JettCLIP - Faster MobileCLIP in Long Text-Image Alignment

Yangdi Yue **Xiaole Wang**
School of Electronics Engineering and Computer Science
Peking University
{2100010763, 2100016902}@stu.pku.edu.cn

Abstract

Contrastive Language-Image Pre-training (CLIP) has become a promising language-supervised visual pretraining framework. MobileCLIP reduces the parameter count of the CLIP model and accelerates its inference speed. However, there is still room for improvement, particularly in handling long texts. In this work, we introduce JettCLIP, a lightweight CLIP model with enhanced capability for handling long texts. JettCLIP modifies the text encoder of MobileCLIP to accept longer text inputs and is trained using a combination of training methods from both MobileCLIP and Long-CLIP. Additionally, we compare various knowledge distillation strategies. We also highlight the potential of using Flash Attention to further accelerate the model's efficiency. Experimental results demonstrate that JettCLIP outperforms MobileCLIP by approximately 8.4% for I2T task and 18.7% for T2I task in accuracy for long-caption image-text retrieval, while achieving inference speeds 1.14 $\times$ faster than Long-CLIP, with accuracy on various metrics close to that of Long-CLIP. Codes and models are released at https://github.com/Phrandime/JettCLIP.[1]

1 Introduction to MobileCLIP

1.1 Model Architecture

MobileCLIP [10] is a new family of efficient image-text models optimized for runtime performance along with a novel and efficient training approach, namely multi-modal reinforced training.

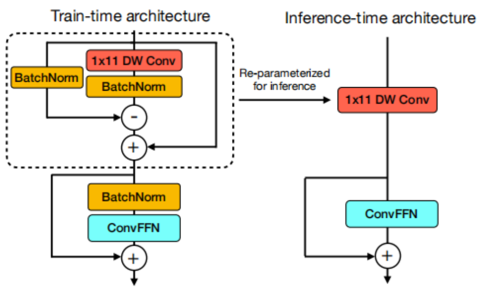


Figure 1: Architecture of convolutional and reparameterizable blocks, called Text-RepMixer used in MobileCLIP's text encoder MCt (reproduced from [10]).

The general Architecture of MobileCLIP is similiar to other CLIP models which consists of a text encoder and a image encoder. But in order to adapt to mobile device, this work designed more efficient

[1]We will continue to maintain our codebase after the deadline. To ensure fairness, please evaluate based on the assignment package we have submitted.

architecture to balance speed and performance. In detail, the text encoder MCt (MobileCLIP's text-encoder) is designed as a hybrid model which makes use of 1-D convolutions and self-attention layers, and Text-RepMixer (refer to Fig. 1) is introduced to decouples training time and inference time. While the image encoder is a variant of FastViT, called MCi (MobileCLIP's image-encoder). Depend on the parameters the MCi have, there are three variants called MCi0, MCi1, MCi2 introduced. With ViT-B/16, the author finally combined four models called MobileCLIP-s0, MobileCLIP-s1, MobileCLIP-s2, MobileCLIP-b.

1.2 Dataset and Training Method

The author introduced a novel multi-modal dataset reinforcement method. The method enhance the dataset through four stages of enhancement, which is synthetic captions, image augmentations, ensemble teacher and reinforced dataset. In state of synthetic captions, the author used CoCa model to generate multiple synthetic captions for each images. And generate multiple augmented images using a parametrized augmentation function in image augmentations. After that, an ensemble teacher CLIP models is used to compute the feature embeddings for augmented images and synthetic captions. Finally, the image augmentation parameters, synthetic captions, feature embeddings with the original image and caption is stored as one item(however, we find the data storage mode is different from the description in practice, which will be talked below).

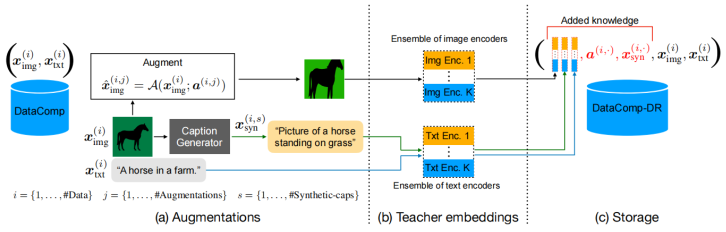

$\left(x_{\text{img}}^{(i)}, x_{\text{txt}}^{(i)}\right)$
DataComp
Augment
$\hat{x}_{\text{img}}^{(i,j)} = \mathcal{A}(x_{\text{img}}^{(i)}; a^{(i,j)})$
$x_{\text{img}}^{(i)}$
Caption Generator
$x_{\text{syn}}^{(i,s)}$
"Picture of a horse standing on grass"
$x_{\text{txt}}^{(i)}$
"A horse in a farm."
$i = \{1, \dots, \#\text{Data}\}$ $j = \{1, \dots, \#\text{Augmentations}\}$ $s = \{1, \dots, \#\text{Synthetic-caps}\}$
(a) Augmentations
Ensemble of image encoders
Img Enc. 1
Img Enc. K
Txt Enc. 1
Txt Enc. K
Ensemble of text encoders
(b) Teacher embeddings
Added knowledge
$\left(, a^{(i,\cdot)}, x_{\text{syn}}^{(i,\cdot)}, x_{\text{img}}^{(i)}, x_{\text{txt}}^{(i)} \right)$
DataComp-DR
(c) Storage

Figure 2: Illustration of multi-modal dataset reinforcement with one image augmentation and one synthetic caption. In practice, the author use multiple image augmentations and synthetic captions (reproduced from [10]).

The Training loss function is consist of two part, one is standard CLIP loss and another is a knowledge distillation loss. Specifically, Let $\mathcal{B}$ denote a batch of b (image, text) pairs, $\Psi_{\text{img}}^{(k)}, \Psi_{\text{txt}}^{(k)}$ the matrices of image and text embeddings of the k-th model in the teacher ensemble for batch $\mathcal{B}$, and $\varPhi_{\text{img}}, \varPhi_{\text{txt}}$ the image and text embedding matrices of the target model. $\mathcal{S}_\tau(\boldsymbol{U}, \boldsymbol{V})$ denotes the similarity matrix obtained by applying row-wise Softmax operation to $\boldsymbol{U}\boldsymbol{V}^\top/\tau$, where τ is a temperature parameter. The training loss is computed as:

$$\mathcal{L}_{\text{Total}}(\mathcal{B}) = (1-\lambda)\mathcal{L}_{\text{CLIP}}(\mathcal{B}) + \lambda\mathcal{L}_{\text{Distill}}(\mathcal{B}), \tag{1}$$

$$\mathcal{L}_{\text{Distill}}(\mathcal{B}) = \frac{1}{2}\mathcal{L}_{\text{Distill}}^{\text{I2T}}(\mathcal{B}) + \frac{1}{2}\mathcal{L}_{\text{Distill}}^{\text{T2I}}(\mathcal{B}), \tag{2}$$

$$\mathcal{L}_{\text{Distill}}^{\text{I2T}}(\mathcal{B}) = \frac{1}{bK}\sum_{k=1}^{K} \text{KL}(\mathcal{S}_{\tau_k}(\Psi_{\text{img}}^{(k)}, \Psi_{\text{txt}}^{(k)}) \| \mathcal{S}_{\hat{\tau}}(\varPhi_{\text{img}}, \varPhi_{\text{txt}})), \tag{3}$$

where KL denotes Kullback-Leibler divergence, $\mathcal{L}_{\text{Distill}}^{\text{T2I}}$ is computed by swapping the text and image embedding terms of $\mathcal{L}_{\text{Distill}}^{\text{I2T}}$, and λ is a tradeoff parameter.

For every data item we get before, we randomly load original image, text and one of augmentation parameters and synthetic captions with their embeddings corresponding to teacher models for training.

1.3 Code Reproduction

The code reproduction can be divided into two part, which is inference reproduction and training reproduction. For inference reproduction, we can normally use the pretrained models to take classification task in practice and produce similiar performance results on specific datasets. According to Fig. 3, we can see the top-1 accuracy result is almost equal. However, our inference latency (which is calculated by the latency sum of text encoder and image encoder) is very slower than the official results, while we use a Nvidia A-100 GPU for test and the author test it on an apple 12 Pro Max.

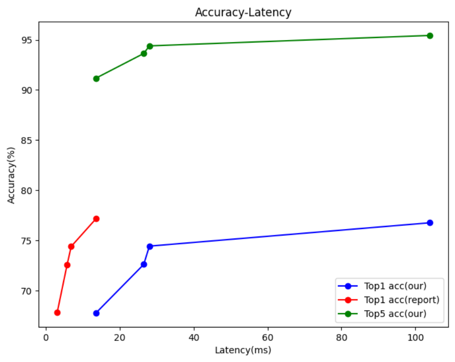
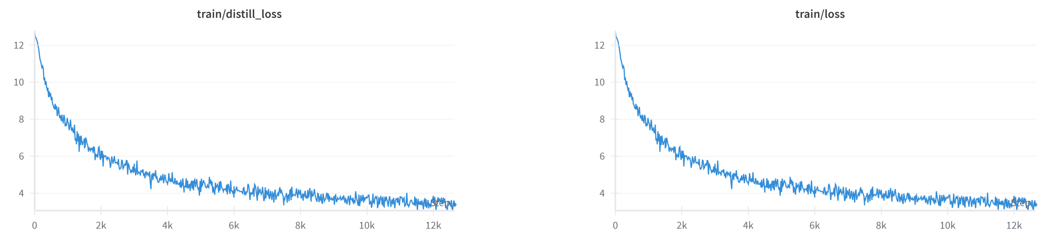


Figure 3: Evaluate results of both four models on ImageNet-1k

For training reproduction, though we can normally run training pipeline for the model. We countered a dilemma about the dataset and Computing power limitation. For the dataset, the DatacompDR was expanded to nearly forty times its original size due to the data reinforcement. That's to say, there is one thousand of packages sized about 2.8G each one, and the size is an result without regard to images. It's a very abstract thing that unlike other datasets that provide package of images, the image data of the dataset can only download one by one according to the url attached to the item. And get one package really cost about 6 hours on my machine.



(a) Distill loss

(b) Total loss

Figure 4: Training loss - steps

Therefore, in order to save time and save on our GPU server usage time for later improvements, we only use one package for the experiment at the reproduction stage. And for the pits we stepped on during the reproduction process, please refer to the attachment below.

1.4 Potential Improvements

Our motivation mainly fall on the following abstracts:

- MobileCLIP performs well in some short-captions tasks like image classification. However, with the increasing complexity of today's demands, traditional short-text alignment capacity are becoming insufficient to meet more complex semantic tasks. Therefore, we aim to leverage the long-text capacity of Long-CLIP[12] to enable MobileCLIP to handle complex application scenarios while maintaining its fast inference capacity.
- Some recent research shows that there are some other knowledge distillation methods that may perform better, so we want to make a comparison.
- Although the speed of MobileCLIP is very fast, we don't get a result as amazing as the work, so we want to further increase the speed.

2 Other Related Work

2.1 CLIP

CLIP (Contrastive Language-Image Pre-training) [9] is a multimodal model designed to learn visual concepts directly from natural language descriptions, enabling zero-shot transfer capabilities across various tasks. It uses an image encoder and a text encoder to transform images and text into feature vectors. Through contrastive learning, it aligns the corresponding feature vectors by maximizing their similarity while minimizing the similarity of unrelated pairs. Fig. 5 shows the pipeline of the training and inference process of CLIP model.

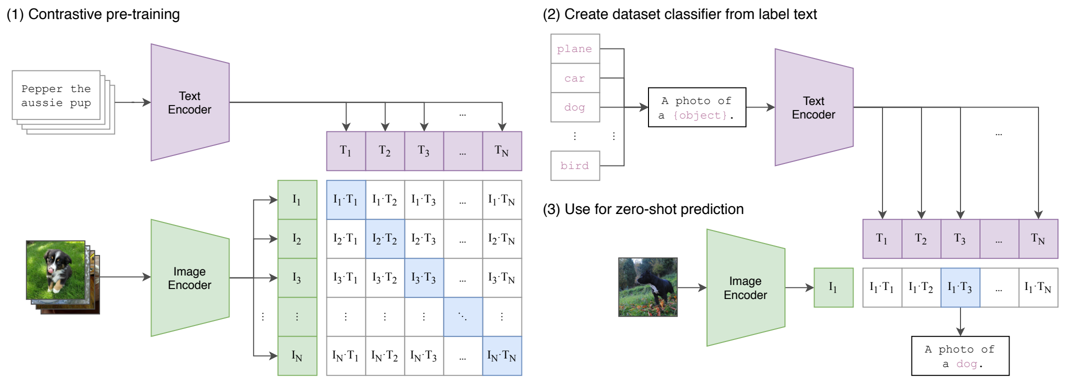


Figure 5: The pipeline of CLIP (reproduced from [9]).

CLIP, as an early approach in multimodal learning, still has significant room for improvement in terms of efficiency and performance. Since its introduction, many variants of CLIP have emerged, such as DeCLIP [5], SLIP[8], and MaskCLIP [3]. The MobileCLIP discussed in Sec. 1 is one of the variants, designed to be more lightweight and achieve faster inference speeds. Below, we will introduce Long-CLIP [12], another variant of CLIP, which is capable of handling longer texts.

2.2 Long-CLIP

A significant limitation of CLIP lies in its severely restricted input text length. The text encoder of CLIP employs an absolute positional embedding limited to 77 tokens. Moreover, only the lower token positions are adequately trained, with the result that the actual effective length for CLIP is only about 20 tokens.

Long-CLIP [12] aims to unlock CLIP's capability to handle longer texts, thereby capturing more details in images. While their model architectures are similar, the text encoder in Long-CLIP utilizes longer positional encodings. To leverage the pre-trained parameters of the CLIP model, Long-CLIP initializes the positional encodings using the following formula:[2]

$$PE^*(pos) = \begin{cases} PE(pos), & 1 \leq pos \leq l \\ (1-\alpha) \times PE(pos^*) + \alpha \times PE(pos^*+1), & l+1 \leq pos \leq 77 \end{cases} \tag{4}$$
$$\text{where} \quad pos^* = \lfloor \frac{pos - l}{\lambda} \rfloor + l, \ \ \alpha = \frac{pos}{\lambda} - \lfloor \frac{pos}{\lambda} \rfloor.$$

Here, $l = 20$ represents the length of the retained positional encodings. This design allows the model to make use of the well-trained positional encodings for the first 20 tokens, while extending the positional encoding length through interpolation. In the implementation of Long-CLIP, $\lambda = 4$, which results in a maximum input length of 248 tokens. However, there is a small bug in the source code of Long-CLIP, which causes l to actually be 21. We have listed this issue in Sec. A.1.

Long-CLIP's training method employs a Primary Component Matching strategy in long-text fine-tuning, which aims to both unlock the long-text capability and preserve the short-text capability. Fig. 6 shows its framework. Therefore, Long-CLIP's dataset requires each image to have both a long text and a short text description. They use the ShareGPT4V [1] dataset for training, which contains approximately 1 million (long caption, image) pairs, and simply extract the first sentence of each text as the short text description.

Since the Long-CLIP model architecture does not incorporate the lightweight techniques used in MobileCLIP, applying the Long-CLIP method to enable MobileCLIP to handle long texts has become an interesting and valuable research problem.

[2]The formula in the original Long-CLIP paper was incorrect, and here we provide the correction.

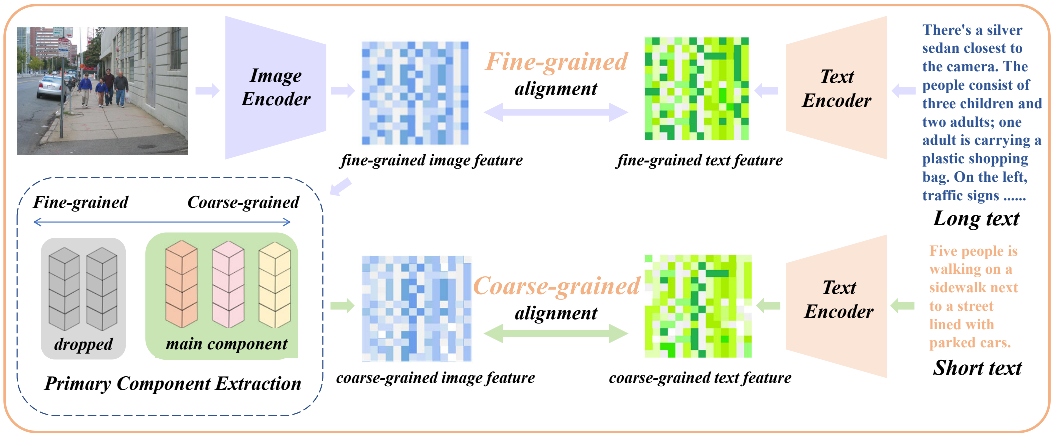


Image Encoder
Fine-grained alignment
fine-grained image feature
fine-grained text feature
Text Encoder
There's a silver sedan closest to the camera. The people consist of three children and two adults; one adult is carrying a plastic shopping bag. On the left, traffic signs ......
Long text
Fine-grained
Coarse-grained
dropped
main component
Primary Component Extraction
Coarse-grained alignment
coarse-grained image feature
coarse-grained text feature
Text Encoder
Five people is walking on a sidewalk next to a street lined with parked cars.
Short text

Figure 6: The Long-CLIP training framework (reproduced from [12]).

2.3 CLIP-KD

CLIP-KD [11] aims to find the best knowledge distillation strategy to enhance a small student CLIP model supervised by a pre-trained large teacher CLIP model. As shown in this work, a simple feature mimicry with Mean Squared Error loss works surprisingly well. This method is called Feature Distillation.

MobileCLIP, as mentioned in Sec. 1.2, also uses knowledge distillation for training, but its distillation method, according to Eqs. (2) and (3), is Contrastive Relational Distillation. The loss function we use to train JettCLIP is similar to that of MobileCLIP. Thus, it is worth trying different distillation methods when we train our JettCLIP.

2.4 Flash Attention

Flash Attention is one of the representative works of attention optimization based on I/O, which improves inference performance by reducing memory access overhead.

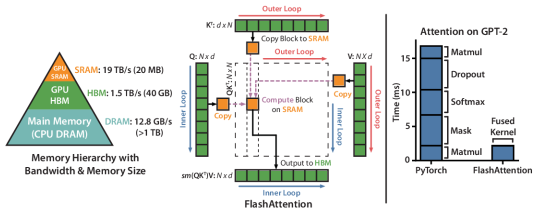


Figure 7: Diagram of Flash Attention (reproduced from [2]).

To address the I/O issue, the main goal of Flash Attention is to avoid reading and writing the attention matrix to high-bandwidth memory(HBM). They employ two techniques to achieve this. First, they chunk the input and traverse the input blocks multiple times to progressively perform softmax reduction, which is also known as tiling. Second, they store the softmax normalization factors from the forward pass to quickly recompute on-chip attention during the backward pass, which is faster than the standard method of reading attention matrix from HBM. We believe that the method is efficient to all device with limited shared memory. Therefore, we want further improve the speed by applying the method. This is a work for exploration, and the result is put on Appendix, refer to Sec. A.2

3 Approach

3.1 Architecture

The architecture of our JettCLIP is nearly identical to that of MobileCLIP, so the details can be referred to from Sec. 1.1. In a word, we utilize hybrid CNN-transformer architectures in both the image and text encoders, and also apply structural reparameterization to enable faster inference.

The only difference lies in the position embedding of the text encoder. To enable JettCLIP to handle long text inputs, we adjust the text encoder by setting the length of the position embedding layer to 248. We then initialize the position embedding layer using the method described in Sec. 2.2, allowing JettCLIP to leverage the pretrained parameters from MobileCLIP.

3.2 Training

Feature encoding with teacher model. We apply knowledge distillation, using the pre-trained Long-CLIP model as the teacher to train our JettCLIP model as the student. The training loss is a combination of CLIP loss and distillation loss, where the distillation loss requires encoded features of the image and text data generated by the teacher model. Encoding the data on-the-fly during training with the teacher model would significantly increase computational overhead. Therefore, we typically pre-encode the data with the teacher model and treat the encoded features as part of the dataset. In this work, we use the Long-CLIP model to encode each sample into four features: the fine-grained image feature obtained by directly encoding the image, the coarse-grained image feature obtained by applying PCA to the fine-grained image feature, the feature from encoding long texts, and the feature from encoding short texts.

CLIP loss. We compute the CLIP loss using the same approach as in Long-CLIP. For each (image, long caption) sample, we first obtain fine-grained image and text features using the image and text encoders. Additionally, we apply PCA to extract coarse-grained image features from the fine-grained image features and encode a short summary caption using the text encoder. For our dataset, the short summary caption is obtained by extracting the first sentence from the long caption. In addition to aligning the fine-grained image feature with its long caption, we also align the coarse-grained image feature with the short summary caption. For a batch $\mathcal{B}$, the CLIP loss is defined as:

$$\mathcal{L}_{\text{CLIP}}(\mathcal{B}) = \lambda_1 \mathcal{L}_{\text{CLIP-long}}(\mathcal{B}) + \lambda_2 \mathcal{L}_{\text{CLIP-short}}(\mathcal{B}). \tag{5}$$

Distillation loss. The distillation loss consists of two parts: one for aligning the fine-grained image feature with its long caption, and the other for aligning the coarse-grained image feature with the short summary caption. The calculation method for both parts is the same. For simplicity, for a batch $\mathcal{B}$, we denote the image and text features encoded by the teacher model (whether fine-grained or coarse-grained) as $\Psi_{\text{img}}, \Psi_{\text{txt}}$, respectively. Similarly, for the student model, we define $\varPhi_{\text{img}}$ and $\varPhi_{\text{txt}}$.

If we use Contrastive Relational Distillation (CRD), the distill loss is calculated in the same way as in Eq. (3), but with the assumption that there is only one teacher model:

$$\mathcal{L}_{\text{Distill}}(\mathcal{B}) = \frac{1}{2}\mathcal{L}^{\text{I2T}}_{\text{Distill}}(\mathcal{B}) + \frac{1}{2}\mathcal{L}^{\text{T2I}}_{\text{Distill}}(\mathcal{B}), \tag{6}$$

$$\mathcal{L}^{\text{I2T}}_{\text{Distill}}(\mathcal{B}) = \frac{1}{|\mathcal{B}|}\text{KL}(\mathcal{S}_{\tau}(\Psi_{\text{img}}, \Psi_{\text{txt}}) \| \mathcal{S}_{\widehat{\tau}}(\varPhi_{\text{img}}, \varPhi_{\text{txt}})), \tag{7}$$

If we use Feature Distillation (FD) then

$$\mathcal{L}_{\text{Distill}}(\mathcal{B}) = \frac{1}{2|\mathcal{B}|}\left(\|\Psi_{\text{img}} - \varPhi_{\text{img}}\|_F^2 + \|\Psi_{\text{txt}} - \varPhi_{\text{txt}}\|_F^2\right). \tag{8}$$

Note that FD requires the feature dimensions of the teacher and student to be equal, whereas CRD does not have this requirement.

Total loss. Finally, we can calculate the total loss of a batch $\mathcal{B}$ as:

$$\mathcal{L}_{\text{Total}}(\mathcal{B}) = \lambda_1 \mathcal{L}_{\text{CLIP-long}}(\mathcal{B}) + \lambda_2 \mathcal{L}_{\text{CLIP-short}}(\mathcal{B}) + \lambda_3 \mathcal{L}_{\text{Distill-long}}(\mathcal{B}) + \lambda_4 \mathcal{L}_{\text{Distill-short}}(\mathcal{B}) \tag{9}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are four hyperparameters that represent the weights of the four losses.

4 Experiments

4.1 Experiment Setup

Evaluation Dataset. We evaluate our model in three downstream tasks.

- **zero-shot image classification.** The main dataset is *CIFAR-10* [4], *CIFAR-100* [4].
- **short-caption image-text retrieval.** For traditional short-caption image-text retrieval, we use *COCO2017* [6]. For *COCO2017*, we use the 5k validation set.
- **long-caption image-text retrieval.** For long-caption image-text retrieval, we use *Urban1k* [12].

Evaluation Setting. All the experiments in this section share same settings, including the class template in zero-shot classification, all directly truncating the input token if it's longer than restriction, and so on.

Training setting. We use a part of ShareGPT4V [1] dataset as training dataset, which contains about the entire *COCOtrain2017* [6] and *LLAVA* [7]. We fine-tune for 4 epoch with batch size 256, in which costs about one hour every epoch. Detailed hyper-parameter settings will be listed in supplementary material.

4.2 Comparing with Other CLIP Model

We compare our model with MobileCLIP and Long-CLIP in the three different downstream tasks mentioned above. As it can see, our model gets remarkably better long-text capacity than MobileCLIP with an almost equally excellent shot-text capacity. Besides, we also take latency test via Urban1k evaluation. Our JettCLIP works faster than LongCLIP that benefits from MobileCLIP's architecture. The image encoder latency is calculated by the text encoding time, while the image encoder latency is calculated with the average of image encoding time. The detailed result is shown in Tab. 1.

Table 1: A comparison of different CLIP models. Best result is in **bold**.

Model	Cifar10	Cifar100	COCO T2I R@10	Urban I2T R@1	Urban T2I R@1
CLIP	90.80	67.30	x	46.5	46.00
MobileCLIP	**93.39**	68.65	59.20	64.70	58.20
Long-CLIP	90.79	69.11	75.70	**79.20**	**79.40**
Ours	92.44	**70.28**	**75.85**	73.10	76.90

Table 2: A latency(ms) comparison of differnent CLIP models.

Model	MobileCLIP	Long-CLIP	Ours
Text encoder	388.9	548.9	472.0
Image encoder	7.9	7.8	7.8

4.3 Ablation experiment

There are two core component strategy in our training method. The first is the mixed loss strategy with different loss weight(loss for long text, loss for short text, distill loss for long text and distill loss for short text), which reflects the trade-off between the distillation loss and the CLIP loss. The second is the distillation type, which includes Contrastive Relational Distillation(CRD) and Feature Distillation(FD)[11]. To demonstrate that both strategies can improve model performance, we tested the model's accuracy on different tasks while adjusting these two strategies. For the loss weight, we designed several weight ratios to highlight the importance of distillation; for the distillation type, we conducted experimental comparisons by fixing the loss weight and applying different distillation types. We find that in the case of a mixed loss with CRD, applying a larger weight to the long-text related loss is more beneficial for model training. The result is demonstrated in Tab. 3 and Tab. 4.

Table 3: A comparison of different weight of loss, the ratio is loss of short : loss of long : distill loss of long : distill loss of short. Best result is in **bold**.

Weight	Cifar10	Cifar100	COCO T2I R@10	Urban I2T R@1	Urban T2I R@1
1:1:0:0	91.48	69.55	74.64	67.80	71.50
0:0:1:1	**92.22**	69.66	75.17	65.60	70.60
1:1:1:1	91.91	69.66	75.32	69.50	73.10
0.8:0.4:1.5:0.5	92.19	**69.84**	**75.66**	**71.20**	**76.30**

Table 4: A comparison of different distillation type (Contrastive Relational Distillation and Feature Distillation). Best result is in **bold**.

Distillation type	Cifar10	Cifar100	COCO T2I R@10	Urban I2T R@1	Urban T2I R@1
FD	89.35	57.31	65.96	59.60	58.20
CRD	**92.19**	**69.84**	**75.66**	**71.20**	**76.30**

Tab. 3 indicates that paying more attention to the alignment of fine-grained features seems to result in better training performance. This suggests that improving the model's long-text capability may be more important than maintaining its short-text capability when training is insufficient. Additionally, it shows that knowledge distillation can help improve our model's training.

Tab. 4 shows that Feature Distillation performs slightly worse than Contrastive Relational Distillation, which could be due to the difficulty of requiring the student model to fully align with the teacher model's features, with CLIP-KD being a special case.

5 Limitations

Despite the promising results achieved in this study, there are several limitations that should be acknowledged:

Insufficient dataset We only use a part of ShareGPT4V dataset, containing 676k (long caption, image) pairs. Besides, we have not fully implemented the detaset reinforcement method proposed in MobileCLIP. Future work could explore using larger datasets and dataset reinforcement techniques, which have the potential to further improve the model's accuracy and generalization ability.

Lacking access to mobile device MobileCLIP is designed for mobile Apple device, but for our limitation of Apple devices, we were unable to experience the demands and challenges faced on a mobile environments. In future work, we look forward to conducting exploration on real mobile devices.

6 Conclusion

We propose a new model called JettCLIP based on MObileCLIP and Long-CLIP, which combines the fast and amazing short-text inference capability of MobileCLIP with complex long-text reasoning capability of Long-CLIP, enabling it to adapt to a variety of complex tasks. Additionally, we explored the acceleration provided by FlashAttention and saw faster inference capability. However, we still have some limitations mentioned above, which will be our further improvement directions. We hope our work will inspire future researchers to combine the lightweight techniques of MobileCLIP with the strengths of other CLIP models, in order to design more complex and efficient models with enhanced capabilities.

References

[1] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. Sharegpt4v: Improving large multi-modal models with better captions, 2023. URL `https://arxiv.org/abs/2311.12793`. 4, 7

[2] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL `https://arxiv.org/abs/2205.14135`. 5, 10

[3] Xiaoyi Dong, Jianmin Bao, Yinglin Zheng, Ting Zhang, Dongdong Chen, Hao Yang, Ming Zeng, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, and Nenghai Yu. Maskclip: Masked self-distillation advances contrastive language-image pretraining, 2023. URL `https://arxiv.org/abs/2208.12262`. 4

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 7

[5] Yangguang Li, Feng Liang, Lichen Zhao, Yufeng Cui, Wanli Ouyang, Jing Shao, Fengwei Yu, and Junjie Yan. Supervision exists everywhere: A data efficient contrastive language-image pre-training paradigm, 2022. URL `https://arxiv.org/abs/2110.05208`. 4

[6] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *ECCV*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014. 7

[7] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *CoRR*, abs/2310.03744, 2023. 7

[8] Norman Mu, Alexander Kirillov, David Wagner, and Saining Xie. Slip: Self-supervision meets language-image pre-training, 2021. URL `https://arxiv.org/abs/2112.12750`. 4

[9] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL `https://arxiv.org/abs/2103.00020`. 3, 4

[10] Pavan Kumar Anasosalu Vasu, Hadi Pouransari, Fartash Faghri, Raviteja Vemulapalli, and Oncel Tuzel. Mobileclip: Fast image-text models through multi-modal reinforced training, 2024. URL `https://arxiv.org/abs/2311.17049`. 1, 2

[11] Chuanguang Yang, Zhulin An, Libo Huang, Junyu Bi, Xinqiang Yu, Han Yang, Boyu Diao, and Yongjun Xu. Clip-kd: An empirical study of clip model distillation, 2024. URL `https://arxiv.org/abs/2307.12732`. 5, 7

[12] Beichen Zhang, Pan Zhang, Xiaoyi Dong, Yuhang Zang, and Jiaqi Wang. Long-clip: Unlocking the long-text capability of clip, 2024. URL `https://arxiv.org/abs/2403.15378`. 3, 4, 5, 7

A Appendix

A.1 Small Issues in Long-CLIP's Source Code

In the file `model/longclip.py` of the source code[3] of Long-CLIP, the authors seem to have overlooked that list indexing starts from 0, which resulted in retaining the first 21 position encodings when using `load_from_clip()` to load the CLIP model and extend the position encoding length. As a result, the last 4 position encodings were not properly interpolated.

```
    length, dim = positional_embedding_pre.shape
    keep_len = 20
    posisitonal_embedding_new = torch.zeros([4*length-3*keep_len,
        dim], dtype=model.dtype)
    for i in range(keep_len):
        posisitonal_embedding_new[i] = positional_embedding_pre[i]
    for i in range(length-1-keep_len):
        posisitonal_embedding_new[4*i + keep_len] =
            positional_embedding_pre[i + keep_len]
        posisitonal_embedding_new[4*i + 1 + keep_len] =
            3*positional_embedding_pre[i + keep_len]/4 +
            1*positional_embedding_pre[i+1+keep_len]/4
        posisitonal_embedding_new[4*i + 2+keep_len] =
            2*positional_embedding_pre[i+keep_len]/4 +
            2*positional_embedding_pre[i+1+keep_len]/4
        posisitonal_embedding_new[4*i + 3+keep_len] =
            1*positional_embedding_pre[i+keep_len]/4 +
            3*positional_embedding_pre[i+1+keep_len]/4
```

[3] `https://github.com/beichenzbc/Long-CLIP/`

```
242
243     posisitonal_embedding_new[4*length -3*keep_len - 4] =
            positional_embedding_pre[length-1] +
            0*(positional_embedding_pre[length-1] -
            positional_embedding_pre[length-2])/4
244     posisitonal_embedding_new[4*length -3*keep_len - 3] =
            positional_embedding_pre[length-1] +
            1*(positional_embedding_pre[length-1] -
            positional_embedding_pre[length-2])/4
245     posisitonal_embedding_new[4*length -3*keep_len - 2] =
            positional_embedding_pre[length-1] +
            2*(positional_embedding_pre[length-1] -
            positional_embedding_pre[length-2])/4
246     posisitonal_embedding_new[4*length -3*keep_len - 1] =
            positional_embedding_pre[length-1] +
            3*(positional_embedding_pre[length-1] -
            positional_embedding_pre[length-2])/4
```

Listing 1: bugging code in Long-CLIP

To fix the bug, we simply modify lines 233 and 234 to the following code, then delete lines 244 to 246.

```
keep_len = 19  # 20
posisitonal_embedding_new = torch.zeros([4*length-3*keep_len - 3, dim], dtype=model.dtype)
```

A.2 Latency evaluation on Flash Attention

This section shows our exploration on speeding up MobileCLIP based on Flash Attention[2]. We found that the speed of MCt can boost by a replacement of MHA with flash attention, especially when the input shape get larger(for that the size changed is meaningless, so we omit the discussion). Our experimental Setup: Inference only, with an average latency over 400 episodes (excluding the first 100 episodes to remove the warm-up effect). Detailed result can be found in Tab. 5 and Tab. 6.

Table 5: A latency(ms) comparison of MobileCLIP's MHA and MHA with flash attention.

Input Shape	(1,50,512)	(100,50,512)	(100,512,512)	(1024,512,512)
MobileCLIP's MHA	52.3	53.9	135.3	13377.2
MHA with flash-attn	41.4	40.7	39.9	41.9
Faster by	20.84%	24.49%	70.51%	99.69%

Table 6: A latency(ms) comparison of original MobileCLIP text encoder w/o flash attention.

Model	s0	s1	s2
Text encoder	52.0	88.2	87.7
Text encoder with flash-attn	43.2	61.6	61.6
Faster by	16.92%	30.16%	29.76%

B Member Contributions

Xiaole Wang:

- Reproduced the code of MobileCLIP.
- Implemented the code for JettCLIP's model architecture and how to load it.
- Conducted extensive experiments with different parameters for JettCLIP and evaluated the checkpoints.
- Proposed the use of Flash Attention and conducted related experiments.
- Organized the code and added documentation files.

Yangdi Yue:

- Designed the framework of JettCLIP and developed the experimental plan.
- Implemented the code for preprocessing data using the teacher model, along with the data loader.
- Designed multiple knowledge distillation methods and implemented the code.
- Implemented the main code for training JettCLIP, tested on small batches to ensure correctness of the code above.
- Found a small bug in Long-CLIP's Source Code.

Together:

- Studied and discussed MobileCLIP and other related works.
- Shared ideas on the project and discussed the details of the process.
- Wrote the report.

C Questions Raised During Class Presentation

Xiaole Wang (2100016902):

- 12.2 Mip-Splatting: Alias-free 3D Gaussian Splatting.
- 12.5 VCP-CLIP: A Visual Context Prompting Model for Zero-Shot Anomaly Segmentation.
- 12.23 3DGS-Avatar: Animatable Avatars via Deformable 3D Gaussian Splatting.

Yangdi Yue (2100010763):

- 12.2 Griffon: Spelling out All Object Locations at Any Granularity with Large Language Models.
- 12.5 Paying More Attention to Image: A Training-Free Method for Alleviating Hallucination in LVLMs.
- 12.23 Exploring the Transferability of Visual Prompting for Multimodal Large Language Models.