Improving Image Recognition by Retrieving from Web-Scale Image-Text Data

Ahmet Iscen Alireza Fathi Cordelia Schmid
Google Research

Abstract

Retrieval augmented models are becoming increasingly popular for computer vision tasks after their recent success in NLP problems. The goal is to enhance the recognition capabilities of the model by retrieving similar examples for the visual input from an external memory set. In this work, we introduce an attention-based memory module, which learns the importance of each retrieved example from the memory. Compared to existing approaches, our method removes the influence of the irrelevant retrieved examples, and retains those that are beneficial to the input query. We also thoroughly study various ways of constructing the memory dataset. Our experiments show the benefit of using a massive-scale memory dataset of 1*B image-text pairs, and demonstrate the performance of different memory representations. We evaluate our method in three different classification tasks, namely long-tailed recognition, learning with noisy labels, and fine-grained classification, and show that it achieves state-of-the-art accuracies in ImageNet-LT, Places-LT and Webvision datasets.*

1. Introduction

Increasing the number of parameters of large transformer models has been a recent successful trend achieving new benchmarks in vision and language tasks. Recent results from T5 [35], GPT-3 [5], PaLM [8], CoCa [50], Flamingo [1], BEIT-3 [46], PaLI [7], Florence [51] and FLAVA [40] show that transformer models are able to store a surprising amount of information when scaled to tens of billions of parameters and trained on vast text and image corpora. These so-called ‘foundation models’ achieve state-of-the-art results when fine tuned and applied to secondary tasks such as language modeling, image captioning, visual question answering and open vocabulary recognition.

In these foundation models, the learned world knowledge is stored implicitly in the parameters of the underlying neural network. This implies that some of the problems of the current ML paradigm are amplified in these models: (a) scaling is challenging, both in learning and serving, given the large number of parameters that are required for storing the knowledge, (b) it is hard to update the model as the world facts change or input data gets modified, (c) these models tend to be black box, which means it is hard to interpret the underlying reason behind their decisions.

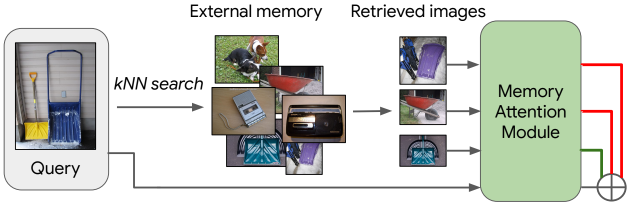

Figure 1. Retrieval augmented classification finds similar images to the query from an external memory. Our memory attention module learns the importance of each retrieved image by assigning high weights (green line in the figure) to the relevant images, and low weights (red line) to the irrelevant images.

To address the above issues, we propose an alternative perspective on the problem. Instead of compiling the world knowledge statically into model weights, we take an interpretive view where the world knowledge gets transformed into a massive-scale index/memory. On the other hand, a relatively low-compute small model learns to use the memory for the given inference task. Instead of increasing the size of the model and training on more data as done in most previous work, we equip models with the ability to directly access a large database to perform predictions—a semi-parametric approach.

To evaluate our approach, we focus on the problem of long-tailed recognition and learning with noisy labels. The distribution of real-world data is often noisy, imbalanced and highly skewed on a per-class basis, with a majority of classes containing a small number of samples. Long-tailed recognition is a well-studied problem [19, 33]. Base approaches are largely variants of the core idea of “adjustment”, where the learner is encouraged to focus on the tail of the distribution. This is achieved either by re-weighting samples during training [21] and cluster-based sampling [10], logit or loss modification [12, 20, 31, 54] or ensembling [47]. Despite being well-studied, commonly occurring, and of great practical importance, classification performance on long-tail distribu-

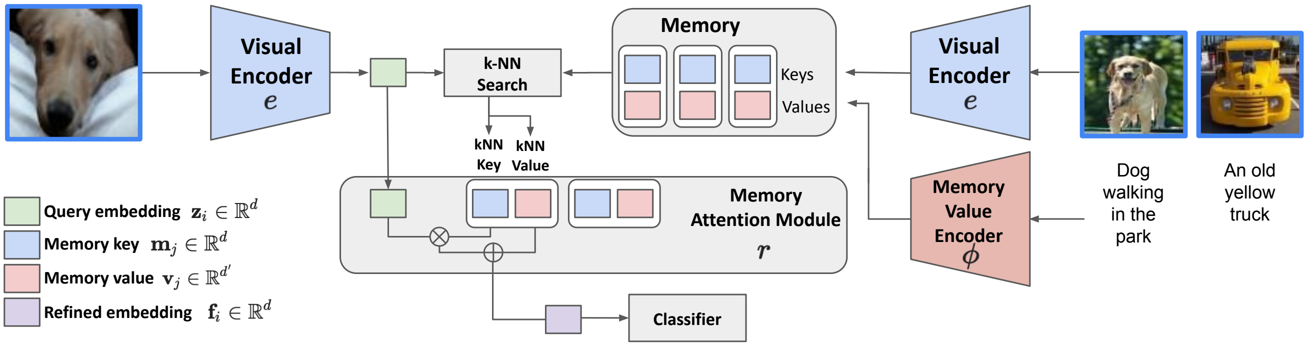
Visual Encoder e
k-NN Search
kNN Key
kNN Value
Memory
Keys
Values
Visual Encoder e
Memory Value Encoder ϕ
Dog walking in the park
An old yellow truck
Memory Attention Module r
Query embedding $\mathbf{z}_i \in \mathbb{R}^d$
Memory key $\mathbf{m}_j \in \mathbb{R}^d$
Memory value $\mathbf{v}_j \in \mathbb{R}^{d'}$
Refined embedding $\mathbf{f}_i \in \mathbb{R}^d$
Classifier

Figure 2. **Overview of our method.** Retrieval augmented classification aims to retrieve relevant images from an external memory dataset when making predictions. Each example in the memory is composed of a *key* and *value* embedding pair. Key embeddings are extracted using the same visual encoder as the query image, but the value embeddings can be extracted with any other encoder. Both visual and value encoders are remain frozen during the training. We perform an approximate k-NN search between the query embedding and memory keys to find relevant images from the memory dataset. The retrieval module receives the query embedding and the k retrieved key-value pairs from the memory. We learn the importance of each memory example by computing the attention weights between the query embedding and the memory keys. The memory values, weighted by their corresponding attention weights, are used to compute the refined embedding, which is then passed to the classifier.

tions lags significantly behind the state of the art for better balanced classes [15, 49, 55].

Long et al. [29] introduce a retrieval-augmented classification model that explicitly stores the tail knowledge. In comparison to this work, we suggest to retrieve from a web-scale vision-text database and augment the input query with the retrieved knowledge using a memory attention module, before making class predictions. We design the external memory as pairs of key-value embeddings. These embeddings are computed by encoding vision and language data from multiple sources (Web images with alt-text such as Webli [7], LAION [39], YFCC100M [42] datasets as well as image classification datasets like ImageNet [37]). Memory key embeddings are used to retrieve the k-nearest neighbors of the input query vectors. Our memory attention module learns the importance of each retrieved memory example by computing attention weights between the query embedding and memory keys. Relevant examples have more influence, whereas the contribution of the irrelevant noisy examples is down-weighted. Learned attention weights are then used to combine memory values and produce a refined embedding, which is then used to make class predictions. Figure 1 shows a high-level visualization of our method.

Our contributions are summarized as follows:

- We propose a retrieval-augmented recognition model that explores efficient means of augmenting visual models with a massive-scale memory without significantly increasing computations.
- We propose a simple yet powerful way to fuse the retrieved knowledge with the input query using a memory attention module.
- Our method achieves state-of-the-art results on various benchmarks, such as long-tail recognition and learning with noisy labels. We achieve 78.9 accuracy on ImageNet-LT dataset, 50.3 accuracy on Places-LT dataset, and 83.6 on Webvision dataset.

2. Related work

External memory collections have been used for various tasks in computer vision and other domains such as NLP. One of the earliest works combining deep network with an external memory is *Neural Turing Machines* [14], where an external memory is updated with *write* and *erase* operations and a learned controller. Santoro *et al.* [38] propose MANN (memory-augmented neural network), where a differentiable external memory is utilized for meta-learning.

In the NLP domain, memory-based methods have been used to access external large-scale datasets. Khandelwal *et al.* [24] propose to interpolate the outputs of a trained language model with a non-parametric k-NN model. REALM [18] retrieves external knowledge from Wikipedia for question answering. Lewis *et al.* [26] use an external memory to generate questions and answers, which are then used for question answering. Wang *et al.* [45] retrieve nearest neighbors of each training example from the training set, and combine each input with the retrieved content. They show the benefit of this approach in various NLP tasks, such as summarization, language modeling and machine translation. Wu *et al.* [48] define the memory as previously seen words in the same document, and learn how to combine them with the input tokens in a transformer.

RETRO [4] systematically evaluates the impact of large-scale external memory datasets for NLP tasks. Our paper is similar to RETRO [4], in that we also utilize a large-scale external memory, but in the vision domain. Similar to RETRO, we use frozen feature extractors to reduce the complexity of large-scale k-NN search, in order to focus on the benefits of massive-scale external knowledge sources.

Recent methods in computer vision also make use of external memory for various tasks. Iscen *et al*. [22] use a memory to store previously seen examples in incremental learning. Nakata *et al*. [32] store feature maps from the training set in the memory, and perform k-NN for classification. Chen *et al*. [6] and Blattmann *et al*. [3] retrieve nearest neighbors from a memory for generative vision models. Basu *et al*. [2] study the generalization of retrieval-based models from a theoretical perspective.

Perhaps the most similar method to our own is *Retrieval Augmented Classification* (RAC) [29]. The authors combine the output of a *base* model, *i.e*. a typical vision encoder, and the retrieval module. The retrieval module is learned by first finding k-NN of the visual input from the memory based on visual embeddings. Then the corresponding raw text labels of the k-NN are concatenated, and a textual embedding is extracted with a pre-trained CLIP model [34].

Our work is different in that we do not assume that every retrieved example has the same importance. By concatenating the raw text labels of each retrieved example in a single sequence of text, RAC assigns the same importance to each retrieved item. In contrast, we explicitly learn the contribution of each retrieved item, and weight them accordingly. Additionally, RAC uses the training set itself as the external memory. In our work, we rigorously evaluate different candidate datasets of varying scales for the external memory. Our experiments use memory datasets up to 1B images, and show that larger memory datasets show benefits.

3. Method

Retrieval augmented classification aims to enhance the query input by retrieving relevant images from an external memory. In this section we first formulate our task, then propose different alternatives to fuse the query and the retrieved information. Figure 2 shows an overview of our method.

Problem formulation. Let us define a *downstream* dataset of N images by $X := \{x_1, \ldots, x_N\}$. Our task is *supervised* classification, meaning that each image is accompanied by its label $Y := (y_1, \ldots, y_N)$ with $y_i \in \mathbb{R}^C$, where C is the number of classes. In a typical classification problem, our goal is to learn a model which takes an input image x_i, and maps it to class prediction scores, *i.e. logits*. The model consists of two parts; the visual encoder, and the classifier. The visual encoder $e : x \to \mathbb{R}^d$ takes an input image x_i and maps it to a d-dimensional vector, *i.e.* $\mathbf{z}_i := e(x_i) \in \mathbb{R}^d$. The resulting feature embedding $\mathbf{z}_i$ is then passed to the classifier $h : \mathbf{z} \to \mathbb{R}^C$ to obtain the logits. The model output, *i.e*. logits, are denoted as:

$$f(x_i) = h(e(x_i)) = h(\mathbf{z}_i). \tag{1}$$

The model parameters are trained by minimizing any supervised loss function, such as cross-entropy, or LACE loss [31] when the training data is imbalanced.

3.1. Retrieval augmented classification

Typically, classification models are trained to make predictions only considering the images x_i in the downstream dataset (1). The classifier $h(.)$ takes a single input x_i and outputs the class logits.

Retrieval augmented classification aims to train more robust and accurate models by leveraging relevant information from an external source of knowledge, *i.e*. a *memory dataset* $M := \{m_1, \ldots, m_L\}$, for each downstream image x_i. More specifically, the model predictions now also depend on M, in addition to x_i. Note that M is collected independently from X. It is therefore not guaranteed that it will contain relevant information w.r.t. to the each $x_i \in X$. We also do not assume that the memory dataset M contains class labels, but the images may be accompanied by additional information, such as free-form text.

In practice, only the most relevant subset of M is directly used for classification for a given training image x_i. Let $\mathbf{M} = [e(m_i), \ldots, e(m_L)]$ be the set of feature embeddings of each image in M. We compute the cosine similarity between $\mathbf{z}_i$ and each embedding $\mathbf{m}_j \in \mathbf{M}$ to find the k-nearest neighbors. The top-k ranked embeddings are then used during the prediction:

$$f(x_i) = h(r(\mathbf{z}_i, \mathbf{M}_{\mathrm{NN}(\mathbf{z}_i;\mathbf{M})})), \tag{2}$$

where $\mathbf{M}_{\mathrm{NN}(\mathbf{z}_i;\mathbf{M})}$ denotes top-k ranked embeddings of $\mathbf{z}_i$ from $\mathbf{M}$, and $r(.,.)$ is a *retrieval module*, which learns how to combine $\mathbf{z}_i$ with the vectors from $\mathbf{M}_{\mathrm{NN}(\mathbf{z}_i;\mathbf{M})}$. Different choices for $r(.,.)$ will be discussed in Section 3.2. Unlike (1), the retrieval augmented model (2) makes predictions while directly leveraging the information from M.

Co-embedded memory. Long *et al*. [29] show that different types of embeddings can be extracted from the memory. This allows us to utilize even more additional information, *e.g*. different modalities, corresponding to the memory examples. We will now describe this scenario in more detail.

Let us assume that there are two sets of embeddings corresponding to each example m_i in the memory. *Memory keys* $\mathbf{M}$, as defined above, are extracted using the same visual encoder $e(.)$ as $\mathbf{z}_i$. Let us also define *memory values* as a set of vectors $\mathbf{V} = [\phi(m_1), \ldots, \phi(m_L)]$, extracted with an encoder $\phi : x \to \mathbb{R}^{d'}$. Note that the output dimensionality of $e(.)$ and $\phi(.)$ are not necessarily equal. As before, the k-NN indices are obtained by computing the cosine similarity between $\mathbf{z}_i$ and $\mathbf{M}$. However, we now select the rows of $\mathbf{V}$

that correspond to the indices of the k-NN search to make the prediction:

$$f(x_i) = h(r(\mathbf{z}_i, \mathbf{V}_{\text{NN}(\mathbf{z}_i;\mathbf{M})})), \tag{3}$$

where $\mathbf{V}_{\text{NN}((\mathbf{z}_i;\mathbf{M})}$ denotes that the k-NN search is done between $\mathbf{M}$ and $\mathbf{z}_i$, but the corresponding indices from $\mathbf{V}$ are selected.

In practice, *memory values* can be extracted from various different types of encoders. One can use a larger visual encoder model, such as ViT-G/14 [52], to extract more robust visual embeddings or use a text encoder, such as T5 [35], to take advantage of a different modality.

3.2. Retrieval fusion module

The retrieval module $r(.,.)$ combines the original query vector $\mathbf{z}_i$ with the retrieved memory values $\mathbf{V}_{\text{NN}(\mathbf{z}_i;\mathbf{M})}$. In this section, we first introduce a simple method which is based on the mean of the retrieved memory values. We then introduce a more powerful method, which learns the amount of contribution each memory value embedding makes to the final prediction. For the sake of simplicity, we drop the $\mathbf{z}_i$ from the notation of memory keys and values in this section, *i.e.* $\mathbf{M}_{\text{NN}}$ and $\mathbf{V}_{\text{NN}}$ denote the k-NN memory keys and values of $\mathbf{z}_i$, respectively.

Mean k-NN fusion module. This method simply computes the mean of retrieved memory values to make the final prediction. The refined output embedding is defined as:

$$r(\mathbf{z}_i, \mathbf{V}_{\text{NN}}) = \mathbf{z}_i + \chi\left(\frac{1}{k}\sum_{\mathbf{v}\in V_{\text{NN}}} \mathbf{v}\right), \tag{4}$$

where $\chi : \mathbb{R}^{d'} \to \mathbb{R}^d$ is a dense layer which maps the mean of memory value embeddings from the d'-dimensional vector space to the initial d-dimensional vector space. We use this method as a baseline in our experiments. It demonstrates the impact of using the retrieved memory values without learning their importance.

Memory attention module (MAM). It is not ideal to assume that every vector in $\mathbf{V}_{\text{NN}}$ has the same importance. Certain memory values may be more relevant for $\mathbf{z}_i$, whereas others may bring noise. We propose to compute attention weights between the query vector $\mathbf{z}_i$ and the retrieved memory keys $\mathbf{M}_{\text{NN}}$, which lie in the same feature space, to learn the contribution of each vector in $\mathbf{V}_{\text{NN}}$:

$$r(\mathbf{z}_i, \mathbf{M}_{\text{NN}}, \mathbf{V}_{\text{NN}}) = \mathbf{z}_i + \chi\left(\sigma\left(\frac{\psi_Q(\mathbf{z}_i)\psi_K(\mathbf{M}_{\text{NN}})}{\sqrt{d}}\right)\mathbf{V}_{\text{NN}}\right), \tag{5}$$

where σ is the softmax, $\psi_Q : \mathbb{R}^d \to \mathbb{R}^d$ and $\psi_K : \mathbb{R}^d \to \mathbb{R}^d$ are dense layers, and $\chi : \mathbb{R}^{d'} \to \mathbb{R}^d$ is another dense layer which maps the output back to the original input space.

Note that Eq (5) can be repeated L times, *i.e.* L layers. Let $\mathbf{f}_i^1 = r(\mathbf{z}_i, \mathbf{M}_{\text{NN}}, \mathbf{V}_{\text{NN}})$ denote the output of the first layer. The output after L layers can be computed as:

$$\mathbf{f}_i^L = \mathbf{z}_i + \chi\left(\sigma\left(\frac{\psi_Q(\mathbf{f}_i^{L-1})\psi_K(\mathbf{M}_{\text{NN}})}{\sqrt{d}}\right)\mathbf{V}_{\text{NN}}\right). \tag{6}$$

Similar attention mechanisms are used for different purposes, such as when learning the relationship between the query vector and class vectors [43]. We show in our experiments that MAM is an excellent option for retrieval augmented classification as well, significantly outperforming the other baselines while achieving state-of-the-art accuracy in various tasks.

4. Memory

In this section, we first discuss different ways of constructing the memory dataset M. We then describe different ways of computing the embeddings for memory keys ($\mathbf{M}$) and values ($\mathbf{V}$).

4.1. Memory datasets

We use various memory datasets of different sizes in our experiments. We will now describe them in more detail. Each of the choices below are thoroughly evaluated in Section 5.2.

Downstream dataset. This is the most straightforward choice for choosing the memory dataset. Under this setting, the memory set M and the downstream dataset X are the same. This guarantees that there will be at least one relevant memory example m_i for each x_i, as most datasets have multiple instances of each class. One disadvantage of this choice is that most of the downstream datasets do not contain rich free-form text descriptions. They have textual class labels, but those do not change between the different instances of the same class, and consequently are not very discriminative.

YFCC. The Yahoo Flickr Creative Commons dataset [42] contains approximately 100M images. Each image is accompanied by various metadata, including free-form text descriptions. We use the subset of 15M images as the recent work [34, 53]. This subset is created by choosing images that have English text of high quality.

LAION. LAION dataset [39] contains 400M image-text pairs. The dataset was built by gathering image-text pairs from random web pages. The low-quality pairs are removed by computing the cosine similarity between their CLIP [34] embeddings. This results in 400M image-text pairs collected from the web.

WebLI. WebLI dataset [7] contains over 10B pairs of image-text pairs from 100 languages. It is built from publicly available image and text data from web pages. It contains various metadata for each image, including free-form text description. We use the subset of 1B images used to train the PaLI model [7]. This subset is created by scoring image-text pairs based on cross-model similarity. Then a threshold is applied on cross-modal similarity scores, which ends up retaining about 1B image-text pairs.

All. This is the combination of all the memory datasets mentioned above, *i.e.* downstream dataset, YFCC, LAION and WebLI. This variant has approximately 1.5B image-text pairs in the memory dataset.

4.2. Memory keys and values

The parameters of the visual encoder $e(.)$, which is used to extract query vectors $\mathbf{z}_i$ and memory keys $\mathbf{M}$, are frozen during training. This choice allows us to efficiently use memory datasets with up to 1B images, as the memory keys $\mathbf{M}$ are computed and indexed only once offline for efficient k-NN search with the query vector $\mathbf{z}_i$. A similar strategy has been employed in RETRO [4], where the authors show the benefit of retrieval augmentation with frozen text embeddings. We use the ViT-B/16 [13] model trained on the JFT-3B dataset [52] for the visual encoder $e(.)$, as it has shown to be a powerful vision encoder.

We explore different choices of $\phi(.)$ to compute memory values $\mathbf{V}$. Memory value encoder $\phi(.)$ is also fixed, and its parameters are frozen during training. This allows us to use very large models for $\phi(.)$, as the memory values $\mathbf{V}$ are only computed once offline, and the actual model is not needed during the training or inference.

Visual encoders. We choose more powerful and bigger visual encoders as $\phi(.)$. More specifically, we choose pre-trained ViT-L/16, ViT-g/14 and ViT-G/14 architectures [52] to compute $\mathbf{V}$. ViT-G/14 has 2B parameters, making it challenging to load it in a GPU memory during the training. However, because $\mathbf{V}$ is computed offline, we only access the ViT-G/14 embeddings, not the model itself, during the training.

Text encoders. Another way of extracting the memory values is to exploit other modalities from M. We use the pre-trained T5-Base [35] text encoder, to extract textual embeddings as $\mathbf{V}$. If the memory set does not have free-form text descriptions, *e.g.* if the memory set is the downstream dataset, we extract textual embeddings from the text labels of each image. Text labels are turned into sentences by using pre-defined prompts [34].

4.3. Retrieval complexity

Our method requires a k-NN search between the input query vector and the memory keys. We use the SCaNN library [16] to perform approximate k-NN search in our experiments. It has a sublinear complexity, meaning that it takes $\mathcal{O}(\log M)$ for a memory dataset of M elements. In practice, querying a memory dataset of 1B elements takes only milli-seconds. Because we use a fixed vision encoder in our experiments, we pre-compute and save all k-NN, which speeds up training time.

5. Experiments

In this section, we first describe the downstream datasets used in our experiments and detail our experimental setup. We then conduct a rigorous study showing the impact of different memory datasets and memory value encoders. Finally, we compare our results against the existing methods in the literature.

5.1. Experimental setup

We report experiments for three different image classification tasks: long-tailed recognition, learning with noisy labels, and fine-grained classification. We now describe the downstream datasets that we use for each of these tasks.

Long-tailed Recognition. Long-tailed recognition assumes that there is a strong imbalance in terms of images per class in the training set. The goal is to learn robust classifiers that can accurately classify every class, regardless of the number of times it appears in the training set.

We use two datasets for this task: ImageNet-LT [28] and Places-LT [28]. ImageNet-LT has 1000 classes and the number of training images per class varies from 5 to $1,280$. It is created by taking a subset of the original ImageNet dataset [11], so that the number of images per class follows a long-tailed distribution.

Similarly, Places-LT is created by taking a subset of the original Places365 dataset [56], such that the number of training images per class follows a long-tailed distribution. There are 365 classes in this dataset, and the number of images per class varies from 5 to $4,980$.

The validation sets for both datasets are balanced. We report the top-1 overall accuracy for both datasets. We also report the accuracy for *many-shot* classes (more than 100 images per class), *mid-shot* classes (between 20 and 100 images) and *few-shot* classes (less than 20 images), separately, following the protocol in [28].

Learning with noisy labels. The Webvision dataset [27] contains 2.4M images and 1000 classes. The data is collected from the web, and the labels are assigned without human supervision. Therefore, some of the labels are noisy. Training set is imbalanced, meaning that there are different number of examples for each class. We report the top-1 overall accuracy for the validation set.

Fine-grained classification. We use the iNaturalist2021-Mini dataset [44] for this task. This dataset contains fine-grained images of species, *e.g.* insects, plants, birds *etc.* There are $10,000$ classes and 50 images per class, making it a total number of $500,000$ training images. The validation set contains $100,000$ images. We report the top-1 overall accuracy.

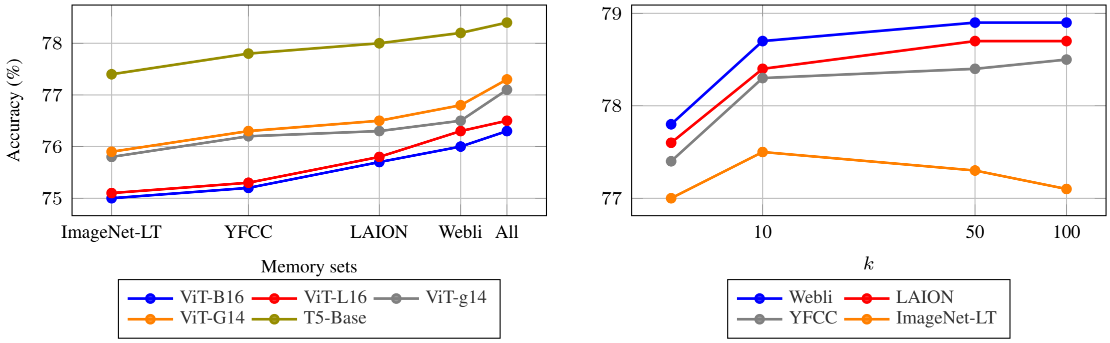

Figure 3. **Ablation study on ImageNet-LT. Left**: We show the impact of different memory sets and memory value encoders. We set $k = 100$ for this experiment. **Right**: We show the impact of k for different memory sets. We use T5-Base to represent memory values for this experiment.

Implementation details. We use a frozen ViT-B/16 as the visual encoder $e(.)$ to represent the query vectors from the downstream datasets and memory key embeddings. Unless otherwise specified, training lasts 10 epochs, with a learning rate of 0.001 and batch size of 512. The learning rate follows a warm-up schedule of 1 epoch, and then is reduced in each epoch using a cosine decay schedule [30]. We use an Adam optimizer [25] with a weight decay of 0.2. We also use label smoothing [41] during training to prevent over-fitting. For the Memory Attention Module (MAM) (5), we use 8 layers, *i.e.* $L = 8$. We retrieve $k = 100$ examples from the memory, unless otherwise specified.

5.2. Impact of the memory

We study the effect of different choices for the memory construction in this section. Section 4 introduces various memory datasets and memory value encoders in detail. We now investigate how different memory datasets and value encoders behave in the ImageNet-LT downstream dataset.

Figure 3 (left) shows the impact of different memory value encoders in different memory datasets. We set $k = 100$ for this experiment. We see that the textual memory value encoder (T5-Base) obtains a better accuracy compared to visual memory value encoders. Even much larger visual memory value encoders, such as ViT-G/14, does not outperform a smaller T5-Base model. We believe that this behavior is due to the fact that T5-Base represents a different modality (text), which otherwise is not available to the input. Textual memory values are thus complementary to the visual signals, which are extracted from the input query in any case.

Figure 3 (left) also shows that the accuracy generally improves as the size of the memory dataset becomes larger. Larger relative improvements are especially observed for visual memory value encoders as the size of the memory dataset increases. The accuracy for visual memory value encoders continues to increase even when all the memory datasets are combined (≈ 1.5B examples).

Figure 3 (right) shows the impact of k for different memory sets. This hyperparameter controls the number of keys and values retrieved from the memory dataset. We use textual memory value embeddings (T5-Base) for this experiment. We see that the large memory datasets, *e.g.* YFCC, LAION, Webli, are less sensitive to the choice of k, whereas the accuracy starts to decrease for a smaller memory dataset such as ImageNet-LT. That is because ImageNet-LT only has a few positive examples for certain images. As k becomes larger, the retrieved set of vectors mostly contain noise. That is not the case for larger memory datasets, as they are likely to have many relevant examples for each image. Thus, they are less sensitive to larger k.

5.3. Comparison to baselines

In this section, we show the benefit of our Memory Attention Module (MAM) by comparing it with different baselines. We report the accuracy for the following baselines. *Linear* classifier learns a fully connected layer on top of frozen downstream dataset embeddings. *MLP* classifier is a two-layer MLP with non-linearity in between the two layers. Linear and MLP classifiers do not use an external memory dataset for retrieval. *Mean k-NN* computes the mean of the retrieved memory values; it do not learn the contribution of each retrieved memory value. See Section 3 for more details. We use the WebLI memory dataset and T5-Base memory values for this experiment.

Table 1 (Top) shows the accuracy for many-shot, mid-shot, and low-shot classes separately for all the baselines on ImageNet-LT and Places-LT datasets. All the methods are trained with the LACE [31] loss, which has a balancing effect between the low-shot and many-shot classes. Nevertheless, the low-shot accuracy suffers for the methods that do not use retrieval, *e.g.* linear and MLP classifiers.

On the other hand, retrieval-based methods, *e.g.* mean k-NN and MAM (Ours), are less prone to over-fitting on many-shot classes. However, mean k-NN overcompensates

			ImageNet-LT				Places-LT			
Method	Retrieval	Backbone	Many-shot	Mid-shot	Low-shot	All	Many-shot	Mid-shot	Low-shot	All
			BASELINES							
Linear Classifier		ViT-B16 🔒	76.5	72.6	66.5	73.5	44.5	44.4	44.0	44.3
MLP Classifier		ViT-B16 🔒	80.1	74.1	66.9	75.2	48.6	46.1	41.3	46.0
Mean k-NN	✓	ViT-B16 🔒	75.9	75.8	**75.7**	75.8	44.3	45.2	45.5	44.9
			EXISTING METHODS							
PaCo [10]		ResNext-101	68.2	58.7	41.0	60.0	36.1	47.9	35.3	41.2
VL-LTR [43]		ViT-B16	84.5	74.6	59.3	77.2	**54.2**	48.5	42.0	50.1
RAC [29]	✓	ViT-B16	-	-	-	-	48.7	48.3	41.8	47.2
RAC† [29]	✓	ViT-B16 🔒	80.9	76.0	67.5	76.7	50.3	48.3	42.5	47.9
RAC† [29]	✓	ViT-B16	**85.9**	79.3	69.3	80.5	51.9	49.8	46.8	50.0
Ours	✓	ViT-B16 🔒	80.6	77.5	74.5	78.3	50.9	49.9	47.5	49.9
Ours + FT	✓	ViT-B16	85.4	**81.5**	**76.4**	**82.3**	52.4	**52.0**	**48.5**	**51.4**

Table 1. **Comprehensive evaluation on ImageNet-LT and Places-LT.** The accuracy for many-shot ($>$ 100 images), mid-shot (20-100 images) and few-shot ($<$ 20 images) classes are reported separately. **Top:** We report the results for various baselines. **Bottom:** We compare our method against the existing methods in the literature. RAC† denotes our re-implementation of RAC [29] in exactly the same setting as our method. 🔒 means that the visual encoder is frozen during the downstream task training.

for the low-shot classes by sacrificing the accuracy for the many-shot classes on ImageNet-LT. This is not the case for our method, which achieves the highest overall accuracy by performing well across all three class categories. Similar observations can be made in the Places-LT dataset. Our method achieves the highest accuracy on many-shot, mid-shot and low-shot classes, and the highest accuracy overall.

The experiments on Table 1 (Top) demonstrate that the retrieval augmented classification alone does not always improve the classification accuracy. This is evidenced by the performance of *mean k-NN*. On the other hand, as we learn the contribution of the each retrieved example from the memory, we are able to filter out the noisy examples more accurately. This results in the highest accuracy overall, while not sacrificing the accuracy for the many-shot, mid-shot and low-shot classes.

Table 2 shows the comparison of our method against the baselines for fine-grained classification (iNaturalist2021-Mini) and learning with noisy labels (Webvision). We use WebLI as the memory dataset, and T5-Base as the memory value encoder for these experiments. We observe that our method displays consistent improvement in both datasets. Note that it overperforms the state-of-the-art in Webvision, without finetuning the visual encoder like the existing work. This shows that our method is suitable for various classification tasks, and not only long-tailed recognition.

	iNat2021-Mini	WebVision
	BASELINES	
Linear Classifier	58.8	78.1
MLP Classifier	59.6	81.0
Mean k-NN	58.9	78.2
	EXISTING METHODS	
MILe [36]	–	75.2
Heteroscedastic [9]	–	76.6
NCR [23]	–	76.8
CurrNet [17]	–	79.3
Ours	**66.2**	**83.6**

Table 2. **Evaluation on iNaturalist2021-Mini and Webvision.** We compare our method against the baselines and existing work in iNaturalist2021-Mini (fine-grained classification) and Webvision (learning with noisy labels) downstream datasets.

5.4. Comparison to existing methods

We now compare our method against the state-of-the-art. Table 1 (Bottom) shows the accuracy of the prior art in ImageNet-LT and Places-LT datasets. VL-LTR [43] and RAC [29] use the same ViT-B/16 backbone as our method. However the pre-training of the ViT-B/16 differs between different methods. VL-LTR uses the ViT-B/16 pre-trained with CLIP [34], whereas RAC uses an ImageNet-21k pre-trained ViT-B/16 architecture [13]. Both methods finetune the visual encoder on the downstream dataset. In this paper, we use a ViT-B/16 visual encoder pre-trained on the JFT-3B dataset [52]. We also re-implement RAC with our visual and text encoder (T5-Base) for a better comparison in Table 1, and denote this variant as RAC†.

Table 1 shows that the VL-LTR achieves the highest

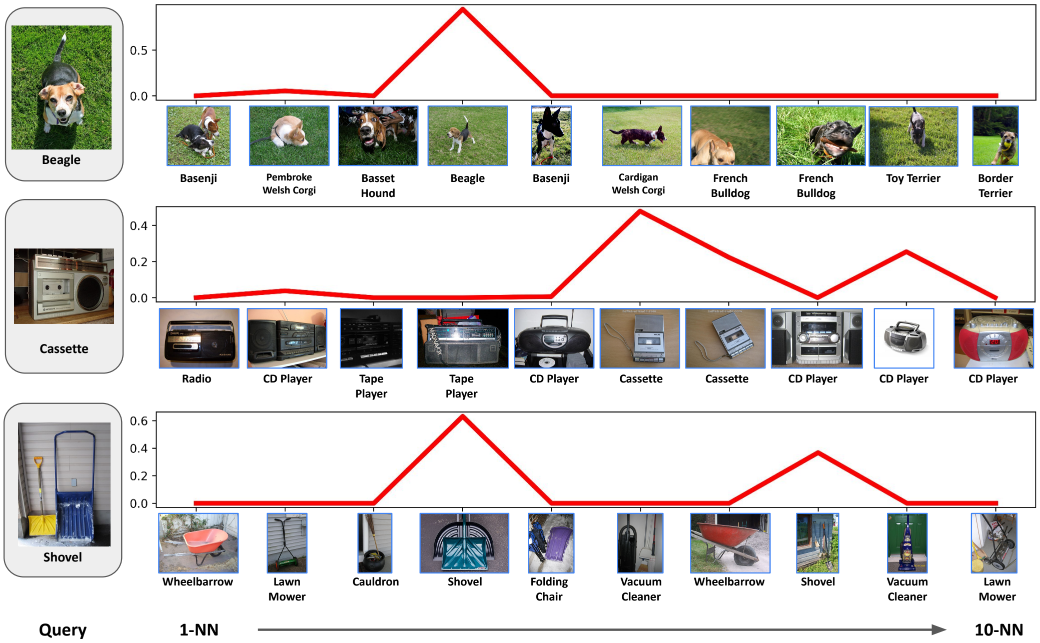

Figure 4. **Qualitative Examples.** We present the output of our method visually. We conduct this experiment by choosing the ImageNet-LT dataset as the query and memory dataset. We display the query images from the test set on the left. Their k-NN from the training set are displayed on the right, and ordered from left to right. We display the attention weight assigned to each k-NN above the corresponding image.

many-shot accuracy on both datasets. Nevertheless, this comes at the expense of a poor performance for low-shot classes. RAC, an existing retrieval augmented classification method, shows a more balanced performance between many, mid and low-shot classes. Our method achieves the highest accuracy on both datasets by obtaining high accuracy across different categories. For example, we do not achieve the highest many-shot nor low-shot accuracy on ImageNet-LT. However, because we do not favor any category of classes above others, we have higher performance across different categories and achieve the highest overall accuracy.

Fine-tuning the visual encoder. In Table 1, we also include a variant of our method which fine-tunes the visual encoder $e(.)$ while learning the memory attention module. We denote this variant by *Ours* + *FT*. The k-NN search is still done with a pre-trained, frozen vision encoder as in previous experiments. We also include a variant of RAC† which follows this setup in Table 1. Our method achieves even further gains of accuracy when fine-tuning the vision encoder along with the memory attention module.

5.5. Qualitative examples

We present some of the qualitative examples in Figure 4. We use ImageNet-LT as both the downstream task and the memory dataset for this task. We display the query images on the left, and the top-10 retrieved nearest neighbors on the right. The retrieved images are ordered such that left-most image is the closest one. Above each retrieved image, we display the learned attention value of our method.

We see that our method assigns higher attention weights to the semantically correct images from the k-NN list. We observe this even if there is only one correct example in the k-NN list (*e.g.* the *beagle* query). When there are multiple relevant images, all relevant examples get higher attention weights (*e.g. cassette* and *shovel* queries). The original rank position does not matter much for our method. For example, one of the relevant retrieved images for the *shovel* query has originally rank eight, but receives the second highest attention weight from our method. Figure 5 in Appendix shows the qualitative examples in Places-LT dataset.

6. Conclusions

We propose a simple, yet effective memory attention module for retrieval augmented classification in this work. Our method learns the importance of each retrieved example, and weights their contributions accordingly. We also present a systematic study of different memory designs, showing the benefit of massive-scale memory datasets up to 1B image-text pairs. The effectiveness of our method is shown by the fact that it achieves state-of-the-art results in long-tailed recognition, learning with noisy labels and fine-grained classification tasks.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022. 1

[2] Soumya Basu, Ankit Singh Rawat, and Manzil Zaheer. Generalization properties of retrieval-based models. *arXiv preprint arXiv:2210.02617*, 2022. 3

[3] Andreas Blattmann, Robin Rombach, Kaan Oktay, Jonas Müller, and Björn Ommer. Semi-parametric neural image synthesis. *arXiv preprint arXiv:2204.11824*, 2022. 3

[4] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *ICML*, 2022. 2, 3, 5

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020. 1

[6] Wenhu Chen, Hexiang Hu, Chitwan Saharia, and William W Cohen. Re-imagen: Retrieval-augmented text-to-image generator. *arXiv preprint arXiv:2209.14491*, 2022. 3

[7] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022. 1, 2, 4, 5

[8] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022. 1

[9] Mark Collier, Basil Mustafa, Efi Kokiopoulou, Rodolphe Jenatton, and Jesse Berent. Correlated input-dependent label noise in large-scale image classification. In *CVPR*, 2021. 7

[10] Jiequan Cui, Zhisheng Zhong, Shu Liu, Bei Yu, and Jiaya Jia. Parametric contrastive learning. In *ICCV*, 2021. 1, 7

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[12] Zongyong Deng, Hao Liu, Yaoxing Wang, Chenyang Wang, Zekuan Yu, and Xuehong Sun. Pml: Progressive margin loss for long-tailed age classification. In *CVPR*, 2021. 1

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 5, 7

[14] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014. 2

[15] Hao Guo and Song Wang. Long-tailed multi-label visual recognition by collaborative training on uniform and re-balanced samplings. In *CVPR*, 2021. 2

[16] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *ICML*, 2020. 5

[17] Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R Scott, and Dinglong Huang. CurriculumNet: Weakly supervised learning from large-scale web images. In *ECCV*, pages 135–150, 2018. 7

[18] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*, 2020. 2

[19] Haibo He and E.A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21, 2009. 1

[20] Youngkyu Hong, Seungju Han, Kwanghee Choi, Seokjun Seo, Beomsu Kim, and Buru Chang. Disentangling label distribution for long-tailed visual recognition. In *CVPR*, 2021. 1

[21] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *CVPR*, 2016. 1

[22] Ahmet Iscen, Thomas Bird, Mathilde Caron, Alireza Fathi, and Cordelia Schmid. A memory transformer network for incremental learning. *arXiv preprint arXiv:2210.04485*, 2022. 3

[23] Ahmet Iscen, Jack Valmadre, Anurag Arnab, and Cordelia Schmid. Learning with neighbor consistency for noisy labels. In *CVPR*, 2022. 7

[24] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *ICLR*, 2020. 2

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 6

[26] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *NeurIPS*, 2020. 2

[27] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*, 2017. 5

[28] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. Large-scale long-tailed recognition in an open world. In *CVPR*, 2019. 5

[29] Alexander Long, Wei Yin, Thalaiyasingam Ajanthan, Vu Nguyen, Pulak Purkait, Ravi Garg, Alan Blair, Chunhua Shen, and Anton van den Hengel. Retrieval augmented classification for long-tail visual recognition. In *CVPR*, 2022. 2, 3, 7

[30] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *ICLR*, 2017. 6

[31] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. In *ICLR*, 2021. 1, 3, 6

[32] Kengo Nakata, Youyang Ng, Daisuke Miyashita, Asuka Maki, Yu-Chieh Lin, and Jun Deguchi. Revisiting a knn-based image classification system with high-capacity storage. *ECCV*, 2022. 3

[33] Yoon-Joo Park and Alexander Tuzhilin. The long tail of recommender systems and how to leverage it. In *RecSys '08*, 2008. 1

[34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 3, 4, 5, 7

[35] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. 1, 4, 5

[36] Sai Rajeswar, Pau Rodriguez, Soumye Singhal, David Vazquez, and Aaron Courville. Multi-label iterated learning for image classification with label ambiguity. *arXiv preprint arXiv:2111.12172*, 2021. 7

[37] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3), 2015. 2

[38] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016. 2

[39] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021. 2, 4

[40] Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. Flava: A foundational language and vision alignment model. In *CVPR*, 2022. 1

[41] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 6

[42] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016. 2, 4

[43] Changyao Tian, Wenhai Wang, Xizhou Zhu, Jifeng Dai, and Yu Qiao. Vl-ltr: Learning class-wise visual-linguistic representation for long-tailed visual recognition. In *ECCV*, 2022. 4, 7

[44] Grant Van Horn, Elijah Cole, Sara Beery, Kimberly Wilber, Serge Belongie, and Oisin Mac Aodha. Benchmarking representation learning for natural world image collections. In *CVPR*, 2021. 5

[45] Shuohang Wang, Yichong Xu, Yuwei Fang, Yang Liu, Siqi Sun, Ruochen Xu, Chenguang Zhu, and Michael Zeng. Training data is more valuable than you think: A simple and effective method by retrieving from training data. *arXiv preprint arXiv:2203.08773*, 2022. 2

[46] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *arXiv preprint arXiv:2208.10442*, 2022. 1

[47] Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella Yu. Long-tailed recognition by routing diverse distribution-aware experts. In *ICLR*, 2021. 1

[48] Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *ICLR*, 2022. 2

[49] Liuyu Xiang, Guiguang Ding, and Jungong Han. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *ECCV*, 2020. 2

[50] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *CoRR*, abs/2205.01917, 2022. 1

[51] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021. 1

[52] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *CVPR*, 2022. 4, 5, 7

[53] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *CVPR*, 2022. 4

[54] Yan Zhao, Weicong Chen, Xu Tan, Kai Huang, Jin Xu, Changhu Wang, and Jihong Zhu. Improving long-tailed classification from instance level. 2021. 1

[55] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. BBN: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *CVPR*, 2020. 2

[56] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Trans. PAMI*, 2017. 5