Image Recognition for the Exploration of Historical Photograph Collections

by

Aaron Maier

Degree Course: Informatics B.Sc.

Matriculation Number: 2054222

Institute of Applied Informatics and Formal Description Methods (AIFB)

KIT Department of Economics and Management

FIZ Karlsruhe

Advisor:	Prof. Dr. Ralf H. Reussner
Second Advisor:	Prof. Dr. Harald Sack
Supervisor:	Tabea Tietz

Abstract

The digitization of archival resources has been an ongoing process for many years, producing large collections of cultural heritage data. Knowledge graphs have been used to structure this information, enabling new means of visualization and exploration. However, current visualizations for less tangible data, such as music or the performing arts, usually only contain sparse metadata, which limits their explorative capabilites.

In this thesis, new metadata based on the contents of historical images will be generated. To this end, a pipeline is introduced that automatically generates object detection results with a state of the art object detection model and converts them into a valid Resource Description Framework (RDF) representation. Additionally, the images are also linked to Wikidata entities, based on the object classes that they contain, using a custom Entity Linking algorithm. Statistics on the final results are generated and used to evaluate their validity in the larger context of the Linked Stage Graph. Furthermore, new explorative search capabilities are showcased, by providing sample SPARQL queries that aggregate images and stage plays based on the new data.

Contents

1 Introduction

A lot can be learned from exploring and researching our history as a society. Cultural heritage is a central pillar of understanding the history of different cultures over time. It envelops commonly known cultural aspects, such as buildings, books and art, but also valuable intangible aspects, such as religious ceremonies, traditions and performing arts [3]. As such, making it more accessible to the public has been a goal of Galleries, Libraries, Archives and Museums (GLAM institutions) for many years. To achieve this, many institutions have been digitizing their archived data. As a result, large collections of cultural heritage data are available on the web in the form of digital archives [37]. However, many of these archives organize and present their data in a way that makes it hard to explore for users who are unfamiliar with the way information is usually structured when using archival practices [6].

In order to tackle this problem, Knowledge Graphs have been utilized to structure the data. Every resource contained in these Graphs can be uniquely identified using a Unique Resource Identifier (URI). Furthermore, relations between resources and hierarchical structures can be provided with an ontology, which increases the semantic expressivity of the data. Moreover, Knowledge Graphs enable logical inferences and reasoning, which can be utilized to describe more complex and even implicit relations between resources.

This re-structured data can then be used to visualize the information in a more user-friendly way. The form of the visualization and the way users are expected to interact with it depend greatly on the target user audience and are often times tailored to a specific use-case.

Problem

Currently, there is a significant lack of visualized information about less tangible data, such as music, linguistic entities (e.g. poems) and the performing arts, when compared to more tangible assets such as tools or artwork [37]. The metadata associated with these types of cultural heritage is usually sparse, which would limit the explorative search capabilities of possible visualizations [34].

The focus of this bachelor's thesis will be historical photographs of the performing arts. More specifically, 7000 grayscale photographs of stage plays from the *"Stuttgarter Staatstheater"* from 1890 to 1940 will be used, which have been integrated into and afterwards visualized using an existing Knowledge Graph called *"Linked Stage Graph"* [35]. These photographs include scenes from theatre, opera and ballet performances, as well as backstage moments[1] and the theatre buildings themselves.

[1]Costume design, prop making and stage building

Motivation

Since the visualization is based on images, state-of-the-art deep learning models for image recognition can be employed to generate additional metadata. Moreover, the information is stored in a Knowledge Graph and can be used to interlink the images and even connect them to other already existing knowledge bases, which will enrich their metadata even more. The main focus, in regards to image recognition, will be on object detection models. These can be used to generate information about the many various elements inside these photographs, such as actors, props, the stage design and also the architecture of the theatre buildings. Object detection, in contrast to image classification, can unambiguously identify each detected object, the quantity of objects of the same type and additional information, such as the position inside the image.

Of course, certain problems can arise if current object detection models are used. These models were trained on modern color images, which could potentially decrease the accuracy of the detection on the grayscale images. Furthermore, objects that did not exist when the photographs were taken could potentially be detected. Therefore, Object detection, it's challenges and possible solutions will make up the first part of this bachelor's thesis. After useful information has been generated, it will then be integrated into the existing Knowledge Graph and lastly, improvements to the current explorational features will be examined.

Contribution

This bachelor's thesis will contribute structured metadata for historical theatre photographs, which can be integrated into an existing Knowledge Graph and will detail new means of exploration for theatre scientists, historians and the general public, based on the new data.

Chapter 2 outlines similar projects that aim at enriching the metadata for historical photograph collections and projects that link important snippets from (fragments of) texts against possibly related Wikidata entities. Additionally, it will contain a more in-depth description of the *Linked Stage Graph*, that contains the images used in this thesis. Chapter 3 describes the process of generating usable data in RDF format and integrating it into an existing Knowledge Graph. First, a suitable object detection model is selected with the help of a user study. Afterwards, the model is used to generate metadata for detected objects on all 7000 photographs, including the object class, the confidence score and the bounding box. This information is then translated into a valid RDF representation, which can be imported into the Linked Stage graph. Chapter 4 contains an evaluation of the results that were achieved by this approach. This includes the results of the user study, the final detections with the chosen model, the correctness of the linked Wikidata entities and the validity of the RDF data. Chapter 5 discusses how well the results from the chosen approach were able to solve the underlying problem of automatically generating metadata for historical photographs. Furthermore, sample queries are provided to showcase new means of exploration. Chapter 6 concludes the work with a summary of the results and their evaluations and provides an outlook on the further development and possible improvements of the project.

2 Related Work

This chapter contains more in-depth explanations of Related Work and preliminaries for this thesis.

2.1 Linked Stage Graph

The Linked Stage Graph, a Knowledge Graph that was created during the Coding da Vinci hackathon, serves as the foundation for the data of this thesis. It contains historical information and images about stage performances of the *"Stuttgarter Staatstheater"* [35] between 1890 and 1940s, such as names and performance date. Additionally, there are also images that are not connected to a certain play, but instead showcase backstage moments, props and theatre buildings. A visualization has been implemented that allows the user to explore the plays through related images. It only contains images that actually depict stage performances. They are arranged in a timeline, allowing the user to explore these plays by scrolling and clicking through images. Furthermore, a public SPARQL Protocol And RDF Query Language (SPARQL) endpoint exists to query the entirety of the data more freely.

2.2 Archival Data

In *"ONSTAGE, the Online Data System of Theatre in Amsterdam from the Golden Age to Today"* [1], Frans R. E. Blom et al. describe what kind of information is stored for the archival records of the *"Shouwburg"*, Amsterdam's most prominent public theatre venue. This includes data on the premier date and performance frequency of plays and account books, which also include information on the revenues. All of this information is linked to the original sources, which are available in a digitized format. Furthermore, they showcase how the archive can be used to research the history of the theatre. The data can, for example, be used to generate statistics for individual playwrights and on trends of the theatres cultural economics and business history. The project is also compared to similar databases for the french *"Comédie-Française Registers Project"* and the british *"The London Stage"* projects. Lastly, an outlook is given on the future of the database in respects to Linked Data principles and how they can be used to connect the database to other such archives.

In 2020, Floortje Bakkeren presented extensive insights into the history of development of the *"History and Contents of the Dutch Theatre Production Database"* [7]. This includes the software used to store the data, as well as the process of digitizing archival records. Furthermore, problems and challenges are explored, when working with large amounts of archival records that have to be kept semantically consistent over multiple references and how some of these were resolved.

The goal of the project *"Theadok"* [11] is to collect and enrich existing metadata about stage plays performed in Austrian theatres for research purposes. This includes information on actors, stages and performances. The data is stored in a performance database and structured with Extensible Markup Language (XML) files, based on the Lightweight Information Describing Objects (LIDO) schema and can be queried using the official website [20]. Additionally, a publicly accessible Application Programming Interface (API) is provided, which returns the requested data in JavaScript Object Notation (JSON) format.

2.3 Entity Linking

Finn Å. N. explored how images contained in the ImageNet collection can be linked to Wikidata entities [23]. ImageNet contains more than 14 million images that have been hand-annotated and all of which are linked against WordNet synsets. Nielsen investigates how these synsets can be used to link the images to entries in Wikidata, which would connect these two extensive knowledge graphs. Problems with discrepancies in semantics between classifications of images in ImageNet, their synsets in WordNet and the related entities in Wikidata and possible solutions are examined. Furthermore, statistics on the success rate of images that are linked to the correct entity and an example application are provided.

"OpenTapioca" is a project from 2020 by Antonin Delpeuch, that is aimed at linking entities in short texts against matching entities in Wikidata and showcases strengths and weaknesses of Wikidata as the main data source [5]. Comparisons are drawn between Wikidata and other Knowledge Graphs that are derived from Wikipedia, such as DBpedia and YAGO. The entity linker itself is trained on a Wikidata dump and can therefore be kept up to date in real time as Wikidata evolves [5]. It is important to note, that OpenTapioca only works with Named Entities that match a specific person, location or organization [21].

The *"Falcon 2.0"* project is a fully linguistic approach to linking entities and relations to Wikidata and is the successor to Falcon 1.0 [21]. It uses the English language model to determine entity and relation candidates in text and contains optimizations for the linking task [28]. The results were studied and it's performance was compared to baselines for other tools and Knowledge Bases. This project only links Named Entities and is intended to work best with short texts and questions [21].

3 Object Detection and Knowledge Graph Integration

The contribution of this thesis can be split into two main parts:

First, information has to be extracted from the photographs. Therefore, a user study was conducted to determine a suitable state of the art object detection model, which was used to generate information about the visual contents of the historical photographs.

Then, the information has to be integrated into an existing Knowledge Graph. To this end, the different object classes that were detected, are mapped to their respective Wikidata entities with a custom Entity Linking approach. This was done to enrich the semantic information of each detected object. Furthermore, a suitable RDF representation of the new information had to be constructed. It includes details about each detected object such as confidence score and bounding box coordinates, as well as the respective Wikidata entity.

3.1 Prerequisites

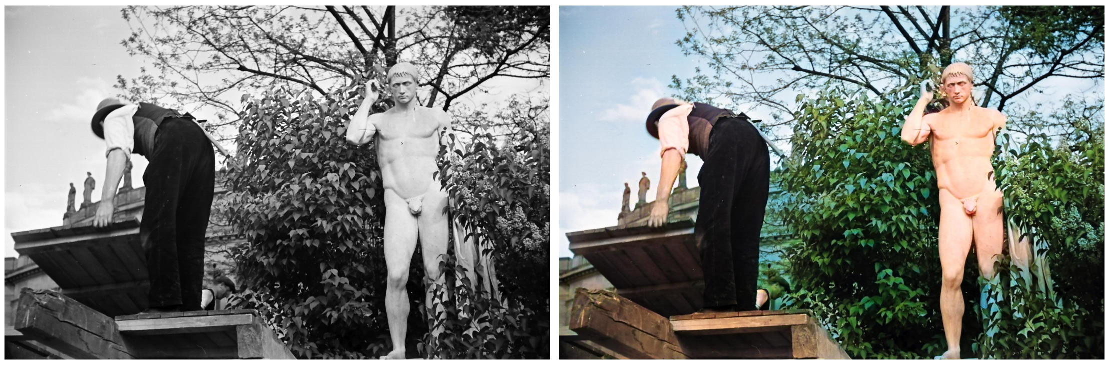
The main focus of this thesis is to expand the metadata of historical photographs by automatically extracting visual information. More specifically, around 7000 grayscale images of stage plays from the *"Stuttgarter Staatstheater"* from 1890 to 1940 were used. They were made available for the Coding Da Vinci hackathon in 2019 and are already part of an existing Knowledge Graph called Linked Stage Graph [35]. Depicted are mostly scenes from stage performances, as well as backstage moments[2] and a few theatre buildings. It is also worth noting, that the set also contains images that display descriptions for the stage plays in the form of text.

To get as close to the training data as possible, the images were colorized before running any object detection. This was done using *DeOldify* [12], which is an open source tool to colorize images and videos using deep learning. It is trained using so called *"NoGAN"* training, which is a modified version of Generative Adversarial Networks (GAN) training that was explicitly developed for this model [25]. It combines the benefits of GAN training while removing side effects like flickering in videos.

Comparing the detections on grayscale images and their colorized versions by hand, showed that colorizing the images improves detection rates for certain objects, such as plants and specific types of clothing. However, in certain edge cases the colors seem to be slightly off. One example of this can be seen in figure1 where the statue was colorized in skin color, instead of the usual marble color expected from such a statue.

For the User Study, the full set of images had to be scaled down. 80 sample images were

[2]Empty stages, Costumes, Rehearsals



(a) Original image (b) Colorized image

Figure 1: Comparison between grayscale and colorized image

chosen in total, 55 of which are images of stage performances. The remaining 25 images can be separated into 21 backstage moments and 4 images of entire theatre buildings, trying to mimic how each category is represented in the full set.

3.2 Object Detection

To extract visual information about the contents of an image automatically, a suitable object detection model had to be found. Training a model specifically for this use case, combined with the remaining tasks, would exceed the capacities for a bachelor's thesis. Therefore, the open source library *TensorFlow* [9] was chosen as the underlying framework for the object detection. It includes many state-of-the-art, pre-trained object detection models, which can be loaded directly from the TensorFlow Hub[3]. The object detection code was implemented in Python using *Jupyter*[4] notebooks and can be found on GitHub [5]. There are two primary notebooks for object detection. One works with multiple models and the sample images and includes functions to compare and display the different outputs. The other one works with a single model and all 7000 images and is used to produce the final results for a model. An example for the object detection can be seen in figure2.

A suitable model can be found by comparing multiple candidates with the help of a user study. After a single model was chosen, it will be used to infer objects on all 7000 images and its results are used to generate the final data, which can then converted into a valid RDF format and integrated into the Knowledge Graph.

[3]`https://tfhub.dev/s?module-type=image-object-detection`

[4]`https://jupyter.org/`

[5]`https://github.com/SolluxRules/bachelorarbeit-object-detection`

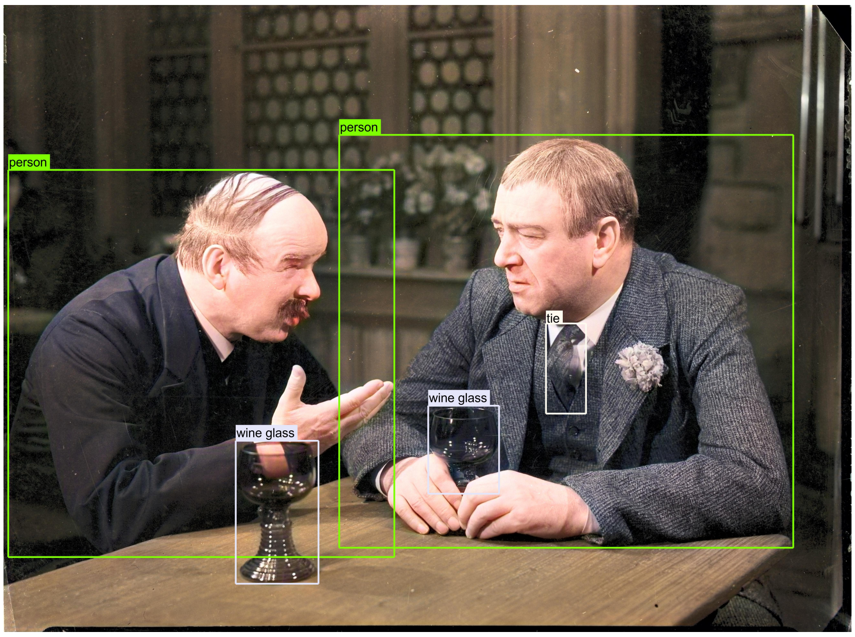


Figure 2: Example detections on an image

3.2.1 Preselection of Models

Since over 50 object detection models are hosted on the TensorFlow Hub, the selection of possible candidates had to be narrowed down to four models by hand. They were chosen to cover a variety of different technologies, architectures and training datasets.

First, all models were categorized based on their training datasets, since only two sets were available: Microsoft Common Objects in Context (MS COCO) and OpenImagesV4. There was only one model based on OpenImages that produced viable results. Therefore, it was chosen as the first candidate so that multiple training sets are represented. Afterwards, the models based on MS COCO were further categorized based on their underlying architecture. Inside each category, the models were compared and ranked by hand, based on direct comparisons between their results on the set of sample images, as well as their technical details. The model that produced the best results from each category was selected. Finally, one more comparison of all the remaining models, based on the same criteria, narrowed the selection down to four models. Figure3 shows the final selection[6].

Technologies

There are two different approaches to detecting and classifying objects on an image:

Two Stage is an approach employed by both Faster Region Based Convolutional Neural Network (R-CNN) models. The first step determines regions of interest that have a higher

[6]Initial evaluation was based on the model versions from the 22.06.2021

Model	Type	Architecture	Dataset
EfficientDet D7	One Stage	SSD with EfficientNet-b7 backbone + BiFPN feature extractor and focal loss	MS COCO 2017
CenterNet	One Stage	CenterNet with ResNet V1 101 feature extractor	MS COCO 2017
Faster R-CNN (1)	Two Stage	Faster R-CNN with Inception ResNet V2 feature extractor	MS COCO 2017
Faster R-CNN (2)	Two Stage	Faster R-CNN with Inception ResNet V2 feature extractor	OpenImages V4

Figure 3: Summary of all four pre-selected models

possibility of actually containing objects, which vastly decreases the amount of so-called "easy negatives" (regions in the image that don't contain any objects) from the image. In the second step, all remaining candidates are either labeled as background or as a foreground object class [26].

One Stage is an approach which both EffcientDet and CenterNet use. Here, the bounding box detection and the object recognition happen in the same step [19]. This removes the overhead of reducing all possible candidate locations beforehand, but results in a much larger set of total candidates.

Architectures

EfficientDet D7 is based on the existing EfficientNet-b7 [31] backbone with a modified Feature Pyramid Network (FPN) feature extractor, called Bi-directional Feature Pyramid Network (BiFPN). Unlike the conventional uni-directional approach of basic FPNs [16], BiFPN uses a bi-directional approach as shown in figure4, which allows for the introduction of learnable weights to determine the importance of different input features based on their importance to the output. To evaluate the extracted features, EfficientDet employs the Single Shot Multibox Detector (SSD) approach to determine an objects category

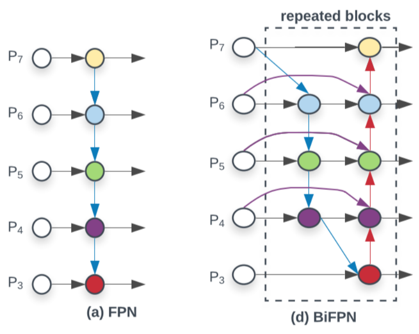


Figure 4: Comparison of typical FPN and BiFPN[32]

(foreground/background) and class, in case its a foreground category. SSD is based on a truncated base network for image classification that has been stripped of it's classification layers [19]. It allows for the detection of objects in varying sizes and aspect ratios with the help of multiple convolutional feature layers that each use different types of predictors. Since it is based on the single stage approach, both the classification of the object and the detection of its corresponding bounding box happen in the same step.

Because of their high sample rate of possible candidates, one stage models often have extreme imbalances in foreground classes and backgrounds [17]. This can lead to ineffective training, since the easy negatives completely overshadow the harder candidates. To deal with this imbalance, Focal Loss is implemented while training the model. Therefore, the training won't be based on the outliers (hard examples), but will instead be focused on the easy candidates and give them less weight. This automatically reduces the contribution of easy examples to the total loss, even if their number is much larger than that of the harder examples. Figure5 shows the complete architecture of the EfficientDet network and how the different parts work together.

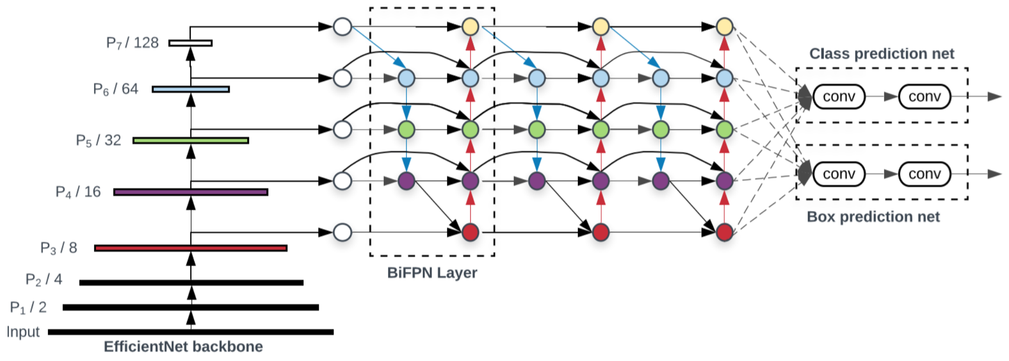


Figure 5: Complete EfficientDet architecture[32]

CenterNet represents objects as single points that are at the center of their bounding boxes. From this single keypoint, other properties such as size or even 3D-location/depth as well as pose keypoints can be inferred [38]. These points are found by running the image through a convolutional network which generates a heatmap, where each peak directly corresponds to an object center. All additional features can be directly inferred from image features at the location of these center points. It is based on a Residual Network (ResNet) backbone with 101 convolutional layers. This architecture applies learning residual functions which are parametrized by a layer's input to every few stacked layers, instead of learning an unreferenced function in each layer [10]. This is done to make use of the great accuracy that comes with deep neural networks, while eliminating the saturation and degradation of accuracy that usually comes with increasing depth.

Faster R-CNN models are advanced versions of their predecessor *"Fast R-CNN"*, which used algorithms implemented on the CPU (allowing shared computation) to generate the region proposals [8]. However, these algorithms were very slow when compared to efficient detection networks. Therefore, Region Proposal Networks (RPNs) were introduced, which could share the convolutional feature maps that were used by region-based detectors [26]. This makes region proposal nearly cost-free and much faster. Furthermore, both models that were used in the study use an Inception ResNet V2 backbone for feature extraction. Inception networks by themselves can be described as not only "deep", but also "wide". They apply multiple different filter-sizes in each layer, the results of which are afterwards combined using filter concatenation before feeding into the next layer [24]. A number of optimizations have been implemented to increase accuracy and reduce computational costs. These include factorization of filters from nxn to two nx1 and 1xn filters, which are a lot cheaper, as well as increasing the filter convolutions up to 7x7. Furthermore, the existing architecture was combined with residual learning, by replacing the filter concatenation process with residual connections. This greatly accelerated training of Inception type networks and also increased their accuracy [30].

Datasets

MS COCO is a dataset that was initially created by Microsoft in 2014. A new version was released in 2017, which three of the four selected models were trained on. It contains 80 object categories with over 5000 labeled instances per category that have been hand-annotated in over 300K images [18]. The images have been selected to contain objects of the given categories in non-icon views, increasing the recognition for known object types that are in the background or partially obscured. Furthermore, the images were selected to depict scenes, rather than isolated objects, to increase the usefulness for contextual reasoning and contain an average of 7.7 objects per image. Lastly, instead of using simple bounding boxes, the annotated objects are represented with a precise pixel-level segmentation.

OpenImage V4 includes a wider range of object categories. The fourth model was trained on this dataset. It includes 600 object classes with a total of over 15M labeled instances in almost 2M images [13]. Many of the images show complex scenes with an average of 8 annotated objects per image and also supports annotated visual relationships. The classes have been narrowed down from almost 20K classes contained in an internal dataset at Google called JFT. Unlike MS COCO, the images were not hand-annotated. Instead, an image classification approach was used to generate candidate labels for all images, which were then verified (either as "is present" or "is not present") by humans. These verifications were then in turn used to better train the classification model. Afterwards, the correct labels were again presented to humans who were tasked with manually inserting bounding boxes for each object in the image that matched these labels.

3.2.2 User Study

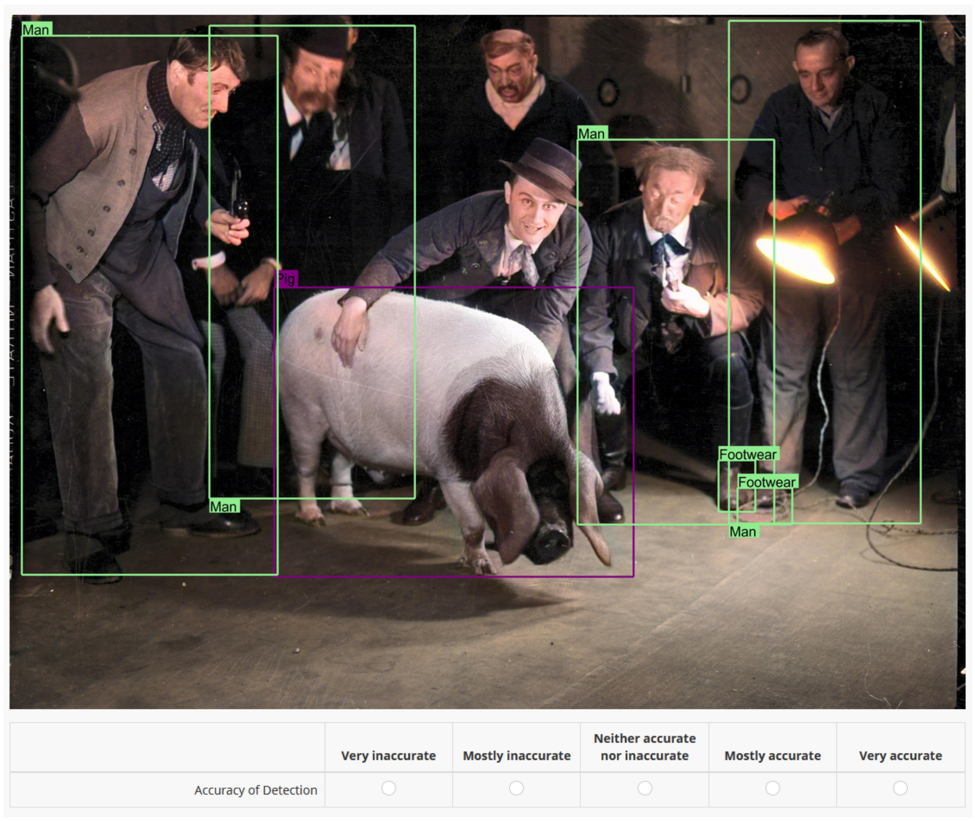
A User Study was conducted to determine which of the results produced by the four models is the most reliable. It included a total of 327 questions for the participants to answer, starting off with 5 questions pertaining to information about the participant, such as age or gender, that were collected anonymously. Afterwards, the participants were shown 320 images displaying the detections of the four models as seen in figure 6. Finally, there were two follow-up questions about the survey and how it was conducted.

The open source framework *LimeSurvey* [15] was used to conduct the user study on a website[7], which reduced the workload of setting up and maintaining the survey. Additionally, the results can be extracted in different formats, such as `Comma Separated Values (CSV)`, which simplifies their evaluation. Furthermore, advanced security features such as spam protection and restricting access to the survey are already included. To eliminate the workload of manually importing all the images into the framework, a script was developed that generates questions for each image of each model. The database entries for each question contain a unique numerical identifier that is incremented automatically. The script prefixes these identifiers with a shortcut that unambiguously connects the question to one of the four models, simplifying the process of generating statistics on a per model basis later on. For the user study, a minimum confidence score of 0.5 (50%) was selected as the threshold for all detections.

Study Design

The first part of the user study focused on information about the age and gender [14], as well as the educational background [29] and the current field of work or study [2] of

[7]`https://bachelorstudy.de`



	Very inaccurate	Mostly inaccurate	Neither accurate nor inaccurate	Mostly accurate	Very accurate
Accuracy of Detection	○	○	○	○	○

Figure 6: Example question from the survey

the participants. The main focus for finding suitable participants was on people who are involved in fields of study/employment that are related to this thesis, such as Cultural Heritage or Informatics. Therefore, most participants came from an official mailing list from the *Department of Informatics* at the Karslruhe Institute of Technology (KIT) and the staff at the *FIZ Karlsruhe – Leibniz-Institute for Information Infrastructure (FIZ)*. Additionally, the team involved in this study reached out to people in their social circles, since evaluating if a classification is correct or not does not require former knowledge of neural networks or other technical details. It also ensures that the survey captures a larger variety of participants with different backgrounds. Figures 7, 8, 9 and 10 showcase, that the survey participants matched our criteria.

The second part of the study contained images with the visualized detections of all four models. The participants had to rate the detections in each image on a 5-point Likert-Scale [33] from "very inaccurate" to "very accurate", based on how well the labels classified the actual objects inside the associated bounding box 6. Each image was evaluated separately and the order of appearance was randomized for each participant. The scale was chosen to provide room for a middle ground between completely accurate and not accurate at all, if images contained incorrect and correct classifications. Additionally, the 5-point scale provides a neutral position, in case images contained no detections at all. Testing this format showed, that it took around 1 minute and 50 seconds to rate 20 images.

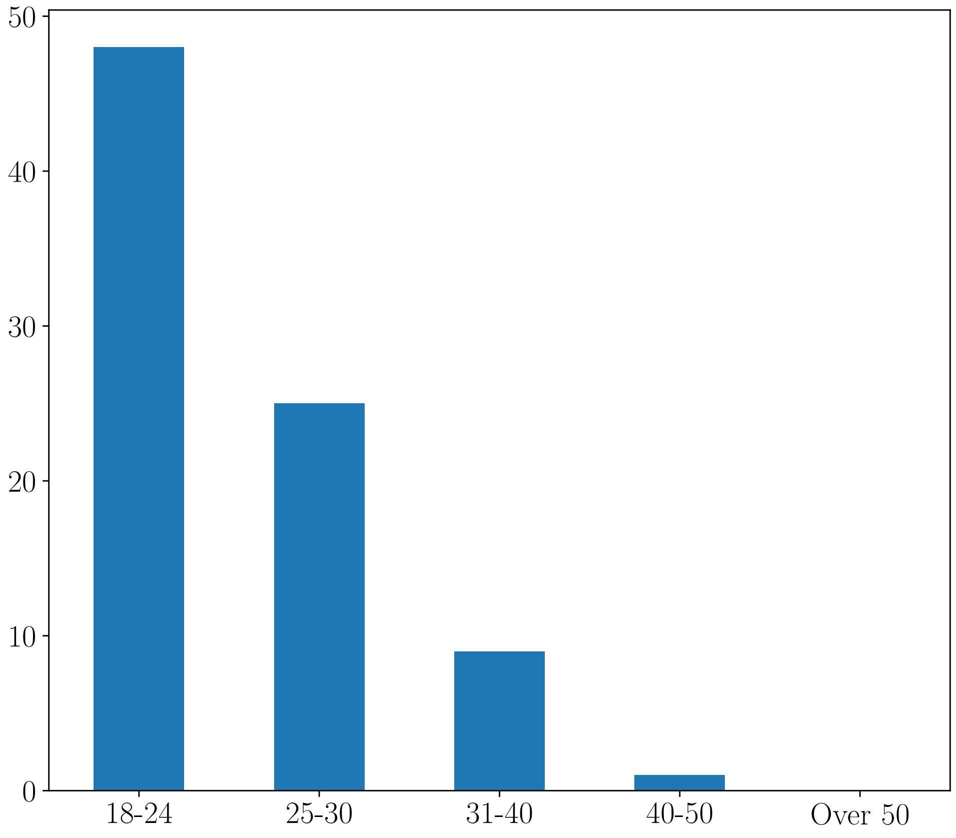


Figure 7: Participants by Age group

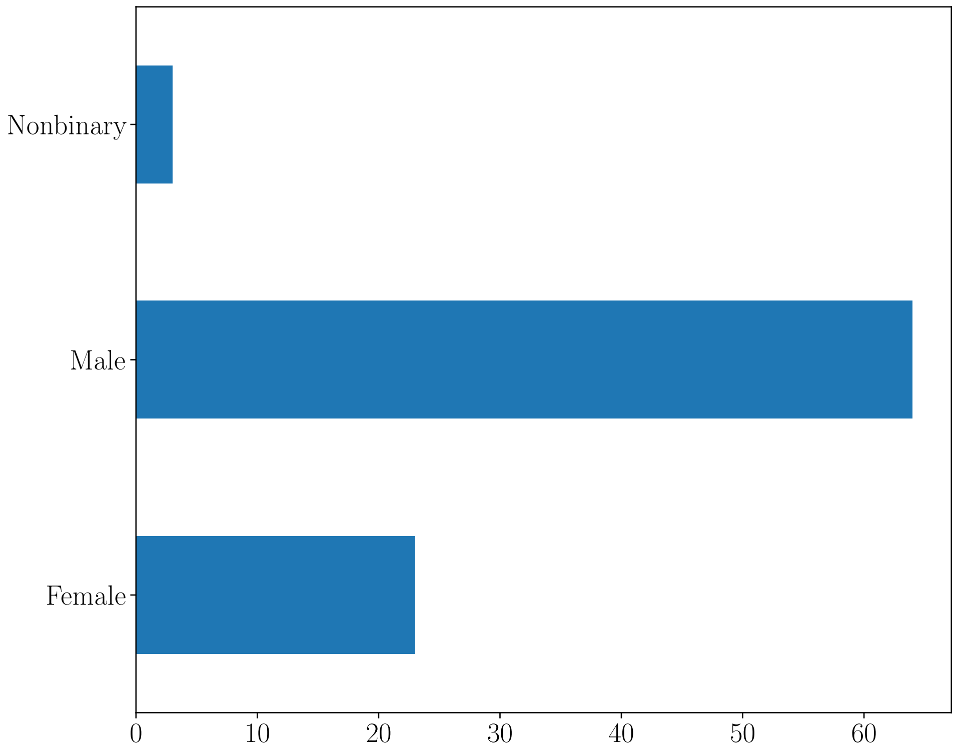


Figure 8: Participants by Gender

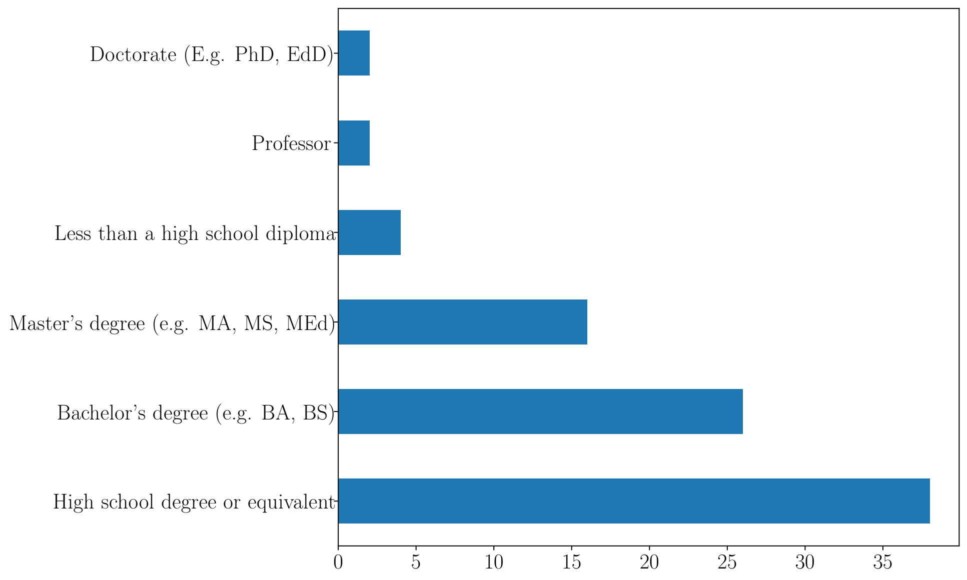


Figure 9: Participants by level of Education

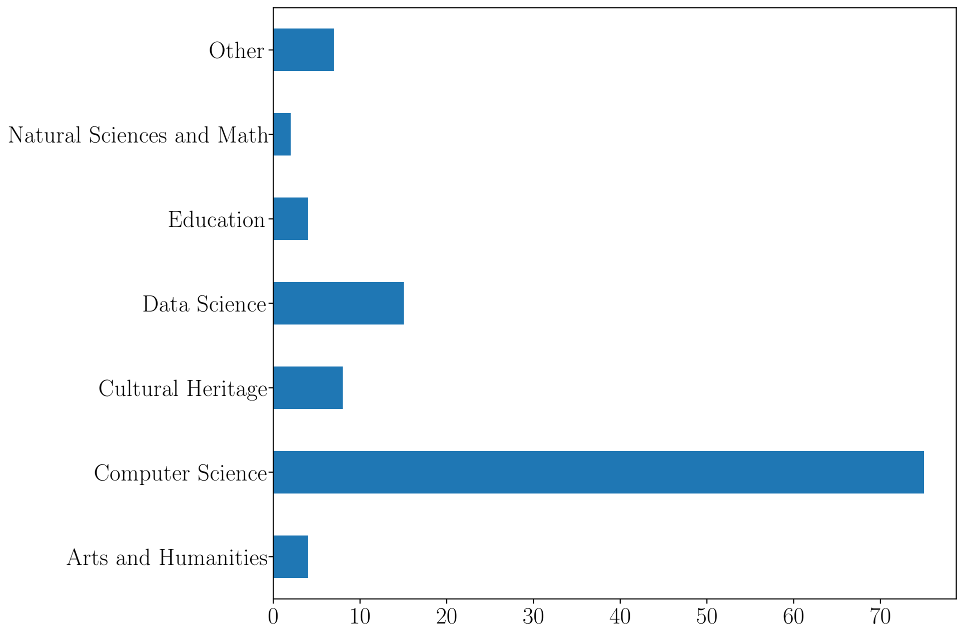


Figure 10: Participants by field of Employment/Study

To prevent participants from leaving their evaluations unfinished, it was determined that completing the survey should not take longer than 30 minutes. Therefore, a set of 80 samples was selected from the whole set of 7000 photographs. The sample images were chosen to best represent the entire set, putting the focus on displays of stage performances which take up a total of 55 out of the 80 images. The remaining images contain 21 backstage moments and only 4 images of theatre buildings, trying to recreate the ratios in the original set. Each model was used to generate detections on the samples, resulting in a total of 320 images, which should make the study around 30 minutes long.

The study concluded with two optional questions. In the first one, the participants were able to enter anything noteworthy about the detections that were displayed during the survey. The latter revolved around the study itself and if there were any problems with it's contents or the way it was conducted. These question were added to simplify the evaluation of possible problems with the study or the images themselves.

3.3 Knowledge Graph Integration

A second *Jupyter* notebook was created to generate the final detections. This notebook serves as a pipeline for a single model and a set of images. It generates detection results with the given model on all images and converts them into valid RDF data. Furthermore, it contains several optimizations to work with a large number of images, since the use-case of this thesis includes running object detection on several thousands of them.

To integrate the detections, they first have to be converted into a valid RDF representation. This includes linking all object classes that were detected on the images to their respective entries in the Wikidata knowledge base[8]. The final RDF data was initially supposed to be integrated into the already existing Linked Stage Graph, but due to issues with time restrictions for this thesis, this last step was not completed. Only the validity of the new RDF data was tested on a local machine, the results of which can be seen in the Evaluation section.

3.3.1 Wikidata Entity Linking

The RDF representation of a detection should include a link to the respective entity in Wikidata, based on the object class that was determined. For example, if an image contains one "Person", the RDF representation of this detection should include a link to the Wikidata entry for "Person"[9]. A visual example of this can be seen in figure14. This further enriches the metadata for each detection, because every relation of this entity in

[8] `https://www.wikidata.org/`

[9] `http://www.wikidata.org/entity/Q215627`

Wikidata (and other Knowledge Graphs that are linked to Wikidata) can then also be used to query and group the images.

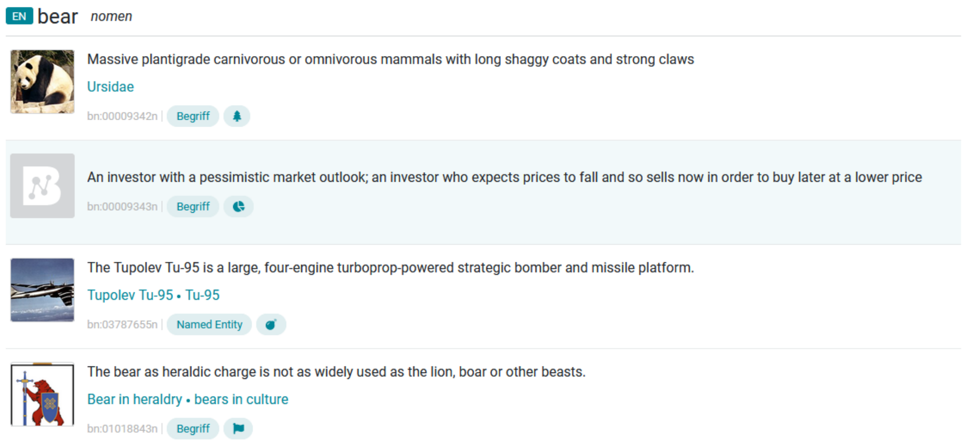
First off, each distinct class label found by the final model was collected. These usually consist of one to two words and are an abstract representation of objects or living beings, such as certain animals or humans. For example, the MS COCO dataset contains object classes, such as "Dining table", "Pig" and "Person" [18].

There already exist several approaches to linking text or fragments of text with entities from Wikidata [21], some of which are described in more detail in the Related Work section of this thesis. However, there are two main problems with these approaches. They either require the input text to have a specific format, such as *Falcon 2.0*, which was designed to work on questions and short texts [28]. Or they are limited to linking Named Entities, such as *OpenTapioca*, which only links parts of texts that correspond to a specific person, location or organization [5]. However, since the generated data cannot provide semantic or contextual information about the class labels and the classes themselves are abstract concepts, rather than Named Entities, these approaches are unfit for this use-case.

Therefore, a custom method was developed, which links these abstract concepts to Wikidata. This was achieved by using the *BabelNet* knowledge base [22], which is a multilingual encyclopedic dictionary that is linked to other large knowledge bases, such as WordNet, Wiktionary and of course Wikidata. Each entry in the dictionary has a "source" property, which includes all knowledge bases that have a known matching entry.

The first approach was to use the official BabelNet API, which can be accessed through a variety of different frameworks such as Python, Java or directly via Hyper Text Transfer Protocol (HTTP). For this purpose, the API was accessed via HTTP, since the Python framework is only commercially available. Although this might seem like the most straightforward approach, the API doesn't seem to order the matching entries based on relevancy, unlike the official search page. The results for the search term "bear" in the English language can be seen in figures11 and12 for the search page and the API respectively. The original results from the API have been extended with indices and a "Description" field, to increase readability. Comparing the two clearly shows a discrepancy in the order of the returned items. In case of the search page, the first and most relevant entry is actually the entry we are looking for. The API returns this same entry at index 31. This makes the API pretty much unusable for determining a matching entry.

Instead, a combination of both methods was used to link the object classes. First, a page crawler for the search page was developed. It collects all entries for a given search term that are "Nouns" and "Concepts", removing "Named Entities" and other word types, since they can never describe the entry we are looking for. Afterwards, information about each entry is requested via the API. It contains the name, description and also the set of



EN bear nomen

Massive plantigrade carnivorous or omnivorous mammals with long shaggy coats and strong claws
Ursidae
bn:00009342n | Begriff

An investor with a pessimistic market outlook; an investor who expects prices to fall and so sells now in order to buy later at a lower price
bn:00009343n | Begriff

The Tupolev Tu-95 is a large, four-engine turboprop-powered strategic bomber and missile platform.
Tupolev Tu-95 • Tu-95
bn:03787655n | Named Entity

The bear as heraldic charge is not as widely used as the lion, boar or other beasts.
Bear in heraldry • bears in culture
bn:01018843n | Begriff

Figure 11: Results for BabelNet search page

linked knowledge bases. All entries that do not contain a link to Wikidata are removed and the remaining entries' names are matched sequentially against the search term. If no entry can be found that matches the term exactly, the first entry will be used. The same logic applies if a matched entry contains multiple Wikidata sources. In this case, the algorithm tries to find a source that contains the class label and if none can be found, the first source is selected.

A number of optimizations have also been introduced, to reduce both the amount of requests to the search page, as well as the amount of API requests. Furthermore, to decrease the runtime of the algorithm the total amount of entries searched was also limited to four. Testing multiple of the detected object classes by hand with the search page showed, that if a match cannot be found within the first three to four entries, then there usually is no exact match. In the end, all matched entries with references to their Wikidata entity are exported as `CSV` and then validated by hand. If a mapping is incorrect or if there was no match, a correct match has to be supplied during validation. The validated mappings can then be re-imported and used during the generation of the RDF data.

3.3.2 RDF Data

Before the detections can be integrated into a Knowledge Graph, their information, such as the confidence score, the associated bounding box, etc. have to be represented in a valid RDF representation that can be linked to the already existing images. The structure for this representation was adapted and extended from the structure described in [27] and can be seen in figure14.

Nodes for plays and their depictions already exist in the Knowledge Graph and are connected via the `foaf:depiction` relation. The depiction nodes served as the anchor to attach the new information. A new node was added that contains the image filename without

```
[
    // ...
    /* Index 12 */ {
        "id": "bn:01018843n",
        "description": "Bear in heraldry",
        "pos": "NOUN",
        "source": "BABELNET"
    },
    // ...
    /* Index 31 */ {
        "id": "bn:00009342n",
        "description": "Ursidae",
        "pos": "NOUN",
        "source": "BABELNET"
    },
    // ...
    /* Index 41 */ {
        "id": "bn:03787655n",
        "description": "Tupolev Tu-95",
        "pos": "NOUN",
        "source": "BABELNET"
    },
]
```

Figure 12: Results for BabelNet API with index

the file extension. It is related to the depiction node via the `schema:image` relation. This new image node was then extended to include a license and a type (`dctypes:StillImage`).

A new blank node is added for each "annotation - bounding box" pair that uniquely connects the pair to the image node with the `oa:hasSource` relation. It will be referred to as "Center Node". The bounding box itself is connected to the Center Node by the `oa:hasSelector` relation and is itself also based on a blank node that has three related values:

1. Its x and y coordinates, as well as the width and height of the box
2. A type (`oa:FragmentSelector`)
3. A specification for URIs of media fragments that the bounding box conforms to

Each annotation is uniquely identified, by adding an index to the id of the depiction and is connected to the Center Node via the `oa:hasTarget` relation. It has a total of 5 directly related values:

1. A label that corresponds to the detected object label
2. The confidence score for the detection
3. The timestamp when it was generated
4. A type (`oa:Annotation`)
5. The link to the related Wikidata entity via the `oa:hasBody` relation

Furthermore, the `dcterms:creator` relation also directly connects the annotation to the object detection model that generated it. The detection model is unique and has two additional values, a type (`slod:ObjectDetector`) and a label that corresponds to the model name.

A working example of this structure can be seen in figure13.

The actual RDF data was generated using the *rdflib*[10] Python package. It provides classes for easy graph generation, as well as serialization in different formats, such as `Turtle` and also includes classes for literals, blank nodes and URIs. Furthermore, many of the well known namespaces, such as Friend Of A Friend (FOAF), Resource Description Framework Schema (RDFS) and Dublin Core Terms (DCTERMS) are already included and new ones can easily be added.

3.3.3 Integration

The final step was to then integrate all of the generated data into the existing Linked Stage Graph. Although this step was not completed in it's entirety due to time restrictions for this thesis, the data was integrated into a dump of the Linked Stage Graph[11] on a local machine.

To this end, *Apache Jena*[12] and it's native TDB triple store were used. First, the dump was imported and tested with demo queries shown in the official Linked Stage Graph "About" section[13], to make sure the local version matches the results from the Linked Stage Graph SPARQL endpoint[14]. The results showed, that the dump contained all the required basic data, but did not include multilingual information. This is fine, since labels in different languages are not required to showcase the new data.

[10]`https://rdflib.readthedocs.io/en/stable/`
[11]Dump was generated on 28.10.2021
[12]`https://jena.apache.org/index.html`
[13]`http://slod.fiz-karlsruhe.de/about`
[14]`http://slod.fiz-karlsruhe.de/sparql`

```
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix dctypes: <http://purl.org/dc/dcmitype/> .
@prefix nif: <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core> .
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix slod: <http://slod.fiz-karlsruhe.de/> .
@prefix wd: <http://www.wikidata.org/entity/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://slod.fiz-karlsruhe.de/annotations/2-2599049-1-1> a oa:Annotation ;
  rdfs:label "Person"@en ;
  nif:confidence "0.89717025"^^xsd:float ;
  dcterms:created "2021-11-01T12:09:09+00:00"^^xsd:dateTime ;
  dcterms:creator _:detector ;
  oa:hasBody wd:Q215627 ;
  oa:hasTarget [ oa:hasSelector [ a oa:FragmentSelector ;
                 dcterms:conformsTo <http://www.w3.org/TR/media-frags/> ;
                 rdf:value "xywh=100,250,160,80"^^xsd:string ] ;
            oa:hasSource <http://slod.fiz-karlsruhe.de/images/slod/2-2599049-1> ] .

<http://slod.fiz-karlsruhe.de/images/slod/2-2599049-1> a dctypes:StillImage ;
  dcterms:rights <https://creativecommons.org/licenses/by/2.0/> ;
  schema:image <http://slod.fiz-karlsruhe.de/images/slod/2-2599049-1.jpg> .

_:detector a slod:ObjectDetector ;
  rdfs:label "EfficientDet"^^xsd:string .
```

Figure 13: Example detection of a Person in Turtle format

Afterwards, the new data was imported on top of the dump and additional queries were run to test it's structure and validity. They can be seen in figures22, 24, 26 and26 and are described in more detail in the Discussion section.

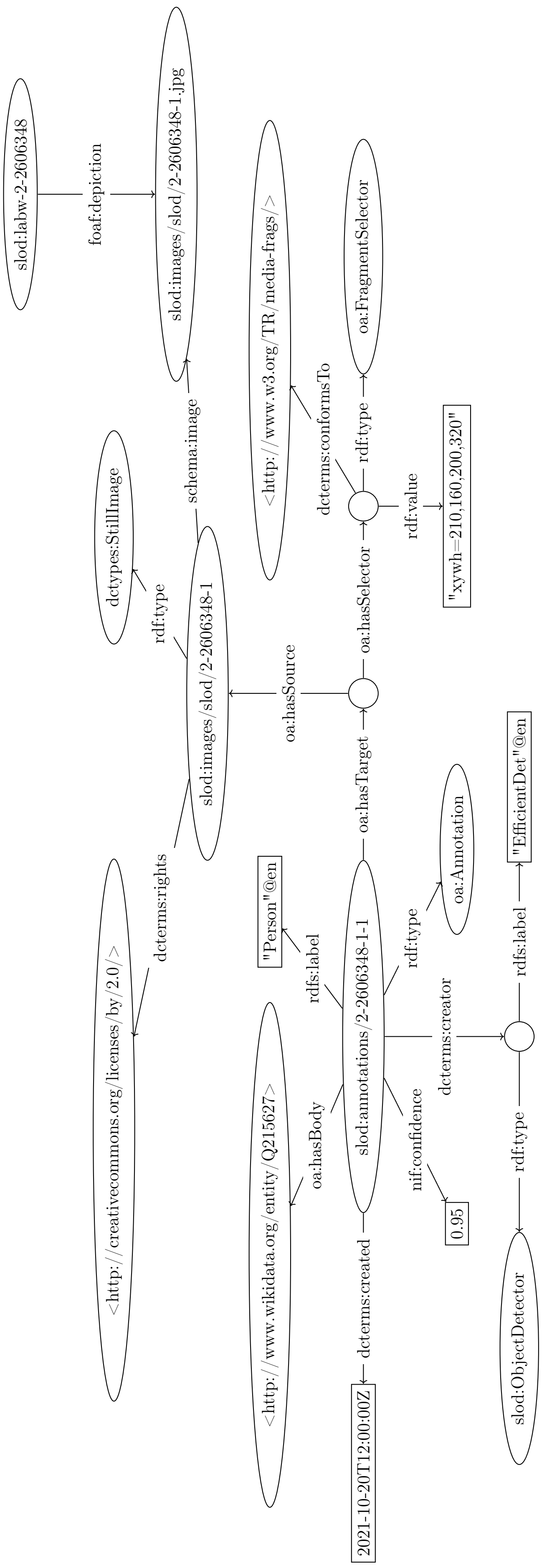


Figure 14: Graph structure for a single detection

4 Evaluation

This section revolves around evaluating the intermediate and final results produced in this thesis, including a comparison of the pre-selected object detection models, the evaluation of the user study and the final model, as well as the results generated by this model and how they were integrated into the existing Knowledge Graph.

4.1 Initial Model Comparison

All four models were applied to the same set of 80 sample images, producing the results seen in figure15. In regards to average detections per image, the Faster R-CNN model trained on MS COCO outperformed the other models with an average of 7 detections per image, whereas both the EfficientDet and Faster R-CNN (OpenImages) models only averaged about 4.2 detections per image. CenterNet takes last place, with an average of only 2.4 detections per image.

An explanation for this discrepancy could be, that the Faster R-CNN model trained on MS COCO works best at detecting multiple people in a single image, since many of the images depicting stage performances contain groups of people.

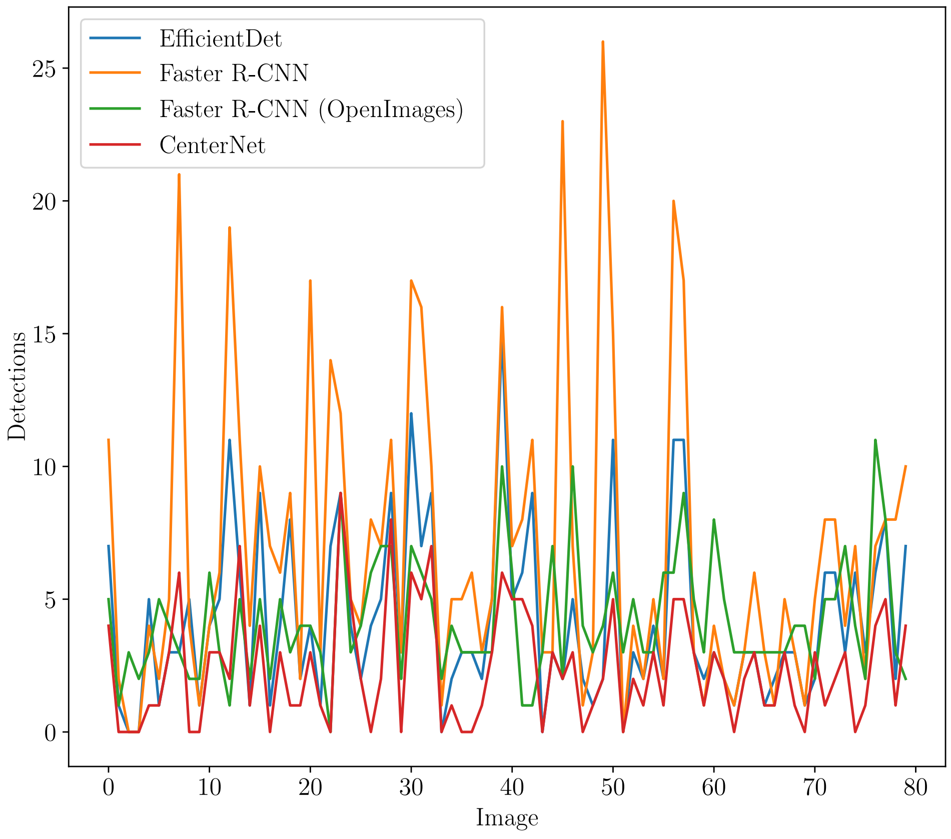


Figure 15: Detections per image for all models

4.2 User Study Results

The study was run over a period of one month, during which a total of 131 people participated. After it concluded, all results were exported and analyzed with the open source Python library *Pandas*[15]. It contains many features that make evaluating and plotting data very easy and is compatible with Jupyter.

Participation

Although the study had a total of 131 participants, 46 of them either did not answer any questions or only answered the background section of the survey and were therefore ignored in the evaluation. Of the remaining 95 participants, only 31 completed the entire survey16. Furthermore, almost all of the other 64 participants quit the survey during the first 100 questions.

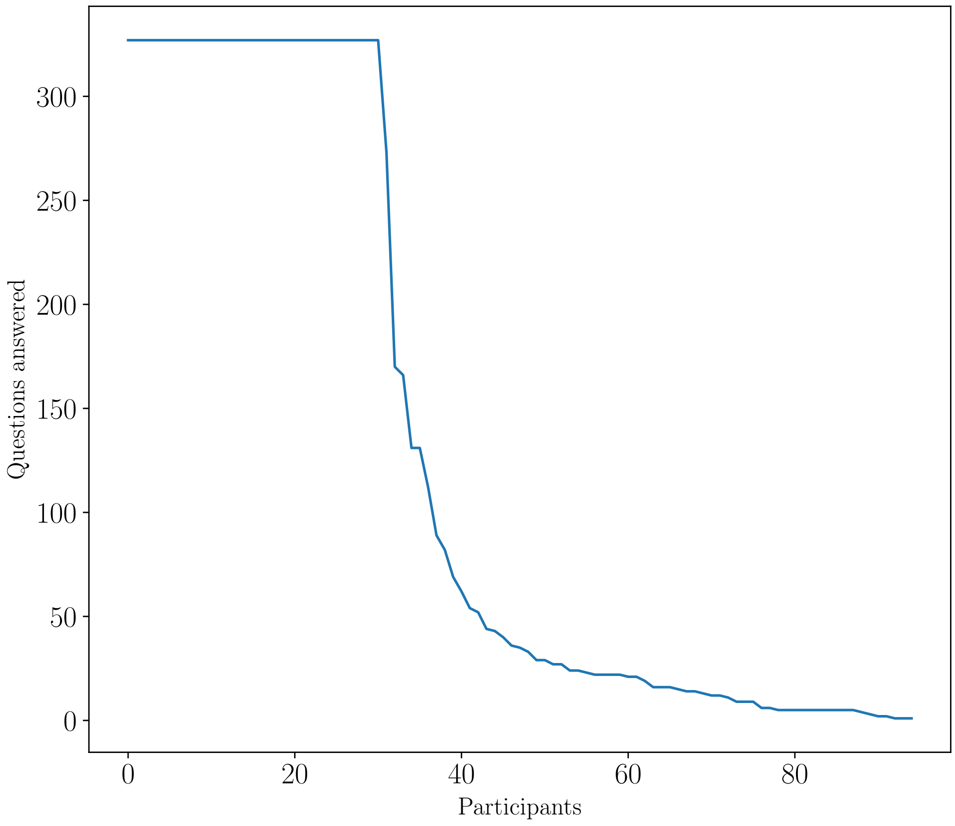


Figure 16: Completed questions by participants

This might seem like a problem, but *LimeSurvey* can also extract answers from unfinished attempts and therefore, every recorded answer was used in the final evaluation of the models. Of course, this could lead to an imbalance in the amount of recorded answers for each model. However, since the order of the questions was randomized for each

[15]`https://pandas.pydata.org/`

participant, the total amount of answers is fairly close for each model as depicted in figure18. The largest difference is 41 answers, between EfficientDet (2977 answers) and CenterNet (2936 answers). This difference amounts to around $\frac{1}{75}th$ of the almost 3000 answers that were recorded for each model and is therefore not be big enough to drastically impact the final scores. This was confirmed by calculating the scores with only 2900 answers for all models, which did not change any of the four scores.

Model Evaluation

Figure19 showcases the results of the study, grouped by each model over all five possible answers. To determine some kind of score for each model that can used to compare the models' results, all recorded answers from the Likert-Scale were mapped to fixed 0.25 increments from 0 to 1 as shown in figure17.

```
Very accurate     => 1
Mosty accurate    => 0.75
Neither           => 0.5
Mostly inaccurate => 0.25
Very inaccurate   => 0
```

Figure 17: Mapping of Likert-Scale answers

The 0.5 for "Neither" was chosen, because this position cannot only represent "no detections", but also "equal amounts of correct and incorrect detections" and cannot be disregarded entirely. Therefore, it was chosen as the middle ground for this score. If there are a lot of incorrect detections, the score gets closer to 0 and if more detections are correct, the score gets closer to 1. If a model does not detect any objects at all or usually produces results that contain similar amounts of good and bad detections, the score will remain close to 0.5. Afterwards, the mean over all mapped answers for each model was determined and used as the final score, which can be seen in figure18.

Model	Total answers	Final score
EfficientDet D7	2977	0.89
Faster R-CNN (Open-Images)	2945	0.85
CenterNet	2936	0.79
Faster R-CNN (MS COCO)	2958	0.74

Figure 18: Total answers and final score by model

Furthermore, an Inter-Annotator Agreement (IAA) was calculated for the survey, which will serve as a benchmark for the expressiveness of the study and these final scores. Depending on the type of survey, this agreement can be calculated in different ways. Since we had a fixed amount of participants larger than 2, the IAA was calculated using the *Fleiss' Kappa* method [4], which is used to determine how similar the participants' answers for each question were. The result is a single number between 0 and 1, where 0 is "virtually no agreement" and 1 is "almost perfect agreement".

For this survey, the resulting *Fleiss' Kappa* value was 0.5. This evaluates to a "moderate agreement" [4] and is sufficient in this case, since additional criteria were considered when selecting the final model.

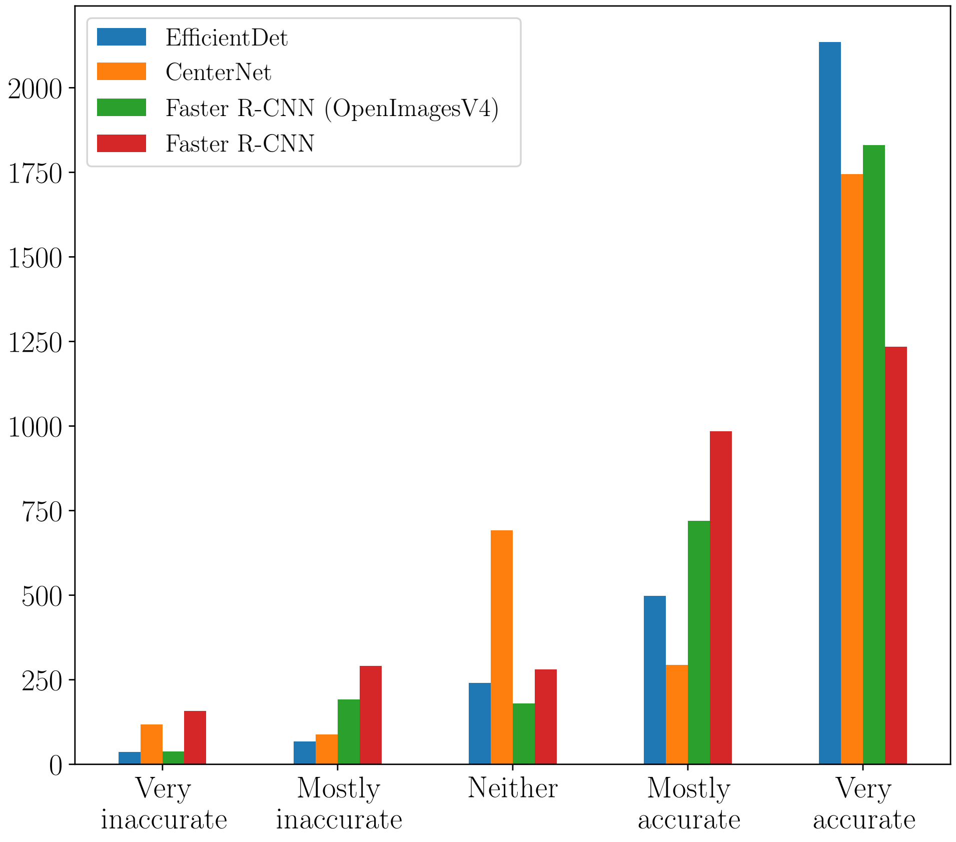


Figure 19: Results for detections from user study

To be more specific, the scores were used in conjunction with the results from figure19 and figure15 to select the EfficientDet model as the most reliable one of the four. Not only does it have the highest score, it also has the most "Very accurate" answers by far and is also the model with the least "Very inaccurate" and "Mostly inaccurate" answers. As previously mentioned during the preselection of the models, the Faster R-CNN model trained on MS COCO produced three detections more per image on average. However, since the correctness of the detections is far more important than generating as many detections as possible, the EfficientDet model is a better choice for our purpose.

The Faster R-CNN model trained on OpenImages V4 was also taken into consideration for the final model, since it contains almost 8 times as many object classes as the models trained on MS COCO and would be able to produce a larger variety of detections. But, as with the other Faster R-CNN model, this selection primarily focused on correctness and not quantity or variety. As such, the models' advantages did not outweigh the difference in accuracy when compared to the EfficientDet model. It has over 200 "Very accurate" detections less and performs either similar or worse in both "inaccurate" categories. Although it has more "Mostly accurate" answers, the difference in this category can be explained, in part, due to the difference in the "Very accurate" category. Obviously, the latter is more important and therefore, the EfficientDet model outperforms the Faster R-CNN model in the vital parts of this evaluation.

Wrap-Up

Of the 31 participants that completed the entire survey, 10 entered additional information in the wrap-up questions.

Two answers for the first question, pertaining to things that were noteworthy about the detections that were displayed, stood out in particular:

The first one was: "Entire theatre scenes were pretty consistently labeled as buildings". This can only refer to the detections from the Faster R-CNN model trained on OpenImages V4, since MS COCO does not contain an object class for "Building". Depending on the stage play, this classification could both be correct and incorrect, since many of the plays actually feature buildings in the background of the stage set. Therefore, additional information would be required to determine the accuracy of these detections.

The second answer was phrased similarly by 3 of the participants: "Larger groups of people were frequently detected as a single person". This could refer to all four models and since they were all pre-trained, optimizing certain behaviors were out of the scope of this thesis. This will be further discussed in the "Future Work" section. However, since MS COCO contains many complex situations and unusual viewpoints of it's objects, one possible candidate for this specific behavior could be the Faster R-CNN model trained on OpenImages V4.

Finally, the answers to the second question, pertaining to the way the study was conducted, did not bring up any key flaws or common problems. They mostly described how the participant evaluated certain situations, e.g. what they categorized as "Neither".

4.3 Knowledge Graph Integration

The chosen EfficientDet model was used to generate the final results, which were then linked against related Wikidata Entities, since these relations are required for the final RDF representation. The data was then converted to RDF and validated against the existing data in Linked Stage Graph.

Final Object Detection Results

The final results were generated for three different minimum confidence scores: 0.5 (50%), 0.75 (75%) and 0.9 (90%). As previously mentioned in the evaluation of the final model, the most important aspect of these detections is their correctness and raising the required score increases the overall quality of the detections. A comparison between the results of different confidence scores can be seen in figure20, which showcases how many images contained a certain number of detected objects. As previously stated in the "Prerequisites" section, there are photographs that only display text. It is important to note that they will never yield object detection results, since text recognition is not what the models were trained for and therefore, there will always be a certain amount of images that contain zero detections.

Although a higher confidence score removes uncertain detections, the figure also clearly shows that setting the threshold too high will remove too many. With a minimum confidence score of 0.9, a total of 5218 out of 7000 images contained no detections at all. The final score was increased to 0.75 from the 0.5 score that was used in the User Study, since it produced a good middle ground between quantity and quality, with an average detection rate of 1.56 detections per image, resulting in 10.925 total detections.

Increasing the confidence score also removed certain object classes entirely. The initial threshold of 0.5 produced detections for 61 out of the 80 object classes in the MS COCO dataset. The new confidence score of 0.75 reduced the total detected classes to 36. Among others, the dropped classes included "Hair drier" and "Truck", which are not displayed in any of the images. However, the classes "Bear" and "Dining table" were also removed, even though both had at least one actual instance in the images.

Entity Linking

The custom BabelNet algorithm was used to find related Wikidata entities for these 36 classes. A shortened version of the results can be seen in figure21. It is important to note, that the columns "Exact match" and "Correct ID" were added while validating the generated mappings by hand and are not part of the inital result. Furthermore, a reference to the ID of the BabelNet entry, as well as full links to the respective BabelNet

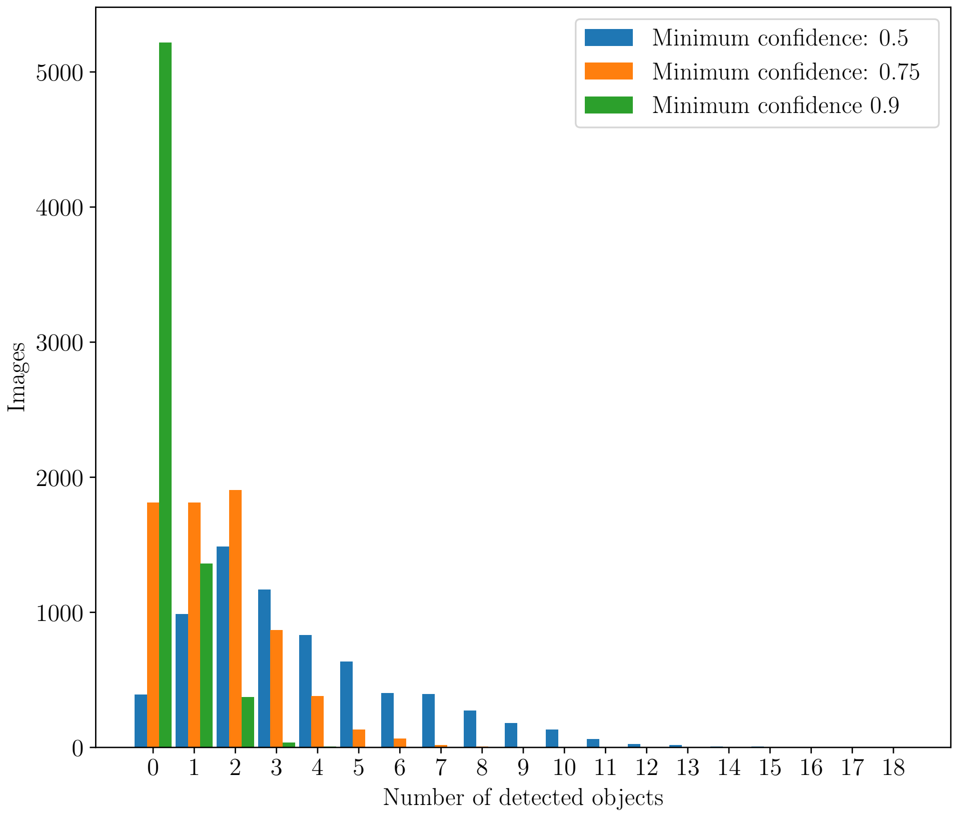


Figure 20: Amount of detected objects in relation to minimum confidence score

and Wikidata web-pages are returned with the initial result, but have been cut in the figure for better readability.

Object class	Wikidata Entity ID	Label matched	Exact match	Correct ID
Person	Q215627	Yes	Yes	-
Tie	Q44416	No	Yes	-
Potted plant	Q203834	No	No	Q89810050
Bowl	No match	No	No	Q153988

Figure 21: Example of different Entity Mappings

First off, the algorithm was unable to determine any kind of match for two of the classes. The first one was "Sports ball", which does not have any matching results when querying BabelNet. The second class was "Bowl". This search term actually produced four BabelNet entries that were semantically correct, but did not have a Wikidata source. In this case, the correct entry the algorithm was looking for was placed at index 10 and was therefore out of range.

Seven other classes did not produce mappings that exactly matched the class label. However, only the mapping for the class "Sink" was incorrect. Even though the correct

BabelNet entry was found, it contained two Wikidata sources that both contained the word "sink" and therefore, the first one was chosen. In this case, the second entry would have been the correct one. Five of the six remaining classes are simply listed under different titles in Wikidata, such as the object class "Tie". It was matched to the entity "Necktie", which is correct even though the titles do not match exactly. The last class is "Potted plant", which was linked to "Houseplant". While this match is not exactly wrong, there is a certain ambiguity involved. A separate Wikidata entity for "Potted plant" exists, which would be the exact match and is a super-class of "Houseplant". Both of these mappings would have been valid, but for the sake of being as exact as possible, the mapping was changed to the "Potted plant" entity.

Overall, 32 out of the 36 mappings were correct and only four had to be selected manually.

RDF Data and Validity

With the validated mappings, the detections were all converted to RDF based on the structure displayed in figure14, resulting in 141.151 total new nodes in the graph. They can be divided into 75.599 new nodes containing URIs, 43.701 new nodes containing literals (e.g. strings and numbers) and 21.851 new blank nodes. Furthermore, a total of 152.072 new triples were introduced.

This new data was then imported into the existing data of the Linked Stage Graph on a local machine to test it's validity and if it can be used in conjunction with the existing data. To this end, two simple queries were run.

The first query was chosen to test the validity of the data on it's own, by only relying on nodes and relations that were newly introduced. It outputs all images that contain an object with the class "Bicycle", which was detected exactly two times. The exact query and its outputs can be seen in figures22 and23.

Figure23 shows exactly two results and the URIs match the images that contain the "Bicycle" detections.

The second query was used to confirm that the data has been integrated into the existing data correctly. It returns all stage plays that contain detections generated by the "EfficientDet" model. The query and a shortened version of the results are displayed in figures24 and25.

The results were confirmed by programmatically collecting the distinct play ids, which are part of the image filenames, for each image that contained detections. The two outputs were an exact match.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX schema: <http://schema.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?depiction
    WHERE {
        ?play foaf:depiction ?file .
        ?depiction schema:image ?file .
        ?blank oa:hasSource ?depiction .
        ?annotation oa:hasTarget ?blank ;
            rdfs:label "Bicycle"@en .
   }
```

Figure 22: Query for images that contain an annotation of object class "Bicycle"

```
-----------------------------------------------------------
| depiction                                               |
===========================================================
| <http://slod.fiz-karlsruhe.de/images/slod/2-3030407-16> |
| <http://slod.fiz-karlsruhe.de/images/slod/2-3030495-9>  |
-----------------------------------------------------------
```

Figure 23: Results for query of images by object class

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX schema: <http://schema.org/>
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?play ?label
    WHERE {
        ?detector rdf:type slod:ObjectDetector ;
            rdfs:label "EfficientDet" .
        ?annotation dcterms:creator ?detector ;
            oa:hasTarget ?target .
        ?target oa:hasSource ?image .
        ?image schema:image ?file .
        ?play foaf:depiction ?file ;
            rdfs:label ?label .
    }
```

Figure 24: Query for stage plays that contain images with annotations created by the "EfficientDet" model

```
----------------------------------------------------------------------------------
| play                | label                                                    |
==================================================================================
| slod:labw-2-2061432 | "Werkstätten des Hoftheaters (Malersaal, Schneiderei ?)" |
| slod:labw-2-2061480 | "Fundus des Hoftheaters"                                 |
| slod:labw-2-2061484 | "Schneider im Hoftheater (Schneidersitz mit Türkenhut)"  |
----------------------------------------------------------------------------------
```

Figure 25: Shortened results for query of stage plays by detector

5 Discussion of the Results

The goal of this thesis was to enrich the semantic metadata of almost 7000 historical photographs contained in the existing Linked Stage Graph with the help of object detection. To this end, the pre-trained TensorFlow model "EfficientDet D7" was selected out of four candidates by conducting a user study. Afterwards, this model was used to generate a total of 10.925 detections on the images. This new information was then converted into a RDF representation, which was specifically designed for this purpose, resulting in over 200.000 new nodes. Furthermore, the 36 detected object classes were linked to matching entities in the Wikidata knowledge base, using a custom linking algorithm and these links were also included in the RDF representation. In this section, the results of these different steps, limitations and possible optimizations will be discussed. In addition, the last paragraphs will discuss how well the final results solved the problem of "increasing the explorative search capabilites of historical photograph collections based on visual information contained in the images". Sample queries for more complex relations will also be provided, to showcase new features.

5.1 Object Detection

First off, a big limiting factor in the generation of reliable object detection results in this thesis was the use of a pre-trained model. As previously mentioned, training a model specifically for this purpose would have exceeded the scope of this thesis, when combined with the remaining tasks. The pre-trained models from TensorFlow are very generic so that they can be fit to a wide variety tasks. However, this also means that they contain classes that are irrelevant in this specific context, such as different animals that were never displayed in any of the images. More importantly, many of the props that can be seen in the images do not have a corresponding object class and the models do not reliably detect animals or other creatures that are depicted by a person in a costume. That being said, the results generated by the EfficientDet model were still quite useful as a starting point, since it was able to extract information on a few generic classes that were depicted quite a lot, especially persons. A possible solution to this limitation would be training an object detection model for theatre related applications from scratch or using transfer learning and fine-tuning methods on an existing model [36]. Both of these solutions could be implement with TensorFlow[16], making the model compatible with this thesis.

Another possible improvement for the object detection would have been to remove or ignore detections of object classes that are never depicted in any of these images. However, this definitely would have limited the scope of application of this thesis, since a different

[16]`https://www.tensorflow.org/tutorials/images/transfer_learning?hl=en`

collection of images could very well depict one or more of the ignored classes.

5.2 User Study

The User Study did produce usable results, but an IAA score higher than 0.5 would have been desirable. The IAA is a statistical tool for measuring the expressiveness of a Study and although a "moderate" agreement was sufficient in this case, because it was used in combination with additional criteria, it leaves room for improvement. Designing the evaluation of the detection results differently might have resulted in a better IAA. For example, the four different detections for each image could have been presented in a side-by-side view and the detections could have been rated by ordering them from best to worst. This would have resulted in a more direct comparison of the models and might have made it easier for the participants to determine which results are good. On the other hand, this approach would have made it relatively simple for a more knowledgeable participant to single out certain models, which could introduce a personal bias in the evaluations.

5.3 Entity Linking

One step that worked quite well was the Entity Linking. It only missed a handful out of the 36 mappings, all of which had a logical explanation. Upon closer inspection, some of the defective mappings could even be attributed to problems with BabelNet and were not necessarily errors in the algorithm. Based on such edge cases, the approach developed in this thesis could also be further optimized to produce even more reliable results. For example, certain fallback strategies could be included in case BabelNet does not return any results for a search term, as was the case with the object class "Sports ball".

5.4 Sample Queries

Lastly, the final results were able to increase the explorative search capabilites of the Linked Stage Graph, solely based on visual information extracted from the photographs. Two simple queries were already introduced in figures22 and24 to showcase how images can now be aggregated by the object class and how annotations can be queried based on the object detector that created them. Here, additional sample queries will showcase more complex functionalities.

The first example shown in figure26 aggregates distinct images that display certain objects with have additional restrictions, by including and querying relations from the Wikidata knowledge base. More specifically, the image has to contain a linked entity that is a subclass of (relation P279) a "domesticated mammal" (entity Q57814795), e.g. a dog, cat or sheep.

```
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX p: <http://www.wikidata.org/prop/>

SELECT DISTINCT ?file ?label
    WHERE {
        ?annotation oa:hasBody ?entity ;
            oa:hasTarget ?blank .
        ?blank oa:hasSource ?image .
        ?image schema:image ?file .

        SERVICE <https://query.wikidata.org/sparql> {
            ?entity p:P279 wd:Q57814795 ;
                rdfs:label ?label FILTER(LANG(?label)="en") .
        }
    }
```

Figure 26: Query for all images that contain annotations of animals in the category "domesticated mammal"

The next example shown in figure27 uses a specific image as a starting point. It distinctly counts all other images that contain detections with the same object classes and that have confidence scores that are equal to or higher than the scores in the given image. For the sake of simplicity, the image chosen for this query only contained exactly one detection of a person. However, the query can be adapted to work with multiple detections.

Additionally, queries that find images with similar bounding box positions can be created. However, this would require a rather complex query, when using plain SPARQL, since the bounding box value would have to be split and compared inline. Depending on the framework used to store and maintain the data, queries that transcend or simplify the functionalities of pure SPARQL queries can be created. For example, when using Apache Jena, custom Java functions can be written and included as filters in these queries, making them much more powerful.

```
PREFIX nif: <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core>
PREFIX oa: <http://www.w3.org/ns/oa#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT (COUNT(?image) as ?matchingScores)
    WHERE {
        ?blank oa:hasSource <http://slod.fiz-karlsruhe.de/images/slod/2-3030425-3> .
        ?annotation oa:hasTarget ?blank ;
            rdfs:label ?label ;
            nif:confidence ?score .

        ?similarAnnotation oa:hasTarget ?blank2 ;
            rdfs:label ?label ;
            nif:confidence ?similarScore .
        ?blank2 oa:hasSource ?image .

        FILTER(?similarScore >= ?score) .
   }
```

Figure 27: Query that counts all images that contain same object class with similar confidence scores

6 Conclusion and Outlook

Archived data of historical theatre photograph collections usually contain very sparse metadata that is mostly limited information about the stage play, e.g. a name, dates when the play was performed or a description. However, more metadata can be generated automatically with modern methods. The goal of this thesis was to extract visual information about the contents of these photographs using a state of the art object detection model. To this end, four pre-trained models from the TensorFlow library were compared with the help of a user study. The results of its evaluation were used to determine the EfficientDet D7 model as the model that produces the most reliable results. It was then used to produce detections on a set of around 7000 historical photographs. This resulted in a total of 10.925 detections. Additionally, the 36 classes detected by this model were linked against their respective entries in the Wikidata knowledge base using a custom algorithm, which correctly mapped 32 of the 36 classes. Finally, the results were converted to RDF, which also includes the link to Wikidata and imported into an existing Knowledge Graph called Linked Stage Graph, extending it by over 140.000 new nodes. This open up new possibilities for the exploration of these images, based on their visual contents.

6.1 Future Work

This section will give an outlook on additions and changes to the different steps of this thesis that could be implemented in the future to improve the overall results.

Custom Model

An entirely new model could be trained or an existing model could be adapted using transfer learning and fine-tuning for the purpose of detecting and classifying object classes that are more closely related to the field of the performing arts. This would drastically improve not only the accuracy, but also the semantic value of the detections.

Optimizing Entity Linking

The evaluation of the generated mappings showed that there are certain edge cases, e.g. no matches are returned by BabelNet or a wrong match is selected even if a correct match exists. These edge cases could be included in the entity linking approach, e.g. by implementing fallback methods, which would further increase its accuracy.

References

[1] Blom, F., Nijboer, H., and van der Zalm, R. Onstage, the online data system of theatre in amsterdam from the golden age to today: Arts and media. Technical report, Amsterdam, Netherlands, 2020.

[2] Cedefop. *Fields of Employment/Study*. European Commision, Directorate-General for Employment, Social Affairs and Inclusion, 2019.

[3] Culture in Development. What is cultural heritage. http://www.cultureindevelopment.nl/cultural_heritage/what_is_cultural_heritage.

[4] de Bruijn, L. *Inter-Annotator Agreement*, 2020.

[5] Delpeuch, A. Opentapioca: Lightweight entity linking for wikidata. *CoRR abs/1904.09131* (2019).

[6] Ferro, N., and Silvello, G. From users to systems: Identifying and overcoming barriers to efficiently access archival data. Technical report, Department of Information Engineering University of Padua, 2016.

[7] Floortje, B. History and contents of the dutch theatre production database. Technical report, Amsterdam, Netherlands, 2020.

[8] Girshick, R. B. Fast R-CNN. *CoRR abs/1504.08083* (2015).

[9] Google Ltd. Tensorflow library. https://www.tensorflow.org/, 2021.

[10] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CoRR abs/1512.03385* (2015).

[11] Illmayer, K. Theadok. https://ndownloader.figshare.com/files/12934127, 2018.

[12] Jason A. Deoldify. https://github.com/jantic/DeOldify#the-technical-details, 2021.

[13] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J. R. R., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Duerig, T., and Ferrari, V. The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale. *CoRR abs/1811.00982* (2018).

[14] LeadQuizzes Team. *Demographic questions*. LeadQuizzes, 2021.

[15] LimeSurvey Project Team / Carsten Schmitz. *LimeSurvey: An Open Source survey tool*. LimeSurvey Project, Hamburg, Germany, 2012.

[16] Lin, T., Dollár, P., Girshick, R. B., He, K., Hariharan, B., and Belongie, S. J. Feature pyramid networks for object detection. *CoRR abs/1612.03144* (2016).

[17] Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. Focal loss for dense object detection. *CoRR abs/1708.02002* (2017).

[18] Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: common objects in context. *CoRR abs/1405.0312* (2014).

[19] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., and Berg, A. C. SSD: single shot multibox detector. *CoRR abs/1512.02325* (2015).

[20] Marschall, B., and Illmayer, K. Theadok website. `https://theadok.at/`, 2018.

[21] Moeller, C., Lehmann, J., and Usbeck, R. Survey on english entity linking on wikidata. *Semantic Web Journal* (2021).

[22] Navigli, R., and Ponzetto, S. P. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence 193* (2012), 217–250.

[23] Nielsen, F. r. Linking imagenet wordnet synsets with wikidata. Technical report, Technical University of Denmark, 2018.

[24] Raj, B. *A Simple Guide to the Versions of the Inception Network.* Towards Data Science, 2018.

[25] Raj, B. *DeOldify training*, 2020.

[26] Ren, S., He, K., Girshick, R. B., and Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR abs/1506.01497* (2015).

[27] Sack, H., Tietz, T., Alam, M., and Waitelonis, J. Knowledge graph based analysis and exploration of historical theatre photographs. In *Proceedings of the Conference on Digital Curation Technologies (Qurator 2020)* (2020), Conference on Digital Curation Technologies (Qurator 2020), CEUR.

[28] Sakor, A., Singh, K., Patel, A., and Vidal, M. FALCON 2.0: An entity and relation linking tool over wikidata. *CoRR abs/1912.11270* (2019).

[29] SurveyMonkey Team. *Education demographic questions.* SurveyMonkey, 2021.

[30] Szegedy, C., Ioffe, S., and Vanhoucke, V. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR abs/1602.07261* (2016).

[31] Tan, M., and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR abs/1905.11946* (2019).

[32] Tan, M., Pang, R., and Le, Q. V. Efficientdet: Scalable and efficient object detection. *CoRR abs/1911.09070* (2019).

[33] The University of St Andrews. *Likert-Scale*, 2021.

[34] Tietz, T., Waitelonis, J., Alam, M., and Sack, H. Knowledge graph based analysis and exploration of historical theatre photographs. Technical report, KIT Institute AIFB and FIZ Karlsruhe, Karlsruhe, 2020.

[35] Tietz, T., Waitelonis, J., Kanran, Z., Felgentreff, P., Meyer, N., Weber, A., and Sack, H. Linked stage graph. Technical report, KIT Institute AIFB and FIZ Karlsruhe, Karlsruhe, 2019.

[36] Vrbančič, G., and Podgorelec, V. Transfer learning with adaptive fine-tuning. *IEEE Access 8* (2020), 196197–196211.

[37] Windhager, F., Federico, P., Schreder, G., Glinka, K., Dörk, M., Miksch, S., and Mayr, E. Visualization of cultural heritage collection data: State of the art and future challenges. Technical report, Danube University Krems and Stiftung Preussischer Kulturbesitz and University of Applied Sciences Potsdam and TU Wien, 2018.

[38] Zhou, X., Wang, D., and Krähenbühl, P. Objects as points. *CoRR abs/1904.07850* (2019).

Abbrevations

AIFB Institute of Applied Informatics and Formal Description Methods

KIT Karslruhe Institute of Technology

FIZ FIZ Karlsruhe – Leibniz-Institute for Information Infrastructure

GLAM institutions Galleries, Libraries, Archives and Museums

URI Unique Resource Identifier

SPARQL SPARQL Protocol And RDF Query Language

RDF Resource Description Framework

XML Extensible Markup Language

LIDO Lightweight Information Describing Objects

API Application Programming Interface

JSON JavaScript Object Notation

GAN Generative Adversarial Networks

MS COCO Microsoft Common Objects in Context

R-CNN Region Based Convolutional Neural Network

FPN Feature Pyramid Network

BiFPN Bi-directional Feature Pyramid Network

SSD Single Shot Multibox Detector

RPN Region Proposal Network

ResNet Residual Network

CSV Comma Separated Values

HTTP Hyper Text Transfer Protocol

FOAF Friend Of A Friend

RDFS Resource Description Framework Schema

DCTERMS Dublin Core Terms

IAA Inter-Annotator Agreement

Assertion

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

A. Maier

Karlsruhe, November 2, 2021

Aaron Maier