# Bi-Encoder Cascades for Efficient Image Search

Robert Hönig[*] robhoenig@gmail.com
Jan Ackermann[*] ackermannj@ethz.ch
Mingyuan Chi minyuang@ethz.ch

Data Analytics Lab, ETH Zurich

## Abstract

*Modern neural encoders offer unprecedented text-image retrieval (TIR) accuracy, but their high computational cost impedes an adoption to large-scale image searches. To lower this cost, model cascades use an expensive encoder to refine the ranking of a cheap encoder. However, existing cascading algorithms focus on cross-encoders, which jointly process text-image pairs, but do not consider cascades of bi-encoders, which separately process texts and images. We introduce the small-world search scenario as a realistic setting where bi-encoder cascades can reduce costs. We then propose a cascading algorithm that leverages the small-world search scenario to reduce lifetime image encoding costs of a TIR system. Our experiments show cost reductions by up to 6x.*

## 1. Introduction

Search engines are the most widely used tool for information retrieval (IR) on the internet — Google alone processes over 8.5 billion searches a day [24]. A search engine takes as input some query $q$ and returns a list of documents $\mathcal{D}$ ranked by how well they match $q$. Keyword-based search ranks results by naively matching query keywords with documents. Semantic search improves keyword-based search by matching queries to documents based on their meaning. A fruitful domain for semantic search is text-image retrieval (TIR), where documents are images and queries are texts. New semantic search engines for TIR leverage recent advances in deep learning for processing images and natural language [31, 28, 11, 23]. Often, these engines use an image encoder $I$ to embed each image $d \in \mathcal{D}$ into a vector $\boldsymbol{v}_d = I(d)$ and a text encoder $T$ to embed a textual query $q$ into an vector $\boldsymbol{v}_q = T(q)$. Then, the engines rank the set of all images $\mathcal{D}$ by some similarity measure $s(\boldsymbol{v}_q, \boldsymbol{v}_d)$.

Encoders for TIR can be divided into bi-encoders (BEs) [26, 33, 6] and cross-encoders (CEs) [34, 32, 16]. BEs process images and texts with separate encoders $I$ and $T$, whereas CEs also add cross-connections between $I$ and $T$.

[*]Equal contribution.

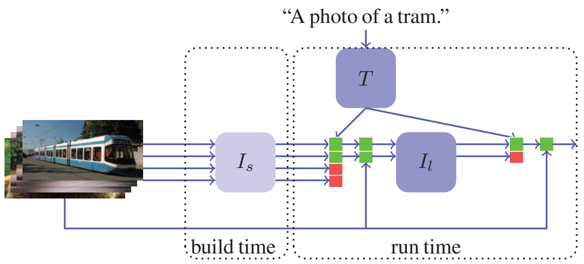

Figure 1. Schematic of our algorithm for a 2-level cascade $[I_s, I_l]$. In this example, encoder $I_s$ computes embeddings (leftmost four squares) $\boldsymbol{V}_s^{\mathcal{D}}$ of all $|\mathcal{D}| = 4$ images at build time. At runtime, $T$ embeds the query into vector $\boldsymbol{v}_q$ to rank embeddings $\boldsymbol{v}_d \in \boldsymbol{V}_s^{\mathcal{D}}$ by the similarity measure $s(\boldsymbol{v}_q, \boldsymbol{v}_d)$. The images with the $m = 2$ highest-ranking embeddings (green) are processed by encoder $I_l$ that produces embeddings $\boldsymbol{V}_l^{\mathcal{D}_m}$ of higher quality. Finally, we rerank the top-2 images with $\boldsymbol{V}_l^{\mathcal{D}_m}$ to output the $k = 1$ highest-ranking images.

Hence, CEs are more powerful but must recompute all image embeddings for new queries, making them impractical for large-scale searches. Thus, we focus on BEs.

A BE TIR system incurs computational costs at *build time* and at *runtime*. At *build time*, the system embeds all images in $\mathcal{D}$. This only happens once. Afterward, at *runtime*, the system repeatedly receives queries $q$, computes their text embedding $\boldsymbol{v}_q$, and ranks the images in $\mathcal{D}$ against $\boldsymbol{v}_q$. During runtime, the system may receive many queries $q_1, \ldots, q_n$. We consider the *asymptotic runtime costs* when $n \to \infty$. We introduce the *lifetime costs* as the sum of build time and asymptotic runtime costs. Lifetime costs are a natural metric to optimize but are usually not explicitly articulated. Instead, existing literature indirectly optimizes lifetime costs by reducing the cost of encoders or individual queries.

In this work, we seek to reduce the image encoding lifetime costs of TIR systems while preserving search quality. To this end, we measure search quality as Recall@$k$, which denotes the fraction of searches that include the desired result in the top-$k$ results. Even small increases in $k$ can significantly improve the Recall@$k$ [33, 26]. Hence, for $m \gg k$, the top-$k$ results of a large and expensive encoder $I_{\text{large}}$ are likely included in the top-$m$ results of a small and cheap encoder $I_{\text{small}}$. This observation leads to our main idea: *At build time, use the small image encoder $I_{small}$ to pre-compute image embeddings for all images and store them in a cache $\boldsymbol{V}^{\mathcal{D}}_{small} = \{d \in \mathcal{D} \mapsto I_{small}(d)\}$. Then, at runtime, to handle a query $q$, first retrieve the top-$m$ results $\mathcal{D}_m \subset \mathcal{D}$ for some $m \gg k$, then compute $\boldsymbol{V}^{\mathcal{D}_m}_{large}$ with $I_{large}$ and return the top-$k$ results. Last, add $\boldsymbol{V}^{\mathcal{D}_m}_{large}$ to the cache for future queries.* Figure 1 illustrates this idea.

How does this approach reduce image encoding lifetime costs? A naive TIR system computes $|\mathcal{D}|$ image embeddings with $I_{\text{large}}$ at build time. It incurs no additional image encoding costs at runtime, so it has image encoding lifetime costs of $|\mathcal{D}|\ c_{\text{large}}$ where $c_{\text{large}}$ is the cost of executing image encoder $I_{\text{large}}$. Our approach computes $|\mathcal{D}|$ image embeddings with $I_{\text{small}}$ at build time. At runtime, we retrieve the top-$m$ images for query $q_n$ as the set $\mathcal{D}^n_m$ and run $I_{\text{large}}$ on all images not in the cache, that is, on $\mathcal{D}^n_m \setminus \bigcup_{1 \le i < n} \mathcal{D}^i_m$. Hence, after $n$ queries we have run $I_{\text{large}}$ for $\left|\bigcup_{1 \le i \le n} \mathcal{D}^i_m\right|$ times, so the asymptotic runtime costs of image encoding are $|\mathcal{D}^\infty_m| c_{\text{large}}$ with $\mathcal{D}^\infty_m = \bigcup_{i \in \mathbb{N}} \mathcal{D}^i_m$. $|\mathcal{D}^\infty_m|$ is bounded by $|\mathcal{D}|$. In practice, it is possible for over 90% of all documents in $\mathcal{D}$ to never be included in any search result over the lifetime of a large-scale search engine [29]. Formally, we have $|\mathcal{D}^\infty_m| < p|\mathcal{D}|$ with $p = 10\%$. We call this the *$p$-small-world search scenario*. In a 10%-small-world search scenario, our approach evaluates $I_{\text{large}}$ on less than 10% of $\mathcal{D}$ at runtime, so the lifetime image encoding costs of our approach is bounded by $0.1|\mathcal{D}|c_{\text{large}} + |\mathcal{D}|c_{\text{small}}$. Thus, our approach saves costs if $0.1|\mathcal{D}|c_{\text{large}} + |\mathcal{D}|c_{\text{small}} < |\mathcal{D}|c_{\text{large}}$, that is if $c_{\text{small}} < 0.9 \cdot c_{\text{large}}$. We ignore the additional costs of reranking the top-$m$ results because they are insignificant compared to the image encoding costs of the neural encoders used in modern TIR systems.

While our approach reduces image encoding lifetime costs, it increases image encoding costs for the first few queries when the cache for $I_{\text{large}}$ embeddings is still mostly empty. In other words, our approach increases early query latency. We construct deep cascades with more than two image encoders and show that they mitigate increases in early query latency.

In this work, we make the following contributions:

- We introduce and formalize the small-world search scenario as an the interesting setting for optimizing TIR.
- We introduce a cascading algorithm for fast TIR with bi-encoders in a small-world search scenario.
- We show that our algorithm reduces TIR image encoding costs on standard benchmarks by up to 6x at no reduction in search quality.
- We investigate the benefits of deep cascades and demonstrate a reduction in early query latency between 1.75x and 2x.

## 2. Related Work

Practitioners may employ a variety of techniques to reduce the lifetime costs of a TIR system. When images are embedded with classical, fast computer vision algorithms, the ranking with similarity measure $s$ may bottleneck the system. A possible remedy is approximate ranking, for example by compressing the image embeddings[12, 36].

For neural TIR systems, the encoder cost greatly outweighs the ranking cost. Then, the most straightforward cost reduction targets model inference, for example via low-bit precision inference [3], teacher-student distillation into a smaller model [8], or pruning of unimportant weights [15]. The prevalence of transformer-based architectures for TIR encoders suggests specializations of these techniques, like pruning transformer heads [14, 22] or masking transformer inputs [19]. The faster models created by any of these techniques can be leveraged by model cascades, which we discuss next.

**Model Cascades** Model cascading is a recurrent theme in the literature on efficient machine learning (ML) systems. FrugalML [2] minimizes access costs of ML APIs by cascading two calls to a cheap API and an expensive API. NoScope [13] speeds up video object detection by splitting a reference model into a sequence of two specialized models. Model cascades have also been applied to facial key point estimation [17] and pedestrian detection [1].

Several methods for fast TIR with CEs have been developed: VLDeformer [35] trains a decomposable CE that can be used as a BE at inference time with minimal loss in quality. CrispSearch [9], LightningDot [30] and "Retrieve Fast, Rerank Smart" [7] all introduce two-level sequences of a BE whose results can be cached for approximate inference and a CE for precise inference on a subset of the BE results. This is superficially similar to our idea, but differs in key ways: Most importantly, our cascades of BEs reduce costs compared to a single BE because they run the most expensive image encoder only on a fraction $p$ of the image corpus $\mathcal{D}$. This requires the $p$-small-world search scenario. In contrast, a cascade of a BE followed by a CE reduces costs compared to a single CE because the BE image embeddings can be pre-computed, enabling a cheap initial image ranking. Then, the CE only needs to rerank the top-$k$ images. Note that the small-world search scenario is not beneficial to BE-CE cascades because the CE needs to rerun on every new image-query pair. Conversely, cascades of BEs can only reduce computational costs with a small-world search scenario because they would eventually evaluate each BE on every image. Finally, note that BE-CE cascades are limited to 2-level cascades, whereas our approach can benefit from deeper cascades. However, our approach can complement BE-CE cascades by replacing the BE with a cascade of BEs.

## 3. Models and Methods

**Cascaded Search** Let $\mathcal{D}$ be a set of images that we want to query with a cascade of BEs. We now generalize our introductory 2-level cascade [$I_{\text{small}}$, $I_{\text{large}}$] to an arbitrarily deep cascade of image encoders $[I_{\text{small}}, I_1, ..., I_r]$ that all use the same text encoder $T$ and that increase in computational cost, that is $c_{\text{small}} < c_1 < ... < c_r$. We propose Algorithm 1 to query $\mathcal{D}$ by ranking all images with $I_{\text{small}}$ and subsequently the top $m_j$ images with $I_j$. With $r = 0$, Algorithm 1 reduces to a standard uncascaded BE search. With $r = 1$, Algorithm 1 reduces to $[I_{\text{small}}, I_{\text{large}}]$.

**Lifetime costs** Let us first restate the $p$-small-world search scenario with respect to Algorithm 1.

**Assumption 1** ($p$-small-world search scenario)**.** *Let $\mathcal{D}^i_{m_1}$ be the set of the top-$m_1$ images pushed to* Top *in Line 4 of Algorithm 1 when handling query $q_i$. Then,*

$$\left| \bigcup_{i \in \mathbb{N}} \mathcal{D}^i_{m_1} \right| < p|\mathcal{D}|.$$

Under Assumption 1, the $r+1$-level cascade $[I_{\text{small}}, I_1, ..., I_r]$ has image encoding lifetime costs bounded by $C_{r+1\text{-level}} := |\mathcal{D}|\, c_{\text{small}} + p\,|\mathcal{D}| \sum_{j=1}^{r} c_j$. The standard BE search uses only the largest image encoder $I_r$, so it has costs $C_{1\text{-level}} = |\mathcal{D}| c_{\text{r}}$. Hence, the $r+1$-level cascade reduces costs by the factor

$$F_{\text{life}} := C_{1\text{-level}} / C_{r+1\text{-level}} = c_r \Big/ \left( c_{\text{small}} + p \sum_{j=1}^{r} c_j \right) \tag{1}$$

It is clear that a 2-level cascade always achieves a greater cost reduction than a cascade with more than two levels because the denominator of $F_{\text{life}}$ grows with $r$.

Uncascaded models encode all images at build time, so they incur no additional image encoding costs at runtime. In contrast, cascades incur image encoding costs at runtime on cache misses. Cache misses occur most frequently for early queries. This means that all cascades experience additional latency for early queries, even though they have lower lifetime costs. We now show that deep cascades may reduce early query latency compared to 2-level cascades.

**Early query latency** Consider the 2-level cascade $[I_{\text{small}}, I_{\text{large}}]$. Algorithm 1 repeatedly processes user queries in Line 14. For early queries (low $i$), we can assume that the image embedding cache $\boldsymbol{V}_{\text{large}}$ is empty when accessed in Line 6. Then, Query($q_i$) needs to compute all $m_{\text{large}}$ required image embeddings, so an early query incurs image encoding cost $c_{\text{large}} m_{\text{large}}$. We refer to this cost as the *early query latency*. We can create a deep cascade by inserting additional image encoders $I_1, ..., I_{r-1}$ between $I_{\text{small}}$ and $I_{\text{large}}$. Concretely, consider the deep cascade $[I_{\text{small}}, I_1, ..., I_r = I_{\text{large}}]$. We choose $m_r < ... < m_2 < m_1 = m_{\text{large}}$. Hence, we use the largest encoder $I_r = I_{\text{large}}$ only on $m_r$ images, less than

**Algorithm 1** Cascaded Search. Here, $\text{Rank}(\mathcal{I}, \boldsymbol{V}, \boldsymbol{v}_q)$ sorts the images in $\mathcal{I}$ by the similarity $s(\boldsymbol{v}_d, \boldsymbol{v}_q)$ of their image encodings $\boldsymbol{v}_d \in \boldsymbol{V}$ with text encoding $\boldsymbol{v}_q$.

```
1:  Input: $[I_{\text{small}}, I_1, ..., I_r]$, $m_1 > ... > m_r \in \mathbb{N}$, $\mathcal{D}$, $k$
2:  Build time: for $d \in \mathcal{D}$ do $\boldsymbol{V}^{\mathcal{D}}_{\text{small}}[d] \longleftarrow I_{\text{small}}(d)$
3:  Function Query(text)
4:  Top $\longleftarrow$ Rank$\big(\mathcal{D}, \boldsymbol{V}^{\mathcal{D}}_{\text{small}}, T(\text{text})\big)$
5:  for $j = 1$ to $r$ do
6:      for $d \in \text{Top}[1...m_j]$ do $\boldsymbol{V}_j[d] \xleftarrow{\text{if empty}} I_j(d)$
7:      Top $\longleftarrow$ Rank(Top$[1...m_j]$, $\boldsymbol{V}_j$, $T$(text))
8:  end for
9:  return Top$[1...k]$
10: EndFunction
11: Runtime:
12: for $i = 1$ to $\infty$ do
13:     $q_i \longleftarrow$ GetUserQuery()
14:     print Query($q_i$)
15: end for
```

the $m_{\text{large}}$ uses in the 2-level encoder. This allows the deep cascade to lower early query latency. At the same time, $m_1 = m_{\text{large}}$ helps ensure that search quality does not degrade.

By how much is early query latency reduced? Similarly to the 2-level cascade, we can assume that for early queries, the image embedding caches $\{\boldsymbol{V}_j\}_{j=1}^{r}$ are empty when accessed in Line 6, so Query($q_i$) incurs image encoding costs $\sum_{j=1}^{r} c_j m_j$. Hence, the deep cascade lowers early query latency by a factor of

$$F_{\text{latency}} := m_{\text{large}} c_{\text{large}} \Big/ \sum_{j=1}^{r} c_j m_j \tag{2}$$

We reduce early query latency ($F_{\text{latency}} > 1$) if the cost of the newly inserted encoders, $\sum_{j=1}^{r-1} c_j m_j$, is lower than the savings from fewer uses of the large encoder, $c_{\text{large}}(m_{\text{large}} - m_r)$.

## 4. Experiments

**Experimental Setup** Given a dataset of image-caption pairs, we measure the Recall@$k$ (R@$k$) metric as the fraction of captions whose corresponding image is among the top-$k$ search results. In line with the IR literature, we report the Recall@$k$ for $k \in \{1, 5, 10\}$ on the Flickr30k [25] test set with 1k samples and the MSCOCO [20] validation set with 5k. In addition, we report the lifetime cost reduction for 2-level cascades and the early query latency reduction for 3-level cascades. Our lifetime cost computation assumes a $p$-small-world search scenario with $p = 0.1$ and $m_1 = 50$. We measure image encoding costs as the number of Multiply-Accumulate (MAC) operations reported by PyTorch-OpCounter [37].

We demonstrate our cascades on the popular and competitive OpenCLIP [10] and BLIP [18] text-image BEs. Both models match images to texts by the cosine similarity of their embeddings. OpenCLIP uses the GPT-2 architecture [27] for the text encoder and offers a broad spectrum of image encoders with convolutional (ConvNeXt [21]) and vision transformer (ViT [5]) architectures. BLIP uses the BERT [4] text encoder and offers a base (BLIP-B) and a large (BLIP-L) large vision transformer as image encoders. To ensure fair comparisons, we only create cascades of models with strictly increasing Recall@k *and* computational costs.

**Results** Table 1 shows our results. For Flickr30k, the uncascaded ConvNeXt models B and L have up to 9.9x lower lifetime costs than the larger ConvNeXt-XXL. However, this comes at a price of significantly lower search quality — R@k drops by up to 5.8%. In contrast, the 2-level cascade [L, XXL] lowers lifetime costs by 3.1x at no loss in search quality. [B, XXL] improves this to 5.0x.

As noted in Section 3, cascades with more than two levels offer no reduced lifetime costs but may reduce early query latency. To demonstrate this, we "sandwich" the medium-cost ConvNeXT-L between the cheap ConvNeXT-B and the expensive ConvNeXT-XXL to obtain the 3-level cascade [B, L, XXL]. Algorithm 1 requires parameters $m_1$ and $m_2$ for a 3-level cascade. We keep $m_1 = 50$ to fairly compare against the 2-level cascade [B, XXL]. $m_2$ is a knob that yields larger savings for smaller values at a potential loss in search quality. We use Equation (2) to compute $m_2 = 14$ as the value for which $F_{\text{latency}} \approx 2$ and use this $m_2$ for all 3-level cascades. Table 1 shows that [B, L, XXL] indeed lowers early query latency by a factor of 1.97x and no significant change in search quality at slightly higher lifetime costs.

Results for ViT models and MSCOCO are similar and match the above analysis.

## 5. Conclusion

We have introduced model cascades that reduce the lifetime computational search costs of BE TIR systems under the $p$-small-world search scenario. Our experiments show that Algorithm 1 can lower costs by over 6x at no reduction in search quality. At the same time, deeper model cascades can mitigate the increase in latency of early queries, which is especially important for expensive modern neural encoders. Anecdotal evidence supports our choice of $p = 10\%$. With that said, practical search scenarios likely vary in $p$ and achieve accordingly different speedups.

Single-digit speedups may not sufficiently reduce computational costs to economically rank large-scale image databases with expensive neural BEs. Instead, a practitioner may use traditional search engines to retrieve the top-$k$ images and apply a neural search cascade on top of it. This heterogeneous cascade may offer a viable path towards integrating state-of-the-art neural networks with established image search platforms.

Flickr30k

| Cascade | R@1 | R@5 | R@10 | $F_{\text{life}}$ ($F_{\text{latency}}$) |
|---|---|---|---|---|
| | | CLIP, ViT | | |
| [g/14] | 75.6 | 92.6 | 95.8 | 1x |
| [B/16] | -6.5 | -3.8 | -1.6 | 15.8x |
| [L/14] | -4.0 | -1.7 | -0.8 | 3.4x |
| [L/14, g/14] | 0.0 | 0.0 | 0.0 | 2.6x |
| [B/16, g/14] | 0.0 | -0.1 | -0.2 | 6.1x |
| [B/16, L/14, g/14] | 0.0 | -0.4 | +0.1 | 5.2x (1.75x) |
| | | CLIP, ConvNeXt | | |
| [XXL] | 75.0 | 93.8 | 96.4 | 1x |
| [B] | -5.8 | -4.8 | -2.5 | 9.9x |
| [L] | -1.2 | -1.6 | -0.3 | 4.4x |
| [L, XXL] | 0.0 | -0.4 | +0.4 | 3.1x |
| [B, XXL] | +1.4 | -0.1 | -0.1 | 5.0x |
| [B, L, XXL] | +0.4 | -0.3 | +0.3 | 4.5x (1.97x) |

MSCOCO

| Cascade | R@1 | R@5 | R@10 | $F_{\text{life}}$ ($F_{\text{latency}}$) |
|---|---|---|---|---|
| | | CLIP, ViT | | |
| [g/14] | 46.3 | 70.68 | 79.84 | 1x |
| [B/16] | -6.2 | -5.3 | -4.5 | 15.8x |
| [L/14] | -2.9 | -2.6 | -1.9 | 3.4x |
| [L/14, g/14] | +0.0 | +0.0 | +0.1 | 2.6x |
| [B/16, g/14] | +0.1 | +0.1 | +0.1 | 6.1x |
| [B/16, L/14, g/14] | +0.1 | +0.6 | +0.2 | 5.2x (1.75x) |
| | | CLIP, ConvNeXt | | |
| [XXL] | 45.78 | 70.74 | 79.60 | 1x |
| [B] | -7.3 | -6.3 | -5.1 | 9.9x |
| [L] | -2.2 | -2.18 | -1.7 | 4.4x |
| [B, XXL] | -0.3 | +0.3 | -0.2 | 5.0x |
| [B, L, XXL] | +0.4 | -0.1 | -0.6 | 4.5x (1.97x) |
| | | BLIP | | |
| [L] | 60.05 | 83.78 | 90.56 | 1x |
| [B] | -0.5 | +0.1 | -0.3 | 3.5x |
| [B, L] | 0.0 | +0.2 | +0.1 | 2.6x |

Table 1. Recall@$k$ in % and cost reduction factors for various uncascaded models, 2-level cascades and 3-level cascades on Flickr30k and MSCOCO with $m_1 = 50$ and $m_2 = 14$. 2-level cascades list the lifetime cost reduction factor $F_{\text{life}}$. B/16, L/14, g/14 and B, L, XXL and B, L denote increasingly expensive CLIP ViT, CLIP ConvNeXt and BLIP models, respectively. 3-level cascades list the early query latency cost reduction factor $F_{\text{latency}}$ relative to the preceding 2-level cascade. We omit results for BLIP on Flickr30k because BLIP-L does not improve over BLIP-B.

## References

[1] Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *Proceedings of the IEEE international conference on computer vision*, pages 3361–3369, 2015.

[2] Lingjiao Chen, Matei Zaharia, and James Y Zou. Frugalml: How to use ml prediction apis more accurately and cheaply. *Advances in neural information processing systems*, 33:10685–10696, 2020.

[3] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3009–3018, 2019.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] 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. *arXiv preprint arXiv:2010.11929*, 2020.

[6] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. Vse++: Improving visual-semantic embeddings with hard negatives. *arXiv preprint arXiv:1707.05612*, 2017.

[7] Gregor Geigle, Jonas Pfeiffer, Nils Reimers, Ivan Vulić, and Iryna Gurevych. Retrieve fast, rerank smart: Cooperative and joint approaches for improved cross-modal retrieval. *arXiv preprint arXiv:2103.11920*, 2021.

[8] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[9] Zhiming Hu, Lan Xiao, Mete Kemertas, Caleb Phillips, Iqbal Mohomed, and Afsaneh Fazly. Crispsearch: low-latency on-device language-based image retrieval. In *Proceedings of the 13th ACM Multimedia Systems Conference*, pages 62–72, 2022.

[10] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. OpenCLIP, July 2021.

[11] Surbhi Jain and Joydip Dhar. Image based search engine using deep learning. In *2017 Tenth International Conference on Contemporary Computing (IC3)*, pages 1–7. IEEE, 2017.

[12] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[13] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. Noscope: optimizing neural network queries over video at scale. *arXiv preprint arXiv:1703.02529*, 2017.

[14] Woosuk Kwon, Sehoon Kim, Michael Mahoney, Joseph Hassoun, Kurt Keutzer, and Asghar Gholami. A fast post-training pruning framework for transformers. 03 2022.

[15] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[16] Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11336–11344, Apr. 2020.

[17] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5325–5334, 2015.

[18] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022.

[19] Yanghao Li, Haoqi Fan, Ronghang Hu, Christoph Feichtenhofer, and Kaiming He. Scaling language-image pre-training via masking. *arXiv preprint arXiv:2212.00794*, 2022.

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[21] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[22] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.

[23] Richa Mishra and Surya Prakash Tripathi. Deep learning based search engine for biomedical images using convolutional neural networks. *Multimedia Tools and Applications*, 80(10):15057–15065, 2021.

[24] Maryam Mohsin. 10 google search statistics you need to know in 2022 [infographic]. `https://www.oberlo.com/blog/google-search-statistics`, 2022. Accessed: 2022-12-02.

[25] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649, 2015.

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[27] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[28] Lester Solbakken. Text-to-image search with vespa. `https://blog.vespa.ai/text-image-search/`, 2021. Accessed: 2022-12-02.

[29] Tim Soulo. Search traffic study. https://ahrefs.com/blog/search-traffic-study/, 2020. Accessed: 2022-12-02.

[30] Siqi Sun, Yen-Chun Chen, Linjie Li, Shuohang Wang, Yuwei Fang, and Jingjing Liu. Lightningdot: Pre-training visual-semantic embeddings for real-time image-text retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 982–997, 2021.

[31] Hima Tammineedi. Evertrove - we made a usable ml-powered image search using openai's clip - search millions of images. https://www.reddit.com/r/MachineLearning/comments/lcjizm/p_evertrove_we_made_a_usable_mlpowered_image/, 2021. Accessed: 2022-12-02.

[32] Hao Tan and Mohit Bansal. LXMERT: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[33] Yan Zeng, Xinsong Zhang, and Hang Li. Multi-grained vision language pre-training: Aligning texts with visual concepts. *arXiv preprint arXiv:2111.08276*, 2021.

[34] Yan Zeng, Xinsong Zhang, Hang Li, Jiawei Wang, Jipeng Zhang, and Wangchunshu Zhou. $X^2$-vlm: All-in-one pre-trained model for vision-language tasks. *arXiv preprint arXiv:2211.12402*, 2022.

[35] Lisai Zhang, Hongfa Wu, Qingcai Chen, Yimeng Deng, Joanna Siebert, Zhonghua Li, Yunpeng Han, Dejiang Kong, and Zhao Cao. Vldeformer: Vision–language decomposed transformer for fast cross-modal retrieval. *Knowledge-Based Systems*, 252:109316, 2022.

[36] Ting Zhang, Chao Du, and Jingdong Wang. Composite quantization for approximate nearest neighbor search. In *International Conference on Machine Learning*, pages 838–846. PMLR, 2014.

[37] Ligeng Zhu. THOP: PyTorch-OpCounter. https://github.com/Lyken17/pytorch-OpCounter, 2022.