CrossCLR: Cross-modal Contrastive Learning For Multi-modal Video Representations
- Supplementary Material-

Mohammadreza Zolfaghari[1*], Yi Zhu[2], Peter Gehler[2], Thomas Brox[2]
[1]University of Freiburg [2]Amazon

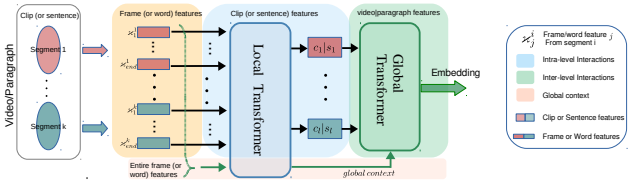

Figure 1. **Overview of architecture** (best viewed in color). We use the same architecture as COOT [1] which consist of two branches: one for video input and one for text input. Since both streams have same design, we here only show one branch. After encoding video data and it's corresponding text, we fed them to local transformer to obtain clip/sentence level features. Then, global transformer aggregates the clip/sentence level features and produces video/paragraph features. The CrossCLR loss is applied to both local (clip/sentence) and global (video/paragraph) features.

1. Architecture

A video v_i is represented as a collection of consecutive clips $vi = [c_{i1}, c_{i2}, \ldots, c_{il}]$, and similarly we define paragraph p_i as a list of sentences $p_i = [s_{i1}, s_{i2}, \ldots, s_{il}]$. Each pair video v_i and paragraph p_i and also their clips (c_{ij}) and sentences (s_{ij}) are considered temporally aligned.

We first apply pre-trained visual encoders (appearance, object, etc) to extract per-frame features, where a clip c_{ij} contains t frames. For the sentence s_{ij} with h words, we utilize *Bert-Based uncased* model to extract features. As explained in Section 4.3, for Youcook2 we use HowTo100m pre-trained model [1] to encode video frames.

Note that our architecture is same as COOT. However, We remove all losses used in COOT and only apply CrossCLR in two points. In the architecture 1, we apply CrossCLR in two locations: clip and sentence features obtained after local transformer; video and paragraph features obtained after global transformer. In all experiments, we train the model with the following objective: $L_{local} + 0.6L_{global}$. For L_{local} we use CrossCLR with queue (hyper-parameters in Table 1) and for L_{global} we use the CrossCLR without queue. Pseudo-code of CrossCLR without queue in Algorithm 1 is shown in PyTorch style.

2. Hyper-parameters

We list the hyper-parameters and ranges used during training. We largely follow prior work [1] for architecture hyper-parameters and for the rest we tuned hyper-parameters based on the performance on validation set. In general, we found retrieval performance to increase with larger queue capacity. In Table 1, we report the hyper-parameters used in the experiments.

Table 1. **Hyperparameters.** This table shows the hyperparameter ranges we considered and the final choices for LSMDC and YouCook2 datasets. First block shows the hyperparameters for optimizer, second block provides the architecture setting and last block shows the hyperparameters for our CrossCLR loss. AF is our Attention-aware Feature Aggregation module.

Hyperparameter	Range	LSMDC	Youcook2
Optimizer	RAdam	RAdam	RAdam
Learning rate	5e-5 - 1e-3	7e-4	7e-4
Weight Decay	0	0	0
Momentum	0.56	0.56	0.56
Warmup Epochs	4	4	4
Reduce on Plateau-Patience	6	6	6
Reduce on Plateau-Cooldown	4	4	4
Attention Layers	1	1	1
Attention Dimension	[384, 768]	768	384
Attention Heads-Local	[4, 8, 16]	16	8
Attention Heads-Global	1 - 8	8	8
AF Heads	[4, 8]	-	2
Pooler	[Max, ATN]	Max	ATN
Dropout	1% - 10%	1%	5%
Temperature τ	0.03	0.03	0.03
Weight scale κ	1e-5 - 1	55e-4	35e-4
Intra-modality weight λ	0.1-1	65e-1	8e-1
Pruning threshold γ	0.1-1	0.9	0.9
Queue size	1,000-10,000	3,000	5,000

2.1. Effect of Pruning Threshold

Figure 4, presents a graph visualization of the Youcook2 dataset embeddings. As can be seen, some samples have dense connections to other samples which means those samples are semantically similar to many samples. Therefore, it's critical to remove these highly similar samples from the negative set to prevent semantic collision, as discussed

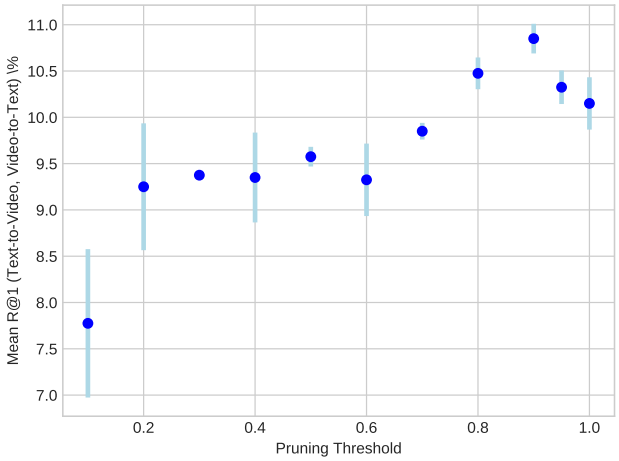

Figure 2. **Impact of Pruning Threshold**. We evaluate the effect of pruning threshold on the retrieval performance for LSMDC dataset using appearnce and action features.

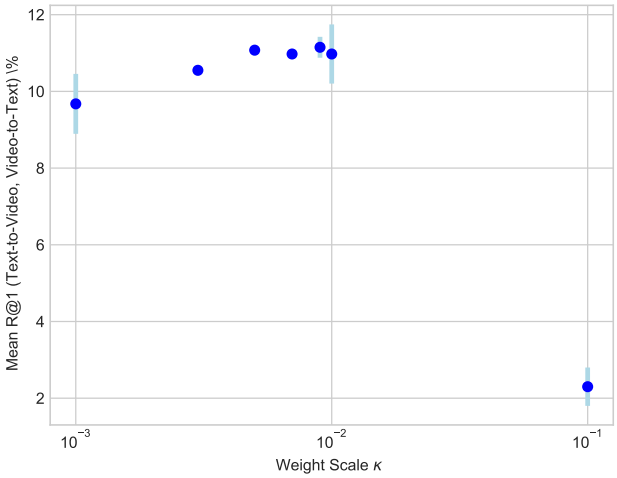

Figure 3. **Impact of Weight Scale** κ. CrossCLR is stable against small changes in κ.

in section 3.2 of paper. We change the pruning threshold from $1e1$ to 1 and show the results in Figure 2.1. To prune samples from the negative set, we first divide all scores in each set by the maximum score and then remove the samples which have score above the threshold. Therefore, increasing threshold means removing samples with higher similarity but keeping the other samples which have scores less than threshold. We consider thresholds $\gamma \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ and $\gamma = 1$ means no pruning. As shown in Figure 2.1, increasing the threshold results in higher performance until $\gamma = 0.9$ and after that again performance decreases. We found that for both LSMDC and Youcook2 datasets $\gamma = 0.9$ is a reasonable number.

2.2. Effect of Weight Scale

Figure 2.2, shows the retrieval performance with different κ values. We train three models for each $\kappa \in \{0.1, 0.01, 0.001, 0.003, 0.005, 0.007, 0.009\}$. When weight scale is too high, CrossCLR does not converge very well. But with numbers around 0.003-0.005 model performs very well.

3. Experiments

3.1. Do multiple positives in CrossCLR help?

In this section, we empirically asses the effect of having multiple positives in objective function. We define our CrossCLR loss with multiple positive samples as following:

$$L_x = -\mathbb{E}_{i \in \mathcal{M}} \left[w(x_i) \log \frac{\delta(x_i, y_i) + \beta \sum\limits_{y_k \in P_y} \delta(x_i, y_k)}{\delta(x_i, y_i) + \sum\limits_{y_k \in \hat{\mathrm{N}}_x^E} \delta(x_i, y_k) + \lambda \sum\limits_{x_k \in \hat{\mathrm{N}}_x^R} \delta(x_i, x_k)} \right] \quad (1)$$

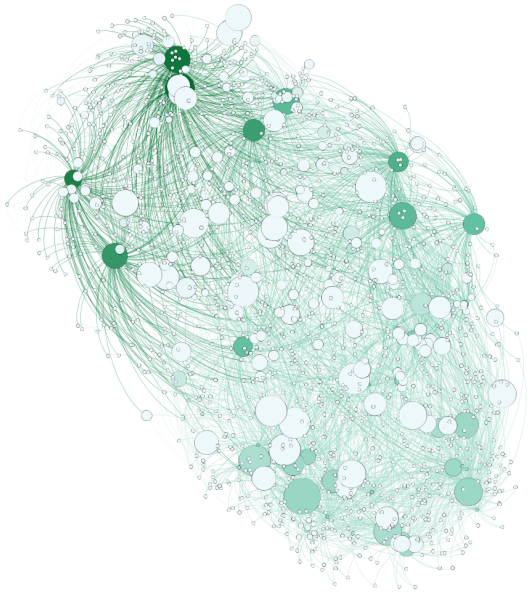

Figure 4. Graph representation of connections among samples in Youcook2 dataset. Each sample is connected to another sample if their feature similarity is above a certain threshold. Influential samples are densely connected to other samples and therefore share similar features with many other samples.

$$L_y = -\mathbb{E}_{i \in \mathcal{M}} \left[w(y_i) \log \frac{\delta(y_i, x_i) + \beta \sum\limits_{x_k \in P_x} \delta(y_i, x_k)}{\delta(y_i, x_i) + \sum\limits_{x_k \in \mathrm{N}_y^E} \delta(y_i, x_k) + \lambda \sum\limits_{y_k \in \mathrm{N}_y^R} \delta(y_i, y_k)} \right] \quad (2)$$

Where β is a scaling factor for extra postive samples, P_x and P_y are positive samples for L_x and L_y respectfully. To construct the positive set, we select the top-K similar samples to the pivot sample among the influential samples. To be precise, $P_x = \{x_k | k \in topK(\mathcal{I}_y)\}$ and $P_y = \{y_k | k \in topK(\mathcal{I}_x)\}$. The results are shown in Table 2.

In this experiment, for Youcook2 dataset we found that using $K = 2$ and $\beta = 0.15$ works best. For LSMDC dataset, we used $K = 5$ and $\beta = 0.2$. CrossCLR+MP on Youcook2 performs similar to the CrossCLR without multiple positives. However, on LSMDC we observe improvement when multiple positives are used. LSMDC dataset is larger than Youcook2 dataset and therefore it's easier to find very close positive samples to the original sample. Note that, removing influential samples from the negative set reduces the effect of pushing away samples with common semantics. But for multiple positives case, optimization tries to align positive samples and therefore it requires samples to be semantically very close otherwise alignment is wrong.

3.2. Impact of different modality combinations:

We study the importance of different modality experts and their combinations on representation learning in Table 3. Using stronger features result in higher performance. The object expert modality gives the lowest performance and action modality produces the best performance in single modality setting. Additionally, we compare the retrieval performance with different combinations of modalities. To combine several modalities, we simply concatenate the features. In addtion, we also trained the network with combining two modalities and inputing them to network (Table 3 second block from top shown with $+$). In this paper, we don't study the architecture design for combining multiple modalities and therefore leave this question for future studies.

Challenges with Object Features: In Table 3, the object features provide lower performance in comparison to scene or action experts. The reason is that current object detection models are not trained on real-world or large scale movie datasets. Therefore these models cannot detect objects very well due to domain shift or high variations in visual features and high number of objects. In our experiments, we used a Faster RCNN [2] model and used all objects detected with score above 0.7. In Figure 5, we visualized some of challenging cases in object detection.

4. Qualitative Results

Figure 6, shows two qualitative examples for video captioning on Youcook2 dataset (samples are selected randomly for a fair comparison with COOT and MART). In Figure 7, we visualize the video frames in the embedding space based on distances between text embeddings for Youcook2 dataset. Two videos are close to each other if their text features are similar to each other. We use t-SNE and project the text embeddings to 2D space and then visualize each point with it's corresponding video frame. As can be seen, the text embeddings are clustered semantically and aligned with visual semantics.

Figure 5. **Challenges in Object Detection for Movie Videos**. We show several qualitative examples of failure in object detection. No labeling: model cannot detect any object in the frame while there is an object, Over labeling: Labeling same object several times, Wrong labeling: Wrong category for the detected object, and Unknown objects: Since some objects are not labeled (or doesn't exist) in the training set of object detection model, these objects that cannot be recognized by the model.

Algorithm 1 Pseudocode of CrossCLR (no queue version) in a PyTorch style.

bmm: batch matrix multiplication; mm: matrix multiplication; cat: concatenation.

```
# enc_v, enc_t: encoder networks for video and text
# in_v, in_t: input video and text embeddings (BxD)
# B: batch size
# t: temperature, n_keep: 1-(pruning_percentage)
# t_w: loss weightening temperature
# mask = 1-np.eye(B)
# def norm_w(x): return x/sum(x)
# def mean_w(x,w,t): return sum(x)/(sum(exp(w/t)))

# Encode input video and text embeddings
emb_v, emb_t = enc_v(in_v), enc_t(in_t)

# Compute positive and negative logits
l_vt = mm(emb_v,emb_t.t())/t #video to text
l_tv = mm(emb_t,emb_v.t())/t #text to video
l_vv = (mm(emb_v,emb_v.t())/t)*mask #video to video
l_tt = (mm(emb_t,emb_t.t()/t)*mask #text to text

# Compute proximity of semantics
prox_vid = mean(mm(in_v, in_v.t()) * mask)
prox_txt = mean(mm(in_t, in_t.t()) * mask)
scores_v = prox_vid / max(prox_vid)
scores_t = prox_txt / max(prox_txt)

# Prune samples from intra-modality negative set
l_vv_p = l_vv[:, scores_v<threshold]
l_tt_p = l_tt[:, scores_t<threshold]

# Prune samples from inter-modality negative set
l_vt_p = l_vt[:, scores_v<threshold]
l_tv_p = l_tv[:, scores_t<threshold]

# Concatenate positive and negative logits
l_vtv = cat([l_vt_p, t_w * l_vv_p], dim=1)
l_tvt = cat([l_tv_p, t_w * l_tt_p], dim=1)

# compute the loss. Crossentropy without reduction
labels = arange(l_vt.shape[0])
loss_vtv = cross_entropy_loss(l_vtv, labels)
loss_tvt = cross_entropy_loss(l_tvt, labels)

# Weight losses based on semantic proximity
w_vtv = norm_w(prox_vid)
w_tvt = norm_w(prox_vid)
loss_vtv = loss_vtv * exp(w_vtv / t_w)
loss_tvt = loss_tvt * exp(w_tvt / t_w)

loss = (mean_w(loss_vtv, w_vtv, t_w) +
        mean_w(loss_tvt, w_tvt, t_w) ) /2
```

References

[1] Simon Ging, Mohammadreza Zolfaghari, Hamed Pirsiavash, and Thomas Brox. Coot: Cooperative hierarchical transformer for video-text representation learning. In *Advances on Neural Information Processing Systems (NeurIPS)*, 2020. 1

Table 2. **Effect of Multiple Positives in CrossCLR Loss.** Comparison between our CrossCLR with multiple positives (MP) and standard CrossCLR which does not have multiple positives. For LSMDC experiment, we use appearance and action features together.

	Youcook2						LSMDC					
	Text ⟹ Video			Video ⟹ Text			Text ⟹ Video			Video ⟹ Text		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
CrossCLR	$19.5_{\pm 0.49}$	$45.9_{\pm 0.55}$	$58.3_{\pm 0.76}$	$18.5_{\pm 0.32}$	$44.8_{\pm 0.82}$	$57.9_{\pm 0.77}$	10.9	26.2	34.7	12.0	26.1	35.3
CrossCLR+MP	$19.3_{\pm 0.51}$	$45.3_{\pm 0.85}$	$58.2_{\pm 0.77}$	$18.4_{\pm 0.45}$	$44.4_{\pm 0.61}$	$57.6_{\pm 0.81}$	11.2	26.6	35.8	12.9	27.4	36.4

Table 3. **Performance of different modality combinations.** + We feed both modalities to network. ⊕ we train the network on each modality separately and then output embeddings are concatenated.

	Text ⟹ Video				
	R@1	R@5	R@10	MdR↓	MnR↓
Random	0.1	0.5	1.0	500.0	500.0
Object	2.1	7.7	14.3	108.0	187.3
Scene	5.9	18.2	24.9	57.0	132.4
HowTo100M	6.4	18.9	26.4	44.0	115.0
Appearance	9.1	22.8	32.0	41.0	120.4
Action	9.3	22.3	30.7	36.0	112.3
Action+Scene	8.8	25.9	34.3	25.0	88.8
Action+HowTo100M	9.6	24.1	35.1	26.0	86.7
Action+Object	9.7	23.9	31.7	31.0	102.8
Action+Appearance	10.9	26.2	34.7	27.0	91.0
Scene ⊕ Object	6.0	17.8	24.8	55.0	130.6
HowTo100M ⊕ Object	6.4	19.4	27.3	44.0	114.8
Appearance ⊕ Object	8.2	22.7	29.4	44.0	121.1
Scene ⊕ HowTo100M	8.7	23.6	30.1	37.0	102.7
Action ⊕ Object	9.0	23.1	31.4	35.0	109.3
Action ⊕ HowTo100M	10.1	26.3	36.0	24.0	90.4
Scene ⊕ Appearance	10.2	24.2	32.0	39.0	111.0
HowTo100M ⊕ Appearance	10.5	25.5	34.4	31.0	97.3
Action ⊕ Scene	11.0	24.9	34.3	30.0	96.9
Action ⊕ Appearance	12.0	27.7	36.2	24.0	92.4
Scene ⊕ Object ⊕ HowTo100M	8.0	22.4	30.8	37.0	106.0
Object ⊕ Appearance ⊕ HowTo100M	9.4	26.5	34.1	34.0	101.1
Scene ⊕ Appearance ⊕ Object	9.2	23.3	31.9	38.0	112.4
Action ⊕ Scene ⊕ Object	10.0	25.2	33.6	32.0	100.1
Scene ⊕ Appearance ⊕ HowTo100M	10.8	26.8	34.4	32.0	97.3
Action ⊕ HowTo100M ⊕ Object	11.3	26.4	34.2	27.0	93.1
Action ⊕ Scene ⊕ HowTo100M	11.8	27.2	36.5	23.0	88.2
Action ⊕ Appearance ⊕ Object	12.4	27.5	35.7	28.0	95.7
Action ⊕ Appearance ⊕ Scene	12.6	27.2	36.7	24.0	91.5
Action ⊕ Appearance ⊕ HowTo100M	13.4	28.4	37.8	22.0	84.9

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 3

MART: Add tomatoes and beef to a pot. Add water to the pan. Add tomato puree and salt. Add the beef and parsley to the soup. Add the beef to the pot. Add water to the soup and let it simmer. Add the soup to the soup.

COOT: Add the tomatoes and onions to a food processor and blend them. Add the tomatoes and a bay leaf to the pot. Add the tomatoes and simmer. Remove the tomatoes from the pot and let it cook. Remove the tomatoes from the pot and let it cook. Strain the soup to a boil and let it boil. Turn on the heat and heat to a boil.

CrossCLR: Add chopped tomatoes and onions to a pot. Add water to the pan. Add the tomatoes and cover to cook for 5 minutes. Drain the tomatoes and let the liquid cool. Add chopped tomatoes to the pot and simmer. Strain the tomato mixture to a blender and blend the soup. Heat the pot on a stove and let it boil.

GT: Add tomato onion green chili and rice to a pan. Add water to the pan. Boil the ingredients and then turn down the heat. Strain the ingredients. Blend the ingredients. Add the water to the mixture and strain. Boil the soup.

MART: Add flour to a bowl and whisk. Cut the chicken into pieces. Coat the chicken in flour. Coat the chicken in flour egg and breadcrumbs. Fry the chicken in a pan. Drizzle the sauce on top of the bread. Add sauce to the pizza. Bake the dish in the oven.

COOT: Mix parmesan cheese black pepper and garlic powder. Cover the chicken in the bag. Coat the chicken in the flour. Coat the chicken in the egg and coat with flour. Place the chicken in a pan and fry it on a pan. Pour sauce on top of the chicken and top with mozzarella cheese. Sprinkle parmesan cheese on top. Bake the chicken in the oven.

CrossCLR: Mix flour salt pepper and cayenne pepper. Pound the chicken. Coat the chicken in the egg. Coat the chicken in the egg and bread crumbs. Fry the chicken in a pan. Spread tomato sauce and mozzarella cheese on the chicken. Sprinkle cheese on top. Bake the dish in the oven.

GT: Mix bread crumbs and parmesan cheese. Pound the chicken. Rub salt and pepper onto the chicken. Rub flour onto the chicken dip it in egg and coat with breadcrumbs. Fry the chicken in a pan. Spread sauce over the chicken. Top the chicken with mozzarella cheese. Bake the chicken in the oven.

Figure 6. **Captioning samples for Youcook2 dataset**. We randomly selected two samples for video captioning to have a fair comparison with COOT and MART methods. Green: correct captioning, Yellow: Ok but not accurate, and Red: wrong captioning.

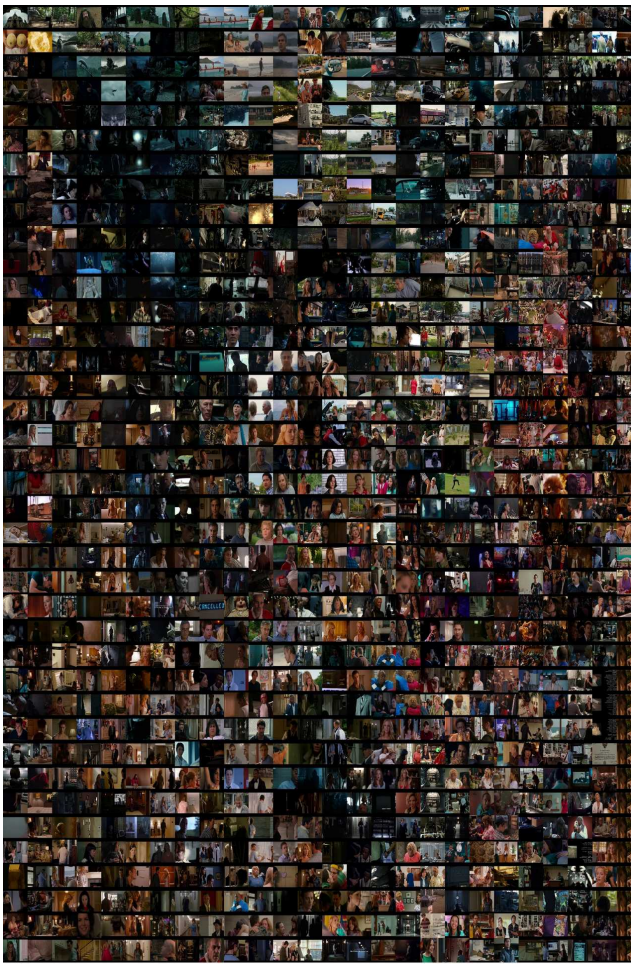

Figure 7. **Visualization of the video frames based on their text embeddings**. We use t-SNE to project text embeddings to 2D space and then present each point with it's corresponding video frame.