Target Cross-Modal Late Interaction (CMLI)

Cross-Modal Late Interaction (CMLI)

Until now, we used the global text and image representations [T_CLS] and [I_CLS], respectively, for contrastive learning and the alignment loss of (TODO: cite removing itc). This has the disadvantage that only global information is utilized, and fine-grained, token/patch-specific, information is not considered. This can make retrieval, and alignment in general, difficult, especially if real-world concepts described by and image and text differ in small, yet important, details. To address this, the authors of FILIP [1] introduce Cross-Modal Late Interaction (CMLI) for a fine-grained comparison of text and image in contrastive learning.

As shown in Figure 1, no cosine similarity between [T_CLS] and [I_CLS] is computed, but instead the cosine similarity between all image patches $\left[\boldsymbol{v}_l^k\right]_{1 \leq k \leq N}$ and text tokens $\left[\boldsymbol{w}_l^j\right]_{1 \leq j \leq M}$, with N being the number of image patches, and M being the number of text tokens. Specifically, N and M denote the number of patches/tokens in a sequence that are not the cls token ([I_CLS]/[T_CLS]) or padding token ([PAD]) [1]. The choice to exclude padding tokens is obvious, as they do not carry any semantic information. The cls token is excluded, as it contains "just" global information. The result is that we now have the cosine similarity between all image patches and text tokens of an image-text pair.

The next step is to find for each image patch k, the text token with the maximum cosine similarity to this image patch.

$$m_k^{\mathrm{i2t}} = \underset{1 \leq j \leq M}{\operatorname{argmax}} \left[\boldsymbol{v}^k\right] \left[\boldsymbol{w}^j\right]^T \tag{1}$$

Likewise, for each text token j, we get the image patch with the maximum cosine similarity to this text token

$$m_j^{\mathrm{t2i}} = \underset{1 \leq k \leq N}{\operatorname{argmax}} \left[\boldsymbol{v}^k\right] \left[\boldsymbol{w}^j\right]^T \tag{2}$$

This has an intersting effect: For each image patch, the semantically most similar text token is found, and vice versa for each text token - the result of this operation can be seen in (2) of Figure 1. Consequently, the model will be able to associate small details of an image with individual text tokens, and vise versa. The actual cosine similarity between an image-text pair is then the average of all associations between an image patch and a text token.

$$s_{\boldsymbol{H}_l^v, \boldsymbol{H}_l^w}^{\mathrm{i2t}} = \frac{1}{N} \sum_{k=1}^{N} \left[\boldsymbol{v}_l^k\right] \left[\boldsymbol{w}_l^{m_k^{\mathrm{i2t}}}\right]^T \tag{3}$$

$$s_{\boldsymbol{H}_l^v, \boldsymbol{H}_l^w}^{\mathrm{t2i}} = \frac{1}{M} \sum_{j=1}^{M} \left[\boldsymbol{v}_l^{m_j^{\mathrm{t2i}}}\right] \left[\boldsymbol{w}_l^j\right]^T \tag{4}$$

Here, for one image-text pair, m_k^{i2t} denotes the index of the text token with the highest cosine similarity to image patch k, and m_j^{t2i} the index of the image patch with the highest cosine similarity to text token j. $s_{\boldsymbol{H}_l^v, \boldsymbol{H}_l^w}^{\mathrm{i2t}}$ denotes the the similarity score between an image representation $\boldsymbol{H}_l^v$ and text representation $\boldsymbol{H}_l^w$. Vice versa, $s_{\boldsymbol{H}_l^v, \boldsymbol{H}_l^w}^{\mathrm{t2i}}$ denotes the similarity score between a text representation $\boldsymbol{H}_l^w$ and an image representation $\boldsymbol{H}_l^v$. l can denote any layer of the model, but we will use, as done in FILIP [1], the last layer of the model, so if a model has L layers, then $l = L$.

In contrast to the standard contrastive learning, this similarity measure is not necessarily symmetric, as e.g. a text token might have a maximum cosine similarity to another image patch, than a image patch to the text token [1]. The process in illustrated in Figure 1.

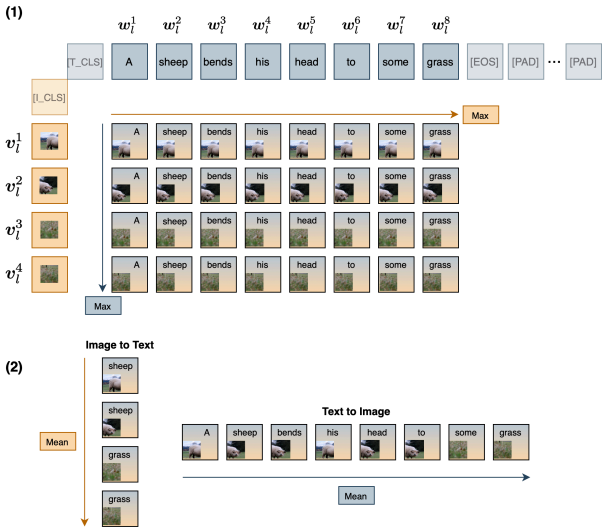

Figure 1: For a token/patch, CMLI finds the semantic timestep with the highest match from the other modality. This enables the model to associate small details of image and text with each other. Notice how through the max-operation patches containing grass are always associated with the word "grass", and the words "sheep" and "head" are matched with the head of the sheep (associations created through max are shown in (2)). The cosine similarity is then the average of all associations between an image-text pair. Figure inspired and adapted from [1].

While this approach allows for a fine-grained alignment of image and text, its practical implementation is very computationally and memory intensive. For standard constrastive learning, is is sufficient to compute the cosine similarity of the global representation (cls token) between every possible image-text pair in a batch. If negative examples are gathered from all devices, then the number of dot products to compute is defined as $(B * P)^2$, with B being the batch size per device, and P being the number of devices (in our case GPUs). As we use a batch size of $B = 256$ per device, and use $P = 2$ GPUs, the number of dot products to compute is $(256 * 2)^2 = 262,144$. Considering that we perform this efficiently using matrix multiplication, and the embedding size is 768, with float32 precision, we already need $262,144 * 768 * 4$ bytes $= 805.31$ MB of GPU memory, which is still manageable, since we have around 2 GB of GPU memory remaining for a step.

However, with CMLI, we need to compute the similarity between all possible image-text pairs, where the similarity for one pair requires the computation of the cosine similarity between all image patches and text tokens of the image-text pair. With a maximum text sequence length of 64 tokens [2], two of which are ignored as they are the cls and eos token, and an image sequence length of 196 (without cls token), the number of dot products to compute for just one image-text pair is $196 * 64 = 12,544$. With a batch size of 256 per device, and 2 GPUs, the number of dot products increases from $262,144$ to $256 * 12,544 = 6,422,528$. Even if the embedding dimension is reduced to 256, which is a simplification done in FILIP [1], we need $6,422,528 * 256 * 4$ bytes $= 6.58$ GB of additional GPU memory, just to store the result. Consequently, this approach is not feasible in our setup.

Method

What is feasible though, is to apply CMLI to a setting where the computation is more lightweight. As in Contrastive Learning, the driving factor behind the memory requirements is that the similarity between all possible image-text pairs in a batch is computed. However, this is not the case when just the similarity between the positive pairs is computed, which is the case for what we call Target-CMLI.

Target-CMLI is not used for contrastive learning, but rather to alleviate the problem of regressing patch-level information of the teacher model. Recall that in the current setting, which is Multimodal Knowledge Distillation, it is merely possible to regress the global representations of the teacher model, and not the patch-level information. This is because the teacher model only outputs patch-level predictions for the image modality, and not for the text modality. Consequently, the student model can replicate the output of the teacher model for the image modality, but not for the text modality, as it is not possible to assign a text token to a specific image patch (without labeled data). This is illustrated in (TODO: cite @mm_kd_cls_token) of (TODO: cite @differences_to_unimodal_knowledge_distillation).

However, as seen in Figure 1, when computing the cosine similarity between all image patches and text tokens of an image-text pair, and selecting the argmax over all image patches with respect to a text token, then a corresponding, or at least similar, image patch can be found for each text token ((2) of Figure 1). This means that the student model can replicate the output of the teacher model for the text modality, by first selecting the most similar image patch for each text token, and then minimizing the Mean Squared Error (MSE) between the teacher's representation of the selected image patch and the representation of the text token.

For the patch-level image representation of the student model that means that we can now also regress the patch-level information of the teacher, and not only the global information. This was also possible in all previous experiments, as the order of the image patches does not change, however, this would heavily bias the parameters of the shared Transformer block towards the image modality. The definition of the loss changes as follows. For a given image representation

$$\boldsymbol{H}^s_{v,L} = \left[\boldsymbol{h}^s_{v,L,[\texttt{I\_CLS}]}, \boldsymbol{h}^s_{v,L,1}, ..., \boldsymbol{h}^s_{v,L,N}\right] \tag{5}$$

of the student, and the image representation

$$\boldsymbol{H}^t_{v,L} = \left[\boldsymbol{h}^t_{v,L,[\texttt{I\_CLS}]}, \boldsymbol{h}^t_{v,L,1}, ..., \boldsymbol{h}^t_{v,L,N}\right] \tag{6}$$

of the teacher, the loss is defined as:

$$\begin{gathered}\mathcal{L}^{\text{i2t}}_{\text{KD}} = \text{MSE}\Big(\boldsymbol{H}^s_{v,L}, \boldsymbol{H}^t_{v,L}\Big) = \\ \frac{1}{N+1}\left(\|\boldsymbol{h}^s_{v,L,[\texttt{I\_CLS}]} - \boldsymbol{h}^t_{v,L,[\texttt{I\_CLS}]}\|^2 + \sum_{k=1}^{N}\|\boldsymbol{h}^s_{v,L,k} - \boldsymbol{h}^t_{v,L,k}\|^2\right)\end{gathered} \tag{7}$$

For a given text representation

$$\boldsymbol{H}^s_{w,L} = \left[\boldsymbol{h}^s_{w,L,[\texttt{T\_CLS}]}, \boldsymbol{h}^s_{w,L,1}, ..., \boldsymbol{h}^s_{w,L,M}, \boldsymbol{h}^s_{w,L,[\texttt{T\_SEP}]}\right] \tag{8}$$

of the student, we first need to find m^{t2i}_k for each text token k, and then define the loss as:

$$\begin{gathered}\mathcal{L}^{\text{t2i}}_{\text{KD}} = \text{MSE}\Big(\boldsymbol{H}^s_{w,L}, \boldsymbol{H}^t_{v,L}\Big) = \\ \frac{1}{M+1}\left(\|\boldsymbol{h}^s_{w,L,[\texttt{T\_CLS}]} - \boldsymbol{h}^t_{v,L,[\texttt{I\_CLS}]}\|^2 + \sum_{k=1}^{M}\|\boldsymbol{h}^s_{w,L,k} - \boldsymbol{h}^t_{v,L,m^{\text{t2i}}_k}\|^2\right)\end{gathered} \tag{9}$$

Notice how in both cases we also regress the global representation of the teacher $\boldsymbol{h}^t_{v,L,[\texttt{I\_CLS}]}$ for the given image. So the first term (the one before the sum-operator) is the loss we used before. In both cases, we weight the loss of the global representation the same as the loss on the patch-level representations. The total Knowledge-Distillation loss remains the same, and is the mean of the two losses:

$$\mathcal{L}_{\text{KD}} = \frac{1}{2} * (\mathcal{L}_{\text{KD}}^{\text{i2t}} + \mathcal{L}_{\text{KD}}^{\text{t2i}}) \tag{10}$$

We use the mean instead of the sum, as we also use the contrastive loss, and we want both losses to have the same weight. As in FILIP [1], we find m_k^{t2i} using an embedding dimension of 256. The hidden size of the model stays at 768, and we use a linear projection to reduce the dimensionality of the image representation from the teacher, and the text representation from the student, to 256. As the goal is to bring the representations of the student as close as possible to the teacher, we use the same linear projection for both the student and the teacher representations. The loss is still computed using the raw outputs, with 768 dimensions. Only CMLI is performed in the lower-dimensional space.

What makes the implementation of Target-CMLI feasible is that we only need to compute the similarity between the positive pairs, and not all possible image-text pairs in a batch. This is because we only need to find the most similar image patch for each text token of a positive pair. For a per-device batch size of 256, Target-CMLI requires $12,544 * 256 = 3,211,264$ dot products to compute. With an embedding dimension of 256 and half-precision float16, just $3,211,264 * 256 * 2$ bytes $= 1.64$ GB of additional GPU memory is required. This is feasible in our setup.

Empty Target

A weakness of the aforementioned approach is that some not all text tokens carry semantic information that can be mapped to image patches. An example of this can be seen in Figure 1, where the tokens “A”, “his”, “to”, and “some” do not contains any information that can be related to an image patch, because they are merely fill words in a sentence. However, with Target-CMLI there will be an image patch that is most similar to these tokens, even if the similarity is very low. Consequently, the model will try to minimize the MSE between the representation of these tokens and the corresponding image patches, which is not meaningful.

To address this, we propose the introduction of an empty target, which is a learnable token to which all text tokens are, next to the image patches, compared to using cosine similarity. If the maximum similarity for a text token is the empty target, then the loss for this token is set to zero, i.e. is ignored.

However, this approach will inevitably lead to a model collapse. This is because the model will try to find the easiest way to minimize the loss ($\mathcal{L}_{\text{KD}}$), and the easiest way to do that is to simply have all text tokens have the highest similarity to the empty target. This will result in a loss of zero, as the loss for all text token is now ignored, i.e. zero. Consequently, the model will not learn any meaningful representations. A potential solution to this is to add another loss term that encourages the model minimize the number of text tokens that have the highest similarity to the empty target, which will strike a balance between utilizing the empty target for meaningless tokens, and using actual image patches for meaningful tokens. We define the loss $\mathcal{L}_{\text{Reg}}$ as follows:

$$\mathcal{L}_{\text{Reg}} = -\log(1 - p) \tag{11}$$

We denote p as the percentage of tokens that have the highest similarity to the empty target, and not an actual image patch. The more text tokens have the highest similarity to an image patch, the closer $\mathcal{L}_{\text{Reg}}$ will be to zero, and vice versa. This forces the model to utilize the empty target for as few tokens as possible.

Increasing CLS Weight

Bibliography

[1] L. Yao *et al.*, "FILIP: Fine-grained Interactive Language-Image Pre-Training," *CoRR*, 2021.

[2] W. Wang *et al.*, "Image as a Foreign Language: BEIT Pretraining for Vision and Vision-Language Tasks," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 19175–19186. doi: 10.1109/CVPR52729.2023.01838.