A. Is this specific to ViT image models?

No. In the main paper, we only used ViT models for all experiments. Could it be that LiT only works with ViT models, or is in some way specific to the Transformer architecture?

In order to verify that this is not the case, we applied the same recipe to comparably-sized models of different families. Table 6 shows the zero-shot performance with LiT on the CC12M dataset for ViT [20], Mixer [60], and ResNet [32]; all pre-trained on ImageNet21k. Following [13], we report parameter count, inference speed, and FLOPs to indicate our attempt to match the "model size". The results show that LiT works for different model families, but also confirm the finding of [45] that ViT models do seem more amenable to learning image-text mappings than other architectures of similar size.

Model	0shot	Adapt	I→T	T→I	Param	Speed	FLOPs
ViT-B/32	60.7	79.1	41.3	25.0	197 M	2855	12 G
Mixer-B/32	57.1	75.9	37.5	22.9	169 M	4208	9 G
BiT-M-R50	55.2	77.6	37.3	23.9	134 M	2159	11 G

Table 6. LiT with different model families. Showing zero-shot top-1 accuracy on ImageNet in comparison to fine-tuning (column "Adapt"). Inference "Speed" is in images per second per core.

B. Larger model capacity yields better results

Increasing the model capacity of the pre-trained image-tower improves zero-shot ImageNet accuracy more than increasing the capacity of the text-tower. Figure 7 shows substantial gains in the private data setup when the image tower capacity is increased from B/32 and base text tower (74.5%) to g/14 and huge text tower (81.2%). We take the pre-trained image towers from [68], and the text towers were trained from scratch.

The improvements in the public CC12M data setup range from 61.1% with a B/32 image tower and base text tower up to 67.6% with the L/16 model combined with a large text tower. In this setup, we used pre-trained BERT text towers [16] and pre-trained image models from [54] (using the "recommended checkpoints"). Note that in this case the increase from B/16 to L/16 is more modest (from 66.9% to 67.6% with the large text tower), and we see a similar improvement in ImageNet zero-shot performance when increasing the text tower size.

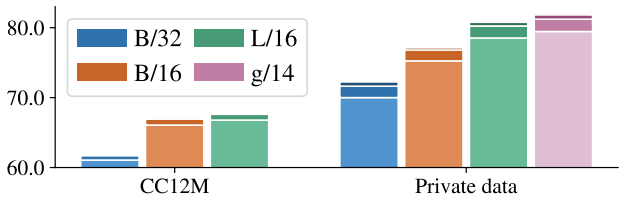

Figure 7. ImageNet zero-shot accuracy [%] with varying model capacity. Incremental improvemments due to larger *text* towers (base → large → huge) are shown as stacked bars.

C. Tuning details on our dataset

We use the pre-trained transformer models from [68]. ViT-B/32 was used for most of the ablation tests, and the larger ViT-B/16, ViT-L/16 and ViT-g/14 models are used in Section B for capacity impact evaluations. For our best `Lu` results, we adopt the ViT-g/14 model pre-trained in [68].

During contrastive-tuning, we use the AdaFactor optimizer [52] following [68]. We use 0.001 learning rate, and the default $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for AdaFactor optimizer. We use batch size 16384 by default, unless otherwise noted. Input image is simply resized to 224×224 resolution (apart from 288×288 resolution for "g/14*" model). No weight decay is used during tuning. We use cosine learning rate schedule with a linear learning rate warmup of 10k steps. We train our models for 55k steps by default, which equals to about 900 million seen image-text pairs during tuning. For our best runs, we scale up the training schedule to 18 billion seen image-text pairs. We use 128 TPU cores by default for the above experiments, and 256 TPU cores for our best run with 18 billion seen image-text pairs.

In the `Lu` setup, we do not attach the optional linear head on the image tower. We observe a very small quality improvement without using the image linear head, thus we remove it for simplicity.

D. Tuning details on CC12m

We use pre-trained ViT models from [54] (unless otherwise noted, we used the "recommended checkpoints" from that repository). On the text side, we use BERT-base and BERT-large from [16] for most experiments. In section 5.4 we use T5-base from [46] and mT5-base from [66].

We use the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$) for all models, except for models with Large text tower that were trained with a modified version of AdaFactor from [68] (same settings as described in Section C). The learning rate is set to 0.001, and the weight decay to 0.0001 (using "decoupled" weight decay as described in [39]). Gradients are clipped at global norm 1.

For training, the images are pre-processed by Inception-style cropping [57] to a size of 224 pixels. For evaluation, the images are resized to 224 pixels with bi-linear interpolation without cropping.

When tuning on the CC12M dataset, we train for 20

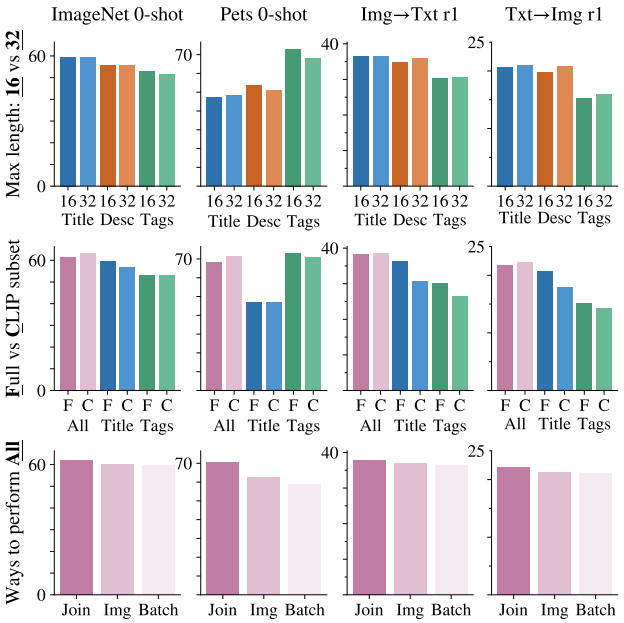

Figure 8. Ablations for YFCC100m. **Top:** even though the description field can be long, the potential benefit of using more than 16 tokens does not outweigh the increased memory and computation cost. **Middle:** When using all text signals, sticking to the CLIP subset is better according to the standard benchmarks, however see also Section 5.7. **Bottom:** Using all three text signals simultaneously for all examples works better than sampling one per image or per batch.

epochs (200 million seen image-text pairs), which corresponds to 12k steps with a batch size of 16384. The first 50k image-text pairs are used as minival validation set. The learning rate is ramped up linearly for the first 2k steps and then follows a cosine decay. Unless otherwise noted, we use the `LU` setup with a linear head on the text tower only.

E. How to use YFCC100m?

This section is an exploratory analysis of the YFCC100m dataset and provides guidance on what is a good setup for LiT. For each experiment we run, we try three learning-rates (0.001, 0.0008, 0.0003) and two weight-decays (0.0001 and 0.00001) and report the best result, this allows avoiding biasing conclusions due to sub-optimal hyper-parameters. We perform the exploration using the small ViT-B/32 AugReg [54] image tower and a BERT base [16] text tower and run tuning for 60 000 steps, although the same conclusions and similar scores are already reachable after 30 000 steps of tuning.

The YFCC100m dataset comes with a rich set of annotations for each image, including camera settings and geolocation. Out of all the annotations, three of them are potential candidates for learning image-text pairings: the image's title, a description, and a set of free-form tags. However, only partially overlapping subsets of 60 M, 30 M, and 65 M images come with a title, description, or tags, respectively. We first explore which supervision signal is most useful. For the description, we simply tokenize the provided text; for the title, we perform basic filtering and remove titles that start with `DSC`, `IMG`, `Picture`, consist of only the word `image` or consist of more than half digits; for the tags, we randomly shuffle their order, and join them with a random space, newline, or basic punctuation character in order to get a string which we then tokenize. The texts vary dramatically in length, we thus try maximum sequence lengths of 16 and 32 tokens. The first row of Figure 8 shows the result of this experiment. The difference between a maximum sequence length of 16 and 32 is small, however the memory savings are substantial and we thus restrict the sequence length to 16 tokens in all further experiments.

In terms of supervision signal, there is no single clear winner. We thus explore three ways of learning from all signals and so also make use of the full 100 M images. We can either *join*tly optimize them by summing up three contrastive losses for each image, or we can randomly sample one of the three sources for each *image* or for a whole mini*batch*. As can be seen in the bottom row of Figure 8, jointly using all signals consistently works better, although it requires triple the amount of passes through the text tower.

Finally, the authors of CLIP [45] provide a curated subset of roughly 15 M images, which contain high quality annotations in English. We refer to this subset as $\text{YFCC}_{\text{CLIP}}$. In the middle row of Figure 8, we compare how using the *F*ull YFCC100m for LiT compares to using the *C*LIP subset of it. Both seem to perform roughly on par for all signals for classification, but when using only titles or tags and performing image-text retrieval, it is better to apply LiT on the full YFCC100m dataset.

Overall, we obtain the best results with LiT using all text signals jointly on the $\text{YFCC}_{\text{CLIP}}$ subset. However, this investigation was performed with the small ViT-B/32 model, it is likely that a larger model may perform better when using the full dataset.

F. Effective batch size for contrastive loss

In this section, we study the impact of the effective batch size for contrastive loss. We use the `Lu` setup with a pre-trained B/32 image model, tuned for 900 million seen image-text pairs. In Figure 9, we see a clear improvement when using global contrastive loss. It has increased the effective batch size for contrastive learning, thus introducing more hard negatives and improving model quality. Interestingly, we found that larger batch size leads to better performance consistently. We leave extremely large batch size exploration to future work.

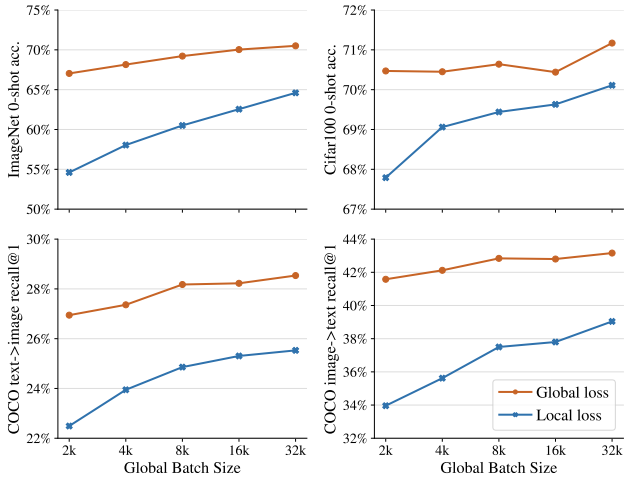

Figure 9. Impact of batch sizes for contrastive loss, including both global contrastive loss and local contrastive loss.

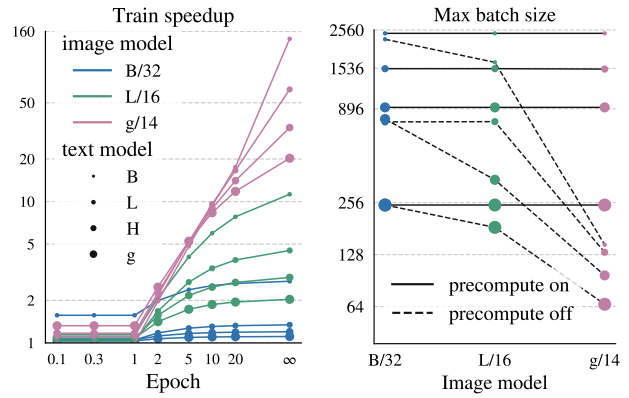

Figure 10. **Left**: Pre-computing image embeddings accelerates LiT, when tuning for more than a single epoch. **Right**: Pre-computing image embeddings in LiT allows larger batch size in memory.

G. Pre-computation for locked image models

In LiT method, the locked image model generates identical embeddings given the same image. Based on this characteristic, we use pre-computed image embeddings during tuning. It allows faster iterations and fitting larger text models in memory, as the image representations are extracted only once and no image models are loaded.

Figure 10 left shows how training speeds up as the number of epochs grows. When training no more than a single epoch, pre-computation keeps a constant speed ratio over re-computation, which increases from one (same speed) to larger than one (speedup) as image model size grows. After one epoch, pre-computation clearly accelerates training due to reused image representations. The speedup ratio becomes more visible as either the number of epochs or the image model size grows.

Model		Param (M)		Max speed			Max batch	
Image	Text	Pre	Non	Pre	Non	Inf	Pre	Non
B/32	B	105	195	2439	893	3294	2448	2262
B/32	L	320	410	924	688	3294	1528	751
B/32	H	640	730	468	390	3294	912	781
B/32	g	1007	1097	242	218	3294	248	248
L/16	B	105	406	2423	215	273	2448	1663
L/16	L	320	621	920	204	273	1528	754
L/16	H	640	942	465	160	273	912	347
L/16	g	1007	1308	240	118	273	248	184
g/14	B	105	1094	2409	17	17	2448	146
g/14	L	320	1310	932	15	17	1520	132
g/14	H	641	1630	467	14	17	912	97
g/14	g	1008	1997	243	12	17	248	66

Table 7. Pre-computation details. *Max speed* and *Max batch* describe metrics collected by maximum speed (img/sec/core) and batch size, respectively, corresponding to Figure 10. *Pre* and *Non* are metrics with and without pre-computation respectively; *Inf* describes pre-computation inference speed, which is only affected by image models. All experiments are run on 8 TPU v3 cores.

For experiments with pre-computed image embeddings, we count both pre-computation inference cost and tuning cost. Pre-computation will be performed on at most a single epoch on the image-text dataset. In practice, the pre-computed embeddings can be shared across different experiments, as long as the image tower is identical. As a result, the actual cost is even lower than our estimation. For experiments without pre-computed image embeddings, we count the actual contrastive-tuning cost.

Pre-computation eliminates loading the image model to memory during training, thus allowing larger batch sizes for contrastive loss. We search maximum batch sizes on each combination of image and text models with and without pre-computation, and show the results in Figure 10 right. We search for the maximum batch size for each model with a unified setup. We report the maximum batch size that the model can fit on 8 TPU v3 cores.

However, if image augmentations are enabled during training, we may not benefit much from pre-computation. The model sees different augmented images in multiple epochs. Nevertheless, the memory benefits still hold. All metric details are in Table 7.

H. Learning rate schedules

For most of the experiments, weights were either completely locked, or trained with the same learning rate schedule (linear warmup and cosine decay). We experimented with different learning rate schedules (Figure 11), mainly varying how the image tower was updated. We observed that training the image tower with a smaller learning rate

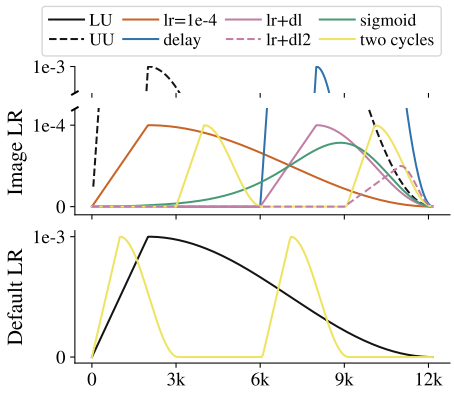

Figure 11. Different learning rate schedules. Note that the default LR schedule is shown in black in the lower part of the figure.

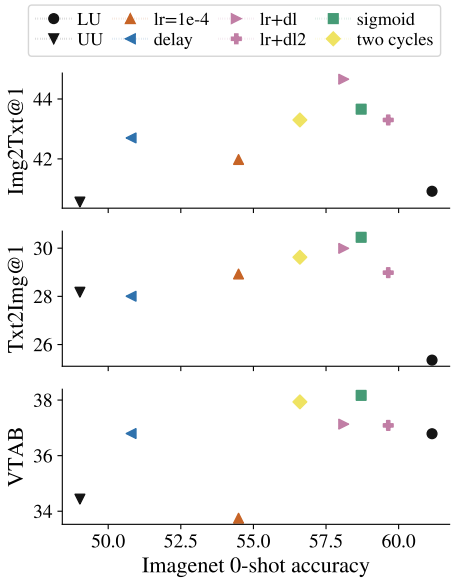

Figure 12. ITR and VTAB metrics as a function of ImageNet 0-shot accuracy for different LR schedules.

and/or delaying training of the image tower resulted in better retrieval metrics (Figure 12).

The default schedules (LU and UU) have the best and worst ImageNet 0-shot accuracy of all tried learning rate schedules. Compared to UU, both ITR/VTAB metrics and ImageNet 0-shot accuracy improve modestly, when the image learning rate is only scheduled for the second half of the training ("delay"). The ImageNet 0-shot accuracy improves more but the VTAB accuracy drops when the learning rate is set to a smaller value ("lr=1e-4"). Combining the delay with the smaller learning rate ("lr+dl") further improves both ITR/VTAB metrics and ImageNet 0-shot accuracy. A similar result is achieved by multiplying the learning rate in the UU setting with a sigmoid function ("sigmoid"). Alternating between freezing image tower and rext tower ("two cycles") finally performs somewhere between "lr+dl" and "lr=1e-4" schedules.

I. Zero-shot transfer details

I.1. Classification

We follow CLIP [45] for the zero-shot transfer evaluation. We use the identical ImageNet class label names and the same 80 prompt templates as in CLIP. During evaluation of private LiT models, we first resize the test image and then central crop with 0.875 aspect ratio to the target resolution. More specifically, we use 224×224 target resolution for CIFAR dataset and 288×288 target resolution for the remaining datasets. For all the public LiT models, we resize all test images to 224×224 for simplicity.

I.2. VTAB Evaluation

The Visual Task Adaptation benchmark [69] consists of 19 diverse visual tasks. We refer readers to the original publication for details about each dataset; here we just mention that they are split into three categories:

- **Natural**: These tasks contain classical "natural" real-world images obtained with a camera, such as vehicles, pets, scenery and household objects.
- **Specialized**: These are datasets of arguably "natural" images which were captured with specialised photographic equipment, such as satellite photographs and medical images.
- **Structured**: These assess understanding of scenes structure in some way, predominately from synthetic environments. Example tasks include 3D depth estimation and counting.

Note that there is significant overlap with the datasets assessed in [45], but it is not guaranteed that the same data splits were used.

Evaluation protocol. Previous works [45] define task-specific prompts and class names, but it is not clear exactly how an optimal set of prompts for a given task was chosen.

For VTAB, we define a search space of image preprocessing, prompt templates and classes, where the latter two are often per-task (e.g. using `a satellite photo of ...` or `an overhead photo of ...` for tasks involving satellite imagery). All such settings are tried on a small validation set of 800 images, and the optimal setting is then run on the official VTAB test set.

We note this is arguably not *zero-shot* transfer, but believe it is a principled and reproducible approach.

Prompts used. For all tasks, we considered 3 default sets of prompts

1. `A photo of a CLASS`

2. `CLASS`

3. The 6 CLIP prompts used for ImageNet[2]

We also consider some task specific prompts/class name settings. Note that these two degrees of freedom are orthogonal, and a text setting is defined by both. They are shown in Table 11 and Table 12. Not all of these prompts were equally useful; some are redundant, providing equal performance gain as other settings, and some do not provide performance gains at all. We show the performance delta comparing only the default prompts versus including a given text variant as well, to give a rough idea of how beneficial it was.

Can we assess zero-shot performance using VTAB? The strength of such a diverse benchmark is in the variety of its label spaces. ImageNet classes, though very fine-grained, are fairly generic. However, VTAB also includes *structured tasks* which are designed to assess the model's competence at tasks which aren't object recognition, such as counting and assessing distances and angles. This presents interesting difficulties for solving in a zero-shot natural language grounded manner. Figure 13 shows the zero-shot performance of many models developed for this paper. Their detailed performance is not important here - the gray lines show what a "random guesser" would achieve on each VTAB category. It is not an obvious number, as performance across categories is an average of all the constituent datasets, which have varying numbers of classes. It is clear from this figure that the structured performance does not significantly deviate from random guessing, despite extensive efforts in prompt engineering. We leave it as an open - and very interesting - research direction to figure how to make such models count and assess distances. Furthermore, though contrastive image-text training on the web can largely match supervised models on natural tasks, further improvements are needed on more specialist tasks.

I.3. Cross-modal retrieval

We compute retrieval metrics on MSCOCO captions [9], reporting the numbers on the test set (5 000 images, 25 010 captions). For the image to text retrieval, we rank all texts by decreasing cosine similarity of their embedding with the image embedding, and then report the fraction of images that ranks the correct text within the first (1, 5, 10) positions as the Recall@1, Recall@5, Recall@10 metrics. For the text to image retrieval, we compute the same metric, but ranking images and averaging over all texts. When showing a single number, we always refer to the Recall@1 metric.

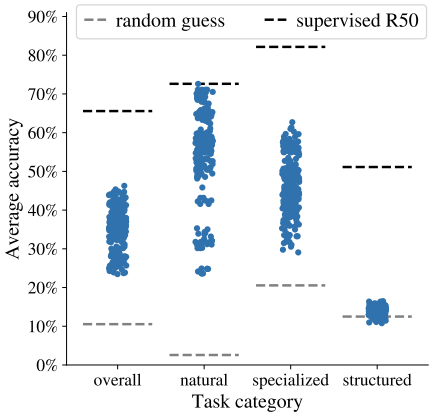

Figure 13. Performance of zero-shot classification models across different VTAB categories. Each dot is a zero-shot model evaluation.

	CLIP subset			Full		
	ImgNet	**T→I**	**I→T**	**ImgNet**	**T→I**	**I→T**
T5	58.9	14.5	22.6	62.4	19.6	34.3
+ *pt*	58.5	17.2	29.1	62.3	**20.1**	**34.5**
mT5	58.7	14.4	23.1	62.1	18.5	32.6
+ *pt*	58.4	15.6	25.1	**62.6**	18.9	33.6

Table 8. Training on the full YFCC100m data significantly improves all metrics compared to the CLIP subset. Gray rows are with text pre-training.

J. Multilingual details and limitations

Extra results. Table 8 shows the English zero-shot ImageNet classification performance of different English and multilingual T5 models, with LiT on $\text{YFCC}_{\text{CLIP}}$ vs. YFCC100m. We note that training on the larger, more diverse, multilingual set does not come at the expense of English performance.

Wiki-Image Text as an evaluation benchmark. We noted qualitatively that, as one may expect from Wikipedia, a large proportion of examples are about entities such as people, places, or art. When translated to other languages, proper nouns are usually kept as is - especially if the two languages share an alphabet. This makes it an imperfect dataset to benchmark multilinguism as monolingual models will score higher than they should.

Tokenization subtleties. The sentencepiece tokenizers, when faced with unknown vocabulary, will default to byte encoding. This is not a perfect catch-all; in such circumstances models cannot take advantage of pre-training, and the resultantly very long sequences will not fit in the 16-token maximum length used in this paper. It is nevertheless

[2]https://github.com/openai/CLIP/blob/main/data/prompts.md

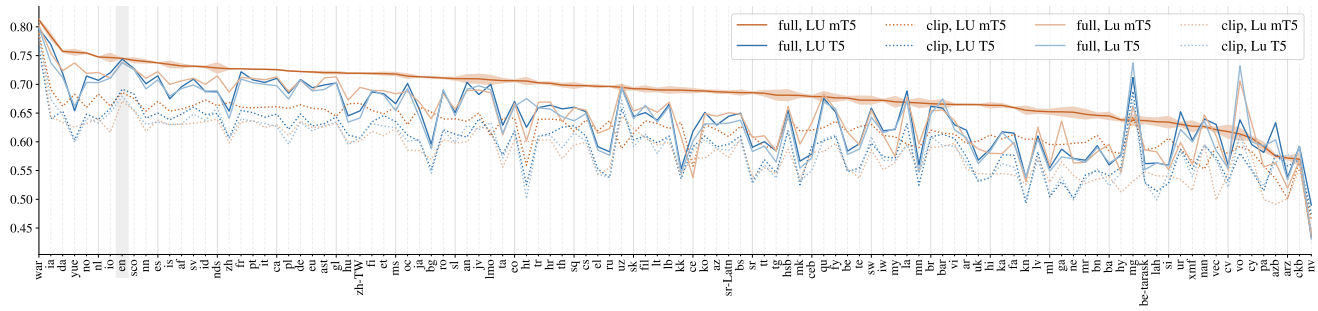

Figure 14. Fully detailed evaluation of the multilingual models on WIT.

better than the [UNK] tokens produced by BERT's WordPiece tokenizer; with SentencePiece, even with an imperfect vocabulary, the model has a chance to adapt. This explains why even with an ill-suited English-only vocabulary, the T5 models can still learn decent representations of non-English languages.

Translation of prompts. One obvious factor worth noting is that, in our setup, non-English languages may be impacted by imperfect translations. This likely means non-English performance is underestimated.

More subtly, we note that many languages - especially those with Latin alphabets - often use the English word for very niche or specific items. For example, at the time of writing, the Vietnamese translation of `I took a photo of an airship` contains the word `airship` verbatim. The contrastive model can in principle pick out the word `airship`, ignore all the Vietnamese, and retain decent performance despite not understanding Vietnamese at all.

Backtranslation as data augmentation. Backtranslation [50] - translating to a language and back again, in order to generate slightly different versions of a given text - is a common augmentation in NLP. We run some experiments to see whether it works for contrastive image-text training. We again use an online translation service to translate the texts in CC12M to and from 9 different languages. This probability is shared across the languages i.e. a backtranslation probability of 0.5 with 5 different backtranslate candidates means there is a 50% chance of picking the original ground truth and a 10% chance each of picking one of the backtranslated candidates. Figure 15 shows the effect of this augmentation on LiT using an AugReg ImageNet21k pre-trained ViT-B/16 model. Backtranslation is fairly useful up to certain point, with 10% giving a good trade-off which improves all metrics.

K. More de-duplication results

We present more ablation test results using larger architectures. We aim to check whether larger architectures benefit more from duplicates, while small architectures do not

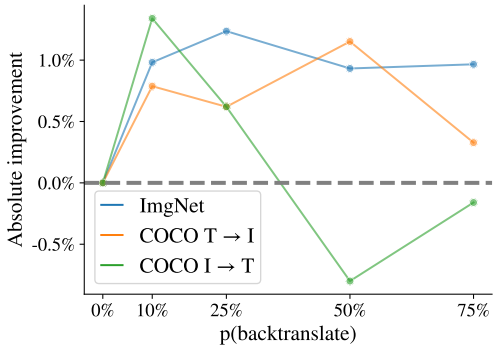

Figure 15. Backtranslating data as a form of data augmentation improves performance across most metrics.

Dedup	# up.	# down.	ImgNet	I→T	T→I
-	0	0	80.2	50.4	34.6
test	2.6M	76K	80.2	49.0	34.3
train+test	3.6M	220K	80.0	49.6	34.6

Table 9. Results on three different de-duplication setups, Lu setup with pre-trained ViT-L/16 image model.

have enough capacity to overfit to the duplications. More specifically, we adopt the `Lu` setup with a pre-trained ViT-L/16 image model [68], and from-scratch L size text model. Table 9 shows the experimental results. We find that the conclusions are consistent with the runs using the ViT-B/32 image model discussed in Section 5.5. This is further evidence suggesting that duplications are not the root cause for good zero-shot transfer results.

L. Image-text dataset comparison

Using simpler text filters for our dataset leads to a larger dataset size compared to the ALIGN dataset: The ALIGN dataset contains 1.8B image-text pairs, while our data set contains 3.6B image-text pairs.

In table 10, we show the results from training a baseline

Task	Pairs Seen	our	ALIGN	Diff.
ImageNet	900M	70.1	69.8	0.3
ImageNet	3.6B	72.0	71.5	0.5
ImageNet	7.2B	72.4	71.8	0.6
ImageNet	18B	72.9	72.2	0.7

Table 10. Comparing the ALIGN data with our data, which uses simpler text filters.

ViT-B/32 model on both datasets, with the same schedules. We vary the training schedule from 900M seen images, to 18B seen images. We use 18B images to make sure that the training process is long enough to benefit from a larger dataset. We find that the difference between the two datasets are small when the model is trained for a short period, i.e. less than a single epoch. As the training becomes longer, the impact of the dataset size becomes more visible.

Overall, the above results indicate that larger dataset with simpler filters slightly outperforms a smaller dataset with more filters. We leave the thorough exploration of this topic to future work.

M. Qualitative examples

Though strong classification & retrieval performance is promising, it arguably probes understanding of very simple concepts. Are LiT models really zero-shot learners capable of understanding open vocabularies?

We touch here on a few qualities these models should ideally have, but note that these are not to be considered representative; benchmarks that investigate more than simply fine grained visual classification should be used to more thoroughly understand these phenomena.

M.1. Private LiT model

In this section, we present model predictions with manually constructed image-text pairs input. Results from private LiT model are shown in Figure 16. We believe that with LiT, we successfully made a pre-trained image model to a zero-shot learner, that supports classification and retrieval with open vocabularies instead of a fixed label set.

M.2. Multi-lingual model

Thanks to LiT on the multilingual dataset, the model also supports inputs using different languages. In Figure 17, we show results both in Thai and Chinese. The model recognized the "Songkran" event in Thai, and the "Chinese Spring Festival" event in Chinese; it nonetheless also ranks English translations or transliterations quite highly, which is likely reflective of the data distribution. Multilingual capability makes our models more inclusive and accessible to non-English speakers.

M.3. Model failures

We present model failures in Figure 18. We show examples of how one can slightly change the text candidastes to manipulate the model output; one can easily force a desired answer by tuning other text candidates to rank lower.

(a) Nuanced context: The model can understand information such as actions or implied symptoms.

(b) Richer information: The model correctly handles colours, background buildings and even car brands.

(c) Counting: The model does a reasonable job at counting, though prompts like "bunch of cats" are preferred.

(d) Esoteric examples: The model has no problems at identifying rare concepts, like a cow on a beach, or an astronaut alien.

Figure 16. Various model predictions.

Figure 17. Training on multilingual data allows the model to recognise concepts in multiple languages, including visual concepts which do not directly exist in English.

Figure 18. Qualitative failures. In the left example, the model ranks the wrong grinning face before the ground truth yawning face. However, by removing the grinning face and adding emoji prompt, the model prefers emoji yawn.

Dataset	Prompts	Delta
dtd *v3.0.1*	`a CLASS texture`	+0.6%
flowers *v2.1.1*	`a CLASS flower`	+1.1%
flowers *v2.1.1*	`a CLASS plant`	+0.4%
pets *v3.2.0*	`a type of pet CLASS`	+1.0%
pets *v3.2.0*	`a CLASS texture`	+0.4%
pets *v3.2.0*	`CLASS , an animal`	+0.7%
svhn *v3.0.0*	`the number CLASS`	+3.0%
svhn *v3.0.0*	`a street sign with the number CLASS`	+2.8%
camelyon *v2.0.0*	`a histopathology slide showing CLASS`	+1.5%
camelyon *v2.0.0*	`histopathology image of CLASS`	+0.9%
eurosat *v2.0.0*	`a satellite photo of CLASS`	+3.2%
eurosat *v2.0.0*	`CLASS from above`	+2.4%
eurosat *v2.0.0*	`an aerial view of CLASS`	+3.3%
resisc *v3.0.0*	`a satellite photo of CLASS`	+3.4%
resisc *v3.0.0*	`CLASS from above`	+2.1%
resisc *v3.0.0*	`an aerial view of CLASS`	+4.7%
retino *v3.0.0*	`a retinal image with CLASS`	+9.7%
retino *v3.0.0*	`a retina with CLASS`	+6.3%
retino *v3.0.0*	`a fundus image with signs of CLASS`	+6.3%
clevr-count *v3.1.0*	`CLASS objects`	+0.1%
clevr-count *v3.1.0*	`CLASS things`	+0.2%
clevr-count *v3.1.0*	`a photo of CLASS objects`	+0.1%
dsprites-pos *v2.0.0*	`an object located CLASS`	+0.0%
dsprites-orient *v2.0.0*	`an object rotated at CLASS`	+0.1%
dsprites-orient *v2.0.0*	`something rotated at CLASS`	+0.0%
dsprites-orient *v2.0.0*	`CLASS rotation`	+0.0%
dsprites-orient *v2.0.0*	`something at a CLASS angle`	+0.0%
smallnorb-azmth *v2.0.0*	`an object rotated at CLASS`	+0.0%
smallnorb-azmth *v2.0.0*	`something rotated at CLASS`	+0.0%
smallnorb-azmth *v2.0.0*	`CLASS rotation`	+0.0%
smallnorb-azmth *v2.0.0*	`something at a CLASS angle`	+0.0%
smallnorb-elev *v2.0.0*	`an object rotated at CLASS`	+0.0%
smallnorb-elev *v2.0.0*	`something rotated at CLASS`	+0.0%
smallnorb-elev *v2.0.0*	`CLASS rotation`	+0.0%
smallnorb-elev *v2.0.0*	`something at a CLASS angle`	+0.0%

Table 11. Prompts swept over for VTAB tasks. Performance deltas are shown as mean test accuracy improvement per-task compared to just using the default three prompts. The default class names from TensorFlow Dataset (TFDS) are used in this table. TFDS versions are given alongside task names.

svhn *v3.0.0*

Prompts:	Class names:	Delta: +2.3%
• `the number CLASS`	1. *zero* 2. *one* 3. *two* 4. *three* 5. *four* 6. *five* 7. *six* 8. *seven* 9. *eight* 10. *nine*	
Prompts:	Class names:	Delta: +2.4%
• `a street sign with the number CLASS`	1. *zero* 2. *one* 3. *two* 4. *three* 5. *four* 6. *five* 7. *six* 8. *seven* 9. *eight* 10. *nine*	
Prompts:	Class names:	Delta: +3.2%
• `a photo of the number CLASS written on a sign` • `an outdoor house number CLASS` • `the number CLASS in the center of the image` • `an outdoor number CLASS written on a sign` • `an outdoor number CLASS` • `a centered image of the number CLASS`	1. *0 · zero* 2. *1 · one* 3. *2 · two* 4. *3 · three* 5. *4 · four* 6. *5 · five* 7. *6 · six* 8. *7 · seven* 9. *8 · eight* 10. *9 · nine*	

camelyon *v2.0.0*

Prompts:
- `a histopathology slide showing CLASS`

Class names:
1. *healthy lymph node tissue*
2. *a lymph node tumor*

Delta: +1.9%

Prompts:
- `histopathology image of CLASS`

Class names:
1. *healthy lymph node tissue*
2. *a lymph node tumor*

Delta: +1.9%

Prompts:
- `an example of CLASS`
- `a histopathology slide of CLASS`
- `an example histopathological image showing CLASS`
- `a histopathology slide showing CLASS`
- `patient's pathology examination indicates CLASS`
- `a CLASS slide`

Class names:
1. *healthy tissue* · *tissue*
2. *dangerous tissue* · *unhealthy tissue*

Delta: +0.8%

eurosat *v2.0.0*

Prompts:
- `an overhead view of CLASS`
- `an aerial view of CLASS`
- `an overhead image of CLASS`
- `a satellite photo of CLASS`
- `a satellite image of CLASS`
- `photo of CLASS from the sky`

Class names:
1. *farmland* · *farms* · *an annual crop*
2. *a forest* · *woodland* · *trees*
3. *a meadow* · *herbaceous vegetation* · *grass* · *fields*
4. *highway or road* · *motorways* · *highways* · *a street* · *roads*
5. *an urban area* · *an industrial area* · *an industrial zone* · *a city* · *factories*
6. *a pasture* · *farmland* · *farms*
7. *permanent crop* · *arable land* · *an orchard*
8. *a suburban area* · *a cul de sac* · *a residential area* · *houses*
9. *a canal* · *a river* · *a waterway* · *a stream*
10. *an ocean* · *a water* · *a sea* · *a reservoir*

Delta: +6.7%

resisc *v3.0.0*

Prompts:

- `a satellite image of CLASS`
- `an aerial view of CLASS`
- `a satellite photo of CLASS`
- `CLASS from above`

Class names:

1. *an airplane* · *a plane* · *a flying plane*
2. *an airfield* · *an airport* · *an aeroport*
3. *baseball diamond* · *baseball court* · *baseball* · *baseball field*
4. *basketball* · *a basketball court* · *an outdoor basketball court*
5. *beach* · *sand*
6. *a walkway* · *a bridge* · *a footbridge*
7. *shrubland* · *chaparral* · *sparse plants* · *desert plants* · *shrubs* · *dry plants*
8. *a church* · *a chapel*
9. *circular farmland* · *circle farm*
10. *cloudy sky* · *clouds* · *cloud*
11. *a commercial area* · *a shopping mall* · *high street* · *shops*
12. *densely populated area* · *lots of houses* · *a dense residential area* · *urban area*
13. *a desert* · *barren land* · *sand dunes* · *wasteland*
14. *woods* · *forest* · *woodland*
15. *expressway* · *roads* · *highway* · *freeway*
16. *golf fields* · *a golf course*
17. *a running court* · *a track court* · *a ground track field*
18. *a harbor* · *a dockyard* · *a haven* · *a jetty* · *a quay* · *a pier*
19. *an industrial zone* · *an industrial area* · *industry*
20. *a busy intersection* · *a crash on an intersection* · *intersection pileup* · *an intersection*
21. *an island in the ocean* · *an island* · *land surrounded by water* · *an ocean island*
22. *a reservoir* · *a lake* · *the ocean* · *the sea*
23. *a pasture* · *a paddock* · *fields* · *grassland*
24. *a medium residential area* · *cul de sac* · *suburban area* · *town*
25. *a mobile home park* · *caravans* · *caravan park*
26. *a mountain* · *a mountaintop* · *a hill* · *a mountain range*
27. *an overpass*
28. *a palace* · *a royal palace* · *a cheateau*
29. *parking* · *a parking lot*
30. *a train track* · *a train* · *a trainline* · *a rail track* · *a railway*
31. *a railway station* · *a train station*
32. *rectangular farmland* · *rectangle farms*
33. *a river* · *a stream*
34. *a roundabout*
35. *runway* · *an airport runway* · *a landing strip*
36. *an iceberg* · *ocean ice* · *sea ice*
37. *a ship* · *a boat*
38. *a snowberg*
39. *sparsely populated area*
40. *a stadium* · *an arena* · *a football stadium* · *a sports arena*
41. *a storage tank* · *tank*
42. *a tennis court* · *tennis* · *a court* · *a badminton court*
43. *rural land* · *a terrace*
44. *a power station* · *a thermal power station*
45. *a marsh* · *wetland* · *peatland* · *a bog*

Delta: +5.1%

clevr-closest *v3.1.0*

Prompts:	Class names:	Delta: +0.3%
• *CLASS* objects	1. *massive* 2. *very large* 3. *large* 4. 5. *small* 6. *very small*	

Prompts:	Class names:	Delta: +2.7%
• *CLASS* objects	1. *very nearby* 2. *nearby* 3. *near* 4. 5. *distant* 6. *very distant*	

Prompts:	Class names:	Delta: +0.6%
• *CLASS* shapes	1. *massive* 2. *very large* 3. *large* 4. 5. *small* 6. *very small*	

Prompts:	Class names:	Delta: +3.9%
• *CLASS* shapes	1. *very nearby* 2. *nearby* 3. *near* 4. 5. *distant* 6. *very distant*	

Prompts:	Class names:	Delta: +1.8%
• *CLASS* thing • the nearest shape in this image is *CLASS* • the closest shape in this rendered image is *CLASS* • the closest shape in this image is *CLASS*	1. *huge* · *super near* 2. *nearby* 3. *big* · *large* 4. *quite small* · *medium sized* · *normal sized* 5. *small* · *distant* 6. *very small* · *very distant*	

clevr-count *v3.1.0*		
Prompts: • `CLASS objects`	Class names: 1. *three* 2. *four* 3. *five* 4. *six* 5. *seven* 6. *eight* 7. *nine* 8. *ten*	Delta: +0.4%
Prompts: • `CLASS things`	Class names: 1. *three* 2. *four* 3. *five* 4. *six* 5. *seven* 6. *eight* 7. *nine* 8. *ten*	Delta: +0.6%
Prompts: • `a photo of CLASS objects`	Class names: 1. *three* 2. *four* 3. *five* 4. *six* 5. *seven* 6. *eight* 7. *nine* 8. *ten*	Delta: +0.7%
Prompts: • `a picture of CLASS` • `there are CLASS` • `there are CLASS in the image` • `a rendered image of CLASS`	Class names: 1. *3 objects* · *three objects* · *3 shapes* · *three shapes* 2. *4 objects* · *four objects* · *4 shapes* · *four shapes* 3. *5 objects* · *five objects* · *5 shapes* · *five shapes* 4. *6 objects* · *six objects* · *6 shapes* · *six shapes* 5. *7 objects* · *seven objects* · *7 shapes* · *seven shapes* 6. *8 objects* · *eight objects* · *8 shapes* · *eight shapes* 7. *9 objects* · *nine objects* · *9 shapes* · *nine shapes* 8. *10 objects* · *ten objects* · *10 shapes* · *ten shapes*	Delta: +1.2%

Table 12. Prompts and customized class names swept over for VTAB tasks. Performance deltas are shown as mean test accuracy improvement per-task compared to just using the default three prompts.

Ref	Dataset	Images	Cfg	H	Image	Text	Tok	Inits	Optim	LR	WD	INet	T→I	I→T	Vn	Vsp	Vst
Fig 1	$YFCC_{CLIP}$	983M	LU	y	vit-B/32	bert-base	WP	AR,Bert	Adam	8e-4	1e-4	63.6	22.1	37.6	59.3	35.0	12.7
Fig 1	$YFCC_{CLIP}$	983M	UU	y	vit-B/32	bert-base	WP	AR,Bert	Adam	3e-4	1e-5	53.3	23.4	37.6	54.9	44.4	14.1
Fig 1	$YFCC_{CLIP}$	983M	uu	y	vit-B/32	bert-base	WP	-,-	Adam	8e-4	1e-4	42.1	17.9	31.1	45.8	49.8	14.3
Tab 1	Ours	18.2B	Lu	n	vit-g/14*	vit-giant	SP	JFT,-	Adaf	1e-3	0	84.5	37.4	54.5	72.6	62.7	15.0
Tab 1	Mixed	983M	LU	y	vit-L/16	bert-large	WP	AR,Bert	Adaf	8e-4	1e-4	75.7	31.2	48.5	63.1	50.3	14.1
Tab 2	Ours	901M	Lu	n	vit-B/32	vit-base	SP	JFT,-	Adaf	1e-3	0	70.1	28.6	43.8	66.6	57.2	14.6
Tab 2	Ours	901M	Uu	y	vit-B/32	vit-base	SP	JFT,-	Adaf	1e-3	0	57.2	27.0	40.1	60.1	58.0	15.0
Tab 2	Ours	901M	uu	y	vit-B/32	vit-base	SP	-,-	Adaf	1e-3	0	50.6	24.1	38.9	55.3	38.9	16.5
Tab 3	$YFCC_{CLIP}$	246M	LU	y	dino-B/16	bert-base	WP	vit,Bert	Adam	8e-4	1e-4	55.5	18.2	33.4	51.5	45.4	14.8
Tab 3	$YFCC_{CLIP}$	246M	LU	y	mocov3-B/16	bert-base	WP	vit,Bert	Adam	8e-4	1e-4	55.4	17.6	33.5	50.8	40.5	12.8
Tab 6	CC12M	200M	LU	n	vit-B/32	bert-base	WP	AR,Bert	Adam	1e-3	1e-4	60.7	25.0	41.3	57.7	49.6	13.9
Tab 6	CC12M	200M	LU	n	bit-50x1	bert-base	WP	M,Bert	Adam	1e-3	1e-4	55.2	23.9	37.3	53.2	49.3	14.3
Tab 6	CC12M	200M	LU	n	mixer-B/32	bert-base	WP	AR,Bert	Adam	1e-3	1e-4	57.1	22.9	37.5	-	-	-
Tab 4	YFCC	901M	LU	y	vit-B/32	mt5-base	SP	AR,mt5	Adam	8e-4	1e-4	59.3	17.4	28.7	55.5	47.3	15.2
Tab 4	YFCC	901M	Lu	y	vit-B/32	vit-base	WP	AR,-	Adam	8e-4	1e-4	56.4	17.3	28.2	53.3	47.4	14.1
Tab 4	YFCC	901M	LU	y	vit-B/32	bert-base	WP	AR,Bert	Adam	8e-4	1e-4	59.5	20.7	36.3	56.7	51.3	12.3
Tab 4	YFCC	901M	Lu	y	vit-B/32	mt5-base	SP	AR,-	Adam	8e-4	1e-4	58.1	16.4	28.3	54.7	41.8	14.4
Tab 4	YFCC	901M	Lu	y	vit-B/32	bert-base	WP	AR,-	Adam	8e-4	1e-4	58.8	20.0	35.2	55.2	51.8	14.6
Tab 4	YFCC	901M	Lu	y	vit-B/32	vit-base	SP	AR,-	Adam	1e-3	1e-4	57.2	16.9	29.7	54.6	47.4	13.5
Tab 4	YFCC	901M	LU	y	vit-B/32	t5-base	SP	AR,t5	Adam	1e-3	1e-4	59.2	18.4	31.0	57.1	47.6	14.1
Tab 4	YFCC	901M	Lu	y	vit-B/32	t5-base	SP	AR,-	Adam	1e-3	1e-4	57.8	17.2	29.4	54.5	46.3	13.2
Fig 7	CC12M	200M	LU	n	vit-B/16	bert-large	WP	AR,Bert	Adaf	1e-3	1e-4	66.9	28.3	44.8	58.6	45.4	13.5
Fig 7	CC12M	200M	LU	n	vit-L/16	bert-large	WP	AR,Bert	Adaf	1e-3	1e-4	67.6	26.9	42.6	57.8	50.3	13.0
Fig 7	CC12M	200M	LU	n	vit-B/16	bert-base	WP	AR,Bert	Adam	1e-3	1e-4	66.1	28.2	45.3	59.0	50.6	14.0
Fig 7	CC12M	200M	LU	n	vit-L/16	bert-base	WP	AR,Bert	Adam	1e-3	1e-4	66.8	26.6	44.3	58.6	45.6	12.7
Fig 7	CC12M	200M	LU	n	vit-B/32	bert-large	WP	AR,Bert	Adaf	1e-3	1e-4	61.7	25.4	41.4	56.4	49.9	13.6
Fig 7	CC12M	200M	LU	n	vit-B/32	bert-base	WP	AR,Bert	Adam	1e-3	1e-4	61.1	24.9	40.9	56.8	49.6	15.4
Fig 7	Ours	901M	Lu	n	vit-g/14	vit-huge	SP	JFT,-	Adaf	1e-3	0	81.8	33.1	48.9	70.6	61.4	15.2
Fig 7	Ours	901M	Lu	n	vit-g/14	vit-large	SP	JFT,-	Adaf	1e-3	0	81.2	32.9	48.5	69.2	50.5	15.3
Fig 7	Ours	901M	Lu	n	vit-L/16	vit-huge	SP	JFT,-	Adaf	1e-3	0	80.8	35.6	51.2	69.2	50.3	13.5
Fig 7	Ours	901M	Lu	n	vit-L/16	vit-large	SP	JFT,-	Adaf	1e-3	0	80.3	34.8	49.8	68.9	60.3	14.8
Fig 7	Ours	901M	Lu	n	vit-g/14	vit-base	SP	JFT,-	Adaf	1e-3	0	79.5	30.7	45.9	68.6	59.6	12.6
Fig 7	Ours	901M	Lu	n	vit-B/16	vit-huge	SP	JFT,-	Adaf	1e-3	0	77.1	34.5	49.7	68.0	59.7	14.0
Fig 7	Ours	901M	Lu	n	vit-L/16	vit-base	SP	JFT,-	Adaf	1e-3	0	78.5	33.5	48.6	68.2	61.0	13.8
Fig 7	Ours	901M	Lu	n	vit-B/16	vit-large	SP	JFT,-	Adaf	1e-3	0	76.8	33.6	49.4	68.5	45.0	14.2
Fig 7	Ours	901M	Lu	n	vit-B/16	vit-base	SP	JFT,-	Adaf	1e-3	0	75.2	31.9	46.8	67.5	57.7	12.8
Fig 7	Ours	901M	Lu	n	vit-B/32	vit-huge	SP	JFT,-	Adaf	1e-3	0	72.2	31.2	46.4	68.3	55.1	13.8
Fig 7	Ours	901M	Lu	n	vit-B/32	vit-large	SP	JFT,-	Adaf	1e-3	0	71.6	30.7	45.6	66.4	55.0	14.5
Fig 7	Ours	901M	Lu	n	vit-B/32	vit-base	SP	JFT,-	Adaf	1e-3	0	70.0	29.2	43.8	65.8	56.9	12.0
Fig 5	$YFCC_{CLIP}$	983M	LU	y	vit-B/32	mt5-base	SP	AR,mt5	Adam	3e-4	1e-4	58.4	15.6	25.1	54.5	36.7	12.3
Fig 5	$YFCC_{CLIP}$	983M	LU	y	vit-B/32	t5-base	SP	AR,t5	Adam	3e-4	1e-4	58.5	17.2	29.1	54.7	40.4	13.6
Fig 5	$YFCC_{CLIP}$	983M	Lu	y	vit-B/32	mt5-base	SP	AR,-	Adam	1e-3	1e-5	58.7	14.4	23.1	53.1	41.3	14.7
Fig 5	$YFCC_{CLIP}$	983M	Lu	y	vit-B/32	t5-base	SP	AR,-	Adam	8e-4	1e-4	58.9	14.5	22.6	53.1	41.6	15.0
Fig 5	YFCC	983M	LU	y	vit-B/32	mt5-base	SP	AR,mt5	Adam	8e-4	1e-4	62.6	18.9	33.6	59.0	47.6	13.8
Fig 5	YFCC	983M	Lu	y	vit-B/32	mt5-base	SP	AR,-	Adam	8e-4	1e-4	62.1	18.5	32.6	58.7	50.0	14.8
Fig 5	YFCC	983M	Lu	y	vit-B/32	t5-base	SP	AR,-	Adam	1e-3	1e-4	62.4	19.6	34.3	60.8	31.5	14.8
Fig 5	YFCC	983M	LU	y	vit-B/32	t5-base	SP	AR,t5	Adam	1e-3	1e-4	62.3	20.1	34.5	61.1	50.3	14.6

Table 13. Detailed configuration and metrics for a selection of models. *Ref* describes the Figure/Table where the model is mentioned. *Dataset* describes the dataset that was used (see Section 4), with "Mixed" referring to alternating batches between CC12M and YFCC100m. *Images* is the number of images seen during contrastive-tuning. Default batch size was 16 384 (only exception model "g/14*" with 32 768). *Cfg* first letter refers to image tower, second letter to text tower (Section 5.2). *H* describes whether a linear head was added to the image tower (note that the text tower always has a linear head). *Image* describes the image tower (all models use 224px input resolution apart from "g/14*" that uses 288px), for details on models see [4, 10, 20, 32, 60, 68]. *Text* describes the text tower, for details see [16, 20, 46, 66]. *Tok* describes whether a SentencePiece or WordPiece tokenizer was used. *Inits* describes the initializations of the image/text towers (AR refers to AugReg "recommended checkpoints" [54]). *Optim* is the optimizer, using default Adam or Adafactor [52]. *LR* is the base learning rate (with linear ramp-up and cosine decay). *WD* is the weight decay (using "decoupled" weight decay [39]). *INet* describes zero-shot top-1 accuracy on Imagenet. *T→I* and *I→T* describe retrieval recall @1 on the MSCOCO test set. *Vn*, *Vsp*, *Vst* VTAB [69] results for "natural", "specialized", and "structured" subsets.