Leveraging Deep Visual Features for Content-based Movie Recommender Systems

Ralph José Rassweiler Filho*, Jônatas Wehrmann*, and Rodrigo C. Barros†
Faculdade de Informática
Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga, 6681, 90619-900, Porto Alegre, RS, Brazil
* Email: {ralph.rassweiler, jonatas.wehrmann}@acad.pucrs.br
† Email: rodrigo.barros@pucrs.br

***Abstract*—The movie domain is one of the most common scenarios to test and evaluate recommender systems. These systems are often implemented through a collaborative filtering model, which relies exclusively on the user's feedback on items, ignoring content features. Content-based filtering models are nevertheless a potentially good strategy for recommendation, even though identifying relevant semantic representation of items is not a trivial task. Several techniques have been employed to continuously improve the content representation of items in content-based recommender systems, including low-level and high-level features, text analysis, and social tags. Recent advances on deep learning, particularly on convolutional neural networks, are paving the way for better representations to be extracted from unstructured data. In this work, our main goal is to understand whether these networks can extract sufficient semantic representation from items so we can better recommend movies in content-based recommender systems. For that, we propose DeepRecVis, a novel approach that represents items through features extracted from keyframes of the movie trailers, leveraging these features in a content-based recommender system. Experiments shows that our proposed approach outperforms systems that are based on low-level feature representations.**

I. Introduction

Traditionally, there are two main approaches for building recommender systems: collaborative filtering (CF) and content-based filtering (CBF). The former method relies strictly on feedback from users, usually in the form of ratings, working under the community-knowledge assumption: if user A gives positive feedback to item X, he will probably like unknown item Y considering that other users have given positive feedback to both items. CBF, on the other hand, makes use of available information about items to infer whether a user will like previously-unseen items. In the movie domain, the intuition of CBF indicates that a Sci-Fi movie recommendation is probably a good genre choice for a user that has given high rating to movies like *Matrix*, *Terminator*, and *Interstellar* [1].

A common technique that is used to represent items and further analyze them for recommendations is the vector space model (VSM). In the movie domain, the VSM can be designed based on high-level features (sometimes referred as metadata) such as cast, director, writer, genre, and any additional textual information that may be available like tags and user reviews [2]. Another way to create a VSM with the intention of capturing specific details about items is low-level feature extraction from multimedia sources: videos, audio, and images. Examples of low-level features extracted from images are color histograms, textures, and lightning conditions [3].

Even though low-level features seem to be useful as an additional source of information, they cannot provide direct information regarding the item's content. For instance, measuring the lighting key or color variance from a given frame will not provide any high-level information regarding the subject that is present in that frame, nor the context it is inserted in.

In the past few years, Deep Neural Networks have been responsible for several breakthroughs in Machine Learning and Computer Vision. Convolutional Neural Networks (ConvNets) [4], [5], for instance, have become the state-of-the-art for several tasks, including image classification, activity recognition, and video understanding, just to name a few. A ConvNet is capable of learning the relevant features from unstructured raw data, thus reducing the need for hand-crafted knowledge, such as low-level and mid-level feature descriptors. Encouraged by these advances, we propose a novel movie CBF recommender system that makes use of an ultra-deep residual ConvNet that learns from thousands of movie trailers in order to provide an informed recommendation to each user, namely DeepRecVis.

Recommendations can be made by treating both user and items' VSM profiles as a classification or regression problem. When treating as a classification problem, the recommender system will predict, for example, whether a user will like or dislike a given item. As a regression problem, the task can be to predict the rating a user will give for some item. In this work, we deal with the task of rating prediction, which will be further discussed in Section III-B.

Advantages of CBF recommender systems include: (i) independence of the user community, since user profiles can be built based only on ratings of each isolated user as source of information; (ii) simplified explanations of recommendations by indicating similar items rated by the user; and (iii) robustness to item cold-start, which means that novel items can be recommended without any previous rating history. Meanwhile, the challenges faced by CBF are: (i) limited content analysis; (ii) lack of serendipitous recommendations, a problem known as over-specialization; and (iii) user cold-start [6]. Despite the advantages, pure content-based recommender systems have been struggling due to the aforementioned shortcomings and by the fact that it is hard to extract meaningful attributes from items in some domains [1].

In this paper, we addressed the following questions:

1) Is it possible to obtain good semantic representations from movies in order to address the limited-content analysis problem, inherent to pure CBF models, by extracting features from trailers through a ConvNet?
2) How can we properly learn user preferences and build item profiles in this context?
3) How can we make recommendations based on items and users profiles and how can we measure the effectiveness of such an approach?
4) Is the proposed method capable of improving the recommendation performance in comparison with the extraction of low-level features?

For addressing these questions, we develop a pure CBF recommender system using three strategies: i) a low-level feature modeling representation that comprises 5 statistical measures obtained from movie trailers; ii) a deep feature based approach, namely DeepRecVis, that extracts semantic information from movie trailers through an ultra-deep residual ConvNet [7] trained over ImageNet [8] and Places 365 [9]; and iii) a hybrid approach, that concatenates low-level and semantic deep features.

To the best of our knowledge, this is the first work that analyzes a pure content-based recommender system performance using features extracted by a ConvNet to recommend items. We empirically show that the success of ConvNets in image recognition and classification tasks also translate to the recommendation domain.

The remainder of this paper is organized as follows. In Section II we review recent strategies that were proposed for CBF recommender systems and semantic representation of items. Section III discusses in detail the low-level and deep features extraction procedures. In Section IV, we present the setup of the experiments that are carried out in this paper, and the results are described and commented in Section V. Finally, we summarise the conclusions and envision topics for future work in Section VI.

II. Related Work

Feature extraction from videos and images has been used in several domains. It can be a rich source of semantic information for items, specially when other kinds of information like text and audio are scarce. Several techniques can be employed to extract features in this context, such as Scale-Invariant Feature Transform (SIFT) [10] and Speed Up Robust Features (SURF) [11].

A. Image Recommendation

In [12], photos were automatically ranked and classified for social network active members. The authors combined textual and image features that are relevant to determine their attractiveness. The selected features were brightness, saturation, saturation variation, colorfulness, naturalness, contrast, RGB contrast, sharpness, and sharpness variation. To determine an image attractiveness score and further recommend it to users, the authors implemented a regression approach that considered the users' favourite photo assignments.

Su et al. [13] proposed a real-time recommender system framework that relies on offline aesthetic modeling with the intent to learn a model of the users' preferences from a feature library. In this approach, images are represented as a bag-of-aesthetics-preserving features (BoAP), and an online aesthetic view finding process makes use of the learned models to suggest views. The BoAP features were composed by color, texture, saliency, and edge information.

In the fashion domain, the work in [14] used full-body photographs from fashion magazines to recommend clothes to users. The visual features were extracted from regions that represent points of user attention on the photos. Visual and textual information for the same domain were reported in [15] and [16]. Color histograms and skin descriptor were employed in the former, while the latter experimented with six different feature extraction techniques: SIFT, SURF, Binary Robust Independent Elementary Features (BRIEF), Binary Robust Invariant Scalable KeyPoints (BRISK), Oriented FAST, and Rotated BRIEF (ORB) Fast Retina KeyPoint (FREAK).

For the tasks of image recommendation and the application of images in recommender systems, the work in [17]–[19] also explored low-level features extraction, usually by employing traditional Bag of Features (BoF) based on k-means clustering. Similarly, the authors in [20] treated the problem of text advertisement generation for photographs with no textual information.

Geng et al. [21] transformed an image-based social network (Pinterest) into low-dimensional representations. To learn such representations, the authors applied a deep learning framework that takes into consideration the users' preferences. The framework was built based on the AlexNet ConvNet [22]. Results indicate that the deep learning representation improved the recommendation accuracy in comparison to traditional content-based and collaborative filtering approaches.

B. Video Recommendation

Multimedia information was explored for the movie and video domains in [23], [24], where content was represented as a combination of audio, textual, and video low-level features. In [25], keyframes from live TV programs were the target of Color and Edge Directivity Descriptors (CEDD) for feature extraction. Video recommendation was performed by aggregating the visual features with user profiles.

C. Movie Recommendation

For movie recommendation, the method of Deldjoo et al. [3], [26], [27] relied on the use of low-level stylistic visual features based on perceptions of media aesthetic. These features include average shot length, lightning key, color variation, and motion content, described as follows.

For calculating the average shot length, one must detect when a novel shot happens within a video. A standard approach, following [3], [28], is to compare every frame to its adjacent neighbor. A low inter-frame similarity usually indicates an exchange of scenes. Such similarity is often computed via histogram intersection $s(i)$ in the HSV (Hue, Saturation, Value) color space. Such an approach works well for detecting abrupt scene changes, though the algorithm fails when soft transitions occur. For fixing that issue, we can iteratively smooth $s(i)$ with a Gaussian kernel, and the *local minima* of the smoothed function $s(i)$ will thus depict a shot boundary. Often in movie understanding tasks, each scene is represented by a single static frame known as the *keyframe*, which is the central frame from the scene.

There are two main lighting categories: high-key lighting and low-key lighting. The first one concerns the brighter color levels and less contrast between dark and light. In the latter, usually darker tones are predominant and there is a high contrast ratio. The lighting key feature for frame i is computed as $\zeta_i = \mu_i \cdot \sigma_i$. Such features is better extracted from the HSV color space by computing the mean (μ) and standard deviation (σ) of the pixel values. A frame with high-key lighting is a consequence of high μ and σ values. Conversely, a low-key frame is a consequence of low values from both μ and σ.

Color variance is calculated by converting the keyframes into the CIE *Luv* space. Then, a covariance matrix is generated (Equation 1), and the overall video color variance is given by the determinant of that matrix.

$$p_{cov} = \begin{bmatrix} \sigma_L^2 & \sigma_{Lu}^2 & \sigma_{Lv}^2 \\ \sigma_{Lu}^2 & \sigma_u^2 & \sigma_{uv}^2 \\ \sigma_{Lv}^2 & \sigma_{uv}^2 & \sigma_v^2 \end{bmatrix} \tag{1}$$

Finally, the motion content feature (Γ) represents the action in a movie, i.e., the amount of active pixels with time. This analysis must be done for every scene/shot with all frames. Given the x, y spatial dimensions, and the temporal dimension t, one can compute Γ following Equation 2.

$$\Gamma = \begin{bmatrix} J_{xx} & J_{xt} \\ J_{xt} & J_{tt} \end{bmatrix} = \begin{bmatrix} \sum_w H_x^2 & \sum_w H_x H_t \\ \sum_w H_x H_t & \sum_w H_t^2 \end{bmatrix} \tag{2}$$

$$\theta = \frac{1}{2} tan^{-1} \frac{2J_{xt}}{J_{xx} - J_{tt}} \tag{3}$$

For building a movie representation, the work described in [3], [27] used a 5-dimensional feature vector, given by:

$$f_v = (\bar{L}_{sh}, \mu_{cv}, \mu_{\bar{m}}, \mu_{\sigma_m^2}, \mu_{lk}) \tag{4}$$

where $\bar{L}_{sh}$ is the average shot length, μ_{cv} is the average of color variance, $\mu_{\bar{m}}$ and $\mu_{\sigma_m^2}$ are the average of both average/standard deviation of motion across all frames, and μ_{lk} is the average lightning key over keyframes.

III. Method

We propose a novel content-based movie recommender system called DeepRecVis, which is a method that provides recommendation based on deep semantic features extracted by a ConvNet. Since full movies are neither publicly available nor easy to process due to the large computational resources required, we decided to use movie trailers as the visual information source. Trailers often present a summary of the movie's main plot along with the most important features regarding its content. Speaking in high-level terms, our method is an automated system that *watches* all movies in a movie-trailer database and automatically learns the user preferences, providing proper recommendations.

A high-level architecture of a CBF recommender system comprises [6]: i) a content analyzer, where the representation of the items are preprocessed, and where the feature extraction technique is applied; ii) a profile learner, where the user profile is built based on what the user liked and disliked in the past; and iii) a filtering component, where the user profile is exploited in order to provide relevant recommendations. The high-level architecture of DeepRecVis is depicted in Figure 1.

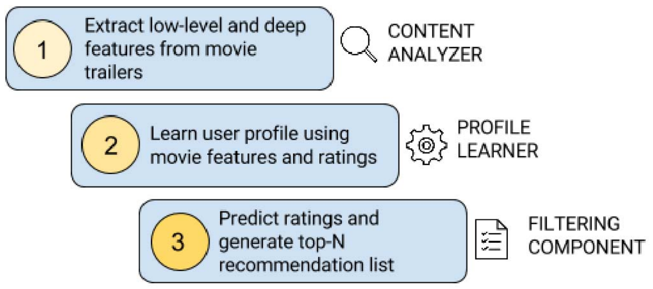
1
Extract low-level and deep features from movie trailers
CONTENT ANALYZER
2
Learn user profile using movie features and ratings
PROFILE LEARNER
3
Predict ratings and generate top-N recommendation list
FILTERING COMPONENT

Fig. 1. DeepRecVis's high-level architecture.

A. Video Representation

Considering the success of ConvNets for video understanding tasks, we decided to investigate their performance for movie recommendation. Our process for movie-trailer feature extraction is based on [29], with the following differences: i) we employ a much deeper residual convolutional network (with 152-layers); ii) the ConvNet is pre-trained in two large datasets, namely ImageNet and Places-365, rather than on the movie-trailer LMTD-4 dataset; iii) only keyframes were used for feature extraction; iv) we perform a slightly different unsupervised training phase than in [29]; and v) we employ a 10-crop strategy to obtain better feature vectors.

Several studies make use of ConvNets pre-trained over ImageNet as off-the-shelf feature extractors [30]. However, ImageNet is an object-centric dataset that makes the ConvNet invariant to the images' background. We do believe that background information, which contain places and environments details, play an important role for movie trailer understanding. Hence, our ConvNet is trained over a conjunction of both ImageNet and Places-365 datasets for improving the network's ability of learning objects and environments at the same time. This conjunction comprises roughly 3 million images (1.2 million from ImageNet and 1.8 million from Places-365) from 1365 classes (1000 from ImageNet and 365 from Places-365).

For reducing the computational time needed for DeepRecVis, we decided to extract the deep features only from the movie trailers' keyframes. Notice that LMTD comprises

≈ 30 million frames and ≈ 1 million keyframes. A standard approach for ConvNet-based feature extraction is using only the center crop as a representation for the whole image. However, most of the movie trailers are in wide-screen aspect ratio, and a central crop could ignore important parts of the frame. Hence, we decided to use a 10-crop evaluation strategy to provide more descriptive vectors. The 10 vectors are averaged for building the ultimate frame vector representation, which can also be considered a scene representation since we extract a single frame keyframe per scene.

In order to combine the extracted information for the whole movie trailers, we employ an unsupervised algorithm for finding natural scene categories, similarly to [29]. Let $\mathcal{S}_j \in \{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, ..., \mathbf{s}_n\}$ be the scene-based visual features of the j^{th} movie trailer that comprises n scenes. We find scene categories by employing the *k*-means algorithm in a random sample of 100,000 feature vectors $\mathbf{s}_i \in \mathbb{R}^{1\times 2048}$. We run *k*-means 10 times to find a proper initialization. This unsupervised training provides k centroids $\mathbf{c} \in \mathbb{R}^{1\times 2048}$ used for building a semantic histogram. To build these histograms, each movie scene $\mathbf{s}_i$ is assigned to its closest centroid $\mathbf{c}_l$ in the Euclidean space. Hence, a movie trailer $\mathcal{T}_j$ is thus represented by an integer vector given by $\mathbf{h}_j \in \mathbb{I}^{1\times k}$, which is normalized to convey the bins' relative frequency via $\mathbf{h}_j = \frac{\mathbf{h}_j}{\sum_i^k \mathbf{h}_{ji}}$.

Figure 2 depicts examples of clusters found via *k*-means ($k = 128$) in the sample of 100,000 movie scenes. Note that some clusters are specialized in specific concepts (guns, explosions, faces) while others are sensitive to environments (landscapes, dark places, etc). Examples of the semantics conveyed by each cluster are as follows: i) cluster 14 is specialized in gun-related scenes, among other similar concepts; ii) cluster 34 unites frames with fire-based elements; iii) cluster 89 seems to pay attention to faces; iv) cluster 99 groups opening and ending credit scenes; and v) cluster 112 presents frames with visuals that point to the Sci-Fi style. Recall that all of these clusters were generated automatically in an unsupervised fashion, i.e., we did not train nor *fine-tuned* the ConvNet for finding these concepts.

To provide some reassurance regarding the content-based similarity measure that we employed (the cosine similarity), we have chosen a small set of movies for querying the most similar ones within the LMTD dataset. For instance, we have chosen the movie *Toy Story* (1994) and queried for the most similar ones (based on the cosine similarity) in the dataset. We did expect a high similarity between the chosen movie and its sequels: *Toy Story 2* (1999) and *Toy Story 3* (2010). Figure 3 demonstrates the retrieved content. Both rows contain the most similar movies to *Toy Story*, with the ranking position identified on the bottom of each movie poster. It is important to recall that there are 5,500 distinct movies to compare with. Hence, we do believe that our method presents much more reliable results than when using low-level visual features.

B. Recommender System

In this section we describe our content-based recommender system implementation, depicted in Algorithm 1. Let $\mathcal{U} \in \{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, ..., \mathbf{u}_n\}$ be the set that contains all users in the dataset. For each user $u_i \in \mathcal{U}$ we create a set $\mathcal{R}_i \in \{\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, ..., \mathbf{r}_u\}$ of randomly-selected unseen items by u_i. Also, for each user $u_i \in \mathcal{U}$ we create a test set $\mathcal{T}_i \in \{\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, ..., \mathbf{t}_s\}$ of items that user u_i has given high ratings. We perform rating prediction based on [31], which is depicted in Equation 5 for every item in $\mathcal{R}_i$ and $\mathcal{T}_i$. Then, we combine both $\mathcal{R}_i$ and $\mathcal{T}_i$ sets and sort it in descending order of predicted rating. Let this combined set be $\mathcal{F}_i$. Finally, every time an item that belongs to $\mathcal{T}_i$ appears in a top-N sub-set of $\mathcal{F}_i$, we count it as a *hit*.

Algorithm 1: Content-based recommender system.

```
Data: $Users$: the full set of Users
Result: Total of hits: items of the test set in the top-N list
begin
    for $u_i \in \mathcal{U}$ do
        $hits \longleftarrow 0$
        $\mathcal{R}_i \longleftarrow GetRandomMovies(u_i)$
        /* movies that $u_i$ rated as $> 4$ */
        $\mathcal{T}_i \longleftarrow GetTestMovies(u_i)$
        $predTest \longleftarrow GetPredictions(\mathcal{T}_i)$
        $predRandom \longleftarrow GetPredictions(\mathcal{R}_i)$
        $predList \longleftarrow Concat(predTest, predRandom)$
        $predList.sortDescending()$
        while $k \leq N$ do
            if $predList[k] \in \mathcal{T}_i$ then
                $hits \longleftarrow hits + 1$
            $k \longleftarrow k + 1$
```

TABLE I
NOTATION FOR THE EQUATIONS.

Variable	Description
$b_{u,i}$	Rating baseline for user u, item i
μ	Average of all ratings
b_u	User baseline rating
b_i	Item baseline rating
I_u	Items rated by user u
$r_{u,i}$	Rating for item i given by user u
U_i	Users that rated item i
$p_{u,i}$	Rating prediction for user u, item i
$s(i,j)$	Cosine similarity between items i and j
f	Feature vector

$$p_{u,i} = \frac{\sum_{j\in I_u} s(i,j)(r_{u,j} - b_{u,i})}{\sum_{j\in S} |s(i,j)|} + b_{u,i} \quad (5)$$

We defined that an item must have a user rating > 4 to be included on the test set $\mathcal{T}_i$ (Movielens ratings are in [0.5, 5] scale). We fixed the random movie set $\mathcal{R}_i$ size equal to 100 and varied the top-N sub-list size from 1 to 15. Equation 6 represents a baseline rating prediction considering the average over all ratings as well as the user baseline b_u (Equation 7) and the item baseline b_i (Equation 8).

$$b_{u,i} = \mu + b_u + b_i \quad (6)$$

$$b_u = \frac{1}{|I_u|} \sum_{i\in I_u} (r_{u,i} - \mu) \quad (7)$$

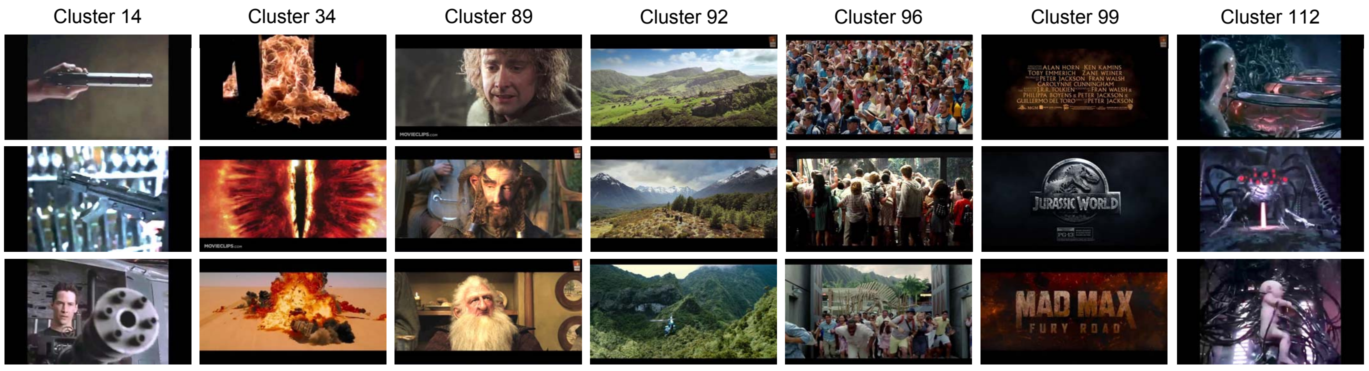

Fig. 2. Clusters of keyframes.

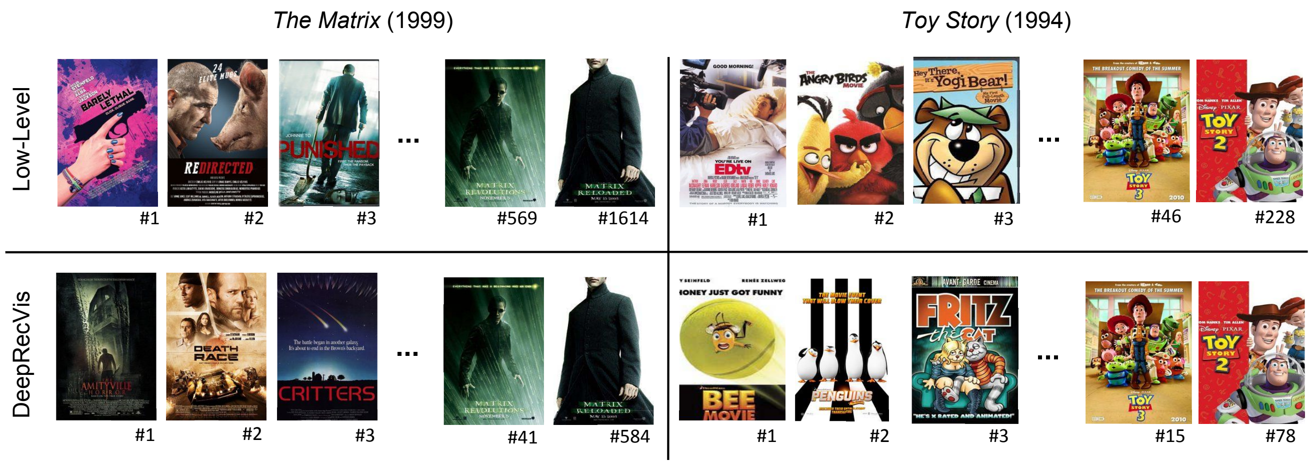

Fig. 3. Movie retrieval based on the cosine similarity. The first row contains low-level features similarities whereas the second row contains deep features similarities. The first column contains movies similar to *The Matrix* (1999). The first row, from left to right: *Barely Lethal* (2015), *Redirected* (2014) and *Punished* (2011). The second row, from left to right: *The Amityville Horror* (2005), *Death Race* (2008) and *Critters* (1986). The last two movies of each row in the first column are *The Matrix Revolutions* (2003) and *The Matrix Reloaded* (2003). The second column contains movies similar to *Toy Story* (1994). The first row, from left to right: *EdTv* (1999), *The Angry Birds Movies* (2016) and *Hey There, It's Yogi Bear* (1964). In the second row, from left to right: *Bee Movie* (2007), *Penguins of Madagascar* (2014) and *Fritz the Cat* (1972). The last two movies of each row in the second column are *Toy Story 2* (1999) and *Toy Story 3* (2010).

$$b_i = \frac{1}{|U_i|} \sum_{u \in U_i} (r_{u,i} - b_u - \mu) \tag{8}$$

Rating prediction for a given item i and user u, as depicted in Equation 5, is calculated by summing the product of the difference between the user rating for each item ($r_{u,j}$) and the user-item baseline ($b_{u,i}$) with the similarity of each item j with item i ($s(i,j)$) over the sum of the absolute values of the similarities $s(i,j)$. The user ratings item-set to be considered in this equation could be a limited neighbourhood of top similar items [31]. However, in our experiments, we found that allowing only positive similarities in the numerator leads to better results. The cosine is used to compute item similarities (Equation 9).

$$s(i,j) = \frac{f_i \cdot f_j}{\|f_i\|_2 \, \|f_j\|_2} \tag{9}$$

IV. Experimental Setup

In this section, we present the experimental methodology that we employ for evaluating DeepRecVis's performance. We describe the baseline algorithms that are compared with DeepRecVis in Section IV-A, the dataset of movie trailers in Section IV-B, and the evaluation measures that assess the quality of the recommender systems in Section IV-C.

A. Baseline Algorithms

For validating our hypothesis of generating better movie recommendations, we compare our novel approach with the work described in [3], which employs low-level features (LLF) to perform such recommendation. The LLF approach makes use of a 5-dimensional vector for providing low-level information regarding movie trailers. We also compare the performance of DeepRecVis with a random recommender, which *predicts* the user ratings by sampling a continuous value from a Gaussian distribution. Features within all methods were normalized to have zero mean and unit variance.

B. Dataset

To evaluate our solution we have used 9,408 movie trailers from the LMTD dataset [29], [32] and joined them by intersection with the MovieLens 20M dataset [33] for aligning movie trailers, available reviews, and metadata (e.g., plot, genre, year, etc). Table II depicts some statistics of the subset used in our experiments. Note that the recommender system can choose within 3,233 movies based on $\approx$ 6.8M reviews from $\approx$ 85k distinct users.

TABLE II
DATASET.

Data	$\#Values$
Movies with trailers	3,233
Users	85,040
Ratings	6,849,872
Average ratings per user	80.54
Average ratings per movie	2,118.73

C. Evaluation Measures

To evaluate the performance of our recommender system, we follow the guidelines of previous work [34]. Measures such as the mean absolute error (MAE), root mean-squared error (RMSE), and normalized mean absolute error (NMAE) have been widely used to evaluate recommender systems. They are useful for evaluating the prediction component of the systems. One drawback of these measures is that they treat errors equally for every item. Considering that items with little relevance for a user are unlikely to impact the recommendation performance, we choose to additionally employ two popular decision-support metrics intended to evaluate classification accuracy, namely precision and recall. For recommender systems, recall measures the ratio of relevant top-N retrieved items (true positives) and all other relevant but not retrieved items (false negatives + true positives). Precision measures the ratio of true positives and all other retrieved items (true positives + false positives). Whereas precision represents the probability a selected item has of being relevant, recall represents the probability that a relevant item has of being selected [35]. Equation 10 describes the MAE measure:

$$MAE = \frac{1}{N} \sum_{u,i} |p_{u,i} - r_{u,i}| \tag{10}$$

where N is the number of predicted ratings, $p_{u,i}$ is the predicted rating for user u and item i, and $r_{u,i}$ is the actual rating of user u and item i.

Equations 11 and 12 present the recall and precision measures:

$$recall = \frac{|hitSet|}{|testSet|} \tag{11}$$

$$precision = \frac{|hitSet|}{|recSet|} \tag{12}$$

where $hitSet$ is the number of relevant items for each user that is retrieved in a $top\text{-}N$ list, $testSet$ is the set of all relevant items for each user, and $recSet$ is the set of retrieved recommendations.

To further analyze the quality of our recommendation solution, we decided to employ one additional measure, which is intra-list similarity [36]. This metric is used to measure the diversity of a recommendation list. Diversity can increase the chance of novel and serendipitous recommendations. A novel recommendation is the one that takes into account any given aspect of the user's preferences, and hence can please the user. For instance, a movie that is starred by an actor or actress that the user likes can be a novel recommendation. Serendipity refers to the act of positively surprising the user with recommendations. Serendipitous recommendations are, by definition, novel. The intra-list similarity is described in Equation 13:

$$ILS_u = \frac{\sum_{i \in recSet_u} \sum_{j \in recSet_u, i \neq j} sim(i,j)}{2} \tag{13}$$

where $recSet_u$ is the full recommendation list for user u and $s(i,j)$ is given by Equation 9.

V. Experiments and Discussion

In this section, we present the experimental results when comparing the performance of DeepRecVis, LLF, a hybrid of deep and LLF, and a random baseline. Table III presents the average and standard deviation values for MAE, recall, precision, and diversity for all users in the dataset. Since MAE is a prediction that is independent of iterations, only a single value is presented. The difference of DeepRecVis to the low-level approach regarding MAE is ≈ 0.13, which means that DeepRecVis obtained $\approx 15\%$ of error reduction. The same behavior occurs to the remaining measures, with DeepRecVis substantially outperforming both random and low-level approaches. Note that the lower the value of diversity, the better: lower values indicate more diversity. Moreover, we can observe that hybridizing deep and low-level features gives a negligible improvement over using deep features alone.

TABLE III
AVERAGE AND STANDARD DEVIATION VALUES.

Metric	Hybrid	DeepRecVis	Low-Level	Random
MAE	**0.92**	**0.92**	1.06	2.10
Recall	**0.63** $\pm$ 0.16	0.62 $\pm$ 0.16	0.47 $\pm$ 0.14	0.08 $\pm$ 0.04
Precision	**0.63** $\pm$ 0.08	**0.63** $\pm$ 0.07	0.51 $\pm$ 0.08	0.12 $\pm$ 0.00
Diversity	**2.25 $\pm$ 1.68**	**2.25 $\pm$ 1.68**	11.35 $\pm$ 8.45	—

Figure 4 illustrates the recall over iterations achieved by each approach. It is possible to see that the more items are retrieved on the top-N list, the larger the difference between low-level and DeepRecVis. Also, as previously stated, there is a very small gain by hybridizing low-level and DeepRecVis: $\approx 0.5\%$. This occurs possibly due to the size of the vectors in the low-level feature approach and DeepRecVis's (5 and 128, respectively). The average difference between low-level and DeepRecVis is around 25%.

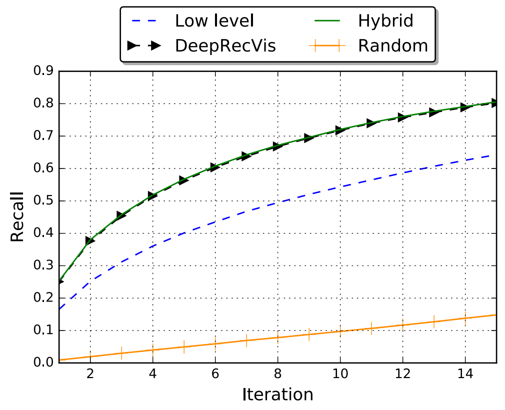

Fig. 4. Recall over iterations.

Precision over iterations is illustrated in Figure 5. We can once again observe that DeepRecViscomfortably outperforms the low-level approach regardless of the iterations. Once again we can see that the difference between DeepRecVis and the hybridization of the two approaches is negligible. On average, the difference in precision of the low-level approach and DeepRecVis is $\approx 11\%$.

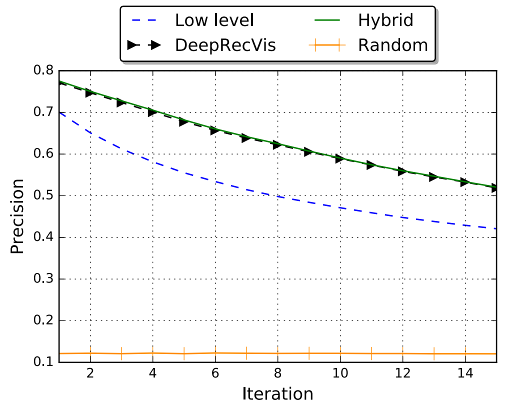

Fig. 5. Precision over iterations.

Finally, we measure the diversity of the recommendations. Considering that we have to use item similarities for computing this metric, it is not possible to measure a random recommender. The hybrid approach is once again very similar to DeepRecVis, hence, we did not include it in Figure 6. The lower the diversity score, the more diverse is the recommendation list. The intent with allowing diversity in a recommendation list is to capture the interest of the users by reducing obvious recommendations and increasing the chance of novel and serendipitous discoveries. We can see in Figure 6 that the recommendations made by DeepRecVis are much more diverse than LLF.

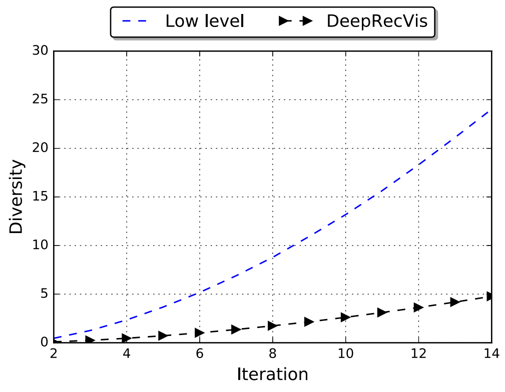

Fig. 6. Diversity over iterations. Smaller values indicate more diverse recommendations.

According to our findings, we confirm our hypothesis that representing items through deep features extracted by ConvNets lead to significantly better recommendations than low-level features in the movies domain. Due to the power of ConvNets in extracting semantics from raw unstructured data, we can build strategies that can properly leverage visual features as an additional source of information for content-based recommender systems.

VI. Conclusions and Future Work

Properly representing the content of items in recommender systems is a difficult task, specially due to the lack of annotated information from sources. Tags can be a good source, but they heavily depend on user interaction, which means that they are often not available for use in automated systems. In this paper, we presented a pure content-based movie recommender system, namely DeepRecVis, and we compared its performance with recommender systems based on low-level visual features, and also with a hybrid approach that combines both strategies to represent items. Early item similarity analyses indicated that the use of ConvNets for generating features from movie trailers yield a robust semantic representation of the movies.

We performed an empirical analysis in which we evaluate three vital aspects of recommender systems: accuracy, decision support, and diversity. We first show that in a traditional rating-prediction method, features extracted from ConvNets are clearly superior than LLF regarding the well-established MAE, precision, and recall metrics. Moreover, we also showed that deep-learning based features obtained a much more diverse recommendation list, which indicates its superiority in recommending novel and serendipitous items.

We conclude that the use of ConvNets to build a bag-of-features vector space model is indeed more suitable for content-based recommender systems than simpler low-level features, which is our main contribution in this paper. The concatenation of both deep and low-level features produces a very small gain in the overall performance.

As future work, we intend to analyze further sources of information from movies, such as the audio of the trailers.

Furthermore, we aim to explore textual data to enhance the content-based recommender approach. Other possible extensions of our work include: i) building a hybrid recommender system that also considers the collaborative filtering strategy; ii) building a user-based recommender system using a bag-of-features representation for user profiles; iii) applying dimensionality reduction techniques such as singular value decomposition and probabilistic models; iv) testing DeepRecVis's performance in different domains. Finally, we also plan to further understand the quality of the proposed content-based recommendation system by conducting a qualitative online study with users that can indicate the usefulness of the proposed approach over time.

VII. Acknowledgments

The authors would like to thank FAPERGS, CAPES, CNPq, and Google for funding this research.

References

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.

[2] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The adaptive web*. Springer, 2007, pp. 325–341.

[3] Y. Deldjoo, M. Elahi, M. Quadrana, and P. Cremonesi, "Toward building a content-based video recommendation system based on low-level features," in *International Conference on Electronic Commerce and Web Technologies*. Springer, 2015, pp. 45–56.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[6] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender systems handbook*. Springer, 2011, pp. 73–105.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[8] 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 (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[9] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in neural information processing systems*, 2014, pp. 487–495.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[11] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[12] J. San Pedro and S. Siersdorfer, "Ranking and classifying attractiveness of photos in folksonomies," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 771–780.

[13] H.-H. Su, T.-W. Chen, C.-C. Kao, W. H. Hsu, and S.-Y. Chien, "Preference-aware view recommendation system for scenic photos based on bag-of-aesthetics-preserving features," *IEEE Transactions on Multimedia*, vol. 14, no. 3, pp. 833–843, 2012.

[14] T. Iwata, S. Wanatabe, and H. Sawada, "Fashion coordinates recommender system using photographs from fashion magazines," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 3. Citeseer, 2011, p. 2262.

[15] E. V. de Melo, E. A. Nogueira, and D. Guliato, "Content-based filtering enhanced by human visual attention applied to clothing recommendation," in *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*. IEEE, 2015, pp. 644–651.

[16] K. Nogueira, A. A. Veloso, and J. A. dos Santos, "Pointwise and pairwise clothing annotation: combining features from social media," *Multimedia Tools and Applications*, vol. 75, no. 7, pp. 4083–4113, 2016.

[17] Q. Bo and J. Peng, "A novel interactive image recommendation system," in *Information Technology and Applications (IFITA), 2010 International Forum on*, vol. 2. IEEE, 2010, pp. 248–251.

[18] Y. Li, T. Mei, Y. Cong, and J. Luo, "User-curated image collections: Modeling and recommendation," in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 591–600.

[19] P. C. Ng, J. She, M. Cheung, and A. Cebulla, "An images-textual hybrid recommender system for vacation rental."

[20] W. Zhang, L. Tian, X. Sun, H. Wang, and Y. Yu, "A semantic approach to recommending text advertisements for images," in *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 2012, pp. 179–186.

[21] X. Geng, H. Zhang, J. Bian, and T.-S. Chua, "Learning image and user features for recommendation in social networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4274–4282.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[23] W. Qu, K.-S. Song, Y.-F. Zhang, S. Feng, D.-L. Wang, and G. Yu, "A novel approach based on multi-view content analysis and semi-supervised enrichment for movie recommendation," *Journal of Computer Science and Technology*, vol. 28, no. 5, pp. 776–787, 2013.

[24] T. Lehinevych, N. Kokkinis-Ntrenis, G. Siantikos, A. S. Dogruöz, T. Giannakopoulos, and S. Konstantopoulos, "Discovering similarities for content-based recommendation and browsing in multimedia collections," in *Signal-Image Technology and Internet-Based Systems (SITIS), 2014 Tenth International Conference on*. IEEE, 2014, pp. 237–243.

[25] S. Gao, D. Zhang, H. Zhang, C. Huang, Y. Zhang, J. Liao, and J. Guo, "Veclp: A realtime video recommendation system for live tv programs." in *AAAI*, 2015, pp. 4274–4275.

[26] Y. Deldjoo, M. Elahi, P. Cremonesi, F. Garzotto, P. Piazzolla, and M. Quadrana, "Content-Based Video Recommendation System Based on Stylistic Visual Features," *Journal on Data Semantics*, vol. 5, no. 2, pp. 99–113, Feb. 2016. [Online]. Available: http://link.springer.com/article/10.1007/s13740-016-0060-9

[27] Y. Deldjoo, M. Elahi, and P. Cremonesi, "Using Visual Features and Latent Factors for Movie Recommendation," *CBRecSys 2016*, p. 15, 2016.

[28] Z. Rasheed, Y. Sheikh, and M. Shah, "On the use of computable features for film classification," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 1, pp. 52–64, 2005.

[29] G. Simões, J. Wehrmann, R. C. Barros, and D. D. Ruiz, "Movie Genre Classification with Convolutional Neural Networks," in *International Joint Conference on Neural Networks*. IEEE, 2016, p. 8.

[30] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 806–813.

[31] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative filtering recommender systems," *Foundations and Trends in Human-Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011.

[32] J. Wehrmann, R. C. Barros, G. Simões, T. S. Paula, and D. D. Ruiz, "(Deep) Learning from Frames," in *Brazilian Conference on Intelligent Systems*.

[33] F. M. Harper and J. A. Konstan, "The MovieLens Datasets: History and Context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 19:1–19:19, Dec. 2015. [Online]. Available: http://doi.acm.org/10.1145/2827872

[34] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of Recommender Algorithms on Top-n Recommendation Tasks," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. New York, NY, USA: ACM, 2010, pp. 39–46. [Online]. Available: http://doi.acm.org/10.1145/1864708.1864721

[35] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5–53, 2004.

[36] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proceedings of the 14th international conference on World Wide Web*. ACM, 2005, pp. 22–32.