PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

DESIGN OF A FRAMEWORK TO BUILD EXPLAINABLE RECOMMENDATION SYSTEMS USING VISUAL CONCEPTS

ANTONIO OSSA GUERRA

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Advisor:
DENIS PARRA
HANS LÖBEL

Santiago de Chile, July 2021

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

DESIGN OF A FRAMEWORK TO BUILD EXPLAINABLE RECOMMENDATION SYSTEMS USING VISUAL CONCEPTS

ANTONIO OSSA GUERRA

Members of the Committee:
DENIS PARRA
HANS LÖBEL
JORGE BAIER
MARCELO MENDOZA
DAVID WATTS

DocuSigned by:
0F3E21932A2E493...

DocuSigned by:
Hans Lobel
892460F6E47C4A7...

DocuSigned by:
71FFF332C40C4D0...

DocuSigned by:
Marcelo Mendoza
F6F93022660646A...

DocuSigned by:
David Watts
0D9AE218682D480...

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Santiago de Chile, July 2021

Gratefully to my family.

ACKNOWLEDGEMENTS

I would first like to thank my family, especially my mother, sister, and niece. You have always loved me, supported me, and motivated me to be a better person each day. Thanks for all the sacrifices made to help me be here today. I wouldn't be able to complete this adventure without you. I love you. You're always in my mind and heart.

I would also like to thank my thesis advisors Denis Parra and Hans Löbel of the Department of Computer Science at Pontificia Universidad Católica de Chile. Both consistently steered me in the right direction and kept me motivated whenever I needed it. Thank you for your advice, your patience, your expertise, and your mentoring in this process. You're great advisors, teachers, and mentors, and also good persons.

I want to thank many people, more than I could remember now, but here I go with my attempt:

First, thanks to Valeria Acevedo, who supported me, encouraged me and helped me from our first year studying to grow as a student, friend, and person. Thank you for keep making me a better person.

Vicente Dominguez and Manuel Cartagena, thanks for your expertise, knowledge, and company in this adventure. I always counted on you, and you never disappointed me. Your advice and conversations are, and always will be, appreciated.

Coordis 2019 (Cami, Rudy, Andrés, Pauli, Ferni, Martín, Pipe, Seba), I have learned a lot about life and friendship with you, knowing you has made me a better person, and I'll always treasure your friendship. Also, thanks to the whole Cuerpo de Tutores for my first year (thanks, Vale) and when I was a tutor. But, to be honest, I will consider myself a tutor forever.

Thanks to the Advanced Programming course team. Being a teacher has been a great experience, challenging but rewarding. It is beautiful to see how a course evolves when love and care are involved in its design. Thanks to Dani C., Dani P., Fran, Ian, Diego, Benja, Tomás, Dante, Enzo, Nacho, Fernando, Cristian, Vicente, Cote, Joaquín, and all the TAs for being part of this. Also, thanks to all of my students, who allowed me to become a better teacher and give them a better learning experience than the one I had.

Members of HAIVis, IAlab, and XAI@IMFD, it's amazing how I can't stop learning from you. Both groups are different, but both are great communities. Knowledge is power when it is shared, and you are very good at doing it. I'm happy to be part of you, and I'll be there to support whoever needs it.

To all the friends I made and people I met while studying: thank you. I promise that I value even our small interactions.

Finally, I would like to thank all those small decisions of different people that led me to be here today. It has been a challenging, complex, but beautiful experience until now, and I'm ready to continue exploring this life. Thanks to everybody, to everything, and me for completing this milestone today.

Antonio.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

An important application of artificial intelligence is recommender systems, models that try to predict people's preferences, usually in a personalized manner. In this context, explanations are very valuable due to the known benefits in satisfaction, trustworthiness, and scrutability. Nowadays, visually-aware recommendation systems are commonly trained using visual latent features extracted from pre-trained deep neural networks. This approach has shown to be performant but lacks interpretability due to the inability to deliver explanations, from both user and model. In this work, we propose a framework to develop explainable recommendation systems by creating a concept-based item representation, transforming existing model architectures into explainable models. This representation is interpretable and can be read as tabular data. We propose an algorithm to create a concept-based item representation, and then create a concept embedding to train DNN models. Then, using feature attribution methods we can deliver explanations for any outcome, transforming a "black box" system into an explainable system. Our results show that visually-aware recommendation systems trained using our concept embedding perform as well as a system trained on latent features. Also, we were able to deliver explanations in terms of such visual concepts due to the interpretable nature of the input. Our research informs the development of a new explainable recommendation approach, based on an interpretable concept-based representation, that does not require the development of new model architectures.

Keywords: recommendation systems, explainable AI.

RESUMEN

Una aplicación importante de inteligencia artificial son los sistemas recomendadores, modelos que intentan predecir las preferencias de las personas, usualmente de manera personalizada. En este contexto, las explicaciones son muy valiosas debido a los conocidos beneficios en satisfacción, integridad, y escrutabilidad. Hoy en día, los sistemas de recomendación visuales son entrenados usando descriptores latentes extraídos con una red de aprendizaje profundo pre-entrenada. Esta solución ha mostrado tener gran desempeño, pero no es interpretable debido a que no pueden generarse explicaciones, tanto para el usuario como para el modelo. En este trabajo, proponemos un framework para desarrollar sistemas de recomendación explicables creando una representación de ítems basada en conceptos, transformando modelos de arquitecturas existentes en modelos explicables. Esta representación es interpretable y puede ser leída como información tabular. Proponemos un algoritmo para crear una representación de ítems basada en conceptos, y luego crear un embedding de conceptos para entrenar modelos de redes de aprendizaje profundo. Luego, usando métodos de atribución de características podemos entregar explicaciones para cualquier salida de un modelo, transformando un sistema de "caja negra" en un sistema explicable. Nuestros resultados muestran que los sistemas de recomendación visuales entrenados usando nuestro embedding de conceptos tienen un desempeño similar al de un sistema entrenado con descriptores latentes. También, pudimos entregar explicaciones en términos de estos conceptos visuales debido a la naturaleza interpretable del input. Nuestra investigación informa el desarrollo de una nueva aproximación a la recomendación explicable, basada en una representación interpretable basada en conceptos, que no requiere el desarrollo de nuevas arquitecturas de modelos.

Palabras Claves: sistemas recomendadores, inteligencia artificial explicable.

1. INTRODUCTION

1.1. Motivation

In recent years, Deep Neural Networks (DNNs) have become the state-of-the-art model in several tasks, such as natural language processing (Socher et al., 2013), speech recognition (Dahl, Yu, Deng, & Acero, 2011), and image processing (Krizhevsky, Sutskever, & Hinton, 2012). The successful development and performance improvement in this kind of model has motivated both private and public sectors to build their own implementations according to their needs. This has resulted in an enormous variety of applications, ranging from self-driving vehicles[1] to medical diagnosis support, and recommendation systems (S. Zhang, Yao, Sun, & Tay, 2019). However, in the pursuit of better performance, implementations of DNNs have been built as opaque "black boxes", which has caused a loss of interpretability: a model's ability to explain in understandable terms to a human (Doshi-Velez & Kim, 2017).

There are multiple domains where a performant "black box" is not enough to justify preferring this approach over a "transparent" alternative. For example, in a medical environment, a deep learning model could classify and identify a particular disease, but a medic still has to understand what the reasoning for a diagnosis is to provide appropriate treatment. Another important consideration is legal requirements, such as the European Union General Data Protection Regulation (GDPR)[2], a regulation on data protection that, among other articles, includes the right to obtain an explanation in an automated decision-making environment. All of these factors pull the available machine learning approaches to be explainable on top of being performant.

[1] https://www.tesla.com/autopilotAI
[2] https://ec.europa.eu/info/law/law-topic/data-protection/eu-data-protection-rules_en

According to recent research (Guidotti et al., 2018), this loss of interpretability can lead to trust issues, a critical property for interaction (Ribeiro, Singh, & Guestrin, 2016), while interaction itself is a key aspect for users in a recommendation environment. The push for XAI has assisted the development of explainable recommendation systems, systems that explain or justify their recommendations to end-users during their interaction. In this work, we outline a framework to build explainable recommendation systems on top of state-of-the-art recommendation systems, by using concept embeddings, a human-understandable item representation.

1.2. Explainable Artificial Intelligence (XAI)

Given its complexity and its large number of parameters, for a DNN "black box" system is hard to provide an explanation about how it came up with a specific outcome. Explainable AI (XAI) is an active research field that aims to make AI systems more understandable to humans. The concept also involves all the efforts made towards AI transparency and trust concerns (Adadi & Berrada, 2018). XAI pushes to move from opaque "black boxes" to interpretable systems, where a user cannot only receive outputs but also study and understand explanations on how inputs are mathematically mapped to outputs (Doran, Schulz, & Besold, 2017).

Due to the public and private interest in the development of XAI, multiple initiatives have been raised to push the development of the area. For example, the Defense Advanced Research Projects Agency (DARPA), a Department of Defense agency of the U.S. government, raised an XAI program[3] in 2018 to fund and support XAI research. Also, in 2021, the European Union published a proposal[4] to lay down the basis to create the first-ever legal framework on AI. Both the public sector interest and the private sector development

[3] `https://www.darpa.mil/program/explainable-artificial-intelligence`
[4] `https://digital-strategy.ec.europa.eu/en/library/proposal-regulation-laying-down-harmonised-rules-artificial-intelligence-artificial-intelligence`

have pushed a growing interest in XAI research, reflected in funding, events and experts demand.

1.3. Recommendation Systems

An important application of machine learning models is recommendation systems, models that try to predict people's preferences (usually in a personalized manner) to alleviate information overload. Recommendation algorithms analyze transaction data and consumption trends to recommend a platform's consumers other items that could attract the interest of the users. This application can be found in e-commerce (recommend items to buy), social networks (recommend people to people), and social media (music, films, images, etc.).

This kind of algorithm can be found in well-known businesses such as Netflix, Amazon, Spotify, Instagram, and others. For some of these companies, using a recommendation system not only can improve the user experience but also the number of interactions. In the case of Amazon, 35% of what consumers purchase comes from product recommendations, and the number rises to 75% in the case of Netflix, where users receive recommendations on what to watch next[5].

1.4. Outline

This thesis is divided into 8 chapters, including this one. In Chapter 2 we survey relevant research from the recommendation systems and XAI domains, and Chapter 3 details the objective and contribution in our work. Then, Chapter 4 describes the datasets and DNN models used to develop our experiments. Chapter 5 introduces the framework proposed in this work. In Chapter 6 we present the results obtained using our framework,

[5] https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

and compare them with their non-explainable counterpart. Finally, Chapter 7 describes some identified future work opportunities, and Chapter 8 concludes our work.

2. RELATED WORK

In this chapter, we provide an overview of relevant related work. First, we review research on visually-aware recommendation systems. Then we introduce works on explainable recommendation systems and how explanations impact users. A third section introduces state-of-the-art research on XAI techniques for the interpretability of DNNs. The final section highlights the novelty and contribution of this work in contrast to previous work and research in the area.

2.1. Visually-aware Recommendation Systems

Early research in the area relied heavily on collaborative filtering (CF) approaches, user ratings, and items metadata to represent items and user-items interactions. More recent approaches additionally consider the context of the recommendation and the content of the items, but only the latter is relevant in our work. In general, content-based recommendation uses descriptors or representations of items (such as text, images, or video) to feed a recommendation system. The intention is to recommend an item to a user based on its description and a profile of the user's interests (Pazzani & Billsus, 2007).

Initial work in the visual domain relied on manually-crafted visual features, such as color, texture, and local geometry, using techniques developed for content-based image retrieval (Rui, Huang, Ortega, & Mehrotra, 1998; La Cascia, Sethi, & Sclaroff, 1998; Smeulders, Worring, Santini, Gupta, & Jain, 2000). In recent years, works in computer vision tasks, such as object detection, image classification, and semantic segmentation, have benefited from visual features obtained using pre-trained DNN models. Recent studies have demonstrated that model architectures specialized in image classification, such as AlexNet (Krizhevsky et al., 2012), VGG (Simonyan & Zisserman, 2014), and ResNet

(K. He, Zhang, Ren, & Sun, 2016), trained for a specific task (in this case, image classification), also perform well on other tasks and even outperform previous approaches (Khan, Sohail, Zahoora, & Qureshi, 2020).

A visually aware recommendation system, or visual recommendation system, recommends based on visual features of the items. Some examples of relevant image recommendation domains include artworks, fashion, and photography, where users' interests align with parts or certain aspects of the available images. Recent works in visual recommendation systems have extracted visual features from pre-trained DNNs to train recommendation models, and results have shown that this approach performs extremely well in image-driven domains, and in some cases even better if used in hybrid settings (Messina, Dominguez, Parra, Trattner, & Soto, 2019).

Inspired by the performance improvement found, He and McAuley (R. He & McAuley, 2016) investigated the usefulness of using visual features from a pre-trained Deep CNN, for personalized ranking on implicit feedback datasets on top of a traditional matrix factorization (MF) approach. This work showed a performance improvement on multiple datasets and even in cold start settings. An extension of this work was able to also generate new items based on the items representations and also improve the performance by training the pre-trained network and the model jointly (Kang, Fang, Wang, & McAuley, 2017). Some of the latest approaches model users' profiles by aggregating the representation from a pre-trained network to improve performance when interactions are sparse (Messina, Cartagena, Cerda, del Rio, & Parra, 2020), and other works have explored the idea of attention mechanisms to assign weights on both item- and component-levels as an explanation (J. Chen et al., 2017). All these models presented in these works were trained using Bayesian Personalized Ranking (BPR) (Rendle, Freudenthaler, Gantner, & Schmidt-Thieme, 2012), a framework for training models on implicit feedback problems.

Implicit feedback is interactions that indirectly reflect opinion through observed user behavior, a type of interaction that can be treated to develop recommendation systems (Hu, Koren, & Volinsky, 2008).

2.2. Explainability in a Recommendation Context

Explainable recommendation refers to personalized recommendation algorithms that not only provide users or system designers with recommendation results but also explanations to clarify why such items are recommended (Y. Zhang & Chen, 2018). Research in this area not only covers the implementation and design of such systems but also how users perception changes with the presence of explanations.

The idea of explaining recommendations dates back to early work in the recommendation domain, with works that use now familiar approaches like explaining using similar items (Schafer, Konstan, & Riedl, 1999) or similar users (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994). As content-based approaches gained popularity and works shifted to align user profile with content features, feature-based explanations proved to improve effectiveness (Vig, Sen, & Riedl, 2009), and users' trust and satisfaction in the recommendations (Ferwerda, Swelsen, & Yang, 2018). Particularly, in the visual recommendation domain, only recently works have provided explanations using item images. For example, (Lin et al., 2018) generate comments based on the image items as explanations, while (X. Chen et al., 2019) highlight image regions of interest. Also, (J. Chen et al., 2017) uses attentive weights to attribute importance at both item- and component-level.

In a recommendation context, explanations are valuable since they increase the user's satisfaction with the system (Tintarev & Masthoff, 2015), improve trustworthiness (Cramer et al., 2008) and scrutability (Balog, Radlinski, & Arakelyan, 2019). Explanations can be evaluated using both offline and online approaches. User studies are encouraged but not always required for explainable recommendation research, depending on the task at hand (Y. Zhang & Chen, 2018).

2.3. Interpretability of Deep Learning Models

In recent years, XAI community efforts have resulted in the development of several approaches, techniques, methods, and models. These have been classified in multiple review works according to multiple criteria. The scope of the explanation can be classified between local interpretability (explanation for a single data point outcome) and global interpretability (explanation for the model as a whole) (Guidotti et al., 2018; Das & Rad, 2020; Adadi & Berrada, 2018). Another criterion distinguishes between model-specific interpretability (solutions that are limited to a family or class of models) and model-agnostic interpretability (usually post-hoc methods that generate explanations in a different stage to prediction) (Adadi & Berrada, 2018; Das & Rad, 2020; Arrieta et al., 2020). Also, other surveys consider the algorithmic approach (such as changes according to perturbation of the inputs versus gradient-based to analyze model internals) (Das & Rad, 2020) or distinguish between perceptive interpretability versus mathematical structures (the former providing explanations that can be humanly perceived while the latter has a mathematical foundation to provide interpretations) (Tjoa & Guan, 2020).

In this work, techniques to extract visual concepts are highly relevant. This kind of technique exploits the emergence of concept detectors in visual DNNs while training; units specialized as object, color, or part detectors (Zhou, Khosla, Lapedriza, Oliva, & Torralba, 2014). This property of DNNs has allowed research to propose frameworks to quantify the presence of visual concepts. For example, (Bau, Zhou, Khosla, Oliva, & Torralba, 2017) proposes a framework to quantify interpretability and identify and report detectors, by measuring the alignment of each unit in a CNN layer to visual concepts. Further work has expanded the framework to identify compositional logic and approximate neuron behavior (Mu & Andreas, 2020). Another proposed framework measures the alignment of activations of a layer with a vectorial representation of the activations for a visual concept (Kim et al., 2017).

2.4. Differences to Previous Research

To the best of our knowledge, this is the first work that proposes a framework to develop explainable recommendation systems based on visual concepts. In the visual recommendation domain, research on explainable recommendation systems is still in development (Y. Zhang & Chen, 2018). Approaches used on the surveyed recent articles on visually-aware recommendation systems use item representations from pre-trained DNNs as embedding. This kind of feature cannot be used with feature-based approaches for explanation, due to the latent nature of their representation. We propose to create a new representation using some of the latest works on XAI for the extraction of visual concepts. In particular, we expand NetDissect (Bau et al., 2017) to create a local explanator on top of their global explanator.

3. OBJECTIVES

In this area, explainable recommendations are of increasing value since users benefit in multiple aspects from their existence. Previous work has proposed novel model architectures to deliver explanations, and even though they're performant and explainable, their complexity is increasing to achieve this goal.

The general objective of this work is to propose a framework to develop explainable visually-aware recommendation systems without the need of developing new models and without losing performance.

3.1. Research Questions

(i) **Is it possible to build an interpretable item representation?**
There is recent work in XAI that explains in terms of visual concepts. We want to test if a custom implementation can deliver local explanations that can be considered explicit item representations.

(ii) **Can we deliver accurate recommendation using concept-based presentations?**
We want to test whether using our proposed explainable approach can achieve a similar or better offline performance that recent works have shown.

(iii) **Can we provide explanations of recommendations in terms of visual concepts?**
We want to provide explanations in terms of visual concepts from the item representation. This problem has already been addressed by research, and XAI has developed solutions like feature attribution methods.

3.2. Contribution

In this work, we propose the first framework to develop explainable recommendation systems by using existing model architectures. Then, we contribute to: (a) the area of

XAI by developing a local explanator as an extension of (Bau et al., 2017) to identify visual concepts in images, and (b) the area of visually-aware recommendation systems by designing and implementing the framework and delivering explanations in terms of visual concepts. Both contributions help to develop the visual explainability domain, by delivering explanations in terms of visual concepts, with a novel approach to this task.

The proposed framework is tested using state-of-the-art DNN models in a recommendation task to show that the approach can be generalized to a family of models and to compare the performance of the proposal to state-of-the-art implementations. To the best of our knowledge, there's no similar research to our proposed framework, that identifies and explains in terms of visual concepts, to develop explainable recommendation systems.

4. MATERIALS AND METHODS

In this chapter, we describe the materials used in our work. First, we introduce the datasets used in the recommendation task. Then, we introduce the deep learning recommendation models (from the literature) whose performance will be analyzed and compared.

4.1. Datasets

In this study, our research questions are related to the performance of a recommendation system using the proposed framework compared to its non-explainable counterpart. To make this comparison possible, we must first choose a recommendation task and then train and evaluate the models in a similar manner. We'll focus on visual recommendation systems, meaning recommendation tasks where the items are images on some domain. In our study we rely on two datasets from different domains:

4.1.1. UGallery

UGallery[1] is an online art gallery implemented as an e-commerce platform, where artists can showcase and sell their art pieces to the platform users. The dataset provided by UGallery consists of 2919 users, 13297 items, and 4897 individual purchases or transactions of different art pieces.

A particular property of the UGallery dataset is its one-of-a-kind nature. In its majority, artworks correspond to physical pieces, meaning that they can only be sold once by the platform. This also causes the number of interactions to be significantly lower compared to other datasets (2.2 interactions per user and 0.49 interactions per item, on average).

[1]https://www.ugallery.com/

These properties of the dataset motivated the authors of CuratorNet (Messina et al., 2020) to design and develop said model to perform well even considering these restrictions.

4.1.2. Wikimedia

Wikimedia Commons[2] is an online repository of free-use media resources and is a project of the Wikimedia Foundation. For the development of this project, a set of Wikimedia Commons images will be used: Featured picture candidates[3]. This dataset contains images nominated by Wikimedia Commons contributors to Featured picture. In this platform, users can comment and vote for or against a nomination, where images that meet certain criteria are selected as highlighted. This dataset contains approximately 33000 images and 190000 interactions between the items and 7500 users.

This dataset is also particular in its nature. In Wikimedia, users can interact with an item in multiple ways and their positive interactions do not mean necessarily “consumption” but “support”, which means that users have interactions with a high number of items (32 interactions per user, on average). Also, users consider not only the content of the image, but also its quality (for example, its resolution). We think that these properties could harm results in traditional metrics, but the dataset is still relevant due to the value of the image content.

4.2. Recommendation Models

We use some of the current state-of-the-art image recommendation models in our recommendation tasks. After the training phase, we will perform an offline evaluation to compare the performance of the proposed framework against its non-explainable counterpart. All models, except the first one, are DNNs. A brief description of each is available below:

[2] `https://commons.wikimedia.org/wiki/Main_Page`
[3] `https://commons.wikimedia.org/wiki/Commons:Featured_picture_candidates`

- **VisRank** (Kang et al., 2017) is a simple model that recommends to users through a simple 'nearest-neighbor' style strategy. This means that a similarity score is calculated between each potential recommendation and an aggregation of the items already consumed by the user. This is not a DNN model and does not require any training, but a common implementation involves calculating all the possible pairing of items.
- **VBPR: Visual Bayesian Personalized Ranking** (R. He & McAuley, 2016) is a personalized ranking model that simultaneously uses visual and latent signals: image content and known interactions. Its architecture uses a pre-trained network to extract visual features, which are combined with latent item and user factors to predict a score.
- **CuratorNet** (Messina et al., 2020) is a visually-aware DNN recommendation model, meaning that it considers the content of the images to calculate a similarity score between user-item pairs. This model architecture generates its internal representation of a user using the content of the already consumed items. Items features are obtained using a pre-trained network, as a latent representation.

Both VBPR and CuratorNet are trained using a Bayesian Personalized Ranking (BPR) framework (Rendle et al., 2012). Using BPR, a model learns a ranking function in a personalized manner from implicit feedback, by learning the correct ordering between pairs of interactions.

The code implementation of the listed models and their training pipeline is publicly available in a GitHub repository[4].

[4] `https://github.com/aaossa/Model-experiments/`

4.3. Offline evaluation metrics

To perform offline evaluation and compare the results of our experiments, we use ranking metrics commonly used in similar studies. All of the following metrics have in common that the values belong to the range $[0, 1]$, where 1 means perfect performance as defined by the metric:

- **AUC**: Short for ROC Area Under Curve (area under the receiver operating characteristic curve), uses the position of the ground truth items in the ranking of all the available items to approximate an "exact" value of the metric, as opposed to calculating the value using a sample of the inventory.
- **MRR**: Mean Reciprocal Rank assigns a score according to the position of the first relevant item in the recommendation list. The metric is defined as follows:

$$MRR = \frac{1}{r}$$

  , where r is the position in the recommendation ranking of the first relevant item.
- **R@k**: Recall at k measures the proportion of relevant items captured by a recommendation list of length k. The metric is defined as follows:

$$Recall = \frac{|Recommended \cap Relevant|}{|Relevant|}$$

- **P@k**: Precision at k measures the proportion of items in a recommendation list of length k that are relevant. The metric is defined as follows:

$$Precision = \frac{|Recommended \cap Relevant|}{|Recommended|}$$

- **N@k**: Short for nDCG at k (normalized discounted cumulative gain), is a measure of ranking that values putting the relevant items higher in the recommendation ranking. The metric is calculated as a proportion of the discounted cumulative gain in the recommendation ranking over the same measurement on an ideal ranking, and is defined as follows:

$$nDCG = \frac{DCG}{iDCG}$$

, where $iDCG$ is the DCG of the ideal ranking. The value of DCG is calculated as follows:

$$DCG = \sum_{i}^{Recommended} \frac{2^{rel_i} - 1}{\log_2(1 + i)}$$

, where rel_i is a piecewise function valued 1 if the item i is relevant and 0 in other case.

5. PROPOSED SOLUTION

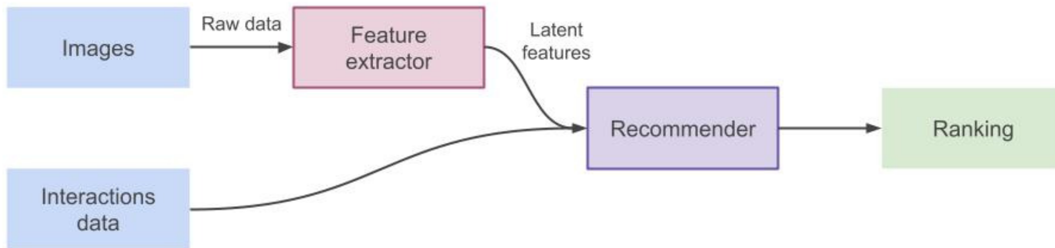
In this chapter, we describe our proposed framework to develop explainable visual recommendation systems. To achieve this goal, we extend recent work to develop both performant and explainable visually-aware recommendation systems. A diagram that compares the traditional approach and our work is shown in figure 5.1.

We present our framework in three parts: (i) Building a representation of visual concepts, which describes our extension of NetDissect (Bau et al., 2017) to develop a local perceptive interpretability method that represents images in terms of their visual concepts, (ii) Computing recommendations, that describes both our proposal and the differences with the state-of-the-art approach, and (iii) Explaining recommendations using visual concepts, that describes our approach to deliver explanations.

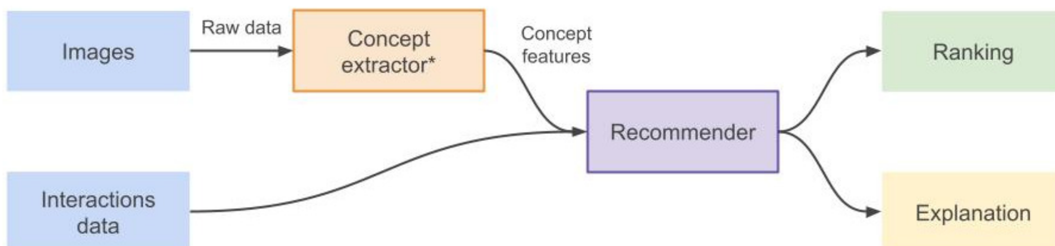
5.1. Building a Representation of Visual Concepts

In the first stage, we aim to construct an item representation based on visual concepts. This means that we need to develop a mechanism that, given an image y returns a vector $V(y)$ of weighted visual concepts that are present in image y. We need a vectorial representation with similar properties to the ones of a latent embedding of visual features, like the embeddings generated using pre-trained DNNs. Inspired by recent works that develop methods to explain DNNs using visual concepts we develop our own local and perceptive explainer capable of generating the desired vector $V(y)$.

To build our local explainer, we require a mechanism to align internal representations of a DNN with human-understandable visual concepts. Some recent works address this problem through different techniques. For example, TCAV (Kim et al., 2017) quantifies



(a) Traditional pipeline



(b) Proposed framework pipeline

Figure 5.1. Comparison of traditional and proposed pipeline. In a traditional setting, latent features are used to train a recommendation system, while our approach proposes using a concept extractor instead, which allows us to explain recommendations in terms of visual concepts.

the sensitivity of a layer for a visual concept as the alignment between activations of images that contain the concept compared to a set of activations from random images (a linear classifier is trained to distinguish both sets of activations). In Summit (Hohman, Park, Robinson, & Chau, 2019), authors aggregate and summarize activations and influences in a DNN to identify relevant concepts and their relations in convolutional layer units. Their work aims to develop an interactive tool to explore the internal representations of a DNN and approximate a graph-like structure to explain a class.

In this study, to achieve our goal we extend NetDissect (Bau et al., 2017), or network Dissection, a framework to quantify interpretability in individual units of a CNN. NetDissect is a global and perceptive explainer, which means that explanations (1) explain model-level logic and are not instance dependent, and (2) can be perceived by humans, in this case, as visual concepts identifiable in an image. NetDissect is described as a general framework to measure the alignment between the activations of hidden units in a convolutional layer and a set of visual concepts. The framework evaluates each unit in a CNN layer against each concept present in Broden, a segmentation dataset presented in the same work.

Both TCAV and Summit share some similarities with NetDissect in their objectives. An advantage of TCAV is the capability of generalization to new concepts, but the technique measures the alignment between activations and concepts in a layer-wise manner instead of at the unit level. Summit shares similarities with NetDissect in the unit activation analysis, but the study differentiates in the aggregation of results and its purpose. Also, Summit does not aim to develop a measurement of alignment between unit activations and concepts, as NetDissect does. We decided to use NetDissect due to its success in different explainability tasks and studies. It is relevant to mention that both TCAV and Summit could be adapted in one way or another to build a different concept extractor, but that is out of the scope of this thesis.

NetDissect is formalized by the following definitions, which we'll reference later. Given an input image x, NetDissect collects the activation map of hidden unit k: $A_k(x)$. Then, collects the distribution of activations a_k for each individual unit, and determines a threshold T_k such that the probability of $a_k > T_k$ is equal to 0.005 over all the locations of the activation maps in the dataset. In the process, $A_k(x)$ has to be scaled-up to be compared against the segmentation masks in the Broden dataset, resulting in $S_k(x)$. Then, $S_k(x)$ is compared against the threshold T_k, creating a binary mask $M_k(x) \equiv S_k(x) \geq T_k$.

To score each unit k alignment with a visual concept c (with a ground-truth segmentation mask L_c), the intersection over union is calculated over the whole dataset and then summed, resulting in $IoU_{k,c}$, a scalar that represents the alignment of unit k activations with the visual concept c. More details are described in the original work of Bau. et al. (Bau et al., 2017).

In our work, we extend the previous method to approximate a local explanation for a single image. First, we apply the NetDissect framework, which means that for each unit k we already know the visual concept c that aligns the most with its activations and the corresponding pair $IoU_{k,c}$ value, and the threshold T_k to determine a significant activation at unit k. Then, given an unseen input image y and a unit k, we make a forward pass to compute the activations in the network, which allows us to compute the activation map $A_k(y)$ and the binary mask $M_k(y)$ using the known threshold value T_k. An overview of the relevant symbols is also contained in Table 5.1. In the next step, we must aggregate this unit-level information to construct our vector $V(y)$. We define 4 criteria to aggregate this information:

Table 5.1. Symbols used in the formalization of NetDissect and our proposed algorithm.

Symbol	**Description**
L	set of convolutional layers in a DNN
C	set of visual concepts
k	a specific unit in a layer
$A_k(x)$	activation map of unit k, given an input image x
T_k	threshold of activation for unit k
$IoU_{k,c}$	accuracy of unit k in detecting concept c
S	score (IoU) threshold to determine if a unit is a *unique detector*
$s_{k,c}$	score of unit k for concept c
w_l	weight of layer l
$f(x, k)$	boolean function, determines if unit k was “activated” by image x
$V(x)$	vector of visual concepts in image x

5.1.1. Criteria 1: Unit score computation

We define the unit score $s_{k,c}$ as the score of unit k for concept c. NetDissect calculates this score as $IoU_{k,c}$, but its computation requires a known L_c mask, a ground truth annotation of concept c in the relevant image. In our algorithm, we don't have such annotation, but we determined 3 possible options. In each option, we only consider activated units (those with a individual unit activation that surpass the unit threshold) and the concept label c assigned by NetDissect, unless said otherwise:

- **Absolute**: This option scores a unit according to the number of individual unit activations a_k, elements in the activation map $A_k(x)$, that are greater or equal to the threshold T_k. To formalize this option, we first define a set of individual unit activations that met the described condition: $G = \{a_k \in A_k(x) : a_k \geq T_k\}$, and then calculate its cardinality: $s_{k,c} = |G|$, where c is the concept label assigned by NetDissect.
- **Discrete**: This option only outputs 1 or 0 to represent unit activation. Under this criteria, a unit is considered activated if there exists an individual unit activation a_k in the activation map $A_k(x)$ such that the threshold T_k of the unit is surpassed. For this purpose, we use the set G of individual unit activations that met the described condition and define $s_{k,c}$ as a piecewise function that outputs 1 if $|G| > 0$ and 0 in other cases, which can be considered a boolean function in our implementation.
- **Categorical**: This option only considers units that are considered activated according to the definition of the "discrete" option. This option assigns a categorical score to a unit, meaning that a unit will have multiple scores for concepts c_1, c_2, and so on, all from different categories. For a particular category j, the score of unit k is calculated as a proportion of the reported score by NetDissect for a concept c_j of said category, IoU_{k,c_j}, and the highest score reported for any concept in that unit, $IoU_{k,c}$. For example, if unit k was assigned label "red", a

concept in the “color” category, its score for the “color” category will be 1, but for another category, say “scene” the value of $s_{c_{scene},k}$ will be $\frac{IoU_{k,c_{scene}}}{IoU_{k,c_{color}}}$.

5.1.2. Criteria 2: Consider unique detectors only

In NetDissect, authors quantify the interpretability of a convolutional layer by counting “the number of unique concepts aligned with units, i.e. *unique detectors*”. They define a *detector* for concept c if the reported $IoU_{k,c}$ surpasses certain threshold, defined by the authors. In our algorithm, we consider two possible options: consider only the score $s_{k,c}$ when unit k is a *unique detector* for concept c, or consider all the units.

5.1.3. Criteria 3: Layer weight computation

After we determine the score $s_{k,c}$ (defined by Criteria 1) of the relevant units (defined by Criteria 2) in a convolutional layer, and before aggregating units from different layers (to be defined in Criteria 4), we must compensate the units score according to the layer results. For example, if layers l_1 and l_2 have 3 activated units each, but l_1 has substantially more units than l_2, they may have different importance on the final outcome.

In NetDissect, the authors quantify the interpretability of a layer, because in their framework the analysis of each layer is independent of the others. In our algorithm, as we aim to construct a representation that considers information from multiple layers, we consider the existence of a layer weight w_l. For its computation, we determined 5 possible options:

- **Equal**: The simplest approach considers the contribution of units in different layers as equal. This can be formalized as $w_l = 1, \forall l \in L$.
- **Proportional to *unique detectors***: This option assigns the weight of a layer as a proportion of the number of unique detectors (obtained from NetDissect) and

the number of units in layer l. To formalize this option, we can define such a proportion as $w_l = \frac{\mathbf{card}(u_k \in l : IoU_k \geq S)}{\mathbf{card}(u_k \in l)}$.

- **Proportional to activated units**: Similarly to the previous option, it assigns the weight of a layer as a proportion of the number of activated units and the number of units in layer l. To formalize this option, we can define such a proportion as $w_l = \frac{\mathbf{card}(u_k \in l : f(x,k)=1)}{\mathbf{card}(u_k \in l)}$, where f has the same definition given in the "discrete" option of Criteria 1.
- **Inverse to proportion of *unique detectors***: This option assigns the weight of a layer as the inverse of the proportion of *unique detectors*. To formalize this option, we can define such a proportion as $w_l = \frac{\mathbf{card}(u_k \in l)}{\mathbf{card}(u_k \in l : IoU_k \geq S)}$.
- **Inverse to proportion of activated units**: This option assigns the weight of a layer as the inverse of the proportion of activated units. To formalize this option, we can define such a proportion as $w_l = \frac{\mathbf{card}(u_k \in l)}{\mathbf{card}(u_k \in l : f(x,k)=1)}$, where f has the same definition given in the "discrete" option of Criteria 1.

5.1.4. Criteria 4: Aggregation of units with the same concept

Now, we have determined a way to compute the score $s_{k,c}$ of relevant units, after weighting said score according to the layer where they're from. In this criteria, we present 2 options to aggregate units' scores from different layers.

- **Addition**: This option adds the score $s_{k,c}$ (multiplied by their corresponding layer weight w_l) of units labeled with the same concept c. This results in the final score assigned for a concept defined as $V_c = \sum_{u_k \in R} s_{k,c} \times w_l$, where R is the set of relevant units of any layer after applying Criteria 2.
- **Average**: This option averages the score $s_{k,c}$ (weighted by the corresponding layer weight w_l) of units labeled with the same concept c. This results in the final score assigned for a concept defined as $V_c = \frac{\sum_{u_k \in R} s_{k,c} \times w_l}{\mathbf{card}(R)}$, where R is the set of relevant units of any layer after applying Criteria 2.

5.1.5. Algorithmic definition

Each of the described criteria corresponds to a decision in our algorithm to build a representation $V(\cdot)$ of an image through visual concepts (Algorithm 1). This means that each combination of criteria results in a slightly different algorithm to transform both the activations of image y and the global information provided by NetDissect into a representation of visual concepts $V(y)$.

Algorithm 1: Build representation $V(y)$ of visual concepts for image y

Input : An image y, and a set L of convolutional layers in a DNN
Output: A vectorial representation $V(y)$ of visual concepts in image y

```
Initialize V as an empty array;
foreach visual concept c do
    Initialize V_c as an empty array;
    foreach convolutional layer l do
        w_l ← Criteria3(l);
        foreach unit u_k ∈ Criteria2(l) do
            s_{k,c} ← Criteria1(x,k,c);
            V_c[k] ← s_{k,c} · w_l;
    V_c ← Criteria4(V_c);
    V[c] ← V_c;
return V;
```

The output of Algorithm 1 is a vector $V(y)$ containing the score for each tested concept. In its i-th index, the mentioned score is a value directly proportional to the presence of visual concept c_i in the input image y.

The code implementation of the proposed algorithm is publicly available in a GitHub repository[1].

[1] `https://github.com/aaossa/NetDissect-experiments/`

5.2. Computing Recommendations

In the second stage, we aim to compute personalized recommendations. For this purpose, we use our proposed algorithm (Algorithm 1) to generate a vectorial representation $V(y)$ of the weighted visual concepts present each item image y from a given dataset. Then, we create our embedding based on the generated explicit representations to train state-of-the-art recommendation models and evaluate their offline performance.

Recent visually-aware recommendation systems works (R. He & McAuley, 2016; Kang et al., 2017; Messina et al., 2020) use a pre-trained DNN as a feature extractor, as shown in Figure 5.1a. These features are then used to create an embedding in which each item is represented by a row in a matrix of latent features. An important disadvantage of these embeddings for explainability is that they are latent in nature, meaning that they're a low-dimensional representation, a compression of relevant information (according to the pre-trained DNN). Said representation is useful for a DNN but is hard to grasp for a human and hard to align to visual concepts.

In our proposal, we use Algorithm 1, detailed in the previous section, to create a vectorial representation $V(y)$ based on the visual concepts present in an image y. We construct the embedding by stacking the obtained representation of every item, like the traditional approach, but instead of using a DNN as a feature extractor, we consider our algorithm as a concept extractor, as shown in Figure 5.1b. Due to the nature of the proposed embedding being similar to the generated by a feature extractor, we can use the embedding to train a visually-aware recommendation system without changing its architecture, except for the number of units in the input layer to match the size of our embedding. Also, because of our guided construction, we know exactly what each column and value represent in our embedding: a concept c and its presence in an image. This means that our data is inherently interpretable, and can be represented as tabular data, where each column c correspond to a visual concept c, each row y represents a particular item in a dataset, and position c, y in the table is the score $V(y)[c]$ directly proportional to the presence of visual concept c in image y.

To evaluate the performance of our proposed embedding, we first define a baseline model. Due to the number of possible configurations of Algorithm 1 the baseline will be used to compare and discard configurations. In (Kang et al., 2017) and (Messina et al., 2020), authors choose Visrank as a baseline model. Visrank is a simple model, based on the distance between a pair of items, for visual recommendation. The model was presented and formalized in Chapter 4.

Then, we compare the traditional feature extractor and our proposed concept extractor by measuring their offline performance on two visual recommendation datasets: UGallery and Wikimedia. To generate recommendations, we use VBPR (R. He & McAuley, 2016) and CuratorNet (Messina et al., 2020), two state-of-the-art visually-aware recommendation systems trained using the traditional pipeline. Both the datasets and DNN models were presented and formalized in Chapter 4.

The code implementation of the mentioned models and their corresponding training process is publicly available in a GitHub repository[2].

5.3. Explaining Recommendations Using Visual Concepts

In the third and last stage, we aim to deliver explanations by exploiting the properties of our embedding of visual concepts. Due to the interpretable nature of our proposed concept embedding, we need to define a mechanism to deliver explanations in terms of human-understandable visual concepts that consider the explicit representation of the relevant items.

[2] `https://github.com/aaossa/Model-experiments`

For this purpose, we choose a feature attribution approach to deliver explanations. Feature attribution methods allow us to identify how much each input feature contributed to the model output. In particular, we are interested in local explainers in this family of methods. The problem we are addressing is how to assign an importance value to the components of our item representation (each visual concept) that reflects the recommendation system logic to deliver a specific recommendation.

Feature attribution methods are particularly useful when working on tabular data because each component (or column) is a known identifiable property of the instances in a dataset. Under a traditional visual recommendation approach, this kind of methods is hard to apply due to the large number of dimensions in an image (number of pixels) or the latent nature of a feature extractor embedding (highly compress data that's not aligned with human concepts), depending on if a model receives the raw image or the output of a feature extractor, respectively.

In our case, feature attribution methods are a good fit because our data can be represented as tabular data. Our explicit embedding aligns our visual concept representations, in a way that each column is a concept c and the values in it represent the presence of said concept in an image. If a feature attribution method is applied and assigns certain importance to a specific column in an input row, we know which concept c is involved and what the specific score represents.

In this work, we use SHAP (Lundberg & Lee, 2017), for Shapley Additive Explanations, a game-theoretic approach to explain the output of any machine learning model. SHAP can be categorized as a local and model-agnostic explainer, which means that explanations (1) explain a single instance at a time, and (2) work on any kind of model. SHAP is described as a framework to assign each feature an importance value for a particular prediction. The framework proposes using SHAP values as a measure of feature importance. SHAP values attribute to each feature the change in the expected model prediction when conditioning on that feature (Lundberg & Lee, 2017).

SHAP is defined as an additive feature attribution method, meaning that the method attributes an effect to each feature and, by summing the effects of all feature attributions, approximates the output of the model. In particular, due to the recommendation systems tested being DNNs, we focus on using Deep SHAP, a combination of DeepLIFT (Shrikumar, Greenside, & Kundaje, 2017) and Shapley values, which is one of the model-type specific approximation methods defined by the cited work.

This method is part of the state-of-the-art techniques on feature attribution and has been used in different contexts. The additive nature of its attributions could be helpful for users to understand explanations in a recommendation context. A limitation in SHAP is that attribution maps are sometimes noisy, resulting in explanations that include small attributions values that are not useful to understand the model outcome. Additionally, SHAP is not specialized in recommendation models and has not been used widely in this context, but can be used in the studied models because of its model-agnostic properties.

The code implementation of the described method to deliver explanations is publicly available in a GitHub repository[3].

[3]`https://github.com/aaossa/Concept-experiments/`

6. RESULTS

In this chapter, we report and analyze the results to present valuable information to discuss our research questions. In the first section, we explore and characterize the results of our algorithm for building a representation of visual concepts. In the second section, we compare the performance of the chosen baseline model and state-of-the-art visually-aware recommendation systems, using both the traditional approach (using a feature extractor) and our proposal (using a concept extractor). Then, in the next section, we explore the explanations obtained with our proposed solution in a given recommendation problem. Finally, we discuss our research questions with our analysis in hand.

6.1. An Interpretable Item Representation

6.1.1. Preprocessing of Concept Embeddings

Due to the tabular nature of our concept embeddings, we can apply common preprocessing methods to our data. We performed offline evaluation on all 60 possible configurations of our algorithm, with no preprocessing, applying standardization, applying normalization, and applying a scaler robust to outliers. For each method, we also dropped the columns (concepts) with zero-variance, due to a limitation in the NetDissect report: some concepts do not align significantly with any unit in a DNN.

Table 6.1 shows a summary of its results in UGallery, and Table 6.2 the same results in Wikimedia. The detailed results for this section can be downloaded from `https://bit.ly/3f3ERTh`. Here we report each preprocessing technique and its performance compared to the others.

Raw data. We tested the performance of the concept embedding without preprocessing or filtering of any kind in VisRank, our baseline model, in both datasets (UGallery and

Wikimedia). We expected the raw data to outperform a random baseline but to be outperformed by some of the preprocessing methods, and our results confirmed this hypothesis.

Normalization. In this approach, we scale the data to a fixed range: 0 to 1. This approach causes the data to reduce the size of its standard deviations, but column values will be very sensitive to outliers. Normalization should outperform both a random baseline and the raw data, and our results confirm this hypothesis.

Robust scaling. We also applied a transformation to scale by subtracting the median and dividing by the interquartile range (difference between 75th quartile and 25th quartile). We expected robust scaling to at least outperform the raw data, but our results show that this preprocessing method performed worse than our no-preprocessing baseline in all the metrics in both datasets.

Standardization. In this approach, we apply Z-score normalization by rescaling the data to have the properties of a standard normal distribution ($\mu = 0$ and $\sigma = 1$). We expected standardization to outperform our baselines and be competitive with other preprocessing methods, but we found that it outperforms all the tested preprocessing approaches by a margin in all metrics and both datasets. Based on these results, from this point forward, we will apply standardization on all our representations, and we won't consider any other preprocessing.

It is important to notice that even if the results of standardization show better performance than any other preprocessing and our random baseline, it is not better than using a feature extractor. This is not a surprise considering the good results that the latent embeddings from a pre-trained DNN perform so well in so many tasks thanks to transfer learning, and even better when finetunning.

Table 6.1. AUC, Mean Reciprocal Rank (MRR), Recall (R), Precision (P), nDCG (N) at different recommendation list lengths (20, 100, 200), in the UGallery dataset. Shown results are an average for each metric of the prepropressing method in the first column. The best absolute average of each metric is highlighted, without considering the first row, latent features.

Preprocessing	**AUC**	**MRR**	**R@20**	**P@20**	**N@20**	**R@100**	**P@100**	**N@100**	**R@200**	**P@200**	**N@200**
Latent features	.66714	.03424	.07326	.00440	.04312	.12722	.00155	.05420	.16431	.00101	.05961
Raw data	.61340	.01549	.03067	.00176	.01828	.06496	.00077	.02513	.08895	.00053	.02897
Normalization	.61805	.01584	.03093	.00179	.01855	.06530	.00076	.02530	.09072	.00054	.02936
Robust scaling	.61157	.01191	.02149	.00122	.01352	.04588	.00054	.01856	.06754	.00040	.02201
Standardization	**.64287**	**.01751**	**.03188**	**.00183**	**.02002**	**.07090**	**.00083**	**.02778**	**.10021**	**.00059**	**.03243**
Random	.49868	.00066	.00137	.00007	.00032	.00904	.00011	.00200	.01087	.00007	.00239

Table 6.2. AUC, Mean Reciprocal Rank (MRR), Recall (R), Precision (P), nDCG (N) at different recommendation list lengths (20, 100, 200), in the Wikimedia dataset. Shown results are an average for each metric of the prepropressing method in the first column. The best absolute average of each metric is highlighted, without considering the first row, latent features.

Preprocessing	**AUC**	**MRR**	**R@20**	**P@20**	**N@20**	**R@100**	**P@100**	**N@100**	**R@200**	**P@200**	**N@200**
Latent features	.59449	.01135	.02006	.00116	.01290	.03390	.00040	.01557	.04735	.00027	.01755
Raw data	.56678	.00812	.01412	.00083	.00921	.02379	.00028	.01106	.03101	.00018	.01215
Normalization	.56741	.00855	.01467	.00084	.00968	.02437	.00029	.01153	.03197	.00019	.01266
Robust scaling	.56105	.00672	.01117	.00065	.00750	.01963	.00023	.00909	.02603	.00015	.01005
Standardization	**.57730**	**.00879**	**.01489**	**.00085**	**.00987**	**.02570**	**.00030**	**.01190**	**.03442**	**.00020**	**.01322**
Random	.50436	.00023	.00000	.00000	.00000	.00263	.00003	.00044	.00585	.00003	.00092

6.1.2. Exploration of embeddings configurations

Due to the large number of possible criteria and their combinations in our algorithm, we try to determine the configurations that yield better performance based on the results on our baseline model VisRank. Table 6.3 and Table 6.4 contain a summary of the average performance on the baseline task for each criteria option, in UGallery and Wikimedia respectively.

In our analysis, some criteria options performed significantly better than others, which will be useful to reduce the size of the performed tests. For example, in Criteria 1 (Unit score), “Categorical” outperformed both “Discrete” and “Absolute” in several metrics. Due to this performance, in the following sections, we will only consider embeddings created with this unit scoring criteria.

Then, we focused on Criteria 3 (Layer weight). In this case, we notice that “Inverse to proportion of unique detectors” performed significantly worse than the other options. Results yield a similar observation with “Inverse to activated units”, but not as apparent as in the previous layer weight criteria. Due to these observations, we will only consider embeddings created with the rest of the possible values in Criteria 3.

Finally, we grouped our results according to two criteria. the first group was composed of configurations where Criteria 2 was “Unique detectors only” and Criteria 4 was “Average”, or Criteria 2 was “Every unit” and Criteria 4 was “Addition”, and the second group contained the opposite cases. Through inspection, we can confirm that the first group performs better than the second. This will also reduce the number of criteria options to be tested in the following sections.

Considering our results and analysis in this subsection, 6 configurations were chosen to be compared with the latent feature embeddings in the following subsection, to train visually-aware recommendation systems. The chosen configurations correspond to

Criteria 1 equal to “Categorical” (1 possible value); Criteria 3 equal to “Equal weight”, “Proportional to unique detectors”, or “Inverse to proportion of unique detectors” (3 possible values); and two pairings of Criteria 2 and Criteria 4: “Unique detectors only” and “Average”, and “Every unit” and “Addition”, respectively (2 possible values). All 6 configurations will be evaluated in the following sections by their performance training visually-aware recommendation systems.

Table 6.3. AUC, Mean Reciprocal Rank (MRR), Recall (R), Precision (P), nDCG (N) at different recommendation list lengths (20, 100, 200), in the UGallery dataset. Shown results are an average for each metric of the embeddings with a given criteria value, from the first column. Consider that "UD" means "unique detectors", and "AU" means "activated units". In each criteria, the highest value for each metric is highlighted.

Criteria	**Value**	**AUC**	**MRR**	**R@20**	**P@20**	**N@20**	**R@100**	**P@100**	**N@100**	**R@200**	**P@200**	**N@200**
-	Latent features	.66714	.03424	.07326	.00440	.04312	.12722	.00155	.05420	.16431	.00101	.05961
Unit score	Discrete	.64773	.01622	.02664	.00152	.01785	.06714	.00079	.02597	.09479	.00056	.03025
	Absolute	.62345	.01630	.02901	.00167	.01850	.06304	.00074	.02536	.09456	.00056	.03028
	Categorical	**.65743**	**.02001**	**.03998**	**.00230**	**.02370**	**.08253**	**.00096**	**.03202**	**.11129**	**.00066**	**.03677**
Unique detectors	UD only	**.64745**	**.01823**	**.03399**	**.00196**	**.02113**	**.07396**	**.00087**	**.02906**	**.10188**	**.00061**	**.03345**
	Every unit	.63829	.01679	.02976	.00171	.01891	.06784	.00079	.02651	.09855	.00058	.03142
Layer weight	Equal weight	.64954	.01827	.03356	.00191	.02084	**.07764**	**.00090**	.02951	.10851	**.00064**	.03449
	Prop. to UD	.64746	.01871	**.03484**	**.00196**	**.02149**	.07743	.00089	.02981	.10834	.00063	.03473
	Prop. to AU	.61916	.01443	.02532	.00145	.01636	.05428	.00063	.02225	.07572	.00044	.02557
	Inv. to UD	**.65359**	**.01874**	.03411	**.00196**	.02139	.07610	**.00090**	**.02987**	**.10856**	**.00064**	**.03498**
	Inv. to AU	.64460	.01740	.03156	.00188	.02000	.06907	.00084	.02747	.09993	.00060	.03241
Aggregate units	Addition	**.64644**	**.01771**	**.03403**	**.00193**	**.02062**	**.07341**	**.00085**	**.02848**	**.10321**	**.00061**	**.03326**
	Average	.63930	.01731	.02973	.00173	.01941	.06840	.00081	.02708	.09721	.00058	.03161
-	Random	.49868	.00066	.00137	.00007	.00032	.00904	.00011	.00200	.01087	.00007	.00239

Table 6.4. AUC, Mean Reciprocal Rank (MRR), Recall (R), Precision (P), nDCG (N) at different recommendation list lengths (20, 100, 200), in the Wikimedia dataset. Shown results are an average for each metric of the embeddings with a given criteria value, from the first column. Consider that "UD" means "unique detectors", and "AU" means "activated units". In each criteria, the highest value for each metric is highlighted.

Criteria	**Value**	**AUC**	**MRR**	**R@20**	**P@20**	**N@20**	**R@100**	**P@100**	**N@100**	**R@200**	**P@200**	**N@200**
-	Latent features	.59449	.01135	.02006	.00116	.01290	.03390	.00040	.01557	.04735	.00027	.01755
Unit score	Discrete	.57662	.00848	.01467	.00085	.00953	.02617	.00031	.01167	.03483	.00020	.01299
	Absolute	.57729	.00836	.01450	.00081	.00949	.02417	.00028	.01126	.03219	.00019	.01248
	Categorical	**.57798**	**.00953**	**.01551**	**.00088**	**.01060**	**.02675**	**.00032**	**.01278**	**.03625**	**.00021**	**.01419**
Unique detectors	UD only	**.57743**	**.00904**	**.01529**	**.00087**	**.01016**	**.02580**	**.00030**	**.01211**	**.03489**	**.00020**	**.01350**
	Every unit	.57717	.00854	.01449	.00083	.00958	.02559	**.00030**	.01169	.03395	**.00020**	.01294
Layer weight	Equal weight	.58370	.00913	.01536	.00087	.01023	.02687	.00032	.01241	.03643	.00021	.01384
	Prop. to UD	.58398	**.00927**	**.01569**	**.00089**	**.01039**	**.02763**	**.00033**	**.01266**	**.03704**	**.00022**	**.01410**
	Prop. to AU	.55258	.00821	.01348	.00078	.00919	.02113	.00025	.01061	.02721	.00016	.01149
	Inv. to UD	**.58412**	.00909	.01539	.00088	.01021	.02751	.00032	.01248	.03647	.00021	.01384
	Inv. to AU	.58210	.00824	.01453	.00083	.00936	.02534	.00030	.01135	.03498	.00021	.01283
Aggregate units	Addition	**.58083**	**.00902**	**.01529**	**.00087**	**.01014**	**.02624**	**.00031**	**.01223**	**.03515**	**.00021**	**.01357**
	Average	.57376	.00855	.01449	.00083	.00961	.02515	.00030	.01157	.03370	.00020	.01287
-	Random	.50436	.00023	.00000	.00000	.00000	.00263	.00003	.00044	.00585	.00003	.00092

6.2. Offline Evaluation of Concept Embedding

In this section, we first analyze the performance of the embedding in the baseline model against our simple recommendation task. Then we will compare the best embeddings in the DNN models.

6.2.1. Performance of embeddings in VisRank

Table 6.6 and Table 6.7 show the results of our baseline model using the concept embeddings (chosen in the previous section) to recommend on UGallery and Wikimedia. Embeddings configurations are codified according to Table 6.5. Both in the UGallery and Wikimedia datasets, the concept embeddings present competitive results in terms of AUC, Recall@200, Precision@200, and nDCG@200. In the case of UGallery, all of our embeddings beat the latent baseline in AUC. However, using latent embeddings is still better in every other case. This is evidence that the concept embedding does not carry as much visual information as the embedding generated using a pre-trained DNN.

Table 6.5. Codes used to represent embedding configurations in this chapter. Models are coded as a number of 4 digits, each corresponding to one of the 4 criteria. For instance, model coded as 2010 means that Criteria 1 was chosen to be "Categorical", Criteria 2 "Unique detectors only", and so on.

Criteria	**Code**	**Value**
Unit score	0	Discrete
	1	Absolute
	2	Categorical
Unique detectors	0	UD only
	1	Every unit
Layer weight	0	Equal weight
	1	Prop. to UD
	2	Prop. to AU
	3	Inv. to UD
	4	Inv. to AU
Aggregate units	0	Addition
	1	Average

Table 6.6. AUC, Mean Reciprocal Rank (MRR), Recall (R), Precision (P), nDCG (N) at different recommendation list lengths (20, 100, 200), in UGallery using VisRank. First row contains the results of a latent feature embedding. Last row contains the results of a random recommender. Embedding versions are codified using the notation of Table 6.5. Our concept embeddings outperformed the latent version only in the AUC metric.

Configuration	**AUC**	**MRR**	**R@20**	**P@20**	**N@20**	**R@100**	**P@100**	**N@100**	**R@200**	**P@200**	**N@200**
Latent	*0,66714*	*0,03424*	*0,07326*	*0,00440*	*0,04312*	*0,12722*	*0,00155*	*0,05420*	*0,16431*	*0,00101*	*0,05961*
v.2001	**0,67723**	0,02312	0,04888	0,00282	0,02808	0,10359	0,00122	0,03824	0,13958	0,00084	0,04426
v.2100	**0,67710**	0,02477	0,04510	0,00254	0,02867	0,08956	0,00102	0,03670	0,12459	0,00074	0,04290
v.2011	**0,67036**	0,02294	0,05048	0,00288	0,02830	0,10703	0,00124	0,03877	0,13130	0,00078	0,04310
v.2110	**0,66933**	0,02425	0,04144	0,00227	0,02725	0,09364	0,00103	0,03647	0,12768	0,00074	0,04239
v.2031	**0,67801**	0,02453	0,04922	0,00282	0,02929	0,09558	0,00111	0,03862	0,13725	0,00081	0,04530
v.2130	**0,68153**	0,02343	0,05082	0,00282	0,02850	0,09494	0,00110	0,03685	0,12402	0,00073	0,04165
Random	0,49868	0,00066	0,00137	0,00007	0,00032	0,00904	0,00011	0,00200	0,01087	0,00007	0,00239

Table 6.7. AUC, Mean Reciprocal Rank (MRR), Recall (R), Precision (P), nDCG (N) at different recommendation list lengths (20, 100, 200), in Wikimedia using VisRank. First row contains the results of a latent feature embedding. Last row contains the results of a random recommender. Embedding versions are codified using the notation of Table 6.5. Our concept embeddings did not outperform the latent version in any metric.

Configuration	**AUC**	**MRR**	**R@20**	**P@20**	**N@20**	**R@100**	**P@100**	**N@100**	**R@200**	**P@200**	**N@200**
Latent	*0,59449*	*0,01135*	*0,02006*	*0,00116*	*0,01290*	*0,03390*	*0,00040*	*0,01557*	*0,04735*	*0,00027*	*0,01755*
v.2001	0,58772	0,01069	0,01628	0,00094	0,01160	0,02946	0,00035	0,01414	0,04656	0,00027	0,01662
v.2100	0,59210	0,01063	0,01724	0,00096	0,01182	0,03200	0,00038	0,01469	0,04260	0,00025	0,01624
v.2011	0,58539	0,01083	0,01885	0,00105	0,01226	0,03062	0,00036	0,01439	0,04106	0,00024	0,01616
v.2110	0,58932	0,00951	0,01584	0,00090	0,01052	0,03164	0,00038	0,01380	0,04333	0,00026	0,01550
v.2031	0,58892	0,00951	0,01512	0,00085	0,01039	0,03363	0,00039	0,01380	0,04316	0,00025	0,01528
v.2130	0,59315	0,01009	0,01783	0,00101	0,01157	0,03069	0,00036	0,01409	0,03858	0,00023	0,01523
Random	0,49868	0,00066	0,00137	0,00007	0,00032	0,00904	0,00011	0,00200	0,01087	0,00007	0,00239

6.2.2. Performance of embeddings in DNNs

Table 6.8 and Table 6.9 show the results of both VBPR and CuratorNet models using the concept embeddings (chosen in the previous section) to recommend on UGallery and Wikimedia. In both recommendation tasks our embeddings performed quite well and in some cases even outperformed the results achieved by a latent embedding. These results are expected to be better than in Table 6.6 and Table 6.7 because the model displayed in the latter (VisRank) only recommends based on visual features and is not personalized, while the models used in the former (VBPR and CuratorNet) learn from interactions, a Collaborative Filtering (CF) approach, and perform personalized recommendation.

UGallery. Results of VBPR and CuratorNet are shown in Table 6.8. Both models performed similarly in all the metrics, but VBPR performed slightly better than CuratorNet for the most part. In VBPR, all the tested concept embeddings outperformed their latent counterpart in all metrics, except Recall and Precision at 200. We think that this is explained in part due to VBPR exploiting non-visual patterns, being able to rely on visual features and interactions in a more equitable manner.

Wikimedia. Results of VBPR and CuratorNet are shown in Table 6.9. Unlike UGallery, our concept embeddings were not able to outperform significantly the latent baselines. In VBPR, some of the concept embeddings outperformed their latent counterpart in some of the metrics, but for the most part, the best model was VBPR trained with a latent embedding. In CuratorNet, none of the tested concept embeddings was able to beat the latent embedding version in any of the metrics.

Table 6.8. AUC, Mean Reciprocal Rank (MRR), Recall (R), Precision (P), nDCG (N) at different recommendation list lengths (20, 100, 200), in UGallery using DNN models. First row in each section contains the results of a latent feature embedding. Last row of the table contains the results of a random recommender. Embedding versions are codified using the notation of Table 6.5. Our concept embeddings outperformed their latent counterparts in multiple metrics.

Model	**Configuration**	**AUC**	**MRR**	**R@20**	**P@20**	**N@20**	**R@100**	**P@100**	**N@100**	**R@200**	**P@200**	**N@200**
VBPR	Latent	*.71603*	*.05241*	*.12211*	*.00728*	*.06874*	*.17988*	*.00214*	*.07924*	*.21850*	*.00135*	*.08574*
VBPR	v.2001	**.72111**	**.06052**	**.12867**	**.00762**	**.07614**	**.18610**	**.00228**	**.08775**	.21538	.00133	**.09243**
VBPR	v.2100	**.71781**	**.06302**	**.13004**	**.00783**	**.07858**	**.19228**	**.00231**	**.09099**	.21587	.00131	**.09458**
VBPR	v.2011	**.72242**	**.05768**	**.13188**	**.00783**	**.07449**	**.18908**	**.00228**	**.08604**	.21500	.00133	**.09030**
VBPR	v.2110	**.71606**	**.06111**	**.13451**	**.00810**	**.07839**	**.19228**	**.00234**	**.08985**	**.22193**	.00135	**.09434**
VBPR	v.2031	**.71902**	**.05923**	**.12803**	**.00769**	**.07499**	**.18473**	**.00225**	**.08655**	.20962	.00129	**.09044**
VBPR	v.2130	**.71964**	**.06505**	**.13600**	**.00817**	**.08180**	**.19297**	**.00232**	**.09320**	.21724	.00133	**.09669**
CuratorNet	Latent	*.72226*	*.03681*	*.09499*	*.00563*	*.04998*	*.16004*	*.00192*	*.06325*	*.19625*	*.00122*	*.06848*
CuratorNet	v.2001	.71652	.03313	.09333	.00536	.04541	**.16340**	**.00196**	.06007	.18989	.00117	.06440
CuratorNet	v.2100	.71619	**.03738**	**.09772**	**.00570**	**.05095**	**.16805**	**.00199**	**.06458**	**.20724**	**.00125**	**.07000**
CuratorNet	v.2011	.71476	.03315	**.09802**	**.00563**	.04705	.14973	.00177	.05790	.18938	.00117	.06433
CuratorNet	v.2110	.71786	.03672	**.09757**	**.00584**	**.05173**	**.16345**	**.00201**	**.06431**	**.19950**	.00121	**.06952**
CuratorNet	v.2031	.71071	.03449	.09413	.00536	.04753	**.16073**	**.00194**	.06128	**.20694**	**.00128**	.06822
CuratorNet	v.2130	.71138	**.03881**	**.10212**	**.00604**	**.05228**	**.17421**	**.00210**	**.06632**	**.20904**	**.00126**	**.07127**
Random	Random	.49868	.00066	.00137	.00007	.00032	.00904	.00011	.00200	.01087	.00007	.00239

Table 6.9. AUC, Mean Reciprocal Rank (MRR), Recall (R), Precision (P), nDCG (N) at different recommendation list lengths (20, 100, 200), in Wikimedia using DNN models. First row in each section contains the results of a latent feature embedding. Last row of the table contains the results of a random recommender. Embedding versions are codified using the notation of Table 6.5. Our concept embeddings performance is close to the performance of the VBPR latent version, but could not outperform CuratorNet in any metric.

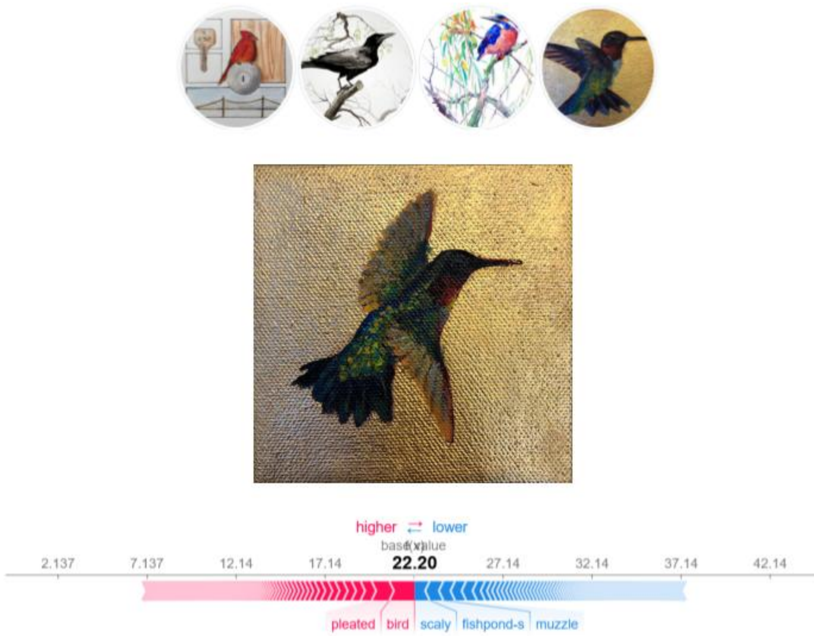
Model	**Configuration**	**AUC**	**MRR**	**R@20**	**P@20**	**N@20**	**R@100**	**P@100**	**N@100**	**R@200**	**P@200**	**N@200**
VBPR	Latent	*.65272*	*.00664*	*.01631*	*.00092*	*.00813*	*.04299*	*.00049*	*.01315*	*.05929*	*.00034*	*.01563*
VBPR	v.2001	**.65285**	.00593	.01406	.00079	.00717	.04000	.00046	.01192	.05863	**.00034**	.01462
VBPR	v.2100	**.65662**	.00599	.01567	**.00092**	.00760	.03715	.00045	.01184	.05746	**.00034**	.01478
VBPR	v.2011	.64688	.00507	.01229	.00072	.00599	.03662	.00042	.01058	.05366	.00031	.01310
VBPR	v.2110	**.65671**	.00556	.01391	.00083	.00689	.03737	.00044	.01143	.05731	.00033	.01440
VBPR	v.2031	**.65449**	.00603	.01262	.00072	.00669	**.04372**	**.00050**	.01240	.05901	**.00034**	.01473
VBPR	v.2130	**.65348**	**.00675**	.01437	.00088	.00793	.03788	.00046	.01258	.05688	.00033	.01529
CuratorNet	Latent	*.62556*	*.00745*	*.01759*	*.00101*	*.00919*	*.03631*	*.00042*	*.01271*	*.05539*	*.00032*	*.01555*
CuratorNet	v.2001	.62197	.00422	.00833	.00050	.00462	.02535	.00030	.00776	.04179	.00023	.01007
CuratorNet	v.2100	.62433	.00607	.01297	.00079	.00716	.03230	.00037	.01064	.04559	.00026	.01257
CuratorNet	v.2011	.61500	.00488	.01249	.00070	.00607	.02514	.00030	.00858	.03719	.00022	.01044
CuratorNet	v.2110	.62287	.00577	.01116	.00068	.00649	.03127	.00037	.01026	.04545	.00026	.01220
CuratorNet	v.2031	.62081	.00486	.00930	.00057	.00555	.02872	.00035	.00925	.03982	.00024	.01075
CuratorNet	v.2130	.61636	.00642	.01187	.00072	.00705	.02879	.00033	.01019	.04351	.00025	.01248
Random	Random	.49868	.00066	.00137	.00007	.00032	.00904	.00011	.00200	.01087	.00007	.00239

6.3. Explanations In Terms of Visual Concepts

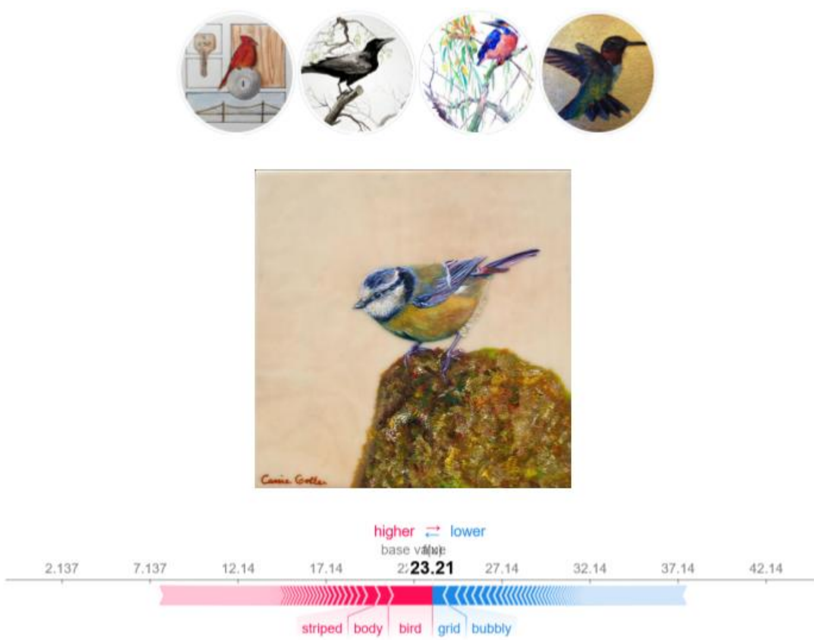
In this section, we will explore the explanations provided by the recommendation system trained using one of the concept embeddings. All the examples will be explanations of recommendations retrieved from a CuratorNet model trained on UGallery, with configuration coded as 2110 (Table 6.5 provides a description of each of the codes).

For each displayed recommendation, the items already consumed by the user are displayed on top, the middle image is the recommended item, and the explanation using the SHAP library is at the bottom of each figure. The SHAP plot that explains the recommendation, attributes each input concept an importance value by analyzing the model internals and modeling how the presence (or absence) of a feature changes the output of the model. In the plot, features in red color influence positively, i.e. drag the prediction value to a higher score, while features in blue color do the opposite. In general, explanations proved to be related to expected concepts, but sometimes unexpected concepts may appear.

Figures 6.1 and 6.2 show examples of recommendation and explanation for two profiles. The first example, figure 6.1, corresponds to the profile of a user that has selected items that contain “Bird” in them. SHAP correctly identifies this concept, but other unrelated concepts also receive a positive attribution without being visibly connected in the image under our perception. The second example, figure 6.2, corresponds to another user profile, where the user has selected images with “Sea” in them.

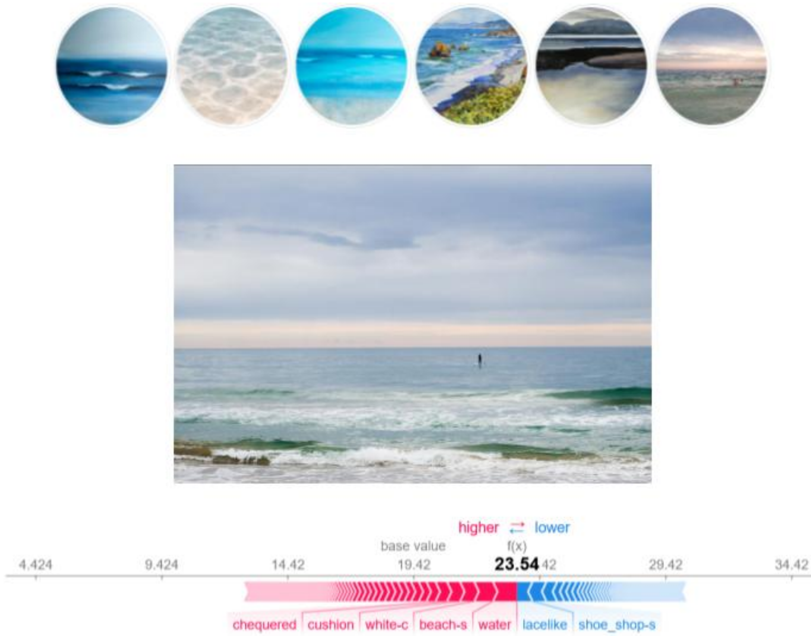


(a) The SHAP visual explanation correctly assigns a positive importance to “Bird”, but also identifies “Pleated” with a high attribution value.

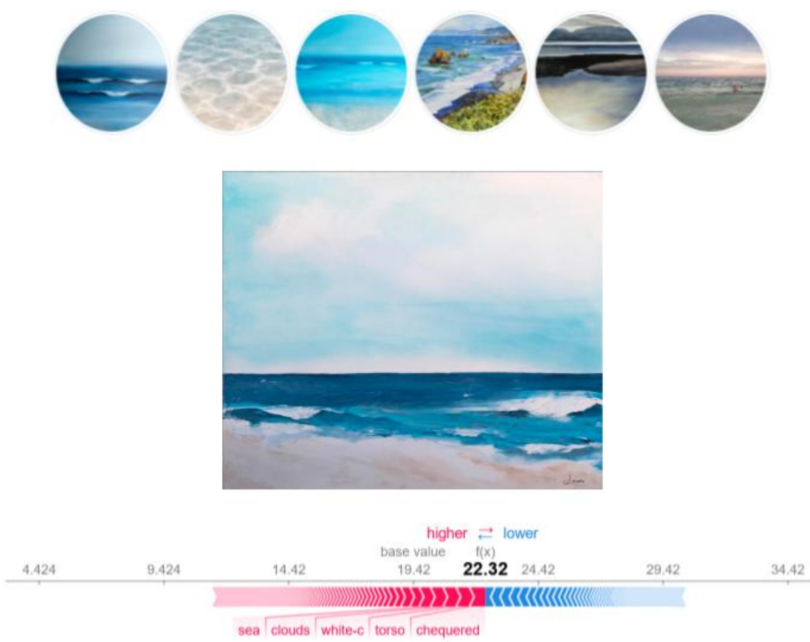


(b) The SHAP visual explanation correctly assigns a positive importance to “Bird”, but other irrelevant concepts appear, such as “Body”.

Figure 6.1. Recommendation and explanation for a user that consumed images containing birds. From top to bottom, each image displays: (1) the consumed images, (2) the recommended item, and (3) a visual explanation provided by SHAP.



(a) The SHAP visual explanation correctly assigns a positive importance to "Water", "Beach" (scene), and "White", but some of the other concepts do not seem to be related to the image.



(b) The SHAP visual explanation correctly assigns a positive importance to "White", "Clouds", and "Sea", but some of the other concepts do not seem to be related to the image.

Figure 6.2. Recommendation and explanation for a user that consumed images containing water and coastlines. From top to bottom, each image displays: (1) the consumed images, (2) the recommended item, and (3) a visual explanation provided by SHAP.

6.4. Discussion

In this section, we will explore the research questions established in section 3.1.

6.4.1. RQ1. Is it possible to build a concept-based item representation?

Our results and analysis indicate that it is possible to build an interpretable item representation based on visual concepts. We were able to treat the data as tabular data and apply traditional preprocessing methods that helped to improve the performance on a baseline model. Even if our performance in the baseline model did not match the results of a latent feature embedding, we were still able to confirm that building a recommendation system using our proposed representation was possible.

6.4.2. RQ2. Can we deliver accurate recommendation using concept-based representations?

The models trained using our proposed concept embeddings performed similarly to models trained using the traditional approach (feature embeddings from pre-trained neural networks) when used to train DNNs. Results are better than expected considering the initial results in our baseline task (mentioned in RQ1). The comparison must be interpreted carefully because of the unusual nature of the datasets (UGallery dataset contains many one-of-a-kind items while Wikimedia dataset has not been studied in a recommendation task), but results are still comparable. In the context of this work, we empirically demonstrated that it is possible to develop explainable visually-aware recommender systems with a performance similar to the non-explainable traditional model.

6.4.3. RQ3. Can we provide explanations of recommendations in terms of visual concepts?

Our interpretable representation allows the models developed using our framework to be compatible with feature attribution techniques. This approach is not possible (or at least, useful) in DNN models trained with a pre-trained neural network. In our framework, we propose using a feature attribution technique (such as SHAP) to deliver explanations of a visually-aware recommendation system using visual concepts.

7. FUTURE WORK

In this study, we evaluated our concept-based item representations using offline evaluation metrics. Compared to traditional non-explainable visual recommender systems, our proposal showed competitive results in every metric used. An interesting aspect to consider is how our concept-based item representations align with human perception. In this aspect, users' perception is important because users must make sense of the explanations in terms of visual concepts. Even if a feature attribution method can explain a particular outcome, users might need to understand the meaning of the visual concepts and how the generation of explanations works. It is necessary to evaluate the item representations and the explanations delivered by the system in a user study.

Another aspect that would be interesting to study is how recommendations change when using a concept embedding. Diversity, coverage, novelty, serendipity are some properties that a user-centric or recommendation-centric evaluation could measure. Is diversity affected positively, negatively, or is it not affected? Does the coverage change significantly? What happens in terms of serendipity and novelty of the recommendations? These relevant properties might be different when our framework is applied.

The last idea for future work is to study if it is possible to develop a performant concept-based model. Visual concepts are not always independent factors, sometimes they're related, and their relations can also provide valuable information. With an appropriate item representation and model architecture, it might be possible to exploit the relations between concepts to achieve better performance without losing interpretability.

8. CONCLUSIONS

In this work, we have studied the possibility of generating an interpretable embedding, based on visual concepts, and training visually-aware recommendation systems to propose a framework to develop explainable visually-aware recommendation systems.

Our first result, aligned with our first research question, shows that an interpretable concept-based item representation is achievable by our extension of NetDissect. Because NetDissect aligns units in a convolutional layer to visual concepts, our representation will often miss several concepts that do not align with any unit. In these cases, the concept information will be discarded in our preprocessing due to having zero variance. This phenomenon should be further considered and studied to develop more robust and representations, that cover more visual concepts.

Regarding RQ2, we see that our concept embeddings show competitive results in state-of-the-art models using an interpretable representation instead of the latent representation used in a traditional pipeline. In the future we will expand this analysis to other datasets and models, to generalize our results and confirm the performant results obtained in this study. Also, we could develop a novel model designed to exploit the visual concepts information specifically.

Finally, concerning RQ3, our resulting proposed models can deliver explanations through known feature attribution methods, particularly we used SHAP. In future work, we would like to study if the explanations based on visual concepts actually make a difference in users' perception and also evaluate how accurate are our item representations compared to human perception.

REFERENCES

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, *6*, 52138–52160.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., … others (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, *58*, 82–115.

Balog, K., Radlinski, F., & Arakelyan, S. (2019). Transparent, scrutable and explainable user models for personalized recommendation. In *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval* (pp. 265–274).

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 6541–6549).

Chen, J., Zhang, H., He, X., Nie, L., Liu, W., & Chua, T.-S. (2017). A entive collaborative filtering: Multimedia recommendation with item-and component-level a ention. In *Proceedings of the 40th international acm sigir conference on research and development in information retrieval.*

Chen, X., Chen, H., Xu, H., Zhang, Y., Cao, Y., Qin, Z., & Zha, H. (2019). Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation. In *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval* (pp. 765–774).

Cramer, H., Evers, V., Ramlal, S., Van Someren, M., Rutledge, L., Stash, N., … Wielinga, B. (2008). The effects of transparency on trust in and acceptance of a content-based art recommender. *User Modeling and User-adapted interaction*, *18*(5), 455.

Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2011). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, *20*(1), 30–42.

Das, A., & Rad, P. (2020). Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*.

Doran, D., Schulz, S., & Besold, T. R. (2017). What does explainable ai really mean? a new conceptualization of perspectives. *arXiv preprint arXiv:1710.00794*.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Ferwerda, B., Swelsen, K., & Yang, E. (2018). Explaining content-based recommendations. *New York*, 1–24.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, *51*(5), 93.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

He, R., & McAuley, J. (2016). Vbpr: visual bayesian personalized ranking from implicit feedback. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 30).

Hohman, F., Park, H., Robinson, C., & Chau, D. H. P. (2019). S ummit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE transactions on visualization and computer graphics*, *26*(1), 1096–1106.

Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *2008 eighth ieee international conference on data mining* (pp. 263–272).

Kang, W.-C., Fang, C., Wang, Z., & McAuley, J. (2017). Visually-aware fashion recommendation and design with generative image models. In *2017 ieee international conference on data mining (icdm)* (pp. 207–216).

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, *53*(8), 5455–5516.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., & Sayres, R. (2017). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *arXiv preprint arXiv:1711.11279*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

La Cascia, M., Sethi, S., & Sclaroff, S. (1998). Combining textual and visual cues for content-based image retrieval on the world wide web. In *Proceedings. ieee workshop on content-based access of image and video libraries (cat. no. 98ex173)* (pp. 24–28).

Lin, Y., Ren, P., Chen, Z., Ren, Z., Ma, J., de Rijke, M., et al. (2018). Explainable fashion recommendation with joint outfit matching and comment generation. *arXiv preprint arXiv:1806.08977*, *2*.

Lundberg, S., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

Messina, P., Cartagena, M., Cerda, P., del Rio, F., & Parra, D. (2020). Curatornet:

Visually-aware recommendation of art images. *arXiv preprint arXiv:2009.04426*.

Messina, P., Dominguez, V., Parra, D., Trattner, C., & Soto, A. (2019). Content-based artwork recommendation: integrating painting metadata with neural and manually-engineered visual features. *User Modeling and User-Adapted Interaction*, *29*(2), 251–290.

Mu, J., & Andreas, J. (2020). Compositional explanations of neurons. *arXiv preprint arXiv:2006.14032*.

Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web* (pp. 325–341). Springer.

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2012). Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 acm conference on computer supported cooperative work* (pp. 175–186).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1135–1144).

Rui, Y., Huang, T. S., Ortega, M., & Mehrotra, S. (1998). Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on circuits and systems for video technology*, *8*(5), 644–655.

Schafer, J. B., Konstan, J., & Riedl, J. (1999). Recommender systems in e-commerce. In *Proceedings of the 1st acm conference on electronic commerce* (pp. 158–166).

Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning important features through propagating activation differences. In *International conference on machine learning* (pp. 3145–3153).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smeulders, A. W., Worring, M., Santini, S., Gupta, A., & Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions on pattern analysis and machine intelligence*, *22*(12), 1349–1380.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1631–1642).

Tintarev, N., & Masthoff, J. (2015). Explaining recommendations: Design and evaluation. In *Recommender systems handbook* (pp. 353–382). Springer.

Tjoa, E., & Guan, C. (2020). A survey on explainable artificial intelligence (xai): Toward medical xai. *IEEE Transactions on Neural Networks and Learning Systems*.

Vig, J., Sen, S., & Riedl, J. (2009). Tagsplanations: explaining recommendations using tags. In *Proceedings of the 14th international conference on intelligent user interfaces* (pp. 47–56).

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, *52*(1), 1–38.

Zhang, Y., & Chen, X. (2018). Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192*.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2014). Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*.