UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

VIDEO SENTENCE MATCHING USING DENSE TRAJECTORIES AND INFERSENT

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

NICOLÁS ANTONIO BRAVO RAMÍREZ

PROFESOR GUÍA:
BENJAMÍN BUSTOS CÁRDENAS

MIEMBROS DE LA COMISIÓN:
NELSON BALOIAN TATARYAN
JUAN MANUEL BARRIOS NÚÑEZ
PABLO HUIJSE HEISE

Este trabajo ha sido financiado por ANID
Programa Iniciativa Científica Milenio - Código ICN17_002

SANTIAGO DE CHILE
2021

Resumen

Debido a la gran cantidad de videos generados en Internet, surge la necesidad de codificar automáticamente su contenido a texto para facilitar, por ejemplo, su búsqueda en indexadores Web como Google o Bing. Debido a esta necesidad esta investigación propone mejorar un modelo de aprendizaje automático especializado en una tarea que relaciona el contenido de un video a una oración textual que lo describa. En este medimos la efectividad de un descriptor de Trayectorias Densas y el modelo de InferSent en la tarea de TRECVID de relacionar video a texto, seleccionando un modelo del estado del arte como base y experimentando múltiples variaciones del mismo utilizando los descriptores mencionados.

La tarea de relacionar video a texto trata en encontrar las frases pre-generadas más adecuadas que describan el contenido de un vídeo y ordenarlas según relevancia utilizando una función de distancia. Los mejores modelos que resuelven esta tarea se basan en una combinación de descriptores espaciales y temporales para vídeo y texto, para luego transformarlos a un espacio vectorial en común, donde una función de distancia ordena los vectores de las oraciones con respecto al vector del video objetivo.

Nuestra hipótesis plantea que los descriptores de Trayectorias Densas pueden codificar la información temporal de un vídeo, obteniendo un mejor rendimiento que un modelo que utiliza descriptores de información estática y redes recurrentes para extraer esta información temporal. Para el modelo de InferSent, experimentamos si el uso de un codificador de oraciones supera una combinación de codificadores de palabras que utiliza el modelo base. También medimos el rendimiento en esta tarea de una arquitectura ResNeXt-101 pre-entrenada en un conjunto de datos de clasificación de objetos diferente al del modelo base, con el objetivo de determinar la importancia de la versión entrenada en Shufflenet que el modelo base utiliza para lograr sus resultados del estado del arte.

Al comparar los resultados del modelo base con los nuestros, se concluye que que las Trayectorias Densas tienen el potencial de describir la información temporal del video, aunque las manera en como este trabajo las integra al modelo base es mejorable. Además se concluye que el uso de InferSent en el modelo base seleccionado es innecesario, ya que este último obtiene los mismos resultados y se comporta de la misma manera. Los resultados también demuestran que añadir los descriptores de estudio como codificaciones adicionales al del modelo base obtiene mejores resultados que los escenarios donde estos reemplazan los codificadores originales. También concluimos que el uso de una ResNeXt-101 pre-entrenada en Shufflenet es un aspecto clave para lograr resultados del estado del arte, aunque es necesaria más investigación para entender este comportamiento.

Abstract

Due to the large amount of videos generated on the Internet, there is a need to automatically encode their content to text to facilitate, for example, their search in Web indexers such as Google or Bing. Due to this need, this research proposes to improve a machine learning model specialized in a task that relates the content of a video to a textual sentence that describes it. In this work we test the effectiveness of Dense Trajectories and InferSent descriptors in the Matching & Ranking task of the TRECVID challenge, by selecting a state-of-the-art model as baseline and testing multiple variations of it where the descriptors under study encode the video and sentence information, respectively.

The Matching & Ranking task comprises on finding the suitable pre-generated sentences that describe the content of a video and ranking them using a score system. The best models that solve this task rely on the combination of spatial and temporal descriptors for video and words that then are mapped into a common vector space, where a distance function ranks the sentence vectors to the target video vector. For video they extract spatial information with the aid of pre-trained convolutional networks, and extract temporal information by using a sequence encoding model over the set of visual vectors features. For the text they combine different word embeddings like Bag of Words, word2vec or recurrent networks.

We hypothesize that Dense Trajectories descriptors can encode the temporal information of a video, leading to better performance of a model that uses static information descriptors and recurrent networks to extract the temporal information from them. For the InferSent model, we test if the use of sentence embeddings with high transfer learning capabilities outperforms a combination of word embeddings of the baseline model. We also test the use of a ResNeXt-101 architecture pre-trained on an object classification dataset different from the one used by the base model, to test the importance of the Shufflenet version this model uses to achieve its state-of-the-art results in the task.

When comparing the results of the base model with our models, we conclude the Dense Trajectory descriptors have the potential to describe the temporal information of a video in the target task, although the approach studied in this work can be improved. Also, we conclude the use of InferSent in the base model is unnecessary since this model behaves the same with this sentence descriptor. The results also show that adding the descriptors of study as an extra video or sentence encoding gets better results than using them as the only embeddings for video and sentence descriptors. We also conclude the use of a ResNeXt-101 pre-trained on Shufflenet is a key approach to achieve state-of-the-art results, although further investigation must be done to find the reason of this behavior.

A mi amada familia

Agradecimientos

Gracias a mi amada familia, en especial a mi padre, a mi madre y a mi querida Kika, quienes me han criado y formado en la persona que soy ahora, por estar siempre a mi lado y por su máximo esfuerzo para tener la oportunidad de elegir mi futuro. Siempre me faltarán palabras para agradecerles por su amor y preocupación.

Gracias a la señorita Yanara que ha estado a mi lado en esta importante etapa de mi vida, la persona con quien he compartido mis mejores y peores momentos, y con quien he vivido miles de experiencias a lo largo de estos últimos años. Agradezco a Dios por que aparecieras en mi camino y tener la oportunidad de compartirlo juntos, gracias por todo tu amor señorita, eres mi tesorito especial. Agradezco también a toda su familia por haberme recibido con los brazos abiertos y hacerme sentir como en casa cada vez que compartimos.

Un gran agradecimiento a Los Cracks: Américo, Belisario, Gabriel, Javier, Joaquín, Juan, Rodrigo y Sergio, con quienes he vivido la experiencia universitaria. Gracias por las incontables juntas donde hemos compartido como si nos conocieramos hace años, por las infinitas conversaciones, por el apoyo y ayuda mutua durante toda la carrera. Un abrazo enorme a todos, deseando que esta amistad perdure más allá de la universidad ya que son unas personas muy valiosa para mí.

Gracias a mis amigos de plan común por el apoyo mútuo que vivimos durante esa etapa tan temida, donde salí de la burbuja de mi colegio y experimenté distintas realidades de todas partes de Chile. Agradezco también la amistad incondicional de Benjamín, Camilo, Adolfo y Martín, la cual ha perdurado todos estos años después del Liceo, a pesar de la distancia entre los distintos caminos que hemos tomado en nuestras vidas.

Agradezco a todos los profesores que han dado de su esfuerzo para mi formación educacional, en especial al Dr. Benjamín Bustos quien me ha guiado en esta importante etapa, siempre ayudandome con mis miles de dudas y preocupándose de no desviarme del objetivo de mi tesis y de que mi progreso no se viera afectado. Agradezco a mi comisión de tesis, Dr. Nelson Baloian, Dr. Juan Manuel Barrios y Dr. Pablo Huijse, por orientarme en mi trabajo para culminarlo de la mejor manera. Quiero también mostrar mi gratitud a mis profesores de enseñanza media y básica, a la Prof. Gabriela Lemus, la Prof. Gabriela Mesa y la Prof. Isabel Gamboa por todo su esfuerzo y preocupación en mis últimos años en el Liceo.

Y muchas gracias al señor por darme la energía y confianza necesarias para cumplir con mis objetivos y otorgarme la oportunidad de ser un profesional.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Over the last years, the need to automatically handle the immeasurable quantity of images, videos and sounds in the age of Big Data has made novel advances in their storage, extraction and processing, with machine learning taking a fundamental role in this last aspect [5]. In many academic areas, like computer vision, many advances are the product of competitions in several international challenges and tasks, where participants try to get the best solution for important technical problems, making life easier and more comfortable for thousands or millions of people. This work aims to give a solution for one of these important tasks: the video-to-text processing task of finding the best suitable pre-defined sentence for a video content, which can help to know the content of a video without watching or hearing it at all.

The Video to Text (VTT) matching & ranking task is a digital video retrieval challenge of the TRECVID (TREC Video Retrieval Evaluation) conference, an international yearly event sponsored by the National Institute of Standards and Technology (NIST). This task aims to find the best pre-defined sentences that describe the content of a single video. The solution chooses sentences from a pre-defined set, concentrating the work only on the matching task and not on generating the sentences. A well-performed solution for this task can help the Information Retrieval field, e.g. giving to web search engines a way to represent videos with text, making easier to index their content; or giving blind users access to videos in social media, using a pre-defined set of generic sentences that could describe its content.

To solve the VTT matching & ranking task, recent work as Li *et al.* [1] or Marsden *et al.* [6] transform video and sentence into a common vector space to do the matching task by similarity or document ranking, respectively. Their solutions extract static information of objects, places or actions from each video using convolutional neural networks [7, 8, 9, 10] over its frames. However, these techniques try to extract the temporal information using sequence encodings over the set of spatial information vectors, losing part of the rich temporal information the raw video frames have.

In this work we focus on testing the effectiveness of Dense Trajectories [4] and InferSent [3] as feature extractors for the Video-to-Text matching & ranking task, where we hypothesize that "the use of Dense Trajectories or InferSent embeddings can improve the performance of the model of Li *et al.* [1] for the Video-to-Text Matching & Ranking task in TRECVID 2018". Our approach is to apply different variations to a state-of-the-art architecture and compare their performance in TRECVID 2018 with the base model results. The base model we select is the triple level encoder by Dong *et al.* [11], where we replace existing encodings or add the feature descriptors of study as additional ones.

1.2 Objectives

1.2.1 Main objective

Compare against the *Dual Encoding for Zero-Example Video Retrieval* model by Dong *et al.* [11] the performance of an architecture based on it that uses Dense Trajectories as temporal descriptors of a video and InferSent for sentence vectorization, on the Video-To-Text Matching & Ranking task of TRECVID 2018.

1.2.2 Specific objectives

In this work, the main objective requires the following:

1. Own implementation of the base model *Dual Encoding for Zero-Example Video Retrieval* and our proposed variations.
2. Replication of the reported performance of the base model on the matching & ranking task of TRECVID 2018 [1] by selecting the exact hyper-parameters used by its authors.
3. Implementations of the base model and its variations trained on the same dataset selected by Dong *et al.* , defined as a combination of the video captioning datasets TGIF [12], MSR-VTT [13] and MSVD [14].
4. Test results using the Mean Inverted Rank metric of the models under study on the VTT matching & ranking official evaluation dataset of TRECVID 2018, and their comparison.
5. Participation on the Video-To-Text Matching & Ranking task of TRECVID 2019 using a model based on the work of Dong *et al.* for TRECVID 2017 [2].
6. Public source code and instructions of our models to solve Video to Text task.

1.3 Methodology

For the creation and evaluation of our model we perform the following steps:

1. Download and pre-process the MSR-VTT, TGIF and MSVD datasets using the same methodology of the baseline model. This discards a possible difference in the input data of our implementation of the baseline model, concentrating the replication efforts in finding the hyper parameters that give the same reported results of Li *et al.* in TRECVID 2018 [1].
2. Implement the video vectorization branch of the base model and its ResNeXt-101 and Dense Trajectories variations.

3. Implement the sentence vectorization branch of the base model and its InferSent variations.
4. Implement the common vector space mapping to join both sides of the network.
5. Validate our base model implementation by finding the correct hyper parameters configuration that replicates the reported results of the base model in the VTT Matching & Ranking challenge of TRECVID 2018.
6. Train our model variations using the MSR-VTT, TGIF and MSVD datasets.
7. Test the models by using the TRECVID 2018 Matching and Ranking dataset and compare their results with the baseline.

1.4 Structure of this Research

This document is organized as follows:

1. Chapter 2 describes the theoretical framework of this work, including different models for object detection and action recognition; we describe Dense Trajectories descriptors and the InferSent model.
2. Chapter 3 describes our baseline model and why we choose it.
3. Chapter 4 introduces our proposed model variations with our Dense Trajectories and InferSent descriptors to test if it can improve the baseline performance.
4. Chapter 5 presents the training, validation and testing dataset, along with the preprocessing methodologies.
5. Chapter 6 shows the validation of our base model implementation, along with the training, validation and testing methodology and results for the models under study.
6. Chapter 7 presents our participation in the TRECVID 2019 Matching & Rankign challenge.
7. Chapter 8 concludes our work and elaborates on future plans.

Chapter 2

Theoretical background

The Matching and Ranking task our model sets out to solve is a multimodal task that relates video with sentences, two types of data from different sources and embeddings. In this chapter we present different representations for these types of data, along with some notorious tasks in the fields of text, image and video processing. We do not use all these embeddings in our research, it is important to show the advantages and disadvantages of each one. To introduce the Matching & Ranking task, we present different approaches for this challenge over the last years.

2.1 Sentence Vectorization

2.1.1 Bag of Words

A classical sentence vectorization method aim to create a fixed sized representation, no matter the quantity of words in the phrase. This technique is called Bag of Words (BoW), which corresponds to a word frequency vector of the sentence with respect to a pre-selected vocabulary. Formally, for a sentence s and a vocabulary V, the definition of the value of each position i in the vector $BoW(s)$ is:

$$BoW(s)_{\mathrm{i}} = \begin{cases} 1 & \text{if word } V_{\mathrm{i}} \text{ in } s \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

The advantage of this method it is easy to define and debug, but this representation depends on a correct selection of the vocabulary, because a too large vocabulary can lead to a sparse representation (too many zero values), possibly loosing fine semantic correlations between words and hindering the pattern learning of a model.

Huang *et al.* [15] cope with the sparse problem by applying a hash algorithm that decomposes each word into tri-words (words of three letters). For example the word 'home' turns into the three-words ['\$ho', ' hom', 'ome', 'me\$']; therefore, the fixed words used in the BoW method are tri-words and not complete ones. This reduces the size of the frequency vector in

BoW because common tri-words shared between different words reduces the number of zeros in the vector.

2.1.2 Word2Vec

Mikolov *et al.* [16] published the Word2Vec technique to create a vector embedding for a word. It considers the *context* of the target word, i.e. words that commonly appear around the target one. The objective of this technique is to generate similar vectors for words with similar contexts. They achieve this by training a recurrent neural network over a big corpus, feeding it with skip-grams or pairs (w, c) of target words w and context word c. The network encodes each word into a vector space, and learns to bring closer words w with the same context word c.

This word embedding can be extended to represent sentences by obtaining the average Word2Vec vector:

$$s_{word2vec}(s) = \frac{1}{|s|} \sum_{w \in s} word2vec(w) \tag{2.2}$$

2.1.3 Recurrent Neural Networks

A valid interpretation of a sentence is seeing it as a sequence of words and, as a sequence, recurrent neural networks (RNN) can encode its semantic meaning. These architectures are of a special neural network that connects several nodes with a temporal sequence. There are two types of recurrent neural networks: finite pulse or infinite pulse. A finite pulse RNN can be unrolled and replaced with a feed-forward neural network, while an infinite pulse RNN cannot be unrolled. A RNN has additional internal states that serve as storage or memory of what the network sees, and the RNN can or cannot have direct control on it to update its information. Among different RNN, we review the Long-Short Term Memory (LSTM) and Gated Recurrent Units(GRU), two popular architectures that have shown excellent results in natural language processing (NLP) tasks like statistical machine translation (SMT) [17, 18], speech recognition [19, 20] and handwriting recognition [21].

Hochreiter and Schmidhuber [22] created Long-Short Term Memory recurrent networks along with a back-propagation algorithm with constant gradient, a mechanism that solved the vanishing or exploding gradient problem these neural networks had when dealing with long-term sequences. [23, 24]. This new recurrent network uses gates to learn which information to keep from the input (input gate), which information to forget from the previous state (forget gate), and which information to pass to the next step (output gate). This sets of gates gives the LSTM more control on the information flow and allows it to learn long-term dependencies more easily than simple recurrent networks. The author's strategy was to combine it with a constant gradient, using the parametric gates to control what data to keep and what to forget, but the approach of making the gradient between the previous state to the current state conditioned on the context proved to be a much better approach [25].

Cho *et al.* [17] creates an encode-decoder for statistical machine translation from English sentences to French. The encoder comprises a RNN that learns to transform an English sentence into an intermediate vector representation. The decoder uses another RNN that reads the intermediate vector and generates the most likely French translation into the probability distribution of the training dataset. With this architecture, the encoder and decoder training aims to maximize the conditional log-likelihood [26] between the related English-French sentence of the training dataset. Along with this approach, the authors create a new recurrent neural unit which they call Gated Recurrent Unit or GRU (see Figure 2.1). This new unit differs from a LSTM unit in that only a single gate controls the forgetting factor of the LSTM forget gate and the update factor of the LSTM input gate. This results in a model with fewer parameters that is easier to understand, simpler to implement and faster to train. With this design, a GRU only depends on 2 parametric gates: update and reset; instead of 3 gates like a LSTM: input, output and forget. In theory, using GRU instead of LSTM has the disadvantage of losing accuracy in datasets with long sequences because a GRU has less internal memory states than a LSTM, however in practice this is not the case [27].

The two gates that define a Gated Recurrent Unit are the reset gate r_t and update gate u_t, where t is a specific step in the sequence. The internal state h_t corresponds to the output of the unit and the input of the next unit in the sequence. The reset gate r_t learns what to forget from the previous state h and creates a new internal state $\tilde{h}_t$ in combination with the input data x_t. The update gate u_t decides if the previous state h_{t-1} needs to be updated with the new internal state $\tilde{h}_t$.

$$
\begin{aligned}
r_t &= \sigma(b_r + W_r x_t + U_r h_{t_1}) \\
u_t &= \sigma(b_u + W_u x_t + U_u h_{t-1}) \\
\tilde{h}_t &= tanh(b_h + W_h x_t + U_h(r_t \odot h_{t-1})) \\
h_t &= (1 - u_t) \odot h_{t-1} + u_t \odot \tilde{h}_t
\end{aligned}
$$

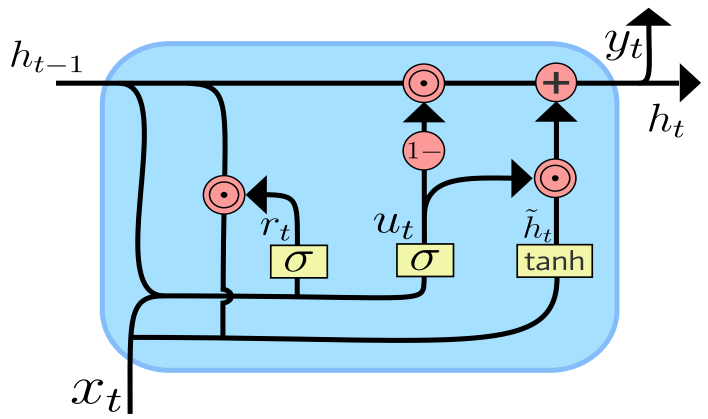
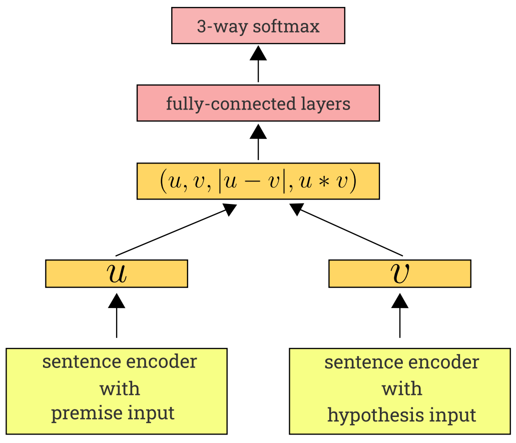


Figure 2.1: Fully Gated Recurrent Unit design. The sigmoid squares represent the 2 gates of the unit. The shortcut from the update gate to the internal state $\tilde{h}_t$ shows how a GRU uses a single gate to control the update and forget behavior of a LSTM.

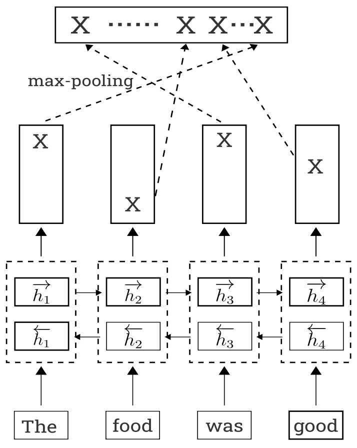
2.1.4 InferSent

Conneaau *et al.* [3] propose *InferSent*, an architecture that aims to create sentence embeddings that can be highly transferable to other NLP tasks. The study of *InferSent* showed that using this embedding for transfer learning, i.e., use the pre-trained InferSent sentence embeddings as input to train other models, in 12 different NLP tasks, getting better results than most of the state-of-the-art approaches. Among these tasks are supervised ones like sentiment analysis and caption-image retrieval, as well as others, like semantic entailment and relatedness using the SICK dataset [28]. The research of *InferSent* focuses on the architecture to encode a sentence and the task to train it.

The Standford Natural Language Inference (SNLI) dataset [29] provides 570k human-generated English sentence pairs, labeled with one of three categories: entailment, contradiction, and neutral. Conneau *et al.* hypothesize the semantic nature of the Natural Language Inference (NLI) task is a good supervised option for learning universal sentence embeddings. They use a training scheme that encodes each sentence of the labeled pair with a shared sentence encoder, generating the vector u for the first sentence and the vector v for the second sentence of the pair (see Figure 2.2a). They create joint representation by using three matching methods: concatenation (u, v), elements-wise product $u * v$ and absolute element-wise difference $|u - v|$. The joint representation is the concatenation of these methods $[(u, v), u * v, |u - v|]$. They then feed this vector into a neural network classifier with fully-connected layers, assigning one of the three labels (entailment, contradiction, or neutral) to the sentence pairs. To choose the sentence encoder, Conneau *et al.* test the transfer learning performance of 7 different architectures on the 12 mentioned evaluation tasks, selecting a BiLSTM [30] with max pooling as the best performed architecture among the 7 in study (see Figure 2.2b). For a sentence of T words $w_t t = 1...T$, the BiLSTM calculates a forward hidden vector $\overrightarrow{h_t} = \overrightarrow{LSTM}_t(w_1, ..., w_T)$ and a backward hidden vector $\overleftarrow{h_t} = \overleftarrow{LSTM}_t(w_1, ..., w_T)$ with $t = 1...T$. Then, they concatenate each hidden state into $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$.Finally, the model applies a max pooling [31] over the set of h_t to get a single encoding of the sentence.


3-way softmax
fully-connected layers
$(u, v, |u - v|, u * v)$
u
v
sentence encoder with premise input
sentence encoder with hypothesis input

(a)



(b)

Figure 2.2: (a) Training scheme used in Conneau *et al.* [3]. (b) InferSent encoding model of Conneau *et al.* [3]. The training of this architecture uses the Standford Natural Language Inference dataset, where it has to detect if two sentence topics are related, neutral, or contradictory.

2.2 Video vectorization

A video, because it is a sequence of images and sound across time, is a complex multimedia data to process. The temporal variable creates relations between continuous frames and sounds, making it computationally expensive to process. Besides this problematic, many approaches to solve specific tasks have appeared over recent years, inheriting methods from image processing and combining them with aggregated techniques to cope with the information complexity of the time dimension. With a vast range of related research, we highlight a few instances that have a connection with this work.

2.2.1 Optical embeddings

To understand the definition of Dense Trajectories, we first present 3 common optical embeddings: Histogram of oriented gradients (HOG), Histogram of Optical Flow Orientation (HOF) and Motion Boundary Histogram (MBH). HOG [32] is a feature descriptor used mainly for object detection in images. It uses the difference of color or brightness of the image using a rectangular or circular mask that weighs the pixel values around a center point, estimating the border or movement direction of the pixel zone. This value is added to its corresponding histogram channel, repeating the same process to all pixel zones of the image to complete the histogram. HOF and MBH are feature descriptors for a sequence of images instead of a single image. HOF [33] follows the flow of pixels across a fixed number of images, using a technique similar to HOG to obtain the orientation of the flow: First it adds the weighted orientation to its corresponding histogram bin, to then calculate the weighted orientation by getting a histogram over each grid cell that splits the image area. MHB [34] is a family of motion boundary descriptors, i.e., techniques that describe the perimeter of moving objects between two frames, that are defined as the derivative of HOF across the x axis and y axis, separately. The primary advantage of these descriptors is the suppression of the noise the camera movements add as a shared constant to all pixel flows of a sequence.

Laptev *et al.* [33] use HOG and HOF to classify human actions from movies. They create spatio-temporal descriptors of the video, pooling them into a fixed size mid-level representation using a standard Bag of Words technique. A non-linear Support Vector Machine makes the classification. This solution was a step forward in human action classification and shows the general pipeline for video processing of this research.

2.2.2 Dense Trajectory

Dense Trajectories by Wang *et al.* [4], correspond to the path a group of pixels take across a set of frames. Figure 2.3 schematize how a Dense trayectory according to Wang *et al.* is obtained: it results from tracking a group of $W \times W$ pixels (dense pixel) across a maximum of $L = 15$ frames. This method uses the Färneback optical flow algorithm [35] $\omega = (u_t, v_t)$ with a Gaussian Pyramid of eight levels with a scaling factor of $1/\sqrt{2}$. In combination with a median filtering kernel M, the expression of the calculation of consecutive positions of the dense pixel P_t and P_{t+1} is:

$$P_{t+1} = (x_{t+1}, y_{t+1}) = (x_t, y_t) + (M * w)|_{\bar{x}_t, \bar{y}_t} \tag{2.3}$$

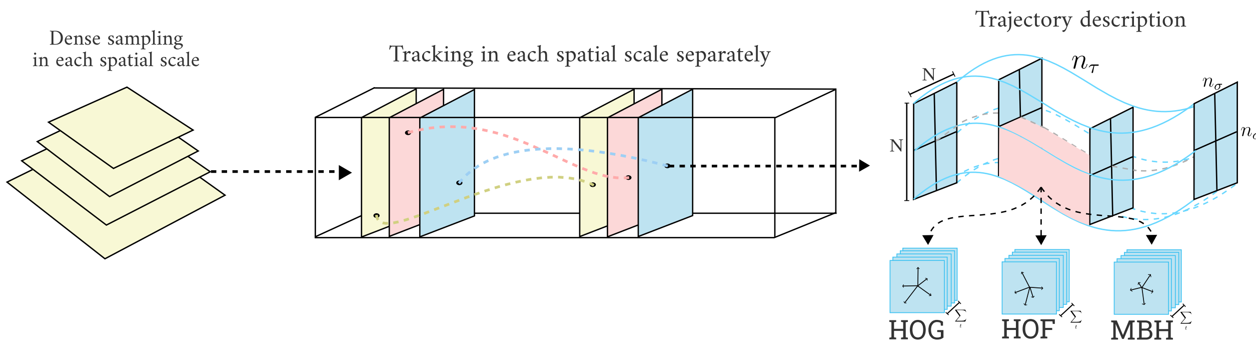

Dense sampling in each spatial scale
Tracking in each spatial scale separately
Trajectory description
N
N
n_τ
n_σ
n_σ
HOG
HOF
MBH

Figure 2.3: Dense trajectories extraction from a video by Wang *et al.* [4]. The process involves tracking multiple group of pixels across a fixed number of frames.

with $(\bar{x}_t, \bar{y}_t)$ the rounded position of (x_t, y_t) and $*$ the element-wise multiplication of matrices. Then a displacement vector $S = (\Delta P_t \dots \Delta P_{t+L-1})$ encodes these positions, where $\Delta P_t = P_{t+1} - P_t$, and normalizing by the sum of lengths of the displacements we get the normalized displacement vector:

$$S' = \frac{(\Delta P_t \dots \Delta P_{t+L-1})}{\sum_{\mathrm{i}=t}^{t+L-1} ||\Delta P_{\mathrm{i}}||} \tag{2.4}$$

The authors set the maximum number of frames $L = 15$ to prevent trajectory drifting, where the path of the dense pixel could warp to another location when, for example, there is a scene transition in the video.

2.2.3 Convolutional networks

In the last decade Convolutional Neural Networks have had a substantial contribution in computer vision, obtaining state of the art results in tasks like Object Recognition [36] and Segmentation [37], Action Recognition [38], among others. These achievements show the capabilities of this kind of neural network of learning complex spatial patterns; being the one presented by Krizhevsky *et al.* [7], commonly known as AlexNet, who popularized the use of convolutional networks by obtaining a Top-1 error of 15.3% in the Imagenet Challenge of 2012. This score was a huge achievement in comparison with the second place of the challenge, who got a Top-5 error of 26.2% [39], which showed the potential of this architectures for computer vision in the year 2012 until the present days. Also, pre-trained architectures can be used to get image embeddings for transfer learning [40, 41, 42], extending its use to video frames encoding.

Because there is a large number of different novel convolutional networks for a handful of tasks, we present two important ones that have a direct relation with the current research: Resnet [10] and ResNeXt [43].

He *et al.* [10] presented the use of residual connections that skip layers of convolution, architecture commonly known as Resnet, solving the gradient vanishing problem many networks with a high number of layers (deep networks) suffered from [44, 45, 46]. This improvement leads to deeper networks with better performance in visual pattern recognition tasks like Object Recognition. This architecture, which they called Resnet, outperformed state-of-the-art convolutional networks.

Xie *et al.* [43] present a highly modular architecture called ResNeXt, that adapts the Resnet architecture with the strategy of split, transform and aggregate similar to the Inception module [47]. ResNeXt splits the residual block of Resnet into a fixed number of thinner and identical blocks, quantity represented by the hyper parameter $\mathcal{C}$ called Cardinality. They show that the top-1 accuracy on the Imagenet-1k dataset [36] improves by increasing the Cardinality of the network than using wider blocks like Resnet.

Mettes *et al.* [48] experiment with two approaches to solve the TRECVID Multimedia Event Detection task, each one using a pre-trained convolutional network to extract visual feature vectors from a video. A key aspect of their research is the design of a reorganized and balanced dataset based on ImageNet called ImagenetShuffle. They propose this re-balance because several state-of-the-art convolutional networks in computer vision are trained in the Imagenet-1k dataset [49, 50], a subset of Imagenet with the top 1,000 classes more important, which represent most of the dataset, but because of the uneven distribution of images per class, this state-of-the-art network never sees most of the 90% of classes of the entire Imagenet dataset. They propose to re-balance the Imagenet dataset class hierarchy tree by merging child nodes to its parent with bottom-to-top operations, using as criteria the redundancy of the class, or the class not having enough images samples. When the merge stage finishes, they apply a sub-sample over the classes with too many images. They prove that using a state-of-the-art convolutional network pre-trained on ImagenetShuffle with a bottom-to-top re-balance of 4,437 classes showed a better precision on the TRECVID Multimedia Event Detection task than using the same convolutional network pre-trained of the standard Imagenet-1k dataset.

Although these architectures had shown excellent results in computer vision, recent researches [51, 52, 53] had shown they are easy to confuse by adding specific noise, unnoticed by to the human eye, that completely changes the output of a well performed convolutional network.

2.3 Video-to-Text Matching & Ranking

Since 2016, the National Institute of Standards and Technology (NIST) has hosted the video-to-text (VTT) challenge in its yearly video processing conference TRECVID. VTT matching subtask aims to match over 50 thousand video-sentence pairs from the social network *Vine Twitter* with predefined sentences. The difficulty remains in the small dataset available for training, with only two-thousand video-sentence registries.

Marsden *et al.* [6] extract video features related to objects, behaviors and places. They use a pre-trained VGG-16 network [54] for objects features. They train an AlexNet network [7] over the *Who What Where* dataset [54] to extract behavior features, and train a VGG-16 network over the *Places2* dataset [55] to obtain visual features for the place of the video. They process these features using *Okapi BM25* and Word2Vec [16] variants to get the best sentence that represents the content of the video.

Dong *et al.* [2] created a deep learning model for the VTT Matching & Ranking task at TRECVID 2017 [56], winning this challenge using an assembled of six models that follow the same architecture with some embedding and pooling variations, getting mean inverted rank scores (see Section 6.1) of 0.37 for test set 2, 0.5 for set 3, 0.6 for set 4 and 0.75 for set 5. The general architecture of their model generates mid-level representations of videos and sentences projecting them into a common space called *Visual Feature Space.* They encode a video with a pre-trained Resnet-152 network [57] over the ImageNet dataset [58]. They apply this transformation over a set of frames from the video, getting low-level representation for the whole video. After this, it passes them into a GRU network to retrieve spatio-temporal relations between the different object representations of the video [17]. A sentence mid-level representation results from concatenating 3 sentence vectorization methods: BoW, Word2Vec and GRU. Finally, a fully connected neuronal network transforms a corresponding video-sentence pair into a *visual feature space*, minimizing a rank error between the pair using a distance function.

Chapter 3

Baseline system overview

The baseline selected to compare against our proposal in the VTT matching & ranking task is the model of Li *et al.* [1], the winner of the VTT Matching & Ranking task of TRECVID 2018 competition. This submission uses the Dual Encoding for Zero-Example Video Retrieval model introduced by Dong *et al.* [11]. As schematized in Figure 3.1, this model encodes video v and sentence s in two multi-level sides. The first level consists of video and sentence embeddings extracted with pre-trained machine learning models, the second level treats the input as a sequence of items by using a Recurrent Neural Network, and the third level uses a Convolutional Neural Network that learns to extract a single vector representation from the second level. The output of each side corresponds to the video and sentence encoding $\phi(v)$ and $\phi(s)$, respectively.

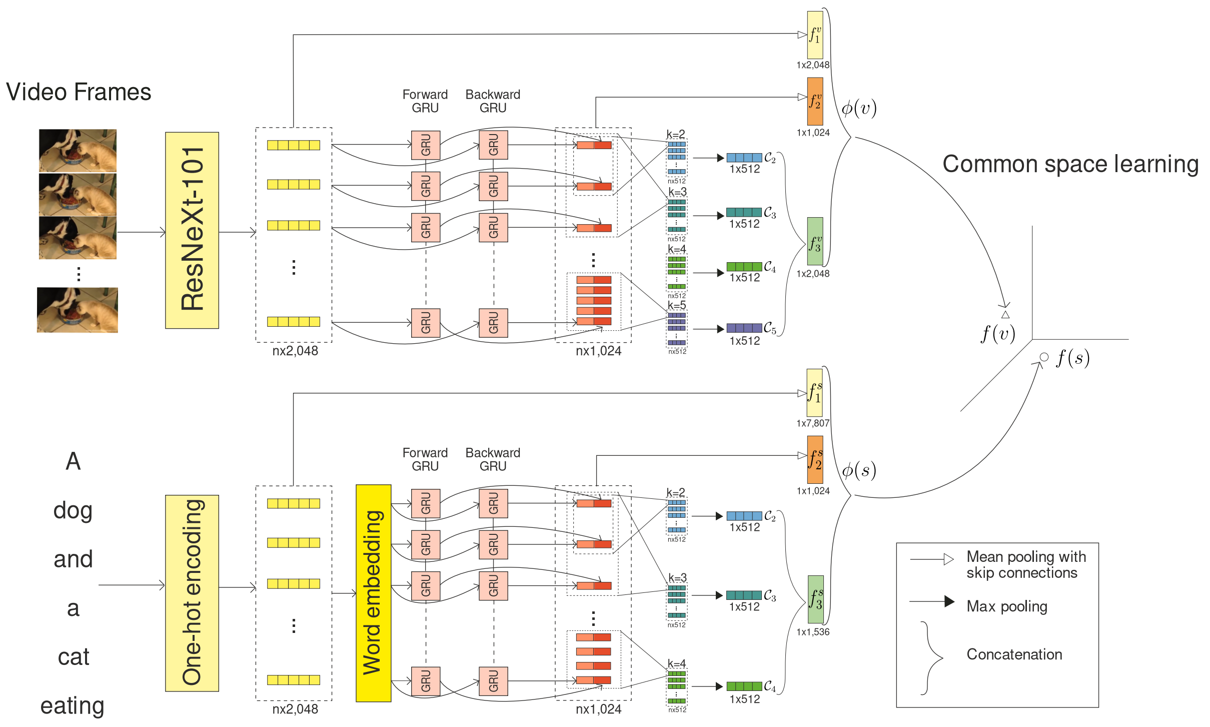


Figure 3.1: Multi-modal video sentence encoding model of Li *et al.* [1]

3.1 Video multi-level side

Given a video v, a set of frames n is collected by sampling every 0.5 seconds. Then they encode each frame using a ResNeXt-101 [43] network, pre-trained on the Shufflenet dataset [48] 4k, a class balanced variance of the Imagenet dataset [36] (see Section 2.2.3). With this pre-trained network they get a sequence of deep visual vectors $\{v_1, v_2 ... v_n\}$ that will serve as input of the first two encoding levels of the model. The output of each encoding level i is a single-dimensional vector f_{i}^v, and the last video encoding is the concatenation of the three encoding levels:

$$\phi(v) = [f_1^v, f_2^v, f_3^v] \tag{3.1}$$

3.1.1 Level 1. Global encoding by Mean Pooling

Dong *et al.* choose mean pooling, that is the average vector of the set of visual vectors $\{v_1, v_2 ... v_n\}$, as the first level encoding for a video because its simple and gives better results than the max pooling method, this is because, by definition, mean pooling considers every extracted visual feature vector, allowing the information of the complete video to flow in the forward and back-propagation stages; also, mean pooling is less affected by stochastic perturbations of the model initialization [59]. Therefore, the definition of the first level encoding is:

$$f_1^v = \frac{1}{n}\sum_{\mathrm{i}=1}^{n} v_{\mathrm{i}} \tag{3.2}$$

3.1.2 Level 2. Temporal-Aware Encoding by biGRU

The effectiveness of a bi-directional recurrent neural network [60] on using past and future context of a sequence makes it a robust candidate architecture to encode the temporal information of the video frames. Because of this, the authors had to choose between using a Bi-directional Gated Recurrent Unit (Bi-GRU) [17] or a Bi-directional Long-Short Term Memory (Bi-LSTM) [22], choosing the former because it has less parameter, needing less training data.

A Bi-GRU is composed of two layers of GRUs, a forward $\overrightarrow{GRU}$ and a backward $\overleftarrow{GRU}$. Let $\overrightarrow{h}_t$ and $\overleftarrow{h}_T$ be their hidden state in an specific step $t = 1...n$. The definition of them is:

$$\begin{aligned} \overrightarrow{h}_t &= \overrightarrow{GRU}(v_t, \overrightarrow{h}_{t-1}) \\ \overleftarrow{h}_t &= \overleftarrow{GRU}(v_t, \overleftarrow{h}_{t-1}) \end{aligned} \tag{3.3}$$

where v_t is the deep visual vector of frame t, and $\overrightarrow{h}_{t-1}$ and $\overleftarrow{h}_{t-1}$ are the previous hidden state of step t. Concatenating the hidden states $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ we get the bi-GRU hidden state for the step $t = 1...n$ $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$. The authors use a hidden state size of 512 for $\overrightarrow{h}_t$ and

$\overleftarrow{h}_t$; therefore, the hidden state size of the bi-GRU is 1024. Finally, the output of the second level video encoding is the mean pooling over the set of hidden states $[h_1...h_t]$:

$$f_2^v = \frac{1}{n}\sum_{t=1}^{n} h_t \tag{3.4}$$

3.1.3 Level 3. Local-Enhanced Encoding by biGRU-CNN

The mean pooling of the second level video encoding treats every h_t equally, potentially combining noise from unimportant video frames with relevant ones for the video encoding. To enhance local patterns between the set of $H = [h_1...h_n]$, Dong *et al.* adapts the 1d-convolutional network from Yoo Kim [61] and use it on top of the bi-GRU output. Let $Conv1\text{d}_{k,r}$ be a 1-d convolutional block that contains $r = 512$ filters of size $k \geq 2$. Feeding the set H of n hidden sizes from the bi-GRU, after zero padding, into $Conv1\text{d}_{k,r}$ produces a $n \times r$ feature map. Then they apply a *ReLU* as the non-linearity function of the mapping. Since n is different for every video, the authors use a max-pooling (a vector with the max values of each dimension over a set of vectors of equal size) after every convolutional block $Conv1\text{d}_{k,r}$ to generate a vector c_k of fixed length r. To ease understanding, a filter of size $k = 2$ allows the interactions of two rows of H, with higher k more adjacent rows interact to get local information between them. Having this, the definition of c_k is:

$$c_k = \text{max-pooling}(ReLU(Conv1\text{d}_{k,r}(H))) \tag{3.5}$$

For a multi-scale representation, Dong *et al.* apply multiple convolutional blocks with $k = 2, 3, 4, 5,$. The concatenation of their outputs c_k produces the third and final level of video encoding:

$$f_3^v = [c_2, c_3, c_4, c_5] \tag{3.6}$$

As a final step, by using skip-connections, the final video encoding is the concatenation of the three encoding levels:

$$\phi(v) = [f_1^v, f_2^v, f_3^v] \tag{3.7}$$

3.2 Text multi-level side

This side of the model has a similar architecture to the video encoding, but, in contrast, it relies on the use of standard word embeddings as input: a *Bag of Words* vector of the sentence and pre-trained *Word2Vec* [62] word embeddings.

Given a sentence s of length m and words $w_{\mathrm{i}} \in s$ with $\mathrm{i} = 1...m$, the model generates a one-hot vector $\overrightarrow{BoW}(w_{\mathrm{i}})$ per word following the standard *Bag of Words* approach described in Subsection 2.1.1. Then the **first encoding level** is equal to the average vector:

$$f_1^s = \frac{1}{m} \sum_{w \in s} \overrightarrow{BoW}(w) \tag{3.8}$$

For the **second encoding level** using bi-GRU, the architecture uses the one-hot vectors $BoW(w_{\mathrm{i}})$ from the previous level to get from a fixed dictionary the *Word2Vec* of each word, $w2v(w_{\mathrm{i}})$. The origin of this word embedding dictionary is a *Word2Vec* model made for a previous multi-modal architecture of Dong *et al.* [63], pre-trained over a dataset of over 30 million Flickr picture tags. The authors use these vectors on a bi-GRU the same way they use the deep visual vectors v_t in the second video encoding level of Section 3.1.2. The rest of this encoding level is the same as its video counterpart, so if we define $h_{\mathrm{i}} = [\overrightarrow{h}_{\mathrm{i}}, \overleftarrow{h}_{\mathrm{i}}]$ as the concatenation of the hidden states of each step of the bi-GRU, the second level encoding definition is:

$$f_2^s = \frac{1}{m} \sum_{\mathrm{i}=1}^{m} h_{\mathrm{i}} \tag{3.9}$$

Finally, the **third encoding level** for a sentence follows the same biGRU-CNN approach as the video encoding, with the difference that they use only three convolutional blocks $k = 2, 3, 4$. With this, the third sentence encoding level is:

$$f_3^s = [c_2, c_3, c_4] \tag{3.10}$$

and the final sentence encoding is the concatenation of the three levels:

$$\phi(s) = [f_1^s, f_2^s, f_3^s] \tag{3.11}$$

3.3 Common vector space

The video and sentence encoding vectors $\phi(v)$ and $\phi(s)$ are not directly comparable mainly because of size miss-match. For this reason, a fully connected architecture is used to map $\phi(v)$ and $\phi(s)$ into a common vector space where a distance function compares them. The authors choose VSE++ [64], a multi-modal architecture that brings image and text to a common vector space using an affine transformation, i.e., a function with the form $f(x) = ax + b$ with x a vector, a a matrix of trainable weights and b a bias vector with trainable values. Let $f(v)$ and $f(s)$ be the affine transformation of a video and sentence, their definition, shown next, in the work of Dong *et al.* adds a batch normalization layer (BN) on top of the definition of VSE++:

$$\begin{aligned} f(v) &= BN(W_v \phi(v) + b_v) \\ f(s) &= BN(W_s \phi(s) + b_s) \end{aligned} \tag{3.12}$$

where W_v and W_s are trainable matrices for the video and sentence mapping, and b_v and b_s their corresponding bias vectors. The vector size in the common space chosen by Dong *et al.* is 2048. Therefore, the sizes of the matrices and bias must satisfy:

$$\begin{gathered} \|W_v\| = 2,048 \times (\|f_1^v\| + \|f_2^v\| + \|f_3^v\|) = 2,048 \times (2,048 + 1,024 + 4 \times 512) \\ \|W_s\| = 2,048 \times (\|f_1^s\| + \|f_2^s\| + \|f_3^s\|) = 2,048 \times (\|Vocabulary\| + 1,024 + 3 \times 512) \\ \|b_v\| = \|b_s\| = 2,048 \end{gathered} \tag{3.13}$$

For the training process, this work uses an improved Margin Ranking Loss [64], which penalizes according to the hardest negative examples for video and sentence in a mini-batch. Let $S_\theta(v, s) = S_\theta(f(v), f(s))$ a similarity function between the a video v and sentence s, according to the parameters θ of the model. The loss function $\mathcal{L}(v, s; \theta)$ between a corresponding video-sentence pair is:

$$\begin{aligned} \mathcal{L}(v, s; \theta) = {} & max(0, \alpha + S_\theta(v, s^-) - S_\theta(v, s)) \\ & + max(0, \alpha + S_\theta(v^-, s) - S_\theta(v, s)) \end{aligned} \tag{3.14}$$

where s^- and v^- are the hardest negative examples in the mini-batch for v and s respectively, and α is a constant that defines a margin condition for negative and positive examples. For ease of understanding, the hardest negative example s^- of a video vector v is the sentence vector different from the corresponding s that maximizes the $S_\theta(v, s^-)$. They train the entire network to minimize this loss function, except for the pre-trained Resnet-152 used to extract the deep visual feature from the video frames.

Chapter 4

Enhanced Video-To-Text model

The Video-to-Text Matching & Ranking task requires a multi-modal architecture that encodes video and sentence. For video encoding, many models extract static information using convolutional models, like Resnet [10] or optical flow histograms [65] pe-trained for object recognition; but to extract temporal information a common technique relies on the use of recurrent networks [17, 22] to encode the sequence of extracted visual vectors into a single vector representation [11]. What if we directly extract the temporal information of the video, complementing the static information retrieval already done by the state-of-the-art models in the VTT matching& ranking task? What if we rely on the movement of pixels in the video across multiple frames to encode the actions portrayed in it? To answer these questions, we propose the use of Dense Trajectories [4] with a Bag of Words approach used in the same paper to encode the temporal information of the video into a single vector representation. We hypothesize that this technique can generate better temporal representation of the video to improve the performance of the baseline on the VTT matching & ranking task of TRECVID 2018..

For sentence encoding, multi-modal VTT models [6, 2] rely on word embeddings, like Word2Vec [62], in combination with a recurrent network to learn long-term sequence patterns [22, 17]. But as Word2Vec is an embedding for words, InferSent [3] is an embedding for a complete sentence, with high transfer learning capabilities; because of this, besides our Dense Trajectory research on Video-to-Text, in this work we also study the effectiveness of the InferSent model in the Video-to-Text Matching & Ranking task.

To prove our hypothesis: "the use of Dense Trajectories or InferSent embeddings improves the performance of the model of Li *et al.* [1] for the Video-to-Text Matching & Ranking task in TRECVID 2018", we focus on modifying a state-of-the-art model in the video-to-text matching task, empirically proving if the use of Dense Trajectories and/or InferSent embeddings increases the performance of the current best-performed model.

The baseline for the models of this research is the approach created by Dong *et al.* [11] for the Video Matching & Ranking task of TRECVID 2018 competition (see Chapter 3). We choose this model for several reasons: the concatenation of different levels of encoding gives us the modularity needed to test new video and sentence embeddings; the authors are the winners of the Video to TextMatching & Ranking task of the TRECVID competition the last 3 years [56, 66, 67] where they won with this model the 2018 competition with state-of-the-art results; the fact the model has a public source code in Python 2.7 using the machine learning framework Pytorch[1]; and its paper [11] has a clear and consistent description.

For the video side we test if the use of Dense Trajectories embeddings increases the performance of the model in two different configurations: the first replaces the existing ResNeXt-101 embedding along with the three original levels of encoding with our Dense Trajectories descriptors, while in the second we add the new Dense Trajectory embedding as an additional encoding to concatenate with the original three levels. Additionally to these proposed configurations, we also test the importance of the specific ResNeXt-101 Dong *et al.* use in their work, an architecture pre-trained on the Shufflenet [48] dataset. To do this, we test every model variation under study that uses a ResNeXt-101 encoding, with the network pre-trained in Shufflenet and the same architecture pre-trained in the standard Imagenet-1k dataset [36] and compare their final results.

For the sentence side we want to test if the use of InferSent [3] embeddings outperforms the baseline model in two different configurations: by replacing with it the complete sentence side of the base model, or adding it as a fourth sentence encoding by concatenating it to the three existing ones. We theorize one of these configurations using the InferSent embedding, whose design generates representations of a complete sentence rather than each word separately, can outperform the three independent words embeddings of the baseline model in the VTT task of TRECVID 2018.

In this chapter, we present the model of study, with its proposed changes to video and sentence encodings, along with the different variations we use in combination with the original ones.

4.1 Video representation

The three video encoding levels of Dong *et al.* [11] rely only on the deep visual vectors extracted using a pre-trained ResNeXt-101 [43, 48] on an object classification task. We hypothesize that this visual task trains the CNN to recognize spatial or static patterns, resulting in a loss of valuable temporal information when extending it to video frames extraction, an issue that the second encoding level may not recover by using a recurrent network. To address this problem, we propose to encode the temporal information of a video using Dense Trajectories [4] in combination with a Bag of Words approach.

As another research axis for video representation, we study the effect of using a ResNeXt-101 architecture pre-trained in a standard object classification dataset like an Imagenet-1k [36] instead of Shufflenet [48]. With this, we determine the importance of such specific architecture on the baseline model performance.

[1]`https://github.com/danieljf24/dual_encoding`

4.1.1 Dense Trajectories embedding

We propose to use the normalized displacement vector S' of Eq. 2.4 to encode the temporal information of the video, combining it with a Bag of Words (BoW) approach to create a summary vector of the thousand of dense trajectories extracted from a video. The BoW technique uses a clustering method for axis x and y of a dense trajectory, independently. Finding a set of codewords or center vectors that can represent the dense trajectories extracted from the target videos, in this case the videos and GIFs of the training dataset described in Section 5.1.

For the clustering algorithm, we choose an extension of K-means [68] for high volumes called MinibatchKmeans [69]. This algorithm solves the classical K-means objective using mini-batch SGD, allowing us to cluster a large number of trajectory descriptors S' that would otherwise not fit in primary memory.

The process of generating the codewords for the dataset comprises four stages:

1. Extraction of the descriptors S' from the training dataset videos of MSR-VTT, getting S'_{i} the normalized displacement vector of video $v_{\text{i}} \in \mathcal{D}$ and $\text{i} \in \{1...|\mathcal{D}|\}$.
2. For the x-axis and y-axis coordinates of a dense trajectory: selection of the number of codewords using the elbow method by looking into the loss of mean quadratic error of the clustering for each one of the number of codewords of study (see Figure 5.9). For this step we only use the vector S' of the 10% of the training dataset videos of the complete training dataset (MSVD + MSR-VTT + TGIF).
3. Once we select the number of codewords for each axis, we cluster all the trajectory descriptors S' of the training dataset, separating the x-axis and y-axis coordinates, creating a final cluster model to assign each displacement vector.
4. Finally, after finishing the complete clustering, we use the cluster model to assign a codeword or cluster to each S' of a video, creating two Bag of Words vectors that represent the thousands of S' with a single vector representation, one vector for the x-axis coordinates of S' and one for the y-axis..

With the Dense Trajectory-Bag of Words (DT-BoW) embeddings created, we propose to use them in the model of Dong *et al.* [11], to test if the model with these new embeddings outperforms the base model using the same training and testing datasets. For this purpose, we present two variations using DT-BoW embeddings to encode the temporal information of the video. All these variations keep the sentence side of the base model with no changes.

- **DT-ONLY** uses the DT-BoW descriptor as the only one to encode a video (see Figure 4.2).
- **DT-ALL** uses the DT-BoW vector as an additional encoding level for the video, concatenating it to the final video representation $\phi(v)$ (see Figure 4.1).

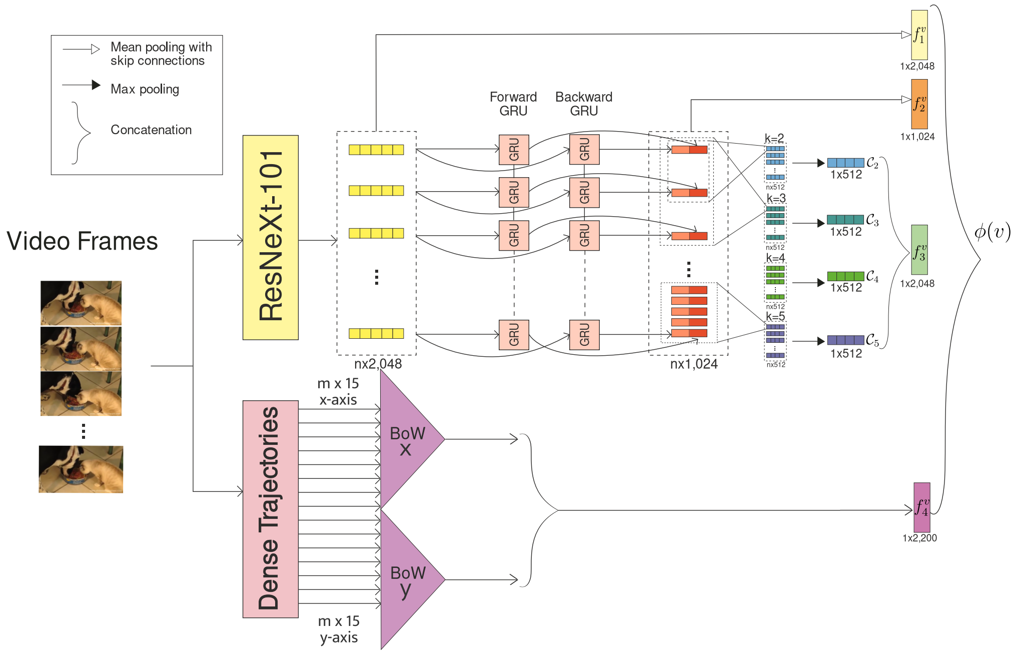


Figure 4.1: DT-ALL variation where the DT-BoW descriptor adds to the video representation. Figure shows only the video side of the model. We emphasize the ResNeXt-101 is already pre-trained, thus it has fixed parameter values.

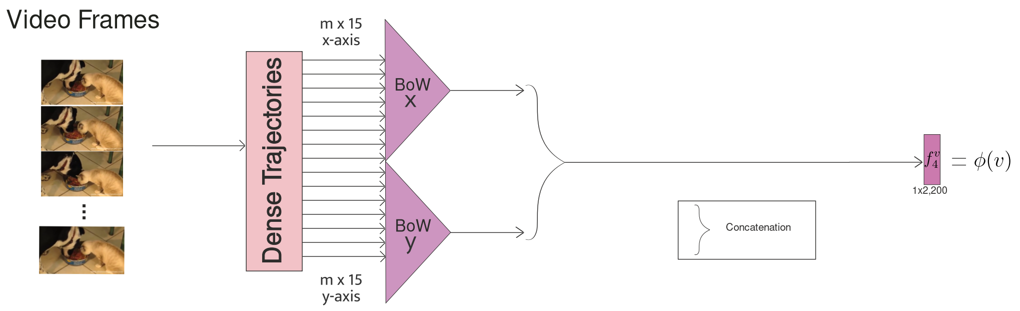


Figure 4.2: DT-ONLY variation where the DT-BoW descriptor replaces the ResNeXt-101 embedding of the base model. Figure shows only the video side of the model.

Dense Trajectory implementation

The official C++ implementation of Dense Trajectories[2] calculates the trajectory descriptors S' using the OpenCV library for the Färneback optical flow algorithm [35], with additional optical flow descriptors (HOG, HOF and MBH) the authors use in their research. For ease of use we implement a Python 3 library[3] that calls the C++ implementation with the Python API of C/C++.

The official implementation uses the CPU version of Färneback optical flow algorithm [35], while the same OpenCV library has a CUDA implementation that can speed up the process. That is why we also modify the C++ implementation to use this CUDA version, showing its speed-up in Section 5.5.

4.1.2 ResNeXt-101 embedding

To encode a video or GIF, the authors of the base model [11] extract visual feature vectors from the video frames using a state-of-the-art convolutional neural network, specifically a ResNeXt-101 pre-trained on a specific object classification dataset called Shufflenet [48]. Section 2.2 of related literature explain the idea behind the ResNeXt architecture and why they are important in vision processing tasks.

We test our models of study with two pre-trained ResNeXt-101 architecture variations: the one pre-trained in Shufflenet [48], and an architecture pre-trained on Imagenet-1k dataset [36]. With these variations of the model we want to prove the importance of the specific architecture that the baseline uses to achieve state-of-the-art performance in the VTT matching & ranking task.

We get the Pytorch implementation of the ResNeXt-101 pre-trained in Shufflenet from the official repository[4], while for the Imagenet-1k variation we get a pre-trained ResNeXt-101-64x4d using the MXNet[5] deep learning framework. We convert this model to Pytorch deep learning framework by using the MMdnn[6] Python library, transforming the MXNet model into a data structure called Intermediate Representation (IR), and then translating it into Pytorch source code with the same structure of the original MXNet implementation and the capacity to load the same parameters. We tag the Shufflenet version with *SHU*, while the imagenet-1k with *MXN*. We emphasize that both models have their parameters trained in their corresponding dataset and this study does not fine tune them on the VTT Matching & Ranking task; we use them to extract visual feature vectors from the frames of a video.

[2]`https://lear.inrialpes.fr/people/wang/dense_trajectories`
[3]`https://gitlab.com/nbravo/pydensetrajectories-gpu`
[4]`https://github.com/danieljf24/dual_encoding/issues/4#issuecomment-497904531`
[5]`https://mxnet.apache.org/`
[6]`https://github.com/microsoft/MMdnn`

4.2 Sentence representation

The triple sentence encoding in the model of Dong *et al.* 2018 uses two standard word embeddings: Bag of Words and Word2Vec [62]. Although these approaches had shown an excellent performance in many NLP tasks [70, 71] independently, their development focus on quantifying words instead of sentences. The authors of our base model cope with this problem by averaging across all one-hot vectors of a sentence or by using a recurrent neural network to capture the temporal relationship across the entire sentence. However, there are sentence encoders like *InferSent* [3] that had shown an excellent transfer-learning performance in multiple NLP tasks [3].

We propose the use of *InferSent* as a sentence embedding for the Matching and Ranking model of Dong *et al.* [11]. We hypothesize that encoding a text using this sentence-specialized embedding could improve the training of the base model because of a better context quantification and, therefore, an easier mapping to the common vector space. We test two different variations of it to compare which one performs better: by replacing the complete multi-level encoding of a sentence, or as an extra encoding level f_4^s we concatenate to the original ones.

Figures 4.3 and 4.4 show the sentence side of Dong *et al.* 2018 model with the different proposals using *InferSent*.

- **INFERSENT-ONLY** corresponds to use the *InferSent* embedding of a sentence as its only encoding vector of the model, mapping it into the common vector space using a fully connected layer. This configuration aims to test if the pre-trained *InferSent* embeddings creates a better complete encode that the concatenation of the three levels of Dong *et al.* 2018, with the hypothesis *InferSent* encodes static and temporal information of the sentence at the same time. The advantage of this approach is the lack of parameters to adjust because of the removed GRU and convolutional network, speeding up the training process by converging the optimization process quicker. (see Figure 4.3).
- **INFERSENT-ALL** uses the three sentence encodings of the original model with the *InferSent* vector as an additional encoding level, concatenating it to the final encoding vector $\phi(s)$. The advantage of this approach is that it allows one to avoid the loss of valuable information when deleting an encoding level, but with the disadvantage of adding more trainable parameters to the fully connected layer that maps to the common vector space. (see Figure 4.4).

The *InferSent* model used in this work is a public version[7] trained in the Stanford Natural Language Inference [29]. The vocabulary size selected for this model is 1,000,000 since it is the recommended size by the authors. Previous to the training and testing process, we generate the embeddings for each sentence in the training datasets since the *InferSent* model is a pre-trained model whose parameters will not be fine-tunning for the Matching and Ranking task.

[7] `https://github.com/facebookresearch/InferSent`

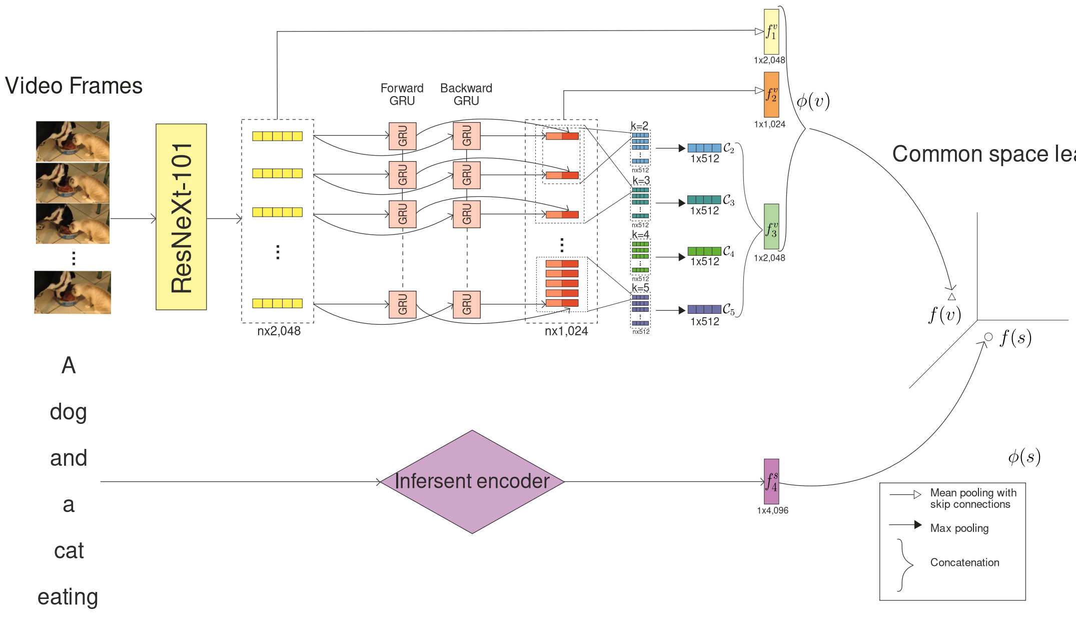


Figure 4.3: INFERSENT-ONLY model where the InferSent descriptor is the only sentence representation in the base model. This figure only shows the sentence side of the model. The video side has no changes.

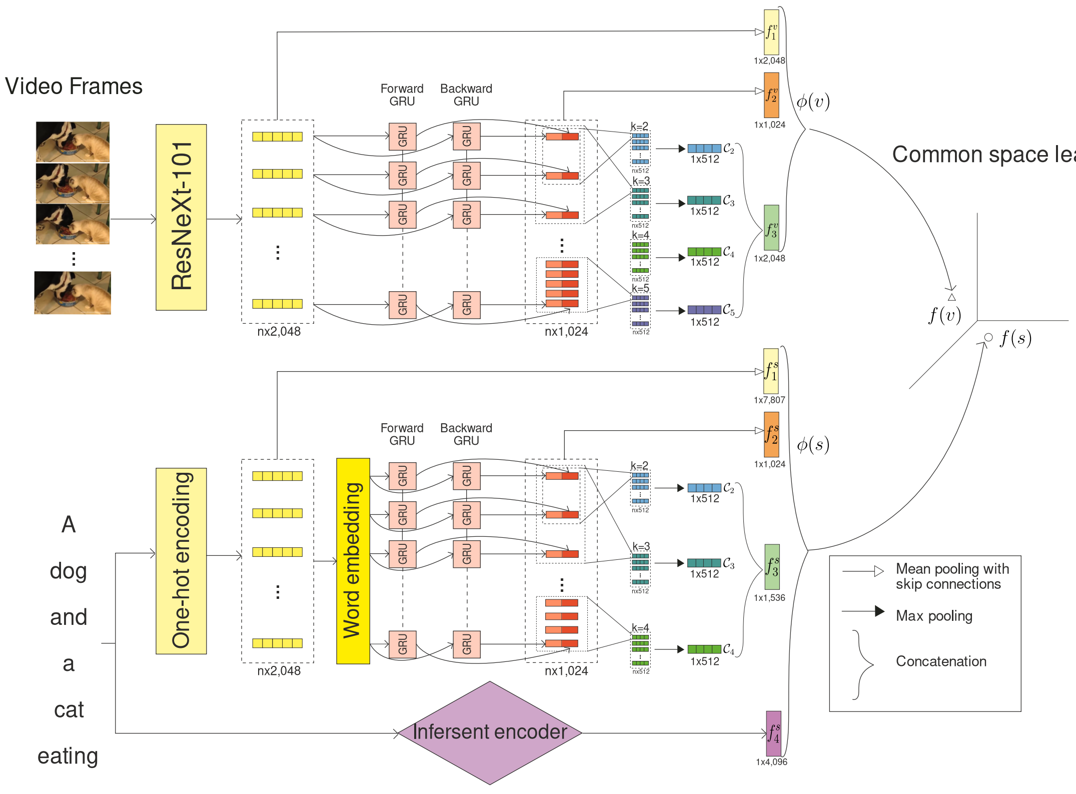


Figure 4.4: INFERSENT-ALL model where the InferSent descriptor adds to the existing descriptors of the sentence representation in the base model.

4.3 Distance functions for the common vector space

In all the different models we test, videos and sentences converge in a common vector space that relates them using a distance function. The selection of this function is a key factor both in the training process, by the Triplet Ranking Loss we define in section 4.3.1, and the testing process.

The distance functions we test with are the **Euclidean Norm** L^2 and the **Cosine similarity** between vectors. To define them, let $f(v)$ be the vector representation of a video in the common vector space, and $f(s)$ the vector of a sentence, both of size N. The definition of the distance functions in study is:

$$\begin{aligned} L^2(f(v), f(s)) &= \sqrt{\sum_{\mathrm{i}=1}^{N}(f(v)_{\mathrm{i}} - f(s)_{\mathrm{i}})^2} \quad \in [0, \infty) \\ \mathrm{Similarity}(f(v), f(s)) &= \frac{f(v) \cdot f(s)}{||f(v)||\ ||f(s)||} = \frac{\sum_{\mathrm{i}=\mathrm{i}}^{N} f(v)_{\mathrm{i}} f(s)_{\mathrm{i}}}{\sqrt{\sum_{\mathrm{i}=1}^{N} f(v)_{\mathrm{i}}^2}\sqrt{\sum_{\mathrm{i}=1}^{N} f(v)_{\mathrm{i}}^2}} \in [-1, 1] \end{aligned} \tag{4.1}$$

The Cosine Similarity represents the cosine of the lowest angle between vectors $f(v)$ and $f(s)$, therefore this distance function does not consider the length of the vectors, only the direction of them.

4.3.1 Optimization

The objective of the model in the common vector space is to bring the vectors of related video-sentence pairs closer while separating the unrelated ones. To achieve goal the model uses a Margin Ranking Loss or Triplet Ranking Loss function [64] of Eq. 3.14, this function uses a video vector as a relative position (anchor) and the nearest non-related sentence (hard negative) to calculate the path its related sentence must do to get closer.

Giving the model parameters θ, and a training dataset of related video-sentence pairs $D_{train} = \{(v_{\mathrm{i}}, s_{\mathrm{i}})\}$ $\mathrm{i} \in [0...|D_{train}|]$, the optimization process aims to minimize the sum of the Margin Ranking Loss $\mathcal{L}$ of every related pair $(v_{\mathrm{i}}, s_{\mathrm{i}})$:

$$\min_{\theta} \sum_{\mathrm{i}=0}^{|D_{train}|} \mathcal{L}(v_{\mathrm{i}}, s_{\mathrm{i}}; \theta) \tag{4.2}$$

To achieve this, we use Stochastic Gradient Descent [72] with the Adam optimization algorithm [73]. For the step size α we use the same approach of Dong *et al.* : an initial value of 0.0001, decreasing by 0.99 every epoch; we halve the step size if there is no validation loss improvement by 3 consecutive epochs. The objective of this practical approach is to force a finer gradient optimization when no validation score gain is obtained with the current step size of Adam. To prevent an exploding gradient during each back propagation [74], a gradient clipping of 2 is set to the norm of the complete gradient vector. There is no weight decay regularization.

4.4 Parametric complexity

We select the values of hyper-parameters of the base model when validating our implementation in Section 6.2. The hyper-parameters for our proposed models are to replicate the performance results of the baseline, and to do a correct comparison against the models with the proposed video and sentence embeddings of this study.

The complexity of each model is a factor to consider when dealing with multiple ones that try to solve the same task. For this reason we present in Table 4.1 the number of parameters of proposed model, comparing them with the original base model of Dong *et al.* [11]. Section 6.2 list the hyper parameters of the base model and their selected values.

Model	Text side parameters	Video side parameters	Total Parameters
Dong *et al.* 2018	52,131,856	46,157,824	98,289,680
DT-ONLY	52,131,856	10,264,144	62,378,000
DT-ALL	52,131,856	56,397,824	108,529,680
INFERSENT-ONLY	8,394,752	46,157,824	54,552,276
INFERSENT-ALL	60,520,464	46,157,824	106,678,288

Table 4.1: Number of trainable parameters of each model of study.

Chapter 5

Data and pre-processing

The datasets for the Matching & Ranking task comprises multimedia data like videos or GIF, and sentence annotations that describe the content of these multimedia. To get a diverse and large amount of data, we combine 3 complete datasets for the training process (MSVD[14], MSR-VTT [13], TGIF [12]), and for the validation and testing of the models we use the official VTT matching & ranking evaluation datasets published by TRECVID. We selected these datasets because they are the data used by the authors of the base model [11], and therefore we focus on using the same data to make a valid performance comparison between the models. In these sections, we present each dataset, explaining in detail their characteristics. We also present the multiple data processes we apply in advance to the training stage to save computation time among multiple models and runs.

5.1 Training dataset

We train every model under study using the joint of the MSVD, MSR-VTT and TGIF datasets. This creates a large amount of videos and sentence patterns the models will try to learn, a necessary aspect considering the unrestricted and unpredictable classes of videos the models will process during their expected use.

5.1.1 MSVD

This dataset [14], part of the Microsoft Research Video Description Corpus, comprises of 1,970 Youtube videos of 10 to 25 seconds of duration. The complete dataset has about 500K multilingual human annotated sentences describing the content of the videos. For our experiments we use only the annotations in English with a *clean source* (parameter set by the authors of the dataset), filtering a great volume of the dataset, and leaving 97,824 video-sentence pairs to use for training as shown in Table 5.1. Figure 5.1 shows an example of a video with its corresponding sentences, where we can see MSVD descriptors have a high diversity of length, vocabulary and grammatical structure for a single video.

- A boy runs into wall.
- A boy runs and throws himself against the wall.
- A boy runs into a wall.
- A guy jumps into a wall.
- A guy runs, bounces off a wall and falls down.
- A kid comically runs and jumps into a wall.
- A kid runs into a wall and falls down.
- A male runs, jumps and slams the front of his body against a wall, sideways.
- A man is jumping into a wall.
- A man runs and jumps into a wall.
- A man runs and jumps on a wall.
- A man runs, jumps and slams himself against a wall.
- A man runs, jumps, bangs his entire body on the opposite wall and falls on the floor after arguing with another man.
- The man threw himself against the wall.

Figure 5.1: MSVD video example where a its corresponding sentences shows a high diversity of length, vocabulary and grammatical structure.

Nº of videos	1,912
Nº of frames	541,866
Max sentences by clip	462
Video-Sentence pairs	97,824

Table 5.1: MSVD dataset main features.

5.1.2 MSR-VTT

The Microsoft Research Video-To-Text dataset [13] is a large-scale video and sentence dataset for video understanding tasks. This dataset has 7,128 videos from Youtube™ with sound, distributed in 10,000 clips of short duration. The authors use 257 popular queries from a commercial engine for the selection of the videos, using the Amazon Mechanical Turk™ service to annotate 20 sentences for each video. The authors supervised the annotation process with a quality control protocol, leading to a video dataset of high volume and quality. Figure 5.2 shows a video example with its corresponding sentences, where this descriptors also have a high diversity of length, vocabulary and structure, but, unlike the other training datasets and because this dataset preserves sound of its videos, some descriptions refereed to the audio of the clip instead of its visual content.

Nº of videos	7,010
Nº of frames	2,846,498
Max sentences by clip	20
Video-Sentence pairs	140,200

Table 5.2: MSR-VTT-train dataset main features.

- A clip about a girl being on the cover of sports illustrated.
- A girl is playing softball.
- A girl playing baseball.
- A girl with a bat.
- A little girl is headed to the major leagues in baseball.
- A man is talking about a baseball player.
- A man is talking about some sport s.
- A man talks about a girl being on the cover of a magazine.
- A man talks about a teenage girl who made the cover of sports illustrated.
- A news story talking about a girl athlete.
- A person is talking about sports courage.
- A reporter does a story on a young female athlete.
- A woman is walking.
- Monet Davis playing various sports.
- Women are playing baseball.
- Women are playing baseball.
- Younger children play sports.
- A reporter does a story on a young female athlete.
- A clip about a girl being on the cover of sports illustrated.
- A person is talking about sports courage.

Figure 5.2: MSR-VTT video example where a some of its sentences describe the audio of the clip instead of its visual content, e.g., "A man talking about a baseball player".

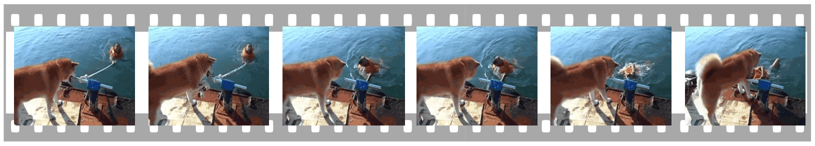
5.1.3 TGIF

The Tumblr GIF Description Dataset [12] comprises human annotated descriptions for a set of GIFs from the Tumbrl social media. In contrast with other video description datasets, TGIF has 125,781 GIF-sentence pairs with only a few repeated GIFs, leading to have on average one sentence per GIF. Also, using GIF media instead of videos does not allow using audio as a feature, therefore its descriptors only focus on the visual content of a GIF, unlike MSR-VTT. Figure 5.3 shows 3 different examples of this datasets with their unique sentence, where we can see these descriptions have a more concise and simple form than some of the other datasets. Because this dataset has only one sentence per GIF, it has much more scenes and visual variaty than the other dataset with more than 125 thousand different GIFs; although, the decision of using GIFs could generate problems because their lack of smoothness of framerate, aspect we review in next Section 5.1.4 and discuss it in Section 6.6.3.

Nº of GIFs	125,782
Nº of frames	3e19
Max sentences by clip	1
GIF-Sentence pairs	125,782

Table 5.3: TGIF dataset main features. The number of frames of this dataset is massive, however, since we extract one frame every 0.5 seconds, we use only a fraction for training.

A couple standing in the sea embrace each other.


A man wearing a life jacket is attached to a rope whilst in the water and a dog releases the rope.

A woman is dancing around as she sings on stage.

Figure 5.3: TGIF GIF examples with their corresponding unique sentence.

5.1.4 Framerate analysis

As shown in Table 5.4, the joint of the 3 described datasets generates a large amount of examples to train the model. On the other hand, this decision of combining videos and GIFs in the same collection can be worrying if we consider these types of multimedia do not share the same features, being the lack of audio its main difference between them; however, no model of this research uses audio as a multimedia descriptor. The feature that must be analyzed is the difference on framerate, also known as frames per second, between the 3 datasets because, by definition, the extraction of Dense Trajectories is highly affected by the smoothness of the video. Figure 5.4 shows the different distribution of framerates between each dataset, where we can see TGIF GIFs tend to have a framerate lower than 20 frames per second; unlike MSR-VTT and MSVD datasets, where the majority of their videos have 30 or more frames per second. This is a factor to consider in the training of DT-ONLY and DT-ALL model variants. To verify if TGIF improves or hinders the performance of a model variant that uses a Dense Trajectory descriptor, in Section 6.4.3 we train and analyze the DT-ALL variant using only the join of MSR-VTT and MSVD as training dataset, repeating the same experiment with the DT-ONLY model in Section 6.4.4. Their testing results are shown in Table 6.3, while Section 6.6.3 discuss these results and concludes if the use of TGIF has a negative effect the two models under study that rely on Dense Trajectory descriptors.

Nº of Videos and GIFs	134,880
Nº of frames from Videos	3,388,364
Nº of frames from GIFs	3e19
Video/GIF-Sentence pairs	363,806

Table 5.4: Training dataset main video and sentence features.

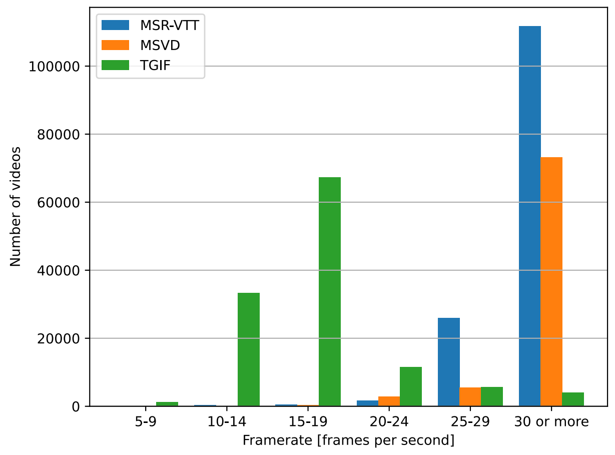


Figure 5.4: Framerate range distribution of the training dataset. Each video is counted 1 for each corresponding sentence. Framerates of 30 or more are combined in the same category because most video/GIFs in this range only have 30 or 60 frames per second.

5.2 Validation and testing datasets

To validate each iteration of a model during training and test the final model to compare its performance, we use official datasets for the VTT matching & ranking task of TRECVID, because they all share the same source for their videos, the Vine social media.

5.2.1 TRECVID 2016 and 2018 VTT official evaluation dataset

The TRECVID competition publishes a testing dataset for every one of their tasks, including Matching & Ranking. For this task, the staff of TRECVID created a dataset of videos from Twitter Vine^{TM} social media. This dataset comprises about 2,000 videos annotated by members of the staff, creating different subsets of videos with their respective annotation. Each year the staff publish the video URLs and sentences in different subsets with no relation to the videos, and after the competition is over, they publish the ground truth of the subsets, showing the relation between each video and sentence.

Each year between 2016 and 2018, the videos and sentences used in the evaluation is different, but because the competitors of 2017 and 2018 know these datasets share the same source, they use the ones from previous years to test their models and fit them. This same technique use Li *et al.* in their submission of 2018 [1], and the same strategy we use in this research. During the base model tunning, where we train and test the parameters of our implementation of Dong *et al.* 2019 [11], we use the set A of the 2016 version to validate each training iteration and the TRECVID 2018 VTT dataset to test its performance and validate if our implementation performs similar to the reported by the baseline authors. We explain this process in Section 6.2. For our model variations, we use the same approach: VTT 2016 set A for training validation and VTT 2018 for testing.

Table 5.5 compares the video and sentence features between the three different dataset. 2016 and 2018 versions have equal-sized subsets of sentences. Unfortunately Twitter Vine^{TM} removed the access to some videos of the VTT 2016 (151 videos) and 2018 (7 videos) datasets, and due to the lack of backup copies in the Web we validate and test the models under study without these videos. This issue is a factor to consider for the VTT 2016 because its high number of missing videos (151 of 1,915), but for the testing dataset VTT 2018, this issue may not have a great impact. However, the correct result reproduction of our baseline implementation (Section 6.2) validate the use of both datasets with missing videos.

Feature	2016	2018
Nº of videos	1,764	1,897
Nº of frames	308,879	337,181
Nº of subsets	2	5
Nº of sentences by subset	1,915	1,915

Table 5.5: VTT Matching & Ranking validation dataset main video and sentence features. 2017 version has multiple subsets with a different number of sentences.

Although the official 2018 dataset has improved its quantity of videos and sentences over the years, the quality of it as a benchmark metric is arguable because it has some sentences that poorly describe their video or do not describe it at all. Figure 5.5 shows some video examples of this problem in the official evaluation dataset for VTT Matching & Ranking of TRECVID 2018.

For the validation during the training of a model, we test each iteration over the subset A of the official VTT matching & ranking evaluation dataset from TRECVID 2016. We decided to not use the subset B for two reasons: it contains the same videos with different sentences and this repetition would complicate the interpretation and correctness of the validation metric described in Section 6.3.1. However, the lack of the subset B did not prevent the correct reproduction of the reported performance of the baseline, as showed in Table 6.2.

To test a model and calculate its performance metric, we use every subset of the official evaluation dataset from VTT matching & ranking of TRECVID 2018. The performance comparison of different models is independently across the five subsets, this is to make a correct comparison against the reported Mean Inverted Rank (see Section 6.1) of our baseline in the same challenge [1].

Subset A - A young woman in a black and pink bikini, lying on a blue, gray, yellow and white striped towel, rises and yells at someone.

Subset B - This file is offensive.

Subset C - A bikini-clad teenage girl with long-blonde hair lies face down on a towel until she hears another girl's voice saying that the bikini-girl likes girls, and the irate sunbather screams "I like boys" and makes an obscene gesture.

Subset D - A Caucasian woman in a two piece pink and black bathing suit is laying on her stomach on a striped towel and sits ups screaming.

Subset E - Girl in bikini, being taunted about sexuality by another girl, who screams that she likes boys, while gesturing oral intercourse.

Subset A - A man with dark hair in a gray t-shirt sits before a purple wall reading Respect, keeps his left hand in his pocket and puts his right hand to his mouth.

Subset B - A man with dark hair in a grey t-shirt pulls his hand from inside his jeans and then smells it as he sits down in front of a background that says "respect" with some public event going on around him.

Subset C - A man being disrespectful.

Subset D - A man wearing a grey teeshirt and black pants takes his hand out of his pants, sits down and smells his hand.

Subset E - A man, patiently waiting for an announcement, at some type of event in an arena setting.

Figure 5.5: Video examples with poor quality descriptors from the TRECVID 2018 Matching & Ranking official evaluation dataset. Sentences that do not correctly describe the content of the video are highlighted in red.

5.3 Testing dataset representation

An important aspect when choosing datasets to train a model is the analysis of their level of representation of the target dataset where the model performance will be tested, in our case the VTT official evaluation dataset of TRECVID 2018. Therefore, we must explore the potential similarities the training dataset (MSVD + MSR-VTT + TGIF) and validation dataset (TRECVID 2016 official VTT evaluation dataset) could share with the testing dataset selection. Since VTT matching & ranking task is a multimodal problem, the video-textual datasets that take place have a high number of characteristics to study, e.g. video and sentence topic distribution, topic unbalance, vocabulary distribution, and sentence grammar complexity. Because of this, we focus the study of representation levels based on the similarity in vocabulary between datasets, where we compare two aspects of a training collection against the testing dataset: its vocabulary distribution and the its most frequent words.

To compare the vocabulary distributions of the training and validation datasets against the testing dataset, the histograms of Figures 5.6 and 5.7 are used, respectively. Theses histograms show the word frequencies of each dataset, using the words of TRECVID 2018 sorted by frequency as a fixed index vocabulary for every histogram. No stop-words where counted because we consider their high frequency as an outlier. We also normalize the distributions by the total number of counted words of their corresponding collection, because the difference in quantity of sentences between different collections. In Figure 5.6 we can see vocabulary distribution of the training dataset (MSVD + MSR-VTT + TGIF) is slightly similar to the vocabulary of VTT TRECVID 2018, where they tend to share some of their most frequent words but not completely. This shows the most significant words of the testing dataset have a representative frequency in the distribution of the training dataset. On the other side, Figure 5.7 shows a similar behavior with the validation dataset VTT TRECVID 2016 set A, where it is also shown that the most important words of the testing dataset have a high representation in the validation dataset, which supports the selection of the VTT TRECVID 2016 set A as the data collection to validate each iteration of a model during its training stage.

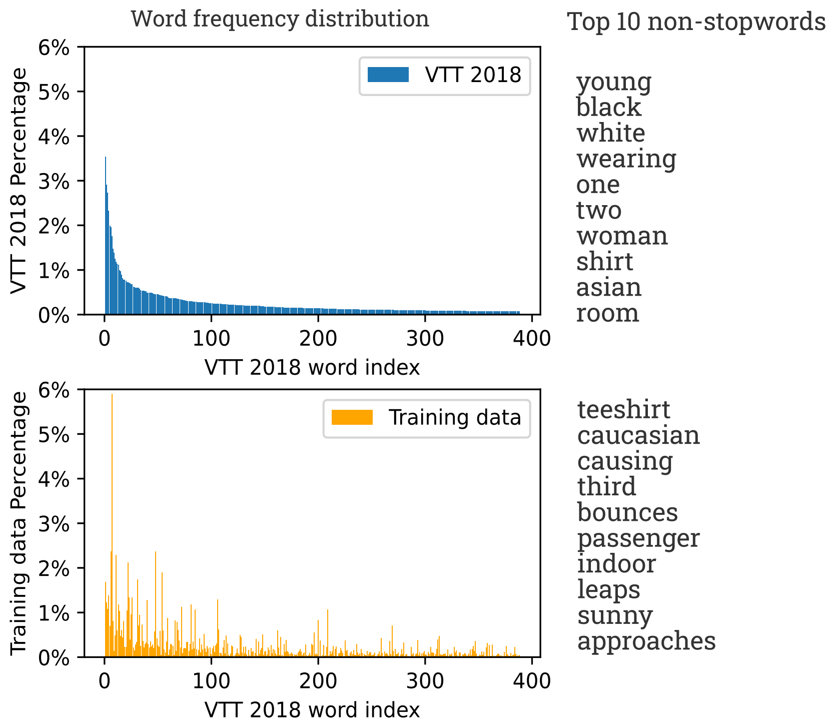


Figure 5.6: Word frequency distribution and top 10 words for the training dataset (MSVD + MSR-VTT + TGIF) and 2018 evaluation datasets. Only non-stop words with more than 50 repetitions are counted. X axis represents the vocabulary index of the VTT 2018 words sorted by frequency, the same index is used for the training dataset distribution for an easy comparison.

The most frequent words of each dataset are listed alongside their vocabulary distribution. This lists show the top 10 non-stop words of each dataset to give an idea of the most repeated objects, action or topics present in each collection. Figure 5.6 shows the training dataset shares some of its top words with the testing dataset, e.g. *teeshirt* and *shirt*, *caucasian* and *white*; but its top words focus less in objects and more in actions, e.g *causing*, *bounces* and *approaches*. In contrast, Figure 5.7 shows many top words of the validation dataset VTT TRECVID 2016 set *A* share similar topics with the top words of the testing dataset, e.g., *asian* and *oriental*, *woman* and *female*, or *shirt* and *tshirt*, characteristic that suggest a level of representation of VTT TRECVID 2016.

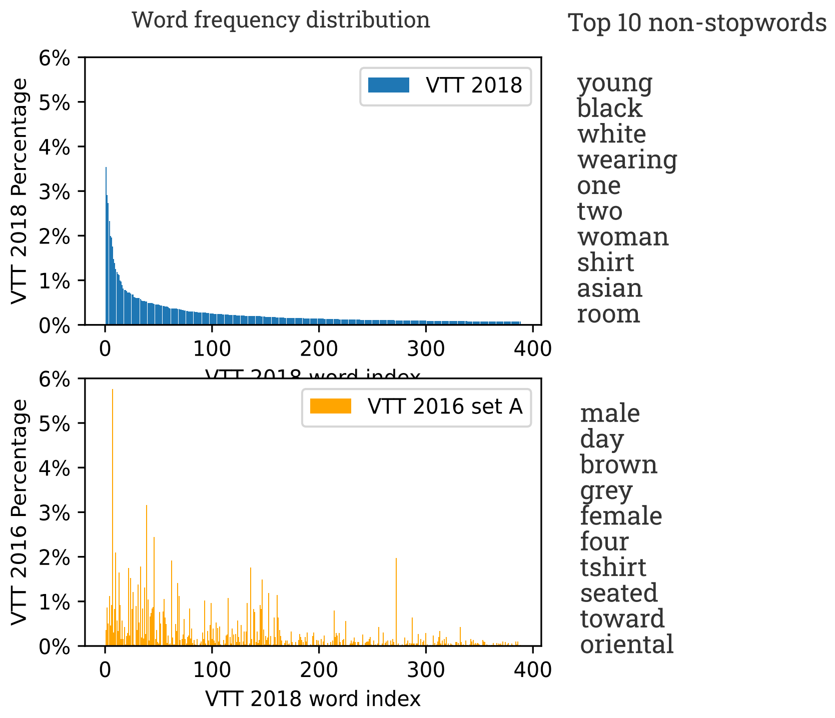


Figure 5.7: Word frequency distribution and top 10 words for the VTT 2016 set A and 2018 evaluation datasets. Only non-stop words with more than 50 repetitions are counted. X axis represents the vocabulary index of the VTT 2018 words sorted by frequency, the same index is used for the VTT 2016 set A distribution for an easy comparison.

The characteristics of interest of the training dataset suggest it has a level of representation of the testing data, especially its vocabulary distribution. The large volume of this dataset suggests its top 10 words being different to the top words of VTT TRECVID 2018 will not generate a major negative effect in the training process of the models under study, because the high quantity of topics, visual patterns, actions and objects it offers is a valuable feature for the VTT matching & ranking tasks, where no categories or topics are defined.

The characteristics of the validation dataset shows a level of representation of the testing dataset, where its reason could be its common social media source and the fact the 2018 version is an iteration of the VTT TRECVID 2016. Furthermore, the decision to use the official evaluation datasets from previous VTT challenges to validate the training of the models under study is a practice the authors of our baseline had done to win the challenge of TRECVID 2017 [2] and 2018 [1], therefore this is the validation dataset chosen to do a correct baseline comparison of the models.

5.4 Data pre-processing

Because of the complexity of the models under study and the high volume of data to train them, we speed up the training, validation and testing stages by pre-proccesing the video and sentence data. The importance of this preparation is to save time by generating the embeddings only once and reducing the secondary memory access during the mentioned stages. The first implementation decision takes importance if we consider the different embeddings the models use for video and text: visual feature vectors and Dense Trajectories for the former; and Bag of Words, Word2Vec [62] and sequence vector (GRU input) [17] the later. We pre-compuute and store every embedding in a solid state drive for fast reading during the training process.

5.4.1 Video and GIF feature extraction

For a GIF or video v we extract a frame every 0.5 seconds, creating a set of n frames $\{v_{\text{i}}\}$ i $\in \{1...n\}$. Each frame is a three-dimensional matrix of size *Width* $\times$ *Height* $\times$ *Channel* where *Channel* is the RGB channel of each pixel; this matrix is the input of the pre-trained ResNeXt-101. Using this model, for each frame we extract the visual feature vectors $M_{resnext101}(v_{\text{i}})$ from the last Mean Pooling layer, getting a set of visual feature vectors of size 2,048, which is the embedding for the video side of our models.

When training each model, every list of visual feature vectors in a batch must have the same length, we can't feed the model with variable-sized lists. Because of this problem, we set a fixed list length as the longest visual feature vector list in the batch, filling with zero vectors of size 2,048 for shorter video visual vectors lists. It is important to choose this list length threshold carefully to prevent GPU memory capacity errors. We choose a limit of 64 visual feature vectors because, as we see in Figure 5.8, there is a negligible number of videos, mainly from MSVD dataset, with more extracted vectors than this limit.

Dense trajectories extraction from a video or GIF use our Python 3 implementation described in Section 4.1.1. For a video v, the library's function *generate* creates a list $\mathrm{d}t(v)$ of trajectory vectors S' from Eq. 2.4. We save $\mathrm{d}t(v)$ into secondary memory for the clustering process. For the clustering we choose *MiniBatchKMeans* from the scikit-learn Python library[1] because it updates the clusters position by batches of data, while still having a convergence similar to the classical K-means algorithm [69]. We use the elbow method [75] to decide an optimum number of clusters for the millions of trajectory vectors of the training dataset. This technique is a heuristic that consists of visually or statistically find a cluster size from where the clustering evaluation function, e.g. its Davies-Bouldin index [76], does not have a significant improvement. For this method we randomly select 10% of the training videos (36,380) and test clusterings with 200 clusters to 8,000 clusters, with a difference of 100 between each cluster. It is important to remember the clustering is done independently for each axis of a dense trajectory, obtaining two histogram vectors that are concatenated in the DT-ALL and DT-ONLY models. We measure the mean squared error of each clustering and plot it against the number of clusters in Figure 5.9. The elbow method visually suggests 2,500 as an optimal number of cluster to work with, therefore we use this number of cluster for a complete clustering of the training dataset. When the complete clustering finish, we get the Bag of Words descriptor $BoW\mathrm{d}t(v)$ of a video v by assigning each of its trajectory vector $S' \in \mathrm{d}t(v)$ to the nearest cluster. Formally:

$$BoWdt(v)_{\text{i}} = \text{number of } S' \in \mathrm{d}t(v) \text{ nearest to cluster i} \tag{5.1}$$

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html`

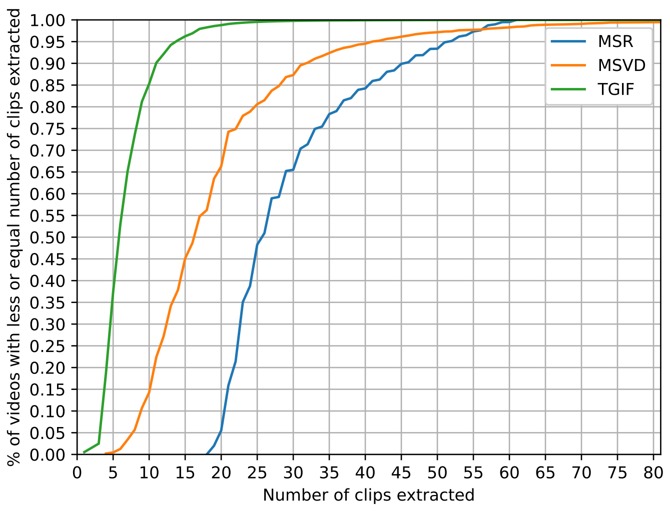


Figure 5.8: Percentage of training videos with less or equal extracted frames. If we consider a limit of 64 frames, only $\sim 1\%$ or 23 videos from MSVD dataset will have its visual features list cropped.

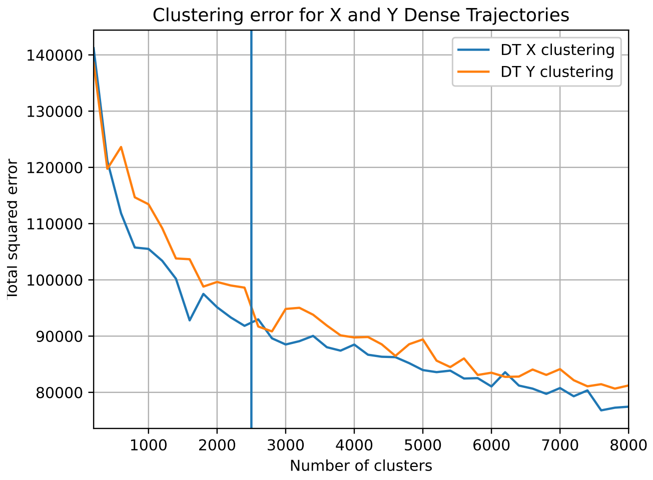


Figure 5.9: Mean square error of K-means clustering of Dense Trajectories over video from MSR-VTT. Cluster size between 200 and 8,000 clusters. we visually select 2,500 using the elbow method. 10% of training videos were used to do these multiple clusterings, processing a total of 110,754,058 dense trajectories.

5.4.2 Sentence vectorization

The first step to generate the input embeddings of a sentence is to create a fixed vocabulary V. We create it by counting every word in the training dataset D_{train} (MSVD + MSR-VTT + TGIF) and keeping only the ones with 5 repetitions or more; we adapt these criteria from the work of Dual Encoding by Dong *et al.* [11]. With this process over, the complete training dataset D_{train} we get a vocabulary of size $|V| = 10,192$ words.

- For the Bag of Words embedding we count each word occurrence in a sentence s, considering only words from the vocabulary V. This creates a vector $BoW(s)$.
- The embedding input necessary for the RNN of the model, in this case a single layer GRU [17] and schematized in Figure 5.10, is sequence of vocabulary indices of each word in the sentence; the network uses these indices to get the Word2Vec vector of each word from a lookup table loaded into memory, creating a sequence of Word2Vec vectors that is the input of the GRU. This lookup table has 4 special vectors with their values uniformly generated between -1 and 1: *INIT, END, UNK* and *PAD*. *INIT* and *END* mark the beginning and end of a sentence respectively, *UNK* is a placeholder for words not present in the vocabulary V, and *PAD* fills the sequence after *END* to create batches during training and validation, since all sequences must be of the same length equal to the larger sequence of the batch.

These two embeddings correspond to the original input selection for the model of Li *et al.* 2018 [1]. We pre-compute every embedding of every sentence of the training datasets MSR-VTT [13], MSVD [14] and TGIF [12] to speed up the training process.

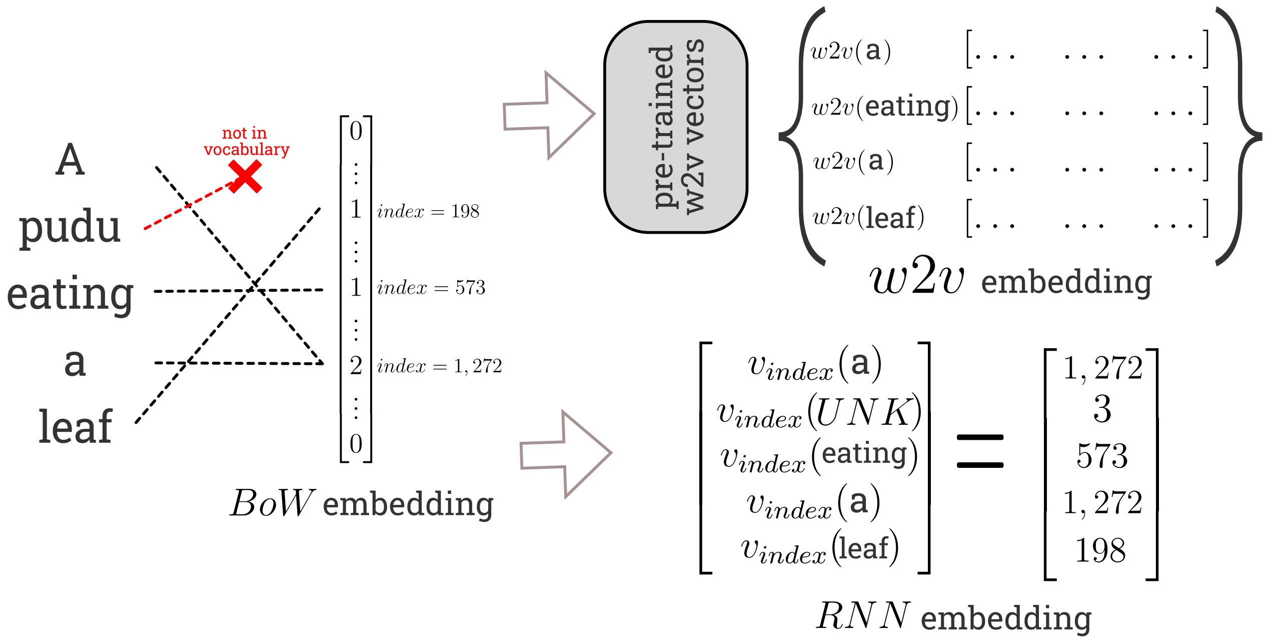


Figure 5.10: Example of the process to generate the 3 sentence embeddings the base model and our models use as input.

5.5 Dense Trajectories with CUDA

As mentioned in Section 4.1.1, we wrapped the official Dense Trajectories implementation[2] into a Python 3 library for easier usability[3]. During this process we noticed the original implementation does the Farneback optical flow [35] calculation in CPU with the OpenCV image processing library[4], but this library has a GPU version for such heavy algorithm[5] which we use to speed up the Dense Trajectory extraction of our Python 3 library.

To test the gain in time complexity of this implementation decision, we select two different sets of 100 random videos from the MSR-VTT dataset, and measure the total time of extracting Dense Trajectories from these videos using our Python library with and without CUDA implementation of the Färneback algorithm. Table 5.6 shows how this implementation decision speeds up in an average factor of 5.18 for the set 1, and a factor of 5.31 for set 2. When we are talking about more than 200 thousand videos and GIFs, this speed up saves a considerable amount of time.

	Set 1		Set 2	
	DT with CPU	DT with CUDA	DT with CPU	DT with CUDA
Mean (s)	1,211.5	233.7	1,174.0	221.2
Std (s)	15.9	6.1	0.7	1.0
Speedup	5.18		5.31	

Table 5.6: Comparison of Dense Trajectory generation with CPU and CUDA. Measure of mean and standard deviation in seconds.

5.6 Hardware

Because of the large number of data to process and parameters to train, we use a high-end custom personal computer to train, validate and test the models in study. Its specifications are:

- CPU: AMD Ryzen 7 2700 3200 MHz
- GPU: Asus Dual RTX 2070 8GB
- RAM: 32 GB 3200 MHz DDR4
- SSD: Kingston A2000 500GB M.2 NVMe
- MB: Asus TUF x470 Plus Gaming
- PSU: Seasonic FOCUS Plus 750W 80+ Gold Certification

we also use an external hard drive of 2 TB to store raw data of each dataset, along with the pre-computed vectors that we add to a database corpus in the main solid state drive.

[2]https://lear.inrialpes.fr/people/wang/dense_trajectories

[3]https://gitlab.com/nbravo/pydensetrajectories-gpu

[4]https://opencv.org/

[5]https://docs.opencv.org/3.4/d9/d30/classcv_1_1cuda_1_1FarnebackOpticalFlow.html

Chapter 6

Experimental evaluation

The different models we present in this research focus on a specific task called Video-To-Text Matching & Ranking created by NIST for their TRECVID competition since 2016. The experiments are focused on validating and comparing the modifications proposed to the base models in terms of their results.

In what follows the experimental setup to train and evaluate the models is outlined. The sections of this chapter are: mean inverted rank description, base model implementation and validation; research model implementation, training and testing; and results comparison against the base model performance.

6.1 Evaluation metric: Mean Inverted Rank

Because its the official score metric of the TRECVID VTT Matching & Ranking challenge [56, 66, 67], we use the Mean Inverted Rank score to compare the performance between the multiple model variations of this research.

For a finite number of queries, this metric summarizes how well the model matches correct pairs, in our case video-sentence pairs, by averaging the inverse of the rank (position of each query). This score is the official evaluation metric in TRECVID for the Video-To-Text Matching & Ranking task, therefore the one we use to compare the performance of the reported base model with our implementation. Formally, for a set of queries $q_{\mathrm{i}} = (v_{\mathrm{i}}, S_{\mathrm{i}})$ where the model matches a video v_{i} with a set of sentences S_{i} with a match order criteria, the Mean Inverted Rank definition is:

$$MIR = \frac{1}{|q|} \sum_{\mathrm{i}=\mathrm{i}}^{|q|} \frac{1}{rank(q_{\mathrm{i}})} \tag{6.1}$$

where $rank(q_{\mathrm{i}})$ is the position $\in [1, |S_{\mathrm{i}}|]$ the corresponding sentence $s_{\mathrm{i}} \in S_{\mathrm{i}}$ of v_{i} is in the matching query q_{i}. Giving this definition, the extent of this metric is the domain $(0, 1]$, where, for example, a value of 0.5 means that the average query ranks the corresponding sentence in the second position.

6.2 Base model validation

Because the focus of this research is to prove the effectiveness of Dense Trajectories [4] and InferSent [3] descriptors in a state-of-the-art model for the Video-To-Text Matching & Ranking description task, the first and crucial step is to validate if our own implementation of the base model reproduces the reported results of the model; a task we successfully achieve as Table 6.2 shows.

The difficulty of this model is to find the correct combination of hyper-parameters that reproduces the reported results of Dong *et al.* [66], mainly because the lack of information the authors published of a fine tuning process they did with the TRECVID 2016 Video-to-Text dataset [77]. This model has a large number of hyper-parameters whose range and selected value we show in Table 6.1.

Our implementation in Python 3 of the base model successfully reproduces the reported results of Li *et al.* in their participation of TRECVID 2018 Matching and Ranking task [1], as presented in Table 6.2 where we compare the baseline report with several runs of our baseline implementation. We validate with multiple runs to discard the effects of randomness the training parameters initialization could generate. We define each hyper-parameters as follow:

- **Initial Learning Rate**: Initial value of the rate the learnable parameters update according to their gradient in the Stochastic Gradient Descent method.
- **Optimizer**: algorithm to update the learnable parameters of the model with each training batch.
- **Gradient clipping**: Maximum norm value of the learnable parameters vectors. It prevents the gradient to explode by limiting the value of them.
- **No improvement epochs to halve LR**: Number of consecutive iterations with no validation error improvement to decide halve the learning rate value.
- **No improvement epochs for early stop**: Number of consecutive iterations with no validation error improvement to decide stop the training process.
- **Video RNN hidden state dimension**: Dimension size of the hidden state vector of the bi-GRU in the second video encoding level.
- **Sentence RNN hidden state dimension**: Sentence side counterpart of "Video RNN hidden state dimension".
- **Video convolution channels**: Number of channels of each convolution in the third video encoding level.
- **Sentence convolution channels**: Sentence side counterpart of "Video convolution channels".
- **Common vector space dimension**: Dimension size of the common vector space between videos and sentence where a distance function is used to match and rank them.
- **Triplet Ranking Loss margin**: Minimum distance the loss function (Eq. 3.14) set between hard negative and hard positive vectors. (see Section 4.3.1)

Parameter	Range	Selection
Initial Learning Rate	0.0001 - 0.005	0.0001
Optimizer	Adam or MSRProp	Adam no weight decay
Gradient clipping	0.0 - 5.0	0.2
No improvement epochs to halve LR	3 - 5	3
No improvement epochs for early stop	10	10
Video RNN hidden state dimension	256 - 2,048	1024
Sentence RNN hidden state dimension	256 - 2,048	1024
Video convolution channels	256 - 2,048	512
Sentence convolution channels	256 - 2,048	512
Common vector space dimension	256 - 4,096	2048
Triplet Ranking Loss margin	0.2 - 0.3	0.2

Table 6.1: Hyper-parameter selection for Li *et al.* 2018 [1] model validation.

Set	A	B	C	D	E
Li *et al.* 2018	0.450	**0.448**	0.430	**0.436**	0.448
run 1	0.433	0.432	0.429	0.428	**0.449**
run 2	**0.451**	0.439	**0.433**	0.432	0.448
run 3	0.439	0.433	0.423	0.417	0.429

Table 6.2: Mean Inverted Rank comparison between the reported by Li *et al.* 2018 [1] in TRECVID 2018, and 3 runs of our implementation. Table 6.1 shows the hyper parameters selection to achieve this model validation. Each column corresponds to an official subset of data from the evaluation dataset of VTT matching & ranking TRECVID 2018. The best result by subset is highlighted in bold.

6.3 Training methodology

For the sake of a correct performance comparison, we use the same training process for every model under study. The method follows the training of the Dual Encoding for Zero-example Video Retrieval model of Dong *et al.* [11] on the TRECVID 2018 matching & Ranking task [1]; with a hyper-parameter tuning using the validation dataset TRECVID 2016 Matching & Ranking set A. The authors used this tuning process, as they described to TRECVID 2018 in [1]. However, they did not publish nor share by email their hyper-parameters selection, leading to repetition of this process for the baseline validation.

6.3.1 Training validation metrics

We use the precision for $RANK@1$, $RANK@5$ and $RANK@10$ to compare between the different model variations we propose. For a set of video-sentences matching queries Q and a given number N, the definition of $RANK@N$ is the number of queries in Q where the correct match is ranked in the first N positions of the query. With this, $RANK@1$ represents how many queries have a perfect match, $RANK@5$ represents the number of queries that rank the correct sentence of the video in the first 5 positions, and $RANK@10$ the first 10 positions. This metric gives an understanding on how well the model performs in a real use case.

6.4 Training results

To show a correct training process for each model of study, we present charts for the optimization of the Margin Ranking Loss (see Section 3.14) over the training dataset (MSR-VTT + MSVD + TGIF), and the $RANK@1$, $RANK@5$, $RANK@10$ and sum of ranks over the validation dataset (TRECVID 2016 subset A). We show results for the base model of Dong *et al.* 2019 [11], its variations INFERSENT-ONLY, INFERSENT-ALL, DT-ONLY, DT-ALL. For the models that depend on ResNeXt-101 embedding we show results of runs using the Shufflenet model of the baseline, and runs for the ResNeXt-101 pre-trained on Imagenet-1k from the MXNet framework.

6.4.1 Baseline Li *et al.* 2018 ResNeXt-101 comparison

For the training of our baseline, we compare in Figure 6.1 the 3 runs of our Dong *et al.* 2018 implementations with the Shufflenet ResNeXt-101, against the original Imagenet-1k model from MXNet. If we focus on one ResNeXt-101, we observe a similar optimization behavior in all 3 runs, which we quantify by calculating the Mean Absolute Error (MAE) between two runs. For two Shufflenet runs P, Q we get their MAE as:

$$MAE^{S}_{P,Q} = \frac{1}{min(|P|,|Q|)} \sum_{k=1}^{min(|P|,|Q|)} |P_k - Q_k| \tag{6.2}$$

with $M^{MX}_{P,Q}$ if they are MXnet runs.

With Eq. 6.2 we have $MAE^{S}_{1,2} = 311.14$, $MAE^{S}_{2,3} = 727.83$ and $MAE^{S}_{1,3} = 806.14$, difference values that are insignificant if we compare them to the order of magnitude of the loss function. This shows the consistency of the optimization for this model. Also, another sign these runs are consistent is the fact they early stop in similar iterations, where run 1 (brown) and 2 (red) stop at epoch 18, while run 3 (yellow) in epoch 20; which shows they converge to their maximum validation score almost at the same iteration.

For the MXnet runs we have $MAE^{MX}_{1,2} = 1,981.12$, $MAE^{MX}_{2,3} = 1,148.03$ and $MAE^{MX}_{1,3} = 2,927.13$, with run (1) stopping at epoch 22, (2) at 27 and (3) at 26. Metrics that show less consistency for these ResNeXt-101 embeddings. However, their difference calculated with MAE it is also insignificant in order of magnitude compared to the loss function values.

If we compare the optimization between different ResNeXt-101 embeddings, we see that the original ResNeXt-101 not only performs a better optimization process but also in fewer iterations than using the convolution network from MXNet. This behavior is present in the validation scores we show in Figure 6.2.

For the validation of each training iteration we calculate the 3 different ranks of Section 6.4, using the sum of them as the validation score for the early stopping indicator. We see in Figure 6.2 the improvement of rank for the 3 runs of the same ResNeXt-101 is identical in the first 2-3 epochs, getting bigger variations in late iterations. These differences are more significant in $RANK@1$ because it is the most strict of the ranks and gets lower precision values than the other metrics.

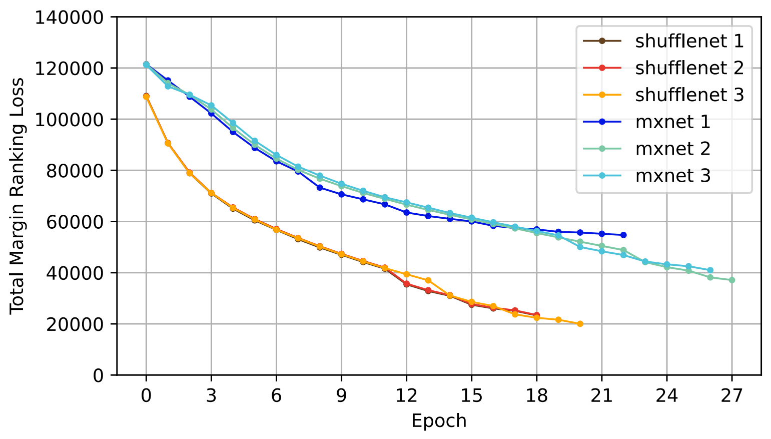


Figure 6.1: Training loss progress of our implementation of Dong *et al.* 2018. The 3 runs have a consistent minimization of the sum of Margin Ranking Loss over the complete training dataset. However, the ResNeXt trained in Shufflenet has a quicker and greater minimization.

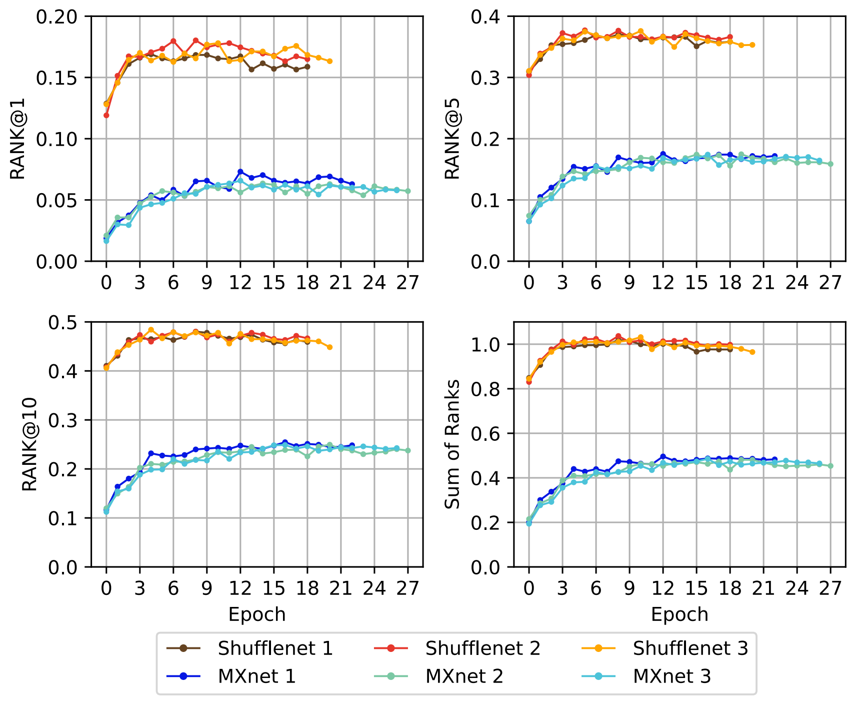


Figure 6.2: Validation loss progress of our implementation of Dong *et al.* 2018, with Shufflenet and MXnet embeddings.

When comparing between different visual vector embeddings, we find the same performance differences as the training loss reflects: Shufflenet embeddings get a greater validation score in fewer iterations than MXnet embeddings, for every all 4 metric validations.

6.4.2 Baseline Li *et al.* 2018 distance function comparison

We also train the baseline using the Euclidean distance for the common vector space instead of the original cosine distance. We test the Euclidean distance with a loss margin of 2.0 because it gives us the best training and testing results. Because the Euclidean and cosine functions have a large difference in the magnitude of their domains, the analysis does not compare their training loss evolution, only their validation scores in Figure 6.3.

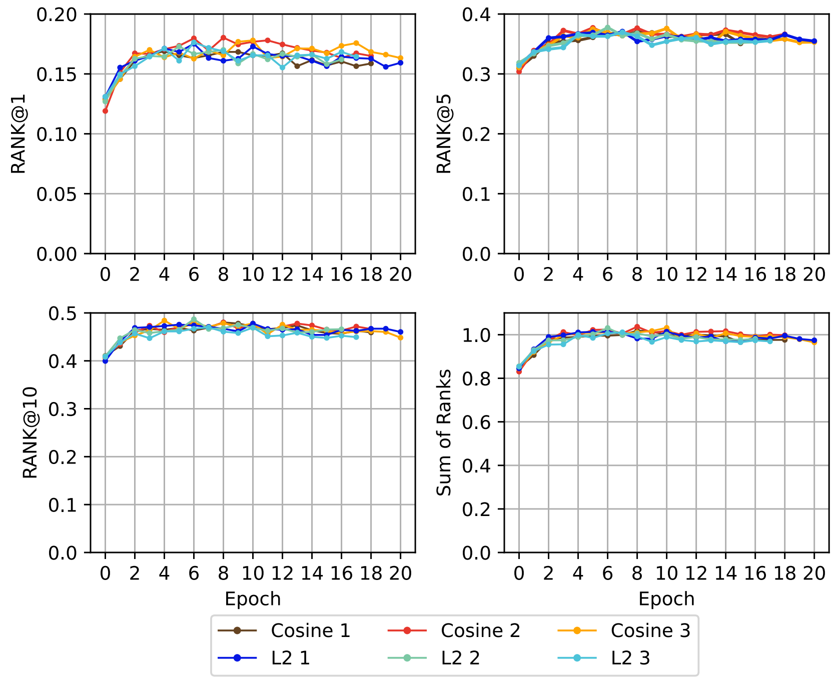


Figure 6.3: Validation loss progress of our implementation of Li *et al.* 2018, with Shufflenet embeddings, using cosine similarity and Euclidian distance as the distance function in the common vector space.

We see a similar behavior in rank with both distances, where the cosine distance has a maximum sum of ranks of 103.68, while the Euclidean distance has a maximum of 103.18. Because there is no gain using the Euclidean compared to the cosine distance, and because the margin for the loss function it is difficult to find in a non-closed numerical space, we do not use the Euclidean distance in the experimentation of the models of this study.

6.4.3 DT-ALL variation

This model has a similar behavior in loss optimization as the baseline for both ResNeXt-101 embeddings as shown in Figure 6.4, with a slightly lower performance where it takes more iterations to get a minimum total loss greater to the minimum of the baseline. For Shufflenet this model has a minimum loss of 29,790, against 20,017 of the baseline; and for MXnet gets a minimum Total Margin Ranking Loss of 45,941, against 37,114 of the base model.

To compare different runs of the same visual vector embedding, we get Mean Absolute Errors for Shufflenet of $MAE^{S}_{1,2} = 700.98$, $MAE^{S}_{2,3} = 1,075.87$ and $MAE^{S}_{1,3} = 449.34$; for MXnet runs we have $MAE^{MX}_{1,2} = 2,844.49$, $MAE^{MX}_{2,3} = 2,817.57$ and $MAE^{MX}_{1,3} = 1,933.30$. These results show the MXnet embeddings have less consistency in the loss minimization.

For the validation process, we also observe in Figure 6.5 a behavior similar to the baseline validation, but with a greater loss of performance with the MXnet embeddings, where for $RANK@1$ does not reach 5% of accuracy, and the iterations for early stopping are much

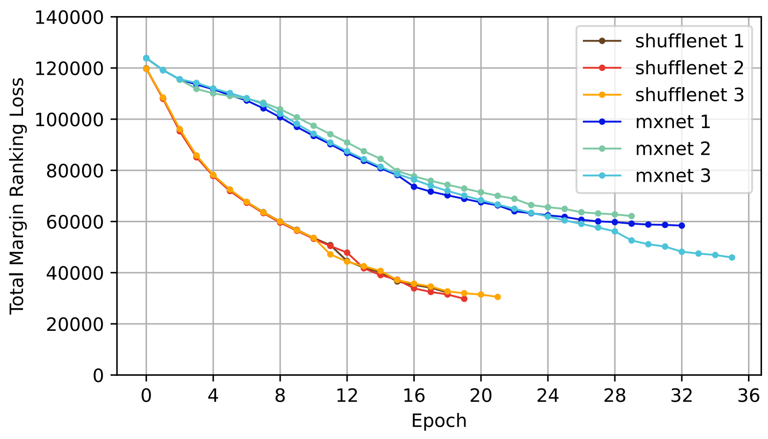


Figure 6.4: Training loss progress of the DT-ALL model, which shows similar behaviors as the baseline of Figure 6.1.

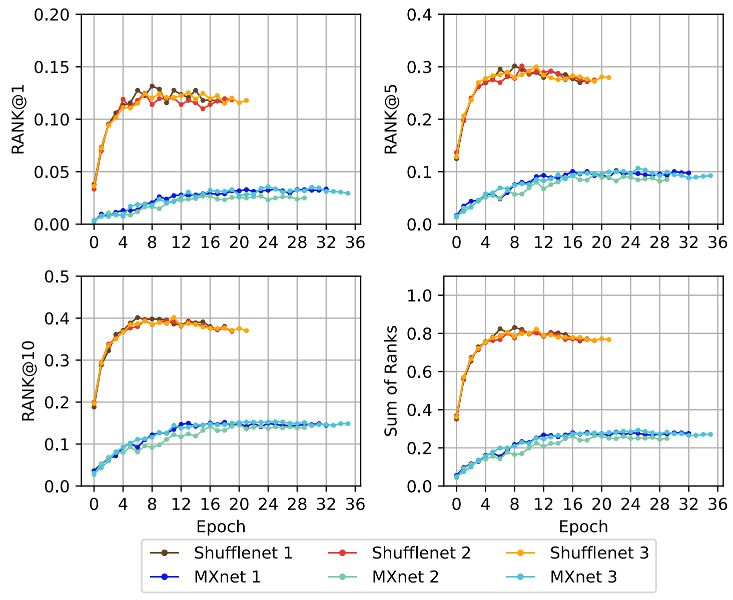


Figure 6.5: Validation loss progress of DT-ALL model, with Shufflenet and MXnet embeddings.

greater than its Shufflenet counterpart or even the baseline with MXnet ResNeXt-101 embeddings.

DT-ALL trained without TGIF dataset

During Section 5.1.4 we study the difference in framerate of the 3 datasets used to train the models under study, observing the majority of GIFs from the TGIF dataset have a framerate between 10 to 20 frames per second, much lower than the videos of MSVD and MSR-VTT, where their framerate is generally 30 frames per second or higher. This difference raises the question if the use of this dataset to train a model with Dense Trajectory descriptors have a negative impact in their performance, because, by definition, a Dense Trajectory is directly affected by the smoothness of the video, because a low framerate generates a small amount of Dense Trajectories and can create shifting trajectories between subsequent frames, i.e., the trajectory jumps from one point to another trying to follow the irregular transition between two frames of its group of pixels. To test this on the DT-ALL model, 3 training runs of this model not using TGIF in the training dataset were done. The DT-ALL model of study is the version with the Shufflenet ResNeXt-101 architecture because its better results showed in Section 6.4.3. Figure 6.6 compares the loss function evolution during training and Figure 6.7 compares the training validation scores of the DT-ALL model with and without the TGIF dataset. The results of removing the TGIF dataset have a similar negative impact on the training of the model that using the MXnet version of a ResNeXt-101, where the sum of ranks of the validation cannot surpass the 0.4 score mark, a performance worst than its counterpart trained with the TGIF dataset. The testing results of the DT-ALL model without TGIF are shown in Table 6.3, with further discussion and conclusion in Section 6.6.3.

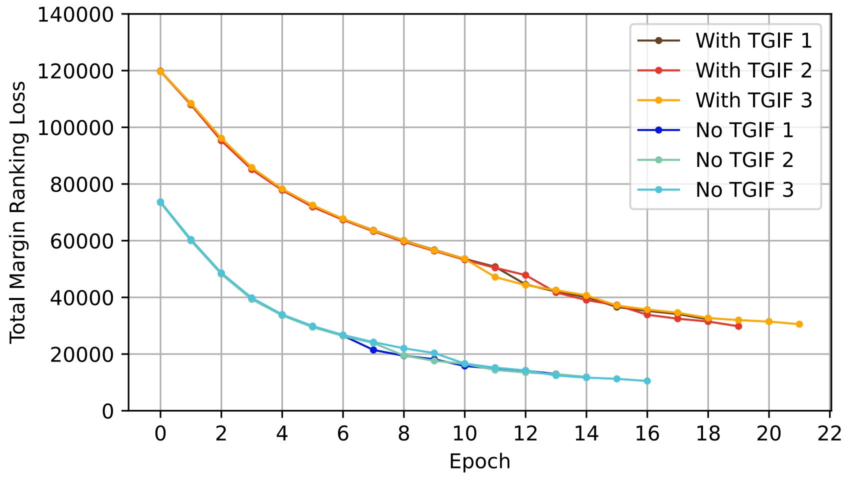


Figure 6.6: Loss function evolution during the training of the DT-ALL model with and without the TGIF in its training dataset. DT-ALL model of study is the version with the Shufflenet ResNeXt-101 architecture. Notice the reason for lower loss function values in the non-TGIF version is due to a smaller training dataset, not a better optimization process.

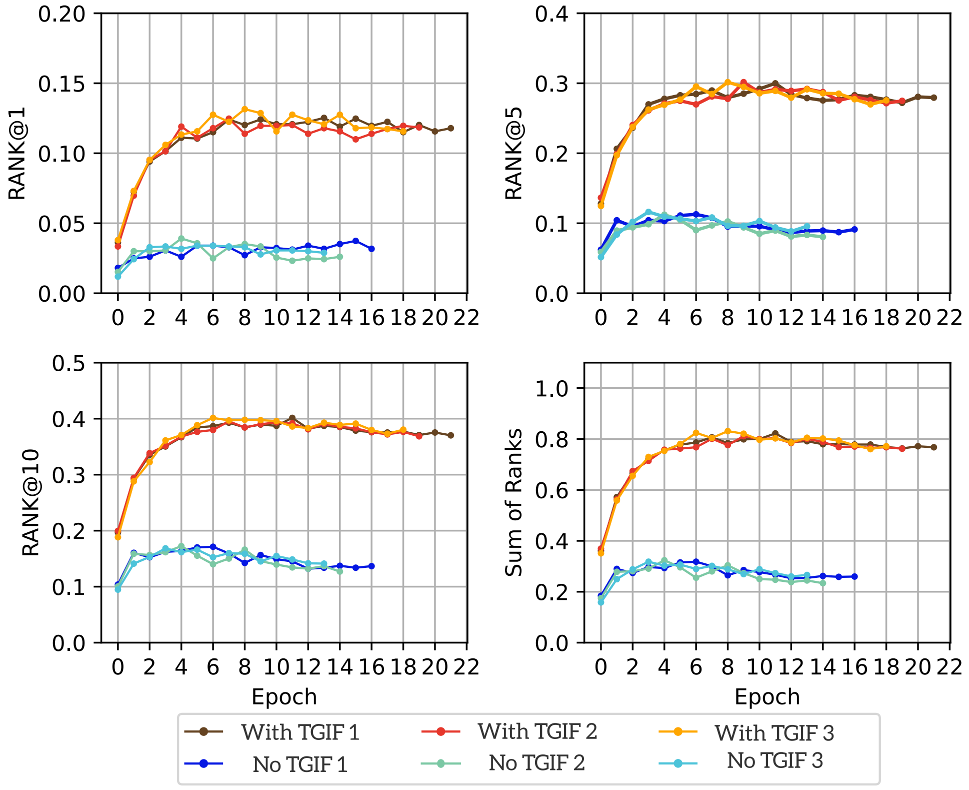


Figure 6.7: Validation scores for the DT-ALL model trained with and without the TGIF dataset.

6.4.4 DT-ONLY variation

This model does not use ResNeXt-101 embeddings for video vectorization, just the DT-BoW we describe in Section 4.1.1. Its loss optimization in Figure 6.8 struggles to reach the 100,000 value, getting similar values for the 3 runs. Their cross difference is $MAE_{1,2} = 605.63$, $MAE_{2,3} = 835.87$ and $MAE_{1,3} = 1,271.94$; which shows the consistency of the model configuration.

The validation process in Figure 6.9 shows a poor performance of this model in every metric when using the same scale as the previous models. The maximum values for each metrics in this model are $max(RANK@1) = 0.51\%$, $max(RANK@5) = 1.36\%$, $max(RANK@10) = 2,38\%$ and $max(\text{sums of ranks}) = 4.82\%$.

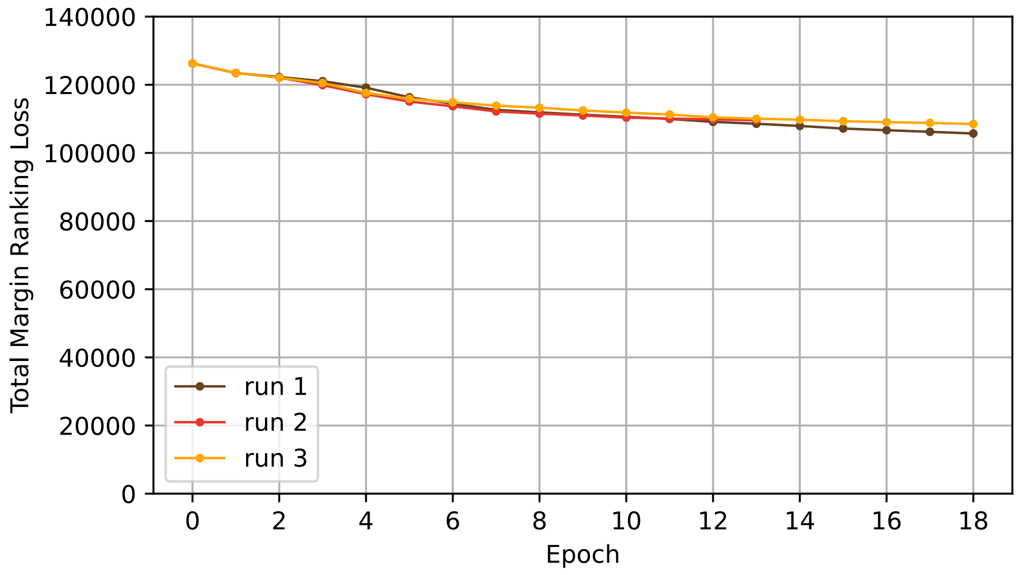


Figure 6.8: Training loss progress of the DT-ONLY model. This model does not use ResNeXt-101 embeddings as input, therefore no comparison between both networks.

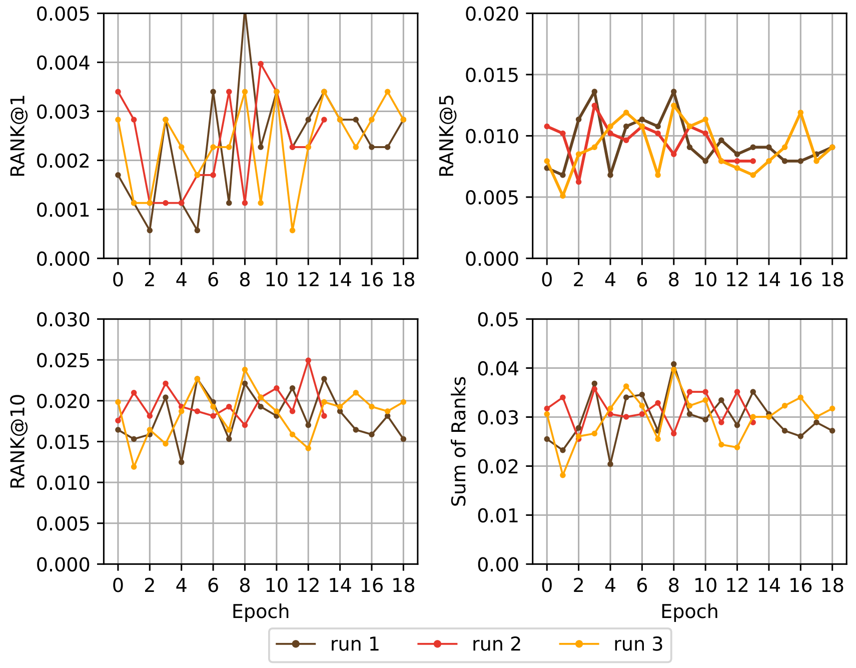


Figure 6.9: Validation loss progress of the DT-ONLY model. Notice, because its low scores, it does not share the same scales as other validation charts.

DT-ONLY trained without TGIF dataset

In the second part of Section 6.4.3, the results of DT-ALL trained without the TGIF collection showed a worse training process for the model, not reaching even half of the validation score of the same model trained with TGIF. In this section the same comparison is done for the DT-ONLY model with and without the TGIF dataset, where Figure 6.10 compares the loss function evolution and Figure 6.9 shows the validation scores comparison between the two training dataset scenarios of DT-ONLY. When removing the TGIF dataset, the negative impact in the loss function evolution of DT-ONLY is lower than in DT-ALL (see Figure 6.6), in fact, a higher loss function optimization can be appreciated since epoch 2 in all 3 runs, in comparison to the non-TGIF version of DT-ALL that seems to have more trouble minimizing the loss function. However, this better optimization process is not reflected in the validation scores of each epoch, where the same range of values is achieved no matter the run or the lack of the TGIF dataset. The testing metrics of the non-TGIF version of DT-ONLY are shown in Table 6.3, with further discussion in Section 6.6.3.

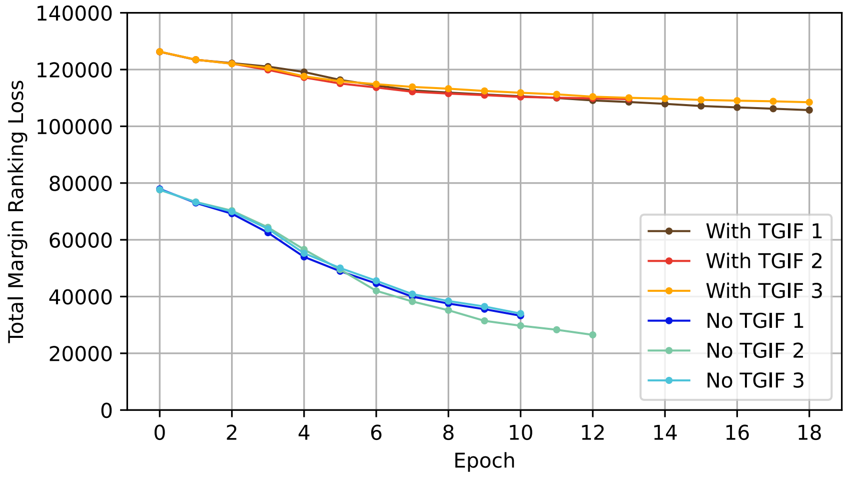


Figure 6.10: Loss function evolution during the training of the DT-ONLY model with and without the TGIF in its training dataset.

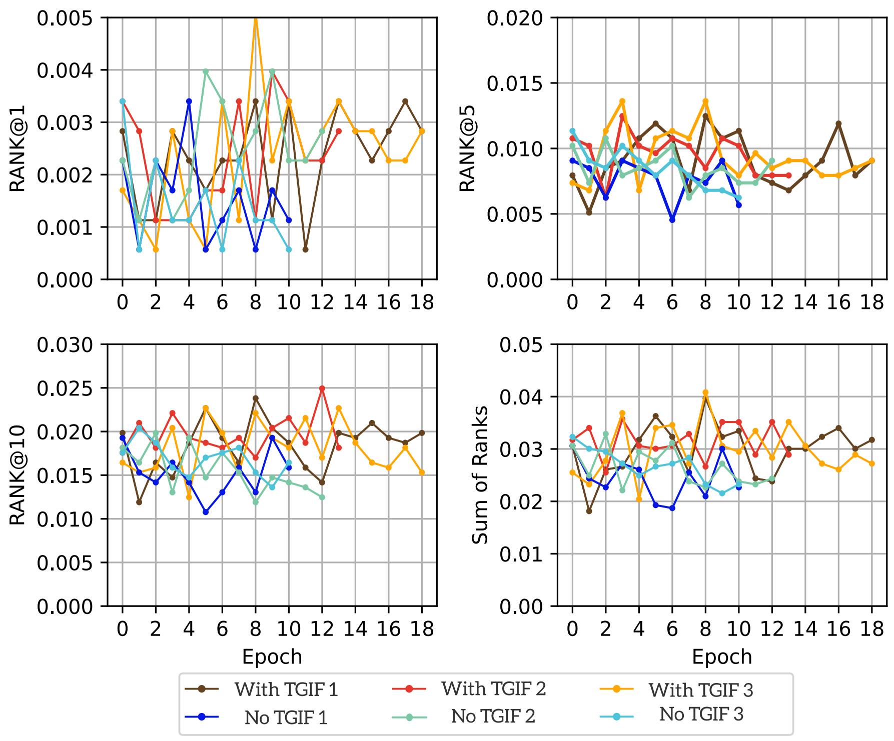


Figure 6.11: Validation loss progress of the DT-ONLY model. Notice, because its low scores, it does not share the same scales as other validation charts.

6.4.5 INFERSENT-ALL variation

The training of the INFERSENT-ALL model share similar behavior with the baseline, but with a slightly lower performance as we see in Figure 6.12, with a minimum loss for Shufflenet runs 5,514 units greater than the minimum of Shufflenet runs of the baseline, but only 664 units greater for the MXnet runs.

For the training loss, the behavior between Shufflenet runs is also consistent, with $MAE^{S}_{1,2} = 1,305.22$, $MAE^{S}_{2,3} = 554.16$ and $MAE^{S}_{1,3} = 856.27$. For MXnet runs we have $MAE^{MX}_{1,2} = 581.75$, $MAE^{MX}_{2,3} = 746.09$ and $MAE^{MX}_{1,3} = 356.18$. These metrics show the consistency between runs of the same ResNeXt-101 embedding.

The validation process of Figure 6.13 shows the Shufflenet and MXnet runs reach a precision very similar to their baseline counterpart, which reflects the testing results in the Section 6.5. Comparing with the DT-ALL model, which adds the DT-BoW embedding to the video vectorization, INFERSENT-ALL outperforms it in every metric, getting a sum of Ranks over 0.4 with MXnet embeddings, in contrast to the slightly over the 0.2 mark for the DT-ALL model.

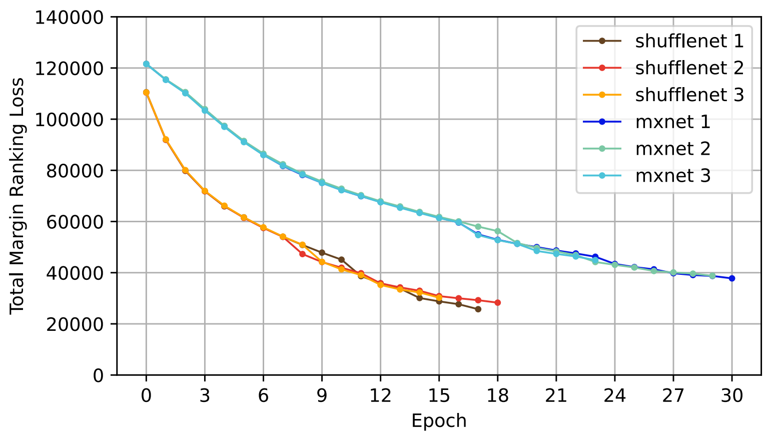


Figure 6.12: Training loss progress of our implementation of the INFERSENT-ALL model, which shows similar behaviors as the baseline of Figure 6.1.

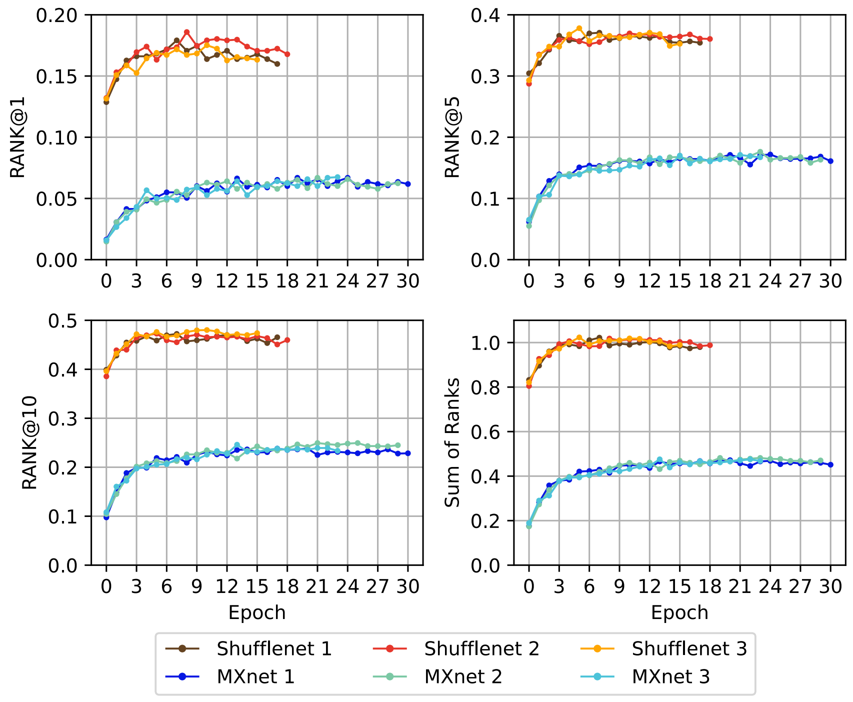


Figure 6.13: Validation loss progress of the our implementation of the INFERSENT-ALL model, with Shufflenet and MXnet embeddings.

6.4.6 INFERSENT-ONLY variation

These models shares many characteristics with the DT-ONLY variation, where, as shown in Figure 6.14, no optimization of the loss function is done, and the model does not get over 5% of accuracy in any rank metrics of sum of ranks, as shown in Figure 6.15.

For both ResNeXt-101 embeddings we see similar loss evolution results in these models, with errors of $MAE^{S}_{1,2} = 978.96$, $MAE^{S}_{2,3} = 1,049.28$ and $MAE^{S}_{1,3} = 771.07$. And for MXnet runs we have $MAE^{MX}_{1,2} = 2,708.36$, $MAE^{MX}_{2,3} = 1,560.42$ and $MAE^{MX}_{1,3} = 1,372.15$.

The validation process leads to low precision metrics as shown in Figure 6.15 where both

Shufflenet and MXnet versions do not follow a visible pattern of behavior. Their overall maximums are $max(RANK@1) = 0.17\%$, $max(RANK@5) = 0.68\%$, $max(RANK@10) = 1.07\%$ and $max(Sum of ranks) = 1,92\%$.

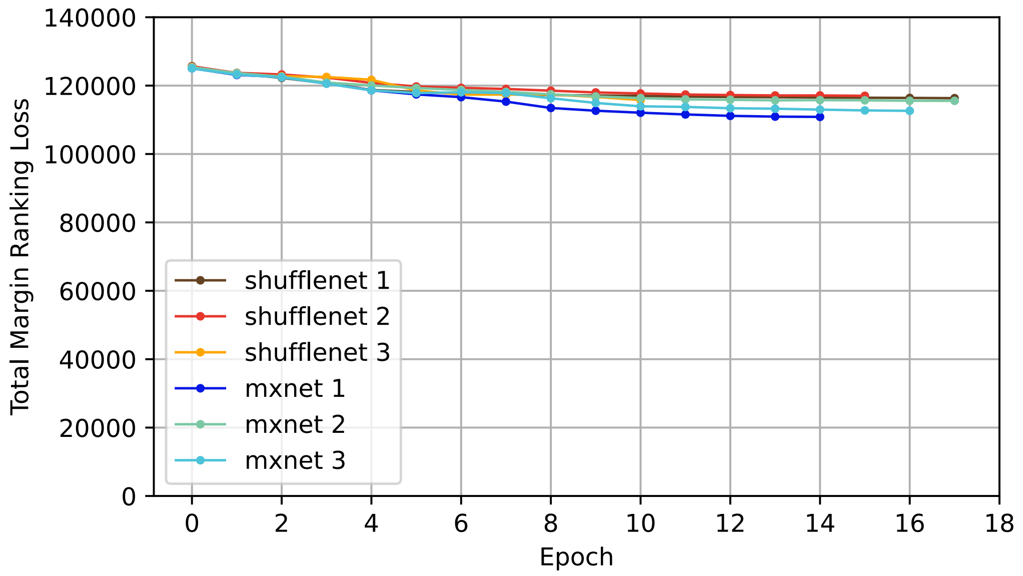


Figure 6.14: Training loss progress of our implementation of the INFERSENT-ONLY model, which shows similar behaviors as the baseline of Figure 6.1.

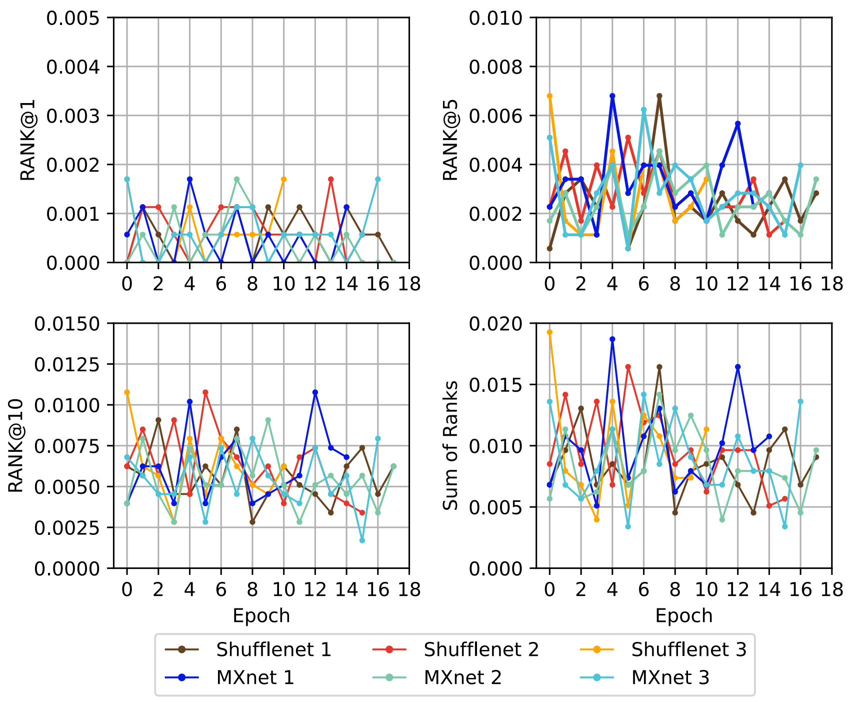


Figure 6.15: Validation loss progress of our implementation of the INFERSENT-ONLY model, with Shufflenet and MXnet embeddings. Notice, because its low scores, it does not share the same scales as other validation charts.

6.5 Testing results

After training a model where we use stochastic gradient descent to optimize the sum of the Margin Ranking Loss (Eq. 3.14) over the training dataset, we test each model by measuring the Mean Inverted Rank (MIR Eq. 3.14) of their Video-to-Text matching over the TRECVID 2018 Matching and Ranking dataset (see Section 5.2.1). This dataset comprises 1,897 videos and 5 sets of 1,915 video-sentence pairs: A, B, C, D and E, where we get the MIR over each set independently.

Model	A	B	C	D	E
Li *et al.* 2018 (SH)	0.441 ± 0.007	**0.434** ± 0.003	**0.428** ± 0.004	0.425 ± 0.006	**0.442** ± 0.009
Li *et al.* 2018 (MX)	0.176 ± 0.009	0.181 ± 0.007	0.179 ± 0.006	0.168 ± 0.003	0.171 ± 0.005
INFERSENT-ALL (SH)	**0.442** ± 0.001	0.432 ± 0.008	0.427 ± 0.008	**0.431** ± 0.007	0.439 ± 0.003
INFERSENT-ALL (MX)	0.169 ± 0.002	0.169 ± 0.006	0.172 ± 0.003	0.165 ± 0.004	0.167 ± 0.002
INFERSENT-ONLY (SH)	0.004 ± 0.001	0.005 ± 0.001	0.005 ± 0.001	0.004 ± 0.001	0.004 ± 0.002
INFERSENT-ONLY (MX)	0.004 ± 0.001	0.004 ± 0.001	0.004 ± 0.001	0.004 ± 0.001	0.004 ± 0.001
DT-ALL (SH)	0.290 ± 0.002	0.280 ± 0.008	0.286 ± 0.007	0.278 ± 0.007	0.293 ± 0.009
DT-ALL (SH no TGIF)	0.093 ± 0.002	0.089 ± 0.005	0.093 ± 0.005	0.088 ± 0.002	0.092 ± 0.001
DT-ALL (MX)	0.077 ± 0.008	0.078 ± 0.007	0.080 ± 0.006	0.075 ± 0.003	0.078 ± 0.003
DT-ONLY	0.012 ± 0.001	0.012 ± 0.001	0.012 ± 0.002	0.011 ± 0.001	0.010 ± 0.002
DT-ONLY (no TGIF)	0.008 ± 0.001	0.008 ± 0.001	0.008 ± 0.001	0.008 ± 0.001	0.009 ± 0.001

Table 6.3: Mean Inverted Rank results for every model of this study. Models with (SH) use the Shufflenet embedding, while (MX) use the MXnet version. Models with no TGIF in their training dataset included (no TGIF). The values correspond to the average model MIR between its 3 experimental runs, with their standard deviation error. Each column correspond to an official subset of data from the evaluation dataset of VTT matching & ranking TRECVID 2018. The best result by subset is highlighted in bold.

Table 6.3 summarizes all the experimentation of the models in study, where we highlight the best performed models overall for each testing set. We notice that Dong *et al.* 2018 and INFERSENT-ALL with Shufflenet embeddings models perform best for this task in this dataset. In contrast, INFERSENT-ONLY is the worst performed model with no apparent advantage by using Shufflenet or MXnet embeddings; even DT-ONLY outperforms this model configuration. If we compare the same models with different ResNeXt-101 embeddings, we see the use of Shufflenet embeddings outperforms MXnet embedding in every model, in an average factor of 2.48 for Dong *et al.* 2018, 2.57 for INFERSENT-ALL, 1.1 for INFERSENT-ALL, and 3.67 for DT-ALL.

To do a visual comparison between ResNeXt-101 embeddings of the same model, we present 4 bar charts of equal scale along with the error bar to represent the standard deviation over the 3 runs of the same configuration. If we compare the baseline of Figure 6.16 with the INFERSENT-ALL model of Figure 6.17, we see a light difference in performance. Using the same Mean Inverted Rank scale on every model in Figures 6.16, 6.17, 6.18, 6.19 and 6.20, we quickly determine poorly performing models INFERSENT-ONLY and DT-ONLY because its corresponding bars collapse.

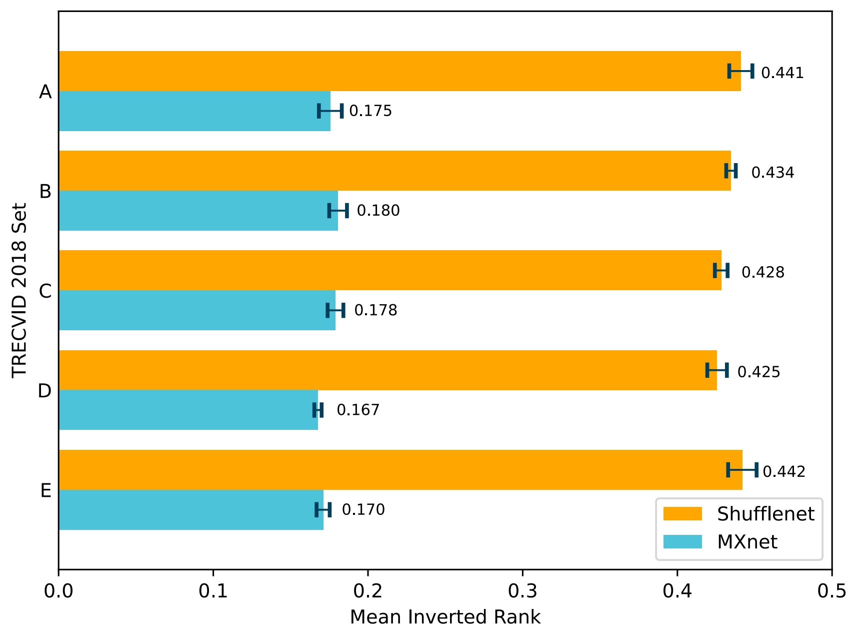


Figure 6.16: Mean Inverted Rank on TRECVID 2018 Matching & Ranking task of our Dong *et al.* 2018 implementation with ResNeXt-101 embeddings of Shufflenet and MxNet.

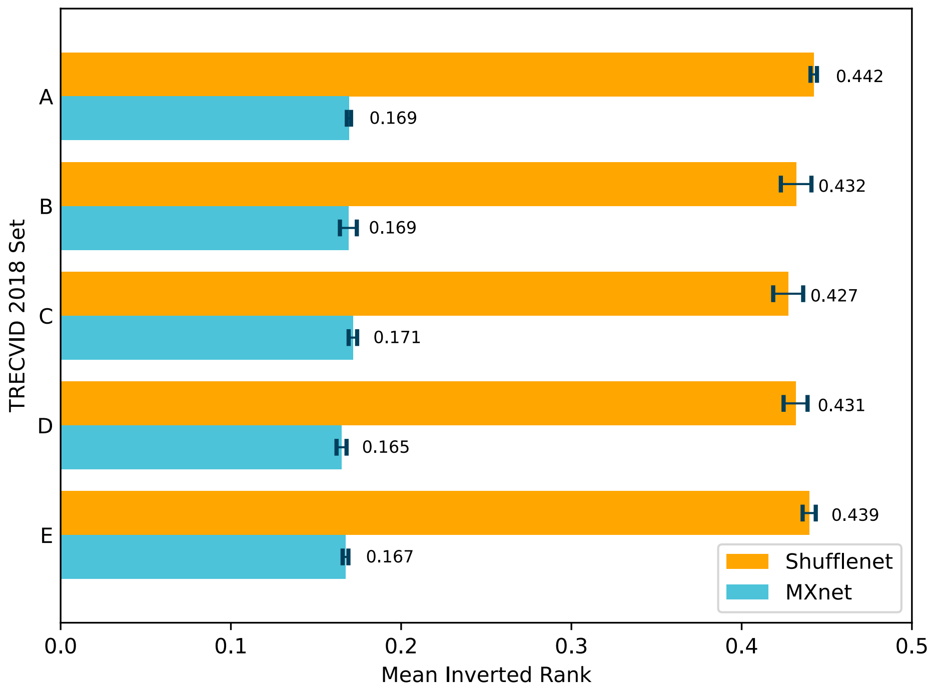


Figure 6.17: Mean Inverted Rank on TRECVID 2018 Matching & Ranking task of INFERSENT-ALL model with ResNeXt-101 embeddings of Shufflenet and MxNet.

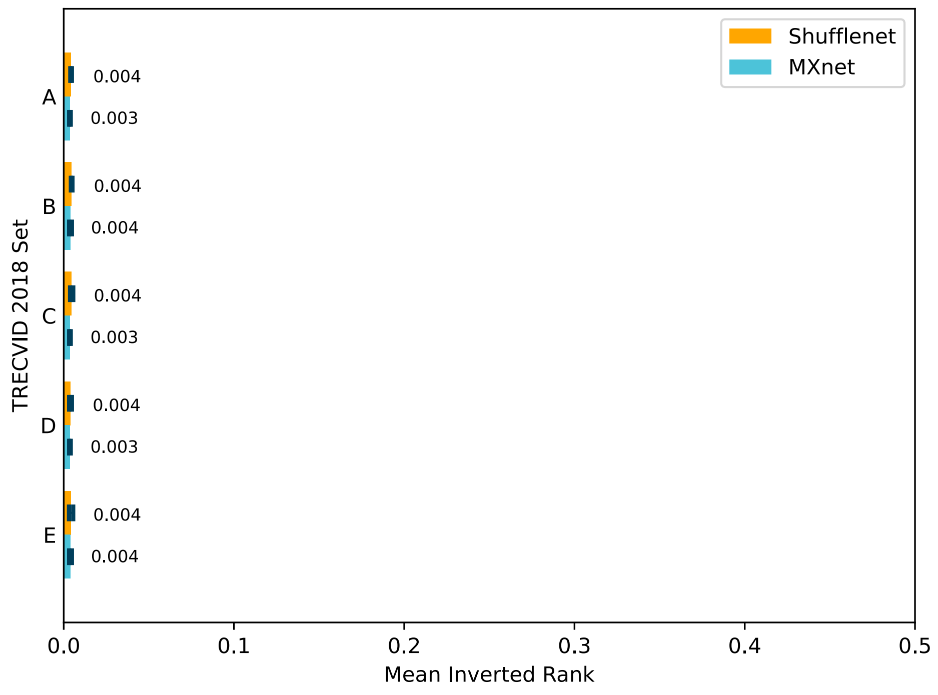


Figure 6.18: Mean Inverted Rank on TRECVID 2018 Matching & Ranking task of our INFERSENT-ONLY implementation using ResNeXt-101 embeddings from Shufflenet and MxNet.

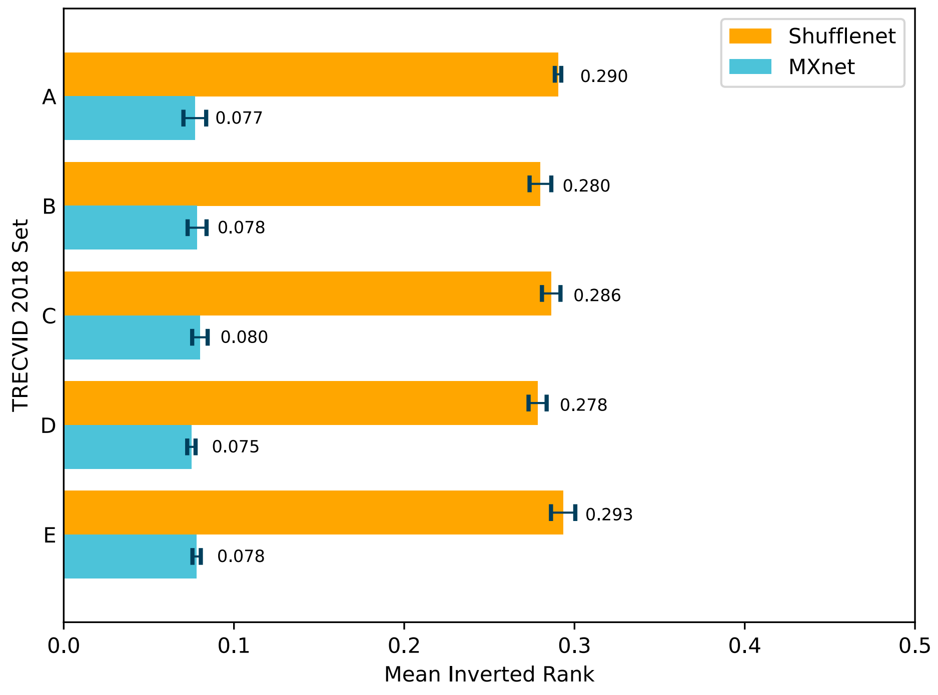


Figure 6.19: Mean Inverted Rank on TRECVID 2018 Matching & Ranking task of our DT-ALL implementation using ResNeXt-101 embeddings from Shufflenet and MxNet.

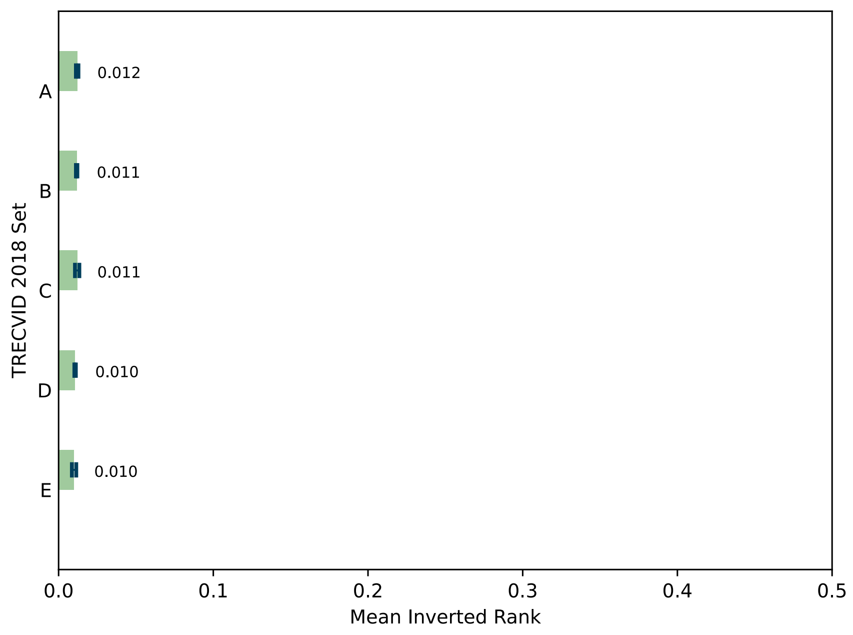
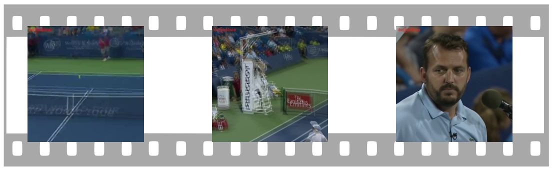


Figure 6.20: Mean Inverted Rank on TRECVID 2018 Matching & Ranking task of our DT-ONLY implementation. This model does not use a ResNeXt-101 embedding.

6.5.1 Video-sentence matching examples

To have a better understanding of each model performance during their expected use case scenario, we present several video-sentence matching examples using the TRECVID 2018 evaluation dataset each model was tested on. An example focus on a single video, showing the top matching sentence for each set (from set A to set E) in every model under study. We decided to show only examples using the Shufflenet versions of each model because the inclusion of the MXnet variations will create too much information that we think is not necessary considering their performance is consistently lower in every configuration. We use only the first run of each model because if we try to summarize the 3 runs into a single result, it is a problem to decide which of the three 1st matching sentences must be shown on the list. Relevant sentences are highlighted in green, while the irrelevant ones in red. The selection of video examples is based on topic variety and interesting ranking behavior with the model variations of the baseline.

Well baseline match video example: man kicking a tennis ball towards the referee.



Li etal. 2018 (baseline)

A At a tennis tournament, a player in white shirt and blue shorts hits the ball into the stands and an announcer grimaces.
B At a tennis match one of the players kicks the tennis ball into the stadium and the announcer looks astonished and shrugs his shoulders.
C A tennis player dislikes a referee call and kicks the ball at the ref, barely missing him.
D At an outdoor tennis match, a player kicks the ball out of bounds into the stands past a man in blue shirt on a high seat in front of a microphone and then the blue-shirted man makes a face.
E A tennis player on the court kicks a tennis ball and it goes into the stands where people sit.

INFERSENT-ALL

A At a tennis tournament, a player in white shirt and blue shorts hits the ball into the stands and an announcer grimaces.
B At a tennis match one of the players kicks the tennis ball into the stadium and the announcer looks astonished and shrugs his shoulders.
C A tennis player dislikes a referee call and kicks the ball at the ref, barely missing him.
D Two male tennis players in blue shirts and white shorts hug across the net.
E A tennis player on the court kicks a tennis ball and it goes into the stands where people sit.

INFERSENT-ONLY

A A Japanese women wearing a red bow tie white shirt and black vest, with short black hair is introducing herself.
B A white dog wearing a vest and sitting in a pink toy car that has writing on the windshield and towing a red and green toy wheel basket holding a smaller dog wearing a scarf.
C An outdoor basketball pick-up game with one young boy being passed a ball and doing a trick shot and scoring.
D A Japanese boy, whose eyes are being shielded by a young girl, is surprised to see who it is behind him.
E A man in a sleeveless white shirt and red earphones around his neck yawns, gets up from bed, and joins a roomful of dancing, arm-pumping people.

DT-ALL

A Michigan State player 18 walks up to mike in field of stadium to accept a trophy.
B In a crowded baseball stadium during the day, a bearded blue-clad Bluejays player is pushed by a red capped opposing player in red and when the bluejay starts to pull back to punch his opponent "Odor #12", the redshirt hits him in the jaw first, knocking him over.
C A woman TV announcer in a crowd of sports fans is pushed and nearly groped by a man and she pushes him away.
D At a baseball stadium sidelines a girl in a uniform reaches up and catches the ball in her hand instead of the glove on her other hand.
E At sport arena, suited man walks toward spectator ranks, removes his jacket, and to cheers, tosses it into the crowd.

DT-ONLY

A A young man in red stocking cap, red shirt and tights walks down a runway carrying a shovel in front of a yelling crowd, in hazy light indoors, he scatters the contents of the shovel into the audience and turns to run off-stage.
B Two people ride tricycles down a runway, with crowds on either side and a big TV screen.
C On an urban street on a cloudy day, a red-shirted man flips a billboard for Quizno's standing on a traffic island at an intersection.
D On a runway, a model comes out in a black outfit, opens it in the front and it turns it inside out and it becomes a red one.
E On a stage runway, two people sitting on tricycles walk themselves along.

Figure 6.21: Top 1 sentence match of video 779 for each model under study and evaluation set of TRECVID VTT 2018.

In Figure 6.21 we can observe the baseline model ranks perfectly the sentences of the video. For its variants we can observe:

- The INFERSENT-ALL results shows a similar ranking than the baseline, reflecting the expected behavior of this model considering there is no statistical difference with the base model, as we can see in the results of Table 6.3. Although, we can see this model did not do a correct match in set D, where it ranked a sentence that shares the same topic (tennis) but with a different description of events. Considering the corresponding sentence in this set was ranked at 2nd place, we can suggest this model has problems differentiating events that happen in same contexts or locations, but it still ranks the corresponding sentence and similar ones in high positions.
- The DT-ALL model does not match correctly any corresponding sentence, but the sentences it suggests share sport topics like baseball, sports, crowds and stadium. This behavior could mean this model learned to match videos with scenes related to sport

(like this video example where a tennis ball and field are portrayed) with words related to sports, but did not learn to differentiate between them. This model ranked the corresponding sentences of each set in positions 32th, 89th, 23th, 25th and 39th, respectively.

- The results of INFERSENT-ONLY and DT-ONLY show they did not learn correct patterns, matching the video with different topics. INFERSENT-ONLY sentences focus primarily in clothes, where the majority of them use the word *shirt*, except for set C. This could suggest this model did not focus on the tennis related objects, but in the shirt the referee is wearing when the camera focus directly on him. The DT-ONLY results do not seem to follow a pattern, where only the word *red* is shared between sets A, C and D but with different topics, however the video has no relevant sections with the red color to suggest a link to this behavior, therefore we conclude the sentence rankings of DT-ONLY follow no pattern.

Confuse video example: model walking on a runaway

Li etal. 2018 (baseline)

A On a runway, a model unbuckles black outfit and peels it back to become a red outfit.
B A model is walking the floor when she appears wearing a long cape jacket, but reveals it to open and reveal a cleavage cut long red dress that is attached, with the onlookers expressing amazement and liking what they see.
C A magician, wearing red outfit, is waving a large white cloth and then shows off four standing persons, during a show.
D At a party where men are in suits, a busty black woman in a tight red dress dances.
E At a fashion show a model comes out wearing a black outfit that transforms into a red outfit.

INFERSENT-ALL

A On a runway, a model unbuckles black outfit and peels it back to become a red outfit.
B A model is walking the floor when she appears wearing a long cape jacket, but reveals it to open and reveal a cleavage cut long red dress that is attached, with the onlookers expressing amazement and liking what they see.
C A magician, wearing red outfit, is waving a large white cloth and then shows off four standing persons, during a show.
D On a runway, a model comes out in a black outfit, opens it in the front and it turns it inside out and it becomes a red one.
E At a fashion show a model comes out wearing a black outfit that transforms into a red outfit.

INFERSENT-ONLY

A A group of children are stuffing bags with items to be given to a charity.
B An ape, a gorilla, is knocking on a door and then jumps on it , in the zoo.
C A racing car slides across grass and track, rolls and breaks apart.
D At a pool, a young boy runs to the edge, stops then slides down into the water.
E A person is holding and moving an own on their hand from side to side.

DT-ALL

A On a runway, a model unbuckles black outfit and peels it back to become a red outfit.
B A model is walking the floor when she appears wearing a long cape jacket, but reveals it to open and reveal a cleavage cut long red dress that is attached, with the onlookers expressing amazement and liking what they see.
C At a fashion show, a woman in a black caftan walks before some men, drops her belt and opens the garment to become a red caftan.
D A singer, Lady Gaga, walks, wearing a black dress in a celebration theater.
E At a fashion show a model comes out wearing a black outfit that transforms into a red outfit.

DT-ONLY

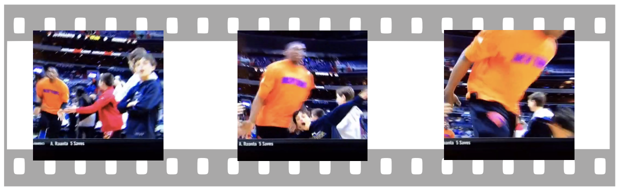
A On a runway, a model unbuckles black outfit and peels it back to become a red outfit.
B Two young females in a parking lot during the day walking and running briskly up a ramp.
C At a fashion show, a woman in a black caftan walks before some men, drops her belt and opens the garment to become a red caftan.
D On a runway, a model comes out in a black outfit, opens it in the front and it turns it inside out and it becomes a red one.
E At a fashion show a model comes out wearing a black outfit that transforms into a red outfit.

Figure 6.22: Top 1 sentence match of video 817 for each model under study and evaluation set of TRECVID VTT 2018.

Figure 6.22 shows interesting rankings for a video where a model walks on a runaway and amazes the judges:

- The model of Li *et al.* 2018 ranks in set C and D descriptions that are similar to the video content, but not quite correct. The corresponding sentences for this video were ranked in position 2nd and 3rd in their respective sets, showing a clear confusion of the model in differentiating between a runaway and a show on a stage. This confusion can be because both incorrect sentences not only talk about a person showing off, but also they focus on the clothes they are using, creating a subtle difference versus the groundtruth.
- The INFERSENT-ALL model does a better job than the baseline in this video, being able to match a relevant sentence for set D, but having the same trouble in set C where it ranked the corresponding sentence in 2nd place. As stated before, this reflects its performance similarity to the base model.
- On the other hand, in the DT-ALL and DT-ONLY models the use of Dense Trajectories is able to match corresponding sentences with a success similar to the baseline and INFERSENT-ALL; with DT-ALL confusing in set D the video with another show in a stage. But in set C both models confuse the video with another fashion show with similar clothes. Focusing in set B, the reason of the incorrect match of DT-ONLY could be the high similarities the video of this matched sentence (video 621) has with the target one, where the first video portraits young girls posing to the camera while changing clothes, a scene alike in terms of behavior to a model walking on a runaway. The good performance of this example could also be due to the clean model's background, where her silhouette is portrayed against a blank wall, which could have facilitated the movement encoding with Dense Trajectories.
- The bad matching results of INFERSENT-ONLY reflect the poor testing results it obtained in Table 6.3, not ranking a single sentence close to the content of the target video, nor following a pattern between these sentences. In fact, this model variation ranks the correct sentence for each set in positions 1,730th, 615th, 364th, 1,661th and 44th, respectively.

Bad example overall: adult man hitting a kid by accident.



Li etal. 2018 (baseline)

A In stands, men cheer and a boy lifts up his shirt and rotates his pelvis.
B Men and a little boy are sitting in stadium chairs cheering.
C A boy wearing some sort team clothes, is moving in his place with happy gestures , at a game.
D Two young boys show their enthusiasm, on in talk, one in movement.
E In a stadium several fans are cheering and one young male fan wearing a green shirt and pants is dancing, with joy.

INFERSENT-ALL

A On a game show, players slap hands with each other.
B Two boys in gray t-shirts on an orange stage with microphones bump fists and hug.
C Inside a studio, a group of contestants are giving each other five on a game show.
D Two young males dressed in sport jerseys on stage holding microphones.
E A Television game show with the players congratulating themselves.

INFERSENT-ONLY

A A small chid, in a car seat in a stationery car, singing aloud to music.
B A group of asian singers are onstage, during a concert, and performing.
C A girl with a mask on her eyes, is talking to a boy with a mask on his eyes.
D An announcer is about to speak, but sticks his tongue out repeatedly.
E At night, a little boy is twirling around on his head break dancing.

DT-ALL

A A child, holding a cotton candy holder, is standing, cheering and making facial gestures as the crowd cheers at a sporting event.
B Gold medal winners of a soccer event are walking away with the gold trophy.
C A group of young men are gathered around a cake celebrating the millionth person who voted v for victory.
D Two young asian girls, in a sports arena, hug for the camera.
E The leader of a winning team in an indoor sports arena holds up and kisses a trophy as cheering fans shout and wave.

DT-ONLY

A A wrestler a professional wrestling event at an indoor arena at night throws another wrestler out of the ring.
B Two wrestlers are shown on a big screen, at a game.
C Video of professional wrestlers as they walk toward the ring at indoor arena at night.
D In an auditorium, a wrestling match shows a long-haired woman in red slapping a burly wrestler and then he picks her up and slams her down on her back in front of a screaming crowd.
E A pro wrestler is in the ring with a pro-woman wrestler, in a live performance, when she slaps him in the face and he responds by body-slamming her to the mat.

Figure 6.23: Top 1 sentence match of video 855 for each model under study and evaluation set of TRECVID VTT 2018.

In the example of Figure 6.23, where a man running towards the camera hits a kid accidentally, all models under study fail to match a relevant sentence for this video, but there are details to show from their results:

- The sentences matched by the base model and INFERSENT-ALL seem to share the topic of a game show, with actions performed by adult players or boys. This shows the models are able to identify the subjects and context of the target video, being the basketball player and the boy the subjects and the basketball stadium the context, but fail to identify the correct events. This could happen because the unusual actions happening in this video, where in a stadium is expected to happen sport related events but instead an adult is accidentally hitting a kid. This reason has a greater importance if we consider we were unable to find any training example that shares a similar situation.
- The match of DT-ALL seems to only identify the context of the video, but fails to identify the objects that take place in it. It ranks sentences that describe sport or competition events, but focusing in crowd events than actual players. This could be

because the large quantity of people portrayed in the background of the video, taking the attention of the model from the main event of the target video.

- DT-ONLY ranking has an interesting behavior. It identifies the video context is about a sport event, but its sentences share the topic of wrestling, a context that shares similar actions, e.g. people hitting or crashing each other, with the target video. With this example it can be stated that the DT-ONLY variant has the same problem with this video as the model walking in a runaway of Figure 6.22, where it is able to identify an action that is portrayed in the video: a person hitting another person, and by combining it with the fact this is happening in a stadium or sport event, the model resolves to identify the target video as a wrestling event.
- The results of INFERSENT-ONLY show some of the sentences share common subjects with the target video like children in sets A and E, but is not sufficient to say the model follows a visible pattern, reflecting again its lack of performance in the testing results.

6.6 Discussion

6.6.1 Training and testing results

Analyzing the overall metrics of the models under study, their training and validation progress in Section 6.4, and their testing results in Section 6.5, it is possible to identify which models will generally do a good job in their expected use-case scenario, and which ones will have trouble ranking sentences relevant to the content of the target video. Table 6.3 shows the baseline variations with the highest testing performance are INFERSENT-ALL and DT-ALL, both with the strategy to add a new video or sentence embedding to the ones the baseline uses. Between these two, the INFERSENT-ALL is the one with most similar results to the baseline, being the only one reaching the 0.4 MIR using the Shufflenet embeddings, and performing better than the baseline in the sets A and D. The DT-ALL model surprisingly does not have a performance similar to the base model as INFERSENT-ALL does, even when its architecture definition is almost identical. These results suggest the *DT-BoW* descriptor defined in Section 4.1.1 for this model is not as well defined as the InferSent descriptor, and the addition of *DT-BoW* to the base model hinders the learning capacity of the model instead of improving it or, as in the case of InferSent for the INFERSENT-ALL model, not affecting it at all. A better approach for the *DT-BoW* descriptor could be achieved by improving the following aspects of it:

- The selected cluster size using the elbow method (2,500 clusters) could be under-efficient and does not have the complexity necessary to encode the majority of Dense Trajectory patterns of the training dataset. The elbow method of this research explores cluster sizes in the range of 100 and 8000, using only 10% of the training dataset videos and GIFs. An improvement to this could be to use more videos from the training dataset to explore the different cluster sizes, or to do a search in a wider range of sizes. This last option must be carefully explored because too many clusters can decrease the representativity of the dense trajectories and will increase the learnable parameters of the models, which could lead to an overfitting scenario.
- The K-means clustering algorithm selected for the Dense Trajectories with Bag of Words descriptor *DT-BoW*, may not be the optimal algorithm to the problem, mainly because

one of the weaknesses of the K-means algorithm: it can select poor cluster borders because its focus on centroid optimization instead of cluster border optimization. A possible approach is to define the *DT-BoW* descriptors using a density-based algorithm like DBSCAN [78], which could have the advantage of finding better dense clusters of actions encoded with dense trajectories, but it also may show poor results if the density of these clusters is too different [79]. Also, this algorithm uses 2 hyper-parameters: *epsilon* ε and the minimum points required to form a dense region, whose values can be experimentally found using the elbow method.

- Another aspect of the *DT-BoW* descriptor is how is added to the architecture. The approach of this study concatenates both x-axis and y-axis histograms into a single 1-dimensional vector f_4^v that is feed into the vector space mapping (see Section 4.1.1). A different approach for the fusion of the two histograms is to add a mapping layer for each axis before the concatenation step. With this configuration the model is able to learn during the training process what dimension of each axis histogram has more importance for the loss optimization, giving a higher weight before concatenating them into f_4^v and transforming it to the common vector space using the affine transformation explained in Equation 3.12.

Going back to the test results of Table 6.3, if we focus on the worst performing models an interesting behavior is noticed: the performance of DT-ONLY surpasses the results of INFERSENT-ONLY in each of its two video embedding versions. This is interesting because both models aim to subtract every embedding of one side of the baseline, but replacing the visual feature embeddings has less negative impact than replacing the three text embedding of the baseline. This result could mean the performance of this model owes more to the encodings chosen for the text than to those chosen for the video. An experimental approach to test if this behavior is true could be to do more performance comparisons of model variations using different text and video encodings, e.g., compare the MIR of a model with only BoW + Word2Vec as sentence encoding versus the MIR of a model with a convolutional network with less accuracy in a common computer vision task than the network used by the baseline model. However, this experimental analysis escapes the objectives of this research and is proposed to be done in future research.

6.6.2 Models performance in real use-case scenarios

Section 6.5.1 presents 3 different video-sentence examples with the top 1 sentence ranked of 5 models under study that use the Shufflenet ResNeXt-101 variant because they consistently show better performance than the MXnet ones. This example analysis helps to understand in more detail the performance of these models, by looking for certain patterns in the ranking examples that could suggest what is the model priority to rank those sentences as the best description for the target video. This section discusses the possible reasons of the matching behavior of a model according to its results showed in the three examples.

The examples from Figures 6.21 and 6.22 show the INFERSENT-ALL variation has a performance similar to the base model under a good real use-case scenario, which suggests its better Mean Inverted Rank scores in set A and D of Table 6.3 are not statistically significant to conclude this model can outperform the baseline. In fact, the 0.001 MIR difference in set A means that both models are able to rank a relevant sentence in the second place on average, interpretation that can also be applied to the 0.006 MIR difference in set D. Also, when analyzing the results of the baseline and INFERSENT-ALL for the bad performed example of Figure 6.23, both models match the video with sentences related to a stadium or sports, with boys, fans or men as the subject, but they fail to recognize the main action in the scene. This result suggests there is no difference on adding the InferSent descriptor as an additional sentence encoding to the base model, it does not hinder its learning capacity nor improves it.

The behavior of the DT-ALL model in the ranked sentences of Figures 6.22 and 6.23 shows this variation model is able to identify the topic of the actions and subjects portrayed in those videos, but has problems with the fine differences between contexts that share similar topics, e.g. can identify a sport stadium but fails to recognize which sport is portrayed, or can identify a stage but fails to identify if is a runaway or another kind of show. This performance could be a mix between the good results of the baseline and the loss of capacity the *DT-BoW* descriptor applies when coupling it to the baseline model.

The DT-ONLY model shows a similar behavior of DT-ALL in the runaway and basketball examples, where it is able to identify the actions of the target video but, unlike DT-ALL, in some cases this model ignores the subjects portrayed in the video, ranking sentences with contexts where the model thinks the action is more suitable of, e.g. in the video example of Figure 6.23 this model ranks sentences of wrestling when the target video is about a basketball player accidentally hitting a kid, action that could be interpreted as two people hitting each other and, therefore, a context of wrestling is more suitable for this action. Although these examples suggest the DT-ONLY model is able to encode actions of a video, they are not enough to prove this statement and further work must be done to analyze, for example, the accuracy of the model on an action topic classification task in a labeled video-sentence dataset.

The results of the INFERSENT-ONLY suggest this model does not follow any visible patterns in its behavior, being the variation with the lowers performance among all the ones under study. This model shows the InferSent descriptor cannot replace the three text descriptors used by the baseline and the three levels of encoding defined when using these descriptors. But also, as shown by the results of INFERSENT-ALL, it does not increase the performance of the baseline when adding as an extra text descriptor.

6.6.3 Training and testing dataset selection

During the exploratory data analysis of the datasets selected to train, validate and test the models under study, two questions raised about the quality of these collections regarding the representation levels of the training and validation dataset, and the generally low framerates of GIFs in the TGIF dataset.

Do the training and validation datasets have a representation level of the testing dataset?

The comparison analysis of the training dataset against the test dataset in Section 5.3 suggests the large corpus of the former gives it an advantage in the quantity of contexts it covers, making it suitable to be used as training data for the VTT matching & ranking task in general, because this problem does not fixes any kind of categories and its solutions must learn a large quantity of scenarios. However, in the third video example of Section 6.5.1, where a basketball player accidentally hits a kid, we analyze the bad matching results of the baseline and INFERSENT-ALL models, both with the best training and testing performance metrics (see Table 6.3), finding an interesting behavior: both models were able to identify the sport-related context and the main subjects of the video (a man and a kid), but they were unable to rank the correct sentence. After manually searching training examples that could share a similar scenario with no success, this lack of learning examples could be the reason that prevented the two well-performed models to identify the unusual situation of the target video. This example suggests the large training dataset composed of 363,806 video-sentence pairs still has room to be improved by adding more diversity of video scenarios, a key aspect to have a high quality training dataset for the target task of this research.

On the other hand, the idea to use the evaluation data of the challenge from previous years (TRECVID 2016) as the validation dataset of a model, strategy we follow from the authors of our baseline [1], has a direct reason: it shares the same source than the testing dataset of TRECVID 2018. Also, during Section 5.2 we make a comparison between the set A of TRECVID 2016, and the complete dataset of TRECVID 2018, showing a level of representation of the former when we explore the topics of them in Figure 5.7.

Does the generally low framerate of TGIF have a negative impact in the performance of Dense Trajectories models, like DT-ALL or DT-ONLY, when trained with it?

After visualizing in Figure 5.4 the framerate distribution of the training dataset and noticing the high difference of TGIF GIFs against the videos from MSVD and MSR-VTT, we study the possible impact of the low framerate collection in the DT-ALL and DT-ONLY models, as stated in Section 5.1.4. For this we train both DT-ALL and DT-ONLY models without the TGIF dataset and compare its training and testing metrics against their version trained with this dataset. The training results of Figures 6.7 and 6.11, and Mean Inverted Rank of Table 6.3 show the addition of TGIF to the training dataset of the models with Dense Trajectory descriptors has a positive impact in their overall performance compared to the non-TGIF models of the table, an effect contrary of what can be stated at first glance. The possible reason of this result is the large volume of visual and sentence data TGIF offers (125,782 GIFs and sentences) adds more to the performance of a model in terms of quantity and variety of complex patterns to learn, than the performance the low framerate of this dataset could take away, if there is existence of this negative effect. This restates the decision of using TGIF as training dataset for all models under study.

The validation scores evolution of the DT-ALL model is compared in Figure 6.7. This chart shows removing the TGIF dataset decreases the generalization of the model, not being able to obtain half of the validation score of the model trained with TGIF in its training dataset. This lower training score is reflected in the testing results of Table 6.3, where the DT-ALL (SH no TGIF) model obtains almost a third MIR value of its counterpart trained with TGIF, in all testing sets. This shows the use of TGIF dataset has a positive effect in the generalization of the DT-ALL model.

For the DT-ONLY model, Figure 6.11 shows there is no significant difference in the validation loss value when training this model with and without TGIF, but the runs without the dataset stop the training 6 to 8 epochs earlier, suggesting they reach their highest value in fewer iterations. The testing result in Table 6.3 shows the DT-ONLY (no TGIF) model has a lower MIR in every evaluation set, confirming the performance benefits of TGIF as a training dataset for this model.

6.6.4 ResNeXt-101: Shufflenet vs MXnet

The results show the importance of using the original ResNeXt-101 visual feature vectors trained in the Shufflenet dataset, a variant of Imagenet with its object classes balanced in terms of quantity of images per class. In contrast, the MXnet ResNeXt-101 has its parameters trained on the standard Imagenet-1k dataset, which only uses the top 1,000 classes of the Imagenet corpus without dealing with class unbalance. This difference in training datasets may lead to a better object generalization of the Shufflenet network, being able to generate better training encodings for a variety of objects that can appear in videos of random content as the ones in the training and testing datasets.

6.6.5 Training methodology

After analyzing the testing results of Table 6.3 and the rank examples of Section 6.5.1, we wonder about how much each model under study overfits on the training data, even when the validation scores of the models increase during training until a certain maximum. The architecture of all models implements dropout and batch normalization as regularization measures to prevent an overfitting scenario, but this unwanted behavior still may find its way in the generalization capacity of the model.

To measure the overfit levels of the base model and its variations, these models are tested on 3 samples from the training dataset, samples composed of 1,915 training videos and 1,915 training sentences related to one of these videos. This recreates the testing scenario of a model when obtaining its MIR score over a set of the official evaluation collection of VTT matching & ranking task of TRECVID 2018, allowing to do a metric comparison of these results with the ones of Table 6.3. The 1,915 videos of each sample are randomly selected between all the training dataset, taking care there is a similar number of videos or GIFs extracted from MSR-VTT, MSVD and TGIF datasets. The Shufflenet version of the base models and variations is used because this ResNeXt-101 version shows better results than the MXnet one. All 3 training runs of each model are tested under the training samples, showing their average MIR and standard deviation error in Table 6.4. These results show almost all models have a better performance in the 3 random training samples than in the testing dataset, being the only exception INFERSENT-ONLY because its low scores both in the training samples and in the testing data.

The base model obtains a minimum MIR of 0.634 on training sample 1, while it obtained a maximum MIR of 0.442 in the test set E of Table 6.3. INFERSENT-ALL gets a minimum of 0.589 in sample 2, and a maximum of 0.439 in testing set E. DT-ALL obtains a minimum of 0.571 in training sample 3, while in the testing set E obtains a maximum of 0.293. DT-ONLY minimum sample MIR is 0.076 in training sample 2, while its maximum MIR in the testing data is 0.012, showing a higher average score in the training data even taking into account the standard deviation error. Finally, the INFERSENT-ONLY model obtains a minimum sample MIR of 0.005 in the training sample 3, while the maximum MIR in the testing dataset is 0.005 in sets B and C, although these results are not sufficient to say INFERSENT-ONLY has a better performance on the training dataset. These results show the base model, INFERSENT-ALL, DT-ALL and DT-ONLY models have a level of overfitting that could be reduced by using more regularization techniques like adding weight decay to the Adam optimizer used in the training process, or by decreasing the training batch size [80]. The reason we choose not to use them is to have the same training configuration than the baseline model to replicate its reported results in TRECVID 2018 [1], and do a correct comparison of its variants.

Model	Sample 1	Sample 2	Sample 3
Dong *et al.* 2018 (SH)	0.634 ± 0.029	0.637 ± 0.023	0.643 ± 0.023
INFERSENT-ALL	0.596 ± 0.038	0.589 ± 0.035	0.605 ± 0.033
INFERSENT-ONLY	0.006 ± 0.001	0.006 ± 0.001	0.005 ± 0.001
DT-ALL (SH)	0.569 ± 0.036	0.569 ± 0.035	0.571 ± 0.033
DT-ONLY (SH)	0.080 ± 0.025	0.076 ± 0.030	0.083 ± 0.033

Table 6.4: Average Mean Inverted Rank with its standard deviation error of the results of the 3 training runs of each model under study, over the 3 random training samples created to test their overfit levels. Only models with a ResNeXt-101 pre-trained on Shufflenet are tested.

Finally, a major bottleneck in this investigation was the missing information about the hyper-parameters of the baseline implementation where all models of this study are based on. Li *et al.* 2018 [1] stipulate they do a hyper-parameter optimization using the TRECVID 2016 Matching & Ranking training dataset, which according to them, comprises 200 video-sentence pairs. However, this dataset does not exist in the official repositories of the TRECVID conference, and they do not provide it either. Fortunately the default hyper-parameters they use in the model's repository, which they use in performance benchmark using the MSR-VTT dataset, give a guideline of the values these important parameters should have.

Chapter 7

TRECVID 2019 participation

We took part in the Matching & Ranking task of the TRECVID 2019 challenge. The submissions for this competition follow the approach of Dong *et al.* for the same task of TRECVID 2017 [2]. Although we did not win the competition, we spotted the flaws of this architecture and made a model base on Li *et al.* 2018 [1] (see Section 3). We emphasize the model of this participation is different from the ones presented in this study.

7.1 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++ [1] 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 [4] as a visual embedding to encode temporal information of the video. We think that combining the pre-trained Resnet-152 [2] and ResNeXt-101 [43] descriptors of W2VV++, with Dense Trajectories descriptors of the video, we extract spatial and temporal information from it, respectively.

Dense Trajectories [4] 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 generates 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 were 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.

$$fusion_s = BoW(s) \; || \; w2v(s) \; || gru(s) \tag{7.1}$$

For the video representation, define $\max_{rs152}(v)$ as the max pooling of the set of pre-trained ResNet-152 [2] vectors: and $\max_{rs101}(v)$ the max pooling of the pre-trained ResNeXt-101 [43] vectors, we get a single vector representation of the video by concatenating them with $BoW_{\mathrm{dt}}(v)$: the Bag of Words vector resulting from the clustering of the Dense Trajectories of the video.

$$fusion_v = BoW_{\mathrm{dt}}(v) \; || \; \max_{rs152}(v) \; || \; \max_{rs101}(v) \tag{7.2}$$

Following the Matching and Ranking model of Dong *et al.* for TRECVID 2017 [2], to transform the sentence and video embedding into a common vector space, we use two fully connected layers with a ReLU activation function. We also use batch normalization [81] in one of the submitted runs, e.g., for the video branch of the model:

$$\begin{aligned} v' &= \sigma(BN(W_1 fusion_v + b_1)) \\ v &= \sigma(BN(W_2 v' + b_2)) \end{aligned} \tag{7.3}$$

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

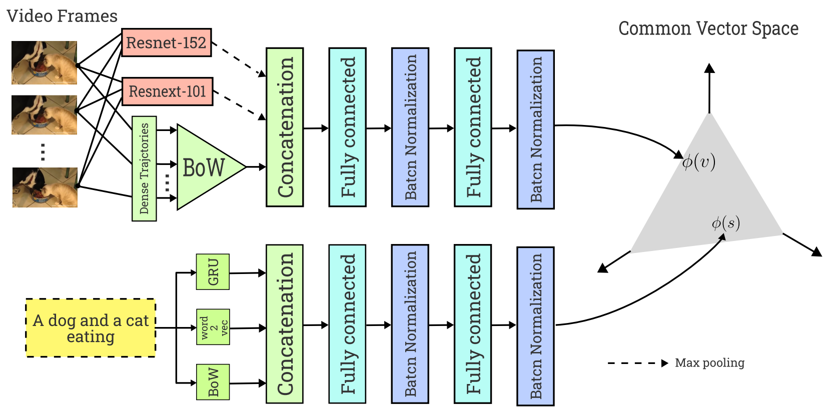


Figure 7.1: Model used for Matching and Ranking subtask. We use this configuration with batch normalization for the run 1, but for the second run this feature is missing.

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

$$l(v, s; \theta) = \max_{s_-}(0, \alpha + S(v, s_-) - S(v, s))) \tag{7.4}$$

where (v, s) is a corresponding video-sentence pair, θ is the set of parameters of the model, 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 s and s_{-} and S is the cosine similarity function. With this, if D is the training dataset, the optimization process aims to minimize the sum:

$$\min_{\theta} \sum_{(v,s)\in D} l(v, s; \theta) \tag{7.5}$$

We also follow Dong *et al.*'s [2] approach for the optimization of equation 7.5. Using RMSProp [82] with initial learning rate $\eta = 0.0003$, weight decay $\gamma = 0.9$ and $\varepsilon = 10^{-9}$. 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, stopping it if there is no improve in 5 iterations.

7.2 Datasets

We trained the model using MSR-VTT [13], which comprises more than 150k of video-sentence pairs. The videos come from the YouTube™ web platform and the 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.

7.3 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 we remove this feature. For a submission subset, to rank a video v with a set of sentences $\{s\}_{1..n}$ we use the cosine similarity function S used in the loss function (7.4), calculating the distance between the vector representation of video in the common space v and the vector representations of the set of sentences $\{s\}_{1..n}$, and sorting them in decreasing order to get the ranking list for v.

Table 7.1 shows the Mean Inverted Rank score (see Section 6.1) results of all 4 participants in the VTT matching & ranking subtask of TRECVID 2019: 4 submissions from the Renmin University of China (RUC_AIM3) [83], 4 from the Renmin University of China and Zhejiang Gongshang University team (RUCMM) [84], a single submission from Nagaoka University of Technology (KsLab) [85], and our 2 described submissions. The results show the first submission of the Renmin University of China, consisting of an ensemble of two models based on our baseline by Dong *et al.* 2018 [11] using BERT [86] sentence descriptors, outperforms every other entry in the competition, obtaining the highest score in the subtask at the moment, although these scores are not comparable with previous years because its different evaluation data, as stated by the TRECVID organizers in their competition report [67]. If we focus on our results, the second run with no batch normalization got better results in the subtask, with a consistently higher score than the batch normalization approach.

Run	A	B	C	D	E
RUC_AIM3 run 1	**0.723**	**0.727**	**0.721**	**0.721**	**0.722**
RUC_AIM3 run 2	0.623	0.635	0.627	0.636	0.630
RUC_AIM3 run 3	0.569	0.581	0.575	0.579	0.575
RUC_AIM3 run 4	0.572	0.580	0.574	0.579	0.572
RUCMM run 1	0.432	0.442	0.429	0.437	0.441
RUCMM run 2	0.436	0.443	0.444	0.447	0.446
RUCMM run 3	0.474	0.480	0.474	0.487	0.481
RUCMM run 4	0.471	0.480	0.470	0.484	0.477
Our run 2	0.015	0.017	0.015	0.016	0.018
Our run 1	0.004	0.005	0.004	0.005	0.005
KsLab run 1	0.002	0.003	0.002	0.002	0.003

Table 7.1: Mean Inverted Rank obtained by all participants in every evaluation set in the VTT Matching and Ranking subtask of TRECVID 2019. Best result by evaluation set is highlighted in bold.

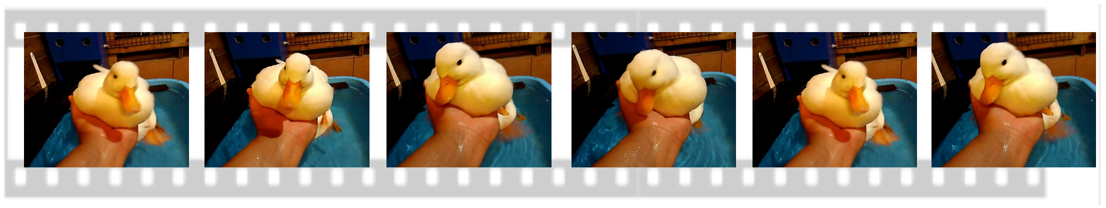
7.4 Result examples

Taking into account the definition of Mean Inverted Rank (see Section 6.1), the 0.005 to 0.004 score obtained of our run 1 show this submission ranked, on average, the corresponding sentence of a video in the position 200th to 250th, respectively: while the 0.018 to 0.015 score of our run 2 means an average rank between positions 55th to 66th. This results show both models do not perform well in the VTT matching & ranking substask, in fact, as shown in Table 7.2 for each evaluation set, only a few videos got their corresponding sentence ranked in position 3 or lower. Because there is a few good examples to choose from, only a single example for our first and second run is shown in Figures 7.2 and 7.3, respectively.

Run	A	B	C	D	E
Our run 2	15	15	13	17	22
Our run 1	2	4	4	2	4

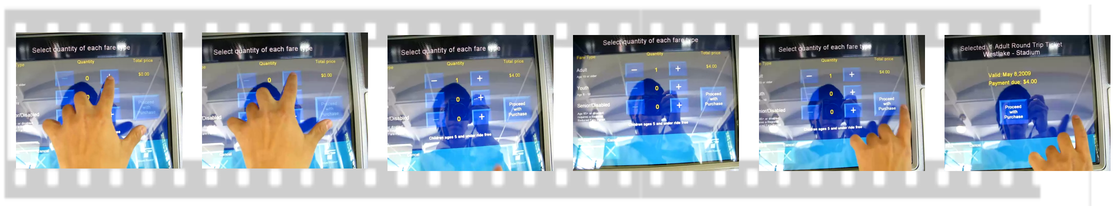
Table 7.2: Quantity of videos, from a total of 2054, with their corresponding sentence ranked in the first 3 positions, for each evaluation set of the subtask.

The ranking example of our first submission in Figure 7.2 shows the model has trouble identifying subject, action and context of the video. In set E our submission with batch normalization is able to rank in third place a relevant sentence of the video where a person is bathing a duck, while in set A the model is able to rank sentences related to an animal bathing or an aquatic animal, but the other evaluation sets do not share any visible patterns related or unrelated to the target topic, reflecting its poor performance reported in Table 7.1.



Set A
1º) A woman bathing a kitten in a sink.
2º) A man is kneeling in front of a cage of a baby hippo and the hippo comes around to the side he is on.
3º) Diapered toddler in room with many toys, picks up a toy weight, hoists it, then throws it and goes into hero stance, head down, hands up, legs apart and knees bent.
Set B
1º) Dog on couch is swatted by black cat reaching up from floor.
2º) A little boy is drumming with two sticks in front of arcade machine,.
3º) Using a red and black marker someone is writing something on a pad on a table.
Set C
1º) A slab of ice is placed into a large grinder and the chipped ice flows down into a container where it is smoothed out by a woman in a hat with a small shovel.
2º) Boys in a dormitory setting of single beds, all crawl under their covers at same time.
3º) A young man takes a puff of a cigarette and blows through a bubble wand, producing white bubbles
Set D
1º) A head of mouse is held on center of a huge wheel, with illuminated spokes, at night.
2º) A woman lets one dog to lick from an ice cream cone, and hands it to the second dog that swallows the entire cone, in a car, at day time.
3º) Two small children playing on the floor with spoons, bowls and pots next to a large stuffed toy tiger during the day.
Set E
1º) An young child stand and laughs in a crib with a young toddler next to them trying to stand up.
2º) A person with a knife repeatedly hits an orange causing it to slice in many pieces on a piece of newspaper on a table in a room.
3º) A person holding a live yellow ducky in an inside room.

Figure 7.2: Top 3 ranked sentences of our run 1 for evaluation video id 1,105, where a person is bathing a white duck.



Set A
1º) A hand selects tickets on a parchuse screen.
2º) Man tilts over backward in armchair into a flaming background.
3º) A white walkie-talkie like machine is blinking and making noises.
Set B
1º) Ribbon cutting ceremony inside an auditorium with Carnegie Mellon University displayed on the screen in back.
2º) Video of hand with the reflection of the person showing on the screen of a self-service ticket machine.
3º) On a lighted stage, person reaches down and pulls up his socks.
Set C
1º) Man operates claw machine and wins a piece of candy.
2º) A man on a ladder at night changing the marquee at The Castro Theater.
3º) Man urinates.
Set D
1º) A group of people are getting ready to put a sarcophagus in a MRI machine.
2º) Man shows a yellow box to others.
3º) A man spins a box which becomes wrapped with a ribbon in mid spin.
Set E
1º) A man at a podium with the logo www.fahd-alrougui.com on it talks.
2º) An indoor digital art exhibit.
3º) Person watching items suspended by wire being raised and lowered.

Figure 7.3: Top 3 ranked sentences of our run 2 for evaluation video id 814, where a person is pressing the touchscreen of a ticket machine.

The ranking of our submission without batch normalization in Figure 7.3 shows better results than the first one, where is able to rank two related sentences in set A and set B, while set C and E the model is able to rank sentences related to machines like "man operates a claw machine" and "An indoor digital art exhibit", respectively. It set D, even though it does not rank sentences related to the video content, it is able to show a shared subject pattern of boxes or sarcophagus between the first 3 ranked sentences. Although in this example the model without batch normalization shows certain patterns, is not enough to prove the model has learned them or overfit them.

The low performance of both submissions can be derived from two reasons:

- The extra parameters in run 1 added by the batch normalization may overfit the model on the training dataset.
- As shown in Table A.1 of Appendix A, our own implementation of the base model [2] selected for this competition submission failed to replicate the reported results of this model, which led to a poor performed base architecture for the variation with Dense Trajectories. The problems with the implementation of the baseline were multiple, being the main ones the inaccurate explanation of the soft-attention mechanism by the authors and the ambiguous description of the validation dataset used by Dong *et al.* .

Chapter 8

Conclusions and future work

8.1 Conclusions

This research tests the effectiveness of Dense Trajectories and InferSent description in the Video-to-Text Matching & Ranking task. For this we selected a state-of-the-art model and applied different variations to use the two descriptors. After the multiple analysis of data and results done in this work, the quantitative analysis of the Mean Inverted Rank performance of the models under study, and the qualitative analysis of video matching examples in Section 6.5.1, the main conclusion of this work are:

- The results show the effectiveness of the three text encoding levels of the base model, not being replaceable by InferSent, a sentence descriptor with a high transfer learning performance. In the INFERSENT-ALL model, the addition of this descriptor as an additional sentence encoding to the baseline, does not improve or hinders the generalization capacity of the base model, obtaining similar scores in their training process and testing results. On the other hand, replacing the 3 text encoders of the base model by Li *et al.* [1] with the InferSent descriptor, as is done in the INFERSENT-ONLY model, has the most negative impact on the base model performance of all the variations of this study, obtaining the lowest MIR scores in the testing dataset and not showing any visible pattern in the sentence rankings of the video examples in Section 6.5.1.
- The results of the DT-ALL model show the addition of the *DT-BoW* descriptor hinders the capacity of the base model, reducing its performance on the testing dataset by a half in almost all evaluations subsets and, as seen in some video examples, having trouble to differentiate between video contexts that share a similar topic. The reasons of this behavior may rely on the *DT-BoW* descriptor definition, the clustering process done to generate it, and the strategy selected to integrate it on the base model architecture, as discussed in Section 6.6.1.
- The DT-ONLY model results show a loss in performance when removing the static information descriptor ResNeXt-101, along with the 3 levels of video encoding of the base model. This architecture, as seen in the sentences ranked in the examples of Figures 6.22 and 6.23, is able to encode some actions but fails to properly encode the subject of the video, matching sentences with subjects and contexts more accordingly to the

action portrayed in the video, independently of the real subject. This is an interesting behavior that can support the approach of separating the information retrieval into two roles: encoding static information retrieval with a pre-trained ResNeXt-101 and encoding temporal information with a Dense Trajectory descriptor, but further investigation must be done to completely prove this.

- The comparison of performance using the original ResNeXt-101 pre-trained on Shufflenet versus the same architecture pre-trained on the standard Imagenet-1k dataset, shows the critical importance of the first pre-trained architecture in the state-of-the-art results of the base model.
- The results of using the InferSent sentence descriptor to encode the textual data in the model of Li *et al.* [1] to solve the VTT matching & ranking task of TRECVID 2018 concludes to be unnecessary because it does not improve the performance of the model in any of the two configurations studied.
- The results for the use of Dense Trajectory descriptors in the base model suggest it is capable of encoding the actions portrayed in certain videos, as the results of the video examples show, but the definition for the *DT-BoW* descriptor can be improved in several aspects, or a different approach to be integrated to the base model architecture is needed.
- The low scores of our TRECVID 2019 for the VTT matching & ranking subtask were obtained because our incorrect implementation of the base model of Dong *et al.* of 2017 [2], but these results gave us the insight necessary to switch to a different baseline for this work, selecting the model presented by Li *et al.* [1] from where we defined and studied different variations in this research.
- The low framerate of GIFs in the TGIF dataset has no significant impact in the quality of this dataset, even for models that rely on the movements of pixels like DT-ALL or DT-ONLY. Sections 6.4.3 and 6.4.4 show that removing the TGIF dataset from the training dataset of these two models drastically reduces the performance of DT-ALL, or has no impact in the low performance of DT-ONLY. A possible reason of this result is the large volume of visual and sentence data TGIF offers (125,782 GIFs and sentences) adds more to the performance of a model in terms of quantity and variety of complex patterns to learn, than the performance the low framerate of this dataset could take away, if there is existence of this negative effect.

8.2 Source Code

This research presents a unique approach to the video-to-text Matching & Ranking task by exploiting a Dense Trajectories descriptor to extract temporal information from a video, and the InferSent sentence encoding model to obtain a vector representation of a text.

We publish the implementation of the validated baseline and its 5 variations using an updated machine learning framework [1]. To give a better access to the use of Dense Trajectories, we compiled the original C++ version into a Python 3 library to generate this descriptor, along with a GPU accelerated version to speed up the process (see Section 4.1.1).

[1] `https://gitlab.com/nbravo/vttm-dt`

8.3 Baseline validation obstacles

The reproduction of results for the baseline has several difficulties that we want to address. The authors do not publish their selection of hyper-parameters in any direct form, and the method they describe in [1] to get them is unclear, creating a bottleneck for further investigations that need to use their model and reproduce their results. Also, the need of using such specific ResNeXt-101 pre-trained in Shufflenet creates another obstacle to reproduce the results, restricting the use of other pre-trained networks to create visual vectors for this model.

8.4 Future work

The results obtained by the DT-ALL and DT-ONLY models suggest the Dense Trajectory descriptors have the potential to describe the actions of a video in the VTT matching & ranking task of TRECVID 2018, but the approach of this investigation, which we call *DT-BoW* (see Section 4.1.1) may not have been obtained the maximum results out of these descriptors, and several aspects can be improved:

- The selected cluster size using the elbow method (2,500 clusters) could be under-efficient and does not have the complexity necessary to encode the majority of Dense Trajectory patterns of the training dataset. The elbow method of this research explores cluster sizes in the range of 100 and 8000, using only 10% of the training dataset videos and GIFs. An improvement to this could be to use more videos from the training dataset to explore the different cluster sizes, or to do a search in a wider range of sizes. This last option must be carefully explore because too many clusters can decrease representation of the dense trajectories and will increase the learnable parameters of the models, which could lead to an overfitting scenario.
- The K-means clustering algorithm selected for the Dense Trajectories with Bag of Words descriptor *DT-BoW*, may not be the optimal algorithm to the problem, mainly because one of the weaknesses of the K-means algorithm: it can select poor cluster borders because its focus on centroid optimization instead of cluster border optimization. A possible approach is to define the *DT-BoW* descriptors using a density-based algorithm like DBSCAN [78], which could have the advantage of finding better dense clusters of actions encoded with dense trajectories, but it also may show poor results if the density of these clusters is too different [79]. Also, this algorithm uses 2 hyper-parameters: *epsilon* ε and the minimum points required to form a dense region, whose values can be experimentally found using the elbow method.
- Another aspect of the *DT-BoW* descriptor is how is added to the architecture. The approach of this study concatenates both x-axis and y-axis histograms into a single 1-dimensional vector f_4^v that is feed into the vector space mapping (see Section 4.1.1). A different approach for the fusion of the two histograms is to add a mapping layer for each axis before the concatenation step. With this configuration the model is able to learn during the training process what dimension of each axis histogram has more importance for the loss optimization, giving a higher weight before concatenating them into f_4^v and transforming it to the common vector space using the affine transformation explained in Equation 3.12.

The results of Table 6.3 indicate that introducing our Dense Trajectories descriptor as an additional video input impacts the model performance by reducing its Mean Inverted Rank almost to its half. This behavior does not occur with the introduction of InferSent in the sentence side of the model, in fact INFERSENT-ALL outperforms the base model in set A and D. A next step would be finding the reason of this, where an initial hypothesis is that these results reflect a high sensitivity of the video side to changes in its architecture, in comparison to the sentence side of the model. An experimental approach to test if this behavior is true could be to do more performance comparisons of model variations using different text and video encodings, e.g., compare the MIR of a model with only BoW + Word2Vec as sentence encoding versus the MIR of a model with a convolutional network with less accuracy in a common computer vision task than the network used by the baseline model. However, this experimental analysis escapes the goals of this research and is proposed to be done in future research.

Further investigation is needed to determinate the exact reasons why the reproduction of the reported results of the baseline needs the convolutional network to be trained on such specific Shufflenet dataset. A preliminary hypothesis is that the Shufflenet dataset has more image samples per class, which gives a better object generalization than Imagenet-1k, allowing the ResNeXt-101 to create better descriptors for more video frames contexts.

After finding a level of overfitting in the base model, INFERSENT-ALL, DT-ALL and DT-ONLY models, the testing results of this research could be improved if more regularization techniques are applied, like adding weight decay to the Adam optimizer used in the training process, or by decreasing the training batch size [80].

Bibliography

[1] X. Li, J. Dong, C. Xu, J. Cao, X. Wang, and G. Yang, "Renmin University of China and Zhejiang Gongshang University at TRECVID 2018: Deep cross-modal embeddings for video-text retrieval," in *2018 TREC Video Retrieval Evaluation, TRECVID 2018*, 2020.

[2] J. Dong, S. Huang, D. Xu, and D. Tao, "DL-61-86 at TRECVID 2017: Video-to-Text Description," *TRECVid Conference, 13-15 Nov 2017, Gaithersburg, Md., USA.*, 2017.

[3] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Stroudsburg, PA, USA), pp. 670–680, Association for Computational Linguistics, 2017.

[4] P. Xu, Z. Yu, W. Jin, and H. Jiang, "Action Recognition by Improved Dense Trajectories," in *Xitong Fangzhen Xuebao / Journal of System Simulation*, vol. 29, pp. 2053–2058, IEEE, 2017.

[5] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *Eurasip Journal on Advances in Signal Processing*, vol. 2016, no. 1, p. 67, 2016.

[6] M. Marsden, E. Mohedano, K. McGuinness, A. Calafell, X. Giró-I-Nieto, N. E. O'Connor, J. Zhou, L. Azevedo, T. Daudert, B. Davis, M. Hürlimann, H. Afli, J. Du, D. Ganguly, W. Li, A. Way, and A. F. Smeaton, "Dublin city university and partners' participation in the INS and VTT tracks at TRECVID 2016," *2016 TREC Video Retrieval Evaluation, TRECVID 2016*, 2016.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Communications of the ACM*, vol. 60, pp. 84–90, 2017.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, pp. 1–9, Cvpr, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, 2016.

[11] J. Dong, X. Li, C. Xu, S. Ji, Y. He, G. Yang, and X. Wang, "Dual encoding for zero-example video retrieval," in *Computer Vision and Pattern Recognition (CVPR), 2019 IEEE/CVF Conference on*, pp. 9346–9355, IEEE, 2018.

[12] Y. Li, Y. Song, L. Cao, J. Tetreault, L. Goldberg, A. Jaimes, and J. Luo, "TGIF: A new dataset and benchmark on animated GIF description," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 4641–4650, 2016.

[13] J. Xu, T. Mei, T. Yao, and Y. Rui, "MSR-VTT: A large video description dataset for bridging video and language," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 5288–5296, IEEE, 2016.

[14] D. L. Chen and W. B. Dolan, "Collecting highly parallel data for paraphrase evaluation," in *ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 190–200, Association for Computational Linguistics, 2011.

[15] P. S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *International Conference on Information and Knowledge Management, Proceedings*, pp. 2333–2338, ACM, 2013.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, 2013.

[17] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 1724–1734, 2014.

[18] Z. Tu, Z. Lu, L. Yang, X. Liu, and H. Li, "Modeling coverage for neural machine translation," *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, vol. 1, pp. 76–85, 2016.

[19] X. Li and X. Wu, "Long short-term memory based convolutional recurrent neural networks for large vocabulary speech recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2015-January, pp. 3219–3223, IEEE, 2015.

[20] T. Zia and U. Zahid, "Long short-term memory recurrent neural network architectures for Urdu acoustic modeling," 2019.

[21] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A

novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2009.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] P. J. Werbos, "Backpropagation Through Time: What It Does and How to Do It," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[24] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[25] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," 2000.

[26] J. D. Kalbfleisch and D. A. Sprott, "Marginal and Conditional Likelihoods," *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, vol. 35, no. 3, pp. 311–328, 1973.

[27] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative Study of CNN and RNN for Natural Language Processing," *arXiv preprint arXiv:1702.01923*, feb 2017.

[28] M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli, "SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment," in *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pp. 1–8, 2015.

[29] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, 2015.

[30] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF Models for Sequence Tagging," *arXiv preprint arXiv:1508.01991*, aug 2015.

[31] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 160–167, ACM, 2008.

[32] R. K. McConnell, "Method of and apparatus for pattern recognition," U.S. Patent US-4075604-A, Jan. 1986.

[33] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 1–8, IEEE, 2008.

[34] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3952 LNCS, pp. 428–441, Springer, 2006.

[35] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2749, pp. 363–370, Springer, 2003.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 779–788, 2016.

[38] C. Zhang, Y. Zou, G. Chen, and L. Gan, "PAN: Persistent appearance network with an efficient motion cue for fast action recognition," in *MM 2019 - Proceedings of the 27th ACM International Conference on Multimedia*, pp. 500–509, 2019.

[39] A. Berg J. Deng and L. Fei-Fei, "Large scale visual recognition challenge," in *ILSVRC 2012 Workshop*, vol. 2012, pp. 2011–2013, 2010.

[40] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 512–519, IEEE, 2014.

[41] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, IEEE, 2014.

[42] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "VQA: Visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 International Conference on Computer Vision, ICCV 2015, pp. 2425–2433, 2015.

[43] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 5987–5995, IEEE, 2017.

[44] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen. Diploma. Technische Universität München," *Diploma, Technische Universität München*, vol. 91, no. 1, 1991.

[45] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, vol. 6, no. 2, pp. 107–116, 1998.

[46] J. F. Kolen and S. C. Kremer, "Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies," 2010.

[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke,

and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, pp. 1–9, Cvpr, 2015.

[48] P. Mettes, D. C. Koelma, and C. 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*, pp. 175–182, 2016.

[49] L. Jiang, S. I. Yu, D. Meng, Y. Yang, T. Mitamura, and A. G. Hauptmann, "Fast and accurate content-based semantic search in 100M Internet Videos," in *MM 2015 - Proceedings of the 2015 ACM Multimedia Conference*, pp. 49–58, 2015.

[50] M. Nagel, T. Mensink, and C. G. M. Snoek, "Event Fisher Vectors: Robust Encoding Visual Diversity of Visual Streams," in *BMVC*, vol. 2, pp. 178.1–178.12, 2015.

[51] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[52] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

[53] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *ASIA CCS 2017 - Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security*, pp. 506–519, 2017.

[54] J. Shao, K. Kang, C. C. Loy, and X. Wang, "Deeply learned attributes for crowded scene understanding," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, pp. 4657–4666, 2015.

[55] B. Zhou, A. Lapedriza, A. Torralba, and A. Oliva, "Places: An Image Database for Deep Scene Understanding," *Journal of Vision*, vol. 17, no. 10, p. 296, 2017.

[56] G. Awad, A. A. Butt, J. Fiscus, D. Joy, A. Delgado, W. McClinton, M. Michel, A. F. Smeaton, Y. Graham, W. Kraaij, G. Quénot, M. Eskevich, R. Ordelman, G. J. Jones, and B. Huet, "TRECVID 2017: Evaluating ad-hoc and instance video search, events detection, video captioning, and hyperlinking," in *2017 TREC Video Retrieval Evaluation, TRECVID 2017*, vol. 2017, 2017.

[57] W. Li, W. Xie, and Z. Wang, "Complex-Valued Densely Connected Convolutional Networks," in *Communications in Computer and Information Science*, vol. 1257 CCIS, pp. 299–309, 2020.

[58] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, IEEE, 2009.

[59] B. Fernando, E. Gavves, M. Jose Oramas, A. Ghodrati, and T. Tuytelaars, "Rank Pooling for Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 773–787, 2017.

[60] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[61] Y. Kim, "Convolutional neural networks for sentence classification," *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 1746–1751, 2014.

[62] T. Demeester, I. Sutskever, K. Chen, J. Dean, and G. Corado, "Distributed Representations of Words and Phrases and their Compositionality," in *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 1389–1399, 2016.

[63] J. Dong, X. Li, and C. G. Snoek, "Predicting Visual Features from Text for Image and Video Caption Retrieval," *IEEE Transactions on Multimedia*, vol. 20, no. 12, pp. 3377–3388, 2018.

[64] F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler, "VSE++: Improving Visual-Semantic Embeddings with Hard Negatives," in *Proceedings of the British Machine Vision Conference*, 2018.

[65] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 6299–6308, IEEE, 2017.

[66] G. Awad, A. A. Butt, K. Curtis, Y. Lee, J. Fiscus, A. Godil, D. Joy, A. Delgado, A. F. Smeaton, Y. Graham, W. Kraaij, G. Quénot, J. Magalhaes, D. Semedo, and S. Blasi, "TRECVID 2018: Benchmarking video activity detection, video captioning and matching, video storytelling linking and video search," in *2018 TREC Video Retrieval Evaluation, TRECVID 2018*, 2020.

[67] G. Awad, A. A. Butt, K. Curtis, Y. Lee, J. Fiscus, A. Godil, A. Delgado, J. Zhang, E. Godard, L. Diduch, A. F. Smeaton, Y. Graham, W. Kraaij, and G. Quénot, "TRECVID 2019: An evaluation campaign to benchmark video activity detection, video captioning and matching, and video search & retrieval," in *2019 TREC Video Retrieval Evaluation, TRECVID 2019*, vol. 2019, 2020.

[68] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[69] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*, pp. 1177–1178, 2010.

[70] L. E. Zhang, J. Zhu, and T. Yao, "An evaluation of statistical spam filtering techniques," *ACM Transactions on Asian Language Information Processing*, vol. 3, no. 4, pp. 243–269, 2004.

[71] W. Ling, C. Dyer, A. Black, and I. Trancoso, “Two/too simple adaptations of Word2Vec for syntax problems,” in *NAACL HLT 2015 - 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, pp. 1299–1304, 2015.

[72] J. Kiefer and J. Wolfowitz, “Stochastic Estimation of the Maximum of a Regression Function,” *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, 1952.

[73] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[74] R. Pascanu, T. Mikolov, and Y. Bengio, “Understanding the exploding gradient problem,” *Proceedings of The 30th International Conference on Machine Learning*, vol. 2, no. 2, pp. 1310–1318, 2012.

[75] R. L. Thorndike, “Who belongs in the family?,” *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953.

[76] D. L. Davies and D. W. Bouldin, “A Cluster Separation Measure,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[77] G. Awad, J. Fiscus, D. Joy, M. Michel, A. F. Smeaton, W. Kraaij, G. Quénot, M. Eskevich, R. Aly, R. Ordelman, M. Ritter, G. J. Jones, B. Huet, and M. Larson, “TRECVID 2016: Evaluating video search, video event detection, localization, and hyperlinking,” in *2016 TREC Video Retrieval Evaluation, TRECVID 2016*, 2016.

[78] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996.

[79] H. P. Kriegel, P. Kröger, J. Sander, and A. Zimek, “Density-based clustering,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.

[80] D. R. Wilson and T. R. Martinez, “The general inefficiency of batch training for gradient descent learning,” *Neural Networks*, vol. 16, no. 10, pp. 1429–1451, 2003.

[81] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, 2015.

[82] G. Hinton and T. Tieleman, “RMSPROP: Divide the Gradient by a Running Average of its Recent Magnitude,” *Coursera: Neural Networks for Machine Learning*, vol. 4, no. 2, pp. 26–31, 2012.

[83] Y. Song, Y. Zhao, S. Chen, and Q. Jin, “RUC_AIM3 at TRECVID 2019: Video to text,” in *2019 TREC Video Retrieval Evaluation, TRECVID 2019*, 2020.

[84] X. Li, J. Ye, C. Xu, S. Yun, L. Zhang, X. Wang, R. Qian, and J. Dong, "Renmin University of China and Zhejiang Gongshang University at TRECVID 2019: Learn to search and describe videos," in *2019 TREC Video Retrieval Evaluation, TRECVID 2019*, 2020.

[85] R. Kondo and T. Yukawa, "An automatic caption generation for video clip with reducing frames in order to shorten processing time," in *2019 TREC Video Retrieval Evaluation, TRECVID 2019*, 2020.

[86] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 4171–4186, 2019.

Appendix A

Dong *et al.* 2017 model validation

Dong *et al.* submission of TRECVID 2017 [2] got the best Mean Inverted Rank score (see subsection 6.1) in the competition: the reason we implemented this approach at the beginning of the research. The code of this model is not public, therefore we made the model from scratch by following the description in the competition's paper [2]. We abandoned this model as a baseline because of several inconsistencies in its definition.

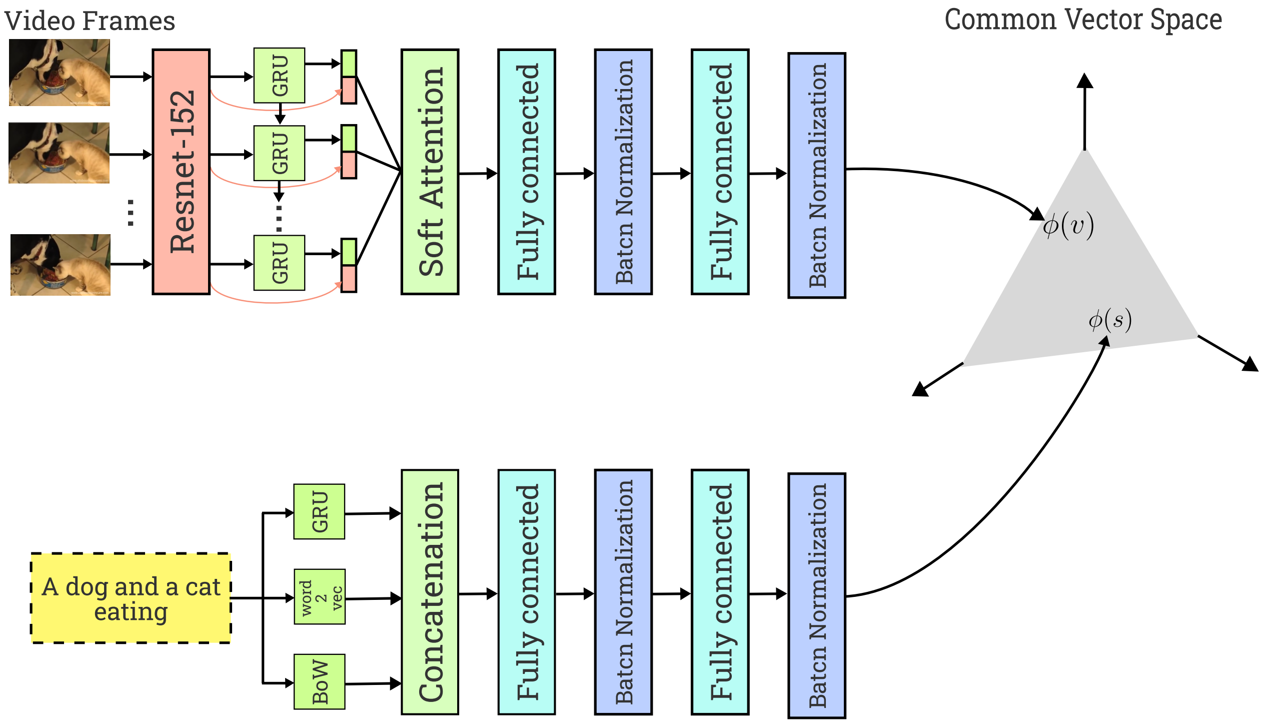


Figure A.1: Model published by Dong *et al.* for Ad-hoc video search and Matching and Ranking tasks in TRECVID 2017.

The selected base model of Dong *et al.* 2018 follows the same strategy of this model: a multi-modal neural network transform video and sentence embeddings into a common vector space where a distance function is used to rank and match video-sentence pairs.

The video encoder extracts visual features using a pre-trained Resnet-152 [10], then a set of Gated Recurrent Units [17] encodes the sequence of visual feature vectors into a set of temporal information vectors, then it concatenates them with the original visual feature vector using a skip connection. Later a specific attention mechanism is used to summarize the set concatenated vectors into a single vector representation. Finally, a fully connected neural network of two layers with Batch Normalization transforms this single vector representation into the common vector space.

The sentence side of the model uses three different text embeddings as input: a Bag of Word mechanism, a pre-trained Word2Vec [62] model and a Gated Recurrent Unit [17] representation, concatenating them to create a single vector representation of the sentence. The transformation of this vector to the common vector space is like the single vector representation of the video: by a fully connected neural network of two layers, with Batch Normalization to prevent over-fitting.

In the explanation of the Soft-attention mechanism, there is a big problem where they say they are using the *concatenation of scalars* operation to create a summary vector representation of the attention, but this operation makes little sense because it applies to numbers between -1 and 1 and if we use the concatenation, we end up losing most of the scalars in non-important float positions. We assumed this issue is a typo and Dong *et al.* meant to apply an addition of scalars instead of a concatenation.

Set	2		3			4				5				
Subset	A	B	A	B	C	A	B	C	D	A	B	C	D	E
Reported Dong *et al.* 2017	0.350	0.350	0.500	0.500	0.500	0.600	0.600	0.600	0.600	0.700	0.700	0.700	0.700	0.700
Our implementation	0.006	0.005	0.008	0.008	0.007	0.014	0.014	0.019	0.013	0.031	0.030	0.046	0.035	0.034

Table A.1: Mean Inverted Rank reported by Dong *et al.* 2017 [2] and that obtained by our implementation.

Because of the inaccurate explanation of the soft-attention mechanism, the missing information of the fully connected network configuration and, because of these issues, the difficulty of recreating the reported results using the same training and testing data, we discarded this model as a baseline for our research.