

# SEMANTIC AND VISUAL SIMILARITIES FOR EFFICIENT KNOWLEDGE TRANSFER IN CNN TRAINING

*Lucas Pascal<sup>1,2</sup>, Xavier Bost<sup>1</sup>, Benoit Huet<sup>2</sup>*

<sup>1</sup>Orkis, Aix-en-Provence, France  
{lpascal, xbst}@orkis.com

<sup>2</sup>EURECOM, Sophia Antipolis, France  
{pascal, huet}@eurecom.fr

## ABSTRACT

In recent years, representation learning approaches have disrupted many multimedia computing tasks. Among those approaches, deep convolutional neural networks (CNNs) have notably reached human level expertise on some constrained image classification tasks. Nonetheless, training CNNs from scratch for new task or simply new data turns out to be complex and time-consuming. Recently, transfer learning has emerged as an effective methodology for adapting pre-trained CNNs to new data and classes, by only re-training the last classification layer. This paper focuses on improving this process, in order to better transfer knowledge between CNN architectures for faster trainings in the case of fine tuning for image classification. This is achieved by combining and transferring supplementary weights, based on similarity considerations between source and target classes. The study includes a comparison between semantic and content-based similarities, and highlights increased initial performances and training speed, along with superior long term performances when limited training samples are available.

**Index Terms**— Transfer learning, fine tuning, image classification, model selection

## 1. INTRODUCTION

With the emergence of large, public and thoroughly annotated datasets [1], along with the ever increasing computing capabilities of GPUs, Deep Neural Networks, and especially CNNs, have rapidly revolutionized many computer vision tasks. Such quantities of data allow to learn visual feature extractors whose relevance and discrimination power surpasses the best hand crafted ones [2], regardless of the problem complexity or the model size. However, the time and associated cost for creating such new huge datasets, and to make new models converge over these are still a huge bottleneck in real-world use cases, so that someone with limited resources cannot reasonably compete with companies running each of their models during weeks over hundreds of GPUs (or TPUs) and gigantic datasets.

Transfer learning is a recent and still evolving approach to address this issue. It consists in reusing a model developed

for a task as a starting point for another related task. This is based on the assumption that two related tasks require some common knowledge, so that some of the knowledge associated to a task could benefit another similar one. In deep learning, this is performed by using some of the model's weights as an initialization for the training over the new task, while the usual practice is to initialize them randomly ([3]). In computer vision tasks, well trained CNN low level and mid-level layers generally detect basic shapes and textures, no matter the specificity of the task. They consequently transfer well between different computer vision problems, as shown in numerous publications ([4][5][6] [7][8][9][10]).

Transfer learning can then be distinguished in two types of applications : domain adaptation, aiming to adapt a pre-trained network to a new task, out of this work's scope, and fine-tuning, which consists in adapting the network to new target data, for the same task. In the latter case, one can extend the transfer to the whole set of weights concerning the features extraction, and just replace the output fully connected layer with one shaped for his target classes. Almost all the knowledge required to perform the task is already present in the network, and can be aligned with the target data by a small training procedure (not necessary on the entire set of weights), lighter than the one required to train a model "from scratch". Note that fine-tuning a pre-trained CNN often leads to better performances, and requires fewer data than a network trained from scratch, as most of the knowledge required for the task is already present in it ([5, 6]).

Fine tuning has become common practice, allowing faster trainings on consequently smaller datasets, and giving the opportunity for researchers and companies to develop their own systems. However, there may still be a lack of efficiency in training a new fully connected layer from scratch, and transfer learning can once again fulfill it. To further improve the transfer, we thus propose to reuse some weights of the last fully connected layer of the original model, based on similarity between source and target classes.

The contribution of this work is fivefold: First, we show that there is some important knowledge within the last layer of pre-trained DNN models which when identified and used properly can be somewhat transferred to the new model, to speed up training and benefit model accuracy for fine tuning.

Second, we propose a novel method to reuse that knowledge in combining multiple relevant source classes. Third, we conduct a study over one visual and two semantic similarity measures to select these relevant source classes. Fourth, we propose an original analysis enabling us to separate cases between three possible types of knowledge transfer, to attest that we are performing well on each of them, and optimize our process. Fifth, we monitor our method while decreasing the amount of training data, to validate the consistency of the results.

The paper is organized as follows. We will first review some works related to ours, then present our approach, before discussing the experimental results. Finally, we will present the conclusions of this work, and propose future developments.

## 2. RELATED WORKS

### *Datasets and architectures*

The most common source dataset to apply transfer learning for computer vision tasks is ImageNet [1], since it presents 1000 classes, shared between various semantic fields (animals, flora life, vehicles, tools, etc...). It is thus very likely to benefit the training of almost any kind of target data, and its efficiency is demonstrated in [11].

As for the choice of the network to use, many state of the art results in image classification tasks (including the ImageNet dataset) have been achieved by (or built on) the ResNet architecture [12, 13], and Inception [14, 15] structures (or combinations of them). Their efficiency and simplicity often places them as the best choice for transfer learning, be it for other classification tasks, or using it as a backbone for other tasks (object detection and segmentation, for example).

### *Transfer Learning process*

Reusing some pre-trained weights for a new task has been pioneered by [4] and [5], showing that the new task can greatly benefit it, not only in terms of training speed, but also of global performance.

However, the way to optimize a transfer learning process is still unclear. We know that the deeper we go in a network, the more specialized are its weights to the task they are trained on [2, 5]. Concretely, if the first few layers of a ResNet (detecting simple visual patterns like geometrical shapes) can benefit any computer vision task, it is still unclear how deep the weights can efficiently be reused. [5] experiments transfers of different depths, and highlights that the transfer of specialized layers can hurt performance on the target task, depending on their depth.

In [6] is given a study taking into account the amount of data available. They show that if transferring weights has only a moderate impact on performance in a context with a lot of data, it becomes crucial for long-term performance as the data decreases. For two sufficiently close tasks (source and target), they generally advise to transfer all the layers except the classification one before a global fine tuning. This advice seems quite reasonable in the case we are investigat-

ing, since only the images contained in the dataset and their classes change, while the classification objective remains the same.

However, with enough populated classes, [5] shows that training only the randomly initialized part of the transferred CNN can break fragile co-adaptations at the boundary. Fine tuning equally the whole network gives better results, allowing to readjust those co-adaptations. [9] proposes a finer process that consists in training the whole network in one go with a lower learning rate applied to the transferred part. This focuses the training on the new part, while allowing co-adaptation between the two parts.

As shown in [10], transfer learning can also be improved by deepening and/or widening the original network, giving more rooms to small adjustments, under the condition of correctly managing the simultaneous training of both transferred and newly created cells.

A systematic process in all these works is to discard the classifying layer and to train a new one, adapted to the target classes, from scratch. We argue and show that, when the source and target are similar to a certain extent, the knowledge contained in the pre-trained classifier layer can be efficiently reused for learning a new model.

### *Semantic similarity between textual content*

One traditional way to get a similarity measure between two concepts is to use the WordNet graph [16]: WordNet is an english lexical database of nouns, verbs, adjectives, and adverbs grouped under lexicalized concepts (named synsets), interlinked by different types of semantic relations. There are five main semantic similarity measures defined for WordNet in the literature : Jiang & Conrath [17], Leacock & Chodorow [18], Lin [19], Resnik [20], and Wu & Palmer [21]. Each of them evaluates the semantic distance between two synsets. [22] evaluated the Wu & Palmer one, making use of the path length between the synsets organized in a 'is-a' hierarchy and the depth of their most specific ancestor node, as the best one for semantic similarities.

More recently, [23] designed an approach using neural networks to project words into feature vectors named Word2Vec representations, to represent efficiently textual content. In this feature space, distances between words are shown to be quite accurate to attest and quantify some semantic relationships [23, 24, 25].

## 3. SIMILARITY-BASED KNOWLEDGE TRANSFER

A standard, basic transfer learning process to train an image classifier for some target classes is to reuse the convolution weights of a network already trained on a similar task on source classes. A fully connected layer fitting the target task is then randomly initialized on top of the network, which is fine tuned following a strategy adapted to the specificities of the problem (amount of available data, possible computational power restrictions, etc...).

The fully connected layers of the pre-trained network represents the knowledge of the task it is devised for. In cases

where the original classification problem and the destination one are close, there might be a gain in transferring some of the knowledge of the original network head (last layer) to the target one.

We hypothesize that re-adjusting this available knowledge to fit the target classes could be more efficient than creating it from scratch, in the usual way. Assume we dispose of  $M$  source classes and  $N$  target ones, each target class (represented by its fully connected weights) could be initialized with a combination of a relevant subset of those  $M$  source classes, instead of randomly. We define the relevance of such a subset of classes by using alternative similarity measures.

### 3.1. Similarity measures

We propose three of them. The first is a visual one, directly based on the image content. The two others are label-based semantic similarities :

**Inference similarity.** The images of the target classes are input to the plain source network. The similarities between source and target classes are computed as F-score measures for each "source network output/target class" couples. In a more practical way, let  $o_j$  be the  $j^{th}$  output of the source network and  $c_i$  the  $i^{th}$  target class with  $j \in \{1, \dots, M\}$  and  $i \in \{1, \dots, N\}$ , we compute  $sim(i, j)$  as the F-score for  $o_j$  discriminating  $c_i$ . This similarity measure aims at leveraging relations based more on pure visual content than semantics.

**WordNet similarity,** using the Wu & Palmer ([21]) measure, as advised in [22].

**Word2Vec similarity.** Using some pre-trained Word2Vec embeddings, we compute the standard cosine similarity between the Word2Vec embeddings of the source and target class names.

In the following section, these three initialization techniques are compared to the classic one (i.e. using random initialization of the neural network weights).

### 3.2. Initialization

We consider the affinity values as the coefficients of a neighboring structure, allowing us to approach the target class as a combination of some source neighbors. We thus compute, for each target class, the weights of its classifier as a linear combination of its  $K$  closest source neighbors with respect to the similarity measure, taking as coefficients these affinity values (normalized, for them to sum to one over  $K$ ).

$$W'_i = \sum_{j=1}^K \left( \frac{sim(i, j)}{\sum_{j=1}^K sim(i, j)} \right) W_j$$

In this way, each of the  $K$  source classes neighbors contributes to the construction of the target class initialization, in proportion to its normalized similarity. The setting of  $K$  with respect to the target classes is studied in the experimental part.

## 4. EXPERIMENTS AND RESULTS

### 4.1. Implementation Details

In the following, to combine architecture simplicity and high performances, we use a ResNet-101 pre-trained on Imagenet, and replace the last fully connected layer to fit the target classes. We train weights from the fourth block (included) to the end, and freeze the rest. One could use a finer transfer strategy to optimize the results obtained [5, 6, 26, 9]. Input images' smallest sides are resized to 256 (preserving aspect ratio), then cropped (randomly for training, center crop for testing) to output  $224 \times 224$  images.

We used Adam optimizer with a learning rate of  $10^{-3}$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  and  $\epsilon = 10^{-8}$ , with a batch size of 64. We apply a dropout of 0.75 on the last fully connected layer to prevent overfitting.

### 4.2. Dataset

For this experiment, we choose as source classes the 1000 classes of the ILSVRC challenge [1], which the ResNet-101 has been trained to classify. We take as target classes 90 ImageNet synsets that are not part of those former 1000. They can be separated in three types :

**Included classes.** The target synset is a child of a source synset, thus representing a more restrictive class than the one in the original problem.

**Inclusive classes.** The target class is an ancestor of some source synset(s), thus representing a more general class.

**Disjoint classes.** Neither child nor ancestor of any already source synset.

For these 90 target classes we select some synsets containing at least 1000 images and equally distributed into the three types of target classes (30 classes each). Within each target class, we pick 100 images for testing and 900 for training, producing balanced training and testing sets. Depending on the experiments, only a certain portion of this training set will be used for training. The list of synsets used for this experiment along with the selected images is available on a GitHub repository.<sup>1</sup>

### 4.3. Similarities and Initialization

We compute the inference similarities as explained earlier, with a pass of the training set through the pre-trained network (with the 1000 class pre-trained classifier).

For the WordNet similarities, we use those given by the WordNet module of the NLTK Python library to obtain a similarity measure based on the shortest path that connects the labels (or synsets) in the "is-a" (hypernym/hyponym) taxonomy.

<sup>1</sup><https://github.com/lucasascal/semantic-and-visual-similarities-for-efficient-knowledge-transfer-in-CNN-training>.

To compute the Word2Vec embedding of a given label, we average the embeddings<sup>2</sup> of all the words composing it, since labels are not always denoted by a single word, but often by an expression.

Each target class is then initialized with the weights of its selected source(s), depending on the chosen similarity measure and the number of source neighbors. For the random initialization baseline, we use a classic Xavier initialization [3].

#### 4.4. Neighboring optimization

We first show the global behavior of our initialization method with a single source class neighbor as initialization for each of the target class, in terms of F-score measure, averaged over the 90 target classes. For this experiment, we populated each of the 90 target classes with 500 training images (limit above which the results did not change significantly), and trained the networks over these. Fig.1 shows this evolution in terms of F-score averaged over all the 90 target classes, with the four initialization strategies (i.e Random, Inference, WordNet and Word2Vec).

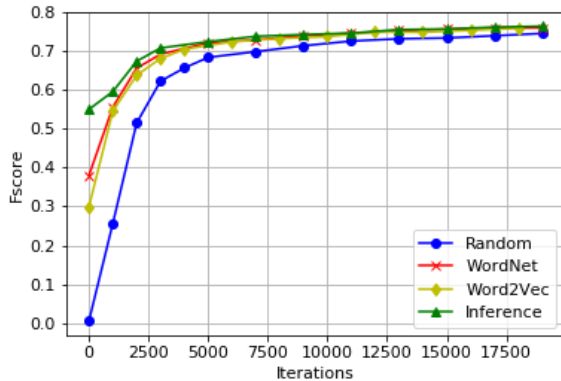


Fig. 1. averaged per-class Fscore





We observe that each of the three initialization strategies performs better than the random baseline: the convergence is accelerated, and the models produce interesting results even without training (iteration 0): from 40% to more than 70% of the F-score achieved at convergence, depending on the model. The initialization by inference similarity is performing best, as one could have expected since the similarities in this case have directly been evaluated with respect to the task’s performance metric. The four models tend to converge to the same value, provided with enough data to fill the gap.

We give in Table 1 an example of source/target class correspondence given by each similarity measure, for the “Bullet, slug” target class. The results of these transfers are shown in Fig. 2. In this case, the Word2Vec method has been mistaken by the “slug” term, and chose the mollusc as a source

class (worst performing). WordNet found a logical source class (“Projectile, missile”), according to the synsets semantic, and the inference by similarity selected “Lipstick, lip rouge”, which has no obvious semantic link with a bullet, but presents some very similar visual patterns, as shown on the images (performing best).

From this example, along with the global results, we conclude that label-based semantic similarities are more likely to select wrong matchings for visual classification, while the inference similarity is able to bring out better ones, out of any semantic consideration.

Table 1. Classes correspondances

Target Class	Inference Affinity	WordNet Affinity	Word2Vec Affinity
bullet, slug	lipstick, lip rouge	projectile, missile	slug
			

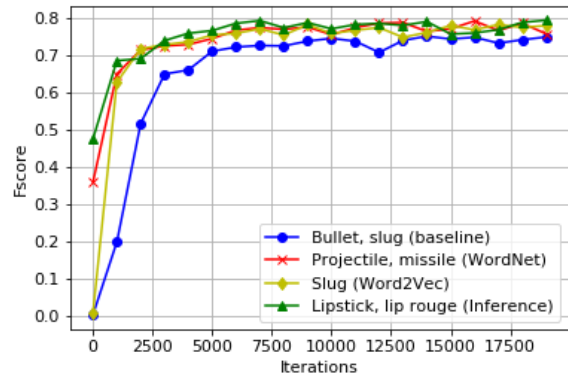


Fig. 2. Evolution of the models on the target class Bullet, slug, for the different source classes determined by the similarity measures.

We then extend the study to the use of multiple source classes neighbors to compute the different initializations: Fig. 3 shows the F-scores of the models directly after initialization (without training), with respect to the number of source class neighbors selected, for each type of target class (i.e disjoint, included and inclusive).

The initialization by inference similarity benefits the most from extending the number of source class neighbors: for any type of initialization, the built classifiers smoothly gain in performances by adding neighbors. Beyond the selection of one best source class, this confirms the superiority of the visual similarity over the semantic ones to estimate the

<sup>2</sup>as trained on Flickr, and publicly available at <https://github.com/li-xirong/hierse/blob/master/README.md>.



**Fig. 3.** Immediate inference results for each type of classes and initialization, with respect to the number of source classes selected to compute the initialization.

**Table 2.** Evolution of the four methods through data reduction. Performances after initialization (left columns) and best registered performances (right columns) are reported, with respect to the number of training samples for each class.

Images per class	Random		Visual similarity		WordNet semantic similarity		Word2Vec semantic similarity	
	First	Best	First	Best	First	Best	First	Best
100	0.00	0.72	<b>0.59</b>	<b>0.73</b>	0.39	0.72	0.35	0.72
50	0.00	0.68	<b>0.58</b>	<b>0.69</b>	0.39	0.68	0.35	0.68
25	0.00	0.62	<b>0.58</b>	<b>0.64</b>	0.39	0.63	0.35	0.63
10	0.00	0.53	<b>0.54</b>	<b>0.54</b>	0.39	0.54	0.35	0.53
5	0.00	0.41	<b>0.50</b>	<b>0.50</b>	0.39	0.43	0.35	0.45
2	0.00	0.26	<b>0.44</b>	<b>0.44</b>	0.39	0.39	0.35	0.35
1	0.00	0.16	<b>0.40</b>	<b>0.40</b>	0.39	0.39	0.35	0.35

relevance of any source class for a transfer. For the WordNet and Word2Vec cases, there is also a significant gain, even if the evolution over the number of neighbors is more chaotic, and it appears to be a good way to compensate bad matchings (like the Word2Vec case in **Table 1**).

#### 4.5. Data reduction study

We then study how well this process generalizes while decreasing the amount of training data. For each of the initialization methods, we initialize a new model, taking for each target class the optimal number of neighbors source classes depending on its type (disjoint, included or inclusive). These optimal numbers are taken from **Fig.3**. **Table 2** shows the scores of these models compared to the random baseline for 100, 50, 25, 10, 5, 2 and 1 training images per class. For each model, the initial performance after initialization (without training) and the best registered performance until convergence are reported.

The source classes selection by inference similarity varies with the number of training images (unlike the two others), since it is computed with those images. Its performance at initialization thus decreases with the amount of training data. However, it still always achieves better performances, which puts aside the idea of combining both types of similarities [27]. Under 5 training images per class, a consequent performance gap remains between the baseline and our models even after training. Under 2 images per class (5

for inference similarity), the best scores are achieved right after initialization, and training only degrades performances. Building the best possible initialization is thus crucial in such cases.

## 5. CONCLUSION AND PERSPECTIVES

This paper addresses transfer learning in an image classification context. In particular, we proposed to study alternative approaches to re-use the knowledge inherent within the original pre-trained deep network in the target one (handling new image classes). Rather than only transferring network weights corresponding to the feature extraction part, we investigated several initialization strategies to re-use and combine specifically identified weights from the pre-trained classifier into the target model. To validate the impact of our method, we presented and tested three different similarity estimators, one visual and two semantics, optimized the models across the different types of target classes, and monitored them while reducing the amount of data.

In the end, our method produced systematically better initializations, faster trainings, and significantly superior long term performances in limited training data configurations. The consistency with which the best model, based on visual similarities, outperforms the baseline across the different types of target classes and amounts of data, along with its computational lightness (a few supplementary inferences in the network) suggest that it can be systematically adopted

when performing transfer learning in this context.

## 6. REFERENCES

- [1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., “Imagenet large scale visual recognition challenge,” *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.
- [2] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012, pp. 1097–1105.
- [3] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.
- [4] G. Hinton and R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [5] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” in *NIPS*, 2014, pp. 3320–3328.
- [6] B. Chu, V. Madhavan, O. Beijbom, J. Hoffman, and T. Darrell, “Best practices for fine-tuning visual classifiers to new domains,” in *ECCV*. Springer, 2016, pp. 435–442.
- [7] H. Azizpour, A. Razavian, J. Sullivan, A. Maki, and S. Carlsson, “Factors of transferability for a generic convnet representation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 9, pp. 1790–1802, 2016.
- [8] A. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese, “Taskonomy: Disentangling task transfer learning,” in *CVPR*, 2018, pp. 3712–3722.
- [9] Y. Tamaazousti, H. Le Borgne, C. Hudelot, M. Seddik, and M. Tamaazousti, “Learning more universal representations for transfer-learning,” *arXiv preprint arXiv:1712.09708*, 2017.
- [10] Y. Wang, D. Ramanan, and M. Hebert, “Growing a brain: Fine-tuning by increasing model capacity,” in *CVPR*, 2017, pp. 2471–2480.
- [11] M. Huh, P. Agrawal, and A. Efros, “What makes imagenet good for transfer learning?,” *arXiv preprint arXiv:1608.08614*, 2016.
- [12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016, pp. 770–778.
- [13] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *ECCV*. Springer, 2016, pp. 630–645.
- [14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *CVPR*, 2015, pp. 1–9.
- [15] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4 inception-resnet and the impact of residual connections on learning,” in *AAAI*, 2017.
- [16] R. Beckwith, C. Fellbaum, D. Gross, and G. Miller, “Wordnet: A lexical database organized on psycholinguistic principles,” *Lexical acquisition: Exploiting on-line resources to build a lexicon*, pp. 211–232, 1991.
- [17] J. J. Jiang and D. Conrath, “Semantic similarity based on corpus statistics and lexical taxonomy,” *arXiv preprint cmp-lg/9709008*, 1997.
- [18] C. Leacock and M. Chodorow, “Wordnet: An electronic lexical database, chapter combining local context and wordnet similarity for word sense identification, pages 265–283,” 1998.
- [19] D. Lin et al., “An information-theoretic definition of similarity,” in *Icml*. Citeseer, 1998, vol. 98, pp. 296–304.
- [20] P. Resnik, “Using information content to evaluate semantic similarity in a taxonomy,” *arXiv preprint cmp-lg/9511007*, 1995.
- [21] Z. Wu and M. Palmer, “Verbs semantics and lexical selection,” in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1994, pp. 133–138.
- [22] M. Capelle, F. Frasincar, M. Moerland, and F. Hogenboom, “Semantics-based news recommendation,” in *Proceedings of the 2nd international conference on web intelligence, mining and semantics*. ACM, 2012, p. 27.
- [23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
- [24] H. Wang, “Introduction to word2vec and its application to find predominant word senses,” 2014.
- [25] A. Handler, “An empirical study of semantic similarity in wordnet and word2vec,” 2014.
- [26] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, “Cnn-rnn: A unified framework for multi-label image classification,” in *CVPR*, 2016, pp. 2285–2294.
- [27] B. Safadi, M. Sahuguet, and B. Huet, “When textual and visual information join forces for multimedia retrieval,” in *ICMR*, 2014.