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.

$$m_k^{\text{i2t}} = \mathop{\text{argmax}}_{1\leq j\leq M}[\boldsymbol{v}^k]\left[\boldsymbol{w}^j\right]^T \tag{1}$$

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

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

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.

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

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

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. $s^{\text{i2t}}_{\boldsymbol{H}_l^v,\boldsymbol{H}_l^w}$ denotes the the similarity score between an image representation $\boldsymbol{H}_l^v$ and text representation $\boldsymbol{H}_l^w$. Vice versa, $s^{\text{t2i}}_{\boldsymbol{H}_l^v,\boldsymbol{H}_l^w}$ 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.

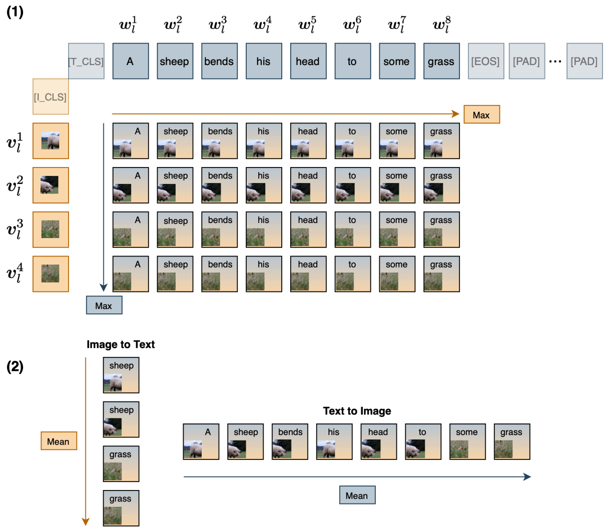

Figure 1: For a token/patch, CMLI finds the semantic timestep with the highest match from the other modality. This enables the model to associate small details of image and text with each other. Notice how through the max-operation patches containing grass are always associated with the word "grass", 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 $262,144 * 768 * 4 \text{ bytes} = 805.31 \text{ MB}$ 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 cosine similarity between all possible image-text pairs, where the cosine 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 an simplification done in FILIP [1], we need $6,422,528 * 256 * 4 \text{ bytes} = 6.58 \text{ GB}$ of additional GPU memory, just to store the result. Consequently, this approach is not feasible in our setup.

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.