IMFD_IMPRESEE at TRECVID 2019: Ad-Hoc Video Search and Video To Text

Rodrigo Hernandez[1], **Jesus Perez-Martin**[1,2], **Nicolas Bravo**[1,2], **Juan Manuel Barrios**[1,3,4], **Benjamin Bustos**[1,2]

[1]Department of Computer Science, University of Chile, Santiago, Chile
[2]Millennium Institute Foundational Research on Data (IMFD), Santiago, Chile
[3]ORAND S.A., Santiago, Chile
[4]Impresee Inc., CA, USA

rohernan@dcc.uchile.cl, jeperez@dcc.chile.cl, nbravo@dcc.uchile.cl, juan.barrios@impresee.com, bebustos@dcc.uchile.cl

Abstract

In this paper we present an overview of our participation in TRECVID 2019 [1]. We participated in the task Ad-hoc Video Search (AVS) and the subtasks Description Generation and Matching and Ranking of Video to Text (VTT) task. First, for the AVS Task, we develop a system architecture that we call "Word2AudioVisualVec++" (W2AVV++) based on Word2VisualVec++ (W2VV++) [11] that in addition to using deep visual features of videos, also uses deep audio features obtained from pre-trained networks. Second, for the VTT Matching and Ranking Task, we develop another deep learning model based on Word2VisualVec++, extracting temporal information of the video by using Dense Trajectories [16] and a clustering approach to encode them into a single vector representation. Third, for the VTT Description Generation Task, we develop an Encoder-Decoder model incorporating semantic states into the Encoder phase.

1 Ad-hoc Video Search

1.1 System Detail

For the ad-hoc video search evaluation, we propose a new model called "Word2AudioVisualVec++" (W2AVV++), which is an extension to the Word2VisualVec++ (W2VV++) [11] model. W2VV++ is a new version of W2VV [6], a deep neural network that projects a given sentence into a visual feature space. The W2VV++ model achieves this by vectorizing the given sentence using a multi-scale encoding strategy, and then, the encoding result is used by a multilayer perceptron (MLP) to produce a feature vector $r(s)$. During training, the network tries to minimize the loss between a given video v and the sentence s. The loss function used by this network is a marginal ranking loss [8] that works as follows: given a video-sentence pair (v, s) the loss is defined as:

$$l(v, s; \theta) = \max_{v}(0, \alpha + S_\theta(v^-, s) - S_\theta(v, s)) \qquad (1)$$

where v^- is a hardest negative video sample of the sentence s. This means that v^- is the most dissimilar sample of s in a mini-batch.

As W2VV++ only uses deep visual features of videos, we propose W2AVV++ which also uses deep audio features of videos. In order to represent a video, we create an audiovisual vector formed by the concatenation of one visual vector and one auditory vector, both obtained from the same video. The architecture of this model consists of two branches: one visual to work with the visual part of a video and an auditory one to work with the audio track of the video.

On the visual branch, we sample frames from a video in a uniformly manner, using an interval of 0.5 seconds. Using pre-trained CNN models ResNet-152 used in [4] and ResNeXt-101 used in [12], we extract deep visual features per frame. Each one of these features have 2,048 dimensions. Then, we obtain two video-level features of 2,048 dimensions by mean pooling over the frames of a video, each one corresponding to one of the two CNN models. We concatenate both feature vectors into a 4,096 dim vector and then use an extra fully connected layer for video feature re-learning [7]. This approach is based on the W2VV++ model.

On the auditory branch, we extract the audio track of each video. Then, we extract deep audio features of each audio track using pre-trained CNN models. In particular, we use SoundNet [2] and AENet [14]. Each of these models are used to extract a 1,024 dim feature vector per audio track and we later concatenate them to obtain one 2,048 dim feature vector.

After obtaining one deep visual feature vector and one deep audio feature vector per video (both of 2,048 dim), we concatenate them creating a 4,096 dim vector and use an extra fully connected layer to perform feature re-learning again. This layer generates one final 2,048 dim deep audiovisual feature vector.

Following the work done in W2VV++ for sentence representation, we use its sentence encoder. This encoder takes a given sentence and vectorizes it using three different vectorization methods: Bag-of-Words (BoW), word2vec and a Gated Recurrent Units (GRU) network. Each of this techniques produces an encoding that is later concatenated with each other and forwarded to a fully connected layer for common space learning.

The proposed W2AVV++ model is trained with the MSR-VTT [17] dataset. For testing, however, there was one particular problem which we had to overcome. The testing dataset for AVS in 2019 was V3C1 [3] (which is drawn from a larger V3C [13] video dataset), which is composed of 7,475 Vimeo shots. This dataset is segmented for the task into 1,082,657 short video segments. The problem is that only 586,730 of those shots have a valid audio track (roughly 54% of the dataset), while the other shots are only frames without audio tracks or the audio track is too short thus SoundNet or AENet cannot successfully extract a deep audio feature. In order to overcome this problem, we follow these steps:

1. We train two models: one W2VV++ visual-only model and one W2AVV++ audiovisual model, both trained with MSR-VTT.

2. If a V3C1 shot allows the extraction of a deep audio feature vector (because the shot has a valid audio track), then we calculate the audiovisual feature vector of the video and also the visual-only feature vector (the one we obtain from the visual branch described earlier).

3. If a V3C1 shot does not allow the extraction of a deep audio feature vector, then we obtain the visual-only feature vector of the video and we set its corresponding audiovisual feature vector as 0 (more precisely, vector of zeros). We save each visual feature in one matrix and each audiovisual feature in another one.

4. Then, for a given sentence, the W2VV++ model predicts a visual feature and the W2AVV++ model predicts an audiovisual feature. We implement the cross-modal similarity between a given sentence and any shot from the V3C1 dataset as the cosine similarity between the sentence feature vector and the calculated video feature vector. For every sentence, we compute the cosine similarity between the visual-only predicted feature and every visual-only feature vector corresponding to each shot, and then the cosine similarity between the audiovisual predicted feature and every audiovisual feature vector from each shot. This will generate two similarity matrices, one from visual-only features and another from audiovisual features.

5. Finally, we select the maximum between each entry in these matrices, and then we retrieve a list of 1,000 videos ranked by the cosine similarity in descending order.

RUN	Mean infAP
Run 1	0.041
Run 2	0.046
Run 3	0.040
Run 4	0.038

Table 1. Mean inferred Average Precision obtained by each run of our team.

1.2 Submissions

We submit four runs: Run 1 and Run 3 consist of the strategy described in this paper. Run 2 and Run 4 consist in systems based only in W2VV++ and are only used as baselines. The details of these runs are:

- Run 1: Uses both visual (ResNet-152 and ResNeXt-101) and audiovisual (visual + SoundNet + AENet) features predicted from each query, using W2VV++ and W2AVV++ models respectively as explained in this paper. For the marginal ranking loss uses an alpha margin of value $\alpha = 0.5$.

- Run 2: Uses only visual features predicted from each query, using W2VV++. This run works as a baseline for Run 1. For the marginal ranking loss uses an alpha margin of value $\alpha = 0.5$.

- Run 3: The same as Run 1, but for the marginal ranking loss uses an alpha margin of value $\alpha = 0.2$.

- Run 4: The same as Run 2, but for the marginal ranking loss uses an alpha margin of value $\alpha = 0.2$. This run works as a baseline for Run 3

The results from our 4 runs can be seen in Table 1. While Run 2 (visual-only) achieves better overall results than Run 1 (audiovisual), Run 3 (audiovisual) achieves better results than Run 4 (visual-only).

It is interesting to notice that for some queries, the proposed audiovisual model achieves better results than only using the visual features. For comparison, we report results from these queries in Table 2:

Query	Run 1: infAP	Run 2: infAP	Run 3: infAP	Run 4: infAP
612	**0.038**	0.030	**0.036**	0.029
618	**0.093**	0.071	**0.135**	0.114
620	**0.268**	0.254	**0.222**	0.208
621	0.031	**0.032**	**0.053**	0.045
624	**0.008**	0.000	0.005	0.005
631	0.109	0.109	**0.138**	0.026
633	0.105	0.105	**0.117**	0.110
636	**0.025**	0.013	**0.014**	0.009

Table 2. Results of every run for certain queries.

We can see that for the queries 612, 618, 620 and 636 the audiovisual models perform better than their visual-only counterparts. For query 621, while Run 2 gets slightly better results than Run 1, Run 3 performs better than Run 4. For query 624, Run 1 gets an infAP of 0.008 and Run 2 gets 0.000, meaning that the visual model did not retrieve any valid results. For queries 631 and 633, Run 3 gets noticeable better results than Run 4. These results suggest that our audiovisual proposal can perform better in certain kinds of scenarios.

2 Video to Text: Matching and Ranking

2.1 System Detail

Similar to our approach in Ad-Hoc video search, for Matching and Ranking subtask we use a deep learning model based on W2VV++ [11] to encode visual and textual embeddings into a common vector space. For this approach, rather than using audio as an extra embedding of the video, we extend W2VV++ by using Dense Trajectories [16] as a visual embedding to encode temporal information of the video. We think that combining the pretrained Resnet-152 [4] and Resnext-101 [12] descriptors of W2VV++, with Dense Trajectories descriptors of the video, we can extract spatial and temporal information from it, respectively. We also transform both video and text into a common space, rather than only from video to text.

Dense Trajectories [16] describes groups of pixels paths across a fixed number of frames of the video, by using Histogram of Oriented Gradients (HOG) and Histogram of Optical Flow (HOF) for pixel tracking. These trajectories are represented by a set of coordinates (x, y) where $x, y \in [-1, 1]$. We fix the trajectory length to 15 frames and the dense pixel size to a 5x5 square of pixels.

Since a single video can generate thousands of dense trajectories, we use K-Means clustering to find a fixed number of codewords across the training videos and represent all the dense trajectories of a video into a single histogram or Bag of Words vector. Using the elbow method, we found that 800 codewords was more than enough to encode all Dense Trajectories of the training dataset.

For the sentence representation, we use the same encoding in Ad-hoc video search: a concatenation of Bag of Words, Word2Vec and a trainable Gated Recurrent Unit layer.

$$fusion_s = BoW(s) \,||\, w2v(s) \,||gru(s) \tag{2}$$

For the video representation, define $\max_{rs152}(v)$ as the max pooling of the set of pretrained ResNet-152 [4] vectors: and $\max_{rs101}(v)$ the max pooling of the pretrained ResNext-101 [12] vectors, we get a single vector representation of the video by concatenating them with $BoW_{dt}(v)$: the Bag of Words vector resulting from the clustering of the Dense Trajectories of the video.

$$fusion_v = BoW_{dt}(v) \,||\, \max_{rs152}(v) \,||\, \max_{rs101}(v) \tag{3}$$

Following the Matching and Ranking model of Dong *et al.* for TRECVID 2017 [5], to transform the sentence and video embedding into a common vector space, we use two fully-connected layers with and ReLU activation function. We also use batch normalization [10] in one of the submitted runs, e.g. for the video branch of the model:

$$\begin{aligned} v' &= \sigma(BN(W_1 fusion_v + b_1)) \\ \vec{\boldsymbol{v}} &= \sigma(BN(W_2 v' + b_2)) \end{aligned} \tag{4}$$

where W_1 are W_2 trainable weight matrices for each fully-connected layer with b_1 and b_2 their corresponding bias. BN represents a batch normalization layer and σ the ReLU function.

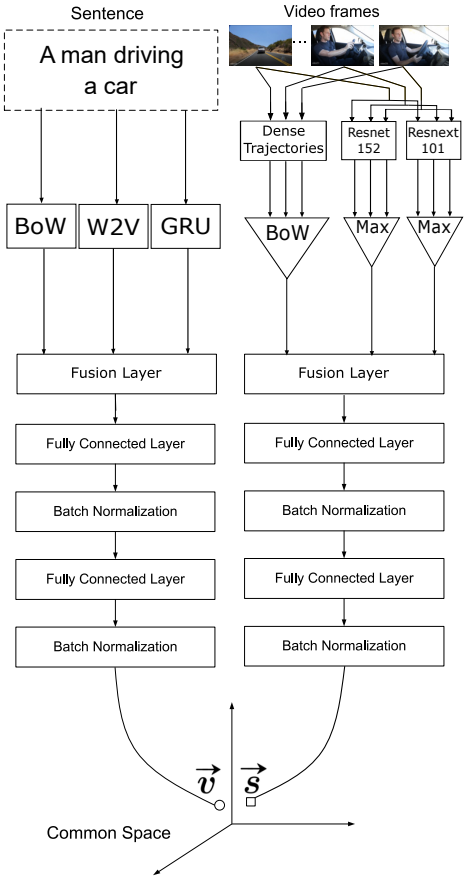

Figure 1. Model used for Matching and Ranking subtask. This configuration with batch normalization is used for the run 1, but for the second run this feature is missing.

The loss function we use is a Triplet Ranking Loss function defined as:

$$l(\vec{\boldsymbol{v}}, \vec{\boldsymbol{s}}; \theta) = \max_{\vec{\boldsymbol{s}}\_}(0, \alpha + S(\vec{\boldsymbol{v}}, \vec{\boldsymbol{s}}\_) - S(\vec{\boldsymbol{v}}, \vec{\boldsymbol{s}}))) \tag{5}$$

where $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{s}})$ is a corresponding video-sentence pair, θ is the set of parameters of the model, $\vec{\boldsymbol{s}}\_$ is the hardest negative sentence in the batch e.i., the sentence in the batch different to s

that is closest to v; α is a constant that sets the minimum margin desired between $\vec{s}$ and $\vec{s}\_$ and S is the cosine similarity function. With this, if D is the training dataset, the optimization process aims to minimize the sum:

$$\min_{\theta} \sum_{(v,s)\in D} l(v, s; \theta) \tag{6}$$

We also follow Dong *et al.* [5] approach for the optimization of equation 6. Using RMSProp [15] with initial learning rate $\eta = 0.0003$, weight decay $\gamma = 0.9$ and $\epsilon = 10^{-9}$. We use dropout on all fully-connected layers with $p = 0.2$. The training process splits in half the learning rate if the validation loss has not improved in 2 iterations, if it does not improve in 5 iterations the training process stops.

2.2 Datasets

We trained the model using MSR-VTT [17] which consists of more than 150k of video-sentence pairs. Videos are extracted from YouTube™ platform and annotations are made by Amazon Mechanical Turk™ with a rigorous quality review from the authors of the dataset. We use the same dataset for cross-validation, randomly choosing 10% of the dataset as validation for every epoch.

2.3 Submissions

For this task, we submitted 2 runs with different configurations of the model. The first one uses batch normalization on the two fully-connected layers of each branch, while in the second one this feature is removed.

For a submission subset, to rank a video v with a set of sentences $\{s\}_{1..n}$ we use the cosine similarity function S employed in the loss function (5), calculating the distance between the vector representation of video in the common space $\vec{v}$ and the vector representations of the set of sentences $\{\vec{s}\}_{1..n}$, and sorting them in decreasing order to get the ranking list for v.

RUN	A	B	C	D	E
Run 1	0.004	0.005	0.004	0.005	0.005
Run 2	0.015	0.017	0.015	0.016	0.018

Table 3. Mean Inverted Rank obtained by run and test subset of our team in Matching and Ranking.

Table 3 shows the second run where there was not batch normalization obtained better results in the subtask, with consistent better score than the batch normalization approach. This may be explained by unnecessary parameters the batch normalization layer adds to the complexity of the model, overfitting it on the training dataset.

3 Video to Text: Description Generation

3.1 System Detail

We develop a deep learning model based on an Encoder-Decoder system. This model combines CNN and RNN architectures in a framework that firstly uses a neural network for visual recognition (the *encoder*), and secondly incorporates a neural network for sequence generation (the *decoder*). The model is an end-to-end trainable deep network model where the two stages are learned simultaneously.

We mainly focused on saturating the semantic information of the most important concepts. We used a Semantic Compositional Network (SCN) [9] to understand effectively individual semantic concepts for videos using 2D CNN and 3D CNN to represent effectively the spatio-temporal visual content of the video. For that, we select K important concepts from the dataset and train a recurrent tag-model for each one. Then we compose the recurrent states of all tag-models as input of our recurrent encoder based into a bidirectional LSTM.

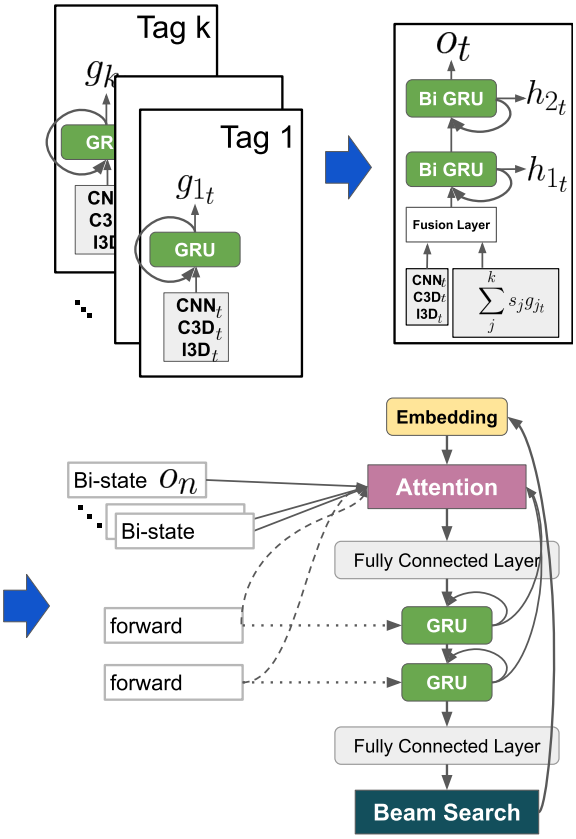

Figure 2. Encoder-Decoder model used for Description Generation subtask.

Acknowledgements

The authors of this paper would like to thank NIST and all the coordinators of TRECVID for organizing this event. Special thanks to Cees Snoek, Pascal Mettes and the University of Amsterdam MediaMill team for sharing their pre-trained ResNeXt-101 model with us.

References

[1] George Awad, Asad Butt, Keith Curtis, Yooyoung Lee, Jonathan Fiscus, Afzal Godil, Andrew Delgado, Alan F. Smeaton, Yvette Graham, Wessel Kraaij, and Georges Qunot. Trecvid 2019: An evaluation campaign to benchmark video activity detection, video captioning and matching, and video search & retrieval. In *Proceedings of TRECVID 2019*. NIST, USA, 2019.

[2] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. Soundnet: Learning sound representations from unlabeled video. In *Advances in Neural Information Processing Systems*, 2016.

[3] Fabian Berns, Luca Rossetto, Klaus Schoeffmann, Christian Beecks, and George Awad. V3C1 dataset: An evaluation of content characteristics. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval, ICMR 2019, Ottawa, ON, Canada, June 10-13, 2019.*, pages 334–338, 2019.

[4] Jianfeng Dong, Shaoli Huang, Duanqing Xu, and Dacheng Tao. Dl-61-86 at trecvid 2017: Video-to-text description. *TRECVID Workshop*, 2017.

[5] Jianfeng Dong, Shaoli Huang, Duanqing Xu, and Dacheng Tao. Dl-61-86 at trecvid 2017: Video-to-text description. In *TRECVID Workshop*, 2017.

[6] Jianfeng Dong, Xirong Li, and Cees G. M. Snoek. Predicting visual features from text for image and video caption retrieval. *IEEE Trans. Multimedia*, 20(12):3377–3388, 2018.

[7] Jianfeng Dong, Xirong Li, Chaoxi Xu, Gang Yang, and Xun Wang. Feature re-learning with data augmentation for content-based video recommendation. In *2018 ACM Multimedia Conference on Multimedia Conference, MM 2018, Seoul, Republic of Korea, October 22-26, 2018*, pages 2058–2062, 2018.

[8] Fartash Faghri, David J. Fleet, Jamie Kiros, and Sanja Fidler. VSE++: improving visual-semantic embeddings with hard negatives. In *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*, page 12, 2018.

[9] Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. Semantic Compositional Networks for Visual Captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1141–1150. IEEE, 7 2017.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[11] Xirong Li, Jianfeng Dong, Chaoxi Xu, Jing Cao, Xun Wang, and Gang Yang. Renmin university of china and zhejiang gongshang university at trecvid 2018: Deep cross-modal embeddings for video-text retrieval. In *TRECVID Workshop. https://github.com/li-xirong/avs*, 2018.

[12] Pascal Mettes, Dennis C. Koelma, and Cees G. M. Snoek. The imagenet shuffle: Reorganized pre-training for video event detection. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval, ICMR 2016, New York, New York, USA, June 6-9, 2016*, pages 175–182, 2016.

[13] Luca Rossetto, Heiko Schuldt, George Awad, and Asad A Butt. V3c–a research video collection. In *International Conference on Multimedia Modeling*, pages 349–360. Springer, 2019.

[14] Naoya Takahashi, Michael Gygli, and Luc Van Gool. Aenet: Learning deep audio features for video analysis. *IEEE Trans. Multimedia*, 20(3):513–524, 2018.

[15] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[16] Heng Wang, Alexander Kläser, Cordelia Schmid, and Liu Cheng-Lin. Action recognition by dense trajectories. 2011.

[17] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. MSR-VTT: A large video description dataset for bridging video and language. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 5288–5296, 2016.