Scaling Language-Image Pre-training via Masking

Yanghao Li* Haoqi Fan* Ronghang Hu* Christoph Feichtenhofer† Kaiming He†

*equal technical contribution, †equal advising

Meta AI, FAIR

https://github.com/facebookresearch/flip

Abstract

We present Fast Language-Image Pre-training (FLIP), a simple and more efficient method for training CLIP [32]. Our method randomly masks out and removes a large portion of image patches during training. Masking allows us to learn from more image-text pairs given the same wall-clock time and contrast more samples per iteration with similar memory footprint. It leads to a favorable trade-off between accuracy and training time. In our experiments on 400 million image-text pairs, FLIP improves both accuracy and speed over the no-masking baseline. On a large diversity of downstream tasks, FLIP dominantly outperforms the CLIP counterparts trained on the same data. Facilitated by the speedup, we explore the scaling behavior of increasing the model size, data size, or training length, and report encouraging results and comparisons. We hope that our work will foster future research on scaling vision-language learning.

1. Introduction

Language-supervised visual pre-training, e.g., CLIP [32], has been established as a simple yet powerful methodology for learning representations. Pre-trained CLIP models stand out for their remarkable versatility: they have strong zero-shot transferability [32]; they demonstrate unprecedented quality in text-to-image generation (e.g., [53, 55]); the pre-trained encoder can improve multimodal and even unimodal visual tasks. Like the role played by supervised pre-training a decade ago [40], language-supervised visual pre-training is new fuel empowering various tasks today.

Unlike classical supervised learning with a pre-defined label set, natural language provides richer forms of supervision, e.g., on objects, scenes, actions, context, and their relations, at multiple levels of granularity. Due to the complex nature of vision plus language, large-scale training is essential for the capability of language-supervised models. For example, the original CLIP models [32] were trained on 400 million data for 32 epochs—which amount to 10,000 ImageNet [16] epochs, taking thousands of GPU-days [32, 36].

Even using high-end infrastructures, the wall-clock training time is still a major bottleneck hindering explorations on scaling vision-language learning.

We present Fast Language-Image Pre-training (FLIP), a simple method for efficient CLIP training. Inspired by the sparse computation of Masked Autoencoders (MAE) [29], we randomly remove a large portion of image patches during training. This design introduces a trade-off between how carefully we look at a sample pair vs. how many sample pairs we can process. Using masking, we can: (i) see more sample pairs (i.e., more epochs) under the same wall-clock training time, and (ii) compare/contrast more sample pairs at each step (i.e., larger batches) under similar memory footprint. Empirically, the benefits of processing more sample pairs greatly outweigh the degradation of per-sample encoding, resulting in a favorable trade-off.
By removing 50%-75% patches of a training image, our method reduces computation by 2-4×; it also allows using 2-4× larger batches with little extra memory cost, which boost accuracy thanks to the behavior of contrastive learning [30, 11]. As summarized in Fig. 1, FLIP trains >3× faster in wall-clock time for reaching similar accuracy as its CLIP counterpart; with the same number of epochs, FLIP reaches higher accuracy than its CLIP counterpart while still being 2-3× faster.

We show that FLIP is a competitive alternative to CLIP on various downstream tasks. Pre-trained on the same LAION-400M dataset [56], FLIP dominantly outperforms its CLIP counterparts (OpenCLIP [36] and our own reproduction), as evaluated on a large variety of downstream datasets and transfer scenarios. These comparisons suggest that FLIP can readily enjoy the faster training speed while still providing accuracy gains.

Facilitated by faster training, we explore scaling FLIP pre-training. We study these three axes: (i) scaling model size, (ii) scaling dataset size, or (iii) scaling training schedule length. We analyze the scaling behavior through carefully controlled experiments. We observe that model scaling and data scaling can both improve accuracy, and data scaling can show gains at no extra training cost. We hope our method, results, and analysis will encourage future research on scaling vision-language learning.

2. Related Work

Learning with masking. Denoising Autoencoders [63] with masking noise [64] were proposed as an unsupervised representation learning method over a decade ago. One of its most outstanding applications is masked language modeling represented by BERT [18]. In computer vision, explorations along this direction include predicting large missing regions [50], sequence of pixels [10], patches [20, 29, 71], or pre-computed features [6, 66].

The Masked Autoencoder (MAE) method [29] further takes advantage of masking to reduce training time and memory. MAE sparsely applies the ViT encoder [20] to visible content. It also observes that a high masking ratio is beneficial for accuracy. The MAE design has been applied to videos [61, 22], point clouds [49], graphs [59, 9, 32], audio [4, 47, 13, 35], visual control [70, 57], vision-language [23, 41, 31, 19], and other modalities [5].

Our work is related to MAE and its vision-language extensions [23, 41, 31, 19]. However, our focus is on the scaling aspect enabled by the sparse computation; we address the challenge of large-scale CLIP training [52], while previous works [23, 41, 31, 19] are limited in terms of scale. Our method does not perform reconstruction and is not a form of autoencoding. Speeding up training by masking is studied in [69] for self-supervised contrastive learning, e.g., for MoCo [30] or BYOL [27], but its accuracy could be limited by the scaling behavior of image-only contrastive learning.

Language-supervised learning. In the past years, CLIP [52] and related works (e.g., [37, 51]) have popularized learning visual representations with language supervision. CLIP is a form of contrastive learning [28] by comparing image-text sample pairs. Beyond contrastive learning, generative learning methods have been explored [17, 65, 2, 74], optionally combined with contrastive losses [74]. Our method focuses on the CLIP method, while we hope it can be extended to generative methods in the future.

3. Method

In a nutshell, our method simply masks out the input data in CLIP [52] training and reduces computation. See Fig. 2.

In our scenario, the benefit of masking is on wisely spending computation. Intuitively, this leads to a trade-off between how densely we encode a sample against how many samples we compare as the learning signal. By introducing masking, we can: (i) learn from more image-text pairs under the same wall-clock training time, and (ii) have a contrastive objective over a larger batch under the same memory constraint. We show by experiments that for both aspects, our method is at an advantage in the trade-off. Next we introduce the key components of our method.

Image masking. We adopt the Vision Transformer (ViT) [20] as the image encoder. An image is first divided into a grid of non-overlapping patches. We randomly mask out a large portion (e.g., 50% or 75%) of patches; the ViT encoder is only applied to the visible patches, following [29]. Using a masking ratio of 50% (or 75%) reduces the time complexity of image encoding to 1/2 (or 1/4); it also allows using a 2× (or 4×) larger batch with a similar memory cost for image encoding.
Text masking. Optionally, we perform text masking in the same way as image masking. We mask out a portion of the text tokens and apply the encoder only to the visible tokens, as in [29]. This is unlike BERT [18] that replaces them with a learned mask token. Such sparse computation can reduce the text encoding cost. However, as the text encoder is smaller [52], speeding it up does not lead to a better overall trade-off. We study text masking for ablation only.

Objective. The image/text encoders are trained to minimize a contrastive loss [48]. The negative sample pairs for contrastive learning consist of other samples in the same batch [11]. It has been observed that a large number of negative samples is critical for self-supervised contrastive learning on images [30, 11]. This property is more prominent in language-supervised learning.

We do not use a reconstruction loss, unlike MAE [29]. We find that reconstruction is not necessary for good performance on zero-shot transfer. Waiving the decoder and reconstruction loss yields a better speedup.

Unmasking. While the encoder is pre-trained on masked images, it can be directly applied on intact images without changes, as is done in [29]. This simple setting is sufficient to provide competitive results and will serve as our baseline in ablation experiments.

To close the distribution gap caused by masking, we can set the masking ratio as 0% and continue pre-training for a small number of steps. This unmasking tuning strategy produces a more favorable accuracy/time trade-off.

3.1. Implementation

Our implementation follows CLIP [52] and OpenCLIP [36], with a few modifications we describe in the following. Hyper-parameters are in the appendix.

Our image encoder follows the ViT paper [20]. We do not use the extra LayerNorm [3] after patch embedding, like [20] but unlike [52]. We use global average pooling at the end of the image encoder. The input size is 224.

Our text encoder is a non-autoregressive Transformer [62], which is easier to adapt to text masking for ablation. We use a WordPiece tokenizer as in [18]. We pad or cut the sequences to a fixed length of 32. We note that CLIP in [52] uses an autoregressive text encoder, a BytePairEncoding tokenizer, and a length of 77. These designs make marginal differences as observed in our initial reproduction.

The outputs of the image encoder and text encoder are projected to the same-dimensional embedding space by a linear layer. The cosine similarities of the embeddings, scaled by a learnable temperature parameter [52], are the input to the InfoNCE loss [48].

In zero-shot transfer, we follow the prompt engineering in the code of [52]. We use their provided 7 prompt templates for ImageNet zero-shot transfer.

Our implementation is based on JAX [8] with the t5x library [54] for large-scale distributed training. Our training is run on TPU v3 infrastructure.

4. Experiments

4.1. Ablations

We first ablate the FLIP design. The image encoder is ViT-L/16 [20], and the text encoder has a smaller size as per [52]. We train on LAION-400M [36] and evaluate zero-shot accuracy on ImageNet-1K [16] validation.

Table 1 shows the ablations trained for 6.4 epochs. Fig. 1 plots the trade-off for up to 32 epochs [52]. The results are benchmarked in 256 TPU-v3 cores, unless noted.

Masking ratio. Table 1a studies the image masking ratios. Here we scale the batch size accordingly (ablated next), so as to roughly maintain the memory footprint. The 0% masking entry indicates our CLIP counterpart. Masking 50% gives 1.2% higher accuracy than the CLIP baseline, and masking 75% is on par with the baseline. Speed-wise, masking 50% or 75% takes only 0.50 or 0.33 wall-clock training time, thanks to the large reduction on FLOPs.

Batch size. We ablate the effect of batch size in Table 1b. Increasing the batch size consistently improves accuracy.

Notably, even using the same batch size of 16k, our 50% masking entry has a comparable accuracy (68.5%, Table 1b) with the 0% masking baseline (68.6%, Table 1a). It is possible that the regularization introduced by masking can reduce overfitting, partially counteracting the negative effect of losing information in this setting. With a higher masking ratio of 75%, the negative effect is still observed when keeping the batch size unchanged.

Our masking-based method naturally encourages using large batches. There is little extra memory cost if we scale the batch size according to the masking ratio, as we report in Table 1a. In practice, the available memory is always a limitation for larger batches. For example, the setting in Table 1a has reached the memory limit in our high-end infrastructure (256 TPU-v3 cores with 16GB memory each). The memory issue is more demanding if using fewer devices, and the gain of our method would be more prominent due to the nearly free increase of batch sizes.

Text masking. Table 1c studies text masking. Randomly masking 50% text decreases accuracy by 2.2%. This is in line with the observation [29] that language data has higher...
information-density than images and thus the text masking ratio should be lower.

As variable-length text sequences are padded to generate a fixed-length batch, we can prioritize masking out the padding tokens. Prioritized sampling preserves more valid tokens than randomly masking the padded sequences uniformly. It reduces the degradation to 0.4%.

While our text masking is more aggressive than typical masked language models (e.g., 15% in [18]), the overall speed gain is marginal. This is because the text encoder is smaller and the text sequence is short. The text encoder costs merely 4.4% computation vs. the image encoder (without masking). Under this setting, text masking is not a worthwhile trade-off and we do not mask text in our other experiments.

**Inference unmasking.** By default, we apply our models on intact images at inference-time, similar to [29]. While masking creates a distribution shift between training and inference, simply ignoring this shift works surprisingly well (Table 1d, ‘w/o mask’), even under zero-shot setting where no training is ever done on full images.

Table 1d reports that if using masking at inference time, the accuracy drops by a lot (e.g., 7.3%). This drop can be partially caused by information loss at inference, so we also compare with ensembling multiple masked views [10], where the views are complementary to each other and put together cover all patches. Ensembling reduces the gap (Table 1d), but still lags behind the simple full-view inference.

**Unmasked tuning.** Our ablation experiments thus far do not involve unmasked tuning. Table 1e reports the results of unmasked tuning for extra 0.32 epoch on the pre-training dataset. It increases accuracy by 1.3% at the high masking ratio of 75%, suggesting that tuning can effectively reduce the distribution gap between pre-training and inference.

Fig. 3 plots the trade-off affected by unmasked tuning (solid vs. dashed). Unmasked tuning leads to a more desirable trade-off for 75% masking; it has a comparable trade-off for 50% masking but improves final accuracy.

**Reconstruction.** In Table 1f we investigate adding a reconstruction loss function. The reconstruction head follows the design in MAE [29]: it has a small decoder and reconstructs normalized image pixels. The reconstruction loss is added to the contrastive loss.

Table 1f shows that reconstruction has a small negative impact for zero-shot results. We also see a similar though slight less drop for fine-tuning accuracy on ImageNet. While this can be a result of suboptimal hyperparameters (e.g., balancing two losses), to strive for simplicity, we decide not to use the reconstruction loss. Waiving the reconstruction head also helps simplify the system and improves the accuracy/time trade-off.

### Table 1. Zero-shot ablation experiments

<table>
<thead>
<tr>
<th>masking_ratio</th>
<th>batch_size</th>
<th>FLOPs (G)</th>
<th>time (h)</th>
<th>accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>16k</td>
<td>1.00×</td>
<td>1.00×</td>
<td>68.6</td>
</tr>
<tr>
<td>50%</td>
<td>32k</td>
<td>0.52×</td>
<td>0.50×</td>
<td>69.6</td>
</tr>
<tr>
<td>75%</td>
<td>64k</td>
<td>0.28×</td>
<td>0.33×</td>
<td>68.2</td>
</tr>
</tbody>
</table>

(a) **Image masking** yields higher or comparable accuracy and speeds up training. Entries are subject to the same memory limit.

<table>
<thead>
<tr>
<th>masking_ratio</th>
<th>batch_size</th>
<th>mask 50%</th>
<th>mask 75%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>16k</td>
<td>68.5</td>
<td>65.8</td>
</tr>
<tr>
<td>50%</td>
<td>32k</td>
<td>69.6</td>
<td>67.3</td>
</tr>
<tr>
<td>75%</td>
<td>64k</td>
<td>70.4</td>
<td>68.2</td>
</tr>
</tbody>
</table>

(b) **Batch size.** A large batch has big gains over smaller batches.

<table>
<thead>
<tr>
<th>masking_ratio</th>
<th>batch_size</th>
<th>mask 50%</th>
<th>mask 75%</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline, 0%</td>
<td>32k</td>
<td>69.6</td>
<td>68.2</td>
</tr>
<tr>
<td>random, 50%</td>
<td>16k</td>
<td>69.6</td>
<td>65.8</td>
</tr>
<tr>
<td>prioritized, 50%</td>
<td>16k</td>
<td>69.6</td>
<td>67.8</td>
</tr>
</tbody>
</table>

(c) **Text masking** performs decently, but the speed gain is marginal as its encoder is smaller. Here the image masking ratio is 75%.

<table>
<thead>
<tr>
<th>masking_ratio</th>
<th>batch_size</th>
<th>mask 50%</th>
<th>mask 75%</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>69.6</td>
<td>68.2</td>
<td></td>
</tr>
<tr>
<td>+ MAE</td>
<td>69.4</td>
<td>67.9</td>
<td></td>
</tr>
</tbody>
</table>

(d) **Inference unmasking.** Inference on intact images performs strongly even without tuning.

(e) **Unmasked tuning.** The distribution shift by masking is reduced by a short tuning.

(f) **Reconstruction.** Adding the MAE reconstruction loss has no gain.

---

![Accuracy vs. training time trade-off](image-url)

Figure 3. **Accuracy vs. training time trade-off in detail.** The setting follows Table 1a. Training is for 6.4, 12.8, or 32 epochs, for each masking ratio. Unmasked tuning, if applied, is for 0.32 epoch. All are benchmarked in 256 TPU-v3 cores. Zero-shot accuracy is on IN-1K validation. The model is ViT-L/16. Our method speeds up training and increases accuracy.
the influence of the data and of the methods. This allows us to better understand the effect of the FLIP method. Our implementation subtleties and allows us to pinpoint the differences. Our CLIP reproduction further helps isolate other effects.

4.2. Comparisons with CLIP

In this section, we compare with various CLIP baselines in a large variety of scenarios. We show that our method is a competitive alternative to CLIP; as such, our fast training method is a more desirable choice in practice.

We consider the following CLIP baselines:

- The original CLIP checkpoints [52], trained on the private dataset WIT-400M.
- OpenCLIP [36], trained on LAION-400M.
- Our CLIP reproduction, trained on LAION-400M.

The original CLIP [52] was trained on a private dataset, so a direct comparison with it should reflect the effect of data, not just methods. OpenCLIP [36] is a faithful reproduction of CLIP yet trained on a public dataset that we can use, so it is a good reference for us to isolate the effect of dataset differences. Our CLIP reproduction further helps isolate other implementation subtleties and allows us to pinpoint the effect of the FLIP method.

For all tasks studied in this subsection, we compare with all these CLIP baselines. This allows us to better understand the influence of the data and of the methods.

Accuracy vs. time trade-off. Fig. 3 presents a detailed view on the accuracy vs. training time trade-off. We extend the schedule to up to 32 epochs [52].

As shown in Fig. 3, FLIP has a clearly better trade-off than the CLIP counterpart. It can achieve similar accuracy as CLIP while enjoying a speedup of >3×. With the same 32-epoch schedule, our method is ~1% more accurate than the CLIP counterpart and 2× faster (masking 50%).

The speedup of our method is of great practical value. The CLIP baseline takes ~10 days training in 256 TPU-v3 cores, so a speedup of 2-3× saves many days in wall-clock time. This speedup facilitates exploring the scaling behavior, as we will discuss later in Sec. 4.3.

<table>
<thead>
<tr>
<th>case</th>
<th>data</th>
<th>epochs</th>
<th>B/16</th>
<th>L/16</th>
<th>L/14</th>
<th>H/14</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP [52]</td>
<td>WIT-400M</td>
<td>32</td>
<td>68.6</td>
<td>-</td>
<td>75.3</td>
<td>-</td>
</tr>
<tr>
<td>OpenCLIP [36]</td>
<td>LAION-400M</td>
<td>32</td>
<td>67.1</td>
<td>-</td>
<td>72.8</td>
<td>-</td>
</tr>
<tr>
<td>CLIP, our repro.</td>
<td>LAION-400M</td>
<td>32</td>
<td>68.2</td>
<td>72.4</td>
<td>73.1</td>
<td>-</td>
</tr>
<tr>
<td>FLIP</td>
<td>LAION-400M</td>
<td>32</td>
<td>68.0</td>
<td>74.3</td>
<td>74.6</td>
<td>75.5</td>
</tr>
</tbody>
</table>

Table 2. Zero-shot accuracy on ImageNet-1K classification, compared with various CLIP baselines. The image size is 224. The entries noted by grey are pre-trained on a different dataset. Our models use a 64k batch, 50% masking ratio, and unmasked tuning.

<table>
<thead>
<tr>
<th>case</th>
<th>data</th>
<th>epochs</th>
<th>model</th>
<th>zero-shot</th>
<th>linear probe</th>
<th>fine-tune</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP</td>
<td>WIT-400M</td>
<td>32</td>
<td>L/14</td>
<td>75.3</td>
<td>83.9†</td>
<td>-</td>
</tr>
<tr>
<td>CLIP, our transfer</td>
<td>WIT-400M</td>
<td>32</td>
<td>L/14</td>
<td>75.3</td>
<td>83.0</td>
<td>87.4</td>
</tr>
<tr>
<td>OpenCLIP [36]</td>
<td>LAION-400M</td>
<td>32</td>
<td>L/14</td>
<td>72.8</td>
<td>82.1</td>
<td>86.2</td>
</tr>
<tr>
<td>CLIP, our repro.</td>
<td>LAION-400M</td>
<td>32</td>
<td>L/16</td>
<td>72.4</td>
<td>82.6</td>
<td>86.3</td>
</tr>
<tr>
<td>FLIP</td>
<td>LAION-400M</td>
<td>32</td>
<td>L/16</td>
<td>74.3</td>
<td>83.6</td>
<td>86.9</td>
</tr>
</tbody>
</table>

Table 3. Linear probing and fine-tuning accuracy on ImageNet-1K classification, compared with various CLIP baselines. The image size is 224. †: CLIP in [52] optimizes with L-BFGS; we use SGD instead.

ImageNet zero-shot transfer. In Table 2 we compare with the CLIP baselines on ImageNet-1K [16] zero-shot transfer.

As a sanity check, our CLIP reproduction has slightly higher accuracy than OpenCLIP trained on the same data. The original CLIP has higher accuracy than our reproduction and OpenCLIP, which could be caused by the difference between the pre-training datasets.

Table 2 reports the results of our FLIP models, using the best practice as we have ablated in Table 1 (a 64k batch, 50% masking ratio, and unmasked tuning). For ViT-L/14, our method has 74.6% accuracy, which is 1.8% higher than OpenCLIP and 1.5% higher than our CLIP reproduction. Comparing with the original CLIP, our method reduces the gap to 0.7%. We hope our method will improve the original CLIP result if it were trained on the WIT data.

ImageNet linear probing. Table 3 compares the linear probing results, i.e., training a linear classifier on the target dataset with frozen features. FLIP has 83.6% accuracy, 1.0% higher than our CLIP counterpart. It is also 0.6% higher than our transfer of the original CLIP checkpoint, using the same SGD trainer.

ImageNet fine-tuning. Table 3 also compares full fine-tuning results. Our fine-tuning implementation follows MAE [29], with the learning rate tuned for each entry. It is worth noting that with our fine-tuning recipe, the original CLIP checkpoint reaches 87.4%, much higher than previous reports [68, 67, 33] on this metric. CLIP is still a strong model under the fine-tuning protocol.

FLIP outperforms the CLIP counterparts pre-trained on the same data. Our result of 86.9% (or 87.1% using L/14) is behind but close to the result of the original CLIP checkpoint’s 87.4%, using our fine-tuning recipe.

For a legacy reason, we pre-trained our ViT-L models with a patch size of 16, following the original ViT paper [20]. The CLIP paper [52] uses L/14 instead. To save resources, we report our L/14 results by tuning the L/16 pre-trained model, in a way similar to unmasked tuning.
Table 4. Zero-shot accuracy on more classification datasets. Table 4 shows the performance of various CLIP baselines on additional classification datasets. The model is ViT-L/14 with an image size of 224. Entries in green are the best ones using the LAION-400M data.

<table>
<thead>
<tr>
<th>case</th>
<th>model</th>
<th>data</th>
<th>Flickr30k</th>
<th>COCO</th>
<th>Flickr30k</th>
<th>COCO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>R@1</td>
<td>R@5</td>
<td>R@10</td>
<td>R@10</td>
</tr>
<tr>
<td>CLIP [52]</td>
<td>L/14@336 WIT-400M</td>
<td>88.0</td>
<td>98.7</td>
<td>99.4</td>
<td>58.4</td>
<td>81.5</td>
</tr>
<tr>
<td>CLIP, our eval.</td>
<td>L/14@336 WIT-400M</td>
<td>88.9</td>
<td>98.7</td>
<td>99.9</td>
<td>58.7</td>
<td>80.4</td>
</tr>
<tr>
<td>CLIP, our eval.</td>
<td>L/14 WIT-400M</td>
<td>87.8</td>
<td>99.1</td>
<td>99.8</td>
<td>56.2</td>
<td>79.8</td>
</tr>
<tr>
<td>OpenCLIP [36], our eval.</td>
<td>L/14 LAION-400M</td>
<td>87.3</td>
<td>97.9</td>
<td>99.1</td>
<td>58.0</td>
<td>80.6</td>
</tr>
<tr>
<td>OpenCLIP, our impl.</td>
<td>L/14 LAION-400M</td>
<td>87.4</td>
<td>98.4</td>
<td>99.5</td>
<td>59.1</td>
<td>82.5</td>
</tr>
<tr>
<td>FLIP</td>
<td>L/14 LAION-400M</td>
<td>89.1</td>
<td>98.5</td>
<td>99.6</td>
<td>60.2</td>
<td>82.6</td>
</tr>
</tbody>
</table>

Table 5. Zero-shot image/text retrieval. Table 5 compares various CLIP baselines on image/text retrieval tasks. The image size is 224 if not noted. Entries in green are the best ones using the LAION-400M data.

<table>
<thead>
<tr>
<th>case</th>
<th>model</th>
<th>data</th>
<th>IN-V2 top@1</th>
<th>IN-A top@1</th>
<th>ObjectNet top@1</th>
<th>IN-Sketch top@1</th>
<th>IN-Vid top@1</th>
<th>PM-0</th>
<th>PM-10</th>
<th>YTBB</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP [52]</td>
<td>L/14@336 WIT-400M</td>
<td>70.1</td>
<td>73.7</td>
<td>68.9</td>
<td>72.5</td>
<td>60.2</td>
<td>95.3</td>
<td>89.2</td>
<td>95.2</td>
<td>88.5</td>
</tr>
<tr>
<td>CLIP, our eval.</td>
<td>L/14@336 WIT-400M</td>
<td>70.4</td>
<td>78.0</td>
<td>80.9</td>
<td>69.3</td>
<td>59.7</td>
<td>95.9</td>
<td>88.8</td>
<td>95.3</td>
<td>89.4</td>
</tr>
<tr>
<td>CLIP, our eval.</td>
<td>L/14 WIT-400M</td>
<td>69.5</td>
<td>71.9</td>
<td>86.8</td>
<td>68.6</td>
<td>58.5</td>
<td>94.6</td>
<td>87.0</td>
<td>94.1</td>
<td>86.4</td>
</tr>
<tr>
<td>OpenCLIP [36], our eval.</td>
<td>L/14 LAION-400M</td>
<td>64.0</td>
<td>48.3</td>
<td>84.3</td>
<td>58.8</td>
<td>56.9</td>
<td>90.3</td>
<td>81.4</td>
<td>86.5</td>
<td>77.8</td>
</tr>
<tr>
<td>OpenCLIP, our impl.</td>
<td>L/14 LAION-400M</td>
<td>65.6</td>
<td>46.3</td>
<td>84.7</td>
<td>58.0</td>
<td>58.7</td>
<td>89.3</td>
<td>80.5</td>
<td>85.7</td>
<td>77.8</td>
</tr>
<tr>
<td>FLIP</td>
<td>L/14 LAION-400M</td>
<td>66.8</td>
<td>51.2</td>
<td>86.5</td>
<td>59.1</td>
<td>59.9</td>
<td>91.1</td>
<td>83.5</td>
<td>89.4</td>
<td>83.3</td>
</tr>
</tbody>
</table>

Table 6. Zero-shot robustness evaluation. Table 6 compares various CLIP baselines on robustness evaluation tasks. The image size is 224 if not noted. Entries in green are the best ones using the LAION-400M data.

<table>
<thead>
<tr>
<th>case</th>
<th>model</th>
<th>data</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>ROUGE-L acc.</th>
<th>CIDEr</th>
<th>SPICE</th>
<th>CIDEr</th>
<th>SPICE</th>
<th>VQAv2 acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP [52], our transfer</td>
<td>L/14 WIT-400M</td>
<td>37.5</td>
<td>29.6</td>
<td>58.7</td>
<td>126.9</td>
<td>22.8</td>
<td></td>
<td>82.5</td>
<td>12.1</td>
<td>76.6</td>
</tr>
<tr>
<td>OpenCLIP [36], our transfer</td>
<td>L/14 WIT-400M</td>
<td>36.7</td>
<td>29.3</td>
<td>58.4</td>
<td>125.0</td>
<td>22.7</td>
<td></td>
<td>83.4</td>
<td>12.3</td>
<td>74.5</td>
</tr>
<tr>
<td>CLIP, our repro.</td>
<td>L/16 WIT-400M</td>
<td>36.4</td>
<td>29.3</td>
<td>58.4</td>
<td>125.6</td>
<td>22.8</td>
<td></td>
<td>82.8</td>
<td>12.2</td>
<td>74.5</td>
</tr>
<tr>
<td>FLIP</td>
<td>L/16 WIT-400M</td>
<td>37.4</td>
<td>29.5</td>
<td>58.8</td>
<td>127.7</td>
<td>23.0</td>
<td></td>
<td>85.9</td>
<td>12.4</td>
<td>74.7</td>
</tr>
</tbody>
</table>

Table 7. Image Captioning and Visual Question Answering. Table 7 compares various CLIP baselines on Image Captioning and Visual Question Answering tasks. The image size is 224 if not noted. Entries in green are the best ones using the LAION-400M data. The results are from the COCO captioning test split of [38], nocaps val split, and VQA v2-test-dev split, respectively.

Zero-shot classification on more datasets. In Table 4 we compare on the extra datasets studied in [52]. As the results can be sensitive to evaluation implementation (e.g., text prompts, image pre-processing), we provide our evaluations of the original CLIP checkpoint and OpenCLIP.

Notably, we observe clear systematic gaps caused by pre-training data, as benchmarked using the same evaluation code. The WIT dataset is beneficial for some tasks (e.g., Aircraft, Country211, SST2), while LAION is beneficial for some others (e.g., Birdsnap, SUN397, Cars).

After isolating the influence of pre-training data, we observe that FLIP is dominantly better than OpenCLIP and our CLIP reproduction, as marked by green in Table 4.

Zero-shot retrieval. Table 5 reports image/text retrieval results on Flickr30k [73] and COCO [42]. FLIP outperforms all CLIP competitors, including the original CLIP (evaluated on the same 224 size). The WIT dataset has no advantage over LAION for these two retrieval datasets.

Zero-shot robustness evaluation. In Table 6 we compare on robustness evaluation, following [52]. We again observe clear systematic gaps caused by pre-training data. Using the same evaluation code (“our eval” in Table 6), CLIP pre-trained on WIT is clearly better than other entries pre-trained on LAION. Taking IN-Adversarial (IN-A) as an example: the LAION-based OpenCLIP [36] has only 48.3% accuracy (or 46.6% reported by [36]). While FLIP (51.2%) can outperform the LAION-based CLIP by a large margin, it is still 20% below the WIT-based CLIP (71.9%).

Discounting the influence of pre-training data, our FLIP training has clearly better robustness than its CLIP counter-
parts in all cases. We hypothesize that masking as a form of noise and regularization can improve robustness.

**Image Captioning.** See Table 7 for the captioning performance on COCO [42] and nocaps [1]. Our captioning implementation follows the cross-entropy training baseline in [7]. Unlike classification in which only a classifier layer is added after pre-training, here the fine-tuning model has a newly initialized captioner (detailed in appendix). In this task, FLIP outperforms the original CLIP checkpoint in several metrics. Compared to our CLIP baseline, which is pre-trained on the same data, FLIP also shows a clear gain, especially in BLEU-4 and CIDEr metrics.

**Visual Question Answering.** We evaluate on the VQAv2 dataset [26], with a fine-tuning setup following [21]. We use a newly initialized multimodal fusion transformer and an answer classifier to obtain the VQA outputs (detailed in appendix). Table 7 (rightmost column) reports results on VQAv2. All entries pre-trained on LAION perform similarly, and CLIP pre-trained on WIT is the best.

**Summary of Comparisons.** Across a large variety of scenarios, FLIP is dominantly better than its CLIP counterparts (OpenCLIP and our reproduction) pre-trained on the same LAION data, in some cases by large margins.

The difference between the WIT data and LAION data can create large systematic gaps, as observed in many downstream tasks. We hope our study will draw attention to these data-dependent gaps in future research.

### 4.3. Scaling Behavior

Facilitated by the speed-up of FLIP, we explore the scaling behavior beyond the largest case studied in CLIP [52]. We study scaling along either of these three axes:

- **Model scaling.** We replace the ViT-L image encoder with ViT-H, which has ~2× parameters. The text encoder is also scaled accordingly.
- **Data scaling.** We scale the pre-training data from 400 million to 2 billion, using the LAION-2B set [36]. To better separate the influence of more data from the influence of longer training, we fix the total number of sampled data (12.8B, which amounts to 32 epochs of 400M data and 6.4 epochs of 2B data).
- **Schedule scaling.** We increase the sampled data from 12.8B to 25.6B (64 epochs of 400M data).

We study scaling along one of these three axes at each time while keeping others unchanged. The results are summarized in Fig. 4 and Table 8.

**Training curves.** The three scaling strategies exhibit different trends in training curves (Fig. 4).

**Model scaling** (Fig. 4a) presents a clear gap that persists throughout training, though the gap is smaller at the end.

**Data scaling** (Fig. 4b), on the other hand, performs similarly at the first half of training, but starts to present a good gain later. Note that there is no extra computational cost in this setting, as we control the total number of sampled data.

**Schedule scaling** (Fig. 4c) trains 2× longer. To provide a more intuitive comparison, we plot a hypothetical curve that is rescaled by 1/2 along the x-axis (dashed line). Despite the longer training, the gain is diminishing or none (more numbers in Table 8).

**Transferability.** Table 8 provides an all-around comparison on various downstream tasks regarding the scaling behavior. Overall, model scaling and data scaling both can consistently outperform the baseline in all metrics, in some cases by large margins.

We categorize the downstream tasks into two scenarios: (i) zero-shot transfer, i.e., no learning is performed on the downstream dataset; (ii) transfer learning, i.e., part or all of the weights are trained on the downstream dataset. For the tasks studied here, data scaling is in general favored for zero-shot transfer, while model scaling is in general favored for transfer learning. However, it is worth noting that the transfer learning performance depends on the size of the downstream dataset, and training a big model on a too small downstream set is still subject to the overfitting risk.

It is encouraging to see that data scaling is clearly beneficial, even not incurring longer training nor additional computation. On the contrary, even spending more computation by schedule scaling gives diminishing returns. These comparisons suggest that large-scale data are beneficial mainly because they provide richer information.

Next we scale both model and data (Table 8, second last row). For all metrics, model+data scaling improves over scaling either alone. The gains of model scaling and data scaling are highly complementary: e.g., in zero-shot IN-1K, model scaling alone improves over the baseline by 1.2% (74.3%→75.5%), and data scaling alone improves by 1.5% (74.3%→75.8%). Scaling both improves by 3.3% (77.6%), more than the two deltas combined. This behavior is also observed in several other tasks. This indicates that a larger model desires more data to unleash its potential.

Finally, we report joint scaling all three axes (Table 8, last row). Our results show that combining schedule scaling leads to improved performances across most metrics. This suggests that schedule scaling is particularly beneficial when coupled with larger models and larger-scale data.

Our result of 78.8% on zero-shot IN-1K outperforms the state-of-the-art result trained on public data with ViT-H (78.0% of OpenCLIP). Also based on LAION-2B, their result is trained with 32B sampled data, 1.25× more than ours. Given the 50% masking we use, our training is estimated to be 2.5× faster than theirs if both were run on the same hardware. As OpenCLIP’s result reports a training cost of ~5,600 GPU-days, our method could save ~3,360
Our study involves controlled comparisons with various language-supervised training. CLIP \cite{radford2021clip} is an outstanding example of \textit{"simple algorithms that scale well"}. The simple design of CLIP allows it to be relatively easily executed at substantially larger scales and achieve big leaps compared to preceding methods. Our method largely maintains the simplicity of CLIP while pushing it further along the scaling aspect.

Our method can provide a 2-3× speedup or more. For the scale concerned in this study, such a speedup can reduce wall-clock time by a great amount (e.g., at the order of thousands of TPU/GPU-days). Besides accelerating research cycles, the speedup can also save a great amount of energy and commercial cost. These are all ingredients of great importance in large-scale machine learning research.

Our study involves controlled comparisons with various CLIP baselines, which help us to break down the gaps contributed by different factors. We show that FLIP outperforms its CLIP counterparts pre-trained on the same LAION data. By comparing several LAION-based models and the original WIT-based ones, we observe that the pre-training data creates big systematic gaps in several tasks.

Our study provides controlled experiments on scaling behavior. We observe that data scaling is a favored scaling dimension, given that it can improve accuracy with no extra cost at training or inference time. Our fast method encourages us to scale beyond what is studied in this work.

Our study provides controlled experiments on scaling behavior. We observe that data scaling is a favored scaling dimension, given that it can improve accuracy with no extra cost at training or inference time. Our fast method encourages us to scale beyond what is studied in this work.

5. Discussion and Conclusion

Language is a stronger form of supervision than classical closed-set labels. Language provides rich information for supervision. Therefore, scaling, which can involve increasing capacity (model scaling) and increasing information (data scaling), is essential for attaining good results in language-supervised training.

CLIP \cite{radford2021clip} is an outstanding example of \textit{"simple algorithms that scale well"}. The simple design of CLIP allows it to be relatively easily executed at substantially larger scales and achieve big leaps compared to preceding methods. Our method largely maintains the simplicity of CLIP while pushing it further along the scaling aspect.

Our method can provide a 2-3× speedup or more. For the scale concerned in this study, such a speedup can reduce wall-clock time by a great amount (e.g., at the order of thousands of TPU/GPU-days). Besides accelerating research cycles, the speedup can also save a great amount of energy and commercial cost. These are all ingredients of great importance in large-scale machine learning research.

Our study involves controlled comparisons with various CLIP baselines, which help us to break down the gaps contributed by different factors. We show that FLIP outperforms its CLIP counterparts pre-trained on the same LAION data. By comparing several LAION-based models and the original WIT-based ones, we observe that the pre-training data creates big systematic gaps in several tasks.

Our study provides controlled experiments on scaling behavior. We observe that data scaling is a favored scaling dimension, given that it can improve accuracy with no extra cost at training or inference time. Our fast method encourages us to scale beyond what is studied in this work.

Broader impacts. Training large-scale models costs high energy consumption and carbon emissions. While our method has reduced such cost to 1/2-1/3, the remaining cost is still sizable. We hope our work will attract more attention to the research direction on reducing the cost of training vision-language models.

The numerical results in this paper are based on a publicly available large-scale dataset \cite{LAION}. The resulting model weights will reflect the data biases, including potentially negative implications. When compared using the same data, the statistical differences between two methods should reflect the method properties to a good extent; however, when compared entries using different training data, the biases of the data should always be part of the considerations.
References


[29] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. arXiv:2111.06577, 2021.


