MediaVerse

A universe of media assets
and co-creation opportunities

D3.3

Interactive New Media Annotation Module and Content Discovery and Recommendation Framework v2

Project Title	MediaVerse
Contract No.	957252
Instrument	Innovation Action
Thematic Priority	ICT-44-2020 Next Generation Media
Start of Project	1 October 2020
Duration	36 months

Deliverable title	Interactive New Media Annotation Module and Content Discovery and Recommendation Framework v2
Deliverable number	D3.3
Deliverable version	V1.0
Previous version(s)	D3.2, D3.1
Contractual Date of delivery	31.03.2023
Actual Date of delivery	29.03.2023
Nature of deliverable	Other
Dissemination level	Public
Partner Responsible	CERTH
Author(s)	Christos Koutlis, Ioannis Sarridis, Manos Schinas, Nikos Lazaridis, Kostas Georgiadis, Fotis Kalaganis, Panagiotis Galopoulos, Manolis Krasanakis, Dimitris Karageorgiou, Symeon Papadopoulos (CERTH), Federico D'Asaro, Sara De Luca, Giuseppe Rizzo (LINKS)
Reviewer(s)	Stephan Gensch (VRAG), Pilar Orero (UAB)
EC Project Officer	Luis Eduardo Martinez Lafuente

Abstract	Deliverable D3.3 includes software components as well as the corresponding documentations and reports for the MAAM fork, improvements and additions in the Media Annotation Service and the adopted retrieval and recommendation technologies.

Keywords	Artificial Intelligence, Media Annotation, Media Retrieval, Recommender System

Copyright

MediaVerse is an H2020 Innovation Project co-financed by the EC under Grant Agreement ID: 957252.
The content of this document is © the author(s). For further information, visit mediaverse-project.eu.

Revision History

VERSION	DATE	MODIFIED BY	COMMENTS
V0.1	02/02/2023	Christos Koutlis (CERTH)	First Draft Table of Content
V0.2	16/02/2023	Christos Koutlis, Ioannis Sarridis, Nikos Lazaridis, Kostas Georgiadis, Fotis Kalaganis, Panagiotis Galopoulos, Manolis Krasanakis, Nikolaos Sarris, Symeon Papadopoulos (CERTH)	Edit New Media and Annotation Service (Section 3)
V0.3	21/02/2023	Manos Schinas, Christos Koutlis, Dimitris Karageorgiou, Nikolaos Sarris, Symeon Papadopoulos (CERTH)	Edit MAAM fork (Section 2)
V0.4	03/03/2023	Federico D'Asaro, Sara De Luca, Giuseppe Rizzo (LINKS)	Edit Retrieval and Recommendations (Section 4)
V0.5	07/03/2023	Christos Koutlis (CERTH)	Draft Executive Summary and Introduction
V0.6	08/03/2023	Christos Koutlis (CERTH)	Comments and suggestions to all sections
V0.7	10/03/2023	Ioannis Sarridis, Manos Schinas, Panagiotis Galopoulos, Nikos Lazaridis, Fotis Kalaganis (CERTH), Federico D'Asaro, Sara De Luca, Giuseppe Rizzo (LINKS)	Comments addressed
V0.8	20/03/2023	Stephan Gensch (VRAG), Pilar Orero (UAB)	Review
V0.9	22/03/2023	Christos Koutlis, Ioannis Sarridis, Manos Schinas, Nikos Lazaridis, Kostas Georgiadis, Fotis Kalaganis, Panagiotis Galopoulos, Manolis Krasanakis, Dimitris Karageorgiou, Nikolaos Sarris, Symeon Papadopoulos (CERTH), Federico D'Asaro, Sara De Luca, Giuseppe Rizzo (LINKS)	Address comments and suggestions from the internal review process
V1.0	31/03/2023	Evangelia Kartsounidou, Symeon Papadopoulos (CERTH)	Final review and quality control

Glossary

ABBREVIATION	MEANING
ALBEF	ALign BEfore Fuse
API	Application Programming Interface
AUC	Area Under Curve
AI	Artificial intelligence
CPU	Central Processing Unit
CLIP	Contrastive Language-Image Pre-Training
CNN	Convolutional Neural Network
DAM	Digital Asset Management
DCD	Disturbing Content Detection
FR	Face Recognition
FC	Fully Connected
GDPR	General Data Protection Regulation
HMD	Head-mounted Display
IR	Information retrieval
IBM	International Business Machines
kNN	k-nearest neighbour
KD	Knowledge Distillation
KL	Kullback-Leibler
LSTM	Long Short-Term Memory
mAP	mean Average Precision
MAAM	Media Assets Annotation and Management
MoD	Momentum Distillation
MI	Mutual Information
MV	MediaVerse
MVC	Model-View-Controller architecture
NDD	Near Duplicate Detection
NSFW	Not Safe For Work
ONNX	Open Neural Network Exchange
RS	Recommender System
TSN	Temporal Segment Networks
UQI	Universal Quality Index
UI	User Interface
ViT	Vision Transformer
VPU	Visual Part Utilisation
WP	Work Package

Table of Contents

Index of Figures

Index of Tables

Executive Summary

In this deliverable, we report the work that has been conducted in the context of MediaVerse WP3: “Next Generation Content Management, Understanding and Interlinking” during the last period of the project. First, we document the development of the features of a variant of the official MediaVerse node, called Media Asset Annotation and Management (MAAM). This section includes the ability to create user-defined object detection and retrieval models that will be integrated in the MAAM fork. We also report the model-related improvements and additions in the Media Annotation Service as well as the final form of the recommendations and retrieval technologies developed within MediaVerse. The reported work has taken place until M30 of the project concluding the developments regarding media annotation and recommendation functionalities.

More precisely, we elaborate on the annotation models that act on uploaded assets through the MAAM fork to provide metadata, its visual similarity feature provided for retrieval purposes as well as a user-friendly model creation approach, an altered version of which will substitute the current underlying model of the visual similarity and retrieval feature. Additionally, we report on a novel Internet image meme classification model developed to detect multi-modal hate speech and a novel model compression methodology that we developed in order to increase the Media Annotation Service efficiency. Moreover, we present a new saliency detection model for 360 content and improvements on the face blurring model that provide all updates included in the latest version of the Media Annotation Service. Finally, we elaborate on the experimental results that lead to the final version of the cross-modal retrieval and recommendations modules. In particular, for retrieval technology we present experiments on new datasets and an alternative architecture to CLIP as an embedding technique, while for the recommender system, we conduct quantitative analysis and we present two testing setups.

1 Introduction

The main objective of deliverable D3.3 is to describe the updates conducted in the framework of WP3 tasks that lead to the final version of MediaVerse components, such as the Media Asset Annotation and Management (MAAM) platform, the Media Annotation Service, and the Content Retrieval and Recommendation functionality.

The motivation behind MAAM is to provide a specialised variant of the official MediaVerse node that can handle media annotations and asset organisation requirements more effectively. The current approach for visual concept similarity and retrieval in MAAM utilises feature vectors extracted from CLIP and applied on the whole image. The observation that in the current implementation semantically-only correlated images are returned in high rank and the fact that CLIP is pre-trained on rich images in terms of objects and semantics which makes it produce less informative representations for single-object images, motivated the development of a new region-based feature extraction retrieval process utilising a model trained on single-object images.

Regarding the Media Annotation Service, we developed a new model for fine-grained image meme classification as a follow-up from deliverable D3.2 - Content Discovery and Recommendation, Annotation and Adaptation Framework in which we reported the development of a meme detection model. The new model is trained to detect hate speech in the multi-modal setting of image macros and provides the assets detected as Internet image memes with an additional label (either being hateful or not) that can then be utilised for automatic flagging or moderation. Additionally, given that the service hosts a big number of models most of which are heavy, we developed a new model compression methodology with state of the art performance and applied it in some of the MediaVerse models leading to considerable computational gains with little drop in accuracy. We also developed a new saliency detection methodology that can be applied to images and both regular and 360 videos. The utility of this functionality is to help users obtain information about the areas of their assets that attract visual attention. This modality can support users to make better decisions about the areas that are suitable for content placement, such as text and buttons. Finally, we report the improvement actions undertaken to enhance the performance of the face blurring module. The initial version of the module was characterised by low time efficiency. Hence, we incorporated three modifications that resulted in a four-fold reduction in processing time.

We also continued working on the retrieval capabilities of MediaVerse via a multimodal perspective, allowing users to retrieve MV assets of different modality (texts or images) with respect to their queries based on semantically relevance. The goal has been to improve the user experience by suggesting relevant and novel assets to MediaVerse users. Most modern recommendation systems leverage the past behaviour of users to recommend an asset that a user has not seen but another user with close interests has. In MediaVerse, however, we avoid the central monitoring of user activities; therefore, we describe our implementation of a recommender system that relies on assets posted by the same user, bypassing the need to centralise user behavior logging on the platform. Our recommender system relies on the semantic similarity of user's previous posts to help the user discover similar novel assets.

2 Media Asset Annotation and Management (MAAM)

MAAM is a specialised variant of the official MediaVerse node designed specifically for managing and utilising media annotations. The improvements made to the MV nodes in MAAM revolve around how media asset annotations are handled, such as changes to the indexing process and the incorporation of annotations into asset retrieval functions. In addition, MAAM supports retrieval functionality based on visual similarity combined with annotation filters, which can help with asset organisation. With a range of use cases in mind, MAAM has been designed to be flexible and adaptable, capable of handling a variety of annotation and asset organisation needs. It is important to note that while MAAM is a fork of the official MediaVerse node, it is still able to participate in the MediaVerse network as long as it adheres to the API response structure expected by other nodes. In that sense, MAAM shows that the official MediaVerse node project can be adopted by organisations with different needs and tailored according to them, while users of these forks can take advantage of the core functionalities of the MediaVerse network.

2.1 MAAM Architecture

MAAM, like the official MediaVerse node, is a service-oriented solution, consisting of three loosely interrelated services: the Digital Asset Management service (DAM), the Media Annotation Service, and the Near-Duplicate Detection (NDD) service. Figure 1 presents an overview of MAAM architecture and the connections between the three services.

The DAM service is the centrepiece of the MAAM platform, as it offers a robust set of core functionalities, ranging from user authentication and authorization to asset organization. Its presentation layer consists of a React-based user interface (UI) that provides a user-friendly and intuitive interface for accessing MAAM main functionalities. The business logic is implemented by a Spring-based application that follows the Model-View-Controller (MVC) architecture pattern. The persistence layer of MAAM is responsible for storing and maintaining the digital assets, their associated metadata and any other entity needed in the application. This layer includes two solutions: PostgreSQL and Elasticsearch. PostgreSQL is an efficient relational database management system, which is used as the primary application storage, replacing MongoDB that is used in the official MediaVerse nodes. The reason for that transition is that MAAM is designed as a tool for user-based media annotations that requires the storage and management of several relations between entities. That makes a relational database the appropriate choice for our needs, as relational databases ensure the consistency of relations at the storage layer, while in noSQL databases relation constraints must be implemented at application level.

Elasticsearch, on the other hand, serves as a search and analytics engine that enables efficient searching of digital assets. The DAM leverages the Elasticsearch ability to perform full-text search, allowing users to find assets based on keywords and phrases contained in the metadata. In addition, the aggregation feature of Elasticsearch is used for faceted search, based on specific attributes such as asset type, upload time, annotation type, etc. In addition, Elasticsearch supports advanced and sophisticated search functionalities, such as k-nearest neighbour (kNN) search based on dense feature vectors generated by the Media Annotation Service.

The Media Annotation Service is responsible for hosting and managing the AI annotation models and it is a critical component of the platform as it provides unique features that differentiate MAAM from other asset management tools. The Media Annotation Service is based on NVIDIA Triton Inference Server, which provides a flexible and scalable solution for deploying AI models in a production environment. Triton can host AI models implemented in most of the major frameworks, such as TensorFlow, PyTorch, ONNX, etc., making MAAM easily

extensible with new state-of-the art models. By leveraging Triton, the Media Annotation Service provides an efficient solution for running AI models, making it possible to perform almost real-time annotation of media assets as these are uploaded to the platform.

The communication between the DAM service and the Media Annotation Service is facilitated by gRPC[1], a high-performance framework for remote procedure calls. In the context of the MAAM platform, this asynchronous communication is particularly important, as the Media Annotation Service hosts multiple AI models with varying levels of complexity and processing requirements. The use of gRPC enables the DAM service to send multiple assets to the Media Annotation Service for annotation, without waiting for the annotation process to complete. The Media Annotation Service can perform the annotation in parallel, without blocking the overall asset management process and affecting its performance. As each asset is annotated by one of the supported models, the results are sent back to the DAM and the asset's metadata are updated asynchronously.

The Near-Duplicate Detection (NDD) service offers reverse search capabilities that enable users to quickly identify and remove duplicate assets from their media collections. NDD is based on our previous work for efficient video retrieval (Kordopatis-Zilos et al., 2022) and provides two key functionalities: image and video indexing and searching. The former function analyses the media asset's visual content provided and adds them to the corresponding index. The latter function searches the built index for near-duplicates to a query multimedia item and ranks the retrieved results based on their similarity to the query. The service provides calls to add images and videos to the relevant indexes using explicit URLs, search the index for near-duplicates given a query multimedia item, and create multimedia item collections for more efficient organisation and searching of near-duplicates. To enable this, the NDD service follows a service-oriented architecture, consisting of multiple modular services for feature extraction, indexing, and searching. Communication between the NDD service and other MAAM platform components is facilitated via a REST API exposed by the NDD service. The similarity calculation options are provided to the user to retrieve near-duplicates based on their requirements.

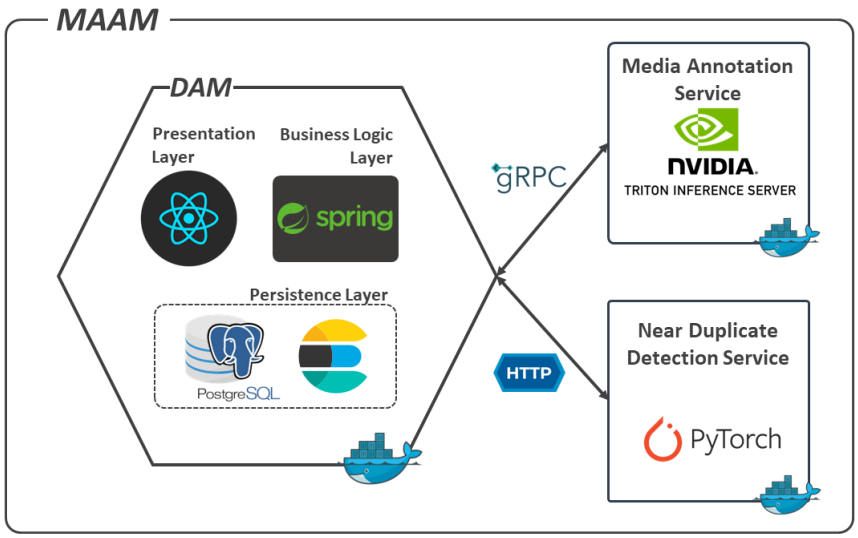

Figure 1: Architecture of the Media Asset Annotation and Management platform

[1] https://grpc.io/

2.2 MAAM Features

As the main purpose of MAAM is to leverage automatic annotations for asset organisation and retrieval, its main features include the core models used in the annotation procedure as well as a visual-based retrieval with the help of a state of the art multimedia representation model. By applying annotations as filters in asset search and retrieval, users can easily find content in large-scale collections that fits their needs and organise it in semantically coherent groups (the so-called Projects, in MAAM). In the same way, using already identified assets, the visual similarity feature can be used to further expand projects with assets that are conceptually similar. As already mentioned, for the asset annotation process, several models have been integrated to provide information ranging from objects and actions recognized in assets, to descriptive text captions that provide a user-friendly description of the assets.

2.2.1 Automatic Media Annotation

Automatic image captioning: Captioning can greatly enhance image retrieval by generating a free-text description for each image, providing additional context for the visual content. We use OFA[2], a state-of-the-art captioning model, and the produced text is indexed in Elasticsearch. Figure 2 shows three asset cards with the corresponding generated captions. By indexing these captions, we are able to retrieve assets even if the user has not provided any relevant metadata or textual description. For example, a user can retrieve the first asset at the left of the figure, which depicts gargoyles, even though the uploader provided no such information. In addition, automatic captions can be used as alternative text for the uploaded assets and therefore, improve the platform's accessibility by fulfilling the WCAG2.0 requirements[3]. It is worth mentioning that for the same examples, Microsoft's O365[4] produces the following: "We are unable to automatically generate alt text for this picture".

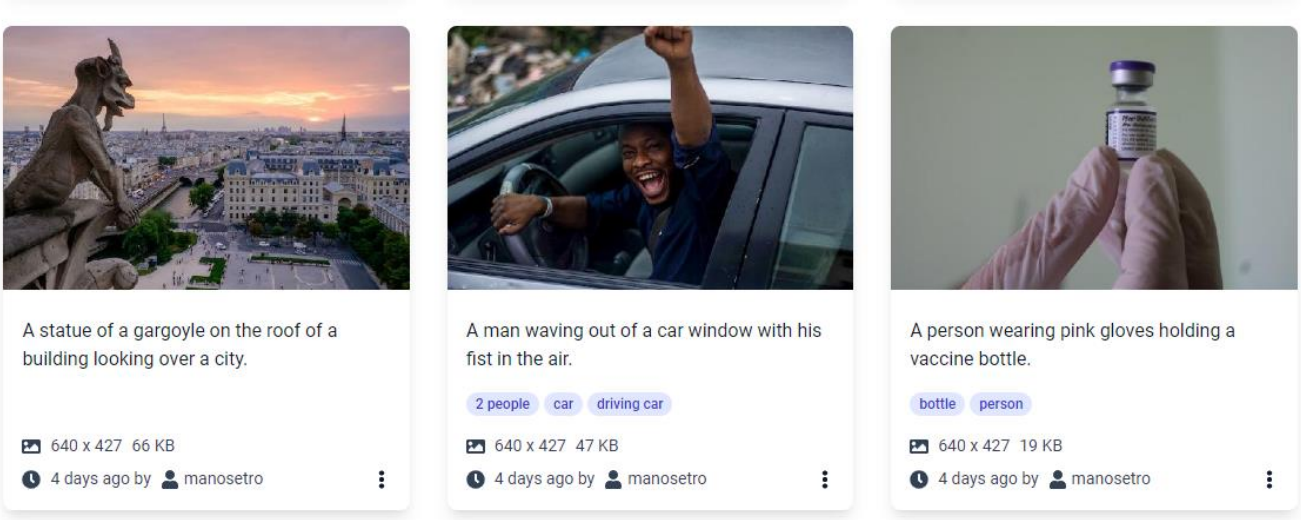

Figure 2: Examples of image captions and annotations.

Object detection: The object detection model is used to identify objects within images and video frames. We use the Faster R-CNN (Ren et al., 2015) with an InceptionV2 backbone (Ioffe & Szegedy, 2015), trained on the 80

[2] https://github.com/OFA-Sys/OFA
[3] https://www.w3.org/WAI/alt/
[4] https://www.office.com/

object classes of the MS COCO dataset[5]. In case of images, we detect and store the bounding box containing the corresponding object, while in videos we also provide time information. In both cases, a confidence score is also included. If a user wants to find images containing a specific object, they can filter assets by using the objects filter and the platform will return all images that have been annotated with that label.

Action recognition: For action recognition in videos, we consider the SlowFast R50 model (Feichtenhofer et al., 2019) trained on the Kinetics400 dataset. For images, since Kinetics400 contains videos and most action recognition models use 3D CNNs, we used a ResNet152 model (He et al., 2016) with the TSN approach (Wang et al., 2016) for training at frame level. At inference, we directly apply the frame level classifier to the images. In Figure 3 we provide three video examples with the corresponding recognized actions.

Face Detection and Recognition: For face detection, we consider the VGGFace2 model (Cao et al., 2018) trained on about 9k faces of the VGGFace2 dataset[6]. In case of videos, we apply the same model on randomly selected key-frames.

Content Moderation: The MAAM platform employs two moderation models to filter content, ensuring the platform remains safe and appropriate for all users. We trained two moderation models that are able to detect disturbing and Not Safe for Work (NSFW) content. For images, the moderation results, including a confidence score, are stored and indexed. For videos, the models are applied to key-frames and a video is considered NSFW or disturbing if at least one scene receives such a tag. At retrieval time, users have the option to include or exclude that type of content, while the UI uses these tags to blur the corresponding assets to prevent user exposure to potentially inappropriate content. For example, in Figure 4, the user has enabled the NSFW filter to get only those that have been tagged as such. Users can choose to reveal the actual content at their own discretion by clicking the reveal button at the upper right corner of each asset.

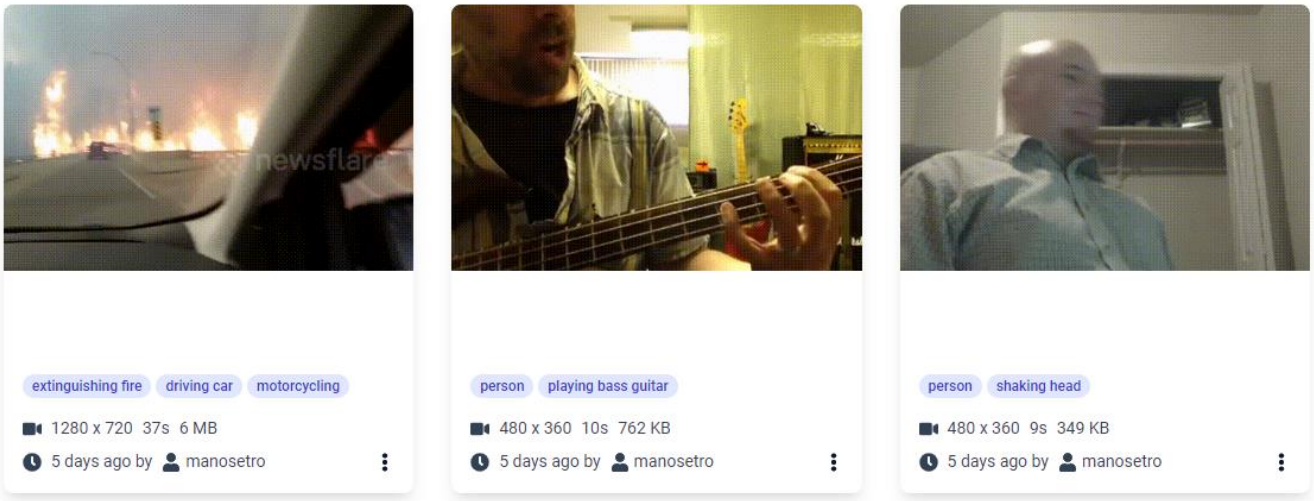

Figure 3: Examples of video assets with the corresponding recognized actions

[5] https://cocodataset.org/#home

[6] https://paperswithcode.com/dataset/vggface2-1

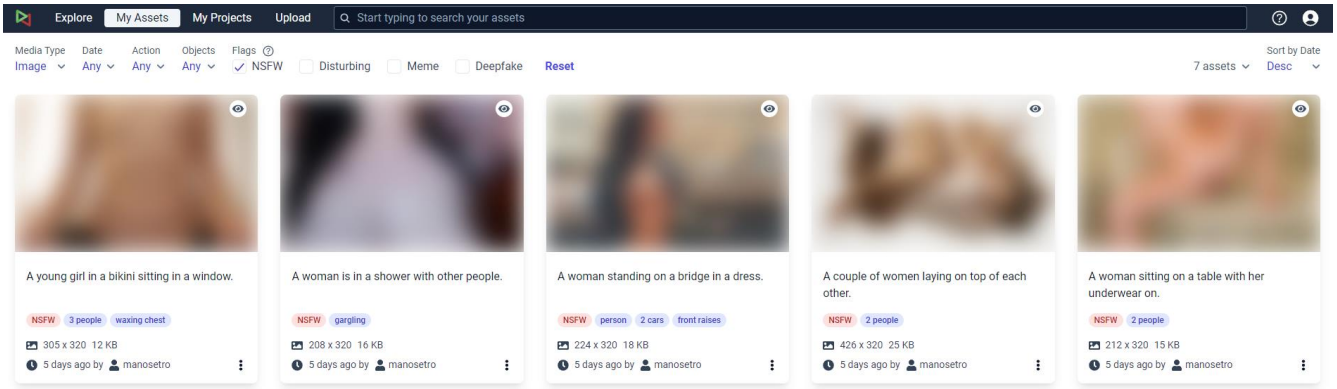

Figure 4: Assets flagged as NSFW

Meme Detection: For images, we determine whether they are memes or not using our previous MemeTector model (Koutlis et al., 2022). We apply the model to each image and the result is stored and indexed along with a confidence score. Figure 5 presents some examples, uploaded in MAAM.

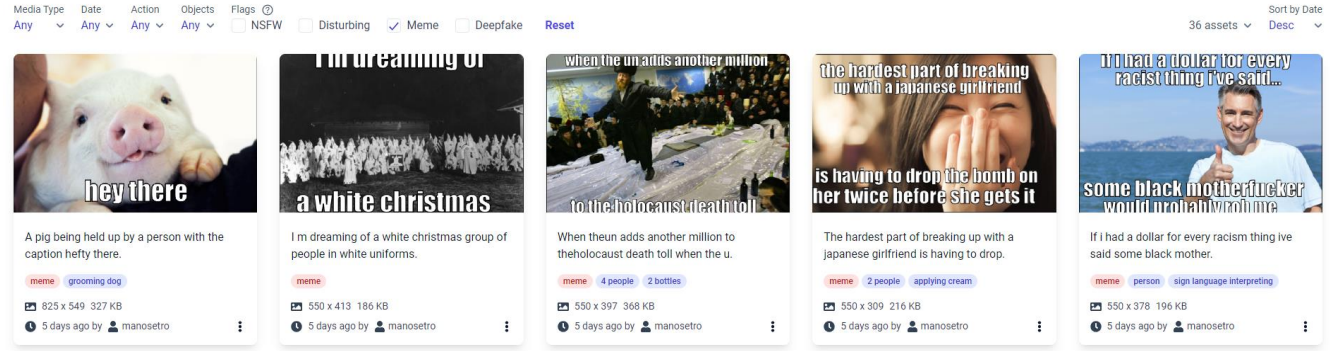

Figure 5: Examples of meme images uploaded in MAAM

2.2.2 Visual Concept Similarity and Retrieval

To support image similarity, we utilised the pre-trained CLIP model (Radford et al., 2021), which was trained on a large image-text dataset, consisting of around 400 million pairs. We used the ViT-B/32 version of the image encoder, which was obtained from its official GitHub repository. For each image uploaded in the DAM, MAAM receives from the Media Annotation Service a dense vector representation of 512 dimensions that encodes the semantic information of the image's visual content. This representation is then indexed in Elasticsearch and used to retrieve semantically similar content through the approximate k-nearest neighbour (kNN) search feature of Elasticsearch. This feature can be used in conjunction with all other features of Elasticsearch, allowing kNN queries to be combined with free text searches, filters, and aggregations. This flexibility enhances the user experience by allowing users to refine further the visual similarity search results to find content of specific interest. In addition to global image-level similarity, we support region-based retrieval. This allows users to select a specific region in an image by defining a bounding box, and retrieve content that is visually similar to that region in a more focused manner. To achieve this, the platform extracts a dense vector representation of the selected region in real-time, as it does for entire images during upload. This representation is then used in the same kNN search process to find relevant content. Figure 6 illustrates an example of the visual similarity feature in action. Using the region selection facility of MAAM, the user can focus the retrieval on the rainbow flag. By selecting an

image region containing a rainbow flag, MAAM retrieves assets containing the same flag since the dense vector extracted with CLIP encodes the meaning of the images and regions.

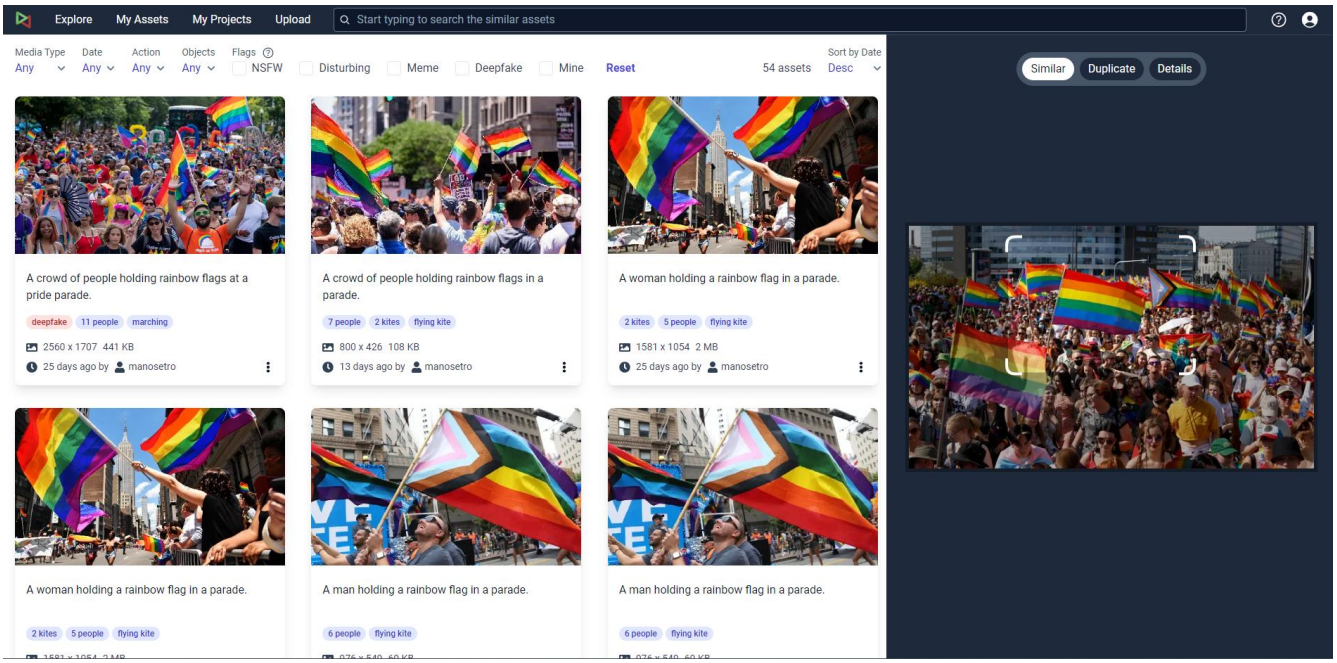

Figure 6: Retrieval of assets based on visual similarity

2.3 User-generated Models for Object Detection and Content Retrieval

In this section, we present a pipeline that lets users create custom models detecting user-selected objects in images. This builds on the work of deliverable D3.2 - Content Discovery and Recommendation, Annotation and Adaptation Framework[7] regarding the non-AI-expert model-building module for the face recognition and image classification tasks. In addition, the proposed process fits to the visual concept retrieval feature (presented in Section 2.2.2) enabling the users to create their own retrieval models for specific concepts, by having the potential to overcome some crucial shortcomings of the current implementation.

2.3.1 Objective

The purpose of this work is to provide the MediaVerse user with a tool that enables efficient and effective creation of personal models for the detection of objects (specific to the user's needs) in images. The need for simplicity both in terms of computations and user background (no machine learning skills are required) led our research to a similarity-based approach between anchor images depicting the object of interest and the candidate images existing in a database. All experiments, simulations and results can be found in the following GitHub repository: https://github.com/ckoutlis/mv-model-building-gui.

2.3.2 Proposed Pipeline

The proposed pipeline comprises the following steps:

[7] https://mediaverse-project.eu/wp-content/uploads/2022/07/D3.2-V1.0.pdf

- The user provides the system with a small number n of images depicting the object of interest in a central position.
- Prototypes are generated by storing the feature vectors extracted by a pre-trained model.
- For each candidate image one feature vector is extracted for the whole image and n_b more feature vectors corresponding to each bounding box (e.g., proposed by a region proposal network) are extracted using the same pre-trained model.
- Maximum similarity between each prototype and candidate feature vector is appropriately thresholded to decide on the presence of the object or utilised for ranking candidates in a retrieval setting.

2.3.3 Experiments on the MS-COCO Dataset

We consider the validation set of the MS-COCO[8] dataset (Lin & Maire, 2014) to assess the effectiveness of the proposed methodology. More precisely, we perform experiments on the first 16 object categories using two pre-trained feature extractors, namely the CLIP (Radford et al., 2021) ViT-L/14[9] pre-trained on ~400M image-text pairs and the ViT base (Dosovitskiy et al., 2020) pre-trained on ImageNet21k (Deng et al., 2009) obtained from the timm[10] library. Additionally, we consider the simplistic visual similarity measure UQI (computed between raw images) as baseline. For each category, we randomly select $n = 10$ object instances as prototypes by cropping the corresponding bounding box if it is of sufficient size (at least 120 pixels width and height). Then, the feature extractors process the cropped parts to obtain their representations. Consequently, these prototypes are compared against 1000 randomly selected candidate images to calculate the similarities through the cosine similarity metric. Most of the candidate images do not contain the objects of interest thus for the evaluation we consider the balanced accuracy metric that accounts for class imbalances. In the results section, we compute this score upon different thresholds and report the best performance along with the corresponding threshold.

2.3.4 Use Cases

In addition to the experiments on MS-COCO, we consider three real-world use cases for which we perform and assess the proposed pipeline. The three objects are:

- IBM logo, which is considered helpful in contexts involving public communication about popular brands;
- Flags, which is considered helpful in journalistic investigations about public events like summits;
- Swastika, which is considered helpful in journalistic and computational social science investigations about white supremacist and extreme right groups.

In order to acquire relevant images, we searched through Google's search engine the corresponding terms. After downloading ~300-400 images per object with the help of Image Downloader chrome extension[11] we first conducted deduplication with fdupes[12] and then manually cleaned the rest from irrelevant or low resolution content. Finally, we split each set to anchor and test images. The anchor images (from which the prototype feature vectors are extracted) are between 7 and 17 and the test images (positive candidates) are between 30 and 259 for each object. Furthermore, we consider a set of images that do not contain any of the three objects as negative candidates. We wanted to provide hard negative examples to our system so we performed the same

[8] https://cocodataset.org/
[9] https://github.com/openai/CLIP
[10] https://github.com/huggingface/pytorch-image-models
[11] https://chrome.google.com/webstore/detail/image-downloader/cnpniohnfphhjihaiiggeabnkjhpaldj
[12] https://manpages.ubuntu.com/manpages/bionic/man1/fdupes.1.html

collection process using the search terms “symbol”, “logo”, “air” and “countryside” which result in images with similar but distinct content. In Figures 7-9 the anchor images per object are illustrated.

2.3.5 Results

In Table 1, we illustrate the performance of our pipeline in terms of balanced accuracy per object category and feature extractor. It is observed that ViT outperforms CLIP and the baseline UQI methods in all three scenarios in which the similarity is computed (1) only based on the whole candidate image, (2) only the object-level bounding boxes and (3) both. In addition, ViT performs best when box similarity is considered while CLIP and UQI perform best when only the whole image’s similarity is computed. In Annex I, more detailed figures are provided (Figures A.1-A.6).

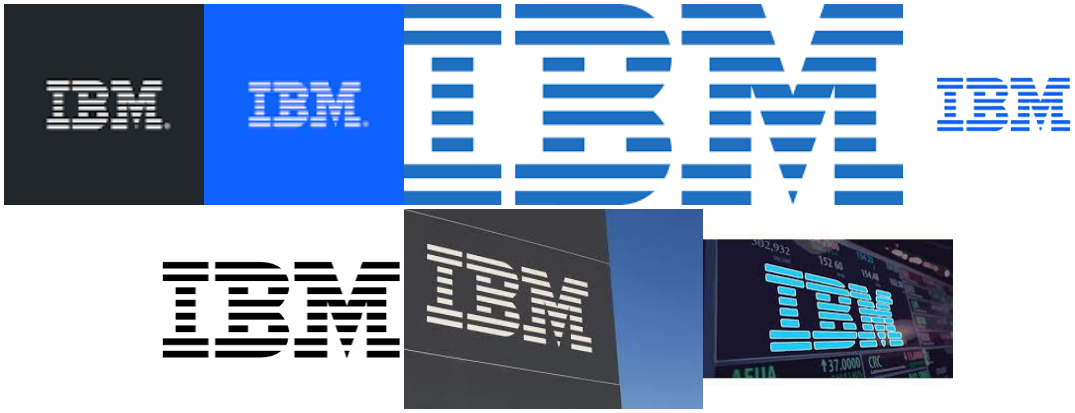

Figure 7: Anchor images for the IBM logo

Figure 8: Anchor images for the flags category

Figure 9: Anchor images for the swastika category

Table 1: Balanced accuracy per category and feature extractor (MS-COCO)

	cos. sim. w/ CLIP feat.			cos. sim. w/ ViT feat.			UQI on raw images		
	image	boxes	both	image	boxes	both	image	boxes	both
person	0.505	0.558	0.550	0.584	0.711	0.715	0.504	0.588	0.563
bicycle	0.701	0.800	0.800	0.723	0.785	0.775	0.505	0.513	0.513
car	0.590	0.613	0.613	0.731	0.784	0.784	0.525	0.570	0.558
motorcycle	0.702	0.649	0.649	0.893	0.844	0.841	0.523	0.517	0.515
airplane	0.867	0.656	0.669	0.916	0.841	0.863	0.667	0.578	0.579
bus	0.747	0.564	0.592	0.900	0.922	0.921	0.569	0.573	0.568
train	0.916	0.808	0.813	0.915	0.863	0.857	0.627	0.509	0.560
truck	0.672	0.678	0.678	0.729	0.839	0.831	0.520	0.558	0.537
boat	0.778	0.711	0.711	0.864	0.876	0.866	0.692	0.659	0.679
traffic light	0.623	0.669	0.669	0.794	0.764	0.752	0.502	0.543	0.521
fire hydrant	0.918	0.778	0.778	0.821	0.887	0.883	0.515	0.513	0.510
stop sign	0.610	0.805	0.805	0.793	0.752	0.771	0.521	0.543	0.563
parking meter	0.835	0.750	0.750	0.750	0.750	0.750	0.549	0.562	0.562
bench	0.557	0.589	0.589	0.646	0.671	0.674	0.581	0.563	0.556
bird	0.808	0.539	0.537	0.730	0.794	0.793	0.580	0.515	0.544
cat	0.890	0.778	0.778	0.889	0.900	0.900	0.503	0.504	0.501
average	0.733	0.684	0.686	0.792	0.811	0.811	0.555	0.551	0.552

Table 2 presents the average optimal threshold across MS-COCO object categories. We observe that for CLIP and ViT the optimal similarity threshold for verifying the object's presence is at low levels as opposed to UQI.

Table 2: Average optimal threshold across categories per feature extractor (MS-COCO)

	cos. sim. w/ CLIP feat.	cos. sim. w/ ViT feat.	UQI on raw images
average threshold	0.250	0.125	0.681

Similarly, Tables 3 and 4 present the balanced accuracy and optimal threshold respectively. While the best performance is similar and at acceptable levels the thresholds are very different from the MS-COCO experiments.

Table 3: Balanced accuracy per use case and feature extractor (use cases)

	cos. sim. w/ CLIP feat.	cos. sim. w/ ViT feat.	UQI on raw images
logo	0.954	0.829	0.502
flag	0.857	0.939	0.524
swastika	0.893	0.882	0.518
average	0.901	0.883	0.515

Table 4: Average optimal threshold across use cases per feature extractor (use cases)

	cos. sim. w/ CLIP feat.	cos. sim. w/ ViT feat.	UQI on raw images
average threshold	0.719	0.474	0.316

2.3.6 Conclusions

The proposed methodology with ViT feature extractor and considering bounding boxes in addition to the original image, produces very good results in terms of accuracy, but the difficulty in determining a universal optimal threshold makes it very challenging to deploy in the wild without loss in performance. Yet, the consideration of bounding box feature extraction as well as the pre-training on ImageNet21k of the ViT feature extractor seems to be a better alternative with respect to the current MAAM content retrieval framework, which is extracting features by applying CLIP on the whole candidate image. This is a good choice if someone wants to encode in only one vector all existing objects (irrespective of size) and at the same time low restrictions on the number of possible objects to be detected. However, it has two shortcomings: (1) semantically-only correlated instances returned in high rank; and (2) not very informative embeddings in cases of one-object images. On the other hand, CLIP applied on bounding boxes containing only one object is not an optimal choice as CLIP has been trained on images with many objects and semantics described by whole images and not just labels.[13] Of course ViT pre-trained on ImageNet21k is not a good choice for feature extraction on whole images as it has been trained on single-object images. ViT on candidate images bounding boxes and ViT on single-object user-provided anchor images is the new proposed approach for content retrieval in MAAM and is expected to work better than CLIP on whole image.

[13] To address this, RegionCLIP has been proposed: https://github.com/microsoft/RegionCLIP. This is a CLIP architecture but trained on a different dataset containing object-description pairs rather than image-text. Moreover, CLIP and RegionCLIP do not produce feature vectors in the same vector space, so similarities cannot be computed between the two.

3 New Media Understanding and Annotation Service

In this section, we present the model additions and improvements in existing models of the Media Annotation Service, as well as a novel approach for model compression as a solution to the evolving number of offered AI models and the consequent computational resource demands.

3.1 Image Meme Fine-grained Classification

Here, we report the progress of T3.1 with respect to the task of image meme fine-grained classification.

3.1.1 Introduction

The main goal is to build a deep learning-based architecture for fine-grained classification of Internet image memes. The main categorisations of memes are hateful/non-hateful, offensive/non-offensive, humour type (humorous, sarcastic, offensive, and motivational) and sentiment (positive, negative, and neutral). Figure 10 shows pertinent examples. The problem is expected to be better solved by multimodal means because of the inherent multimodal nature of image memes, but unimodal approaches have shown comparable results in several studies and thus should also be considered as part of an ablation analysis. Our main contribution is the adoption of a dual stage modality fusion approach. The first fusion stage produces feature vectors containing modality alignment information, which expresses the non-trivial connection between modalities in image memes. The second fusion stage leverages the power of a Transformer encoder to learn inter-modality correlations at the token level and yield an informative and discriminative representation. We also exploit external knowledge to provide richer input to the model's cross attention modules as well as background image caption supervision as a regularising component that constrains the model from producing only task specific features leading to potential over-fitting. Code available on GitHub[14].

3.1.2 Methodology

Here, we present the proposed approach to address the image meme fine-grained classification task. We consider a multimodal pre-trained deep neural network, for feature extraction of both image meme modalities, namely image and text. In addition, we incorporate external knowledge to our model by providing the predicted demographics of depicted people. A dual-stage modality fusion module processes the three groups of inputs i.e., image, text, and external knowledge. Finally, a dense layer produces the classification output and a caption supervision is imposed for regularisation purposes. Figure 11 presents the proposed model architecture (the symbol $\otimes$ denotes broadcastable element-wise multiplication).

[14] https://github.com/ckoutlis/memefier

Figure 10: Examples of image memes from different classes

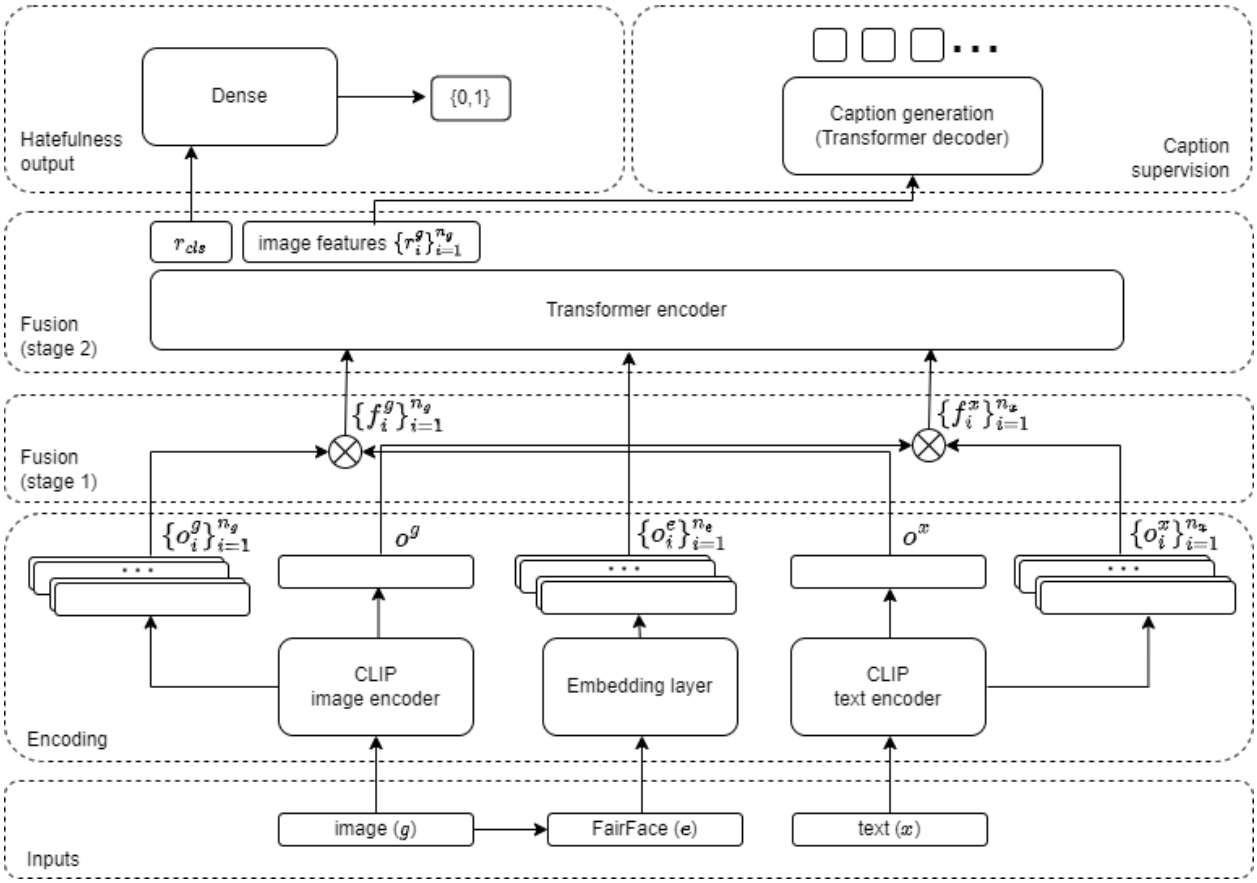

Figure 11: Model architecture

Modality Encoding

Kiela et al. (2020) claim that considering two unimodally pre-trained feature extractors for image and text, respectively, results in worse performance compared to feature extractors trained with a multimodal objective, when the task of interest is multimodal in nature. Thus, for the encoding of the image and the text, we consider a multimodally pre-trained model, and more precisely, we opt for CLIP (Radford et al., 2021). After processing the image $g \in \Re^{h \times w \times 3}$ and the text $x \in \aleph^L$, where h and w are the width and height of the image while L is the text's number of words, by CLIP's image and text encoders, we get the image embedding $o^g \in \Re^d$ and its patch embeddings $\{o_i^g\}_{i=1}^{n_g} \in \Re^{n_g \times d}$ as well as the text embedding $o^x \in \Re^d$ and its token embeddings $\{o_i^x\}_{i=1}^{n_x} \in \Re^{n_x \times d}$, where n_g and n_x are the number of patches and the number of tokens, respectively.

External Knowledge Retrieval and Encoding

Incorporating external knowledge, on top of the given visual and textual cues, in the processing of image memes, is inspired by the fact that hate is mainly targeted to certain groups of the population, who are minorities or more vulnerable, in general (e.g., Muslims, women, etc.). Without the external knowledge retrieval module, the trained models decide purely based on image-text learned correlations, which neglects real world understanding as valuable input. Utilising such a signal hopefully creates richer feature embeddings by enabling deeper semantics information exchange among the modalities. Other researchers have realised this issue as well, for instance in (Kiela et al., 2020, p. 3) the authors mention:

"The definition employed here has some notable exceptions, i.e., attacking individuals/famous people is allowed if the attack is not based on any of the protected characteristics listed in the definition. Attacking groups perpetrating hate (e.g., terrorist groups) is also not considered hate. This means that hate speech detection also involves possibly subtle world knowledge."

We incorporate external knowledge to our model with the following procedure. For each image we get FairFace predictions[15] regarding gender, race and age of all depicted persons (if any) and denote it $e \in \aleph^{3\times n_p}$, where n_p is the number of persons and $n_e = 3 \times n_p$. We then encode this information through a typical embedding layer which is trained along with the rest of the model and produce the corresponding embeddings $\{o_i^e\}_{i=1}^{n_e} \in \Re^{n_e \times d}$.

Fusion

We consider a dual-stage modality fusion approach. During stage 1, we produce token-level modality representations that are aware of the level of alignment with the other modality. More precisely, we compute:

$$f_i^g = o_i^g \otimes o^x$$
$$f_i^x = o_i^x \otimes o^g$$

where $\otimes$ denotes element-wise multiplication, to get the fused image and text features, respectively. During stage 2, a Transformer encoder $T(\cdot)$ processes f_i^g, f_i^x and o_i^e along with a learnable classification token $< CLS >$ and produces the corresponding feature representations:

$$\left[r_{cls}, \quad \{r_i^g\}_{i=1}^{n_g}, \{r_i^x\}_{i=1}^{n_x}, \{r_i^e\}_{i=1}^{n_e}\right] = T\left(\left[< CLS >, \{f_i^g\}_{i=1}^{n_g}, \{f_i^x\}_{i=1}^{n_x}, \{o_i^e\}_{i=1}^{n_e}\right]\right)$$

Classification

As classification head for the hatefulness output we consider a typical fully-connected and sigmoid-activated layer of one unit $D(r_{cls})$.

Caption Supervision

In the typical multi-modal training setting, the vision encoder is likely to extract reduced image features that are advantageous only for the hatefulness detection, ignoring part of the background's semantics. In this case, the model will probably overfit and yield spuriously good performance. In order to mitigate this potential deviation, we propose to consider an additional supervision signal, namely by reconstructing a description of the background image through a standard image captioning decoder. This will enable the vision encoder to preserve the image's semantics in addition to building features pertinent to the hatefulness property. In order to achieve this we first crop the visual part of the image memes extracted with VPU and then consider the OFA generated caption as the target. Finally, a Transformer decoder is utilised to produce the caption based on the fused image features $\{r_i^g\}_{i=1}^{n_g}$.

3.1.3 Experimental Setup

In this section, we describe the datasets and the details of the conducted experiments.

[15] We utilise the pretrained models provided in https://github.com/dchen236/FairFace.

Datasets

- **Facebook Hateful Memes** (Kiela et al., 2020): This dataset has been released in the context of the Hateful Memes Challenge. It contains 10K image meme instances namely, 8.5K training data, 0.5K validation data (dev) and 1K test data. It is annotated by humans in 4 phases (filtering, meme construction, hatefulness rating and benign confounders). We do not use the test split for which the labels are not released, but we evaluate all models and report results on the dev set.
- **Memotion7k** (Sharma et al., 2020): This dataset has been released in the context of a challenge at SemEval-2020. It contains 9871 image meme instances, i.e. 1K trial data, 6992 training data and 1879 test data. It is annotated by humans through Amazon's Mechanical Turk with regards to the following tasks: (a) sentiment prediction (negative, positive, neutral), (b) overall emotion prediction (humor, sarcastic, offensive, motivational) and (c) estimation of the corresponding intensities. We randomly split training data to training and validation sets at 90%-10%, and report results on the test data. Trial data are not publicly available and thus not used here.
- **MultiOFF** (Suryawanshi et al., 2020): This rather small dataset contains 743 image meme instances manually annotated as either offensive or non-offensive. The training set has 445 images and the validation and test sets have 149 each.

Baselines

We consider three baseline models, one that processes only the image, one that processes only the text and one that processes both modalities:

- image only: ResNet18 → FC(dim * 2) → FC(num_classes)
- text only: Embedding → LSTM(dim) → FC(dim * 2) → FC(num_classes)
- multimodal:
    - image encoder: the image only baseline (w/o classification layer) with 2 FC on top
    - text encoder: the text only baseline (w/o classification layer) with 2 FC on top
    - early fusion: concatenate the above encoders' output vectors and process them through 2 FC layers the last being activated by a softmax function

Hyperparameter Grid

For the baselines we conduct experiments for hyperparameter tuning accounting for different (i) initial learning rates (1e-2, 1e-3, 1e-4, 1e-5), (ii) ResNet18 visual feature extractor being pre-trained on ImageNet or not (True, False), (iii) number of hidden dimensions for the fully connected as well as for the LSTM (64, 128, 256) and (iv) number of LSTM layers (1, 3). We report the maximum performance.

For the proposed method we consider the following hyperparameter grid: (i) initial learning rates (1e-4, 1e-5), (ii) number of epochs (16, 32), (iii) contribution of the caption supervision α (see 'Implementation details' section) to the final loss function (0.2, 0.8), (iv) model dimension (512, 1024), (v) Transformer encoder settings:

- 4 heads, 512 feedforward dimension, 1 layer
- 16 heads, 2048 feedforward dimension, 3 layers

(vi) Transformer decoder settings:

- 64 input dimension, 4 heads, 64 feedforward dimension, 1 layer
- 256 input dimension, 16 heads, 256 feedforward dimension, 3 layers

Implementation Details

For the baselines, the input training images are resized to 256, then randomly cropped at 224, randomly horizontally flipped and finally standardised per channel using the ImageNet statistics. The validation images are only resized to 224 and standardised. For the text, we consider lowercasing and removing punctuation, numbers and double spaces. The vocabulary size is determined by the number of words having at least five occurrences in the corpus and the maximum sequence length by the 90% quantile of the distribution of lengths. Binary cross entropy is employed for the binary classification tasks and the multi-label classification tasks (e.g., Memotion7k task b), while categorical cross entropy is employed for the multi-class classification tasks. As optimizer we opt for Adam, training for 10 epochs, with batch size 128 and the learning rate is reduced by 10x at epoch 5.

For the proposed method, we consider the specific image preprocessing pipeline provided by the CLIP model and the text preprocessing pipeline considered for the baselines. For the input text vocabulary, we consider the same approach as for the baselines while for the captions vocabulary we consider all words and the actual maximum sequence length. We conduct the model training using Adam optimizer, binary cross entropy loss for the hatefulness output and categorical cross entropy for the caption supervision. Batch size is set to 32 due to memory constraints. The initial learning rate is reduced by a factor of 10 after half of the training epochs. The two loss functions are combined as below:

$$\mathcal{L} = \mathcal{L}_{hate} + \alpha \cdot \mathcal{L}_{caption}$$

where α denotes the contribution of the caption supervision, $\boldsymbol{\mathcal{L}_{hate}}$ denotes binary cross-entropy, and $\boldsymbol{\mathcal{L}_{caption}}$ denotes categorical cross-entropy.

Evaluation Protocol

We report F1 score for Memotion7k and MultiOFF datasets and AUC for Facebook Hateful Memes dataset in order to be comparable with the state of the art, which uses these conventions. Additionally, we report the accuracy metric for all our experiments.

3.1.4 Results

In Table 5, we present the performance of the baseline models as well as the proposed method on the Facebook Hateful Memes dataset. We observe that our best configuration achieves 0.801 AUC and 0.736 accuracy, namely +28.8% relative improvement compared to the best baseline performance in terms of AUC. In addition, all experimental settings of the proposed method, exhibiting a minimum AUC of 0.673, surpass the baselines and the results seem very robust to hyperparameter changes as the top-5 settings exhibit a minimum variation of 0.014 range. In Figure 12, we get a broader picture of how our method's hyperparameters affect the results. It seems that learning rate is the most sensitive parameter to tune while the rest produce similar score distributions across different values.

Table 5: Baselines' and proposed method's performance on FBHM

Method	Accuracy	AUC
Baselines		
image	0.530	0.573
text	0.544	0.622
multi	0.554	0.613
Top-5 hyperparameter settings of the proposed method		
• learning rate: 0.0001 • epochs: 16 • a: 0.2 • d: 1024 • encoder: {'h': 16, 'dff': 2048, 'L': 3} • decoder: {'d': 64, 'h': 4, 'dff': 64, 'L': 1}	**0.736**	**0.801**
• learning rate: 0.0001 • epochs: 16 • a: 0.8 • d: 1024 • encoder: {'h': 16, 'dff': 2048, 'L': 3} • decoder: {'d': 64, 'h': 4, 'dff': 64, 'L': 1}	0.726	0.795
• learning rate: 0.0001 • epochs: 32 • a: 0.2 • d: 1024 • encoder: {'h': 16, 'dff': 2048, 'L': 3} • decoder: {'d': 64, 'h': 4, 'dff': 64, 'L': 1}	0.736	0.794
• learning rate: 0.0001 • epochs: 16 • a: 0.2 • d: 1024 • encoder: {'h': 4, 'dff': 512, 'L': 1} • decoder: {'d': 64, 'h': 4, 'dff': 64, 'L': 1}	0.708	0.789
• learning rate: 0.0001 • epochs: 32 • a: 0.2 • d: 1024 • encoder: {'h': 4, 'dff': 512, 'L': 1} • decoder: {'d': 64, 'h': 4, 'dff': 64, 'L': 1}	0.708	0.789

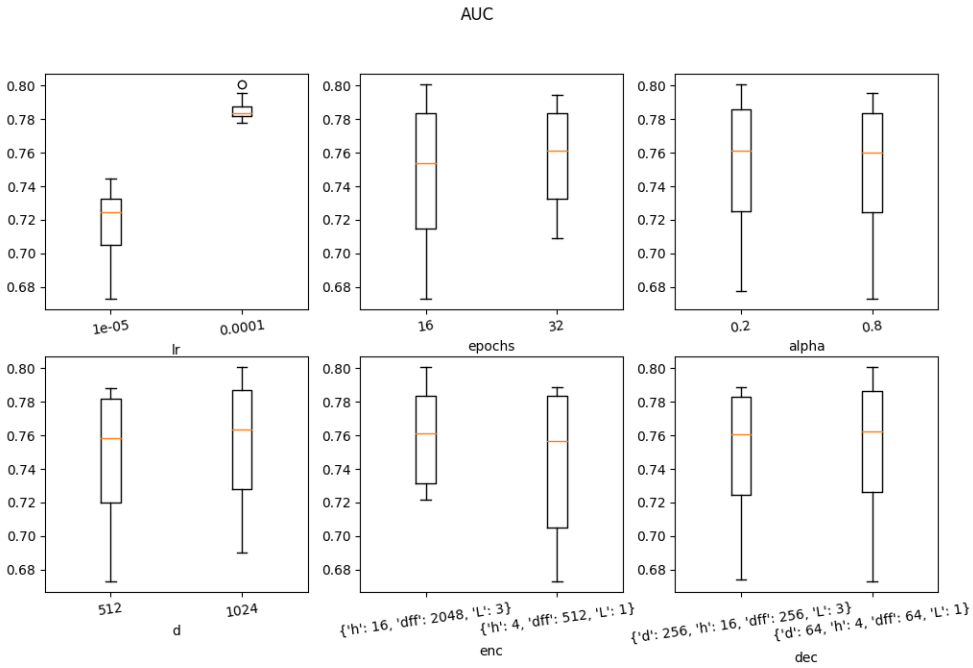

Figure 12: Box plots of AUC with respect to different hyperparameter values

Table 6 demonstrates the performance of the baseline models and the proposed method on the three tasks of Memotion7k. Regarding the baselines (i.e., image, text, and multimodal), inconsistency is observed between accuracy and F1 score while the combination of input modalities does not lead to best results in most cases. The latter entails the dominance of one of the two modalities in terms of exploitable information for solving the task of interest and it has already been demonstrated in previous papers as well (Das et al., 2020). The proposed method performance is better than the baselines in all tasks in terms of both accuracy and F1 score.

Table 6: Performance of baselines and proposed method on Memotion7k (tasks a, b and c)

Method	Task a	Task b	Task c
Accuracy			
image	0.560	0.703	0.474
text	0.559	0.703	0.472
multimodal	0.562	0.703	0.475
proposed	**0.562**	**0.704**	**0.476**
F1			
image	0.333	0.502	0.315
text	0.350	0.481	0.279
multimodal	0.346	0.493	0.310
proposed	**0.396**	**0.519**	**0.343**

In Table 7, we illustrate the performance of the proposed method and the baseline models on the MultiOFF dataset. We observe that image processing surpasses the text processing scores, while the combination of the

two modalities leads to the best outcome in terms of both accuracy and F1 score. The proposed method's performance is better than the baselines in terms of accuracy while almost identical in terms of F1 score.

Table 7: Performance of baselines and proposed method on MultiOFF

Method	Accuracy	F1
image	0.638	0.619
text	0.571	0.508
multimodal	0.671	**0.626**
proposed	**0.685**	0.625

3.2 Model Compression

In this section, we present the methodologies employed for compressing Convolutional Neural Networks (CNNs) that we deploy through the Media Annotation Service.

3.2.1 Knowledge Distillation Combined with Pruning

Pruning and Knowledge Distillation (KD) constitute two widely applied approaches for compressing a neural network. The objective of KD is to transfer knowledge from a powerful teacher network to a smaller and faster one, in order to extend its performance capabilities, while pruning aims to discard the redundant parameters of a neural network in order to reduce its storage requirements or/and the inference time. Taking advantage of the capabilities of these two model compression techniques, we developed a state-of-the-art method, termed InDistill (Sarridis et al., 2022), that combines knowledge distillation and channel pruning in a unified framework for the transfer of the critical information flow paths from a heavyweight teacher to a lightweight student. Such information is typically collapsed in previous methods due to an encoding stage prior to distillation. On the contrary, InDistill leverages a pruning operation applied to the teacher's intermediate layers reducing their width to the corresponding student layers' width. In that way, we force architectural alignment enabling the intermediate layers to be directly distilled without the need of an encoding stage. Additionally, a curriculum learning-based training scheme is adopted considering the distillation difficulty of each layer and the critical learning periods in which the information flow paths are created.

3.2.2 Quantization

Quantization approaches enable neural networks to perform computations and store tensors using fewer bits than the standard floating point precision (i.e., FP32). In particular, by applying quantization, a model can execute some or all of its operations on tensors with lower precision, while enabling high-performance vectorized operations on various hardware platforms. PyTorch, which is an open-source machine learning framework, offers several types of quantization methodologies, namely:

- **Dynamic quantization**: The weights are quantized ahead of time, while the activations are quantized dynamically during inference. This approach is recommended for LSTM architectures where the model

execution time is dominated by loading the weights rather than the complexity of the matrix multiplications.

- **Static quantization aware training**: During the training phase of a network, all the calculations are performed in FP32 precision, with the quantization module modelling the effects of quantization by clamping and rounding to simulate the effects of INT8. Then, both the weights and the activations are quantized and the activations are fused into the preceding layers.
- **Static quantization**: Similarly to the static quantization aware training, this approach is employed to quantize both weights and activations, while requiring only a calibration step to determine the optimal quantization parameters for activations, instead of re-training the model.

Here, we focus on the static quantization with INT8 precision, as most of the MediaVerse models are pre-trained CNNs on large visual datasets. Table 8 presents the code for performing static quantization on a user-defined neural network.

Table 8: Static quantization in Pytorch

```
import torch
from torch.ao.quantization import (
  get_default_qconfig_mapping,
  get_default_qat_qconfig_mapping,
  QConfigMapping,
)
import torch.ao.quantization.quantize_fx as quantize_fx
import copy

model_fp = UserModel()

# post training static quantization
model_to_quantize = copy.deepcopy(model_fp)
qconfig_mapping = get_default_qconfig_mapping("qnnpack")
model_to_quantize.eval()

# prepare
model_prepared = quantize_fx.prepare_fx(model_to_quantize, qconfig_mapping,
example_inputs)

# calibrate
for imgs, _ in train_loader:
    model_prepared(imgs)

# quantize
model_quantized = quantize_fx.convert_fx(model_prepared)
```

3.2.3 Limitations

The architectural characteristics and the volume of training data of each MediaVerse model constitute the two major criteria for selecting the proper model compression methodology. In particular, InDistill can only be applied on 2D CNN networks and requires training the lightweight model from scratch. On the other hand, the

static quantization by PyTorch supports several network architectures and does not require any re-training procedure, which set it as an optimal solution for pretrained models on large-scale datasets that require extreme resource power for training them from scratch.

3.2.4 Creation of Efficient MediaVerse Models

The content moderation models developed in the context of the MediaVerse project, namely Disturbing Content Detection (DCD) model and NSFW detection model are 2D CNN models trained on small and mid-scale datasets, respectively. Thus, InDistill methodology can be employed in order to reduce their inference times and storage requirements, while demonstrating competitive performance in terms of accuracy. As presented in deliverable D5.4 - Content Moderation Toolset[16], the EfficientNet-b1 architecture was selected for both DCD and NSFW detection models. Here, we employ InDistill to transfer the knowledge of the EfficientNet-b1 (i.e., teacher) to EfficientNet-b0 models (i.e., student). Table 9 demonstrates the performance of the compressed models in terms of accuracy, inference time, and storage size compared to the original content moderation models. Although the teacher models demonstrate high efficiency in the first place, InDistill further achieves a compression rate of 38.3% at minimal performance costs.

Table 9: Evaluation of InDistill on content moderation models

Models	Accuracy (%)	Inference time (seconds)	Storage size (MB)
DCD			
EfficientNet-b1-teacher	95.0	0.077	49.9
EfficientNet-b0-student	93.4	0.062	30.8
EfficientNet-b0-InDistill	94.3	0.062	30.8
NSFW			
EfficientNet-b1-teacher	99.0	0.077	49.9
EfficientNet-b0-student	98.5	0.062	30.8
EfficientNet-b0-InDistill	99.0	0.062	30.8

Note: Accuracy values refer to the Disturbing Images Dataset and the Pornography-2k dataset (see D5.4) for the DCD and NSFW tasks, respectively. EfficientNet-b0-student refers to the student model architecture trained from scratch without any distillation or pruning.

As regards the pre-trained Face Recognition model (i.e., Resnet50) employed in the context of MediaVerse as presented in deliverable D3.1 - Next Generation Content Model and Algorithms for New Media Types[17], we opted for the static quantization methodology for increasing its efficiency. Table 10 demonstrates the results of FR model, trained on VGGFace2 dataset, before and after the quantization. In particular, the quantized model reduces the inference time by 63.1% (i.e., 0.021 sec.) and the model's size by 74.9% (i.e., 39.4 MB), while

[16] https://mediaverse-project.eu/wp-content/uploads/2022/10/MediaVerse_D5.4-V1.0.pdf
[17] https://mediaverse-project.eu/wp-content/uploads/2021/10/MediaVerse_D3.1_NextGeneration-ContentModel-and-Algorithms-for-NewMediaTypes.pdf

demonstrating 1.3% accuracy drop compared to the unquantised model. Finally, neither InDistill nor PyTorch's static quantization can be applied to the Object Detection model described in D3. 1 - Next Generation Content Model and Algorithms for New Media Types , as it requires extremely high computational resources to train and quantization libraries do not support the Feature Pyramid Networks that are involved in the FasterRCNN architecture. However, an architecture introduced recently, namely YOLOv5, exhibits considerably higher efficiency than FasterRCNN's, while demonstrating enhanced performance in terms of mean Average Precision (mAP) as well. Table 11 presents the performance of the efficient object detection model, namely YOLOv5-s, compared to the FasterRCNN model trained on MS-COCO dataset.

Table 10: Evaluation of static quantization on Face Recognition model

Models	Accuracy (%)	Inference time (seconds)	Storage size (MB)
Resnet50	86.4	0.057	157.4
Resnet50 quantized	85.1	0.021	39.4

Table 11: Evaluation of YOLOv5-s efficient architecture for object detection

Models	mAP (%)	Inference time (seconds)	Storage size (MB)
FasterRCNN-ResNet50	36.9	0.984	159.8
YOLOv5-s	43.3	0.041	27.9

3.3 Saliency Detection in Videos

Here we explain the task of saliency prediction and present the novel architecture we designed to challenge this task, based on a pure-Transformer network.

3.3.1 Introduction

Saliency Prediction (SP), in the context of Computer Vision, aims to model the visual attention mechanism of humans. Visual attention refers to the human brain ability to select relevant sensory information for preferential processing, improving performance in visual and cognitive tasks (Zanca et al., 2020). Visual attention occurs in two distinct phases. The first phase involves the acquisition and parallel processing of visual feature maps, while the second phase entails merging the information from these maps to select a single location for further and intricate computations and reasoning. The computational representation of this process is challenging, particularly when considering its temporal dynamics.

The outcome of Saliency Prediction is a salient mask, which represents the probability distribution of the location where humans would fixate on an image. SP has demonstrated utility in event prediction (Shimoda & Yanai, 2016), semantic segmentation (Gan et al., 2015), video surveillance, and video captioning by enabling extraction of information from visual input. In addition, SP holds substantial potential to advance understanding of human

brain mechanisms involved in image perception. In the context of MV, saliency detection will serve as a guideline that will provide users with insight information for their media assets (i.e., static images, videos, and 360-videos). Particularly, in combination with Fader, saliency maps can be a useful tool, which will help users understand which part of the image has no visual interest in order to place content (e.g., buttons, text, etc.) on these areas.

Traditional/Conventional approaches for Video Saliency Prediction (VSP) take advantage of 2D-CNNs for the extraction of the visual/spatial features in combinations with LSTMs for the aggregation of the temporal features and 3D-CNNs, which process spatio-temporal information in a simultaneous manner. However, a disadvantage of CNN architectures is the fact that they cannot model long-range dependencies for video representations.

Here, in order to address the problem of VSP, we employ a transformer-based architecture to cope with the long-range dependencies of the spatio-temporal data. Vaswani et al. (2017) introduced Transformer networks initially, for NLP problems. Later, Dosovitskiy et al. (2020) introduced Vision Transformer in a computer vision task, such as image classification. The basic element of Transformer networks is the self-attention mechanism that Figure 13 shows. The self-attention mechanism in transformers is implemented using three sets of linear transformations, known as “query”, “key”, and “value”. These transformations are learned during training and are used to compute the attention scores between pairs of input image patches. The attention scores are then used to compute weighted sums of the values, which are used to compute the new representations. In general, the self-attention mechanism is a powerful way to capture relationships between different parts of sequence (temporal/spatial).

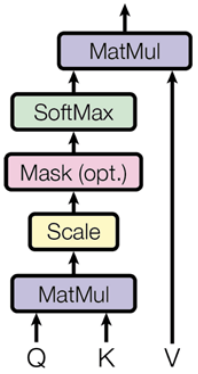

Figure 13: Attention computation

The decision of employing a pure-transformer architecture is based on the fact that transformers have shown a strong capability to capture both short-range and long-range correlations while not constrained by the inductive bias of CNNs (Cao et al., 2022). Our approach is based on the Visual Saliency Transformer (Liu et al., 2021) and introduces a temporal module to adapt it for video settings.

3.3.2 Typical Approaches

The first models that had been used to face the task of video saliency detection, leveraged feature extraction methods based on the low-level characteristics of the image, such as texture, contrast, colour, and orientation (Itti et al., 1998; Le Meur et al., 2006; Cerf et al., 2018, Walther et al., 2009, Erdem et al., 2013). However, as a dynamic medium, videos offer a blend of spatial and temporal information. Each frame contains its own spatial details, while the continuity between consecutive frames provides the crucial temporal context. When it comes to capturing human attention, both low-level visual cues and high-level semantic content play a role, but it is the interplay between these features across time that really draws us in. To overcome this issue, many video saliency

detection methods used temporal recurrence to predict the saliency map. Some models, such as DeepVS (Jiang et al., 2018) and ACLNet (Wang et al., 2019), used sub-networks for objects and motion, with ConvLSTM modules for prediction. SalEMA (Linardos et al., 2019) compared exponential moving average and ConvLSTM for video saliency modelling. STRA-Net (Lai et al., 2019) used two-stream models with dense residual cross-connections and multiple local attentions for saliency map prediction. SalSAC (Wu et al., 2020) improved robustness with a shuffled attention module and correlation-based ConvLSTM. ESAN-VSP (Chen et al., 2021) used a multi-scale deformable convolutional alignment network and Bi-ConvLSTM for motion prediction. UNISAL (Droste et al., 2020) is a unified image and video saliency detection model, which can extract static features and determine whether to predict temporal information through a controllable switch. It also uses domain adaptation technology for high-precision saliency detection on various datasets.

The majority of contemporary state-of-the-art models rely on 3D convolutional neural network (CNN) architectures that are capable of processing spatial and temporal features in a synchronised manner. RMDN (Bazzani et al., 2016) is a model that is based on C3D (Tran et al., 2015) and exploits temporal consistency in videos in a hierarchical manner. It employs deep 3D convolutional features to represent spatial and short-term time relations at the clip level, while a long short-term memory network aggregates clip-level representations of sequential clips, thus expanding the temporal domain. However, most 3D-CNN approaches use S3D, as a backbone. TASED-Net (Min et al., 2019) is a powerful end-to-end 3D fully convolutional network that uses auxiliary pooling to obtain switches with a reduced temporal dimension, enabling max-unpooling layers of the prediction network to function correctly. HD2S's 3D-CNN (Bellitto et al., 2021) extracts multi-scale features that are combined for the final input. The different abstraction levels enable the model to learn both generic and dataset-specific features. ViNet (Jain et al., 2021) is an encoder-decoder architecture that is visual-only and uses common deep-learning concepts. It claims that visual saliency prediction is agnostic to audio, which is contrary to many other publications in the field. Lastly, TSFP (Chang et al., 2021) is a multi-scale 3D encoder-decoder architecture. The encoder generates a feature pyramid of various scales that contain rich temporal-spatial semantic features, which are then decoded using a hierarchical 3D convolutional decoder.

3.3.3 Saliency Detection in MediaVerse: MV-SD Model

Previous studies (Wu et al., 2020; Wang et al., 2021) have shown that attention, particularly for capturing long-range information, has powerful representation capabilities that could contribute to predicting gaze. However, the use of transformers in video saliency prediction has not been explored. Therefore, in our implementation, we aim to investigate the benefits and potential applications of transformer components in VSD. In this section, we describe the details of our method to predict visual saliency in videos. Figure 14 shows the overall architecture of the network. We designed a pure-Transformer architecture, whose main components are a transformer encoder, a temporal transformer module and a transformer decoder.

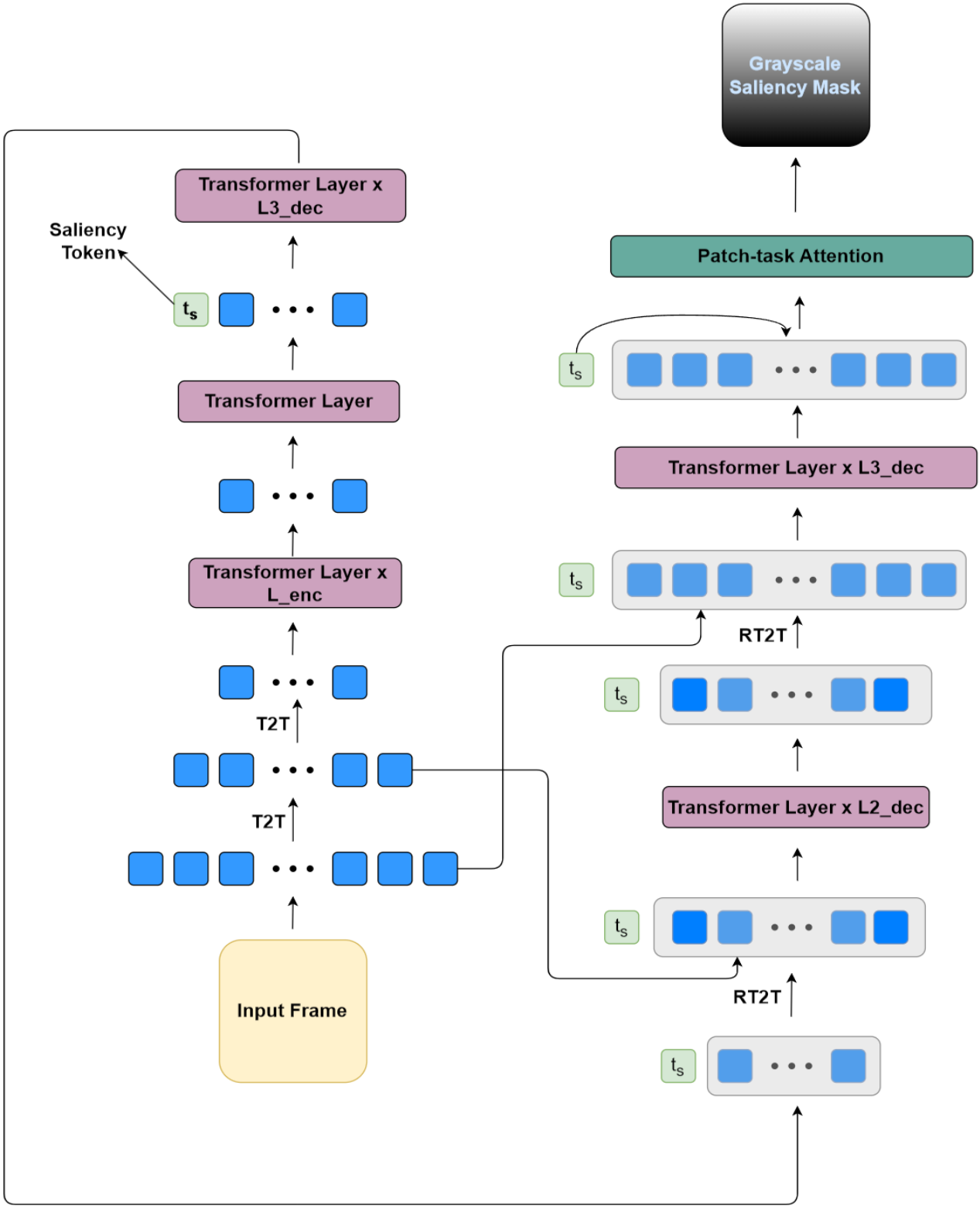

Figure 14: Temporal Visual Saliency Transformer's architecture

Transformer Encoder

Our encoder utilises the pretrained Tokens-to-Token Vision Transformer (T2T-ViT) (Yuan et al., 2021) to extract image features from the video frames. Tokens-to-Token module (Figure 15) is employed in an effort to enable ViTs to capture local structure of neighbouring pixels and to increase feature richness. The main stages of T2T are **Re-Structurization** and **Soft split**

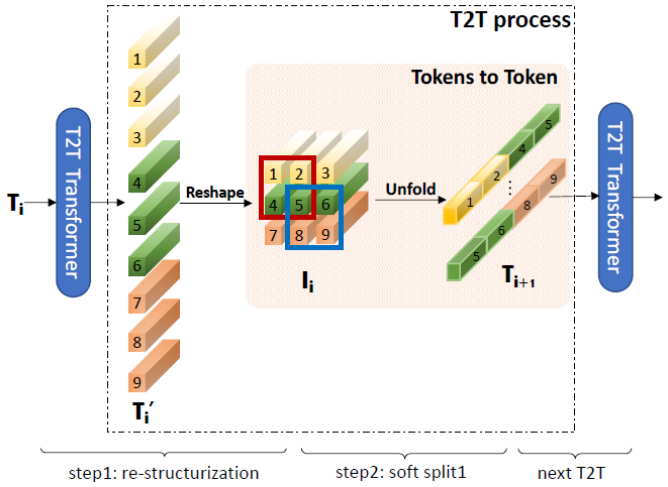

Figure 15: Main stages of Tokens-to-Token patch-conversion process
Image source: (Liu et al., 2021)

In re-structurization, the tokens sequence is transformed by the self-attention block.

$$T' = MLP(MSA(T))$$

Each time, the re-structurization step first transforms previous token embeddings to new embeddings. In the three T2T-modules that we use t, the self-attention block is designed in a manner that produces token sequences of length 196, 784 and 3136, respectively. These lengths have been selected to simplify the Reshape step, in which the token sequences mentioned before get converted to 14 x 14, 28 x 28, 56 x 56 images, respectively. Then, soft split is applied to model local structure information and reduce the length of tokens.

$$T_i' = MLP(MSA(T_i))$$

$$I_i = Reshape(T_i')$$

$$T_{i+1} = SS(I_i), \quad i = \ldots(n-1)$$

Where MLP is multilayer perceptron, MSA is multi-head self-attention and SS is soft split. In SS, the overlapped patch splitting is the one that introduces local correspondence within neighbouring patches. $\mathbf{I_1}$ is split into k x k patches with s overlapping. Image boundaries get padded with p zero-padding. The image patches are unfolded to a sequence of tokens $T_o \in \mathcal{R}^{l_o \times ck^2}$ where the sequence length is computed as:

$$l_o = h_o \times w_o = [\frac{h + 2p - k}{k - s} + 1] \times [\frac{w + 2p - k}{k - s} + 1] \quad (1)$$

Where h and w stand for the height and width of the initial image, respectively and h_o and w_o stand for the height and width of the new image that will occur after the reshape of the tokens.

The T2T transformation can be performed in multiple iterative steps, where in each step, previous token embeddings are restructured to obtain new embeddings. Among the three soft split steps, the patch sizes are set to $k = [7, 3, 3]$, the overlappings are set to $s = [3, 1, 1]$, and the padding sizes are set to $p = [2, 1, 1]$. As such, we can obtain multilevel tokens $T1 \in R^{l_1 \times c}$, $T2 \in R^{l_2 \times c}$, and $T3 \in R^{l_3 \times c}$. Given the width and height of the input image as H and W, respectively, then $l_1 = \frac{H}{4} \times \frac{W}{4}$, $l_2 = \frac{H}{8} \times \frac{W}{8}$, and $l_3 = \frac{H}{16} \times \frac{W}{16}$. We follow (Yuan et al., 2021) to set $c = 64$ and use a linear projection layer on T_3 to transform its embedding dimension from c to $d = 384$. The final token sequence T_3 consists of l_3 c-dimensional tokens, which encode the visual information of the correspondent frame.

Temporal Module

Next, we insert a transformer module between the transformer encoder and decoder, which applies **Divided Space-Time Attention** (Bertasius et al., 2021) (see Figure 16). Given a frame at instant t and one of its patches as a query, Divided Space-Time Attention computes the spatial attention over the whole frame and then the temporal attention in the same patch of the query but also in the rest of the frames that exist in the clip. The final token sequence models both temporal and spatial information for the whole clip of frames.

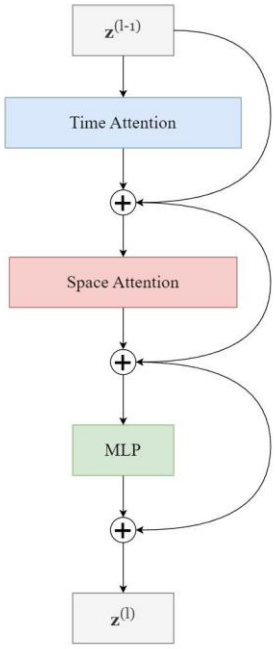

Figure 16: Divided Space-Time Attention

Transformer Decoder

Our decoder aims to decode the patch tokens to saliency maps. Since Saliency Maps have high resolution, we use Reverse Tokens-to-Token (Liu et al., 2021) procedure to upsample the patch tokens and make dense predictions. Furthermore, we fuse low-level tokens from the T2T-ViT encoder with the upsampled tokens to leverage accurate local structural information. A saliency token is added on the patch token sequence and is updated through self-attention in each transformer decoder. Finally, for saliency prediction we perform patch-saliency-attention between the final decoder patch tokens and the saliency token. Then, we apply two linear transformations with the sigmoid activation to scale them in the range [0,1] and reshape them to a 2D saliency map. The decoder's process is formulated as:

$$T^D{}_i = MLP(MSA(Linear([RT2T(T^D{}_{i+1}), T_i]))$$

where the operator $[\cdot, \cdot]$ expresses the concatenation along the token embedding dimension. *Linear* means linear projection to reduce the embedding dimension after the concatenation to c. RT2T refers to reverse Tokens-to-token transformation which upsamples tokens by expanding each token into multiple sub-tokens. Particularly, we decrease the embedding dimension of the input patch tokens from $\mathrm{d} = 384$ to $\mathrm{c} = 64$ by projecting them. Then, we utilize another linear projection to extend the embedding dimension from c to ck^2. In a similar manner to the soft split step in T2T, every token is treated as a $\mathrm{k} \times \mathrm{k}$ image patch, and the neighbouring patches overlap by s. Subsequently, we can fold the tokens as an image with p zero-padding, and the output image size can be determined using Equation (1) inversely. Given the length of the input patch tokens as $h_o \times w_o$, the spatial size of the output image is $h \times w$. Finally, we reshape the image back to the upsampled tokens with size $l_o \times c$, where $l_o = h \times w$ By setting $\mathrm{s} < \mathrm{k} - 1$, the RT2T transformation can expand the token length. Motivated by T2T-ViT, we perform RT2T thrice and set $\mathrm{k} = [3, 3, 7]$, $\mathrm{s} = [1, 1, 3]$, and $\mathrm{p} = [1, 1, 3]$. Consequently, the length of the patch tokens can be gradually increased to $\mathrm{H} \times \mathrm{W}$, which equals the original size of the input image.

Dataset

For the training, evaluation and testing phases we used the DHF1K dataset. DHF1K is the largest and most diverse video saliency dataset including 600, 100, 300 videos for each phase respectively. The videos have 640x360 resolution, a rate of 30 fps and the fixation maps have been collected from 17 observers by an eye-tracker device.

3.3.4 Implementation Details

The input frames have been resized to 256 x 256. For the training phase, a random clip of 8 or 16 frames gets extracted from each video. Then, five different augmentation techniques are used, each one with 0.5 probability to get applied in each video clip. More specifically, the following augmentation procedures have been used:

1. **RandomCrop**: crops the given clip at random location with size 224 x 224.
2. **ColorJitter:** randomly changes the brightness, contrast, saturation and hue of an image.
3. **DropFrame**: randomly selects and rejects one frame from the clip.
4. **FrameRate**: applies different frame rates in the frame selection process.
5. **Inversion**: inverses the temporal sequence of the frames.

As a loss function, we used a combination of saliency metrics, which are common and effective in static and dynamic saliency prediction models. We take the weighted summation of Kullback-Leibler (KL), Linear Correlation Coefficient (CC) and Similarity (SIM) to represent the loss function and our ablation study proved that the weighted summation of these three losses achieves better results than using one of them. The final loss function can be expressed as:

$$L(S, G) = L_{KL}(S, G) + a_1 L_{cc}(S, G) + a_2 L_{SIM}(S, G)$$

where $\mathrm{S} \in [0,1]$ is the predicted flattened saliency map with length i and $\mathrm{G} \in [0,1]$ the ground truth flattened saliency map with length i. We set $a_1 = 0.5$ and $a_2 = 0.5$, to scale the amplitudes of each loss. The calculation formulas for the three loss can be represented as:

$$L_{KL}(S,G) = \sum_i G_i ln \frac{G_i}{S_i}$$

$$L_{CC}(S,G) = -\frac{cov(S,G)}{\rho(S)\rho(G)}$$

$$L_{sim}(S,G) = -\sum_i min(G_i, S_i)$$

where cov computes the covariance and ρ computes the standard deviation.

3.3.5 Results

The test set of DHF1K is not released, and only the authors of the dataset can confirm the testing scores after submission. The metrics that are used on DHF1K and are the most common for saliency prediction are Normalised Scanpath Saliency (NSS), Pearson's Correlation Coefficient (CC), Similarity (SIM) and variants of Area Under ROC Curve (AUC-Judd and shuffled AUC). Our method's evaluation metrics can be seen on Table 12 and through comparison on the benchmark's website[18] can be seen that it is equally compared with all the state-of-the-art methods.

Table 12: Our model's score on DHF1K benchmark compared with state-of-the-art CNN architectures

Models	NSS ↑	CC ↑	SIM ↑	AUC-J ↑	sAUC ↑
SalEMA	2.574	0.449	**0.446**	0.890	0.667
Tased-NET	2.667	0.470	0.361	0.895	0.710
UNISAL	**2.776**	0.490	0.390	0.901	0.610
MV-SD	2.735	**0.4936**	0.3731	**0.904**	**0.716**

On Figure 17 some qualitative results of our model are exemplified. Frame line shows the original frame of the clip, GT line shows the original frame overlapped with the ground truth fixation map and SP line shows the original frame overlapped with the predicted saliency mask. The results of the model are satisfactory, with both the ground truth fixation map and the predicted saliency mask being very close to each other. This indicates that the model is performing well in accurately predicting the regions of an image that are most likely to draw a viewer's attention. Overall, these results suggest that the saliency prediction model is a promising tool for applications such as image and video processing, where identifying important visual information is critical.

[18] https://mmcheng.net/videosal/

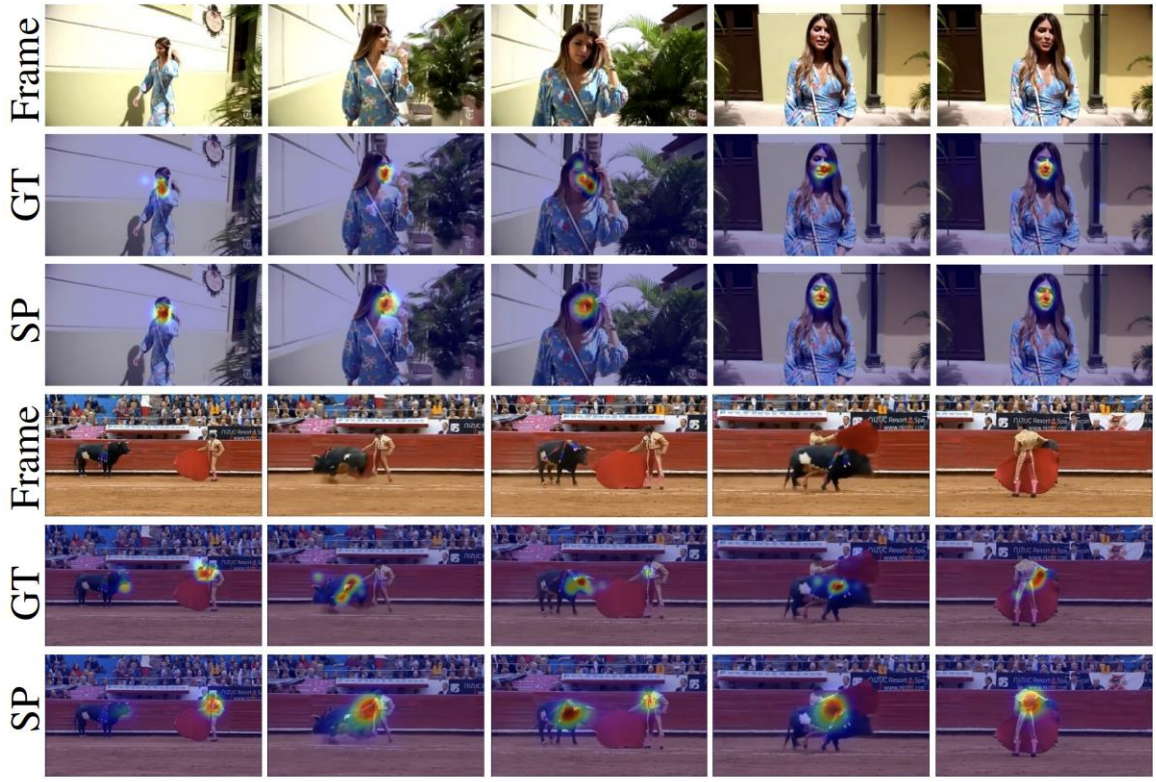

Figure 17: Qualitative results of our approach

3.4 Face Blurring

In this section, we present the modifications we implemented to enhance the time performance of the face blurring module.

3.4.1 Current State

Previously, to tackle face blurring, the SCRFD (Guo et al., 2021) model was deployed to detect faces in 360 videos and then these faces were blurred with GaussianBlur. This model was proposed by Deng et al. (2009) and uses sample and computation redistribution to achieve efficient face detection. During training data augmentation, square patches get cropped from the original images using a random size from the range [0.3, 1.0] of the short edge of the original images. The model re-distributes positive training samples across different scales of feature maps to cope with the smaller testing scale and the computation redistribution across different components is explored. This is done to maintain a predefined computation budget and to find the relationship between computation distribution and performance from populations of models. The above implementation has been integrated into MediaVerse, however due to high resource requirements we did not proceed to make it available for the users.

3.4.2 Modifications

In order to reduce the resource requirements of our implementation, we implemented the following three modifications:

1. Decreased the value of the variable that determines the size of the input image that is fed into the model during inference and affects the size of the detected faces. This modification decreases the computational cost of the model. Furthermore, this decrease results in missing some faces which were already small, however this is not common, and the undetected faces already lack detailed information, so the facial characteristics do not reveal the subject's identity.
2. Replaced GaussianBlur method with BoxBlur method. Both of them belong to PIL Library and particularly to ImageFilter module. BoxBlur blurs the image by setting each pixel to the average value of the pixels in a square box extending radius pixels in each direction. We selected BoxBlur because in contrast with GaussianBlur, it uses an optimised implementation, which runs in linear time relative to the size of the image for any radius value and its results are equally acceptable.
3. Instead of checking every single frame to detect faces, we implemented a different approach inspired from binary search algorithm. Our algorithm checks for face detection every ***n*** frames. A frame can belong in two categories: face detected/ no face. If our algorithm finds that two consecutive checked frames belong to the same category, then it is assumed that all frames in between those two belong to the same category. If the two checked frames belong to different categories then we examine the intermediate frame of those two. Then, the algorithm gets called iteratively for the two new pairs of frames (start, mid), (mid, end).

In Table 13, we observe a decrease of time requirements of our model around 75%, after the application of the new modifications. These modifications enhance the user experience by improving the responsiveness of the module. It is noteworthy that the magnitude of time reduction varies depending on how often faces appear in frames. This happens due to the binary-search-inspired algorithm, since when faces appear rarely, the algorithm will not check every single frame for face detection. Moreover, longer duration, higher resolution and bigger fps rates of videos lead to higher time requirements, however they do not affect reduction percentage.

Table 13: Time Requirements of our model pre- and post-modifications

Video Characteristics	Without Modifications (sec.)	With Modifications (sec.)	Reduction Percentage (%)
Resolution: (480,360), FPS: 30, Duration: 10 sec	33.5	7.7	77.01
Resolution: (1280, 720), FPS: 25, Duration: 10 sec	42.3	11.1	73.75
Resolution: (1280,720), FPS: 30, Duration: 24 sec	117.7	29.6	73.5
Resolution: (1296, 760), FPS: 24, Duration: 40 sec	189.1	44.3	76.57
Resolution: (1920, 960), FPS: 30, Duration: 27 sec	262.0	54.8	79.08

3.5 Deployment

The Media Annotation Service, shown in Figure 18, exposes a gRPC API described in the Table B.1 of Annex II. For flexibility, it consists of both unidirectional (from server to client) and unary endpoints, while the bidirectional endpoints, discussed in D3.2, have been removed to simplify the connection management logic of the client. The unidirectional endpoints accept an image, video or 3D annotation request and stream the responses of the relevant annotation models as they become available. The unary endpoints, on the other hand, send the response in one message after all annotation models have run. In the annotation request, in addition to specifying the asset by including a downloadable link, the client can now also directly send the asset bytes.

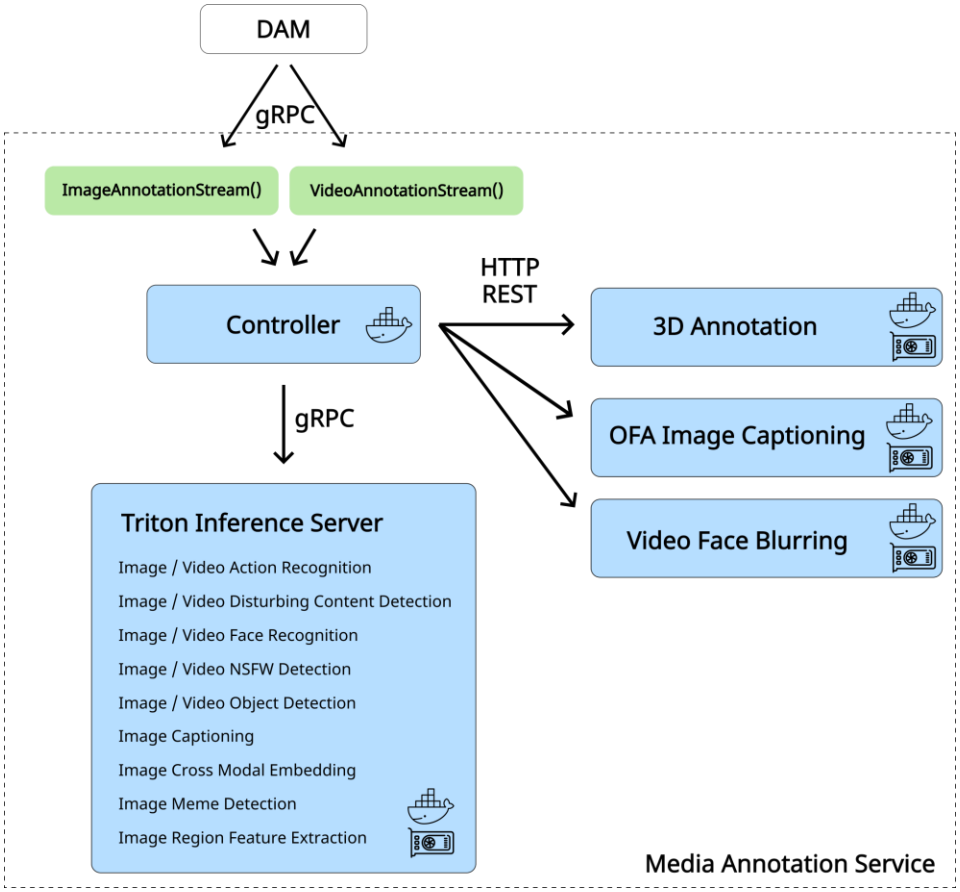

Figure 18: The Media Annotation Service architecture

We have also expanded our annotation models including NSFW detection for both images and videos, a new image captioning model based on the OFA[19] checkpoint, a model for extracting regional image features to facilitate region based retrieval and a video model for face blurring. The NSFW and regional feature extraction models have been exported as ONNX and integrated in the Triton Inference Server. However, the OFA based image captioning and the video face blurring models have been implemented as separate microservices due to their complexity and incompatibility with the Triton's inference framework. These microservices expose an HTTP REST API, which the controller queries to get the respective annotations. The new regional feature extraction model uses an object detector to obtain region proposals and for each region extracts a feature vector.

[19] https://github.com/OFA-Sys/OFA

4 MediaVerse Retrieval and Recommendation Systems

In this section, we further extend what we discussed in Section 3 of the Deliverable 3.2 - Content Discovery and Recommendation, Annotation and Adaptation Framework[20], providing new progress and implementations. As regards the Retrieval Technology, we provide further experiments on new datasets as well as ALBEF (Li & Selvaraju, 2021), an alternative architecture to CLIP (Radford & Kim, 2021), capable of acting in a large-scale context for asset retrieval. It obtains state-of-the-art results on MS-COCO (Lin & Maire, 2014) and FLICKR30K (Young & Lai, 2014), as well as lower multimodal encoding and search times. Concerning Recommender System (RS), we make recommendations based on the semantics of previous posts of the user that makes the request for recommendation. Here we provide further quantitative analysis on the RS and we present two dataset augmentations with which we perform the testing. Finally, we provide a projection of retrieval and recommendation systems' implementation in the MediaVerse platform through use examples.

4.1 MediaVerse Cross-Modal Retrieval System

Cross-modal retrieval is the task of searching data using different data modalities (e.g., image-text). It aims to enable flexible retrieval experience across different modalities (e.g., texts vs. images). The core of cross modal retrieval research is to learn a common subspace where the items of different modalities could be directly compared to each other. Specifically, we expect an arrangement of vectors in the common space based mainly on asset semantics. In an Information retrieval (IR) context, given an input text query, a retrieved dissimilar text being more similar than similar image results in poor feature quality and an unwanted prioritisation of assets of the same query modality.

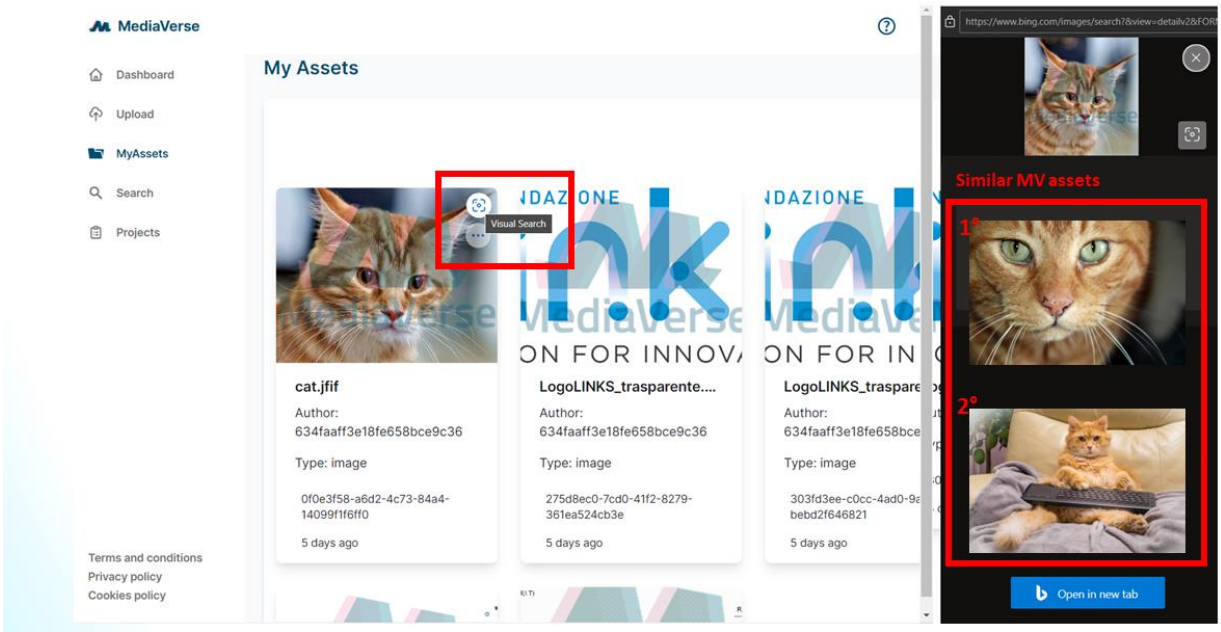

Figure 19: Example of Mediaverse "Visual Search" experience inside "MyAsset" section

In this section, the focus is on content retrieval, also called "search". We show two examples of the retrieval module implementation inside the MediaVerse platform. Within the "MyAssets" section, as shown in Figure 19,

[20] https://mediaverse-project.eu/wp-content/uploads/2022/07/D3.2-V1.0.pdf

the user can exploit the "Visual Search" feature, associated with an asset, to receive suggested similar MV assets. The most similar assets in the MV platform are ranked starting from the most similar. Our cross-modal retrieval technology is capable of directly processing the MV assets without relying on auxiliary tags.

Within the "Search" section, as shown in Figure 20, the user can search for other assets in the MV Platform. MediaVerse requires the user to insert a query (a text) as input. The user aim is to find the MediaVerse assets most similar to its input query regardless of it being a text or an image. The system shows to the user the most similar assets already present in the MediaVerse Platform. Similarly, in the "MyAsset" section, all assets have the "Visual Search" feature, as shown in Figure 20. The semantically similar assets are ranked starting from the most similar with respect to the selected one.

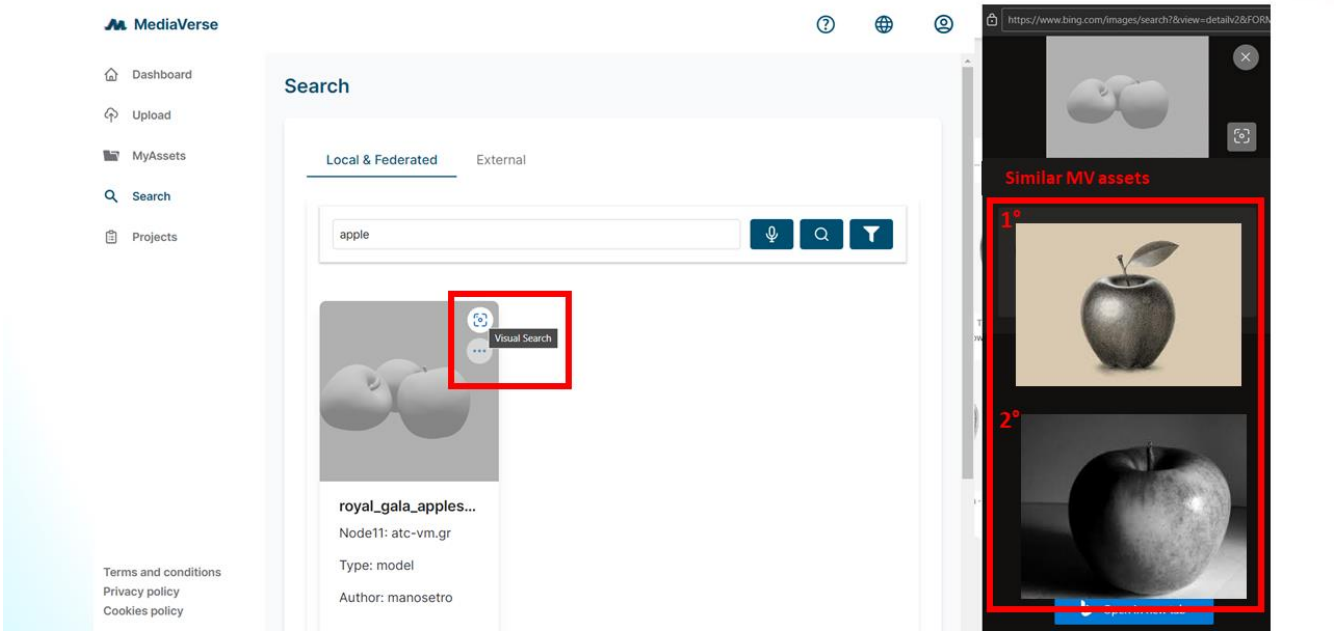

Figure 20: Example of MediaVerse "Visual Search" experience inside the "Search" section

The "Visual Search" feature does not exploit any Retrieval technology yet. Semantic asset alignment is possible thanks to pre-trained multimodal architectures that generate close vector representations when MV assets are similar. In Deliverable 3.2 - Content Discovery and Recommendation, Annotation and Adaptation Framework, we referred to CLIP and we described its architecture and the results obtained. In the following subsections, we will do the same with ALBEF, a possible CLIP alternative, as well as to provide additional CLIP results beyond those already provided in D3.2.

4.1.1 ALign BEfore Fuse (ALBEF)

In this section, we show ALign BEfore Fuse (**ALBEF**), Figure 21. Similarly to CLIP, it first encodes the image and text independently with a detector-free image encoder and a text encoder. Then a multimodal encoder fuses the image features with the text features through cross-modal attention. An intermediate image-text contrastive (ITC) loss is applied on representations from the unimodal encoders, which serves three purposes:

1. it aligns the image features and the text features, making it easier for the multimodal encoder to perform cross-modal learning;
2. it improves the unimodal encoders to better understand the semantic meaning of images and texts;

3. it learns a common low-dimensional space to embed images and texts, which enables the image-text matching objective to find more informative samples through our contrastive hard negative mining.

To improve learning under noisy supervision the authors propose Momentum Distillation (MoD), a simple method, which enables the model to leverage a larger uncurated web dataset. During training, a momentum version of the model is kept by taking the moving-average of its parameters and using the momentum model to generate pseudo-targets as additional supervision. With MoD, the model is not penalised for producing other reasonable outputs that are different from the web annotation. MoD is proven not only to improve pre-training, but also downstream tasks with clean annotations.

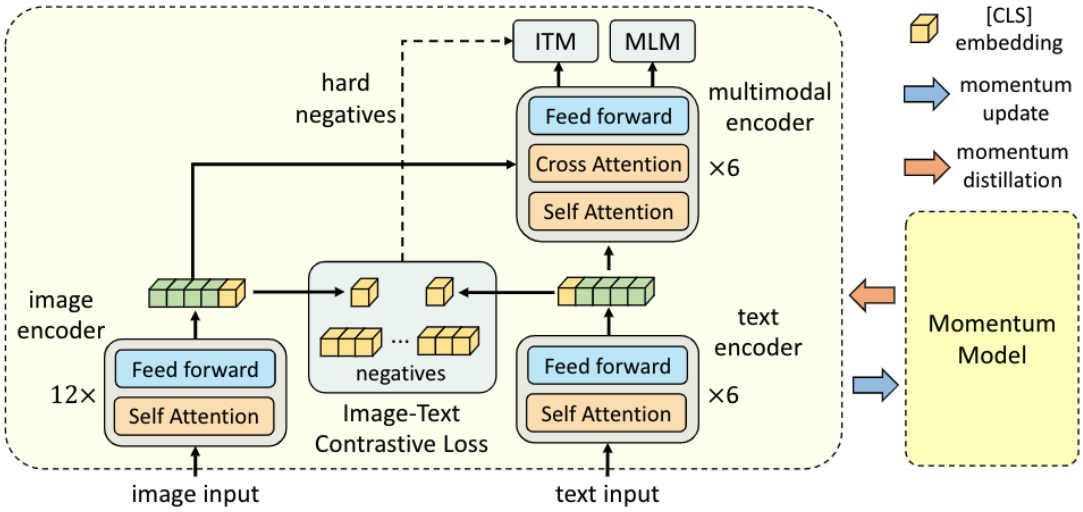

Figure 21: Illustration of ALBEF. Image source: (Li & R. Selvaraju, 2021).

Vision Encoder: 12-layer visual transformer ViT-B/16 (Dosovitskiy & Beyer, 2021) (85.8M parameters).

Text Encoder: 6-layer transformer initialized using the first 6 layers of the BERTbase (Devlin & Chang, 2019) (123.7M parameters).

Multimodal encoder: 6-layer transformer initialized using the last 6 layers of the BERTbase.

The final text and image representations are 256-d vectors.

Table 14 shows the hyperparameters adopted in (Li & R. Selvaraju, 2021).

Table 14: ALBEF general hyperparameters

HYPERPARAMETER	VALUE
Batch size	512
Training epochs	30
Weight decay	0.02
Warm-up iterations	1000

Training data: two web datasets (Conceptual Captions (Sharma & Ding, 2018), SBU Captions (Ordonez & Kulkarni, 2011)) and two in-domain datasets (COCO (Lin & Maire, 2014) and Visual Genome (Krishna & Zhu, 2017)). The total number of unique images is 4.0M, and the number of image-text pairs is 5.1M. To show that the method is scalable with larger-scale web data, a noisier Conceptual 12M dataset (Changpinyo & Sharma, 2021) is

introduced, increasing the total number of images to 14.1M. Table 15 shows the number of images and texts over the datasets used to train ALBEF.

Table 15: Statistics of the pre-training datasets

	COCO (KARPATHY-TRAIN)	VG	CC	SBU	CC12M
# image	113K	100K	2.95M	860K	10.06M
# text	567K	769K	2.95M	860K	10.06M

4.1.2 Experimental Setup

Retrieval quality: The pre-trained CLIP and ALBEF models are available online for inference and fine-tuning purposes. In addition to CLIP, we downloaded ALBEF trained on 14M images and tested it on MS-COCO 5K test split (Karpathy) for both text and image crossmodal retrieval. For the robustness, we tested the two models also on FLICKR30K 5k test split. For both datasets, the set of retrievable assets contains 5000 assets.

Search run time: Since ALBEF vectorizes assets to 256d embeddings, it is useful to evaluate what benefits there are in terms of time required to carry out the search by means of FAISS indexer. The setup is the same as for 512 dimension vectors: 5000 retrieved assets over a space of 5000 retrievable ones.

Encoding time: The encoding time has been computed for both CLIP and ALBEF in terms of time required to vectorize texts and images. Both computations take into consideration the pre-processing phase necessary to generate the input of language and vision transformers.

4.1.3 Results

Tables 16 and Table 17 show results in terms of Recall (R@1, R@5, R@10) for MS-COCO (also employed in D3.2) and FLICKR30K (new evaluation dataset introduced in this deliverable) respectively. The information included in the first two rows of Table 16 is also presented in D3.2. On both MS-COCO and FLICKR30K, ALBEF outperforms CLIP. Table 19 (first three rows also presented in D3.2) shows that the encoding times of ALBEF are also more advantageous. In line with this, Table 18 shows that ALBEF, which maps assets to 256-d instead of 512-d vectors, is associated with shorter search times (by about 0.2ms) with respect to the CLIP ones. In a large-scale scenario it can lead, jointly with shorter encoding time, to considerable gains in terms of retrieval time.

Table 16: Recalls of img2txt and txt2img tasks (MS-COCO val 5k)

	TEXT RETRIEVAL			IMAGE RETRIEVAL		
encoder	R@1	R@5	R@10	R@1	R@5	R@10
VGG19 BiGRU (3 layers, 128 units, 128 embedding dim)	0.094	0.283	0.406	0.086	0.260	0.379
CLIP (ViT-B/32)	0.322	0.623	0.747	0.237	0.518	0.656
ALBEF (14M)	**0.505**	**0.782**	**0.850**	**0.440**	**0.741**	**0.846**

Table 17: Recalls of img2txt and txt2img tasks (FLICKR30K val 5k)

	TEXT RETRIEVAL			IMAGE RETRIEVAL		
CLIP (ViT-B/32)	0.433	0.681	0.774	0.411	0.663	0.753
ALBEF (14M)	**0.452**	**0.714**	**0.801**	**0.430**	**0.692**	**0.780**

Table 18: Search methods run time

	N. CONTENTS	EMBEDDINGS SIZE	K	RUN TIME (MS)
vanilla cosine	5000	512	5000	129.31
filter + cosine	5000	512	5000	105.84
faiss (CLIP)	5000	512	5000	0.72
faiss (ALBEF)	5000	**256**	5000	**0.56**

Table 19: Encoding time

	TEXT ENCODING TIME (MS)	IMAGE ENCODING TIME (MS)
CLIP	9.10	17.43
ALBEF	**8.15**	**15.71**

4.2 MediaVerse Recommendation System

In this section, we propose a formal description and a further quantitative analysis of the content-based recommender system proposed in Section 4 of Deliverable 3.2 - Content Discovery and Recommendation, Annotation and Adaptation Framework[21].

In general, we can classify recommendation systems in different categories based on the approach and on the different sources of information for providing users with recommendations of items. The algorithm proposed in D3.2 belongs to the content-based class of recommender systems. Content-based recommender systems suggest items to users based on their preferences and the characteristics of the items themselves. Content-based systems make recommendations by analysing the attributes of the items a user has shown interest in and suggesting similar items with matching attributes (Chen et al., 2021).

First, we propose a use case example of recommender system, performed with the *skimage* python library, and an example of how the recommendation feature can be integrated in the MediaVerse Dashboard. Imagine a scenario where user *u* requests for recommended content belonging to both text and image modality. For the purpose of this example, *u* posted the images and texts shown in Figure 22 in the form of MV posts. In this example, images and texts represent separate posts that the user might have uploaded.

[21] https://mediaverse-project.eu/wp-content/uploads/2022/07/D3.2-V1.0.pdf

Figure 22: Example of possible posts in image and text modality uploaded by the user

The MV recommender system takes as input user *u*’s previous posts and searches for similar assets in MV space, ranking the output assets of any modality, meaning that we can have images or texts in output.

Figure 23 presents a hypothetical scenario where the MV space is populated by a pool of images and texts. The recommended assets, thus the recommendation output, extracted from other user’s posts are highlighted in red squares. As we can observe the recommended images and texts are semantically similar to the previous content of the user that made the recommendation request.

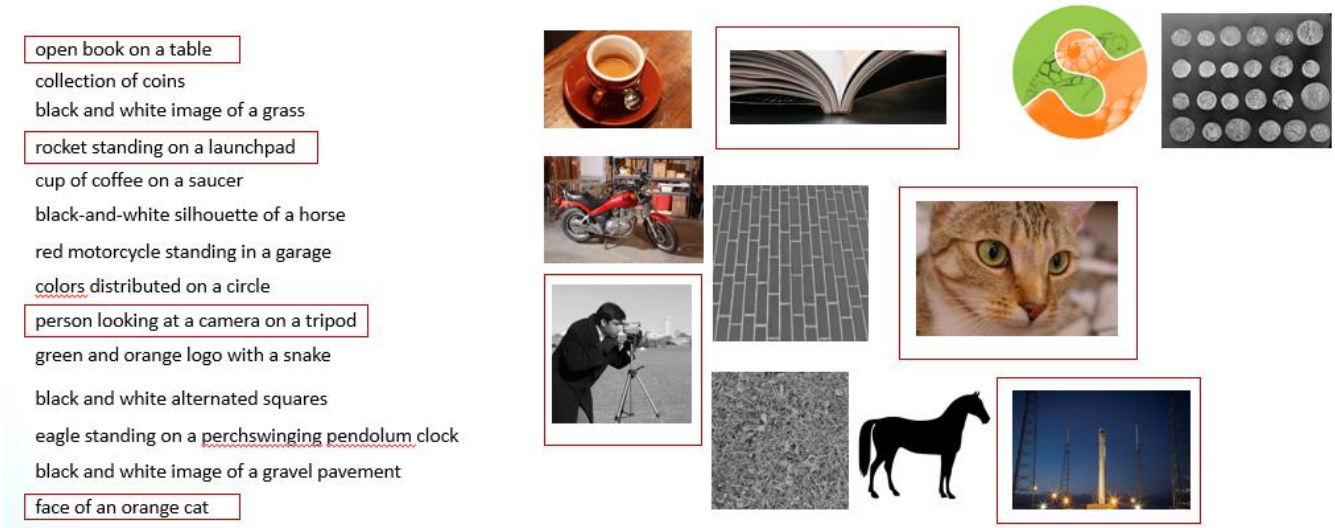

Figure 23: Example of the pool of posts available in MV space and of what the RS would retrieve

In Figure 24, there is a possible integration of the recommender system in the MediaVerse Dashboard. We can see that a recommendation request can be activated after a click on the “Discover” tab in the “Search” page of the Dashboard. At the moment the “Discover” tab is not implemented yet.

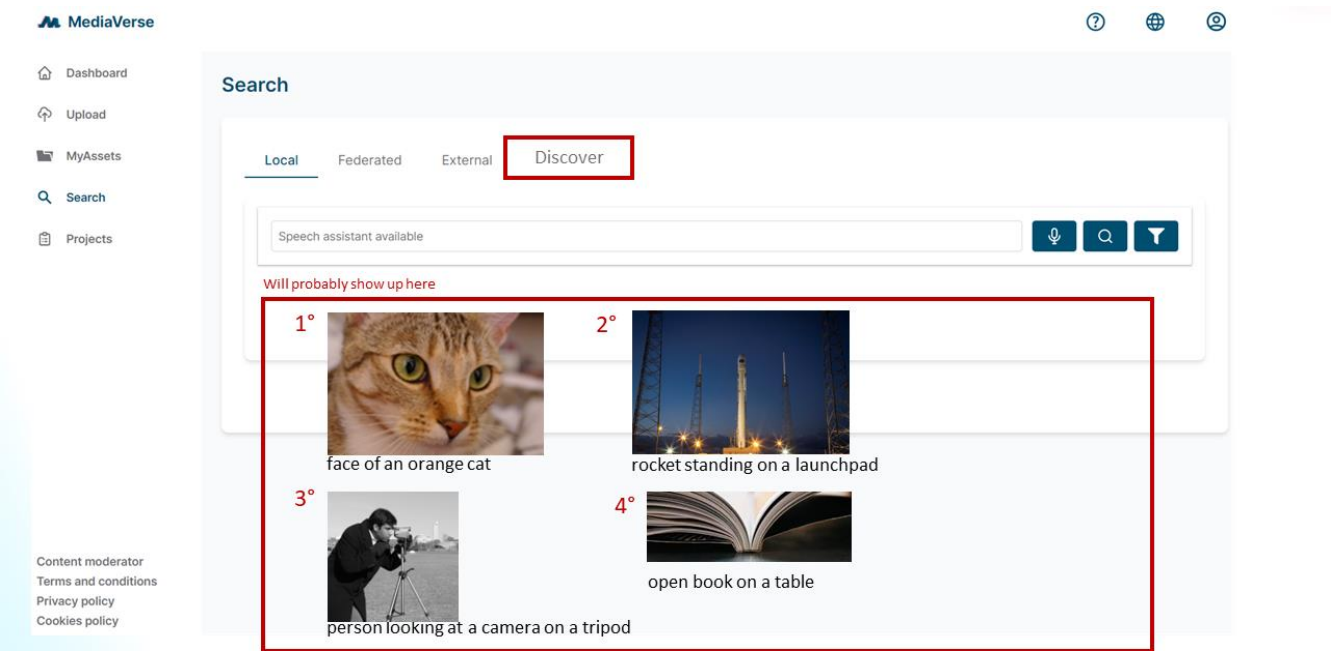

Figure 24: Possible integration of the RS output in the MV Dashboard

4.2.1 Task Description

In this section, we discuss the recommendation task with a formal description of value proposition and method. As stated in the previous section, the value proposition of the recommendation task is to present the user with a new set of images and texts that the user has never seen before, using as input of the recommendation request the posts that the user has previously shared.

We define:

- u : user that makes the recommendation request
- $C^T = \{c_1{}^T, \ldots, c_j{}^T\}$: set of previous posts of user u in text modality
- $C^I = \{c_1{}^I, \ldots, c_j{}^I\}$: set of previous posts of user u in image modality
- $q(I,T)$: recommendation request (query) of user u
- $R(I,T) = \{r_k{}^{I,T}, \ldots, r_k{}^{I,T}\}$: ranked set of recommended assets in both text and image modality for user u

The objective is to recommend, for each user, a ranked list of images and texts $R(I,T)$, based on the set C^T and C^I of content posted by user *u.*

Figure 25 presents the method to obtain the recommendations and it follows the following steps:

1. User u makes a query $q(I,T)$, i.e., a request to discover new content.
2. The RS accesses the content stored when user u first uploaded the previous posts C^T and C^I and retrieves their embeddings (for the moment performed through CLIP, but there is the possibility to switch to ALBEF), separating the process for images and texts.
3. Then it performs a clustering through HDBSCAN of the embeddings and creates the so-called “areas of interests”.

4. From the clusters it extracts the medoids of each cluster as representative samples of their "area of interest".
5. Then uses each medoid as input to the retrieval system to discover similar assets in the FAISS index. It performs this operation in all four possible modality pairs (image to image, image to texts, text to image and text to text).
6. Finally, it ranks all assets according to similarity score in a unified output $R(I,T)$.

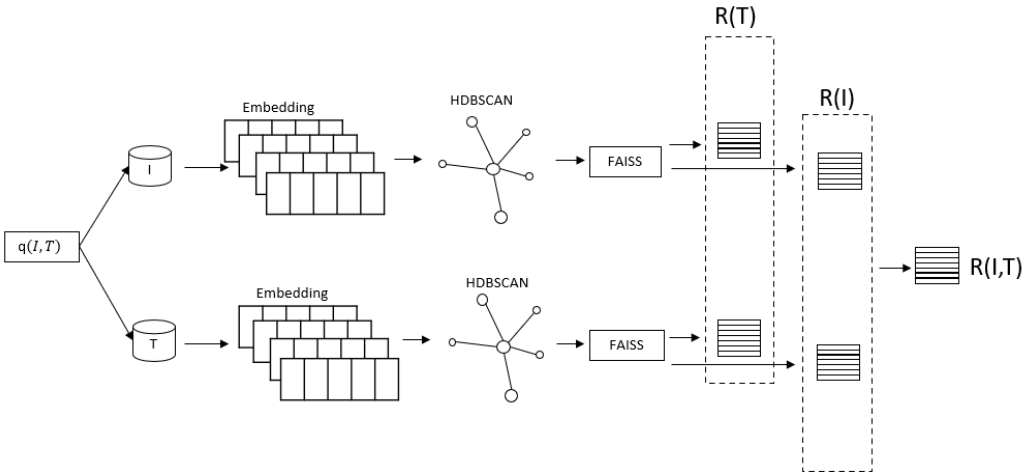

Figure 25: Functional diagram of the Recommender System

4.2.2 Experimental Setup

Differently from the retrieval system, that requires one asset as input query, in the recommendation system we need as input query multiple assets, to simulate the previous posts of user *u*. Therefore, for the testing and evaluation of the performance of the recommender system we implemented two experimental setups creating two custom datasets for the occasion:

- **Augmented MS-COCO dataset**: the reason behind the augmentation is that MS-COCO only has a maximum of five captions associated with each image and one image associated with each caption. Therefore, we could not evaluate the recommender system; since we need to input several images and texts from the same "area of interest", in order to test the clustering. Thus, we need a dataset with multiple images and texts belonging to the same class to establish a *ground truth*: the evaluation of the relevance of the output depends on whether the asset in output belongs to the same class as the input. The augmentation was performed with the following tools:
    - We used a paraphraser to increase the number of captions while preserving the overall semantics. We randomly sampled two of the 10 paraphrases applied by tuner007/pegasus_paraphrase · Hugging Face[22] to each of the five captions. In this way, we obtained 20 captions associated with each image.
    - We used the Web Search API | Microsoft Bing[23] provided by Microsoft to increase the number of images. We made a query for each of the 5 MS-COCO original captions and we retrieved the

[22] https://huggingface.co/tuner007/pegasus_paraphrase

[23] https://www.microsoft.com/en-us/bing/apis/bing-web-search-api

first four image results from Bing Web Search. In this way, we obtained 20 images associated with the 20 captions already generated from the paraphraser.

- **CIFAR-100 captioned dataset**: CIFAR-100 train dataset is composed of 100 classes with 500 images per class. In order to obtain both images and texts belonging to the same class, we perform a captioning of the images with the pretrained network from bipin/image-caption-generator · Hugging Face[24]. Thus, we obtain 500 images and 500 texts belonging to the same CIFAR-100 class, since we assume that a caption generated from an image belongs to the same class as the image.

In order to evaluate the Mean Average Precision (MAP) of the RS we perform multiple queries. One query $q(I,T)$ is one recommendation request, giving as input (i.e., as user's previous posts) N_c classes with N_i images or texts per class. For each class, there are k relevant assets to retrieve in a pool of P items in MediaVerse space. The Ground Truth is that a recommended asset is considered relevant if it belongs to one of the N_c classes in input.

4.2.3 Results

The first testing was performed with MS-COCO augmented dataset. We evaluate MAP@10 for each of the four modality comparing CLIP and ALBEF encoder. This experiment is performed for each modality with the following setup: we upload as user previous posts 10 images and 10 texts belonging to the same "area of interest" (i.e., images and texts generated from the same MS-COCO original sample). Therefore, we set *Nc* = 1 and *Ni* = 10. Then we upload 1000 random images and texts to simulate the pool of MV posts from which we want to retrieve the 10 relevant items generated from the same MS-COCO original sample, so we set *P* = 1000 and *k* = 10. Table 20 shows the mean of the average precision over 10 different queries, for both ALBEF and CLIP encoder.

Table 20: MAP@10 with MS-COCO augmented

	CLIP	ALBEF
Img2Img	0.8836	**0.9739**
Img2Txt	**0.7715**	0.7394
Txt2Img	**0.9033**	0.8421
Txt2Txt	**0.9549**	0.9091

The second testing was performed with captioned CIFAR-100. This experiment is performed for each modality with the following setup: we upload as user previous posts 100 images and 100 texts belonging to three "areas of interest" (i.e., images and texts belonging to the same CIFAR-100 class). With this setup, we want to test how the RS performs when the user has uploaded semantically various content. Therefore, we set *Nc* = 3 and *Ni* = 100. We upload a pool of 1000 random images (*P* = 1000) and texts as MV discoverable items, from which we want to retrieve the *k* = 10 relevant items the 3 classes in input (selected at random from the 3 classes). Table 21 shows the mean of the average precision over 10 different queries, for both ALBEF and CLIP encoder.

[24] https://huggingface.co/bipin/image-caption-generator

Table 21: MAP@10 with captioned CIFAR-100

	CLIP	ALBEF
Img2Img	0.9084	**0.9635**
Img2Txt	**0.7925**	0.6753
Txt2Img	**0.8951**	0.7815
Txt2Txt	0.9066	**0.9616**

By comparing Table 20 and Table 21 we can observe how ALBEF tends to perform better in uni-modality, while in cross-modality CLIP performs better.

4.3 Deployment

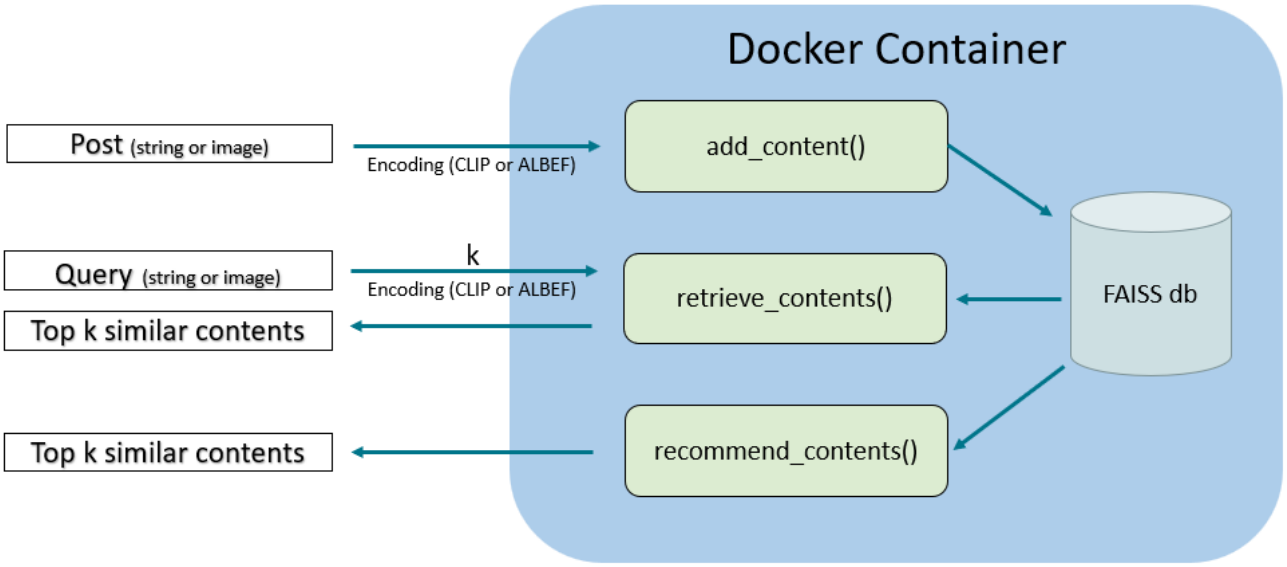

Figure 26: Rest API - Retrieval and Recommendation systems

The user can access the system by means of three functions as depicted in Figure 26, and described with more details in Table 22.

Duplicate content: The application is designed to check for duplicated contents even if they are associated with different MediaVerse IDs. So, if the user tries to add an image/text already present in the collection but with a different ID, the application will append the new ID to the tail of the list of IDs associated to a specific Faiss index of the user. Hence, the application is designed to not store multiple times the same content inside the Faiss index (for the sake of memory footprint), but it keeps track of different IDs associated with that same content. When the system is asked to retrieve the most similar contents with respect to an input query, the IDs associated with the retrieved contents are the most recent (last element of the IDs list).

Table 22: MediaVerse Retrieval and Recommendation system functions

FUNCTION	INPUT	OUTPUT
add_content(): *In this way the user can add content (text or image) to the system. It will be encoded as a 512 (or 256 in the case of ALBEF) embedding vector and stored into the Faiss index.*	- **username**: unique identifier of the user who posts the content - **text or image binary data**: refer to usage.py example - **id**: MediaVerse ID of the content to load - **type**: "text" or "image", string describing the data type of the content to be loaded	- a success message in the field "msg" - the elapsed time for the operation in the field "time"
retrieve_contents(): *The user enters a query (text or image) with the aim of retrieving the K most similar elements among those stored inside the Faiss index. These top K contents are ranked according to the cosine similarity with respect to the input query.*	- **username**: unique identifier of the user who search for a content - **text or image binary data**: refer to usage.py example - **k**: number of similar content to retrieve - **type**: "text" or "image", string describing the data type of the input query	Two fields ("text", "image") each having as subfields the following: - **contents**: ordered list containing the ids (string) of the retrieved texts/images. The list is ordered based on decreasing values of similarity scores (i.e., the first content is the most similar to the query). - **scores**: ordered list of similarity scores for the retrieved contents (i.e., the first score represents the similarity between the query and the first content in the "contents" subfield).
recommend_contents(): *The user does not enter any query. Starting from a seed, the System suggests to the user new contents based on user post history and a certain degree of novelty. The top K contents are ranked according to the cosine similarity with respect to the generated seed.*	- **username**: unique identifier of the user who searches for a content (its contents are excluded from the recommendation process) - **k**: number of similar assets to recommend	- **text** (if the user seed is built starting from text contents) with subfields "text2text" and "text2image" - **image** with subfields "image2text" and "image2image" Each subfield contains two subsubfields: - **assets**: ordered list containing the ids (string) of the retrieved images. The list is ordered based on decreasing values of similarity scores (i.e., the first asset is the best one retrieved for that query). - **scores**: ordered list of similarity scores for the retrieved texts

The CMRR (cross-modal retrieval and recommendation) project is available on GitHub and contains the following elements (see Figure 27):

- app.py: Flask Rest Api exposing three services: add_content(), retrieve(), recommend(). Remember that all three services reside in the same rest api instance, hence in the same docker container.
- docker: folder containing:
    - Dockerfile: file to create the docker image. It creates a conda virtual environment where dependencies are installed together with OpenAI CLIP github to load the model instance and run the encoding of images and texts.

    - docker-compose.yml: which build the docker image and run a container all at once by means of the “docker-compose up” command.
    - .env: file containing:
        - the PORT variable at which the service is exposed. This value can be edited to choose switch service port. Its default value is 8007.
        - the MODEL variable takes value ‘clip’ or ‘albef’ depending on which model is used to encoding MV assets.
- usage: folder containing two examples of retrieval and recommendation services usage.
- albef: folder containing utility scripts to load and run the ALBEF deep learning architecture.
- requirements.txt: list of application dependencies which is read by the Dockerfile.
- .png: images showed in the README.md
- README.md: description of exposed services: expected input and output.
- scripts: folder containing shell scripts to run the rest api.
- log: folder with a log file listing activity of the last session.
- Download and install Docker Desktop available at: https://www.docker.com/products/docker-desktop/.
- Run “docker-compose up –build” to start the REST application.

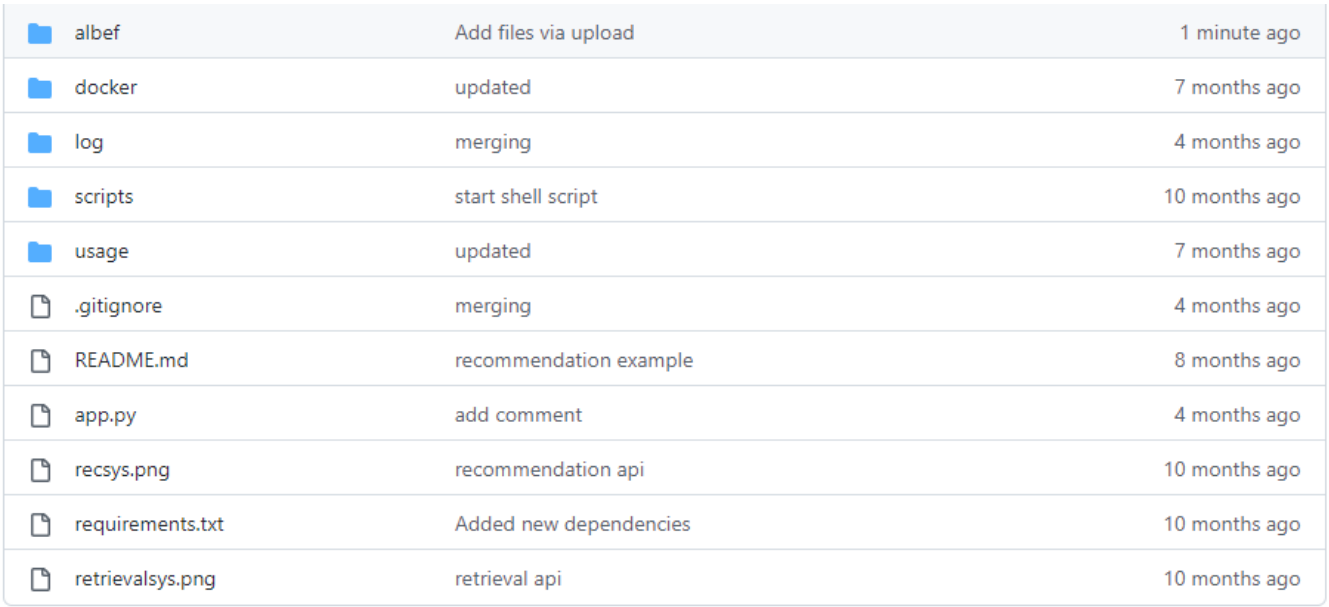

Figure 27: Project folder

5 Conclusions

This deliverable reported the work conducted in the framework of WP3 pertinent to the Media Assets Annotation and Management (MAAM) development, the Media Annotation Service new model additions and updates on previous models as well as the MediaVerse Retrieval & Recommendation Systems updates.

The MAAM application is developed to provide edge-technology-based functionalities pertinent to: (i) automatic media annotation with object labels, free text description, moderation flags (NSFW, disturbing content), action recognition, celebrity recognition and meme/non-meme labels; (ii) visual concept similarity and retrieval enabling the users to create their own models for retrieving relevant content based on visual features that fit to their needs and purposes; as well as (iii) near-duplicate detection for images and videos.

The Media Annotation Service has been updated by the consideration of an effective model compression functionality mainly applied on heavy annotation models to decrease their inference time and the total storage requirements of the system. In addition, a state-of-the-art multi-modal hate speech recognizer has been developed for detecting hateful memes with a potential moderation flag raising future usage. A saliency detection model has been developed in order to provide the users with insights into the areas of their assets that attract visual attention. The deployed model is capable of achieving state of the art results. Finally, several updates on the previously deployed face blurring model have taken place in the direction towards efficiency and effectiveness in order to optimise its performance. These updates have led to a significant reduction in processing time, resulting in a four-fold increase in efficiency. As a result, the model's deployment is more user-friendly, as it can deliver outputs more quickly and with greater reliability.

Further experiments have proved the robustness of CLIP as a multimodal vectorizer module of our Retrieval system. ALBEF proved to be a more advantageous model in terms of both the retrieval quality and search and encoding times. For this reason, we decided to provide an application that allows you to choose one between the two encoders to study, CLIP or ALBEF. Moreover, we provided a more formal description of the Recommender System. Finally, the experiments to evaluate the performance of the RS have proved that it can achieve good levels of Mean Average Precision under two different experimental setups.

6 References

Bazzani, L., Larochelle, H., & Torresani, L. (2016). Recurrent mixture density network for spatiotemporal visual attention. https://doi.org/10.48550/arXiv.1603.08199

Bellitto, G., Proietto Salanitri, F., Palazzo, S., Rundo, F., Giordano, D., & Spampinato, C. (2021). Hierarchical domain-adapted feature learning for video saliency prediction. *International Journal of Computer Vision*, *129*, 3216-3232. https://doi.org/10.1007/s11263-021-01519-y

Bertasius, G., Wang, H., & Torresani, L. (2021, July 18-24). *Is space-time attention all you need for video understanding?*. ICML, Virtual. https://proceedings.mlr.press/v139/bertasius21a.html

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018, May 15-19). *VGGFace2: A dataset for recognising faces across pose and age*. 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), Xi'an, China. https://doi.org/10.1109/FG.2018.00020

Cao, Y. H., & Wu, J. (2022, February 22-March 1). *A Random CNN Sees Objects: One Inductive Bias of CNN and Its Applications*. AAAI, Virtual. https://doi.org/10.1609/aaai.v36i1.19894

Cerf, M., Harel, J., Einhaeuser, W., & Koch, C. (2007, December 3-6). *Predicting human gaze using low-level saliency combined with face detection*. Advances in Neural Information Processing Systems, Vancouver, British Columbia, Canada. https://proceedings.neurips.cc/paper/2007/hash/708f3cf8100d5e71834b1db77dfa15d6-Abstract.html

Chang, Q., & Zhu, S. (2021). Temporal-spatial feature pyramid for video saliency detection. https://doi.org/10.48550/arXiv.2105.04213

Changpinyo, S., & Sharma, P. (2021,June 19-25). Conceptual 12M: Pushing Web-Scale Image-Text Pre-Training To Recognize Long-Tail Visual Concepts. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual. https://doi.org/10.48550/arXiv.2102.08981

Chen, J., Song, H., Zhang, K., Liu, B., & Liu, Q. (2021). Video saliency prediction using enhanced spatiotemporal alignment network. *Pattern Recognition*, *109*, 107615. https://doi.org/10.1016/j.patcog.2020.107615

Chen, X., Lu, Y., Wang, Y., & Yang, J. (2021). CMBF: Cross-Modal-Based Fusion Recommendation Algorithm. *Sensors 21*(16):5275. https://doi.org/10.3390/s21165275

Das, K. A., Baruah, A., Barbhuiya, F. A., & Dey, K. (2020, December 12-13). *KAFK at SemEval-2020 Task 8: Extracting Features From Pre-trained Neural Networks To Classify Internet Memes*. Proceedings of the Fourteenth Workshop on Semantic Evaluation, Barcelona, Spain. http://dx.doi.org/10.18653/v1/2020.semeval-1.152

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). *ImageNet: A large-scale hierarchical image database*. IEEE conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2009.5206848

Devlin, J., & Chang, M. W. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL. https://arxiv.org/pdf/1810.04805v2.pdf

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. https://doi.org/10.48550/arXiv.2010.11929

Droste, R., Jiao, J., & Noble, J. A. (2020). Unified image and video saliency modeling. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16 (pp. 419-435). Springer International Publishing. https://doi.org/10.1007/978-3-030-58558-7_25

Erdem, E., & Erdem, A. (2013). Visual saliency estimation by nonlinearly integrating features using region covariances. Journal of vision, 13(4), 11-11. https://doi.org/10.1167/13.4.11

Feichtenhofer, C., Fan, H., Malik, J., & He, K. (2019). Slowfast networks for video recognition. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 6202-6211).

Gan, C., Wang, N., Yang, Y., Yeung, D. Y., & Hauptmann, A. G. (2015). Devnet: A deep event network for multimedia event detection and evidence recounting. In CVPR, pages 2568–2577. https://doi.org/10.1109/CVPR.2015.7298872

Guo, J., Deng, J., Lattas, A., & Zafeiriou, S. (2021). Sample and computation redistribution for efficient face detection. arXiv preprint arXiv:2105.04714. https://doi.org/10.48550/arXiv.2105.04714

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). https://doi.org/10.1109/CVPR.2016.90

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). pmlr.

Itti, L., Koch, V, & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 11, pp. 1254–1259.https://doi.org/10.1109/34.730558

Jain, S., Yarlagadda, P., Jyoti, S., Karthik, S., Subramanian, R., & Gandhi, V. (2021). ViNet: Pushing the limits of visual modality for audio-visual saliency prediction. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 3520-3527). IEEE. https://doi.org/10.48550/arXiv.2012.06170

Jiang, L., Xu, M., Liu, T., Qiao, M., Wang, Z. (2018). DeepVS: A Deep Learning Based Video Saliency Prediction Approach. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds) Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science(), vol 11218. Springer, Cham. https://doi.org/10.1007/978-3-030-01264-9_37

Kiela, D., Firooz, H., Mohan, A., Goswami, V., Singh, A., Ringshia, P., & Testuggine, D. (2020). The hateful memes challenge: Detecting hate speech in multimodal memes. Advances in Neural Information Processing Systems, 33, 2611-2624.

Kordopatis-Zilos, G., Tzelepis, C., Papadopoulos, S., Kompatsiaris, I., & Patras, I. (2022). DnS: Distill-and-Select for Efficient and Accurate Video Indexing and Retrieval. International Journal of Computer Vision, 130(10), 2385-2407.

Koutlis, C., Schinas, M., & Papadopoulos, S. (2022). MemeTector: Enforcing deep focus for meme detection. https://doi.org/10.48550/arXiv.2205.13268

Krishna, R., & Zhu, Y. (2017). Visual genome: Connecting language and vision using crowdsourced dense image annotations. IJCV. https://arxiv.org/pdf/1602.07332.pdf

Le Meur, O., Le Callet, P., Barba, D., & Thoreau, D. (2006). A coherent computational approach to model bottom-up visual attention, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 5, pp. 802–817. https://doi.org/10.1109/TPAMI.2006.86

Lai, Q., Wang, W., Sun, H., & Shen, J. (2019). Video saliency prediction using spatiotemporal residual attentive networks. IEEE Transactions on Image Processing, 29, 1113-1126. https://doi.org/10.1109/TIP.2019.2936112

Li, J., & Selvaraju, R. (2021). Align before Fuse: Vision and Language Representation Learning with Momentum Distillation. NeurIPS 2021. https://arxiv.org/pdf/2107.07651.pdf

Lin, T. Y., & Maire, M. (2014). Microsoft COCO: Common Objects in Context. European conference on computer vision. https://arxiv.org/pdf/1405.0312.pdf

Linardos, P., Mohedano, E., Nieto, J. J., O'Connor, N. E., Giro-i-Nieto, X., & McGuinness, K. (2019). Simple vs complex temporal recurrences for video saliency prediction. arXiv preprint arXiv:1907.01869. https://doi.org/10.48550/arXiv.1907.01869

Liu, N., Zhang, N., Wan, K., Shao, L., & Han, J. (2021). Visual saliency transformer. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 4722-4732). https://doi.org/10.48550/arXiv.2104.12099

Min, K., & Corso, J. J. (2019). Tased-net: Temporally-aggregating spatial encoder-decoder network for video saliency detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 2394-2403). https://doi.org/10.1109/ICCV.2019.00248

Ordonez, V., & Kulkarni, G. (2011). Im2Text: Describing Images Using 1 Million Captioned Photographs. NIPS. https://papers.nips.cc/paper/2011/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark,J., Krueger, G., Sutskever, I. (2021, July). Learning transferable visual models from natural language supervision. In International conference on machine learning (pp. 8748-8763). PMLR. https://doi.org/10.48550/arXiv.2103.00020

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.

Sarridis, I., Koutlis, C., Kordopatis-Zilos, G., Papadopoulos, S., & Kompatsiaris, I. (2022). InDistill: Information flow-preserving knowledge distillation for model compression. *arXiv preprint arXiv:2205.10003v2*. https://doi.org/10.48550/arXiv.2205.10003

Sharma, C., Bhageria, D., Scott, W., Pykl, S., Das, A., Chakraborty, T., ... & Gamback, B. (2020). SemEval-2020 Task 8: Memotion Analysis--The Visuo-Lingual Metaphor!. arXiv preprint arXiv:2008.03781.

Sharma, P., & Ding, N. (2018). Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning. Association for Computational Linguistics. https://aclanthology.org/P18-1238.pdf

Shimoda, W. and Yanai, K. (2016). Distinct class-specific saliency maps for weakly supervised semantic segmentation. In ECCV, pages 218–234. https://doi.org/10.1007/978-3-319-46493-0_14

Suryawanshi, S., Chakravarthi, B. R., Arcan, M., & Buitelaar, P. (2020, May). Multimodal meme dataset (MultiOFF) for identifying offensive content in image and text. In Proceedings of the second workshop on trolling, aggression and cyberbullying (pp. 32-41).

Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3D convolutional networks. In Proceedings of the IEEE international conference on computer vision (pp. 4489-4497). https://doi.org/10.1109/ICCV.2015.510

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In NIPS, pages 5998– 6008 https://doi.org/10.48550/arXiv.1706.03762

Walther, D., Koch, C. (2009). Modeling attention to salient proto-objects, Neural Networks 19 (9) 1395–1407. https://doi.org/10.1016/j.neunet.2006.10.001

Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., & Van Gool, L. (2016). Temporal segment networks: Towards good practices for deep action recognition. In European conference on computer vision (pp. 20-36). Springer, Cham. https://doi.org/10.48550/arXiv.1608.00859

Wang, Z., Liu, Z., Li, G., Wang, Y., Zhang, T., Xu, L., & Wang, J. (2021). Spatio-temporal self-attention network for video saliency prediction. IEEE Transactions on Multimedia. https://doi.org/10.1109/TMM.2021.3139743

Wang, W., Shen, J., Xie, J., Cheng, M. M., Ling, H., & Borji, A. (2019). Revisiting video saliency prediction in the deep learning era. IEEE transactions on pattern analysis and machine intelligence, 43(1), 220-237.https://doi.org/10.1109/TPAMI.2019.2924417

Wu, X., Wu, Z., Zhang, J., Ju, L., & Wang, S. (2020). SalSAC: A Video Saliency Prediction Model with Shuffled Attention and Correlation-Based ConvLSTM. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 07, pp. 12410-12417). https://doi.org/10.1609/aaai.v34i07.6927

Young, P., & Lai, A. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. Transactions of the Association for Computational Linguistics. https://aclanthology.org/Q14-1006.pdf

Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z. H., ... & Yan, S. (2021). Tokens-to-token vit: Training vision transformers from scratch on imagenet. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 558-567). https://doi.org/10.48550/arXiv.2101.11986

Zanca, D., Gori, M., Melacci, S. (2020) Gravitational models explain shifts on human visual attention. Sci Rep 10, 16335. https://www.nature.com/articles/s41598-020-73494-2

Annex I: MAAM - Content Retrieval Experiment

In the Figures A.1-A.6, we present scatter plots of similarity vs. sum of relevant object area for image based similarity as well as bounding box based similarity and different feature extractors.

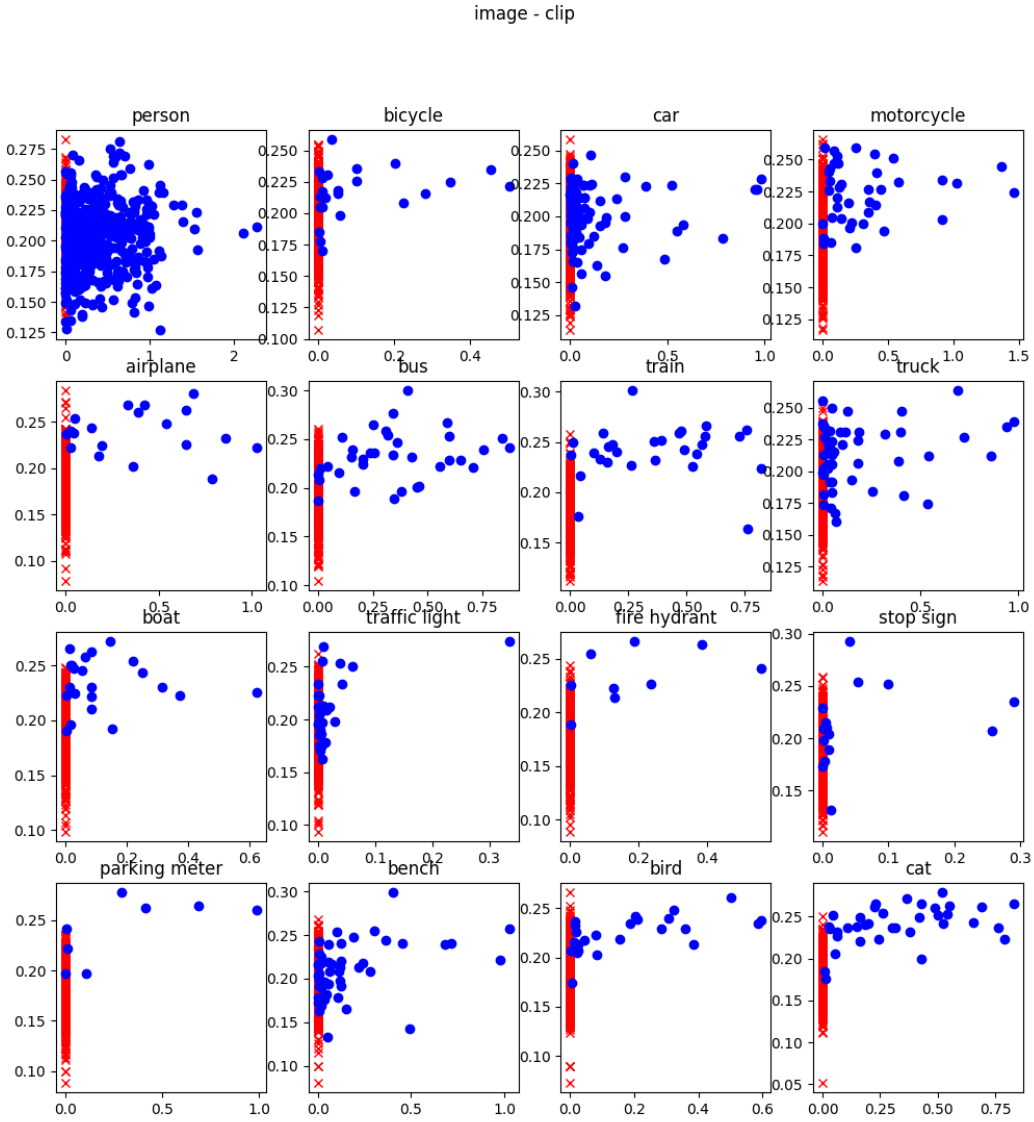

Figure A.1: Each scatter plot point represents a candidate image of the MS-COCO validation set (1000 randomly sampled per object category). The maximum cosine similarity between prototypes' CLIP feature vectors and candidate image's CLIP feature vector is on Y-axis and sum of area occupied by bounding boxes containing the object of interest is on X-axis. Red x denotes images without the object and blue dots denote images with the object. The more distinct the Y-axis distributions of the two the better the performance is based on an optimal threshold.

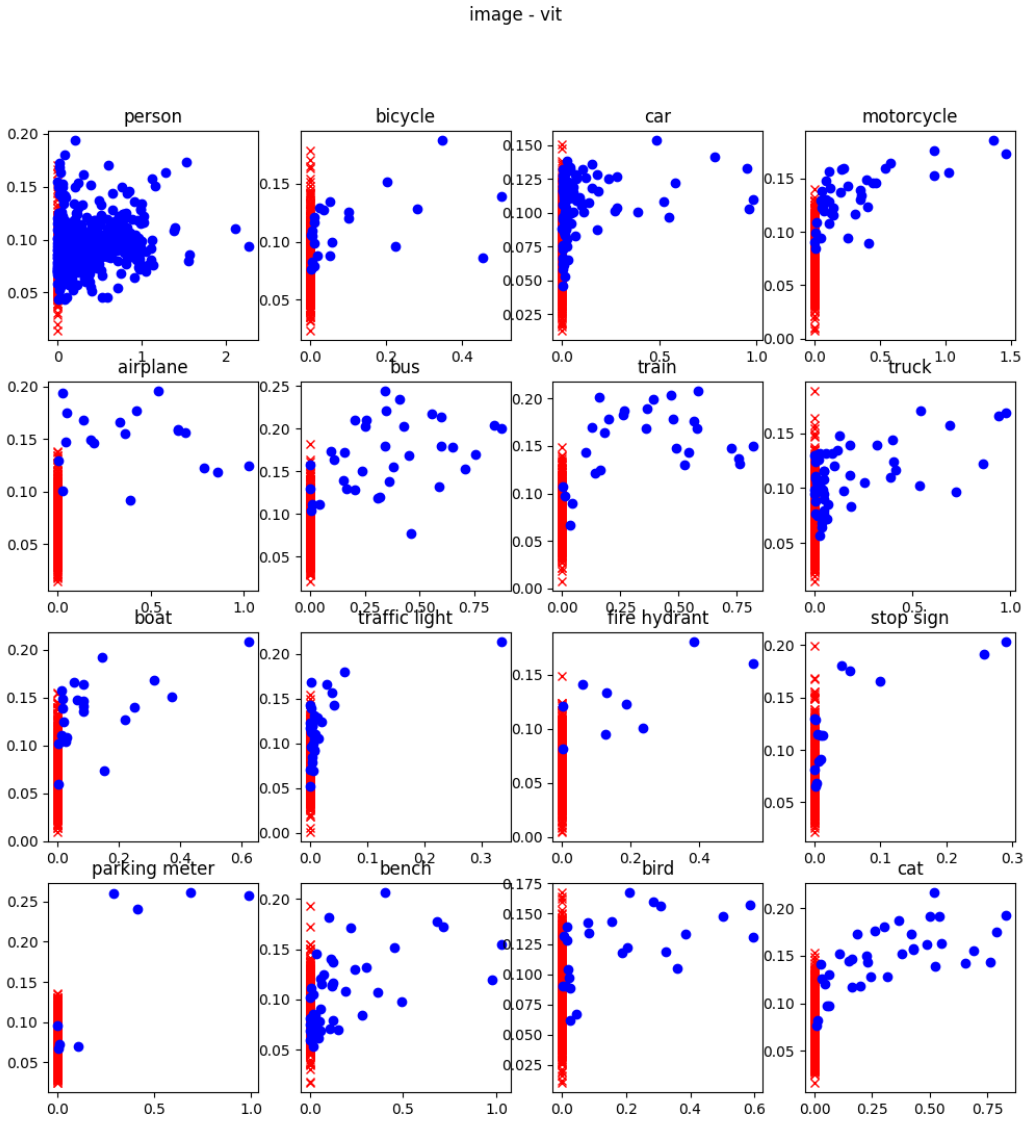

Figure A.2: Each scatter plot point represents a candidate image of the MS-COCO validation set (1000 randomly sampled per object category). The maximum cosine similarity between prototypes' ViT feature vectors and candidate image's ViT feature vector is on Y-axis and sum of area occupied by bounding boxes containing the object of interest is on X-axis. Red x denotes images without the object and blue dots denote images with the object. The more distinct the Y-axis distributions of the two the better the performance is based on an optimal threshold.

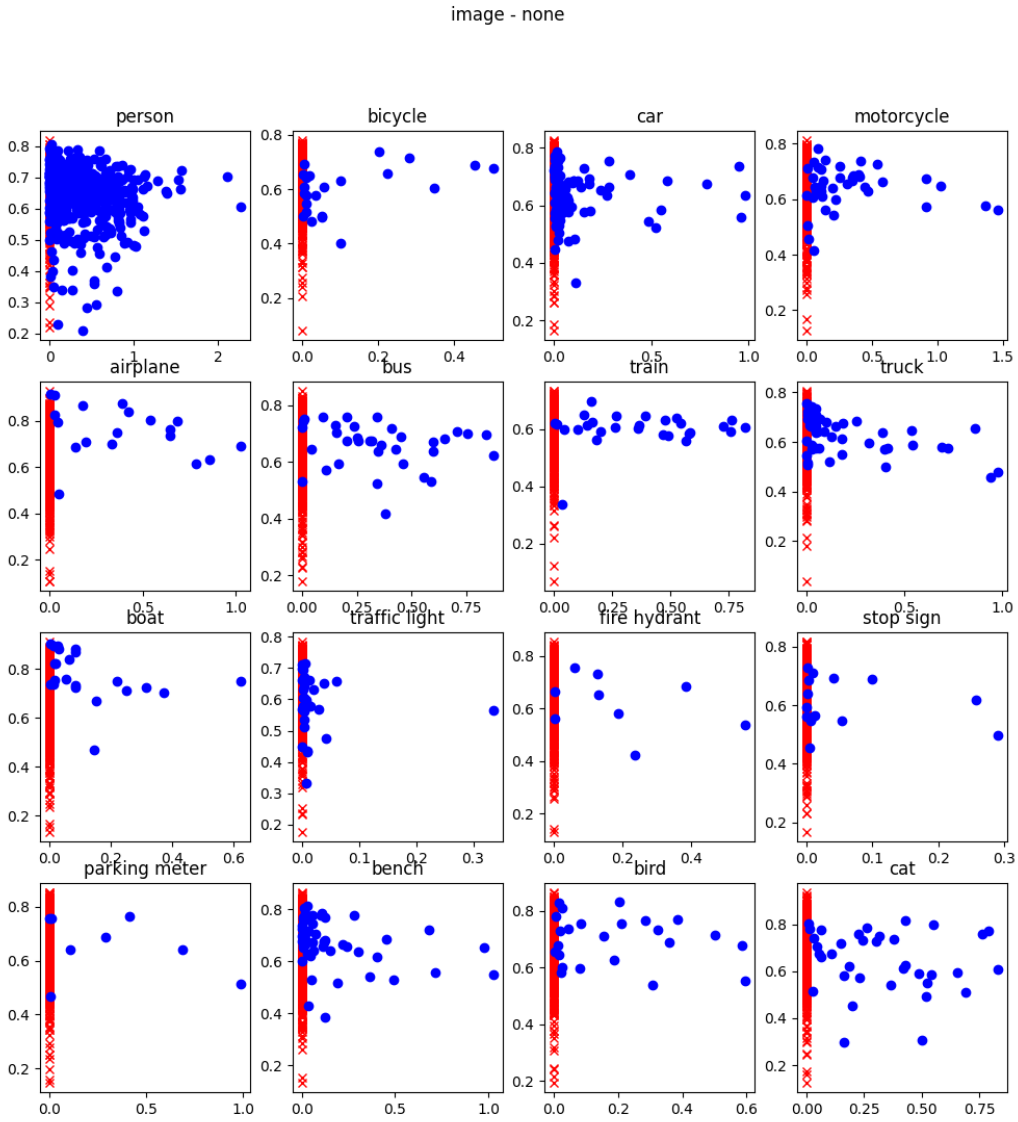

Figure A.3: Each scatter plot point represents a candidate image of the MS-COCO validation set (1000 randomly sampled per object category). The maximum UQI between prototype images and each candidate image is on Y-axis, and sum of area occupied by bounding boxes containing the object of interest is on X-axis. Red x denotes images without the object and blue dots denote images with the object. The more distinct the Y-axis distributions of the two the better the performance is based on an optimal threshold.

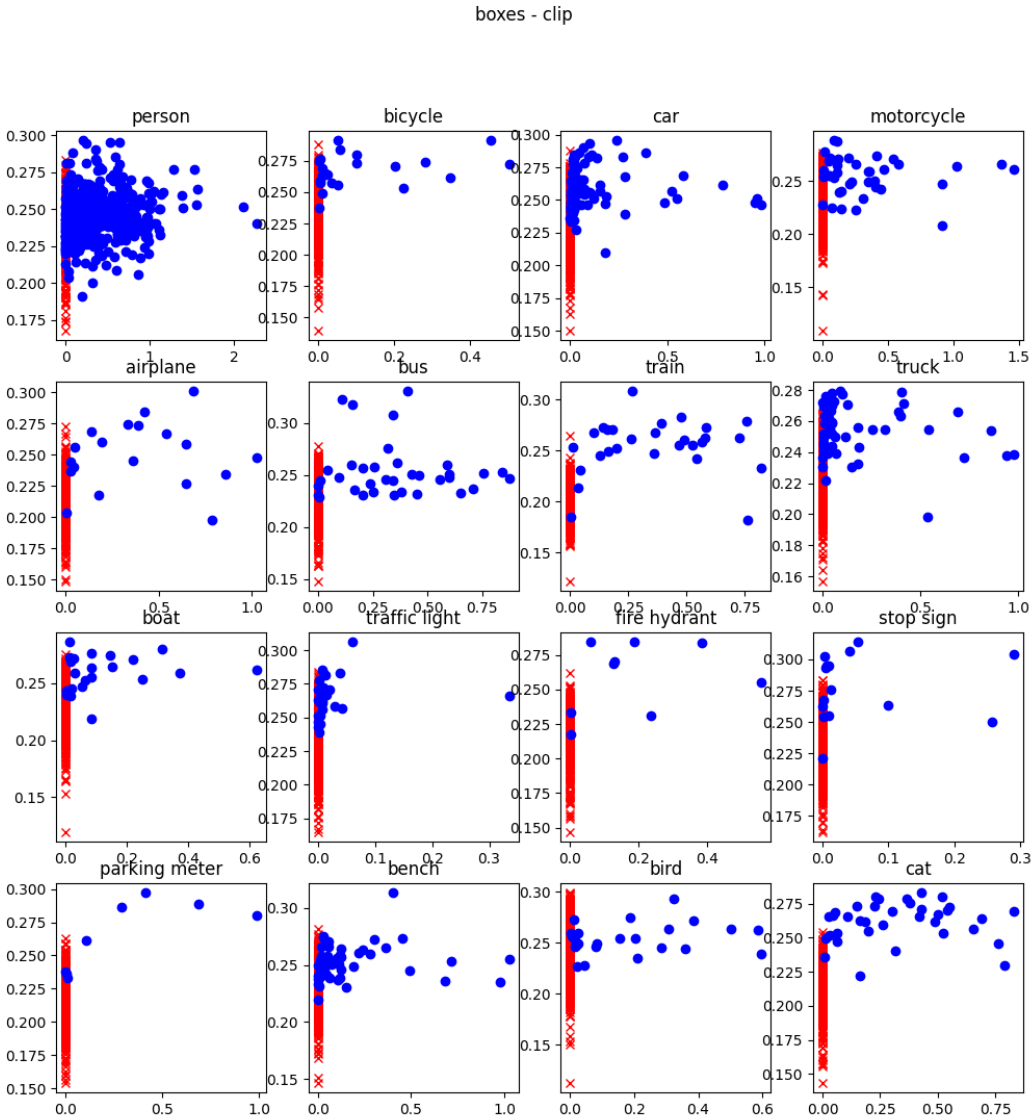

Figure A.4: Each scatter plot point represents a candidate image of the MS-COCO validation set (1000 randomly sampled per object category). The maximum cosine similarity between all prototypes' CLIP feature vectors and all candidate image cropped bounding boxes' CLIP feature vectors is on Y-axis and sum of area occupied by bounding boxes containing the object of interest is on X-axis. Red x denotes images without the object and blue dots denote images with the object. The more distinct the Y-axis distributions of the two the better the performance is based on an optimal threshold.

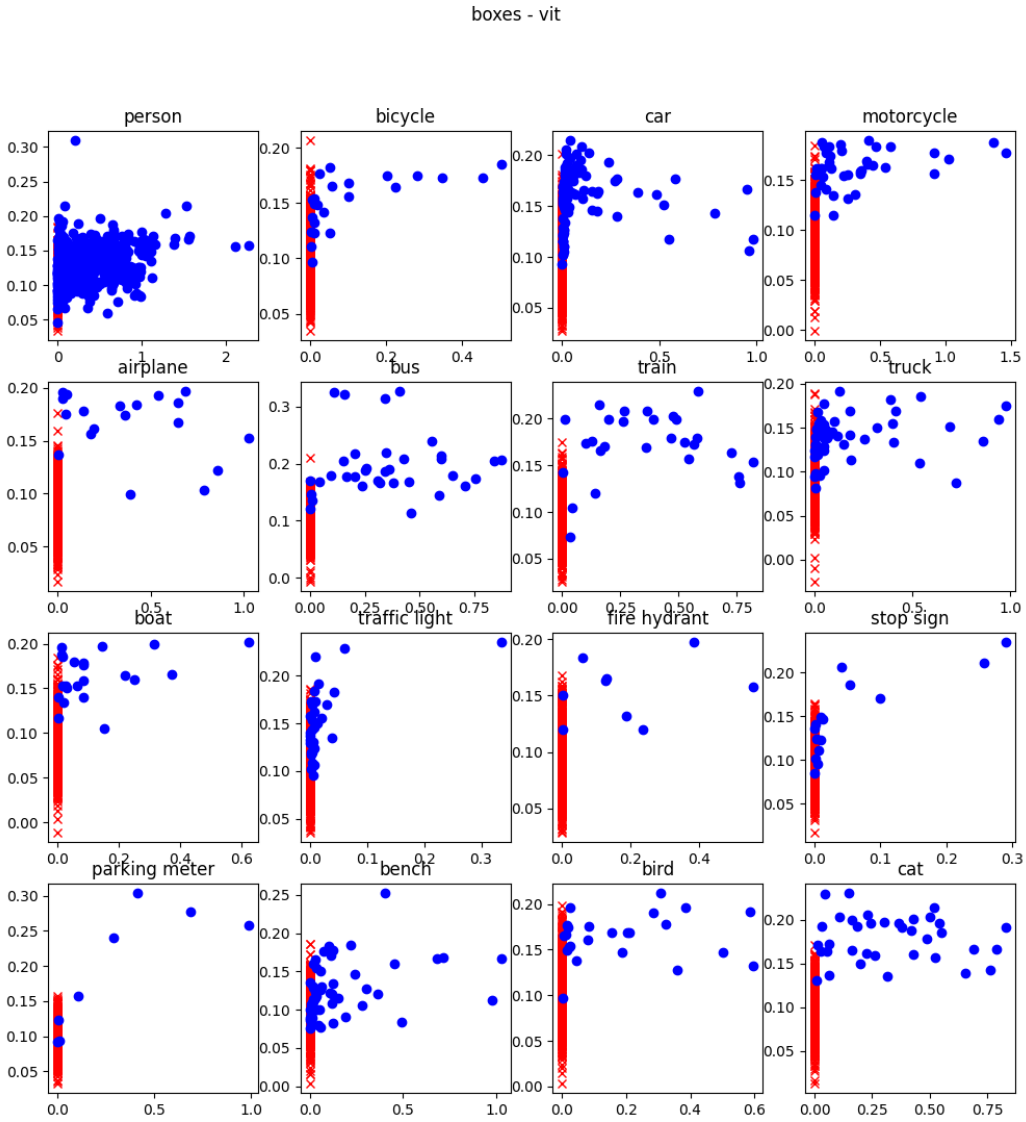

Figure A.5: Each scatter plot point represents a candidate image of the MS-COCO validation set (1000 randomly sampled per object category). The maximum cosine similarity between all prototypes' ViT feature vectors and all candidate image cropped bounding boxes' ViT feature vectors is on Y-axis and sum of area occupied by bounding boxes containing the object of interest is on X-axis. Red x denotes images without the object and blue dots denote images with the object. The more distinct the Y-axis distributions of the two the better the performance is based on an optimal threshold.

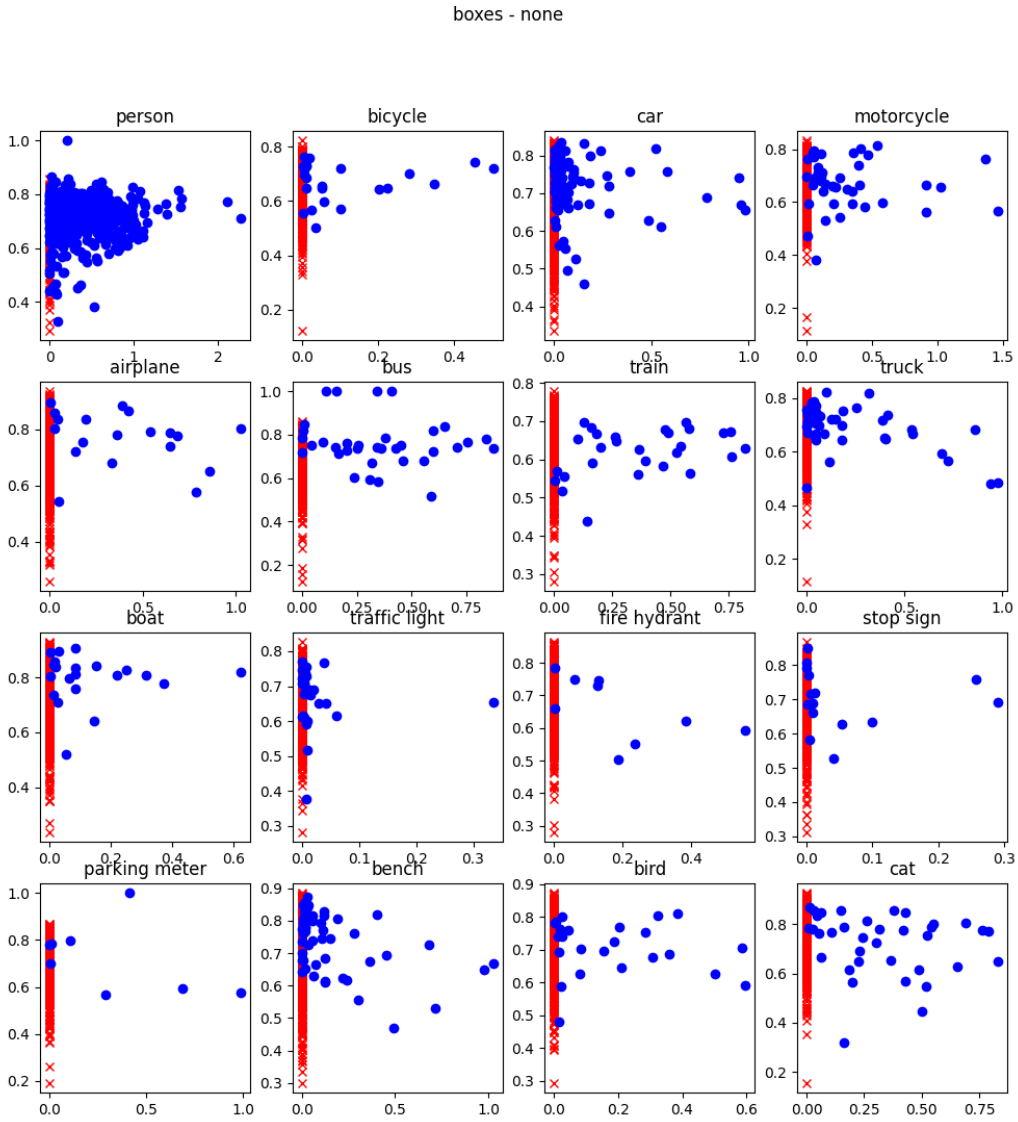

Figure A.6: Each scatter plot point represents a candidate image of the MS-COCO validation set (1000 randomly sampled per object category). The maximum UQI between all prototype images and all candidate image cropped bounding boxes is on Y-axis and sum of area occupied by bounding boxes containing the object of interest is on X-axis. Red x denotes images without the object and blue dots denote images with the object. The more distinct the Y-axis distributions of the two the better the performance is based on an optimal threshold.

Annex II: Media Annotation Service gRPC API

Table B.1: The Media Annotation Service gRPC API

```
syntax = "proto3";

service MediaAnnotationServer {
    // Server streaming annotation endpoints
    rpc ImageAnnotationStream(ImageAnnotationRequest) returns (stream
ImageAnnotationStreamResponse);
    rpc ImageAnnotationStreamBatch(ImageAnnotationRequestBatch) returns (stream
ImageAnnotationStreamResponseBatch);
    rpc VideoAnnotationStream(VideoAnnotationRequest) returns (stream
VideoAnnotationStreamResponse);

    // Unary annotation endpoints
    rpc ImageAnnotation(ImageAnnotationRequest) returns (ImageAnnotationResponse);
    rpc ThreeDAnnotation(ThreeDAnnotationRequest) returns (Annotation3dResponse);
    rpc VideoAnnotation(VideoAnnotationRequest) returns (VideoAnnotationResponse);

    // Single model endpoints
    rpc ImageActionRecognition(AnnotationRequest) returns
(ImageActionRecognitionResponse);
    rpc ImageCaptioning(AnnotationRequest) returns (ImageCaptioningResponse);
    rpc ImageCrossModalEmbedding(AnnotationRequest) returns
(ImageCrossModalEmbeddingResponse);
    rpc ImageDisturbingContentDetection(AnnotationRequest) returns
(ImageDisturbingContentDetectionResponse);
    rpc ImageFaceRecognition(AnnotationRequest) returns (ImageFaceRecognitionResponse);
    rpc ImageMemeDetection(AnnotationRequest) returns (ImageMemeDetectionResponse);
    rpc ImageNSFWDetection(AnnotationRequest) returns (ImageNSFWDetectionResponse);
    rpc ImageObjectDetection(AnnotationRequest) returns (ImageObjectDetectionResponse);
    rpc ImageRegionFeatureExtraction(AnnotationRequest) returns
(ImageRegionFeatureExtractionResponse);
    rpc VideoActionRecognition(AnnotationRequest) returns
(VideoActionRecognitionResponse);
    rpc VideoDisturbingContentDetection(AnnotationRequest) returns
(VideoDisturbingContentDetectionResponse);
    rpc VideoFaceBlurring(AnnotationRequest) returns (VideoFaceBlurringResponse);
    rpc VideoFaceRecognition(AnnotationRequest) returns (VideoFaceRecognitionResponse);
    rpc VideoNSFWDetection(AnnotationRequest) returns (VideoNSFWDetectionResponse);
    rpc VideoObjectDetection(AnnotationRequest) returns (VideoObjectDetectionResponse);
}

message AnnotationRequest {
    // asset_url and asset_data are mutually exclusive
    string asset_url = 1;
    bytes asset_data = 2;
}

message ImageAnnotationRequest {
    // image_url and image_data are mutually exclusive
    string image_url = 1;
```

```
    bytes image_data = 2;
}

message ImageAnnotationResponse {
    // Its value should be interpreted as documented
[here](https://grpc.github.io/grpc/core/md_doc_statuscodes.html)
    // 0 means OK, non-zero values mean error
    int32 status = 1;

    // If set, status will be non-zero and this field will describe the error that
happened.
    string error_msg = 2;

    repeated string valid_models = 3;

    ImageActionRecognitionResponse image_action_recognition_result = 10;
    ImageCaptioningResponse image_captioning_result = 11;
    ImageCrossModalEmbeddingResponse image_cross_modal_embedding_result = 12;
    ImageDisturbingContentDetectionResponse image_disturbing_content_detection_result =
13;
    ImageFaceRecognitionResponse image_face_recognition_result = 14;
    ImageMemeDetectionResponse image_meme_detection_result = 15;
    ImageObjectDetectionResponse image_object_detection_result = 16;
    ImageNSFWDetectionResponse image_nsfw_detection_result = 17;
}

message ImageAnnotationStreamResponse {
    oneof response {
        ImageActionRecognitionResponse image_action_recognition_result = 10;
        ImageCaptioningResponse image_captioning_result = 11;
        ImageCrossModalEmbeddingResponse image_cross_modal_embedding_result = 12;
        ImageDisturbingContentDetectionResponse
image_disturbing_content_detection_result = 13;
        ImageFaceRecognitionResponse image_face_recognition_result = 14;
        ImageMemeDetectionResponse image_meme_detection_result = 15;
        ImageObjectDetectionResponse image_object_detection_result = 16;
        ImageNSFWDetectionResponse image_nsfw_detection_result = 17;
    }
}

message VideoAnnotationRequest {
    // video_url and video_data are mutually exclusive
    string video_url = 1;
    bytes video_data = 2;
}

message VideoAnnotationResponse {
    // Its value should be interpreted as documented
[here](https://grpc.github.io/grpc/core/md_doc_statuscodes.html)
    // 0 means OK, non-zero values mean error
    int32 status = 1;

    // If set, status will be non-zero and this field will describe the error that
happened.
    string error_msg = 2;
```

```
    repeated string valid_models = 3;

    VideoActionRecognitionResponse video_action_recognition_result = 10;
    VideoDisturbingContentDetectionResponse video_disturbing_content_detection_result =
13;
    VideoFaceRecognitionResponse video_face_recognition_result = 11;
    VideoNSFWDetectionResponse video_nsfw_detection_result = 14;
    VideoObjectDetectionResponse video_object_detection_result = 12;
}

message VideoAnnotationStreamResponse {
    oneof response {
        VideoActionRecognitionResponse video_action_recognition_result = 10;
        VideoDisturbingContentDetectionResponse
video_disturbing_content_detection_result = 13;
        VideoFaceRecognitionResponse video_face_recognition_result = 11;
        VideoNSFWDetectionResponse video_nsfw_detection_result = 14;
        VideoObjectDetectionResponse video_object_detection_result = 12;
    }
}

message ThreeDAnnotationRequest {
    string asset_url = 1;

    // The format of the 3D object. Available values are obj, off, ply, stl, glb, gltf
    string asset_type = 2;
}
```

MediaVerse is an H2020 Innovation Project co-financed by the EC under Grant Agreement ID: 957252.