LexLIP: Lexicon-Bottlenecked Language-Image Pre-Training for Large-Scale Image-Text Sparse Retrieval

Ziyang Luo1, Pu Zhao2, Can Xu2, Xiubo Geng2, Tao Shen2, Chongyang Tao2, Jing Ma1, Qingwei Lin2, Daxin Jiang2†
1 Hong Kong Baptist University, Hong Kong SAR, China
2 Microsoft Corporation

Abstract

Image-text retrieval (ITR) aims to retrieve images or texts that match a query originating from the other modality. The conventional dense retrieval paradigm relies on encoding images and texts into dense representations with dual-stream encoders. However, this approach is limited by slow retrieval speeds in large-scale scenarios. To address this issue, we propose a novel sparse retrieval paradigm for ITR that exploits sparse representations in the vocabulary space for images and texts. This paradigm enables us to leverage bag-of-words models and efficient inverted indexes, significantly reducing retrieval latency. A critical gap emerges from representing continuous image data in a sparse vocabulary space. To bridge this gap, we introduce a novel pre-training framework, Lexicon-Bottlenecked Language-Image Pre-Training (LexLIP), that learns importance-aware lexicon representations. By using lexicon-bottlenecked modules between the dual-stream encoders and weakened text decoders, we are able to construct continuous bag-of-words bottlenecks and learn lexicon-importance distributions. Upon pre-training with same-scale data, our LexLIP achieves state-of-the-art performance on two ITR benchmarks, MSCOCO and Flickr30k. Furthermore, in large-scale retrieval scenarios, LexLIP outperforms CLIP with $5.8\times$ faster retrieval speed and $19.1\times$ less index storage memory. Beyond this, LexLIP surpasses CLIP across 8 out of 10 zero-shot image classification tasks.

1. Introduction

Image-text retrieval (ITR) is a critical problem that involves retrieving relevant images and texts based on textual and visual queries. Its practical applications span multiple domains, including e-commerce product search [27] and social media image search [19]. Due to the lack of publicly available large-scale benchmarks, the evaluation of existing ITR models [20, 32, 39, 44] is commonly conducted on small-scale datasets such as MSCOCO [29] and Flickr30k [38], which contain a limited number of samples in their test sets. Thus, the significance of retrieval speed is frequently disregarded. However, real-world scenarios, such as Google Image Search, may involve a massive number of candidate samples, easily exceeding 1M. Hence, retrieval speed is a crucial concern. The current dense retrieval paradigm, in which each image and text is represented as a dense vector (as shown in Figure 1a), can become computationally expensive, resulting in slow retrieval speed. The large-scale exact k-nearest neighbor (KNN) dense retrieval involves calculating the similarity between the query and all candidate samples, resulting in a linear increase in retrieval time as the number of samples increases [5]. This limitation poses a challenge for the dense retrieval paradigm in real-world applications and underscores the need for more efficient and

---

* Work done during the internship at Microsoft.
** Corresponding author
In this work, we present a novel **sparse retrieval paradigm** for ITR, as illustrated in Figure 1b. This paradigm encodes images and texts as sparse representations in the vocabulary space, wherein the relevant lexicons are assigned high weights, and the others are set to zero. The retrieval process involves transforming these lexicon-weighted representations into inverted indexes, as depicted in Figure 2. Then, we apply the Exact Lexicon Matching Search algorithm [40] to find matching pairs, which only calculates similarity scores with candidates that share common lexicons. This mechanism avoids iterating over all samples and substantially reduces retrieval latency. Moreover, this paradigm leverages lexicon-level contextualization by considering both implicit object expansion [15] and explicit object concurrence [36].

The sparse retrieval paradigm poses a significant challenge for image processing because images are continuous data that need to be projected into a discrete vocabulary space. To bridge this gap, we propose a novel pre-training framework, termed **Lexicon-Bottlenecked Language Image Pre-Training (LexLIP)**, to learn importance-aware lexicon representations. **LexLIP** comprises two pre-training phases: i) lexicon-bottlenecked pre-training and ii) momentum lexicon-contrastive pre-training.

In the first pre-training phase, we introduce two novel objectives: image/text lexicon-bottlenecked masked language modeling. These objectives aim to establish the lexicon-weighting representations as the bottlenecks between the images/texts data and the sparse vocabulary space. Specifically, we pass an image or masked text into a vision or language encoder to derive the lexicon-weighting representations. Meanwhile, we utilize a weakened masking-style text decoder to reconstruct the masked text from these representations. Due to the aggressive masking, the decoder is inclined to recover masked tokens based on the lexicon-weighting representations. As a result, the **LexLIP** encoders assign higher importance scores to crucial vocabulary lexicons of the image or text and lower scores to trivial ones. This aligns well with the goal of the sparse retrieval paradigm and enhances its performance.

The second pre-training phase is momentum lexicon-contrastive learning, where images and texts are further aligned in the sparse vocabulary space with a large-scale negative sample size. The experimental results reveal that our **LexLIP** pre-trained with same-scale image-text pairs achieves state-of-the-art (SOTA) performance on the widely used ITR benchmarks, MSCOCO [29] and Flickr30k [38]. In the large-scale retrieval scenario (i.e., the candidate pool has one million samples), **LexLIP** demonstrates a remarkable improvement in retrieval speed with a 5.8 times faster and a significant reduction in index storage memory with a 19.1 times decrease, compared to CLIP [39]. Beyond this, **LexLIP** surpasses CLIP across 8 out of 10 zero-shot image classification tasks. Our codes are available at [https://github.com/ChiYeungLaw/LexLIP-ICCV23](https://github.com/ChiYeungLaw/LexLIP-ICCV23).

Our contributions can be listed as follows:

1. We introduce the novel **Sparse Retrieval Paradigm** to ITR. By representing images and texts in the lexicon vocabulary space, this approach significantly improves the efficiency of large-scale ITR.

2. We propose a new framework, **Lexicon-Bottlenecked Language Image Pre-Training (LexLIP)**, to learn the lexicon-weighting representations.

3. We conduct extensive experiments on both small-scale and large-scale ITR benchmarks. Pre-trained with same-scale data, our **LexLIP** achieves SOTA performance on small-scale ITR datasets, MSCOCO and Flickr30k. Moreover, our **LexLIP** achieves 5.8× speed-up and 19.1× less index storage memory than CLIP on large-scale ITR. Beyond this, our **LexLIP** outperforms CLIP across 8 out of 10 zero-shot image classification tasks.
2. Related Work

Image-Text Retrieval. ITR has received considerable attention in the cross-modal community. Traditional approaches to ITR utilized Convolutional Neural Networks (CNNs) [3] as encoders to individually encode images and texts [12, 45]. In recent years, the popularity of transformer-based models and large-scale language-image pre-training have seen a surge [20, 26, 28, 32, 33, 34, 39, 44, 51, 52, 53]. These models introduce several methods to enhance ITR performance, including knowledge distillation, data augmentation, multitask learning, and gigantic-scale pre-training. They have achieved remarkable performance on various ITR benchmarks. However, these models rely on the traditional dense retrieval paradigm, which faces the challenge of slow retrieval speed in large-scale scenarios.

Sparse Retrieval Paradigm. This paradigm originates from the BM25 [40] for exact lexicon matching, which utilizes an inverted index to reduce retrieval latency by only considering samples with overlapping lexicons during the retrieval process. This method has recently gained popularity in NLP document retrieval [14, 15, 24, 42, 43, 55], due to its efficiency in handling large-scale text data and the ability to integrate well with neural network-based models. However, while the text data is naturally discrete and can be projected into a vocabulary space, images are continuous and pose a challenge for sparse lexicon representation.

Bottlenecked Pre-training in Retrieval. This method is widely studied in the document retrieval [16, 17, 30, 42, 46]. The masked language modeling objective is conditioned on dense representations. Despite its proven success in NLP, this method has not yet been widely explored in ITR.
this work, we aim to fill this gap by proposing a novel pre-
training method, which leverages sparse lexicon represen-
tations as bottlenecks to enhance the performance.

3. LexLIP

Overview. Adhering to the recent trends in ITR [20, 32,
39, 44], our LexLIP framework employs the dual-stream
encoding structure. Both images and texts are embedded into
distinct sparse lexicon representations. To achieve this, we
introduce two pre-training phases in our framework, namely:
(i) Lexicon-Bottlenecked Pre-training (as shown in Figure 3),
and (ii) Momentum Lexicon-Contrastive Pre-training (as
shown in Figure 4). In the following sections, we will delve
into the encoding, pre-training, and inference details of our
framework.

3.1. Dual-Stream Encoders and Sparse Lexicon
Representations

Following recent works [32, 39], the backbone of the
visual encoder is the Vision Transformer [11], while the
Language Transformer [10] serves as the backbone of the
textual encoder. The input image is first transformed into
a series of flattened 2D patches, which are subsequently
processed by the visual encoder to generate the correspond-
ing hidden states. Formally, given all patches of an image
\( x = [x_1, \ldots, x_m] \), the visual encoder transform them into
fixed-length vectors:

\[
H^v = \text{Trans}^v ([\text{CLS}^v; x]) \in \mathbb{R}^{(n+1) \times d},
\]

where \( \text{Trans}^v \) is the visual encoder and \( d \) is the model size.
In the dense retrieval paradigm, it is a common practice to
utilize the first hidden state of \( H^v \) as the dense representa-
tions of images [32, 33, 39]. Differently, our visual encoder
is followed by a language model head which projects the
hidden states into the sparse vocabulary space:

\[
S_x^{(enc)^v} = \text{LM-Head}^v (H^v) \in \mathbb{R}^{(n+1) \times |V|},
\]

where \( |V| \) is the vocabulary size. We denote \( S_x^{(enc)^v} \) as the
LM logits of images from visual encoder. Then, we follow
the SPLADE model in document retrieval [15] to represent
an image in the high-dimensional vocabulary space by

\[
p^v = \log (1 + \text{MaxPool}(\max (S_x^{(enc)^v}, 0))) \in \mathbb{R}^{|V|},
\]

where \( \max (\cdot, 0) \) ensures all values greater than or equal to
zero for the sparse requirements, \( \text{MaxPool}(\cdot) \) denotes max
pooling along with the sequence axis, and the saturation
function \( \log (1 + \text{MaxPool}(\cdot)) \) prevents some terms from
dominating. \( p^v \) stands for the lexicon-weighting sparse represen-
tation of an image.

Similarly, the language encoder generates the lexicon-
weighting sparse representation of the input text \( y =
[y_1, \ldots, y_n] \) by

\[
H^l = \text{Trans}^l ([\text{CLS}^l; y]) \in \mathbb{R}^{(n+1) \times d},
\]

\[
S_y^{(enc)^l} = \text{LM-Head}^l (H^l) \in \mathbb{R}^{(n+1) \times |V|},
\]

\[
p^l = \log (1 + \text{MaxPool}(\max (S_y^{(enc)^l}, 0))) \in \mathbb{R}^{|V|},
\]

where \( \text{Trans}^l \) is the language encoder, \( S_y^{(enc)^l} \) is the LM logits
of texts from language encoder, \( \text{LM-Head}^l \) is the language
model head for texts, and \( p^l \) is the lexicon-weighting sparse
representation of a text.

3.2. Phase 1: Lexicon-Bottlenecked Pre-training

As shown in Figure 3, this pre-training phase consists of
four different objectives, including self-supervised masked
language modeling, two lexicon-bottlenecked masked lan-
guage modelings and in-batch lexicon-contrastive learning.

Self-Supervised Masked Language Modeling (Self-
MLM). Consistent with the standard practice of pre-training
the language encoder in an unsupervised manner, the masked
language modeling (MLM) objective is utilized for pre-
training our language encoder, \( \text{Trans}^l \). Formally, the tokens
in the input text \( y \) are masked to obtain \( \bar{y} \), with \( \alpha \% \) tokens
being replaced by a special token [MASK] or a random to-
ken in the vocabulary set, \( V \), and the remaining being kept
unchanged. The masked \( \bar{y} \) is then processed by the language
encoder to generate the language model (LM) logits, \( S_y^{(enc)^l} \),
and reconstruct the masked tokens through the following
objective function:

\[
L_{\text{self}} = - \sum_{y_j \in \bar{y}} \sum_{j \in \mathbb{R}^{(enc)}} \log P(w_j = y_j | \bar{y}),
\]

where \( P(w_j) \) is calculated as \( \text{softmax} \left( S_y^{(enc)^l} [j,:], 1 \right) \). \( \mathbb{D} \)
represents the set of all samples, \( \mathbb{M}^{(enc)} \) denotes the set of
masked indices in \( \bar{y} \), \( w_j \) represents the discrete variable over
\( V \) at the j-th position of \( y \), and \( y_j \) refers to its original token.

Lexicon-Bottlenecked Masked Language Modelings
(LexMLM). Regarding to the token-level logits from Eq. 2
and 5 defined in the lexicon vocabulary space, we propose
to calculate the lexicon-importance distributions of images
and masked texts by

\[
a^v = \text{Normalize} \left( \text{MaxPool}(S_x^{(enc)^v}) \right) \in [0, 1]^{|V|},
\]

\[
a^l = \text{Normalize} \left( \text{MaxPool}(S_y^{(enc)^l}) \right) \in [0, 1]^{|V|},
\]

where \( \text{Normalize}(\cdot) = \text{softmax}(\cdot) \) denotes the normal-
ization function (let \( \sum a_i = 1 \)), \( a^v \) denotes lexicon-
importance distribution over \( V \) to indicate the relative import-
ance of the different lexicons in the vocabulary.
To obtain the lexicon-importance distributions, we are inspired by the bottleneck-enhanced dense representation learning strategy from recent works in document retrieval [16, 17, 30, 42]. Our framework introduces a lexicon-bottlenecked module to utilize these distributions as a bridge to guide the reconstruction of masked lexicons, leading the vision and language encoders to focus on the most critical tokens/words in the data. However, directly utilizing the high-dimensional distribution vectors \( d^{(s)} \in [0, 1]^{[v]} \) as the bottlenecks faces challenges. First, the distribution over the entire vocabulary space possesses a capacity to encapsulate the majority of data semantics [50]. Consequently, the efficacy of the bottleneck is reduced. Second, it is difficult to input the high-dimensional vector into a decoder for text reconstruction.

Therefore, we present a novel approach in which we generate continuous bag-of-words (CBoW) representations as the bottlenecks, guided by the lexicon-importance distributions acquired from Equations 8 and 9. That is

\[
b^{(c)} = a^{(s)} \cdot s_g \left( W^{(te)} \right) \in \mathbb{R}^d, \tag{10}\]

where \( W^{(te)} \in \mathbb{R}^{[v] \times d} \) is the token embeddings matrix of the language encoder and \( s_g(\cdot) \) refers to stop gradient. Thereby, \( b^{(c)} \) stands for CBoW bottleneck representations.

To guide the learning of the bottleneck representations \( b^{(c)} \), which in turn leads to the learning of the lexicon-importance distributions \( a^{(s)} \), we use two decoders, one for vision and one for language, to reconstruct the masked text \( \hat{y} \) from \( b^{(c)} \). This approach follows recent advancements in bottleneck-enhanced neural structures [16, 17, 42]. The two decoders, which we refer to as weakened masking-style decoders, are designed to place a heavy reliance on the bottleneck representations by employing two strategies: (i) an aggressive masking strategy, and (ii) using only two shallow transformer layers.

In particular, given the masked text \( \hat{y} \), we adopt an aggressive masking strategy to produce the masked text \( \hat{y} \) with a larger masking rate. This prompts the encoders to compress the rich contextual information into the bottleneck representation, \( b^{(c)} \). Subsequently, the bottleneck representation prefixes \( \hat{y} \) by replacing the [CLS] special token. Therefore, our weakened masking-style decoding can be formulated as

\[
S^{(dec)}_{\hat{y}} = \text{Decoder}^v \left( [b^{(c)}; \hat{y}] \right) \in \mathbb{R}^{(n+1) \times [v]}, \tag{11}\]

\[
S^{(dec)}_{\hat{y}} = \text{Decoder}^l \left( [b^{(c)}; \hat{y}] \right) \in \mathbb{R}^{(n+1) \times [v]}, \tag{12}\]

Similar to the Self-MLM, the loss functions are:

\[
L^{(2t)}_{\text{baco}} = - \sum_D \sum_{j \in M^{(dec)}} \log P^v(w^j = y^j | \hat{y}), \tag{13}\]

\[
L^{(2i)}_{\text{baco}} = - \sum_D \sum_{j \in M^{(dec)}} \log P^l(w^j = y^j | \hat{y}), \tag{14}\]

where \( P^{(i)}(w^j) = \text{softmax} \left( S^{(dec)}_{\hat{y}}(j; : \right) \right) \), and \( M^{(dec)} \) denotes the set of masked tokens in \( \hat{y} \).

**In-Batch Lexicon-Contrastive Learning (BaCo).** Given the lexicon sparse representations from Eqs. 3 and 6, we perform in-batch contrastive learning in this phase to align images and texts in the vocabulary space. The models learn by contrasting the lexicon-weighting sparse representations of different samples within a single batch of data. The loss functions are:

\[
L^{(2t)}_{\text{baco}} = - \sum_D \log \frac{\exp \left( p^v(p^l)^T / \tau \right)}{\sum_{j \in B} \exp \left( p^v(p^l_j)^T / \tau \right)} + \lambda \mathcal{F}(p^v), \tag{15}\]

\[
L^{(2i)}_{\text{baco}} = - \sum_D \log \frac{\exp \left( p^l(p^l)^T / \tau \right)}{\sum_{j \in B} \exp \left( p^l(p^l_j)^T / \tau \right)} + \lambda \mathcal{F}(p^l), \tag{16}\]

where \( B \) denotes all the data in a batch, \( \tau \) is the temperature hyperparameter, \( \mathcal{F}(\cdot) \) is the FLOPS function introduced in SPLADE [15] for representation sparsity, and \( \lambda \) is the regularization hyperparameter. The overall loss function is:

\[
L_{\text{baco}} = \left( L^{(2t)}_{\text{baco}} + L^{(2i)}_{\text{baco}} \right) / 2. \tag{17}\]

**Phase 1 Learning.** The final loss function of the Lexicon-Bottlenecked Pre-training is a direct addition of all losses:

\[
L_{p1} = L_{scl} + L_{i2t} + L_{i2i} + L_{\text{baco}}. \tag{18}\]

**3.3. Phase 2: Momentum Lexicon-Contrastive Pre-training.**

After learning the sparse lexicon representations in the first phase, we further align the representations of images and texts in the vocabulary space. It has been shown that the large-scale negative samples are crucial for achieving good performance in ITR [39]. However, the negative sample size is limited by the mini-batch size in traditional in-batch contrastive learning, which can be constrained by the GPU’s memory. To address this issue, we adopt the momentum contrastive learning in MoCo [18] to cache negative samples with two different queues, \( Q^v \) and \( Q^l \), for images and texts, respectively. This approach decouples the negative sample size from the mini-batch size, making the learning process more computationally feasible.

In accordance with prior works [32, 33], two momentum encoders, \( \theta^v_m \) and \( \theta^l_m \), are employed to update the samples in the queues. These encoders share the same structures and initial parameters as the original encoders, but they have truncated gradients and are updated utilizing the exponential moving average (EMA) mechanism:

\[
\theta^v_m = m \theta^v + (1 - m) \theta^v, \tag{19}\]

\[
\theta^l_m = m \theta^l + (1 - m) \theta^l. \tag{20}\]
where The momentum contrastive loss functions are:

These momentum encoders are dropped after pre-training.

The overall momentum contrastive loss is:

overcome this challenge, we have employed the quantization term-based sparse retrieval systems like Anserini [49]. To
to-end learning. However, it is infeasible for open-source product is necessary for gradient back-propagation and end-
representations to measure the similarity, where the dot-
product between the real-valued sparse lexicon-weighted methods. As in Eq. 15, 16, 21, and 22, we use the dot-
notable differences between the dense and sparse retrieval representations of images and texts are denoted as

is the EMA decay weight. The momentum lexicon sparse

Table 1: Evaluation our LexLIP in the small-scale retrieval scenario after fine-tuning.

where \( \theta_m \) is the parameters of the original encoders and \( m \) is the EMA decay weight. The momentum lexicon sparse representations of images and texts are denoted as \( \tilde{l}^e \) and \( \tilde{p}^f \). These momentum encoders are dropped after pre-training. The momentum contrastive loss functions are:

\[
L_{moco}^{21t} = -\sum_\mathcal{D} \log \frac{\exp \left( (\tilde{p}^f(\tilde{l}^e))^T \right)}{\tau} + \lambda \mathcal{F}(\tilde{p}^f),
\]

\[
L_{moco}^{21i} = -\sum_\mathcal{D} \log \frac{\exp \left( (\tilde{l}^e(\tilde{p}^f))^T \right)}{\tau} + \lambda \mathcal{F}(\tilde{l}^e).
\]

The overall momentum contrastive loss is:

\[
L_{moco} = \frac{L_{moco}^{21t} + L_{moco}^{21i}}{2}.
\]

3.4. Exact Lexicon Search for Large-Scale Retrieval

In the inference phase of large-scale retrieval, there exist notable differences between the dense and sparse retrieval methods. As in Eq. 15, 16, 21, and 22, we use the dot-product between the real-valued sparse lexicon-weighted representations to measure the similarity, where the dot-product is necessary for gradient back-propagation and end-to-end learning. However, it is infeasible for open-source term-based sparse retrieval systems like Anserini [49]. To overcome this challenge, we have employed the quantization method to transform the high-dimensional sparse vectors into the corresponding lexicons and their virtual weights. The lexicons are obtained from the non-zero elements of the high-dimensional sparse vector, while the weights are determined through a simple quantization approach, i.e., \([100 \times p^{l(f)}]\). Given a query and a candidate sample, the exact lexicon matching score is defined as:

\[
\text{score} = \sum_{l \in L^q \cap L^s} W_q(l) \times W_s(l),
\]

where \( L^q \) and \( L^s \) denote the lexicon lists of the query and candidate sample, and \( W_q(l) \) is the weight of the lexicon.

Overall, our framework, LexLIP, comprises the following large-scale retrieval steps: i) converting all candidate samples into high-dimensional sparse representations, and subsequently into lexicons and frequencies (weights); ii) constructing a term-based inverted index using Anserini [49] for the entire sample collection; iii) generating the lexicons and frequencies for a test query similarly; and iv) querying the inverted index to retrieve the relevant samples.

4. Small-Scale Retrieval Experiment

4.1. Setup

Pre-Training and Evaluation. We use two different image-text datasets to pre-train our LexLIP: (1) CC4.3M contains 4.3M image-text pairs from Conceptual Captions 3.3M [41] (about 2.8M urls are valid), SBU [37], MSCOCO [29] training set and Flickr30K [38] training set. (2) CC14.3M consists of CC4.3M and Conceptual Captions 12M [6] (about 10M urls are valid), which contains 14.3M image-text pairs. For the downstream tasks, models are

<table>
<thead>
<tr>
<th>Model</th>
<th>#I-T</th>
<th>Flickr30k Test (1K Images)</th>
<th>MSCOCO Test (5K Images)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>R@1</td>
<td>R@5</td>
</tr>
<tr>
<td>Frozen (ICCV21 [1])</td>
<td>5.5M</td>
<td>61.0</td>
<td>87.5</td>
</tr>
<tr>
<td>LD (NAACL21 [44])</td>
<td>9.5M</td>
<td>69.9</td>
<td>91.1</td>
</tr>
<tr>
<td>COOKIE (ICCV21 [47])</td>
<td>5.9M</td>
<td>68.3</td>
<td>91.1</td>
</tr>
<tr>
<td>VISTA (CVPR22 [7])</td>
<td>9.5M</td>
<td>68.9</td>
<td>91.1</td>
</tr>
<tr>
<td>oCLIP (ICML21 [39])</td>
<td>4.3M</td>
<td>70.6</td>
<td>90.4</td>
</tr>
<tr>
<td>†Dense (ours)</td>
<td>4.3M</td>
<td>74.0</td>
<td>92.8</td>
</tr>
<tr>
<td>†Dense (ours)</td>
<td>4.3M</td>
<td>75.6</td>
<td>93.6</td>
</tr>
<tr>
<td>†Dense (ours)</td>
<td>14.3M</td>
<td>76.5</td>
<td>93.9</td>
</tr>
</tbody>
</table>

Sparse-vector Dual-Stream Retriever

<table>
<thead>
<tr>
<th>Model</th>
<th>#I-T</th>
<th>Flickr30k Test (1K Images)</th>
<th>MSCOCO Test (5K Images)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>R@1</td>
<td>R@5</td>
</tr>
<tr>
<td>COTS (CVPR22 [32])</td>
<td>5.3M</td>
<td>75.2</td>
<td>93.6</td>
</tr>
<tr>
<td>COTS (CVPR22 [32])</td>
<td>15.3M</td>
<td>76.5</td>
<td>93.9</td>
</tr>
</tbody>
</table>

* #I-T corresponds to the number of image-text pairs during pre-training.

‡ Represent data with the dense CLS representations and conduct the similar pre-training process as LexLIP.

† Re-implement the CLIP model with the same-scale pre-training as our models.

‡ The state-of-the-art dense-vector dual-stream retriever, pre-trained with the same-scale image-text data.

11211
### Table 2: Evaluating our LexLIP in the large-scale retrieval scenario. “Index Size” corresponds to the storage requirement to embed all candidate images. “Repr Byte” denotes the storage requirement for an embedded image. Each activated (non-zero) term in a lexicon-weighed sparse vector needs 3 bytes (2 bytes for indexing and 1 byte for its weight). “QPS” corresponds to query-per-second (the higher, the faster). “Time” denotes the average time for a query to reach the retrieval result (the lower, the faster). Both “QPS” and “Time” measure the retrieval latency.

<table>
<thead>
<tr>
<th>Model</th>
<th>Index Size</th>
<th>Repr Byte</th>
<th>QPS</th>
<th>Time</th>
<th>Large-Scale Flickr30k Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T2I Retrieval</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R@1</td>
</tr>
<tr>
<td>BM25 [40]</td>
<td>195M</td>
<td>Avg 195</td>
<td>780.18</td>
<td>1.28ms</td>
<td>16.8</td>
</tr>
<tr>
<td>Cross-Modal Dense Dual-Stream Retriever</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLIP [39]</td>
<td>2.9G</td>
<td>Avg 2897</td>
<td>3.42</td>
<td>292.40ms</td>
<td>45.8</td>
</tr>
<tr>
<td>Dense (ours)</td>
<td>2.9G</td>
<td>Avg 2897</td>
<td>3.55</td>
<td>281.69ms</td>
<td>47.7</td>
</tr>
<tr>
<td>Our Cross-Modal Sparse Dual-Stream Retriever</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LexLIP (ours)</td>
<td>152M</td>
<td>Avg 152</td>
<td>19.70</td>
<td>50.76ms</td>
<td>48.6</td>
</tr>
<tr>
<td>-top64 sparsify</td>
<td>114M</td>
<td>Upto 192</td>
<td>63.43</td>
<td>15.77ms</td>
<td>67.6</td>
</tr>
<tr>
<td>-top32 sparsify</td>
<td>71M</td>
<td>Upto 96</td>
<td>4.02ms</td>
<td>2.55ms</td>
<td>42.0</td>
</tr>
<tr>
<td>-top16 sparsify</td>
<td>47M</td>
<td>Upto 48</td>
<td>610.08</td>
<td>1.64ms</td>
<td>28.3</td>
</tr>
<tr>
<td>-top12 sparsify</td>
<td>41M</td>
<td>Upto 36</td>
<td>796.65</td>
<td>1.26ms</td>
<td>22.3</td>
</tr>
<tr>
<td>-top8 sparsify</td>
<td>34M</td>
<td>Upto 24</td>
<td>985.40</td>
<td>1.01ms</td>
<td>14.6</td>
</tr>
</tbody>
</table>

### Table 3: Accelerate CLIP with Approximate Nearest Neighbor Searching (ANN). The T2I retrieval scores are reported on the large-scale Flickr30k test set. Though the retrieval speed can be increased, the recall decreases a lot.

<table>
<thead>
<tr>
<th>Model</th>
<th>Time</th>
<th>R@1</th>
<th>R@5</th>
<th>R@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP (w/o accelerate)</td>
<td>292.40ms</td>
<td>45.8</td>
<td>64.1</td>
<td>69.5</td>
</tr>
<tr>
<td>LexLIP (w/o accelerate)</td>
<td>50.76ms</td>
<td>48.6</td>
<td>66.8</td>
<td>71.8</td>
</tr>
<tr>
<td>CLIP (ANN accelerate)</td>
<td>4.65ms</td>
<td>25.9</td>
<td>33.7</td>
<td>35.8</td>
</tr>
<tr>
<td>LexLIP (top16 sparsify)</td>
<td>1.64ms</td>
<td>28.3</td>
<td>44.2</td>
<td>50.6</td>
</tr>
</tbody>
</table>

### Table 4: Zero-shot image classification.

evaluated on MSCOCO and Flickr30k test sets with fine-tuning. Each image in these datasets is accompanied by 5 different captions. We follow the Karpathy split [22] to divide the datasets into train/val/test sets, with 113.2k/5k/5k (MSCOCO) and 29.8k/1k/1k (Flickr30k) images. For evaluation, we use the standard R@k (k=1,5,10) to calculate the retrieval scores of our models, the same as previous works [32, 39, 44, 47].

### Implementation Details.
For computational efficiency, we follow [32] to initialize the dual-stream encoders with the pre-trained vision [2] and language transformer [10], whereas the other parts are randomly initialized. Both of them are the base-size, 12-layer transformer encoders with 768 hidden size. The pre-trained input image resolution is 224 × 224. The fine-tuning resolution is 384 × 384. Models are pre-trained with 20 epochs in the first phase, 10 epochs in the second phase, and fine-tuned with 10 epochs. The AdamW optimization algorithm [31] with a learning rate of 5e-5, linear learning rate decay and 10% warm-up steps, and mixed-precision training are employed. The masking rate for the language encoder is set to 30% and 50% for the decoder, with an EMA weight of 0.99 and a temperature τ of 0.05. The regularization term λ is set to 0.002. Further details can be found in the supplementary material section A.1.

### 4.2. Results
As shown in Table 1, under a fair comparison (excluding the models pre-trained with billions of image-text pairs), our LexLIP achieves the SOTA performance over all previous
works for most evaluation metrics. Specifically, in comparison to the previous SOTA COTS [32], LexLIP obtains higher results by 1.5% (76.7% vs. 75.2%) for T2I R@1 and 1.4% (89.6% vs. 88.2%) for I2T R@1 on Flickr30k, while utilizing less pre-training data (4.3M vs. 5.3M). Furthermore, with a larger pre-training dataset, LexLIP further enhances performance by 1.9% (78.4% vs. 76.5%) for T2I R@1 and 0.8% (91.4% vs. 90.6%) for I2T R@1 on Flickr30k, while still utilizing less data (14.3M vs. 15.3M).

5. Large-Scale Retrieval Experiment

5.1. Setup

Baselines and Large-Scale Benchmark. In addition to our re-implemented CLIP model and Dense model, we have incorporated another baseline, the single-modal sparse text retriever BM25 [40]. By leveraging captions to represent images, BM25 can perform image-text retrieval as caption-text retrieval. For large-scale ITR, we expand the test set of Flickr30k [38] by including 1M randomly selected image-text pairs from Conceptual Caption 12M [6]. Each image in Flickr30k is associated with 5 captions, from which we randomly select one as the alternative image representation for BM25 retrieval. In text-to-image retrieval, models retrieve images from the 1k images of Flickr30k and an additional 1M images, given 4k captions as queries from Flickr30k. Conversely, for image-to-text retrieval, models retrieve captions from the 4k captions of Flickr30k and an additional 1M captions, given 1k images as queries from Flickr30k.

Pre-training and Retrieval. To compare with BM25 in the zero-shot settings, we exclude the Flickr30k training set from the CC4.3M and result in a new dataset CC4.2M for pre-training. For all models, we first embed them into the index file and then retrieve the results on the CPU. The dense retrieval is conducted with the efficient dense vector similarity search library, Faiss [21]. The sparse retrieval is conducted with the efficient sparse vector similarity search library, Anserini [49]. More details can be found in the supplemental material section A.2.

5.2. Results

Figure 5 compares the retrieval time per query between CLIP and our LexLIP, highlighting the inefficiencies of the dense retriever as the number of samples in the candidate pool increases. With a candidate pool of 1M samples, Table 2 shows that the retrieval speed of CLIP is 292.40ms per query. In contrast, our sparse retriever, LexLIP, demonstrates a substantial improvement, with a 5.8 times faster retrieval speed (50.76ms vs. 292.40ms) and 19.1 times less storage memory (152M vs. 2.9G). Moreover, our LexLIP also exhibits superior performance on the benchmark.

To further analyze our LexLIP efficacy-efficiency trade-off, we adopt a simple yet effective sparsification method by retaining only the top-K weighted terms in the representation of the sample during the inference phase and constructing the inverted index using the sparsified samples. As shown in Table 2, our LexLIP achieves the best efficacy-efficiency trade-off among all baselines. With top-12 sparsity, LexLIP has the 4.8 times smaller index size, faster speed, and better retrieval performance than the sparse text retriever BM25.

Furthermore, we employ the widely utilized technique of Approximate Nearest Neighbor (ANN) Searching to enhance the retrieval efficiency of the dense retriever, CLIP. The results presented in Table 3 demonstrate that while the retrieval speed of CLIP indeed experiences a significant increase, it is accompanied by a notable reduction in recall. Our LexLIP with top-16 sparsity has around 2.8 times faster speed, and better retrieval performance than CLIP with ANN acceleration.

6. Zero-shot Image Classification Experiment

6.1. setup

In this study, we undertake a zero-shot image classification experiment utilizing the same models, LexLIP and CLIP, as section 5. The experiment adopts 10 image classification tasks, including CIFAR10, CIFAR100 [23], Caltech101 [13], Food101 [4], SUN397 [48], DTD [8], Pets [54], Flowers [35], MNIST [25], and ImageNet1K [9]. All evaluation settings are the same as the paper of CLIP [39].

6.2. Results

Table 4 provides a comparative analysis of the zero-shot image classification performance between CLIP and our LexLIP. It is noteworthy that LexLIP exhibits superior performance over CLIP across 8 out of 10 tasks. This noteworthy outcome underscores the efficacy of our model, demonstrating its prowess not solely in retrieval tasks but also in the domain of image classification tasks.

7. Further Analysis

Ablation Study. In Table 5, we present the impact of different pre-training objectives and phases of our LexLIP. A pre-training dataset was constructed by randomly sampling 1.4M image-text pairs from Conceptual Captions 3.3M [41] and including all pairs from the Flickr30k [38] training set. The retrieval performance was evaluated on the Flickr30k test set. The results indicate that all pre-training objectives and phases contribute positively to the retrieval performance. The greatest effects were observed for the in-batch contrastive learning in Phase 1 and the momentum contrastive learning in Phase 2, which may be attributed to the alignment of these objectives with the retrieval target. The MLM-based
objectives were also found to be beneficial for learning the lexicon-importance distributions.

**Lexicon-Weighting Examples.** In Figure 6, we visualize the lexicons of 4 images and their captions. If the lexicon has a high weight, the size is large in the lexicon cloud. We can find that the major features of the images and texts are successfully captured by the lexicons. For example, “hat” is an important lexicon in the first image and caption. More examples are in the supplemental material section B.

8. Conclusion

In this study, we present the novel Sparse Retrieval Paradigm for ITR. To overcome the challenge of projecting continuous image data onto the discrete vocabulary space, we introduce the innovative Lexicon-Bottlenecked Language-Image Pre-training (LexLIP) framework. Our experiments demonstrate that LexLIP outperforms the SOTA models on small-scale retrieval when pre-trained with the same-scale data. Furthermore, in large-scale retrieval, LexLIP achieves a substantial improvement in both retrieval speed (5.8× faster) and index storage requirements (19.1× less) compared to the traditional dense retrieval paradigm. Beyond this, LexLIP outperforms CLIP across 8 out of 10 zero-shot image classification tasks.

9. Acknowledgments

This work is partially supported by National Natural Science Foundation of China Young Scientists Fund (No. 62206233) and Hong Kong RGC ECS (No. 22200722). We would like to thank the anonymous reviewers for their excellent feedbacks.

References


