MDMMT: Multidomain Multimodal Transformer for Video Retrieval Supplementary materials

Maksim Dzabraev[1,2], Maksim Kalashnikov[1], Stepan Komkov[1,2], Aleksandr Petiushko[1,2]
[1]Lomonosov Moscow State University
[2]Huawei Moscow Research Center
dzabraev.maksim@intsys.msu.ru, kalashnikov.maxim@intsys.msu.ru,
stepan.komkov@intsys.msu.ru, petyushko.alexander1@huawei.com

A. Pretrain Experts Usage

The important data preparing stage is how to sample frames from a video to achieve the best performance. For s3d experiments the input video is converted to 30 frames per second, for all other experiments we convert the input video to 32 frames per second. As a result we compute a single embedding for each second, having 1 second window with 1 second shift (no overlapping).

The input frame size is important. We use the different sizes for the different models. For each model we use the recommended input size. For s3d we resize a video to 256 on the short side and then take a 224x224 center crop. For SlowFast 32x2 R101 we resize a video to 256 on the short side and then take a 256x256 center crop. For ipCSN 152 and irCSN 152 we resize a video to 224 on the short side and take a 224x224 center crop. For r(2+1)d 152 and r(2+1)d 34 we resize a video to 112 on the short side and then take a 112x112 center crop.

Pretrained models for ipCSN, irCSN and r(2+1)d are available here[1], for SlowFast 32x2 R101 here[2], and for s3d here[3].

For the CLIP model [15] we resize a video to 224 on the short side and take a center crop, then we extract 1 frame per second. We use a publicly available image encoder. We do not use the text encoder from CLIP.

Model s3dg MIL-NCE is a video encoder from the work [9]. This network is trained from scratch on HowTo100M dataset. For this network we resize the input video stream to the size of 228x228 pixels, then take a center crop.

B. Datasets Combination

In Fig. 1,2,3 we present 6 models. Abbreviations M_cALV, M_cALVYMTS and M_cALVYMTS represent the same three models on these figures. The first model, called M_c, is trained on the MSRVTT full clean split only, the second one, called A, is trained on ActivityNet only. And the third model, called L, is trained on LSMDC only. These three models are taken as baselines. Adding more datasets should be not worse than these baseline. The forth model is called M_cALV. This model is trained on the combination of MSRVTT, ActivityNet, LSMDC and TwitterVines. As we can see $M_c \rightarrow M_c$ALV gives +3.07% on MSRVTT (full clean split), A$\rightarrow M_c$ALV gives +1.06% on ActivityNet, and L$\rightarrow M_c$ALV gives +1.77% on LSMDC. The next model is called M_cALVYMT and it is trained on combination of MSRVTT, ActivityNet, LSMDC, TwitterVines, YouCook2, MSVD, TGIF. The transitions $M_c \rightarrow M_c$ALVYMT, A$\rightarrow M_c$ALVYMT, L$\rightarrow M_c$ALVYMT give +4.85%, +1.45% and +2.63% correspondingly. The last transitions $M_c \rightarrow M_c$ALVYMTS, A$\rightarrow M_c$ALVYMTS, L$\rightarrow M_c$ALVYMTS slightly improve the performance on ActivityNet and LSMDC and significantly improve the performance on MSRVTT. Finally, the combination of all datasets gives +5.5% for MSRVTT, +1.47% for ActivityNet and +2.74% for LSMDC.

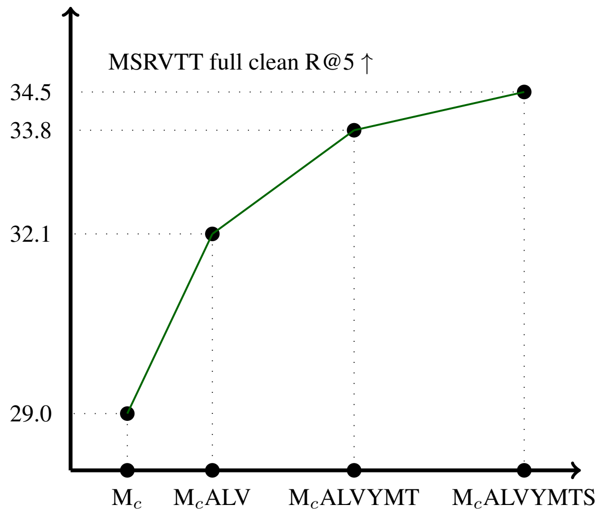

Figure 1: Increasing R@5 metric on the MSRVTT full clean split while enriching the train part.

1. https://github.com/facebookresearch/VMZ
2. https://github.com/facebookresearch/SlowFast/blob/master/MODEL_ZOO.md
3. https://github.com/princeton-vl/d3dhelper/blob/master/d3d_helper.ipynb

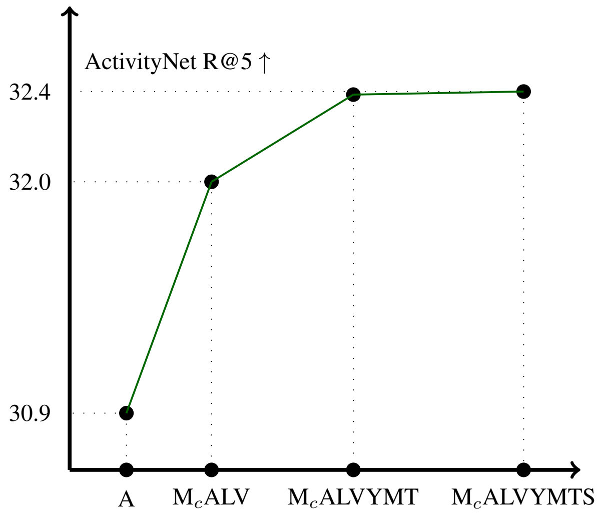

Figure 2: Increasing R@5 metric on the ActivityNet test set while enriching the train part.

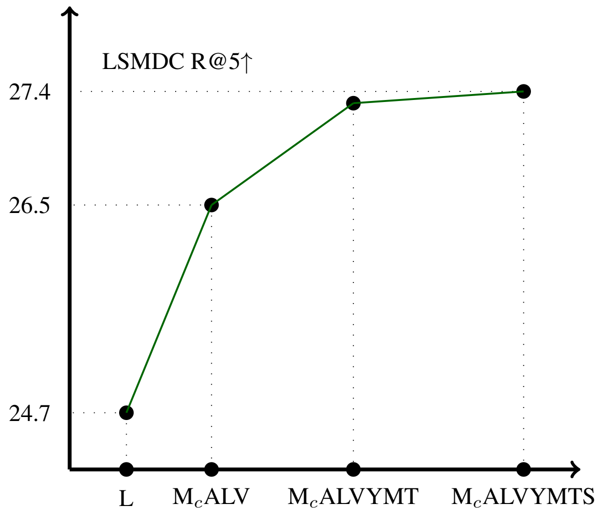

Figure 3: Increasing R@5 metric on the LSMDC test set while enriching the train part.

C. Test and Train Intersection

In this section we present our analysis of overlapping of popular text-to-video datasets. Since we compose the train dataset from several different datasets it is important to be sure that there is no the same video segment in the train part and in the test part. Our aim is to find the overlap between the train part of used datasets — MSRVTT, ActivityNet, LSMDC, YouCook2, MSVD, TGIF, TwitterVines, HowTo100M, Kinetics700 and the test parts of MSRVTT, ActivityNet and LSMDC, and then to remove found duplicates from the train parts.

Note that for training we use Something to Something V2 dataset, but we do not try to find overlap between it and test datasets because this dataset is artificially created, thus the probability to find duplicates is very low.

We decided to find the overlap only for MSRVTT, ActivityNet and LSMDC because these are the most popular datasets and we do not have enough human resources to find the overlap for the test part of all other datasets.

Our cleaning method consists of two stages. The first stage is to match video segments by the YouTube ID (if the ID is available) and remove from train parts all video segments that have the corresponding pair in test parts. In Main Article Tab. 1 the information about the availability of YouTube IDs in datasets is presented. We collect the YouTube ID for all videos from MSRVTT full test and ActivityNet validation 1,2 and remove corresponding video segments from the train part.

The second stage is based on matching frames by embeddings. For each video we compute several embeddings then we compute the similarity between each video from the train part and the test part. After we manually assess several thousands of video segments with highest scores for each pair of datasets. Then we extend found duplicates by either the YouTube ID or the internal dataset ID. This means that if a video V_1 is marked as a duplicate and a video V_2 is not marked as a duplicate, but they have the same YouTube ID or same internal dataset ID, we will remove V_1 and V_2 from the train part. In case of LSMDC we do not have the YouTube ID, but have the name of the movie from which the video segment is taken, so if a video segment V_1 is marked as a duplicate, we remove all segments taken from the movie of V_1. The detailed description of the second stage is described in Sec. C.1.

Surprisingly we found that the MSRVTT test has a significant overlap with the MSRVTT train part. This problem is relevant for the full, 1k-A and 1k-B splits. The ActivityNet dataset suffers from the same problem.

For large datasets like HowTo100M and Kinetics700 we can not find the whole intersection, but we estimate the approximate number of videos in the intersection. We found that HowTo100M may have about 300 (10% of the MSRVTT full test part) video segments that can be in the MSRVTT full test part.

The similar situation is about Kinetics700 and ActivityNet datasets. Kinetics700 may have approximately 500-600 video segments (10% of the ActivityNet test) that may have duplicates in ActivityNet validation 1,2. Another problem with the Kinetics dataset is that many motion models are pretrained on it.

This circumstance means that researchers should carefully use HowTo100M and Kinetics700 along with MSRVTT and ActivityNet correspondingly, because for today we do not know whether a neural network overfits for some portion of this intersection or not.

All duplicates can be considered as two groups of pairs. Pairs from the first group have the same videos, but different brightness, aspect ratio, size, presence/absence of a logo and so on. The second group has pairs with quite similar videos, for example it can be the same person on the same background, doing the same things, but wearing different clothes. We think that it is better to remove such videos from the train part to prevent overfitting. Several found examples are presented in Fig. 4.

C.1. Near duplicate video search

C.1.1 Approach

In this section we explain our approach that is used to find the same or quite similar video segments in test and train parts.

Suppose we have two sets of videos $Q = \{q_1, ..., q_k\}$ and $G = \{g_1, ..., g_n\}$ called the query set and the gallery set. We want to find all pairs (q_i, g_j) where q_i and g_j have a common video segment.

From each q_i and g_j we extract 1 frame per second. Each video is then represented by a sequence of pictures: $q_i = [q_i^1, ..., q_i^{s_i}]$ and $g_j = [g_j^1, ..., g_j^{p_j}]$. Then a 2D pretrained neural network is used to extract features from each image: $\bar{q}_a^b = \mathtt{neuralnet}(q_a^b)$ and $\bar{g}_a^b = \mathtt{neuralnet}(g_a^b)$.

Then we compute the matrix of cosines between the features from Q and G: $s_{ij}^{ab} = \frac{<\bar{q}_i^a, \bar{g}_j^b>}{||\bar{q}_a^b||_2 ||\bar{g}_a^b||_2}$.

Now each pair (q_i, g_j) is represented by the matrix:

$$\begin{array}{c|ccc} & g_j^1 & ... & g_j^{p_j} \\ \hline q_i^1 & s_{ij}^{11} & ... & s_{ij}^{1p_j} \\ ... & & & \\ q_i^{s_i} & s_{ij}^{s_i 1} & ... & s_{ij}^{s_i p_j} \end{array} \quad (1)$$

Suppose that videos q_i and g_j are intersected at time moments t_q and t_g, it is naturally to assume that the next several seconds $t_q + 1, ..., t_q + K - 1$ and $t_g + 1, ..., t_g + K - 1$ $(K \le \min(s_i, p_j))$ represent the same video segment. Motivated by this fact we compute the mean cosine for each interval of K seconds (we use K=4): $S_{ij}^{t_q t_j} = \frac{s_{ij}^{t_q t_g} + ... + s_{ij}^{t_q+K-1, t_g+K-1}}{K}$. The sum in the numerator is the sum of diagonal elements started with $s_{ij}^{t_q t_j}$.

We define the intersection score between (q_i, g_j) as

$$\mathbf{S_{ij}} = \max_{\substack{a=1,...,s_i-K \\ b=1,...,p_j-K}} S_{ij}^{ab} \quad (2)$$

and the corresponding video segments as

$$(\mathbf{a}, \mathbf{a} + K), (\mathbf{b}, \mathbf{b} + K) \quad (3)$$

where

$$\mathbf{a}, \mathbf{b} = \operatorname*{argmax}_{\substack{a=1,...,s_i-K \\ b=1,...,p_j-K}} S_{ij}^{ab} \quad (4)$$

Finally we sorted all $\mathbf{S_{ij}}$ in the descending order and manually assess candidate pairs.

C.1.2 Number of Pairs to Assess

Suppose we search duplicates in datasets Q and G and we have seen N pairs with the highest scores and find M pairs with duplicates. The important question is: what is the total number of duplicates and how many percents of them have we found.

For each pair of Q and G we construct the following test procedure. The first step is to augment Q, and let us call the result of augmentation as $\hat{Q}$. To augment a dataset we apply two transformations: 1. we randomly crop a side of each video, where each side can be 70%–100% of original side length (aspect ratio can be changed); 2. we randomly shift the start of the video by a random value between 0 and 1 seconds.

Having Q, $\hat{Q}$ and G we compute sets of positive and negative scores: Pos and Neg. The Pos is the set of scores between i-th video from Q and the corresponding augmented video from $\hat{Q}$. Neg is the set of scores between each video from Q and G. Having Pos and Neg sets we can plot a curve, where x axis represents the fraction of found pairs with duplicates and Y axis represents the number of negative pairs that we need to assess to find fraction x of positive pairs, call this curve $F(x)$. We present the algorithm that computes F using Pos and Neg sets in Lst. 1. Suppose we have seen $N + M$ pairs and have found M pairs with duplicates. The total number of pairs with duplicates can be estimated as $M/F^{-1}(N)$. By the definition $F(x)$ connects the fraction of found positive pairs with the number of seen negative pairs. The value $F^{-1}(N)$ represents approximation of the fraction of found positive pairs. So if we know, that M is approximately $100 * F^{-1}(N)\%$ of positive pairs, then we can approximately compute 100% of positive pairs as $M/F^{-1}(N)$.

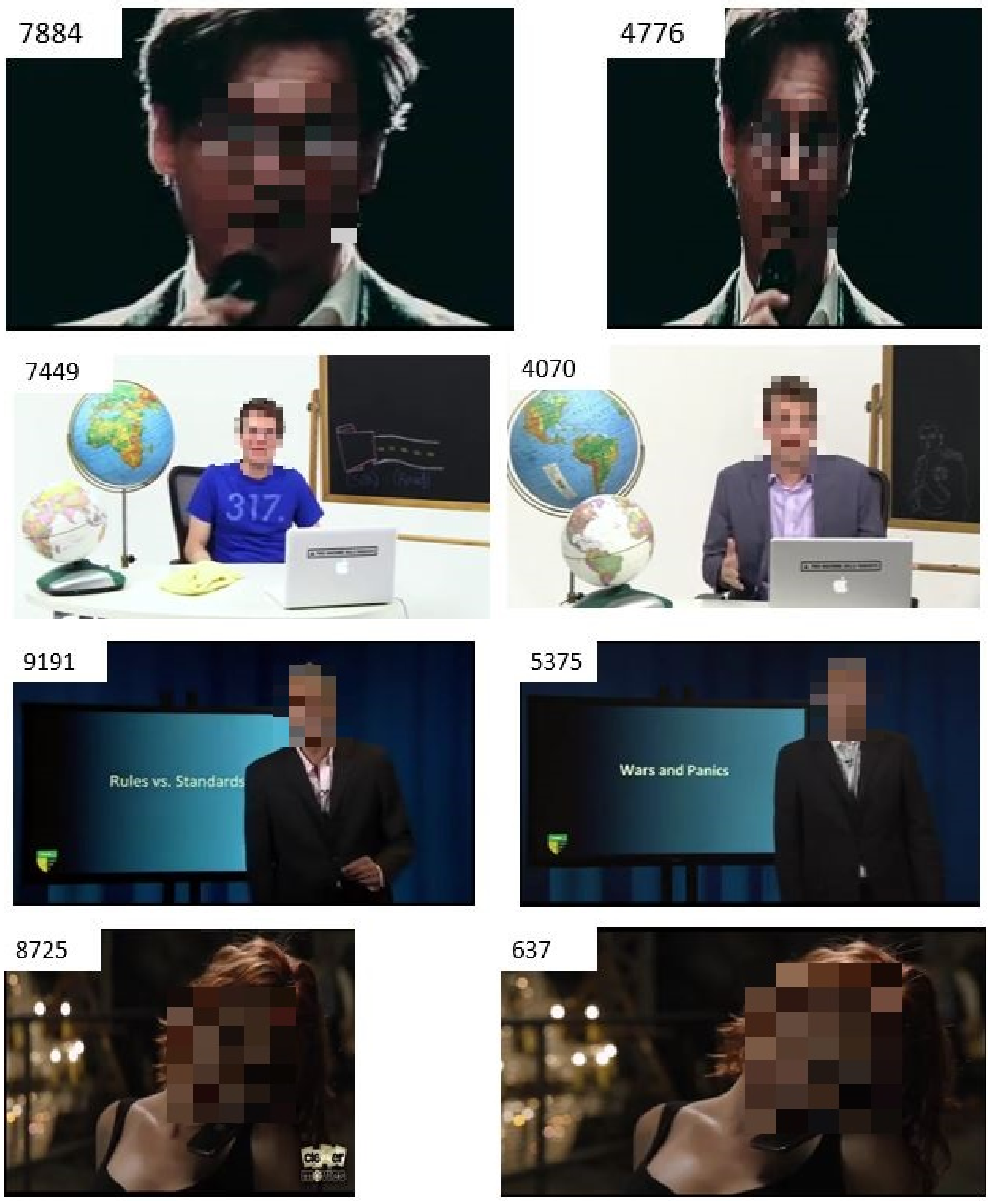

Figure 4: The left image is taken from the MSRVTT test split and the right one from MSRVTT Train. The numbers in the upper left corner represent the MSRVTT video ID. The faces are blurred in order to avoid legal claims.

```
# first element is highest
P = np.sort(P)[::-1] # Pos
N = np.sort(N)[::-1] # Neg
xs = []
ys = []
for x, p in enumerate(P):
        # how many negative scores
        # greater than p ?
        j = np.searchsorted(N, p)
        xs.append(x)
        ys.append(j)
```

Listing 1: Numpy pseudocode for building the search curve $F(x)$

C.1.3 Best 2D Feature Extractor

The key component of a duplicate search system is a feature extractor. A good feature extractor significantly reduces the number of pairs for manual assessment. To compare different 2D feature extractors we use the following test procedure. The test consists of two datasets. The first dataset is the train part from the MSRVTT full split. The second dataset is random 596k videos from the HowTo100M dataset. From each video of the taken part of HowTo100M we take a random 30 seconds segment. We apply random augmentation to MSRVTT, as described in Sec. C.1.2. Define MSRVTT as Q, the augmented MSRVTT dataset as $\hat{Q}$ and the taken part of HowTo100M as G. For each feature extractor we compute curve $F(x)$, as described in Sec. C.1.2.

The best expert has the lowest curve. For example, if we want to find 95% of duplicates, we should see many of candidates, some of them are duplicates, but majority of them are not. So, the value $F(0.95)$ is the approximation of how many not duplicates we need to see to find 95% of duplicates. Ideally $F(0.95) = 0$, where all seen candidates are duplicates. So, a lower value $F(0.95)$ requires to see less number of false candidates, that is why the lower curve is better.

We consider several feature extractors: resnet18 and resnet101 [5] pretrained on ImageNet [1], resnet50 pretrained on Places365 [20] and resnext101-32x8d, resnext101-32x32d, resnext101-32x48d pretrained on one billion images from Instagram [8] and finetuned on ImageNet. We report search curves $F(x)$ for these pretrained networks in Fig. 5.

There exist networks [15] [6] trained especially for match the duplicate frames or video segments, but they are not publicly available.

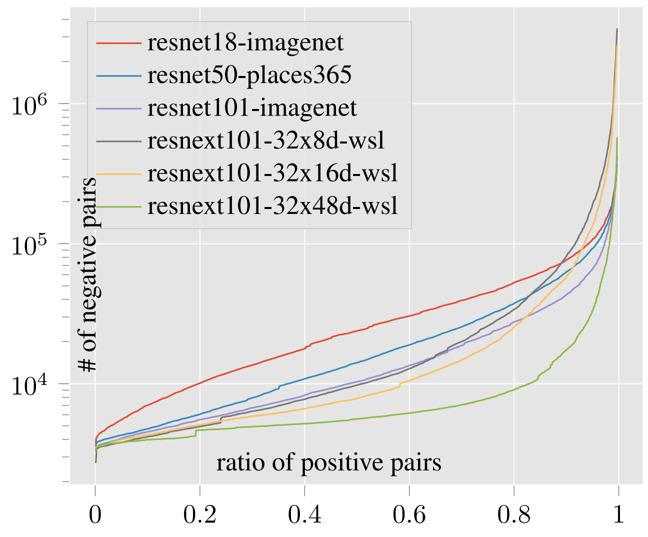

Figure 5: Search curves F for different pretrained models. Curve F is used to estimate the minimal number of negative pairs (y = $F(x)$) that human assessors need to inspect before they find the fraction x of positive pairs. The lower the curve F the better (need to inspect manually less pairs). The curves are built with the query set Q = MSRVTT full train, the gallery set G = random 596k videos from HowTo100M.

As we see resnext101-32x48d-wsl shows the best result. We use this network for searching for duplicates.

It is worth to mention that here we just compare different networks on a fixed benchmark, and pick the best one. But the search curve $F(x)$ significantly depends on data. This curve should be estimated for each used pair of datasets Q and G.

C.1.4 Black Frames

Often two consecutive video segments are glued with several black frames. The cosine similarity of embeddings of two black or near black frames are close to 1. In this case the most probable candidates for duplicates are black video segments. To prevent this we apply the following rule. Suppose we have a frame U and the unit length embedding v computed from U. We find the prevalent color in U and compute the area S_0 filled by this color. Then we compute the value $S_0/(hw)$, where h and w are the height and width of U. If this fraction is greater than 0.7 we define $\mu_v = 1 - S_0/(hw)$, otherwise $\mu_v = 1$. To calculate similarity between embeddings v_1 and v_2 we use weighted cosine similarity: $\mu_1 \mu_2 \text{cos} v_1, v_2$, instead of classical cosine similarity. This rule removes majority of all near black frames from the most relevant candidates for duplicates.

C.1.5 Screensavers Detection

Many videos from ActivityNet, HowTo100m, YouCook2 contain screensavers at the beginning or at the end. It causes a problem like mentioned above with near black frames, because most of relevant proposals are the same screensavers, but the video content of the remainder video part are different.

Using the system described in Sec. C.1.6 we search for duplicates in the ActivityNet dataset, where a lot of the most relevant segments are screensavers. We collect several hundreds of screensavers and then compute embeddings for each of them. Let us call the resulting set of embeddings as E. Then we apply the following rule: if some embedding v has the similarity greater that 0.9 to one of embeddings from E, we set $v = 0$. So if the video segment has a part of a screensaver, it will never be in the most relevant proposals.

C.1.6 GUI

The important part of the video duplicate search system is the user interface. Without ergonomic and fast interface it is impossible to assess tens thousands of video pairs. Our system is presented in Fig. 6.

The system shows video pairs with the highest scores on top. A user needs to scroll down a web page (new videos are loaded dynamically with ajax), and if a video duplicate is detected, a user should press the *Duplicate* button, if there are no duplicates in the current viewport, no action is required. When a user scrolls a web page, all non-duplicate

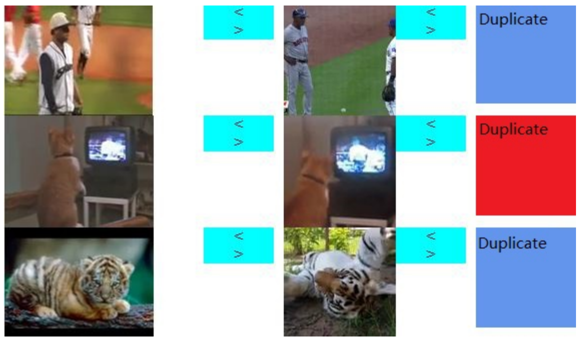

Figure 6: Web system used to find duplicates. Images on the first and third row are not duplicates and the second row contains duplicate.

pairs automatically are saved to a log file. Additionally several users at the same time can assess video pairs.

dataset	M		A		L	
	test	train	test	train	test	train
M	114	223	6	10	0	0
A	10	6	127	163	0	0
L	6	2744	0	0	0	0
YouCook2	13	27	7	10	0	0
MSVD	1	1	1	1	1	1
TGIF	6	8	0	0	0	0
Twitter Vines	3	3	0	0	0	0
Kinetics700	4	5	456	464	0	0
HowTo100M	177	154	209	209	0	0

Table 1: The leftmost column represents training parts of datasets, and the upper row represents test parts of datasets. Column ”test” means how many video segments are in the test part that have the corresponding pair in the training part either with the same YouTube ID or manually marked as a duplicate. Column ”train” represents the number of video segments in the training part that have corresponding pair in the test dataset either with the same YouTube ID or manually marked as a duplicate. All segments counted in the ”train” column are removed from the training part. For example consider the column ”A” and the row ”M”. train=10 means that the MSRVTT training part contains 10 video segments that have a pair in the ActivityNet test part. These 10 videos are removed from training part when dataset are combined. test=6 means that ActivityNet test part has 6 video segments that have a pair in the MSRVTT training part.

C.2. Cleaning Results

Recall that our cleaning method consists of two stages. In the first stage we throw out from the train part all video segments that have a pair with the same YouTube ID in test parts of MSRVTT or ActivityNet. The second stage is matching video segments by embeddings and manually assess several thousands pairs with the highest score.

In Tab. 1 we report how many duplicates are found for each pairs of datasets. This table represents the final result after applying these two stages.

Separate results for the first and the second stages are reported in Sec. C.2.1.

Note that columns ”test” and ”train” in Tab. 1 may have different values. Consider the situation when the test part have a video segment A, and the train part have two video segments A1 and A2. And both are marked as duplicates with A. In this case the video segment A brings +1 to the ”test” column and A1, A2 bring +2 to the ”train” column.

The most problematic datasets in terms of the number of duplicates are MSRVTT and ActivityNet. These datasets overlap with itself (*e.g.* MSRVTT test overlap with MSRVTT train). We found more than 100 duplicate pairs for both of them. Other problematic datasets are HowTo100M and Kinetics700, these datasets are large, so we can not assess the required number of video pairs to find 95% or 99% of duplicates. But we can assess a smaller number of pairs and using search curves F (see Sec. C.1.2) can extrapolate this value to 100%. We found that HowTo100M may have the intersection with MSRVTT test full by about 300 videos (10% of the MSRVTT test full). The similar situation is about the ActivityNet test set and Kinetics700, the intersection could be near 500-600 videos (10% of the ActivityNet test set).

In Tab. 2 we report results on MSRVTT for MMT retraining with no cleaning, after cleaning by the YouTube ID and cleaning combination by the YouTube ID and the manual assessment. The manual cleaning for 1k-A and 1k-B is incomplete because we only do cleaning for the full split. The following situation takes place for 1k-A, 1k-B splits: when 1k videos from the full test are taken for test and the remaining 2k videos are moved to the train part, the additional overlapping is introduced, because these 1k and 2k videos are overlapping. We do not remove this overlap in this research.

split	no clean	by ID	by ID + manual
full	$31.1_{\pm 0.1}$	$31.1_{\pm 0.1}$	$30.2_{\pm 0.4}$
1k-A	$54.8_{\pm 0.5}$	$50.7_{\pm 0.9}$	$49.4_{\pm 0.5}$
1k-B	$51.1_{\pm 0.9}$	$46.1_{\pm 0.1}$	$46.4_{\pm 0.6}$

Table 2: Comparison for original MMT trained (7 modalities) on MSRVTT without cleaning, with cleaning by the YouTube ID only, and with cleaning by the YouTube ID plus the manual assessment.

As you can see after cleaning the performance is significantly decreased on 1k-A and 1k-B splits for original MMT.

C.2.1 Intersection by YouTube ID and Embeddings

In Tab. 3 we report the intersection by the YouTube ID between test parts of MSRVTT (full, 1k-A, 1k-B) and ActivityNet with train parts of MSRVTT (full, 1k-A, 1k-B), ActivityNet, Kinetics700, YouCook2, HowTo100m, MSVD.

data set	M test	M train	M_{1k-a} test	M_{1k-a} train	M_{1k-b} test	M_{1k-b} train	A test	A train
M	0	0	0	0	104	179	2	4
M_{1k-a}	2362	1990	372	415	827	1007	2	4
M_{1k-b}	1689	1367	563	634	380	407	2	4
A	0	0	0	0	0	0	0	0
K	5	4	1	1	0	0	408	408
Y	8	4	2	2	2	2	3	3
HT100M	147	117	39	38	57	53	175	175
MSVD	3	1	2	1	0	0	1	1

Table 3: First stage. The leftmost column represents train parts of datasets, and the upper row represents test parts of datasets. Column "test" represents number of video segments in test part that have corresponding video in train part with the same YouTube ID. Column "train" represents number of video in train part that have corresponding pair in test part with the same ID. For example: if we combine M and YouCook2, we should remove 4 video from YouCook2 train.

It is worth to mention that MSRVTT 1k-A test and 1k-B test have a large overlap ratio by the YouTube ID with the 1k-A train and the 1k-B train parts correspondingly. Both splits have the overlap ratio of about 38% between the train part and the test part. We also emphasize that the original MSRVTT full split does not overlap by the YouTube ID between the test and train parts.

In Tab. 4 we report the statistics for the second deduplication stage (searching by embeddings). We do not compute an intersection for MSRVTT 1k-A and 1k-B splits.

In this table we present the number of manually found duplicates and the estimated maximum number of duplicates for a given pair of datasets. We managed to find the intersection for almost all pairs of datasets.

The maximum number of duplicates is computed based on the search curve $F(x)$. As we told in Sec. C.1.2 the search curve significantly depends on data. We compute the search curve for all pairs of datasets in Tab. 4. The search curve for each particular pair of datasets is build exactly in the same way as described in Sec. C.1.2. For example, to compute the search curve for MSRVTT test and ActivityNet train we define MSRVTT test as Q, ActivityNet train as G, then augment Q to produce $\hat{Q}$, and use the algorithm described in Sec. C.1.2.

data set	M seen	M found	M total	A seen	A found	A total	L seen	L found	L total
M	10k	114	114	1k	6	6	1k	0	0
A	10k	10	10	15k	127	142	1k	0	0
L	3k	6	6	2k	0	0	—	—	—
Y	2k	13	13	1k	7	7	1k	0	0
MSVD	1k	1	1	1k	1	1	1k	1	1
T	2k	6	6	2k	0	0	3k	0	0
V	2k	3	3	0k	0	0	1k	0	0
K	2k	1	2	30k	227	539	2k	0	0
HT100M	5k	15	320	—	—	—	—	—	—

Table 4: Second stage. The leftmost column represents train parts of datasets, and the upper row represents test parts of datasets. Column "seen" represents the number of video segments that we manually assess for a given pair of datasets. Column "found" represents the number of videos in the test part for which there exists the corresponding duplicate video segment in the train part. Column "total" represents the approximately estimated total number of videos from the test part that have a duplicate pair in the train part. Symbol "—" means that the intersection is not computed because it requires too much human resources.

Using the column "seen" from Tab. 4 we can compute how many pairs need to be assessed to find the full overlap between datasets. For example, inspect 5k pairs for HowTo100M dataset and MSRVTT (the row "HT100M" and the column "M"), we found 15 duplicates, so the approximate maximum number of duplicates is 320: 5k * (320 / 15) = 106k. So, to find the full overlap using the current version of algorithm it is needed to manually assess 106k video pairs and it is too much, that is why we do not find full intersection for this specific pair of datasets.

D. Hyperparameters

To train our best networks (MMT(MALVYMTS) L9H8 CLIP+audio, MDMMT(MALVYMTS) L9H8 irCSN152+audio and MMT(MALVYMTS) L9H8 CLIP+irCSN152+audio) we use 50 epochs and define a single epoch as 150K examples per GPU (in total 1.2M examples per epoch on 8 GPUs). We use Adam optimizer without weight decay, the initial value for a learning rate is 5e-5, after each epoch we multiply the learning rate by 0.95. Batch size of 32 examples per GPU is used. We do not exchange embeddings between GPUs. We use bi-directional max-margin ranking loss with margin 0.05. In Bert and the video transformer encoder we use dropout 0.2 in attention and in FFN block. We use 8 Nvidia V100

32GB GPUs. The training time is about 14 hours.

E. Pretrained Model

The well known method to boost the performance in video retrieval tasks is to use a pretrained model. First the neural network is trained on some large dataset, then at second stage it is finetuned for target target dataset. In video retrieval task HowTo100M dataset is often used for pretraining. In this work we use HowTo100M for pretraining in the same way.

In our training procedure we use 8 Nvidia V100 32Gb GPUs, we train for 200 epochs where one epoch is defined as 80k examples on each GPU (in total network sees 640k examples on 8 GPUs per epoch). We use batch size 64 for each GPU and do not exchange embeddings between GPU. Initial learning rate is 5e-5. After each epoch we multiply learning rate by 0.98. We use the full HowTo00M dataset. The model is trained either with two modalities: motion/RGB and audio or with three modalities: motion, RGB and audio, depending on how many modalities are used in final model. The total training time is about 24 hours. We use bi-directional max-margin ranking loss with margin 0.05.

In Tab. 5, 6 and 7 we compare two our models: MDMMT(M$_c$ALVYMTS) L9H8 irCSN152+audio and MDMMT(M$_c$ALVYMTS) L9H8 CLIP+audio when they are trained from the pretrained model or not. In these three tables we present the same four models (no special finetuning for the target dataset) tested on different datasets.

As we can see in Tab. 5 the pretrained model increases R1 metric by 1% and R5 by 2%. The pretrained model also increase performance on ActivityNet dataset, see Tab. 6. For R1 metric the improvement is about 2% and for R5 metric is about 4%. For LSMDC dataset, see Tab 7, we have approximately the same results with and without pretraining.

F. Results

	model	pretr	MSRVTT full clean text → video				
			R@1↑	R@5↑	R@10↑	MnR↓	MdR↓
Ours	MDMMT(M_cALVYMTS) L9H8 irCSN152+audio	yes	$15.8_{\pm 0.1}$	$38.9_{\pm 0.1}$	$51.0_{\pm 0.1}$	$76.4_{\pm 0.5}$	$10.0_{\pm 0.0}$
Ours	MDMMT(M_cALVYMTS) L9H8 irCSN152+audio	no	$14.5_{\pm 0.1}$	$36.8_{\pm 0.3}$	$48.8_{\pm 0.3}$	$82.2_{\pm 0.6}$	$11.0_{\pm 0.0}$
Ours	MDMMT(M_cALVYMTS) L9H8 CLIP+audio	yes	$21.5_{\pm 0.1}$	$47.4_{\pm 0.2}$	$59.6_{\pm 0.1}$	$57.7_{\pm 0.4}$	$6.0_{\pm 0.0}$
Ours	MDMMT(M_cALVYMTS) L9H8 CLIP+audio	no	$20.0_{\pm 0.1}$	$45.1_{\pm 0.1}$	$57.3_{\pm 0.1}$	$63.1_{\pm 0.1}$	$7.0_{\pm 0.0}$

Table 5: Performance on the MSRVTT full clean split with and without pretrained model (HowTo100m).

	model	pretr	ActivityNet text → video				
			R@1↑	R@5↑	R@10↑	MnR↓	MdR↓
Ours	MDMMT(M_cALVYMTS) L9H8 irCSN152+audio	yes	$15.1_{\pm 0.1}$	$38.3_{\pm 0.1}$	$51.5_{\pm 0.3}$	$92.4_{\pm 2.3}$	$10.0_{\pm 0.0}$
Ours	MDMMT(M_cALVYMTS) L9H8 irCSN152+audio	no	$12.0_{\pm 0.1}$	$33.7_{\pm 0.4}$	$46.3_{\pm 0.3}$	$119.9_{\pm 2.1}$	$13.0_{\pm 0.0}$
Ours	MDMMT(M_cALVYMTS) L9H8 CLIP+audio	yes	$17.7_{\pm 0.1}$	$41.6_{\pm 0.3}$	$54.3_{\pm 0.2}$	$76.0_{\pm 1.0}$	$8.3_{\pm 0.5}$
Ours	MDMMT(M_cALVYMTS) L9H8 CLIP+audio	no	$15.2_{\pm 0.3}$	$37.9_{\pm 0.3}$	$50.1_{\pm 0.2}$	$93.4_{\pm 2.0}$	$10.3_{\pm 0.5}$

Table 6: Performance on ActivityNet with and without pretrained model (HowTo100m). The performance reported for the text-to-video retrieval task on our own subset of the original ActivityNet test part. See Sec. 2.1 for details.

	model	pretr	LSMDC text → video				
			R@1↑	R@5↑	R@10↑	MnR↓	MdR↓
Ours	MDMMT(M_cALVYMTS) L9H8 irCSN152+audio	yes	$13.1_{\pm 0.5}$	$31.3_{\pm 0.3}$	$40.1_{\pm 0.0}$	$74.5_{\pm 0.7}$	$19.3_{\pm 0.5}$
Ours	MDMMT(M_cALVYMTS) L9H8 irCSN152+audio	no	$12.6_{\pm 0.7}$	$30.2_{\pm 1.5}$	$39.6_{\pm 0.9}$	$76.1_{\pm 0.8}$	$19.7_{\pm 1.3}$
Ours	MDMMT(M_cALVYMTS) L9H8 CLIP+audio	yes	$17.2_{\pm 0.6}$	$34.9_{\pm 0.4}$	$45.3_{\pm 1.0}$	$65.6_{\pm 0.8}$	$14.0_{\pm 0.8}$
Ours	MDMMT(M_cALVYMTS) L9H8 CLIP+audio	no	$16.2_{\pm 1.1}$	$35.4_{\pm 1.3}$	$45.1_{\pm 0.7}$	$64.9_{\pm 1.9}$	$14.7_{\pm 0.5}$

Table 7: Performance on LSMDC with and without pretrained model (HowTo100m).

	model	LSMDC text → video				
		R@1↑	R@5↑	R@10↑	MnR↓	MdR↓
	CT-SAN [19]	5.1	16.3	25.2	—	46
	JSFusion [18]	9.1	21.2	34.1	—	36
	MEE [10]	9.3	25.1	33.4	—	27
	MEE-COCO [10]	10.1	25.6	34.6	—	27
	CE [7]	$11.2_{\pm 0.4}$	$26.9_{\pm 1.1}$	$34.8_{\pm 2.0}$	$96.8_{\pm 5.0}$	$25.3_{\pm 3.1}$
	CLIP agg [14]	11.3	22.7	29.2	—	56.5
	CLIP [15]	12.4	23.7	31.0	142.5	45
	MMT (L) 7mod [4]	$12.9_{\pm 0.1}$	$29.9_{\pm 0.7}$	$40.1_{\pm 0.8}$	$75.0_{\pm 1.2}$	$19.3_{\pm 0.2}$
Ours	MDMMT(M_cALVYMTS) L9H8 irCSN152+audio	$13.1_{\pm 0.5}$	$31.3_{\pm 0.3}$	$40.1_{\pm 0.0}$	$74.5_{\pm 0.7}$	$19.3_{\pm 0.5}$
Ours	MDMMT(M_cALVYMTS) L9H8 CLIP+audio	$17.2_{\pm 0.6}$	$34.9_{\pm 0.4}$	$45.3_{\pm 1.0}$	$65.6_{\pm 0.8}$	$14.0_{\pm 0.8}$
Ours	MDMMT(M_cALVYMTS) L9H8 CLIP+irCSN152+audio	$\mathbf{18.8}_{\pm 0.7}$	$\mathbf{38.5}_{\pm 0.4}$	$\mathbf{47.9}_{\pm 0.7}$	$\mathbf{58.0}_{\pm 1.1}$	$\mathbf{12.3}_{\pm 0.5}$

Table 8: Test results on LSMDC public test (1k video)

	model	ActivityNet text $\rightarrow$ video				
		R@1↑	R@5↑	R@10↑	MnR↓	MdR↓
	MMT ($A_{p/r}$) motion+audio [4]	7.3	22.5	31	283.9	30
	CLIP [15]	9.4	22.8	31.3	302.3	35
Ours	MDMMT(M_cALVYMTS) L9H8 irCSN152+audio	$15.1_{\pm 0.1}$	$38.3_{\pm 0.1}$	$51.5_{\pm 0.3}$	$92.4_{\pm 2.3}$	$10.0_{\pm 0.0}$
Ours	MDMMT(M_cALVYMTS) L9H8 CLIP+audio	$17.7_{\pm 0.1}$	$41.6_{\pm 0.3}$	$54.3_{\pm 0.2}$	$76.0_{\pm 1.0}$	$8.3_{\pm 0.5}$
Ours	MDMMT(M_cALVYMTS) L9H8 CLIP+irCSN152+audio	$\mathbf{20.1}_{\pm 0.5}$	$\mathbf{45.1}_{\pm 0.5}$	$\mathbf{58.0}_{\pm 0.6}$	$\mathbf{70.8}_{\pm 0.1}$	$\mathbf{7.0}_{\pm 0.0}$

Table 9: Test results on our split (see Sec. 2.1) on ActivityNet.

	model	split	MSRVTT text $\rightarrow$ video				
			R@1↑	R@5↑	R@10↑	MnR↓	MdR↓
	Random baseline	full	0.0	0.2	0.3	1500	1500
	VSE [12]		5.0	16.4	24.6	—	47
	VSE++ [12]		5.7	17.1	24.8	—	65
	Multi Cues [12]		7.0	20.9	29.7	—	38
	W2VV [2]		6.1	18.7	27.5	—	45
	Dual Enc. [3]		7.7	22.0	31.8	—	32
	CE [7]		$10.0_{\pm0.1}$	$29.0_{\pm0.3}$	$41.2_{\pm0.2}$	$86.8_{\pm0.3}$	$16.0_{\pm0.0}$
	MMT (M) 7mod [4]		$10.7_{\pm0.2}$	$31.1_{\pm0.1}$	$43.4_{\pm0.2}$	$88.2_{\pm0.7}$	$15.0_{\pm0.0}$
	CLIP [15]		15.1	31.8	40.4	184.2	21
	CLIP agg [14]		21.5	41.1	50.4	—	**4**
Ours	MDMMT(MALVYMTS) L9H8 irCSN152+audio		$15.7_{\pm0.1}$	$38.8_{\pm0.1}$	$51.1_{\pm0.2}$	$76.0_{\pm0.7}$	$10.0_{\pm0.0}$
Ours	MDMMT(MALVYMTS) L9H8 CLIP+audio		$21.7_{\pm0.2}$	$47.6_{\pm0.3}$	$59.8_{\pm0.1}$	$55.9_{\pm0.2}$	$6.0_{\pm0.0}$
Ours	MDMMT(MALVYMTS) L9H8 CLIP+irCSN152+audio		**23.1**$_{\pm0.1}$	**49.8**$_{\pm0.1}$	**61.8**$_{\pm0.1}$	**52.8**$_{\pm0.2}$	$6.0_{\pm0.0}$
	MMT (M_c) 7mod [4]	full clean	$10.4_{\pm0.1}$	$30.2_{\pm0.4}$	$42.3_{\pm0.2}$	$89.4_{\pm0.6}$	$15.7_{\pm0.5}$
Ours	MDMMT(M_cALVYMTS) L9H8 irCSN152+audio		$15.8_{\pm0.1}$	$38.9_{\pm0.1}$	$51.0_{\pm0.1}$	$76.4_{\pm0.5}$	$10.0_{\pm0.0}$
Ours	MDMMT(M_cALVYMTS) L9H8 CLIP+audio		$21.5_{\pm0.1}$	$47.4_{\pm0.2}$	$59.6_{\pm0.1}$	$57.7_{\pm0.4}$	$6.0_{\pm0.0}$
Ours	MDMMT(M_cALVYMTS) L9H8 CLIP+irCSN152+audio		**22.8**$_{\pm0.2}$	**49.5**$_{\pm0.1}$	**61.5**$_{\pm0.1}$	**53.8**$_{\pm0.3}$	**6.0**$_{\pm0.0}$
	Random baseline	1k-A	0.1	0.5	1.0	500.0	500.0
	JSFusion [18]		10.2	31.2	43.2	—	13
	E2E [9]		9.9	24.0	32.4	—	29.5
	HT [11]		14.9	40.2	52.8	—	9
	CE [7]		$20.9_{\pm1.2}$	$48.8_{\pm0.6}$	$62.4_{\pm0.8}$	$28.2_{\pm0.8}$	$6.0_{\pm0.0}$
	CLIP [15]		22.5	44.3	53.7	61.7	8
	MMT ($M_{\text{1k-A}}$) 7mod [4]		$26.6_{\pm1.0}$	$57.1_{\pm1.0}$	$69.6_{\pm0.2}$	$24.0_{\pm0.8}$	$4.0_{\pm0.0}$
	AVLnet[16]		27.1	55.6	66.6	—	4
	SSB [13]		30.1	58.5	69.3	—	3.0
	CLIP agg [14]		31.2	53.7	64.2	—	4
Ours	MDMMT($M_{\text{1k-A}}$ALVYMTS) L9H8 irCSN152+audio		$31.3_{\pm0.1}$	$60.4_{\pm1.2}$	$71.8_{\pm1.0}$	$24.0_{\pm0.4}$	$3.0_{\pm0.0}$
Ours	MDMMT($M_{\text{1k-A}}$ALVYMTS) L9H8 CLIP+audio		$38.9_{\pm1.0}$	$68.3_{\pm0.7}$	$78.8_{\pm0.2}$	$17.3_{\pm0.5}$	**2.0**$_{\pm0.0}$
Ours	MDMMT($M_{\text{1k-A}}$ALVYMTS) L9H8 CLIP+irCSN152+audio		**38.9**$_{\pm0.6}$	**69.0**$_{\pm0.1}$	**79.7**$_{\pm0.6}$	**16.5**$_{\pm0.4}$	**2.0**$_{\pm0.0}$
	Random baseline	1k-B	0.1	0.5	1.0	500.0	500.0
	MEE [10]		13.6	37.9	51.0	—	10.0
	JPose [17]		14.3	38.1	53.0	—	9
	MEE-COCO [10]		14.2	39.2	53.8	—	9.0
	CE [7]		$18.2_{\pm0.7}$	$46.0_{\pm0.4}$	$60.7_{\pm0.2}$	$35.3_{\pm1.1}$	$7.0_{\pm0.0}$
	MMT ($M_{\text{1k-B}}$) 7mod [4]		$24.5_{\pm0.5}$	$54.4_{\pm0.8}$	$68.0_{\pm0.5}$	$26.6_{\pm0.2}$	$4.7_{\pm0.5}$
	CLIP [15]		24.5	46.2	56.8	60.9	7
Ours	MDMMT($M_{\text{1k-B}}$ALVYMTS) L9H8 irCSN152+audio		$28.8_{\pm0.9}$	$58.8_{\pm0.3}$	$71.2_{\pm0.3}$	$28.5_{\pm0.5}$	$3.7_{\pm0.5}$
Ours	MDMMT($M_{\text{1k-B}}$ALVYMTS) L9H8 CLIP+audio		$35.1_{\pm0.1}$	$66.5_{\pm0.9}$	$77.6_{\pm0.3}$	$21.5_{\pm0.4}$	$2.7_{\pm0.5}$
Ours	MDMMT($M_{\text{1k-B}}$ALVYMTS) L9H8 CLIP+irCSN152+audio		**37.4**$_{\pm1.5}$	**68.8**$_{\pm0.4}$	**79.4**$_{\pm0.4}$	**21.3**$_{\pm0.4}$	**2.0**$_{\pm0.0}$

Table 10: Results on MSRVTT dataset.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 4

[2] Jianfeng Dong, Xirong Li, and Cees G. M. Snoek. Predicting visual features from text for image and video caption retrieval. *IEEE Transactions on Multimedia*, 20(12):33773388, Dec 2018. 10

[3] Jianfeng Dong, Xirong Li, Chaoxi Xu, Shouling Ji, Yuan He, Gang Yang, and Xun Wang. Dual encoding for zero-example video retrieval, 2019. 10

[4] Valentin Gabeur, Chen Sun, Karteek Alahari, and Cordelia Schmid. Multi-modal transformer for video retrieval, 2020. 9, 10

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 4

[6] Giorgos Kordopatis-Zilos, Symeon Papadopoulos, Ioannis Patras, and Yiannis Kompatsiaris. Near-duplicate video retrieval with deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 347–356, 2017. 5

[7] Yang Liu, Samuel Albanie, Arsha Nagrani, and Andrew Zisserman. Use what you have: Video retrieval using representations from collaborative experts, 2020. 9, 10

[8] Dhruv Kumar Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018. 5

[9] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos, 2020. 1, 10

[10] Antoine Miech, Ivan Laptev, and Josef Sivic. Learning a text-video embedding from incomplete and heterogeneous data, 2020. 9, 10

[11] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In *ICCV*, 2019. 10

[12] Niluthpol Chowdhury Mithun, Juncheng Li, Florian Metze, and Amit K Roy-Chowdhury. Learning joint embedding with multimodal cues for cross-modal video-text retrieval. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 19–27, 2018. 10

[13] Mandela Patrick, Po-Yao Huang, Yuki Asano, Florian Metze, Alexander Hauptmann, Joo Henriques, and Andrea Vedaldi. Support-set bottlenecks for video-text representation learning, 2021. 10

[14] Jess Andrs Portillo-Quintero, Jos Carlos Ortiz-Bayliss, and Hugo Terashima-Marn. A straightforward framework for video retrieval using clip, 2021. 9, 10

[15] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *Image*, 2:T2. 1, 5, 9, 10

[16] Andrew Rouditchenko, Angie Boggust, David Harwath, Dhiraj Joshi, Samuel Thomas, Kartik Audhkhasi, Rogerio Feris, Brian Kingsbury, Michael Picheny, Antonio Torralba, and James Glass. Avlnet: Learning audio-visual language representations from instructional videos, 2020. 10

[17] Michael Wray, Diane Larlus, Gabriela Csurka, and Dima Damen. Fine-grained action retrieval through multiple parts-of-speech embeddings, 2019. 10

[18] Youngjae Yu, Jongseok Kim, and Gunhee Kim. A joint sequence fusion model for video question answering and retrieval, 2018. 9, 10

[19] Youngjae Yu, Hyungjin Ko, Jongwook Choi, and Gunhee Kim. End-to-end concept word detection for video captioning, retrieval, and question answering, 2017. 9

[20] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 5