Making Text-Image Connection Formal and Practical

Carlos-Gustavo Salas-Flores, Dongmian Zou, and Luyao Zhang [1]

Abstract

Text and image feature extraction is at the core of several state-of-the-art artificial intelligence algorithms, including DALLE-2, Stable Diffusion, and Segment Anything. However, models that connect images and texts are usually trained using hundreds of GPUs and tens or even hundreds of millions of data points, making it infeasible for most agents to perform the training from scratch. Furthermore, these groundbreaking works necessitate more formally defined algorithms to enable easier adoption and implementation. To address these issues, this paper elaborates on a formal and intuitive algorithm for text-image connections and proposes an alternative to train CLIP, a neural network model that learns joint representations from text and images, on low computing resources. In our experimentation, two models were trained on 85% of WKIT, a dataset of text-image pairs, by making use of mixed precision in back-propagation and shrinking the input images' resolution and the query's maximum length relative to the original CLIP in a setting constrained to a single GPU. Our results show that it is not only feasible to train image-text connection models from scratch in this constrained setting but also that reducing the input image resolution image results in better accuracy for zero-shot classification.

1. Introduction

CLIP (Contrastive Language-Image Pretraining), a neural network that efficiently learns visual concepts from natural language supervision, has gained significant attention recently. It is at the core of not only image synthesis algorithms such as DALLE-2 (Ramesh et al., 2022) and Stable Diffusion (Rombach et al., 2022), but also segmentation algorithms such as Segment Anything (Kirillov et al., 2023). Furthermore, its general principle has also been adapted to enable audio-text connections (Huang et al., 2022) for feature extraction in music generation (Agostinelli et al., 2023). Therefore, it works as a multi-purpose deep learning model that can be used for zero-shot predictions, geo-localization, object detection, and a feature extractor to feed generative models (Ramesh et al., 2022; Nichol et al., 2022).

Despite the effectiveness of CLIP, its training requires a huge amount of computational resources. Most practitioners need to rely on pre-trained models in their own applications. This limits potential improvements of these models. To this end, we propose a lightweight alternative to training CLIP, with the aim to enable a broader range of researchers and AI practitioners to implement CLIP for their own purposes.

The remaining sections of the paper are arranged as follows. Our motivation is elaborated in Section 2. The algorithm and pseudo-code can be found in Section 3. Our experimental setup is described in Section 4, immediately followed by an analysis of the results in Section 5. Section 6 presents our conclusion. And our source code is available at `github.com/SciEcon/CLIP_implementation`.

2. Motivation

Research Question 1: How does CLIP operate and what are its underlying mechanisms? After more than 250 artificial intelligence papers on arXiv with CLIP in the title, as of today, there is no formal algorithm for implementing CLIP. Some build upon improvements in training and efficiency (Li et al., 2023; Sun et al., 2023). However, just a handful of them describes the model in mathematical language or using diagrams (Chefer et al., 2022; Crowson et al., 2022; Li et al., 2022; Singha et al., 2023; Wang et al., 2022; Zhu et al., 2023) and only one describes a formal algorithm for the training stage (Song et al., 2022). This work is meant to fill this gap by providing the formal algorithm to implement Text-Image connections.

Research Question 2: Can state-of-the-art models be effectively trained from scratch using limited computational resources? As with many problems in engineering, there is a trade-off between memory use and speed.

[1]Data Science Research Center, Duke Kunshan University, Suzhou, Jiangsu, China. Correspondence to: Dongmian Zou and Luyao Zhang <dongmian.zou@dukekunshan.edu.cn, luyao.zhang@dukekunshan.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

CLIP's original implementation focuses on memory saving by using gradient checkpointing (Chen et al., 2016) and half-precision statistics (Dhariwal et al., 2020). However, these approaches slow down the training and reduce the precision in backpropagation. We tackle this problem by using a smaller image resolution and token sequence size, and mixed-precision (Micikevicius et al., 2018).

Research Question 3: Can CLIP be trained effectively with a reduced amount of data, or is a large-scale dataset necessary for its training? By using a diverse enough dataset, it might be possible to achieve significant results with just a fraction of the data. We created WKIT (Salas-Flores et al., 2023), a new dataset of 24 million text-image pairs specifically made for tasks similar to Radford et al. (2019).

3. Methodology

To carry out Text-Image connections, CLIP makes use of three components: a Vision Transformer (ViT) as an image encoder (Dosovitskiy et al., 2021), a Generative Pre-Training (GPT) as a text encoder (Vaswani et al., 2017; Liu et al., 2018; Radford et al., 2018; OpenAI, 2023), and a multi-modal embedding (Radford et al., 2021). As proposed by Phuong and Hutter (2022), the model is discussed in formal algorithm form having Multihead Self-Attention (MSA), and Layer Normalization (LN) as starting building blocks.

3.1. Tokenization

CLIP receives text and images as inputs. Those texts are queries describing the images. Similar to GPT2 and GPT3, CLIP uses Byte-Pair Encoding (BPE), a sub-word tokenizer that's more flexible as it's able to handle previously unseen words (Sennrich et al., 2015; Radford et al., 2019; Brown et al., 2020; Radford et al., 2021).

3.2. Transformer Layers

The unit block of both encoders is the Transformer Layer $g(f(\cdot))$. It consists of an MSA block f and a Multilayer Perceptron (MLP) g, Layer Normalization (LN) is applied before going through MSA and MLP. Specifically, the MSA and MLP are given by

$$f(\mathbf{x}|\text{Mask}) = \text{MSA}(\text{LN}(\mathbf{x})|\text{Mask}) + \mathbf{x}, \quad (1)$$

$$g(\mathbf{x}) = \text{GELU}(\mathbf{x}\mathbf{W}_h + \mathbf{b}_h)\mathbf{W}_o + \mathbf{b}_o + \mathbf{x}. \quad (2)$$

This recursive call is shared across both kinds of layers, with the only difference on the mask. The image encoder uses a Transformer Encoder-only architecture (Devlin et al., 2019), while the text encoder uses a Decoder-only architecture (Radford et al., 2019) that forces each word to ignore all words coming after in the sentence. This is currently the state-of-the-art in text generation (Touvron et al., 2023; OpenAI, 2023; Anil et al., 2023).

Algorithm 1 Vision Transformer (ViT)

Input: $I \in R^{C\times H\times W}$, an image.
$\mathbf{x_p} \leftarrow f_{\text{r}}(I)$
$\mathbf{x} \leftarrow [\mathbf{x}_{\text{class}}; E\mathbf{x_p}^1; ...; E\mathbf{x_p}^N] + E_{\text{pos}}$
for l = 1, ..., L **do**
  $\mathbf{x} \leftarrow g(f(\mathbf{x}|\text{Mask}_{\text{BID}}))$
end for
$\mathbf{y} \leftarrow g(\mathbf{x}_{\text{class}})$
Return: $\mathbf{y}$

3.3. Image Encoder

The image encoder is a Vision Transformer (ViT) and it follows the same implementation as the original work (Dosovitskiy et al., 2021). An image $I \in \mathbb{R}^{C\times H\times W}$ comes as input, and is transformed into a sequence of $N = WH/P^2$ non-overlapping flattened patches $\mathbf{x_p} \in \mathbb{R}^{N\times P^2C}$ of resolution (P, P), we call this transformation f_r; in addition, a `[class]` token $\mathbf{x}_{\text{class}}$ is prepended a the beginning of this sequence resulting in an effective sentence length size of $\ell_{\text{image}} = N + 1$. This sequence passes through L layers of Transformer Encoders with a bidirectional mask: $\text{Mask}_{\text{BID}} = \mathbf{1}$. Finally, only the output corresponding to the class token is considered when extracting the features, not before passing through a bilayer perceptron. We summarize the ViT in Algorithm 1.

3.4. Text Encoder

The text encoder is a Transformer Decoder-only architecture, similar to that used in GPT models (Radford et al., 2019; Brown et al., 2020). The input is a sequence of token IDs. Each sequence is enclosed by `[SOS]` and `[EOS]` tokens; zero padding and truncation are applied accordingly to guarantee a sequence of size ℓ_{text}. Each token $T[t]$ has a corresponding embedding $E_{\text{tkn}}[:, T[t]]$ as well as a position embedding $E_{\text{poss}}[:, t]$, these are added together and fed to the GPT encoder. It also takes a unidirectional mask: $\text{Mask}_{\text{UND}}[t', t''] = \mathbf{1}[t' \leq t'']$. This allows each word to ignore all words coming after. The GPT model takes a fixed sequence of words ℓ_t and passes them through L_t layers of Transformer Decoders. The output corresponding to the `[EOS]` token is considered as the feature representation of the text after applying Layer Normalization. We describe the GPT model in Algorithm 2.

3.5. Multi-modal Embedding

The last module is the core of CLIP, it projects the text and image features onto the same two-dimensional space. This

Algorithm 2 Generative Pre-Training (GPT)

Input: T, a sequence of token IDs s.t. $T[\mathrm{t}] \in V$.
$\forall t, \mathbf{x} \leftarrow E_{\text{tkn}}[:, T[t]] + E_{\text{pos}}[:, t]$
for $l = 1, ..., L$ **do**
  $\mathbf{x} \leftarrow g(f(\mathbf{x}|\text{Mask}_{\text{UND}}))$
end for
$\mathbf{y} \leftarrow \text{LN}(\mathbf{x}_{\text{eos}})$
Return: $\mathbf{y}$

Algorithm 3 CLIP

Input: $\mathbf{I} \in R^{N \times C \times H \times W}$ a batch of N images.
Input: $\mathbf{T}$, a batch of N sequences of token IDs.
$\forall n, I^n_{fe} \leftarrow \text{ViT}(I^n)E_i$
$\forall n, T^n_{fe} \leftarrow \text{GPT}(T^n)E_t$
$\forall i \forall j, Z^{\text{img}}_{i,j} \leftarrow S_C(I^i_{fe}, T^j_{fe})$
$Z^{\text{text}} \leftarrow (Z^{\text{img}})^T$
Return: $Z_{\text{img}}, Z_{\text{text}}$

module is dependent on the batch size as it connects N image and query features and computes their cosine similarities. First, for an arbitrary image I', the image encoder returns the features $\text{ViT}(I') = I'_f$ as output, similarly, the text encoder returns the features $\text{GPT}(T') = T'_f$ for an arbitrary query text T'. Since these features have different corresponding dimensions, they are then projected onto the same space $\mathbb{R}^{d_{\text{mme}}}$ using embeddings E_i and E_t for images and text features accordingly. Lastly, we compute their cosine similarities as

$$S_C(I'_{fe}, T'_{fe}) = \frac{I'_{fe} \cdot T'_{fe}}{\|I'_{fe}\|_2 \|T'_{fe}\|_2}. \tag{3}$$

3.6. Optimization

The objective of the loss function is to maximize the scaled cosine similarities across the main diagonal $Z^{\text{img}}_{1,1}, ..., Z^{\text{img}}_{N,N}$, this will be pulling together all similar pairs, while simultaneously pushing apart those that are different. We describe the loss function as follows. First, an image-to-text loss for a single pair is calculated as

$$\ell_n^{(\text{I}\to\text{T})}(Z^{\text{img}}) = -\log \frac{\exp(Z^{\text{img}}_{n,n}/\tau)}{\sum_{k=1}^{N} \exp(Z^{\text{img}}_{n,k}/\tau)}. \tag{4}$$

Then, the text-to-image loss for a single pair follows the same form with $Z^{\text{text}} = (Z^{\text{img}})^T$ as the input. That is,

$$\ell_n^{(\text{T}\to\text{I})}(Z^{\text{text}}) = -\log \frac{\exp(Z^{\text{text}}_{n,n}/\tau)}{\sum_{k=1}^{N} \exp(Z^{\text{text}}_{n,k}/\tau)}. \tag{5}$$

We remark that both equations (4) and (5) take a learnable temperature parameter τ. Finally, the average loss across all N pairs is calculated as follows (Zhang et al., 2022):

$$\mathcal{L}(Z^{\text{img}}, Z^{\text{text}}; \theta) = \frac{1}{2}\text{E}\left[\ell_n^{(\text{I}\to\text{T})}(Z^{\text{img}}) + \ell_n^{(\text{T}\to\text{I})}(Z^{\text{text}})\right]. \tag{6}$$

3.7. Low-Resource From-Scratch Training

The original setup of CLIP made it seem impossible to run any model on a single GPU. However, three main steps were taken to accelerate training and improve accuracy.

Shrinked Inputs. One of the first steps that quickly reduced the model's size was to shrink the size of the input image and the query's maximum length as previous work showed its effectiveness (Li et al., 2023).

Eliminate Checkpointing. Radford et al. (2023) use checkpointing (Chen et al., 2016) to save memory, however, in our experiments, such savings in memory did not compensate for the longer training time so we decided to proceed without it to accelerate the training stage.

Mixed-precision. Different from the original paper, instead of using mixed-precision for the forward and backward stages, we only implement it in the feed-forward stage to improve the precision in calculating the weights (Radford et al., 2021).

4. Experimental Setup

4.1. Data

Prior to CLIP, existing datasets such as COCO (Lin et al., 2015) and Visual Genome (Krishna et al., 2016) were too small for the given task and YFCC100M (Thomee et al., 2016) did not meet OpenAI's requirements. Later, Shrinavitsan et al. (2021) introduced WIT, which closely resembles WQI and includes multiple languages but it falls short in terms of image samples with only 11.5 million unique images.

Therefore, we created a new training dataset specifically for this task consisting of 24 million images named WKIT (Salas-Flores et al., 2023). Similar to the original work, 14M Wikipedia English pages were scrapped to collect all words occurring more than 100 times resulting in $\sim$ 50,000 labels. From those, at most 10,0000 images per label were scrapped from `unsplash.com`. We used approximately 85% of the data ($\sim$ 21 million images) to train the following two models.

4.2. Model Implementation

Two variants of this model were implemented from scratch following the formal algorithm described in Algorithm 3.

Both models are essentially the same, but the large one (ViT-B/32@224) takes a 224×224 image with a patch of

resolution $P = 32$, while the small one (ViT-B/16@112) takes a 128×128 one with a patch of resolution $P = 16$. The first is the smallest model used in (Radford et al., 2021) while the second one, with the reduced input resolution, results in ∼80% the size (in terms of the number of trainable parameters).

The GPT encoder consists of 12 layers, 8 heads, a 2048 feed-forward size, and a 512 width while the ViT encoder uses 12 layers, 12 heads, a 3072 feed-forward size, and a 768 width. We recall that in both cases the maximum length of the query was shrunk from 76 to 32 tokens each time as well.

4.3. Training

The training was carried out on a virtual machine using a single RTX TITAN X. Both models were trained on a batch of 128 pairs over 4 epochs. Optimization was performed using Adam Optimizer with weight decay $\beta_1 = 0.9$ and $\beta_2 = 0.99$ and warmup iterations for 2,000 steps up to a learning rate of 5×10^{-4}. The learnable temperature value τ was initialized at 0.07 and clipped at a minimum of 0.01 and a maximum of 100.

5. Results and Analysis

Figure 1 shows the loss of training in both approaches. We have the following two observations. Firstly, both models show very similar behavior throughout the training process, this could be because they use the same architecture and the input dimension is not affecting the learning process. Secondly, the variance for the training loss shows to be very high towards the end which might imply a higher bias with each step towards the last part of the training. For a more elaborate discussion refer to Appendixes A, B, C, D, and E.

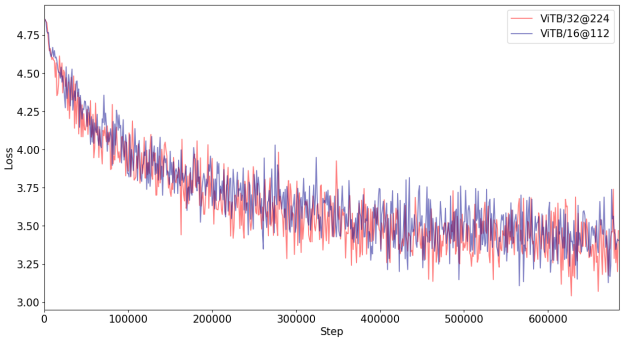

Figure 1. Training Loss Comparison for CLIP Models with ViT-B/32@224 and ViT-B/16@112 Architectures. This figure showcases the training loss comparison between ViT-B/32@224 and ViT-B/16@112 architectures. The x-axis represents the training iterations (steps), while the y-axis represents the loss.

Table 1. CLIP Zero-shot Classification Accuracy (%) Assessment by model.

DATASET	B/32@224	B/16@112	RANDOM
CIFAR-10	17.89	19.45	∼ 10.0
CALTECH101	2.21	2.58	∼ 1.0
IMAGENET	0.16	0.20	∼ 0.1

5.1. Zero-shot classification

Now, when it comes to zero-shot classification we find a few limitations, as noted in Table 1, the largest model is still far from achieving at least a 50% accuracy in two standard datasets such as CIFAR-10, ImageNet, and Caltech. Yet, it does confirm that such results are not just random. Thus, reasserting that it has learned to extract key features of text and images.

Nevertheless, to our surprise, the smaller model shows better results in performing zero-shot classification on all tested datasets. We made sure to run these tests several times to ensure the reliability of our results. We can observe about a 1.5 % difference in CIFAR-10, almost 0.4% in Caltech101, and a 0.04% difference in ImageNet. For more details on zero-shoot classification see Appendix B.

6. Conclusion and Future Research

This paper shows the feasibility of training a state-of-the-art text-image connections algorithm from scratch by using very limited computational resources at the training stage. It also carefully shows a full description in the formal algorithmic form of CLIP which we plan to later modify, scale, fine-tune, and so forth for more specific or general purposes according to the needs of the situation.

We demonstrate that reducing the input size does not necessarily affect the performance of the model. If we adjust the patch size accordingly, it can actually improve its performance. We further show that the results obtained have been able to identify different elements in an image which, though has its limitations for classification tasks, may be useful in feature extraction for generative models. Further research may shed light on this task.

In conclusion, our findings show that while employing multiple GPUs can provide significant advantages, they are not indispensable. The primary motivation for utilizing these extensive computational resources is to handle vast amounts of data in a very short time. However, we have demonstrated that by reducing the size of the models, downsampling the input image, and selecting a sufficiently diverse, yet relatively small dataset, it remains feasible to train CLIP from scratch on a single RTX TITAN X.

References

Agostinelli, A., Denk, T. I., Borsos, Z., Engel, J., Verzetti, M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., Sharifi, M., Zeghidour, N., and Frank, C. Musiclm: Generating music from text, 2023.

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., Chu, E., Clark, J. H., Shafey, L. E., Huang, Y., Meier-Hellstern, K., Mishra, G., Moreira, E., Omernick, M., Robinson, K., Ruder, S., Tay, Y., Xiao, K., Xu, Y., Zhang, Y., Abrego, G. H., Ahn, J., Austin, J., Barham, P., Botha, J., Bradbury, J., Brahma, S., Brooks, K., Catasta, M., Cheng, Y., Cherry, C., Choquette-Choo, C. A., Chowdhery, A., Crepy, C., Dave, S., Dehghani, M., Dev, S., Devlin, J., Díaz, M., Du, N., Dyer, E., Feinberg, V., Feng, F., Fienber, V., Freitag, M., Garcia, X., Gehrmann, S., Gonzalez, L., Gur-Ari, G., Hand, S., Hashemi, H., Hou, L., Howland, J., Hu, A., Hui, J., Hurwitz, J., Isard, M., Ittycheriah, A., Jagielski, M., Jia, W., Kenealy, K., Krikun, M., Kudugunta, S., Lan, C., Lee, K., Lee, B., Li, E., Li, M., Li, W., Li, Y., Li, J., Lim, H., Lin, H., Liu, Z., Liu, F., Maggioni, M., Mahendru, A., Maynez, J., Misra, V., Moussalem, M., Nado, Z., Nham, J., Ni, E., Nystrom, A., Parrish, A., Pellat, M., Polacek, M., Polozov, A., Pope, R., Qiao, S., Reif, E., Richter, B., Riley, P., Ros, A. C., Roy, A., Saeta, B., Samuel, R., Shelby, R., Slone, A., Smilkov, D., So, D. R., Sohn, D., Tokumine, S., Valter, D., Vasudevan, V., Vodrahalli, K., Wang, X., Wang, P., Wang, Z., Wang, T., Wieting, J., Wu, Y., Xu, K., Xu, Y., Xue, L., Yin, P., Yu, J., Zhang, Q., Zheng, S., Zheng, C., Zhou, W., Zhou, D., Petrov, S., and Wu, Y. Palm 2 technical report, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chefer, H., Benaim, S., Paiss, R., and Wolf, L. Image-based clip-guided essence transfer, 2022.

Chen, T., Xu, B., Zhang, C., and Guestrin, C. Training deep nets with sublinear memory cost, 2016.

Crowson, K., Biderman, S., Kornis, D., Stander, D., Hallahan, E., Castricato, L., and Raff, E. Vqgan-clip: Open domain image generation and editing with natural language guidance, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I. Jukebox: A generative model for music, 2020.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Goh, G., Cammarata, N., Voss, C., Carter, S., Petrov, M., Schubert, L., Radford, A., and Olah, C. Multimodal neurons in artificial neural networks. *Distill*, 6(3):e30, 2021.

Huang, Q., Jansen, A., Lee, J., Ganti, R., Li, J. Y., and Ellis, D. P. W. Mulan: A joint embedding of music audio and natural language, 2022.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., Bernstein, M. S., and Li, F.-F. Visual genome: Connecting language and vision using crowdsourced dense image annotations, 2016.

Li, J., Shakhnarovich, G., and Yeh, R. A. Adapting clip for phrase localization without further training, 2022.

Li, R., Kim, D., Bhanu, B., and Kuo, W. Reclip: Resource-efficient clip by training with small images, 2023.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. Microsoft coco: Common objects in context, 2015.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. Generating wikipedia by summarizing long sequences, 2018.

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., and Wu, H. Mixed precision training, 2018.

Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. Glide: Towards photorealistic image generation and editing with text-guided diffusion models, 2022.

OpenAI. Gpt-4 technical report, 2023.

Phuong, M. and Hutter, M. Formal algorithms for transformers, 2022.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision, 2021.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models, 2022.

Salas-Flores, C.-G., Zou, D., and Zhang, L. Replication Data for: Making Text-Image Connection Formal and Practical, 2023. URL https://doi.org/10.7910/DVN/PKEGOX.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Singha, M., Jha, A., Solanki, B., Bose, S., and Banerjee, B. Applenet: Visual attention parameterized prompt learning for few-shot remote sensing image generalization using clip, 2023.

Song, H., Dong, L., Zhang, W.-N., Liu, T., and Wei, F. Clip models are few-shot learners: Empirical studies on vqa and visual entailment, 2022.

Srinivasan, K., Raman, K., Chen, J., Bendersky, M., and Najork, M. WIT: Wikipedia-based image text dataset for multimodal multilingual machine learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, jul 2021. doi: 10.1145/3404835.3463257. URL https://doi.org/10.1145%2F3404835.3463257.

Sun, Q., Fang, Y., Wu, L., Wang, X., and Cao, Y. Eva-clip: Improved training techniques for clip at scale, 2023.

Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. YFCC100m. *Communications of the ACM*, 59(2):64–73, jan 2016. doi: 10.1145/2812802. URL https://doi.org/10.1145%2F2812802.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2017.

Wang, Z., Liu, W., He, Q., Wu, X., and Yi, Z. Clip-gen: Language-free training of a text-to-image generator with clip, 2022.

Zhang, Y., Jiang, H., Miura, Y., Manning, C. D., and Langlotz, C. P. Contrastive learning of medical visual representations from paired images and text. In *Machine Learning for Healthcare Conference*, pp. 2–25. PMLR, 2022.

Zhu, X., Zhang, R., He, B., Zhou, A., Wang, D., Zhao, B., and Gao, P. Not all features matter: Enhancing few-shot clip with adaptive prior refinement, 2023.

A. Sample Text-Image Connections Analysis

Figure 2 reports the cosine similarities from 7 (out of sample) text-image pairs. While not perfect, it shows very reasonable results and a capability of generalizing certain concepts. From this sample, we can appreciate that contrasting concepts are being pulled apart, for instance, the largest model is able to predict that the concept of ”a red toy truck” is very different from that of ”a gray sky” or ”a glass of water”. Similarly, it also learned that an image of ”a blue motorcycle” is very different in concept from that of ”a plant”.

However, the model still makes some misconceptions, for instance, it wrongly connects the concept of ”a glass of water” as more similar to an image of ”a remote and a brush” than the actual image. Nonetheless, it is notable to outstanding that the results of this demonstration are similar to those of the original work.

Figure 2. Sample text-Image cosine similarities. Test carried out on ViT-B/16 @ 224px.

B. Zero-shot Classification Accuracy Across Training Stages

Figures 3, 4 and 5 show that the smallest model is consistently better at performing zero-shot classifications on these three different datasets. Additionally, it shows that the smallest model decreases in performance in at least two datasets, namely, CIFAR-10 and Caltech101. This possibility is further elaborated in Appendix C.

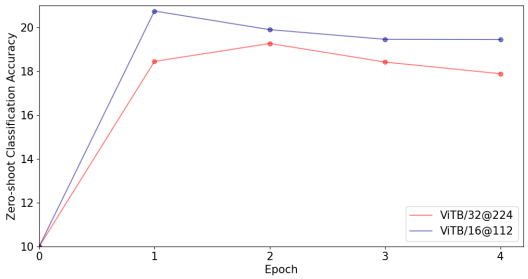

Figure 3. Zero-shot Classification Accuracy on CIFAR-10. The red line represents ViTB/32@224 while the blue one represents ViTB/32@112.

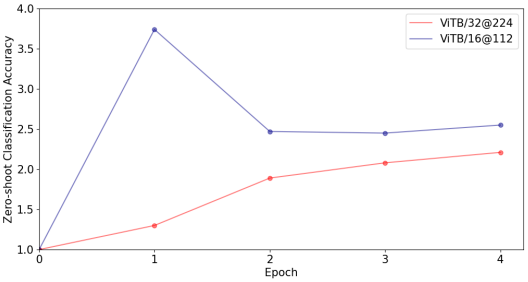

Figure 4. Zero-shot Classification Accuracy on Caltech101. The red line represents ViTB/32@224 while the blue one represents ViTB/32@112.

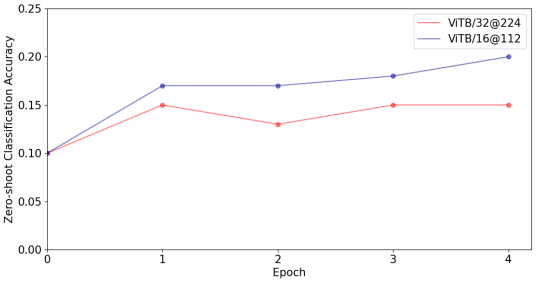

Figure 5. Zero-shot Classification Accuracy on ImageNet. The red line represents ViTB/32@224 while the blue one represents ViTB/32@112.

C. Cosine similarities analysis across stages of training

Results in Figures 6 and 7 show that while the models become maybe less accurate across time, they increase their confidence in contrasting different concepts. We observe, for instance, that the range of values for cosine similarities is smaller in figure 6 than in figure 7.

Figure 6. Sample cosine similarities for ViTB@224 after one epoch.

Figure 7. Sample cosine similarities for ViTB@224 after 3 epochs.

D. Typographic Attacks

In their paper, Goh et al. (2021) show that CLIP's zero-shot classification capabilities are prone to attacks that trick the algorithm to think photos with written text are things that are not. We also use this opportunity to analyze the cosine similarities between related and unrelated concepts.

Figure 8 show that all typographically attacked images result in non-sense results, what is worth noticing is that there is no caption that turns off these images, it seems to have heavily flawed the images just as previous works show (Goh et al., 2021).

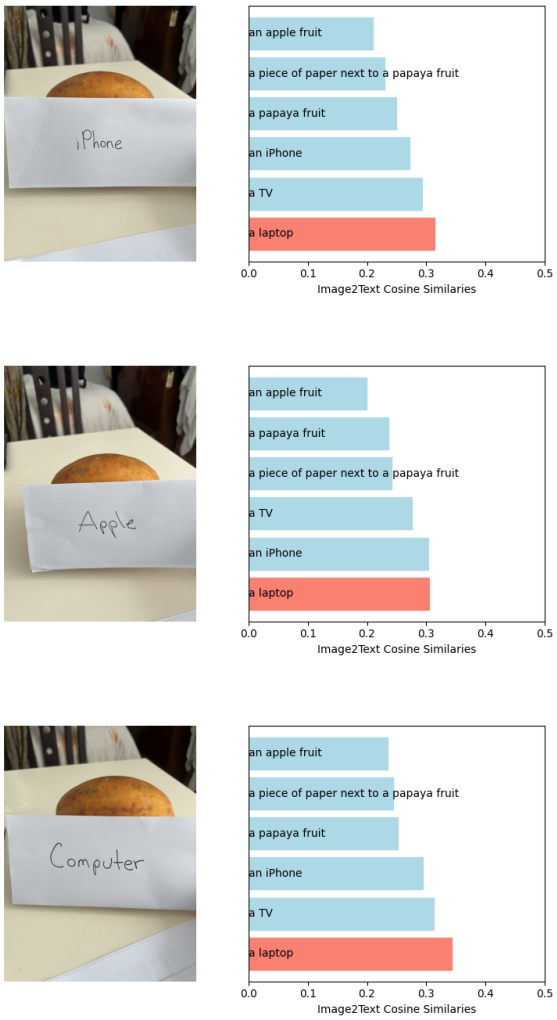

Figure 8. Images analysed by ViTB/32@224px under typographic attack.

E. Assesement on Cosine Similarities for Classification tasks

We recognize some bias on very popular objects which are recognized more easily or not easily fooled. When different images share similar components such as a display as in the case of a computer, an iPhone, and a TV, the model classifies them as more similar to each other. However, it's still worth noticing that it can make some misjudgments e.g. in figure 9 in Appendix D it concludes that "a papaya fruit" and "an apple fruit" are closer in concept to the image of an iPhone than "a TV".

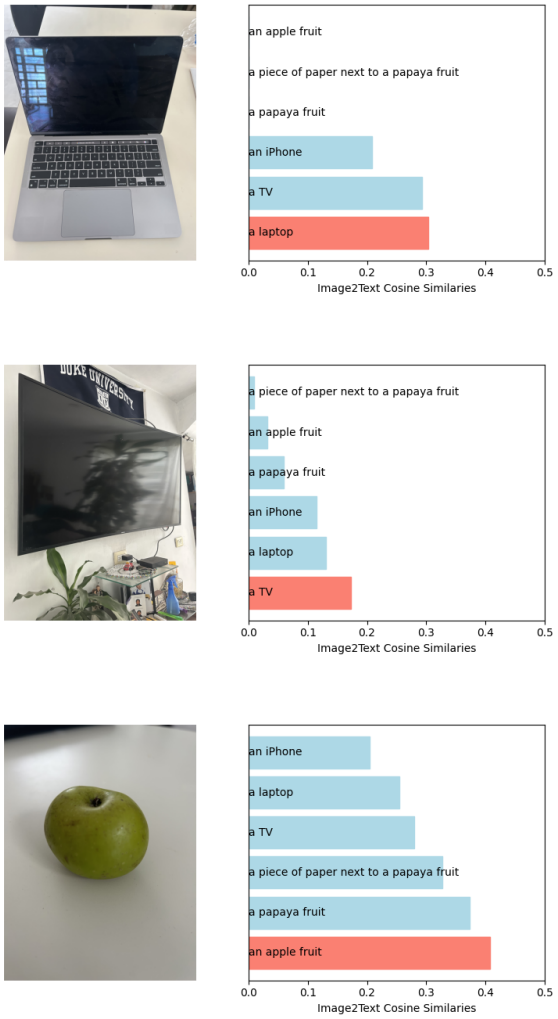

Figure 9. Sample of images analyzed by ViTB/32@224px.