohan, and Anshumali Shrivastava. Extreme classification in log memory using count-min sketch: A case study of amazon search with 50m products. *arXiv preprint arXiv:1910.13830*, 2019. 1
- [28] Sanath Narayan, Akshita Gupta, Salman Khan, Fahad Shahbaz Khan, Ling Shao, and Mubarak Shah. Discriminative region-based multi-label zero-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8731–8740, 2021. 8
- [29] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*, 2013. 8
- [30] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 1
- [31] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018. 3
- [32] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses, 2021. 1
- [33] Tal Ridnik, Hussam Lawen, Asaf Noy, Emanuel Ben Baruch, Gilad Sharir, and Itamar Friedman. Tresnet: High performance gpu-dedicated architecture. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1400–1409, 2021. 1, 3, 6, 7
- [34] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017. 1
- [35] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 1, 3
- [36] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007. 1
- [37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2, 3
- [38] Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–37, 2019. 1
- [39] Zhouxia Wang, Tianshui Chen, Guanbin Li, Ruijia Xu, and Liang Lin. Multi-label image recognition by recurrently discovering attentional regions. In *Proceedings of the IEEE international conference on computer vision*, pages 464–472, 2017. 3, 12
- [40] Yunchao Wei, Wei Xia, Junshi Huang, Bingbing Ni, Jian Dong, Yao Zhao, and Shuicheng Yan. Cnn: Single-label to multi-label. *arXiv preprint arXiv:1406.5726*, 2014. 1
- [41] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 8, 13
- [42] Tong Wu, Qingqiu Huang, Ziwei Liu, Yu Wang, and Dahua Lin. Distribution-balanced loss for multi-label classification in long-tailed datasets. In *European Conference on Computer Vision*, pages 162–178. Springer, 2020. 2
- [43] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning—the good, the bad and the ugly. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4582–4591, 2017. 1
- [44] Hao Yang, Joey Tianyi Zhou, Yu Zhang, Bin-Bin Gao, Jianxin Wu, and Jianfei Cai. Exploit bounding box annotations for multi-label object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 280–288, 2016. 1, 12
- [45] Renchun You, Zhiyao Guo, Lei Cui, Xiang Long, Yingze Bao, and Shilei Wen. Cross-modality attention with semantic graph embedding for multi-label classification. In *AAAI*, pages 12709–12716, 2020. 7
- [46] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit S. Dhillon. Large-scale multi-label learning with missing labels, 2013. 1
- [47] Wenjie Zhang, Junchi Yan, Xiangfeng Wang, and Hongyuan Zha. Deep extreme multi-label learning. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 100–107, 2018. 1
- [48] Yang Zhang, Boqing Gong, and Mubarak Shah. Fast zero-shot image tagging. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5985–5994. IEEE, 2016. 8
- [49] Ke Zhu and Jianxin Wu. Residual attention: A simple but effective method for multi-label recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 184–193, 2021. 2, 3, 5

Appendices

A. Query Types Comparison

Type of Queries	mAP [%]
Learnable	88.1
Fixed NLP-based	88.1
Fixed Random	88.1

Table 7. **Comparison of MS-COCO mAP scores for ML-Decoder with different query types.** We used $K = N = 80$.

B. ZSL Group Decoder

We will now describe how to incorporate group-decoding in the ZSL setting in order to improve the ML-Decoder’s scalability with respect to the number of classes. Group-decoding cannot be trivially applied for ZSL, since in group-decoding each query is associated with a group of labels, while in the ZSL setting we assume that each query is associated with a specific word embedding. To alleviate this issue, we propose to construct each query by concatenating linear projections of all the word embeddings assigned to its group (See Figure 7). Formally, the k^{th} group query is given by

$$q_k = \text{concat}(\{W_a \cdot N_i\}_{i \in \mathcal{G}_k}) \quad (5)$$

where \mathcal{G}_k is the set of labels assigned to the k^{th} group, $\{N_i\}$ is the set of word embeddings ($N_i \in \mathbb{R}^{d_w}$), $W_a \in \mathbb{R}^{\frac{d}{g} \times d_w}$ is the parameter matrix, and $g = \frac{N}{K}$.

In addition, as can be seen in Table 8, the group fully-connected head, even with shared weights, does not generalize well to unseen classes in a group-decoding setting. Therefore, we implemented a different pooling strategy for ZSL group-decoding: we decompose the group fully-connected parameter matrix (W_k from Eq. 3) into two components: $\{N_i\}$ - the set of word embeddings which is label-specific, and $W_b \in \mathbb{R}^{d \times d_w}$ - a learned parameter matrix which is shared for all labels. Formally, the parameter matrix W_k of the k^{th} group-query (Eq. 3) is constructed by the following:

$$W_k = W_b \cdot M_k \quad (6)$$

where $M_k \in \mathbb{R}^{d_w \times g}$ is constructed by stacking the word embedding vectors $\{N_i\}_{i \in \mathcal{G}_k}$ of group k . Inserting Eq. 6 into Eq. 3, we get the output logits for the ZSL group-decoder. (see Fig. 7, and pseudo-code in appendix M). Table 8 shows ablation experiments for the different modifications.

Query Type	Head Type	mAP [%] (ZSL)
NLP	Learnable	2.4
NLP	Learnable (shared)	2.6
Learnable	NLP	23.4
NLP	NLP	28.7

Table 8. **Comparison of NUS-WIDE mAP scores for ML-Decoder with different query types and head types.** ”NLP” signifies using NLP projections to generate query/head parameters, and ”Learnable” signifies using regular parameters (as described in Section 2.3.2)

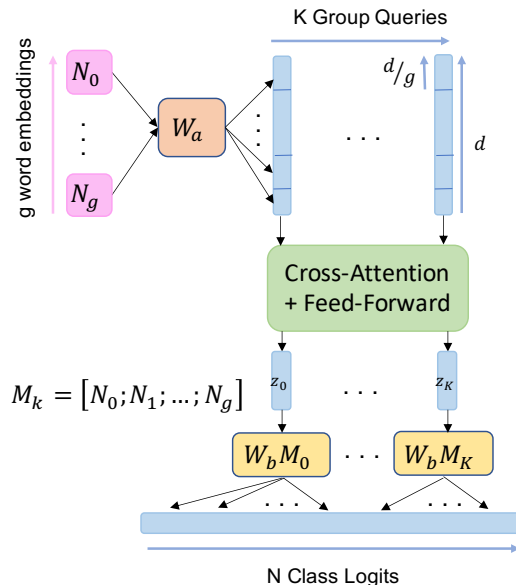


Figure 7. **Group decoding scheme for ZSL.**

C. MS-COCO Training Details

Unless stated explicitly otherwise, for MS-COCO we used the following training procedure: We trained our models for 40 epochs using Adam optimizer and 1-cycle policy, with maximal learning rate of $2e-4$. For regularization, we used Cutout factor of 0.5, True-weight-decay of $1e-4$ and auto-augment. We found that the common ImageNet statistics normalization does not improve results, and instead used a simpler normalization - scaling all the RGB channels to be between 0 and 1. Our input resolution was 448. For ML-Decoder, our baseline was full-decoding ($K = N = 80$). Similar to [2], we used Open Images pre-training for our models’ backbone. ML-Decoder weights were used with random initialization. The number of token embeddings was $D = 768$. We adjusted the backbone embedding output to D via a 1×1 depth-wise convolution. As a loss function, we used ASL with $\gamma_- = 4$, $\gamma_+ = 0$ and

Classification Head	Number of Classes	Number of Queries	Training Speed [img/sec]	Inference Speed [img/sec]	Maximal Training Batch Size	Flops [G]
GAP	100	—	706	2915	520	5.7
	1000	—	703	2910	512	5.7
	5000	—	698	2846	504	5.8
Transformer-Decode	100	100	556	2496	424	6.6
	1000	1000	44	916	112	14.2
	5000	5000	2	17	4	61.1
ML-Decoder	100	100	575	2588	464	6.3
	1000	100	568	2563	456	6.3
	5000	100	562	2512	448	6.4

Table 9. **Comparison of throughput indices for different classification heads.** All measurements were done on Nvidia V100 16GB machine, with mixed precision. We used TResNet-M as a backbone, with input resolution 224. Training and inference speed were measured with 80% of maximal batch size.

$m = 0.05$. All results were averaged over three seeds for better consistency. TResNet-L model is V2¹.

D. Speed Comparison for Different Classification Heads

In Table 9 we provide full speed-accuracy measurements. We can see that in practice, using ML-Decoder on TResNet-M architecture reduces inference speed by 15%, independent with the number of classes, while transformer-decoder classification head scales badly with the number of classes, reducing the inference speed (and other throughput metrics) by orders of magnitudes.

E. MS-COCO Results for Different Input Resolutions

Method	Backbone	Input Resolution	Flops [G]	mAP [%]
ML-Decoder	TResNet-L	224x224	9.3	85.5
ML-Decoder	TResNet-L	448x448	36.2	90.0
ML-Decoder	TResNet-L	640x640	73.5	91.1
ML-Decoder	TResNet-XL	640x640	103.8	91.4

Table 10. **Comparison of MS-COCO mAP scores for different input resolutions.**

F. NUS-WIDE ZSL Dataset and Training Details

NUS-WIDE is a multi-label ZSL dataset, comprised of nearly 270K images with 81 human-annotated categories, in addition to the 925 labels obtained from Flickr user tags.

¹see: <https://github.com/Alibaba-MIIL/TResNet>

The 925 and 81 labels are used as seen and unseen classes, respectively.

To be compatible with previous works [3], as a loss function we used CE, our backbone was TResNet-M, and our input resolution is 224. Unless stated otherwise, our baseline ML-Decoder for ZSL was full-decoding, with $K = N$, and shared projection matrix, as discussed in Section 2.3.3. Other training details for ZSL NUS-WIDE were similar to the one used for MS-COCO.

G. Pascal-VOC Training Details and Results

Pascal Visual Object Classes Challenge (VOC 2007) is another popular dataset for multi-label recognition. It contains images from 20 object categories, with an average of 2.5 categories per image. Pascal-VOC is divided to a trainval set of 5,011 images and a test set of 4,952 images. As a backbone we used TResNet-L, with input resolution of 448. For ML-Decoder, our baseline was full-decoding ($K = N = 20$). Other training details are similar to the ones used for MS-COCO. Results appear in Table 11.

Method	mAP [%]
RNN [39]	91.9
FeV+LV [44]	92.0
ML-GCN [8]	94.0
SSGRL [6]	95.0
BMML [23]	95.0
ASL [2]	95.8
Q2L [25]	96.1
ML-Decoder	96.6

Table 11. **Comparison of ML-Decoder to known state-of-the-art models on Pascal-VOC dataset.** For ML-Decoder, we used TResNet-L backbone, input resolution 448.

H. Open-Images Training Details and Results

Open Images (v6) [21] is a large-scale dataset, which consists of 9 million training images, 41,620 validation images and 125,436 test images. It is partially annotated with human labels and machine-generated labels. For dealing with the partial labeling methodology of Open Images dataset, we set all untagged labels as negative, with reduced weights. Due to the large the number of images, we trained our network for 25 epochs on input resolution of 224. We used TResNet-M as a backbone. Since the level of positive-negative imbalancing is significantly higher than MS-COCO, we increased the level of loss asymmetry: For ASL, we trained with $\gamma_- = 7, \gamma_+ = 0$. For ML-Decoder, our baseline was group-decoding with $K = 100$. Other training details are similar to the ones used for MS-COCO.

Method	mAP [%]
CE [2]	84.8
Focal Loss [2]	84.9
ASL [2]	86.3
ML-Decoder	86.8

Table 12. Comparison of ML-Decoder to known state-of-the-art results on Open Images dataset.

I. Single-label Classification with Different Logit Activations

Logit Activation	Classification Head	Top1 Acc. [%]
Sigmoid	GAP	79.7
	ML-Decoder	80.3
Softmax	GAP	79.3
	ML-Decoder	80.1

Table 13. ImageNet classification scores for different classification heads and logits activations. For ML-Decoder, we used group-decoding with 100 groups. Our training configuration is A2 [41]. Backbone - ResNet50.

J. Comparison of ML-Decoder to State-of-the-art Models on Single-label Transfer Learning Datasets

In this section, we will compare our ML-Decoder based models to known state-of-the-art models from the literature, on two prominent single-label datasets - CIFAR-100 [20] and Stanford-Cars [19]. The comparison is based on ² and ³.

²<https://paperswithcode.com/sota/image-classification-on-cifar-100>

³<https://paperswithcode.com/sota/fine-grained-image-classification-on-stanford>

Dataset	Model	Top-1 Acc.
CIFAR-100	CvT-W24	94.05
	ViT-H	94.55
	EffNet-L2 (SAM)	96.08
	Swin-L + ML-Decoder	95.1
Stanford-Cars	EffNet-L2 (SAM)	95.95
	ALIGN	96.13
	DAT	96.2
	TResNet-L + ML-Deocder	96.41

Table 14. Comparison top of state-of-the-art models.

We can see from Table 14 that our ML-Decoder based solution is highly competitive, achieving 1st and 2nd place on Stanford-Cars and CIFAR-100 datasets respectively.

K. Query Augmentations Illustration

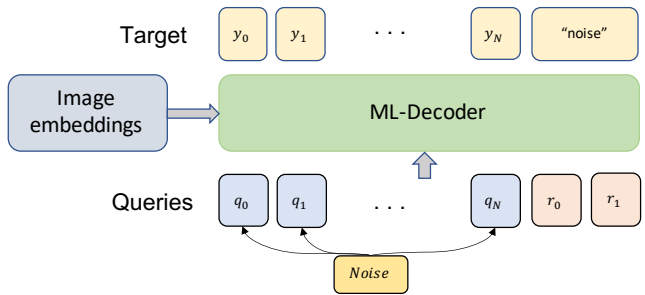


Figure 8. Query augmentations *rand-query*: adding random queries which are assigned label "noise". *additive noise*: adding random noise to the input queries.

L. Group Fully-connected Pseudo Code

```
1 def GroupFullyConnected(G, group_weights, output, num_of_groups):
2     '''
3     - G is the group queries tensor. G.shape = [groups, embeddings]
4     - group_weights are learnable (group) fully connected weights.
5     - group_weights.shape = [groups, embeddings, classes//groups]
6     - output is the interpolated queries tensor. output.shape = [groups, classes//groups]
7     '''
8     for i in range(num_of_groups):
9         g_i = G[i, :] # [1, embeddings]
10        w_i = group_weights[i, :, :] # [embeddings, classes//groups]
11        output_i = matmul(g_i, w_i) # [1, classes//groups]
12        output[i, :] = output_i
13    logits = output.flatten(1)[:self.num_classes] # [1, classes]
14    return logits
```

Notice that the loop implementation is very efficient in terms of memory consumption during training. Implementing the group-fully-connected in a single vectoric operation (without a loop) is possible, but reduces the possible batch size. Also, the proposed implementation is fully suitable for compile-time acceleration (@torch.jit.script)

M. Group Fully-connected ZSL Pseudo-Code

```
1 def GroupFullConnectedZSL(G, wordvecs, output, W, num_groups, num_classes):
2     '''
3     - G is the group queries tensor. G.shape = [num_groups, embeddings_dim]
4     - wordvecs is the word-embedding tensor [num_classes, word_embedding_dim]
5     - W is a learnable projection matrix [embeddings_dim, word_embedding_dim]
6     - output is the interpolated queries tensor. output.shape = [num_groups, num_classes//num_groups]
7     '''
8     labels_per_group = num_classes // num_groups
9     group_weights = zeros(num_groups, embedding_dim, labels_per_group)
10    for i in range(num_classes):
11        group_weights[i // labels_per_group, :, i % labels_per_group] = W * wordvecs[i, :]
12
13    logits = GroupFullyConnected(G, group_weights, output, num_groups)
14    return logits
```
