



MediaVerse

A universe of media assets
and co-creation opportunities

D3.1

Next Generation Content Model and Algorithms for New Media Types

| | |
|--------------------------|-----------------------------------|
| 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 | Next Generation Content Model and Algorithms for New Media Types |
| Deliverable number | D3.1 |
| Deliverable version | V1.0 |
| Previous version(s) | N/A |
| Contractual Date of delivery | 30.09.2021 |
| Actual Date of delivery | 30.09.2021 |
| Nature of deliverable | Report |
| Dissemination level | Public |
| Partner Responsible | ATOS |
| Author(s) | Francesco D'Andria, Enrique Quiros, Ruben Ramiro (ATOS), Elisavet Chatzilari, Manos Schinas, Symeon Papadopoulos (CERTH), Stephan Gensch (VRAG) |
| Reviewer(s) | Ronny Esterluß (VRAG), Nikos Sarris (CERTH) |
| EC Project Officer | Alberto Rabbachin |

| | |
|-----------------|--|
| Abstract | The main objective of this deliverable named D3.1 is to provide a preliminary overview of the architecture of the MediaVerse Next Generation Content Management, Understanding and Interlinking layer. |
| Keywords | Blockchain, Media, Urban Journalism |

Copyright

© Copyright 2021 MediaVerse Consortium

This document may not be copied, reproduced, or modified in whole or in part for any purpose without written permission from the MediaVerse Consortium. In addition to such written permission to copy, reproduce, or modify this document in whole or part, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced.

All rights reserved.



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 | 28/06/2021 | Francesco D’Andria | First Draft Table of Content |
| V0.2 | 01/07/2021 | Francesco D’Andria | Final Table of Content |
| V0.3 | 25/07/2021 | Francesco D’Andria, Enrique Quiros, Ruben Ramiro, Elisavet Chatzilari, Manos Schinas, Symeon Papadopoulos, Stephan Gensch | First Draft |
| V0.4 | 02/09/2021 | Francesco D’Andria, Enrique Quiros, Ruben Ramiro, Elisavet Chatzilari, Manos Schinas, Symeon Papadopoulos, Stephan Gensch | Second Draft |
| V0.5 | 19/09/2021 | Francesco D’Andria | Final Second Draft |
| V0.6 | 21/09/2021 | Ronny Esterluß | Review |
| V0.7 | 23/09/2021 | Nikos Sarris | Review |
| V0.8 | 27/09/2021 | Francesco D’Andria | Revisions |
| V0.9 | 29/09/2021 | Symeon Papadopoulos | Final QA |
| V1.0 | 30/09/2021 | Francesco D’Andria, Enrique Quiros, Ruben Ramiro | Final Document |

Glossary

| ABBREVIATION | MEANING |
|--------------|--|
| BAAF-NET | Bilateral Augmentation and Adaptive Fusion |
| BiFPN | bi-directional feature pyramid networks |
| CNN | Convolutional Neural Network |
| CDN | Content Delivery Network |
| CIFAR | Canadian Institute for Advanced Research |
| CMS | Content Management Service |
| DAM | Digital Asset Management |
| DASH | Dynamic Adaptive Streaming over HTTP |
| DL | Deep Learning |
| FGFA | Flow-Guided Feature Aggregation |
| FOCS | Face and Ocular Challenge Series |
| GDPR | General Data Protection Regulation |
| GUI | Graphical User Interface |
| HEVC | High Efficiency Video Coding |
| HLS | HTTP Live Streaming |
| HMD | Head-mounted Display |
| IJB-B | IARPA JANUS Benchmark B |
| IJB-S | IARPA Janus Surveillance Video Benchmark |
| KG | Knowledge Graph |
| QoE | Quality of Experience |
| LSTM | Long Short-Term |

| | |
|--------|--|
| MBGC | Multiple Biometric Grand Challenge |
| MEP | Media Processing Entity |
| MTIL | Multi-Trajectory Incremental Learning |
| MNIST | Mixed National Institute of Standards and Technology |
| MSCOCO | Microsoft's Common Objects in Context |
| MTCNN | Multitask Cascade Convolutional Neural Network |
| MV | MediaVerse |
| NBMP | Network-based Media Processing |
| NEL | Named Entity Linking |
| NMS | Non-maximum suppression |
| OMAF | Omnidirectional Media Format |
| QoE | Quality of Experience |
| NER | Named Entity Recognition |
| PVCNN | Point-Voxel CNN |
| R-CNN | Region-Based Convolutional Neural Network |
| RBF | Radial Basis Function |
| RPN | Region Proposal Networks |
| RSRC | Regularized Sparse Representation Classification |
| SSCNs | Submanifold Sparse Convolutional Networks |
| SS | Semantic Segmentation |
| SVM | Support Vector Machine |
| TSN | Temporal Segment Network |
| UI | User Interface |
| VAE | Variational Autoencoder |
| VR | Virtual Reality |
| VV-Net | Voxel VAE Net |
| WCAG | Web Content Accessibility Guidelines |
| WDD | Workflow Description Document |
| WP | Work Package |
| YOLO | You Only Look Once |

Table of Contents

| | |
|---|----|
| Revision History | 3 |
| Glossary | 3 |
| Index of Figures | 7 |
| Index of Tables..... | 8 |
| Executive Summary | 10 |
| 1 Introduction..... | 11 |
| 1.1 Purpose and structure of the document..... | 12 |
| 2 New media Understanding and Annotation..... | 13 |
| 2.1 Usage Scenarios..... | 13 |
| 2.2 Methods and Materials | 14 |
| 2.2.1 Image data | 15 |
| 2.2.2 3-Dimensional Data | 49 |
| 2.3 Technical Implementation & API..... | 61 |
| 3 Multilingual Annotation Tool..... | 65 |
| 3.1 Named Entity Recognition..... | 65 |
| 3.2 Description of the Tool | 66 |
| 3.2.1 Transner..... | 66 |
| 3.2.2 List of Recognized Entities | 68 |
| 3.2.3 List of Supported Languages..... | 68 |
| 3.2.4 Quantitative Results | 68 |
| 3.3 API Description | 69 |
| 3.3.1 Project folder | 69 |
| 3.3.2 Installation and Environment Setup..... | 70 |
| 3.3.3 How to Use | 72 |
| 4 Content Discovery and Interlinking services | 74 |
| 4.1 Introduction..... | 74 |
| 4.2 Usage Scenario and Requirements..... | 75 |
| 4.2.1 Retrieval Scenario | 75 |
| 4.2.2 Recommendation Scenario | 75 |
| 4.2.3 Requirements | 75 |
| 4.3 Methodology | 76 |
| 4.3.1 Training Procedure | 77 |

| | | |
|-------|---|-----|
| 4.3.2 | Training Data | 79 |
| 4.3.3 | Results | 80 |
| 4.3.4 | Next Steps..... | 80 |
| 4.4 | Implementation..... | 81 |
| 4.4.1 | Project folder | 82 |
| 4.4.2 | Installation and Environment Setup | 83 |
| 4.4.3 | Usage | 84 |
| 5 | Content Adaptation Pipeline | 85 |
| 5.1 | Introduction..... | 85 |
| 5.2 | Content Provider | 85 |
| 5.3 | MediaVerse Content Delivery | 85 |
| 5.3.1 | Video Encoding and Transcoding | 86 |
| 5.3.2 | Audio Transcoding | 89 |
| 5.3.3 | Picture Optimization..... | 89 |
| 5.3.4 | Streaming Engine..... | 89 |
| 6 | MediaVerse Data Model for content description | 94 |
| 6.1 | Introduction..... | 94 |
| 6.2 | MediaVerse Metadata..... | 94 |
| 6.2.1 | Other Specifications..... | 94 |
| 6.2.2 | Handmade Specification..... | 96 |
| 6.2.3 | EBUCore..... | 97 |
| 6.2.4 | Metadata Section. | 97 |
| 6.3 | API to Manage the MediaVerse Metadata..... | 101 |
| 6.3.1 | Metadata API..... | 101 |
| | References..... | 104 |
| | Annex I: Implementation Details of API | 115 |

Index of Figures

| | |
|--|----|
| Figure 1: MediaVerse WP3 structure in Tasks..... | 11 |
| Figure 2: MediaVerse Content Ingestion, Adaptation and Delivery Pipeline..... | 12 |
| Figure 3: Baseline image captioning model architecture (keras model)..... | 21 |
| Figure 4: (Left) image meme, (right) regular image. | 23 |
| Figure 5: Misclassified images (1st experiment) | 25 |
| Figure 6: Examples of image memes from the MemeGenerator dataset..... | 26 |
| Figure 7: Misclassified image memes (3rd experiment) | 27 |
| Figure 8: All images of the “novel images set”. | 28 |
| Figure 9: Example image from the MSCOCO dataset. | 30 |
| Figure 10: The contrastive learning framework. | 31 |
| Figure 11: Retrieved images for query “football” | 32 |
| Figure 12: Image retrieval from image query (top image) | 35 |
| Figure 13: Image relevant to the “Bernie Sanders” meme. Source: dw.com | 35 |
| Figure 14: Deep Face recognition architecture evolution. | 37 |
| Figure 15: Distribution of continents for YouTube dataset..... | 39 |
| Figure 16: Distribution of continents for YouTube dataset..... | 39 |
| Figure 17: Choropleth map for YouTube dataset..... | 39 |
| Figure 18: Choropleth map for VGG Face 2 dataset..... | 39 |
| Figure 19: Video object detection methods sorted in different groups (Zhu et al., 2020) | 46 |
| Figure 20: General overview of CNN-RNN video caption generation method (Islam et al., 2021)..... | 47 |
| Figure 21: General overview of video caption generation method where RNN is used in Encoding stage | 47 |
| Figure 22: A representation of different 3D data types | 50 |
| Figure 23: PointNet++ architecture which can be used for classification and segmentation..... | 53 |
| Figure 24: The entire BAAF-Net architecture (Qiu et al. 2021) | 55 |
| Figure 25: Semantic Segmentation results in S3DIS dataset..... | 58 |
| Figure 26: Results of annotation mismatch..... | 61 |
| Figure 27: gRPC communication flow (Google. 2021)..... | 62 |
| Figure 28: Flow diagram of MV-annotation service | 64 |
| Figure 29: Results of NER applied on a piece of a Wikipedia article | 66 |
| Figure 30: Example of nested entities. | 67 |
| Figure 31: Transner main components..... | 67 |
| Figure 32: Folders structure of Transner | 69 |
| Figure 33: Structure of the folder called Transner | 70 |
| Figure 34: OneDrive page for project download..... | 70 |
| Figure 35: Project unzip with 7zip | 71 |
| Figure 36: Creation of the Python virtual environment for the project with pipenv..... | 71 |
| Figure 37: Installation of the requirement for the project..... | 72 |
| Figure 38: Figure 38: Activation of the Python virtual environment with pipenv..... | 72 |
| Figure 39: Figure 39: Running of Transner with Python..... | 72 |
| Figure 40: Making the call to Transner REST API..... | 73 |
| Figure 41: Result of the Transner REST API | 73 |
| Figure 42: Neural Network architecture..... | 79 |
| Figure 43: Diagram block showing the inner workings of the developed service..... | 82 |

| | |
|--|-----|
| Figure 44: Project folder..... | 82 |
| Figure 45: Download page..... | 83 |
| Figure 46: Unzipping of the downloaded project archive | 83 |
| Figure 47: Installation of the Python environment | 83 |
| Figure 48: Installation of the required packages..... | 84 |
| Figure 49: How to run the service. Remember to set the --port parameter..... | 84 |
| Figure 50: Example of request for the service..... | 84 |
| Figure 51: Example of service response | 84 |
| Figure 52: MediaVerse Content adaptation pipeline | 85 |
| Figure 53: Power of Compression | 86 |
| Figure 54: MediaVerse Content adaptation workflow..... | 88 |
| Figure 55: Video-on-Demand test | 90 |
| Figure 56: Overview on the scope of Fraunhofer OMAF generator and player implementation..... | 91 |
| Figure 57: OMAF JavaScript player for web browsers..... | 92 |
| Figure 58: NBMP reference architecture [OMAF4CLOUD] | 93 |
| Figure 59: Dublin Core conceptual diagram | 95 |
| Figure 60: Netflix metadata environment cycle | 96 |
| Figure 61: Example of EBUCore..... | 97 |
| Figure 62: EBUCore asset extended and adapted to MediaVerse | 99 |
| Figure 63: EBUCore JSON representation | 100 |
| Figure 64: EBUCore basic metadata template | 101 |
| Figure 65: MediaVerse metadata template API service | 102 |
| Figure 66: getArchive example | 103 |
| Figure 67: updateAddArchive example | 103 |
| Figure 68: updateAddAnnotation example | 103 |
| Figure 69: deleteAnnotation example..... | 103 |

Index of Tables

| | |
|---|----|
| Table 1: List of relevant user requirements | 13 |
| Table 2: Components of T3.1 annotation module. | 14 |
| Table 3: Image classification inference results using ResNet50 and Inception ResNet v2 | 17 |
| Table 4: Object detection inference results using Faster R-CNN Inception v2 and EfficientDetD7 | 18 |
| Table 5: Inference results from the hybrid approach..... | 19 |
| Table 6: Computational efficiency and accuracy of the chosen pre-trained models for image classification..... | 20 |
| Table 7: BLEU scores for multiple alterations performed on the base model termed here LSTM. | 22 |
| Table 8: Captions generated by the baseline and the best model among the investigated options..... | 22 |
| Table 9: Counts of classes and training/validation/test sets (1st experiment)..... | 24 |
| Table 10: Performance of VGG16 on training, validation and test sets in terms of log-loss and accuracy | 24 |
| Table 11: Performance of VGG16 on novel images (1st experiment)..... | 24 |
| Table 12: Counts of classes and training/validation/test sets (2nd experiment) | 26 |

| | |
|---|-----|
| Table 13: Performance of VGG16 on training, validation and test sets in terms of log-loss and accuracy | 26 |
| Table 14: Performance of VGG16 on novel images (2nd experiment) | 26 |
| Table 15: Counts of classes and training/validation/test sets (3rd experiment) | 27 |
| Table 16: Performance of VGG16 on training, validation and test sets in terms of log-loss and accuracy | 27 |
| Table 17: Performance of VGG16 on novel images (3rd experiment) | 27 |
| Table 18: Model performance on meme detection task..... | 28 |
| Table 19: Positive image/text pairs of MSCOCO dataset in training and validation sets..... | 30 |
| Table 20: Evaluation metrics for text to image (T2I) and image to text (I2T) retrieval | 32 |
| Table 21: Words of average frequency in the validation set of MSCOCO dataset..... | 33 |
| Table 22: Comparative study for image and text retrieval on MSCOCO validation set | 33 |
| Table 23: Performance of variants of multi-head cross-modal attention method on the MSCOCO 5K test set . | 36 |
| Table 24: Dataset research..... | 38 |
| Table 25: Statistics for Celeb-A and YouTube datasets | 41 |
| Table 26: First training batch using pretrained networks | 41 |
| Table 27: SVM training using VGG Face 2 and dataset statistics | 42 |
| Table 28: Transfer learning on VGG Face 2 using VGG Face 2 dataset | 42 |
| Table 29: Fine tuning of VGG Face 2 | 43 |
| Table 30: Datasets for 3D scenes. | 51 |
| Table 31: Semantic Segmentation (6-fold cross-validation) results on the S3DIS dataset | 56 |
| Table 32: Results using pre-trained models from BAAF-NET in S3DIS dataset. | 57 |
| Table 33: Annotation results for all areas of S3DIS | 59 |
| Table 34: Annotation results for all areas of S3DIS dataset using 5000 threshold | 60 |
| Table 35: Example of a config.pbtxt file | 62 |
| Table 36: Example of Protobuf file | 63 |
| Table 37: Command to run protoc compiler | 63 |
| Table 38: Function example of the buffer | 64 |
| Table 39: Statistics for the WikiNER portion of data we used to assess the quality of the annotation tool | 68 |
| Table 40: Quantitative results over precision, recall, and F1 metrics | 69 |
| Table 41: Requirement satisfied by T3.2 | 76 |
| Table 42: Requirements necessary for T3.2 | 76 |
| Table 43: Results on the test split averaged over 10 runs | 80 |
| Table 44: EBUCore | 98 |
| Table 45: MovieLabs..... | 99 |
| Table 46: Example of the face recognition service with one known face | 115 |
| Table 47: Example of the face recognition service with multiple known faces | 116 |
| Table 48: Example of the face recognition service with an image with an unknown face | 117 |
| Table 49: Example of the face recognition service with an image without faces | 118 |
| Table 50: First example of the object detection service | 118 |
| Table 51: Second example of the image annotation service | 119 |
| Table 52: Example of the image annotation service | 121 |
| Table 53: First example of meme detection with a meme type image..... | 122 |
| Table 54: Second example of meme detection with a regular image..... | 122 |

Executive Summary

This public deliverable named D3.1 “Next Generation Content Model and Algorithms for New Media Types” aims to provide a first version of the MediaVerse Next Generation Content Management, Understanding and Interlinking layer. It provides an overview of functionalities for multimedia content management and discovery including multimodal feature extraction through deep learning pipelines for New Media annotation and indexing, content filtering, classification, recommendation and retrieval as well as content adaptation and finally definition of a data model for New Media content description that fits the requirements of the MediaVerse use cases.

Section 2 regarding new media understanding and annotation elaborates on the conducted experiments and the results as well as the current state of the art of several approaches for visual feature extraction and tagging. More precisely, we experiment on images with pre-trained and trained-from-scratch deep learning models for classification, object detection, captioning, meme detection, cross modal retrieval and celebrity recognition with fair results' quality so as to be incorporated in the first version of the platform. Additionally, we provide a review of the current state of the art on video annotation (classification, object detection, captioning) that will concern us the next period of the project. Also, immersive content analysis has been conducted with a comparative study of models on the task of semantic segmentation of 3D scenes, while 360 content annotation will be on our focus on the second year of the project. Finally, the media annotation API is designed in a very flexible way so as to facilitate model replacement when improved versions of the existing components arise at any time which is a core goal of T3.1.

As part of contribution for T3.1, Section 3 presents Transner, a multilingual annotation tool to be applied on social media posts for the purpose of extracting named entities from text in different languages

Section 4 on content Discovery and Interlinking services describes the challenge of content retrieval inside MediaVerse together with the approach we use to implement a learning-to-rank model based on Deep Learning. The current version of the model uses content's metadata extracted from T3.1 to compute the similarity between contents. In particular, we use the information about objects detected inside images to create a semantic representation of visual contents. The “results” section contains the quantitative evaluation of our approach using metrics derived from the scientific literature. We also include a detailed description of the setup and usage of the API. Finally, we highlight the limitations of the current version together with the new features we are implementing to improve the tool.

The content adaptation pipeline has been described in section 5, presenting the content adaptation and optimization services, with a focus on codec selection for MediaVerse and how to optimize the content provided by the creators in order to serve the users with the quality level customized to their hardware and network. This set of services will communicate with the DAM to generate several optimized copies of the original content.

Section 6 on the Next Generation Content Model creates and manages data information about the assets (videos, images, audios, 360 content, 3D content), this data will be called metadata. Among different types of metadata, there is technical metadata, such as resolution, codecs, bitrate, HDR specifications. Another type is the content metadata such as subtitles, audio description, or sign language, and finally, copyright metadata as the type of licenses, author, or collaborators. The section also links the different versions of metadata generated in the transcoding service (T3.3). The metadata will be useful to feed and train the recommendations service (T3.2). Besides, the section aims to describe the different standard formats adopted in MediaVerse.

1 Introduction

The main objective of this deliverable named D3.1 “Next Generation Content Model and Algorithms for New Media Types” is to provide the first MediaVerse “Next Generation Content Management, Understanding and Interlinking layer” architecture.

WP3, following the requirements gathered in WP2, designs and develops a media content management architecture for the ingestion, automatic adaptation and delivery of different types of multimedia content (pictures, audio, flat video, immersive 360 videos and 3D models). As already stated in deliverable D2.2 “Conceptual Design of the MediaVerse Framework”, the MediaVerse ecosystem is a federation of MediaVerse nodes. Each MediaVerse node is organized as a set of distributed micro-services. Figure 1 presents the MediaVerse WP3 structure highlighting main functionalities, the activity it is going to provide and how it is structured.

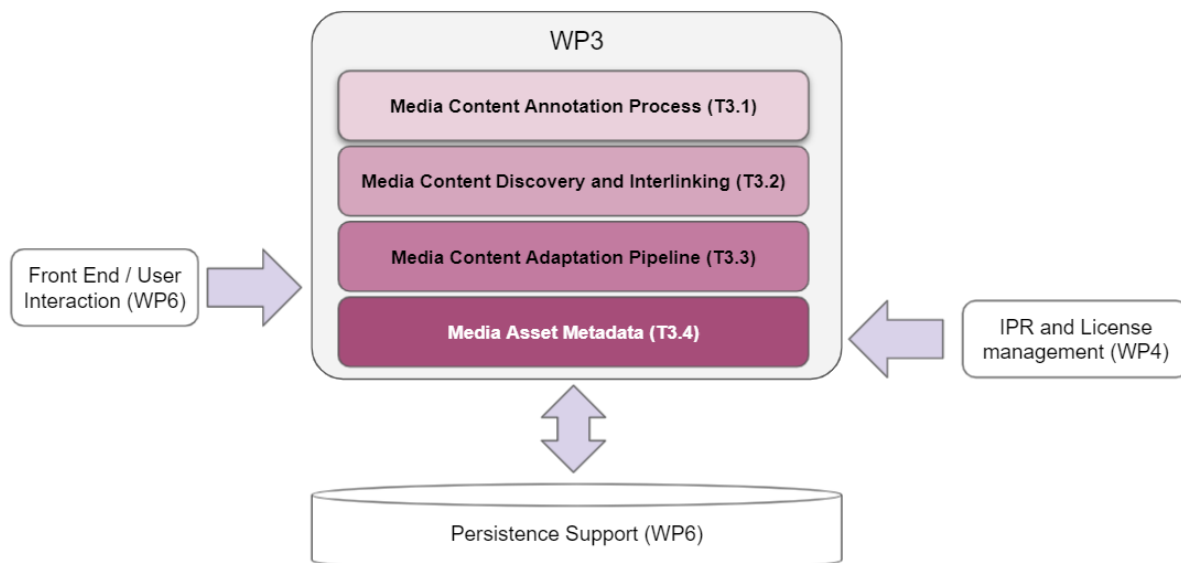


Figure 1: MediaVerse WP3 structure in Tasks

Figure 2 provides an overview of the conceptual architecture of WP3 providing a finer grained view of the WP3 modules with respect to the three considered pipelines: a) media content ingestion pipeline, b) media content management and adaptation pipeline and c) media content delivery pipeline.

In the next sections of this document, the reader will find a detailed analysis of the WP3 modules.

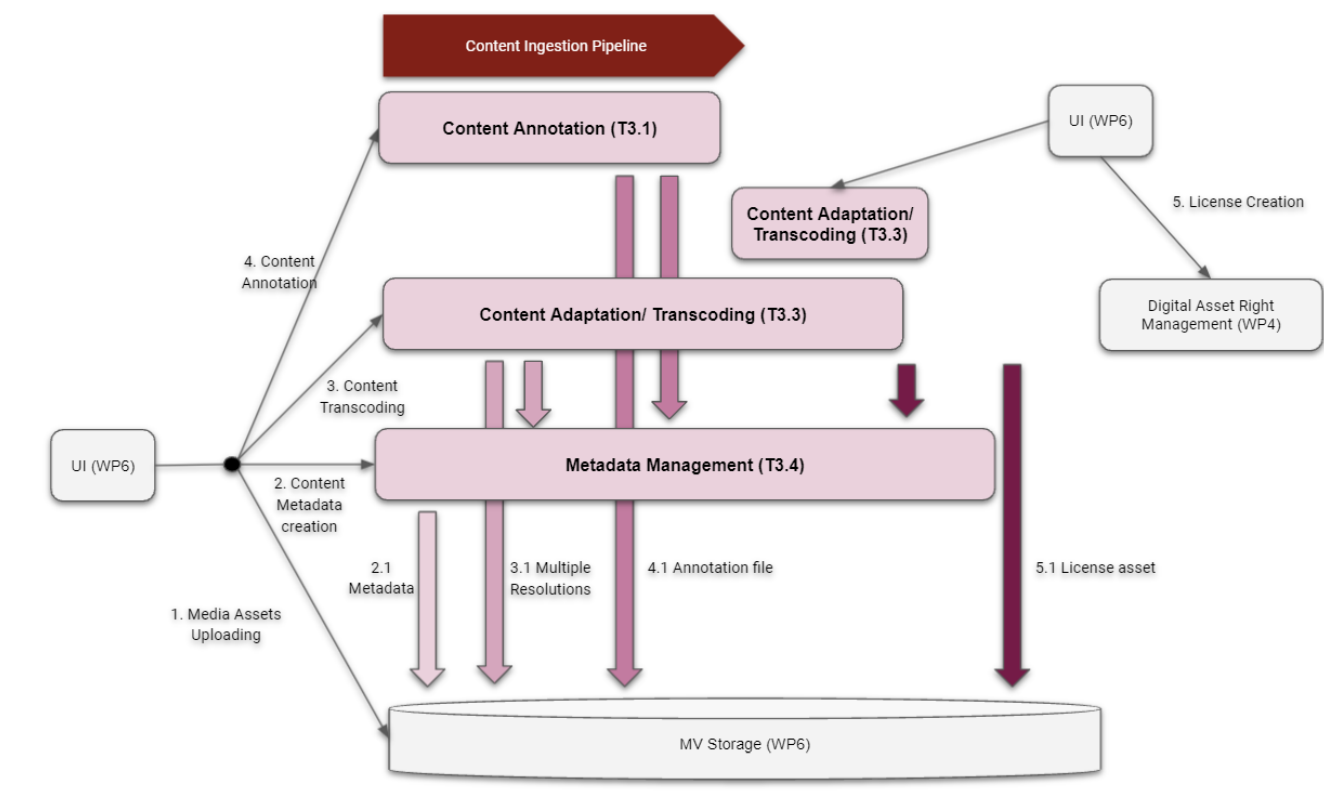


Figure 2: MediaVerse Content Ingestion, Adaptation and Delivery Pipeline

1.1 Purpose and structure of the document

Deliverable D3.1 provides the first version of the architecture of the MediaVerse “Next Generation Content Management, Understanding and Interlinking” layer. This is the first of three deliverables WP3 will deliver during its lifecycle. The deliverable D3.2 “Content Discovery and Recommendation, Annotation and Adaptation Framework” released on project month 21 and the deliverable D3.3 “Interactive New Media Annotation Module and Content Discovery and Recommendation Framework v2” released on project month 30 will release the second and the final version of the technologies implemented in the context of this WP3 activity.

The document is structured as follows. **Section 2** overviews the work carried out in Task T3.1 related to automatic understanding and annotation of digital assets stored in MediaVerse nodes, such as images, videos, 3D and 360 content. **Section 3** extends the work done in T3.1 proposing a multilingual annotation tool called Transfer to be applied on social media posts. **Section 4** introduces the Content Discovery and Interlinking services as part of the Task T3.2. **Section 5** outlines the work carried out in Task T3.3 related to the development of the project multimedia content adaptation pipeline (pictures, audio, flat video, and 360 videos). **Section 6** presents the MediaVerse Data Model for content description as part of the Task T3.4.

2 New media Understanding and Annotation

This section describes the work in task T3.1 related to automatic understanding and annotation of digital assets stored in MediaVerse nodes, such as images, videos, 3D and 360 content. Also, it provides detailed information on the API we build to facilitate other partners to use our models.

2.1 Usage Scenarios

D3.1 aims to address certain user requirements defined in deliverable D2.1. More specifically, requirements ANNO-01, SRCH-01, SRCH-07, MDRT-01 which are shortly presented in Table 1, are directly or implicitly addressed by task T3.1. The focus of the task is on automatically understanding several aspects of the new media content that is uploaded to each MV node and then on providing useful tags to annotate the content. This consequently aims to facilitate the content discovery process across the whole MV network.

Table 1: List of relevant user requirements

| REQUIREMENT ID | SHORT DESCRIPTION | COMMENTS |
|----------------|---|---|
| ANNO-01 | The system will add tags to each new asset | Directly related to T3.1 |
| SRCH-01 | Content can be searched across nodes | Largely related to T6.2 but it requires the outcomes of T3.1 and T3.2 |
| SRCH-07 | Search results can be ordered/ranked according to certain aspects | Also related to T3.2 |
| MDRT-01 | New incoming content should be tested wrt whether it is appropriate | Directly related to T5.1 and indirectly to T3.1 |

Mainly, here we construct a basis for addressing the content annotation requirements and more precisely we make the first steps by focusing on ANNO-01. These first steps include 1-vs-all content classification, detection of objects, automatic free-text description of the content as well as face recognition for celebrities like politicians, athletes and artists. All these are performed on images for now but we also provide a plan for our next steps on the processing and annotation of videos. We also build and provide to the consortium API endpoints which deploy the corresponding models for content annotation that will be certainly useful to the work of T3.2 as a necessary input for the content discovery process. The provided models are the best based on our up to now experimentation, however changing the models of this component with other more effective ones during the project or even later is feasible from a technical perspective as elaborated in section 3.3.

Moreover, we investigate options for cross-modal retrieval with the work on shared embedding space between text and images which is different from standard content annotation yet might be useful for providing better recommendations and searching results. With this work we contribute to requirements SRCH-01 and SRCH-07 by annotating the content with a vector of real numbers that can then be used for cross-modal similarity estimation. MDRT-01 is also implicitly facilitated by the work conducted in this deliverable and more specifically from the work on meme detection. After the pre-classification of an image as an image meme it can be further analysed (this is worked to be conducted in WP5/T5.1 for content moderation) in order to reveal its nature which potentially could be offensive or hateful and thus should comply with the moderation rules of the user and the corresponding MV node.

Similar to the image data, task T3.1 will contribute to the research and implementation of the content annotation task regarding 3D and 360 data. So far, the work is only related to 3D, and it can be mostly used on ANNO-01. For example, when a MV's user creates a VR scene in the MV platform, tags will be generated and added for that scene. Later on, these tags can be used to fulfil some of the remaining requirements.

Finally, we should mention that although the already existing state of the art pre-trained deep learning models have shown impressive performance in the field of content understanding, there is a lot of room for research in order to extract more useful tags in a context such as that of the MediaVerse platform. As we describe in this blog post¹, existing state of the art solutions perform great on detecting more general concepts encountered in an image but lack the ability to extract more abstract information (i.e., humour, irony) or to detect more specific objects (i.e., celebrities).

2.2 Methods and Materials

Here, we present literature review and experiments conducted in the framework of task T3.1. We split our work in two main pillars, namely analysis on image data and analysis on 3-dimensional data. In Table 2, we summarize the individual components built for this task that are elaborated in detail below.

Table 2: Components of T3.1 annotation module. The 1st column presents the component's name, the 2nd column presents the best method based on our up to now experimentation, the 3rd informs about the status of experimentation and the 4th presents integration

| COMPONENT | CURRENT METHOD | EXPERIMENTATION STATUS | INTEGRATION STATUS |
|---|--|--|--|
| IMAGE CONTENT | | | |
| image classification and object detection | Hybrid pipeline involving Faster RCNN Inception v2 (Ren et al, 2016) for object detection and EfficientNetB7 (Tan & Le, 2019) for image classification | final for first version | only the object detection part is integrated in Media Annotation API |
| image captioning | Show, Attend and Tell (Xu et al, 2015) | can be improved | integrated in Media Annotation API |
| image meme detection | Ours, involving fine-tuning of Inception v3 on custom dataset | can be further evaluated and potentially be improved | integrated in Media Annotation API |
| cross-modal retrieval (image-text similarity) | CLIP (Radford et al, 2021) | final for first version | not integrated |
| celebrity recognition in images | MTCNN (Zhang et al., 2016) for face detection and | final for first version | integrated in Media Annotation API |

¹ <https://mediaverse-project.eu/2021/03/04/new-media-analysis-and-tagging/>

| COMPONENT | CURRENT METHOD | EXPERIMENTATION STATUS | INTEGRATION STATUS |
|--|---|---|--------------------|
| | VGGFace2 (Cao et al., 2018) for celebrity recognition | | |
| VIDEO CONTENT | | | |
| video annotation (classification, object detection, captioning) | n/a | literature review | not integrated |
| IMMERSIVE CONTENT | | | |
| 3D annotation (Semantic Segmentation on 3D scenes) | BAAF-Net (Qiu et al. 2021) | can be further evaluated in more datasets | not integrated |
| 360° annotation (360 image classification, viewpoint prediction) | n/a | literature review | not integrated |

2.2.1 Image data

Image classification and detection of objects in images

State of the art

As a starting point for our media annotation service, we will rely on widely used computer vision pre-trained models. There are two main directions to produce annotations, namely image classification and object detection in images.

Over the years, image classification progresses and advances with multiple research teams working on this field and providing new models surpassing in rapid succession the state of the art. Today, most studies consider widely-used pre-trained architectures including ResNet (He et al, 2015) which utilizes residual connections in order to effectively train deeper networks, Inception (Szegedy et al, 2015) which processes visual information at various scales in each convolutional building block, Inception-ResNet (Szegedy et al, 2016) which combines the benefits of the previous two network architectures and EfficientNet (Tan & Le, 2019) that scales depth, width and resolution of the network using a compound coefficient.

Similarly, in object detection today's most utilized methods are Faster R-CNN (Ren et al, 2016) which proposes Region Proposal Networks (RPN) making the training and inference processes even faster and the whole architecture unified, You Only Look Once (YOLO) model family (Redmon et al, 2015; Redmon & Farhadi, 2016; Wang et al, 2020) which considers less complex architectures being as a consequence less effective but a lot faster than R-CNN enabling for real time object detection and EfficientDet (Tan et al, 2020) which holds the

current state of the art and makes use of bi-directional feature pyramid networks (BiFPN) and compound scaling of depth, width and resolution of the whole architecture.

Widely-used benchmark datasets in these two fields are Microsoft’s Common Objects in Context (MSCOCO)² including ~328K images annotated among others with objects and the corresponding bounding boxes, ImageNet³ that contains ~14M annotated images with one of 1,000 classes according to WordNet, Canadian Institute for Advanced Research (CIFAR-100)⁴ that contains 60,000 32x32 color images labeled with one of 100 classes and one of 20 superclasses and MNIST which contains 70,000 images of hand-written digits.

Experiments

Here, we present inference results based on widely used pre-trained models, on six example images randomly obtained from Google, to explore potential options for the automatic tagging task of T3.1. The concepts that we can provide as tags by using these models are the 90 object classes of MSCOCO dataset⁵ and the 1000 classes of ImageNet⁶. We explore three approaches: (1) image classification using ResNet50, Inception ResNet v2 and EfficientNet-B7 pre-trained on ImageNet, (2) object detection in images using Faster R-CNN Inception v2 and EfficientDetD7 pre-trained on MSCOCO dataset and (3) a hybrid approach in which we first detect the objects in the image along with their corresponding classes (with Faster R-CNN Inception v2 and EfficientDetD7) and then pass the cropped parts of the image (the predicted bounding boxes) through a classifier (Inception ResNet v2 and EfficientNet-B7) to obtain more informative tags. No further training for fine-tuning the models has been conducted. In Table 3, we illustrate the results of the classification approach, in Table 4 we illustrate the results of the object detection and in Table 5, we present the results of the hybrid approach.

The pre-trained models were chosen based on two criteria: (1) they provide a good trade-off between computational efficiency and accuracy and (2) providing state of the art results on the aforementioned benchmarks, as we can see in Table 5⁷. It seems that the image classification approach in many cases fails to provide useful tags when there exists more than one object in the image. However, EfficientNetB7 gives more relevant responses as expected. For instance, although it is wrong, the prediction for image number 4 to be a tractor is very plausible. The object detection approach focuses on the relevant parts of the content and provides very informative tags as well. And also, the addition of a classifier on top of the object detector provides even more informative tags. For instance, in the first row of Table 5 the hybrid approach informs us that some of the persons detected by the object detectors are wearing suits, loafers and ties.

² <https://cocodataset.org/>

³ <https://www.image-net.org/>

⁴ <https://www.cs.toronto.edu/~kriz/cifar.html>

⁵ https://github.com/tensorflow/models/blob/master/research/object_detection/data/mscoco_label_map.pbtxt

⁶ <https://gist.github.com/aaronpolhamus/964a4411c0906315deb9f4a3723aac57>

⁷ The figures are obtained from Keras applications (<https://keras.io/api/applications/>) and TensorFlow object detection model zoo (https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md and https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md) respectively.

Table 3: Image classification inference results using ResNet50 and Inception ResNet v2 pre-trained on ImageNet







| IMAGE | TOP-5 RESNET50 PREDICTIONS | TOP-5 INCEPTION RESNET V2 PREDICTIONS | TOP-5 EFFICIENTNETB7 PREDICTIONS |
|---|--|--|---|
|  | groom: 53.2% bassoon: 11.7% prison: 3.6% kimono: 2.1% torch: 1.9% | pay-phone: 5.2% suit: 4.6% accordion: 4.3% cellular_telephone: 3.7% Loafer: 3.7% | suit: 67.0% Windsor_tie: 7.6% groom: 3.0% library: 2.0% Loafer: 1.4% |
|  | coho: 71.3% gar: 21.0% sturgeon: 3.8% barracouta: 3.8% great_white_shark: 0.1% | barracouta: 60.5% coho: 17.0% reel: 8.4% gar: 3.6% sturgeon: 1.5% | barracouta: 46.2% gar: 26.4% coho: 2.6% tench: 1.7% sturgeon: 1.3% |
|  | African_elephant: 45.8% tusker: 42.5% Indian_elephant: 11.7% triceratops: 0.0% Arabian_camel: 0.0% | African_elephant: 47.4% tusker: 32.3% Indian_elephant: 10.5% triceratops: 0.0% dragonfly: 0.0% | African_elephant: 40.9% tusker: 23.0% Indian_elephant: 16.8% triceratops: 0.1% zebra: 0.1% |
|  | forklift: 10.4% tractor: 7.3% electric_fan: 7.3% turnstile: 5.6% stove: 3.8% | turnstile: 60.1% seat_belt: 3.7% minibus: 1.9% forklift: 1.6% limousine: 1.5% | tractor: 54.9% forklift: 13.8% harvester: 12.0% thresher: 0.8% seat_belt: 0.7% |
|  | brown_bear: 91.6% American_black_bear: 4.8% bison: 2.7% hyena: 0.4% wild_boar: 0.1% | brown_bear: 92.0% American_black_bear: 0.4% sloth_bear: 0.2% hyena: 0.1% bison: 0.1% | brown_bear: 77.3% American_black_bear: 2.2% sloth_bear: 0.3% bison: 0.2% wild_boar: 0.1% |
|  | passenger_car: 76.7% electric_locomotive: 13.7% streetcar: 9.4% bullet_train: 0.1% freight_car: 0.0% | passenger_car: 65.0% streetcar: 15.1% electric_locomotive: 7.6% bullet_train: 1.4% freight_car: 0.8% | passenger_car: 58.7% streetcar: 8.1% electric_locomotive: 6.3% bullet_train: 0.9% freight_car: 0.7% |

Table 4: Object detection inference results using Faster R-CNN Inception v2 and EfficientDetD7 pre-trained on MSCOCO

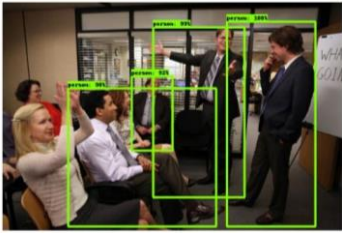
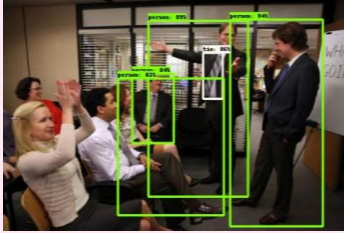


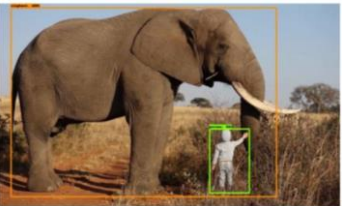
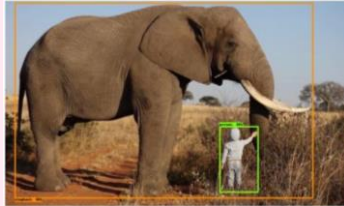


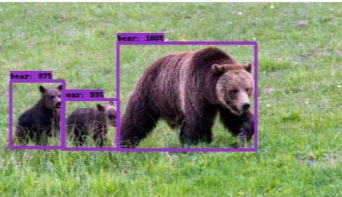
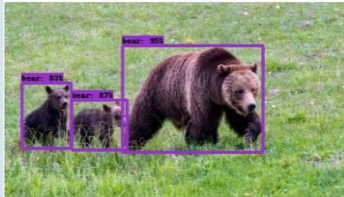


| DETECTIONS WITH FASTER R-CNN INCEPTION V2 | PREDICTIONS WITH FASTER R-CNN INCEPTION V2 | DETECTIONS WITH EFFICIENTDET7 | PREDICTIONS WITH EFFICIENTDET7 |
|---|---|--|--|
|  | <p>person: 99.71%</p> <p>person: 99.12%</p> <p>person: 97.70%</p> <p>person: 91.94%</p> |  | <p>person: 93.68%</p> <p>person: 89.44%</p> <p>tie: 86.11%</p> <p>person: 83.82%</p> <p>person: 81.83%</p> |
|  | <p>person: 99.76%</p> <p>surfboard: 91.15%</p> |  | <p>person: 95.01%</p> |
|  | <p>elephant: 99.98%</p> <p>person: 99.29%</p> |  | <p>elephant: 98.29%</p> <p>person: 87.90%</p> |
|  | <p>person: 99.09%</p> <p>tie: 87.86%</p> |  | <p>person: 90.68%</p> |
|  | <p>bear: 99.98%</p> <p>bear: 98.76%</p> <p>bear: 97.16%</p> |  | <p>bear: 95.11%</p> <p>bear: 86.60%</p> <p>bear: 83.38%</p> |
|  | <p>train: 99.92%</p> |  | <p>train: 93.26%</p> |

Table 5: Inference results from the hybrid approach using Faster R-CNN Inception v2 and EfficientDetD7 for object detection and Inception ResNet v2 and EfficientNetB7 for classification. *Italics denote the tags obtained from the 2nd stage of the hybrid approach*





| INPUT IMAGE | HYBRID: FASTER R-CNN INCEPTION v2 + INCEPTION RESNET v2 | HYBRID: FASTER R-CNN INCEPTION v2 + EFFICIENTNETB7 | HYBRID: EFFICIENTDET7 + INCEPTION RESNET v2 | HYBRID: EFFICIENTDET7 + EFFICIENTNETB7 |
|---|---|---|--|--|
|  | person: 99.71% person: 99.12% person: 97.70% person: 91.94% <i>suit: 14.5%</i> <i>suit: 58.6%</i> <i>Loafer: 62.3%</i> <i>Windsor_tie: 40.8%</i> | person: 99.71% person: 99.12% person: 97.70% person: 91.94% <i>suit: 35.7%</i> <i>suit: 47.7%</i> <i>suit: 24.2%</i> <i>Windsor_tie: 47.0%</i> | person: 93.7% person: 89.4% tie: 86.1% person: 83.8% person: 81.8% <i>cellular_telephone: 12.2%</i> <i>suit: 33.3%</i> <i>Windsor_tie: 87.7%</i> <i>Windsor_tie: 54.0%</i> <i>Loafer: 85.9%</i> | person: 93.7% person: 89.4% tie: 86.1% person: 83.8% person: 81.8% <i>suit: 33.4%</i> <i>suit: 41.7%</i> <i>Windsor_tie: 35.9%</i> <i>Windsor_tie: 59.0%</i> <i>suit: 31.9%</i> |
|  | person: 99.76% surfboard: 91.15% <i>coho: 69.6%</i> <i>barracouta: 55.1%</i> | person: 99.76% surfboard: 91.15% <i>barracouta: 46.8%</i> <i>electric_ray: 22.0%</i> | person: 95.0% <i>coho: 79.9%</i> | person: 95.0% <i>barracouta: 48.2%</i> |
|  | elephant: 99.98% person: 99.29% <i>African_elephant: 62.2%</i> <i>curly-coated_retriever: 33.9%</i> | elephant: 99.98% person: 99.29% <i>African_elephant: 40.9%</i> <i>bow: 7.6%</i> | elephant: 98.3% person: 87.9% <i>African_elephant: 58.8%</i> <i>curly-coated_retriever: 55.7%</i> | elephant: 98.3% person: 87.9% <i>African_elephant: 42.5%</i> <i>ski_mask: 7.7%</i> |
|  | person: 99.09% tie: 87.86% <i>golfcart: 30.0%</i> <i>Windsor_tie: 61.1%</i> | person: 99.09% tie: 87.86% <i>minibus: 15.7%</i> <i>Windsor_tie: 40.4%</i> | person: 90.7% <i>golfcart: 21.8%</i> | person: 90.7% <i>minibus: 7.3%</i> |
|  | bear: 99.98% bear: 98.76% bear: 97.16% <i>brown_bear: 92.2%</i> <i>brown_bear: 75.8%</i> <i>brown_bear: 82.8%</i> | bear: 99.98% bear: 98.76% bear: 97.16% <i>brown_bear: 82.9%</i> <i>brown_bear: 65.3%</i> <i>brown_bear: 62.0%</i> | bear: 95.1% bear: 86.6% bear: 83.4% <i>brown_bear: 90.9%</i> <i>brown_bear: 78.2%</i> <i>brown_bear: 81.9%</i> | bear: 95.1% bear: 86.6% bear: 83.4% <i>brown_bear: 82.5%</i> <i>brown_bear: 69.9%</i> <i>brown_bear: 66.4%</i> |
|  | train: 99.92% <i>streetcar: 83.9%</i> | train: 99.92% <i>streetcar: 31.8%</i> | train: 93.3% <i>streetcar: 76.7%</i> | train: 93.3% <i>passenger_car: 40.8%</i> |

Table 6: Computational efficiency and accuracy of the chosen pre-trained models for image classification and object detection tasks

| CLASSIFICATION MODEL | SIZE | TOP-5 ACCURACY (IMAGNET) |
|--------------------------|--------|--------------------------|
| ResNet50 | 98MB | 0.921 |
| Inception ResNet v2 | 215MB | 0.953 |
| EfficientNetB7 | 256 MB | 0.971 |
| OBJECT DETECTION MODEL | SPEED | mAP (COCO) |
| Faster RCNN Inception v2 | 58 ms | 0.28 |
| EfficientDetD7 | 325 ms | 0.51 |

Image captioning

State of the art

Another approach to annotate images is to automatically produce captions that describe the content of the image. During the last decade, many studies have been conducted on the automatic generation of text for image description (Bernardi et al., 2017). Mainly, these studies utilize a first encoding stage in which informative features are extracted from the image by a pre-trained CNN and a second decoding stage in which the caption generation process is implemented. Vinyals et al. (2015), consider the Inception network with batch normalization as encoder CNN and the Long Short-Term Memory (LSTM) recurrent neural network as decoder, while Xu et al. (2015) add the well-known Bahdanau attention mechanism (Bahdanau et al, 2015) in order to force the decoder to focus only on the relevant part of the image when predicting each word of the generated caption. Many other studies have proposed different attention mechanisms, for instance (Song et al., 2019) who propose a mechanism that progressively modulates the attention weights by a language grounding module, (Wang et al, 2019) that leverage object and text representations on top of the usual encoding CNN vectors and a hierarchy feature pyramid of attention mechanisms, and (Guo et al, 2020a) that consider a Transformer-like architecture which takes as input object representations obtained from Faster R-CNN and applies self-attention in the encoding phase and both self-attention and cross-attention in the decoding phase. Additionally, vision-language pre-training over huge datasets with the aim to improve performance on downstream tasks, such as image captioning, has attracted research interest lately (Li et al, 2020; Jia et al, 2021).

Commonly used datasets in image captioning are Flickr8K (Hodosh et al, 2013) that contains 8,000 images each paired with five different captions, Flickr30K (Young et al, 2014) that contains 31,000 images collected from Flickr together with 5 captions provided by human annotators and MSCOCO (Lin et al, 2014) that contains 123,287 images annotated with 5 reference sentences each.

Experiments

Here, we consider encoder-decoder neural network architectures for image captioning. In the encoding phase, the model processes the image with the aim to generate a semantically meaningful vector representation usually through pre-trained convolutional neural networks. We opted for VGG16 pre-trained on ImageNet in this first approach, whose second to last layer produces a 4096-element vector. Then the decoder is a recurrent neural network (we used LSTM) that produces each word of the caption based on all previously predicted words and the vector representation. The selection of words is done through the final softmax layer that acts on the

vocabulary of all words seen in the training phase. In order to train the model to produce the correct phrases we minimize the categorical cross entropy loss function. In Figure 3, we illustrate the architecture that we used for the baseline image captioning model.

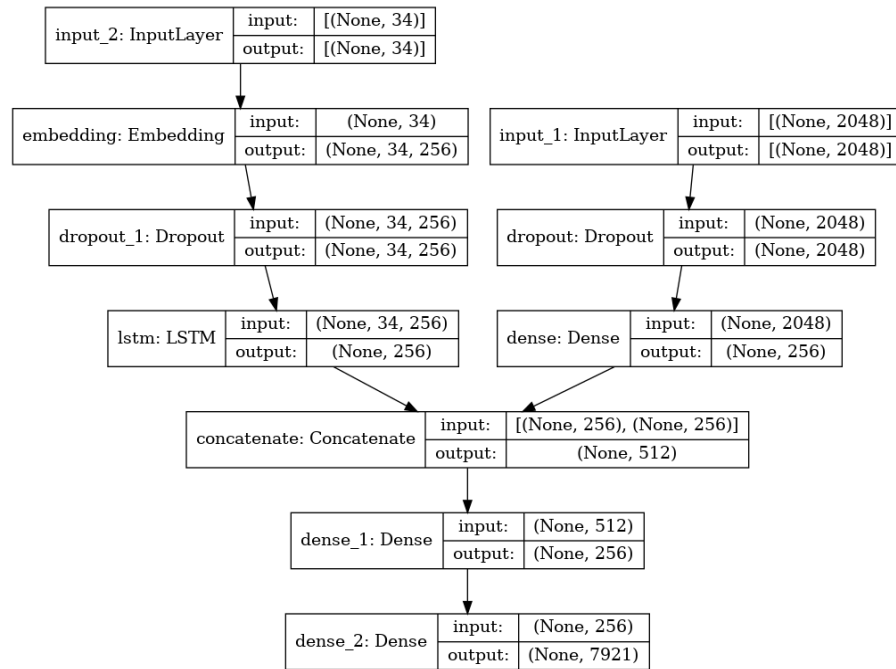


Figure 3: Baseline image captioning model architecture (keras model)

We have trained the baseline architecture on the Flickr8K dataset⁸ which contains ~8000 images along with multiple captions per image. The training, validation and test sets are randomly formed containing 80%, 10% and 10% of the images respectively. The performance of our caption generating model is evaluated through BLEU scores (Papineni et al, 2002) on the test set resulting in BLEU-4 0.156, namely low performance.

Our first attempt with the baseline architecture and the small-scale training dataset provides results of rather low quality. In order to improve the results, we further explored options including the following:

- better vector representations: investigate other models
- better decoder: include attention mechanism
- consider bigger and more representative datasets

In detail, we try a) vector representations produced by Inception ResNet v2 and b) vector representations produced by NASNetLarge, which are the top two architectures by Keras Applications in terms of top-1 accuracy on ImageNet, c) vector representations produced by Inception v3, as another well-known good option for feature extraction, d) the “Show, Attend and tell” encoder-decoder with attention architecture, e) Flickr30K as training dataset and f) MSCOCO as training dataset. We adopt these options in an ablation study, to assess their boosting potential in the model’s performance. Table 7 presents the results: the attention-based architecture combined with a relatively big training dataset produces the best results among the investigated options. Finally, we generate captions using the best model based on our up to now experimentation and the baseline model for 6 example novel images as we did in the previous paragraph and present them in Table 8. The best results are qualitatively better than the ones’ provided by the baseline but there is still room for improvement.

⁸ <https://www.kaggle.com/adityajn105/flickr8k/activity>

Table 7: BLEU scores for multiple alterations performed on the base model termed here LSTM. With bold we denote best performance

| ENCODER [ImageNet] | DECODER | DATASET | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 |
|---------------------|------------------|-----------|--------------|--------------|--------------|--------------|
| VGG16 | LSTM | Flickr8K | 0.567 | 0.360 | 0.275 | 0.155 |
| Inception ResNet v2 | LSTM | Flickr8K | 0.550 | 0.350 | 0.267 | 0.145 |
| NASNetLarge | LSTM | Flickr8K | 0.598 | 0.367 | 0.270 | 0.145 |
| Inception v3 | LSTM | Flickr8K | 0.607 | 0.384 | 0.291 | 0.163 |
| Inception v3 | Show/Attend/Tell | Flickr8K | 0.600 | 0.388 | 0.296 | 0.171 |
| Inception v3 | LSTM | Flickr30K | 0.575 | 0.336 | 0.243 | 0.125 |
| Inception v3 | Show/Attend/Tell | Flickr30K | 0.629 | 0.395 | 0.291 | 0.159 |
| Inception v3 | LSTM | COCO | 0.651 | 0.424 | 0.319 | 0.185 |
| Inception v3 | Show/Attend/Tell | COCO | 0.682 | 0.463 | 0.352 | 0.211 |

Table 8: Captions generated by the baseline and the best model among the investigated options, for 6 example novel images

| | |
|---|---|
|  | <p>Baseline: Two people are standing in front of the other of the other people</p> <p>Best: Group of people sitting on couch with wii</p> |
|  | <p>Baseline: Man in blue shirt is holding his arms</p> <p>Best: The man is standing on the ground</p> |
|  | <p>Baseline: Two people are standing on the grass</p> <p>Best: An elephant standing in the dirt next to an elephant</p> |
|  | <p>Baseline: Man in blue shirt is sitting on the top of the side of the side of the camera</p> <p>Best: Man in suit and tie standing next to window</p> |
|  | <p>Baseline: Dog running in the grass</p> <p>Best: Brown bear walking through the grass in the grass</p> |
|  | <p>Baseline: People walking on subway station</p> <p>Best: Train is sitting on the tracks</p> |

*Discriminating image memes from regular images***State of the art**

Internet image meme is a special type of image that pairs a text with a representative image and is used to express specific concepts or emotions like humour, irony, sarcasm and frustratingly hate. Its niche applicability also entails a specialized treatment, different from the one on regular images. Thus, the automatic detection of internet image memes is deemed important and also a precedent methodological step (pre-classification step). However, the topic of image meme detection has not yet received considerable attention. After literature investigation we found only one dataset for meme detection, namely the DankMemes⁹, which was released in 2020 and is currently publicly unavailable. It contains 2000 images (which are split in 80%-20% for training and testing), half of which are memes and the rest are regular images. Also, some scientific papers are published that use this dataset to address the meme detection problem in the framework of the corresponding competition (Fiorucci, 2020; Vlad et al, 2020; Setpal & Sarti, 2020). The rather small size of this dataset as well as its current unavailability forced us to investigate options for the creation of a suitable dataset in order to address the problem of meme detection. Finally, some other attempts exist on the Internet^{10,11}, but are not peer reviewed or published. Just for clarity reasons, we should mention that our work here focuses on internet image memes and not internet memes in general which might be text for example as the topic of (He et al, 2019).

Experiments

The objective of this task is to create a classifier able to separate regular images and image memes like the ones shown in Figure 4. This means that we need to train a model to perform binary classification on the two aforementioned classes.

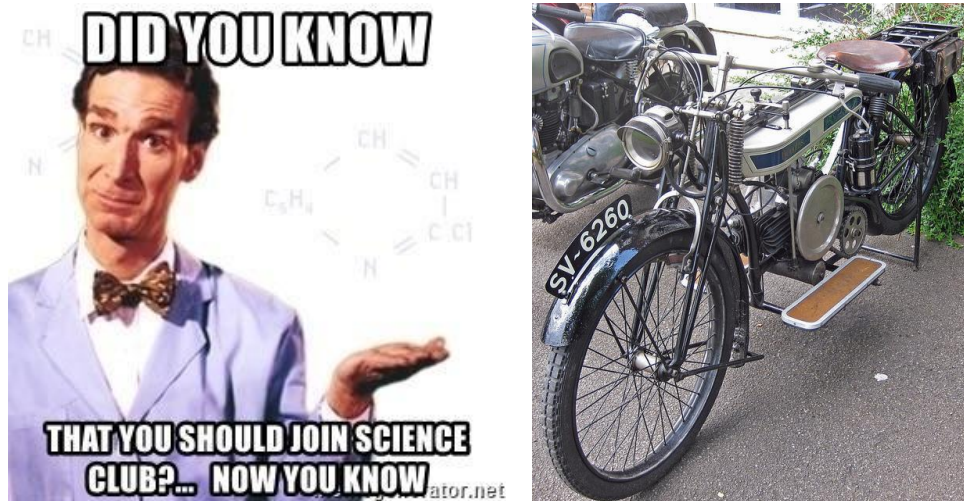


Figure 4: (Left) image meme, (right) regular image.

To this end we downloaded the Meme Generator dataset¹², that contains image memes and we also used regular images from the MSCOCO dataset¹³, already obtained for the image captioning task. In Table 9, we illustrate the utilized dataset's counts with regards to classes and training, validation, test separation.

⁹ <https://dankmemes2020.fileli.unipi.it/>

¹⁰ <https://github.com/matyasbohacek/meme-detection>

¹¹ <https://medium.datadriveninvestor.com/memes-detection-android-app-using-deep-learning-d2c65347e6f3>

¹² <https://www.kaggle.com/electron0zero/memegenerator-dataset>

¹³ <https://cocodataset.org/>

Table 9: Counts of classes and training/validation/test sets (1st experiment)

| CLASS/SET | TRAINING (85%) | VALIDATION (5%) | TEST (10%) | TOTAL |
|-------------------------------|----------------|-----------------|------------|---------|
| Meme [MemeGen] (50%) | 47,992 | 2,864 | 5,730 | 56,586 |
| Regular [MSCOCO] (50%) | 48,204 | 2,794 | 5,588 | 56,586 |
| total | 96,196 | 5,658 | 11,318 | 113,172 |

For our first experiment, we consider the widely-used VGG16 deep neural network pre-trained on ImageNet as a base model. The last fully connected layer is discarded and replaced with one dense layer of one unit with sigmoid activation. Binary cross-entropy is used as a cost function and adam is used as an optimizer. The batch size is set to 64 and the model was trained for up to 30 epochs with early stopping (patience 2) and checkpoint (minimum validation loss) callbacks. The results are summarized in Table 10.

Table 10: Performance of VGG16 on training, validation and test sets in terms of log-loss and accuracy (1st experiment)

| SET | LOG-LOSS | ACCURACY |
|------------|----------|----------|
| training | 0.00418 | 99.90% |
| validation | 0.00402 | 99.91% |
| test | 0.00292 | 99.92% |

It seems that the model's performance is great on this dataset, yet it is crucial to perform an extra check on novel images cherry-picked from the web or elsewhere. Indeed, we consider 12 image memes downloaded from Pinterest, 12 regular images randomly selected from the MSCOCO validation set (unseen by the model) and another 12 regular images randomly selected from Google's Conceptual Captions dataset¹⁴. You can find these novel images in Figure 5. Then we load the previously trained model, produce predictions on these 36 images and present the summarized results in Table 11.

Table 11: Performance of VGG16 on novel images (1st experiment)

| CLASS | CORRECT | WRONG | TOTAL |
|-------------------------------|---------|-------|-------|
| Meme [Pinterest] | 8 | 4 | 12 |
| Regular [MSCOCO] | 12 | 0 | 12 |
| Regular [Concep. Cap.] | 9 | 3 | 12 |
| total | 29 | 7 | 36 |

The results on novel image memes are not encouraging as only 8 out of 12 images are correctly identified as memes while regular images from the MSCOCO dataset are all classified correctly. The latter is expected as these images are very carefully selected by the dataset creators, with fair resolution and of similar nature with training set's images. But, when we test the model's performance on 12 regular images from the noisy Conceptual Captions dataset the results are still good but not as good as the initial ones, with only 9 correct predictions out of 12 total guesses. In order to obtain insights with regards to mitigation actions, we plot the wrongly classified images. Here, in Figure 5, we exemplify 4 of them, two memes predicted as regular and two regular predicted as memes.

¹⁴ <https://ai.google.com/research/ConceptualCaptions/>



Figure 5: Misclassified images (1st experiment)

Through visual inspection, we found that the wrongly classified memes are of different nature from the ones contained in the MemeGenerator dataset, most of which are similar to the ones presented in Figure 5. Hence, one mitigation action would be to enlarge the dataset with more memes from different sources. Regarding regular images, one possible solution would be to add images from the Conceptual Captions dataset to the training set, as they are more representative of the online content than the MSCOCO ones. For the former we will use the Reddit Memes dataset¹⁵ (which however requires cleaning as many of the files are irrelevant) and for the latter we will enrich the training data by replacing half of the regular images with images from the Conceptual Captions. Two consecutive experiments are performed adding gradually the new contents - first we add the MSCOCO images and then the Reddit Memes. The new data counts for the second experiment are given in Table 12, results are presented in Table 13 and model performance on the novel images is shown in Table 14.

¹⁵ <https://www.kaggle.com/sayangoswami/reddit-memes-dataset>



Figure 6: Examples of image memes from the MemeGenerator dataset

Table 12: Counts of classes and training/validation/test sets (2nd experiment)

| CLASS/SET | TRAINING (85%) | VALIDATION (5%) | TEST (10%) | TOTAL |
|-------------------------------------|----------------|-----------------|------------|---------|
| Meme [MemeGenerator] (50%) | 47,984 | 2,871 | 5,731 | 56,586 |
| Regular [MSCOCO] (25%) | 24,124 | 1,390 | 2,779 | 28,293 |
| Regular [Concep. Cap.] (25%) | 24,088 | 1,397 | 2,808 | 28,293 |
| total | 96,196 | 5,658 | 11,318 | 113,172 |

Table 13: Performance of VGG16 on training, validation and test sets in terms of log-loss and accuracy (2nd experiment)

| SET | LOG-LOSS | ACCURACY |
|------------|----------|----------|
| training | 0.01405 | 99.77% |
| validation | 0.01143 | 99.78% |
| test | 0.01216 | 99.74% |

Table 14: Performance of VGG16 on novel images (2nd experiment)

| CLASS | CORRECT | WRONG | TOTAL |
|-------------------------------|---------|-------|-------|
| Meme [Pinterest] | 1 | 11 | 12 |
| Regular [MSCOCO] | 12 | 0 | 12 |
| Regular [Concep. Cap.] | 12 | 0 | 12 |
| total | 25 | 11 | 36 |

The results of the second experiment show great performance of the model on the utilized dataset itself in terms of cost function and test accuracy. Also, great performance is exhibited on new regular images. However, the model could not recognize image memes performing very poorly on the overall meme detection task. Thus, we proceed with the third experiment in which the dataset will be enriched with more image memes from the Reddit Memes dataset. This dataset initially contains 3,321 images but after visual inspection we realized that many of them are just funny pictures, screenshots of Twitter posts, corrupted files or just irrelevant content, which should be discarded in order not to confuse the model. After manual cleansing 1,995 images are left for training, validation and testing. In Table 15, the new data counts for the 3rd experiment are presented, in Table 16 the results are presented and in Table 17 the model's performance on the novel images set is illustrated.

Table 15: Counts of classes and training/validation/test sets (3rd experiment)

| CLASS/SET | TRAINING (85%) | VALIDATION (5%) | TEST (10%) | TOTAL |
|-------------------------------------|----------------|-----------------|------------|---------|
| Meme [MemeGen] (48.3%) | 48,008 | 2,828 | 5,750 | 56,586 |
| Meme [Reddit] (1.7%) | 1,691 | 103 | 201 | 1,995 |
| Regular [MSCOCO] (25%) | 24,978 | 1,427 | 2,885 | 29,290 |
| Regular [Concep. Cap.] (25%) | 24,909 | 1,500 | 2,881 | 29,290 |
| total | 99,586 | 5,858 | 11,717 | 117,161 |

Table 16: Performance of VGG16 on training, validation and test sets in terms of log-loss and accuracy (3rd experiment)

| SET | LOG-LOSS | ACCURACY |
|------------|----------|----------|
| training | 0.04375 | 98.89% |
| validation | 0.04234 | 98.83% |
| test | 0.05473 | 98.79% |

Table 17: Performance of VGG16 on novel images (3rd experiment)

| CLASS | CORRECT | WRONG | TOTAL |
|-------------------------------|---------|-------|-------|
| Meme [Pinterest] | 10 | 2 | 12 |
| Regular [MSCOCO] | 12 | 0 | 12 |
| Regular [Concep. Cap.] | 12 | 0 | 12 |
| total | 34 | 2 | 36 |

It seems that the model is performing very well, so in the test set as in the novel images set, after introducing Reddit Memes and Conceptual captions examples to it. The two misclassified novel image memes are presented in Figure 7, whereas we see they are not the usual image memes. On the contrary, all the image memes of usual appearance (you can find them in Figure 8) are correctly classified.



Figure 7: Misclassified image memes (3rd experiment)

Moreover, now that we settled the training dataset, we can experiment more on the feature extraction phase and test more contemporary and state-of-the-art deep neural networks. In Table 18, we present a comparative study on meme detection with regards to different models trained on the dataset as presented in Figure 8.

Table 18: Model performance on meme detection task in terms of log-loss and accuracy both on the test set and on the novel images set

| MODEL | LOG-LOSS (TEST) | ACCURACY (TEST) | LOG-LOSS (NOVEL) | ACCURACY (NOVEL) |
|--------------|-----------------|-----------------|------------------|------------------|
| VGG16 | 0.05473 | 98.79% | 0.27181 | 94.44% |
| ResNet50 | 0.03265 | 99.12% | 0.39792 | 91.66% |
| Inception V3 | 0.01540 | 99.56% | 0.13104 | 91.66% |

We observe that on the test set of the utilized dataset Inception V3 performs best but on the novel images set VGG16, a rather shallow model, exhibits superior performance.

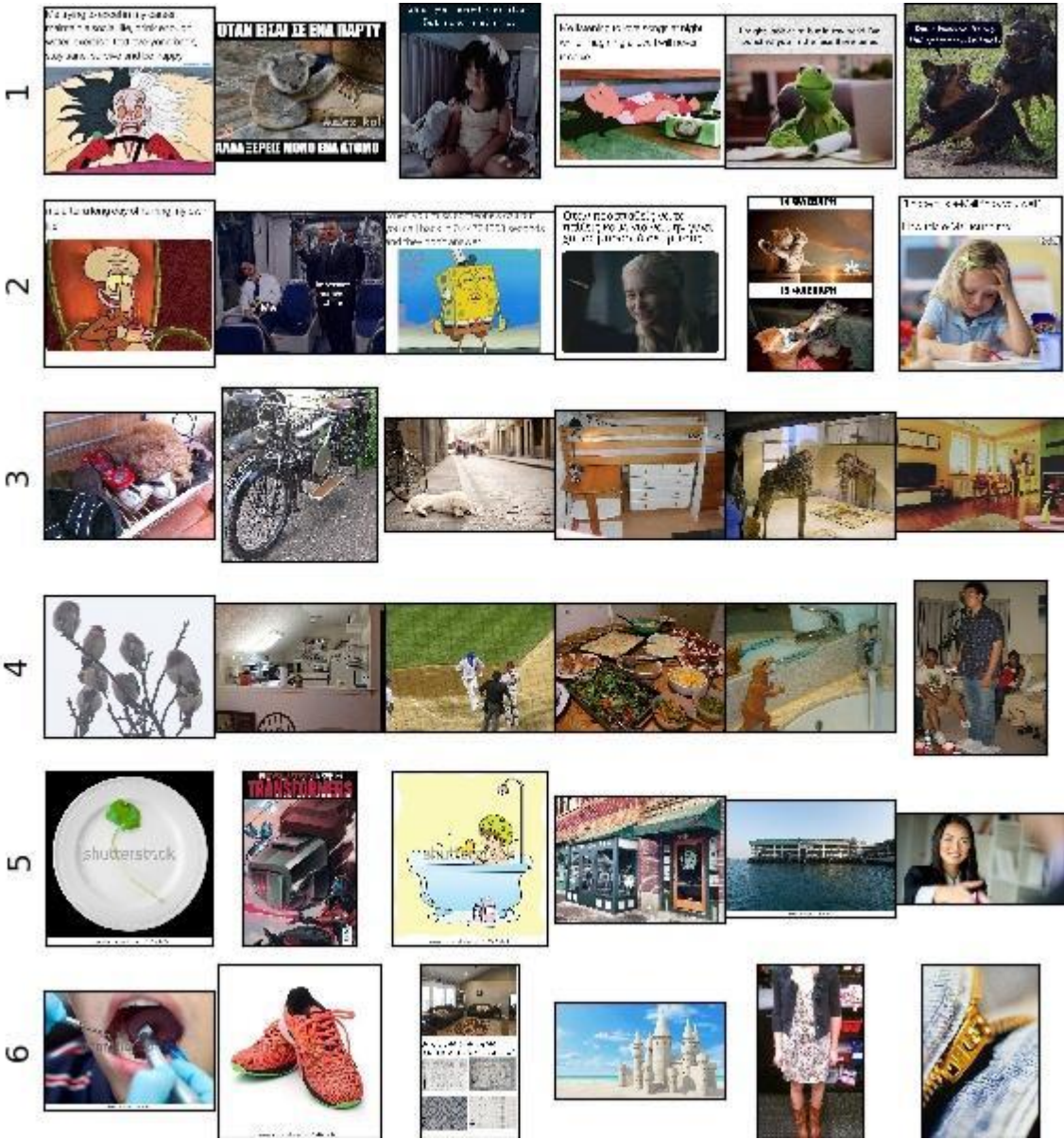


Figure 8: All images of the “novel images set”. The first two rows contain novel image memes, while the rest contain novel regular images

*Shared vector space between images and text for cross-modal retrieval***State of the art**

Cross-modal retrieval has attracted great research interest lately. Chen et al (2020a) review the related literature and classify the models for this problem in four main categories, namely pairwise, adversarial, interaction and attributes learning models. In more general terms, we can also separate the related work in 1) methods that use large-scale visual-language pre-training of huge BERT-based models (the current state of the art results in image-text retrieval) and 2) methods focusing on methodological improvements on neural network architectures that are trained on the dataset of interest, for example the MSCOCO, Flickr8K or Flickr30K.

More precisely, Lu et al (2019) extend the BERT model by adding also region features from an object detection model as input and a co-attention mechanism to introduce interactions in some layers. Similarly, Su et al (2020) propose a unified BERT model with input region features from Faster RCNN along with word features pre-trained with masked words and masked ROIs. Chen et al (2020b) consider four pre-training tasks: masked language, masked region, image text matching and word region alignment. They mask one modality at a time unlike other previous studies and also, they train with respect to one task per minibatch. Li et al (2020) utilize a BERT model with triplet input (sequence and object labels, region representations from faster RCNN) and two combined loss functions: masked token loss and contrastive loss. Finally, Radford et al (2021) and Jia et al (2021) (who provide the current state of the art results) follow a similar approach considering simple text and image encoders, yet of huge capacity, and the same contrastive learning framework. The former is trained on 400M image text pairs, while the latter on 1.8B noisy image text pairs from the web. All the aforementioned models are pre-trained on large-scale datasets composed of millions or billions of pairs and are then fine-tuned and evaluated on downstream understanding and generation tasks, one of which is image-text retrieval.

Regarding the second class of models, initial attempts include (Kiros et al, 2015) in which an encoder-decoder RNN network is trained to predict the surrounding sentences in a text document. In such a way they create generic and useful representations for text sentences. Then they use these representations for cross-modal retrieval among other tasks, with non state of the art results but very close to at the time of publication. More recently, in (Lee et al, 2018) a cross attention mechanism is proposed for cross modal retrieval. Also, Wei et al (2020) and Zhang et al (2020) introduce attention mechanisms that take into account self- and cross-dependencies between the two modalities with comparable to state-of-the-art results. The former utilizes multi-head attention mechanisms on top of BERT word features and image features extracted by object detection models. In (Guo et al, 2020b), the authors consider a LSTM text encoder and a LSTM image encoder with attention on the feature map of image patches (created by VGG-19), with the number of time steps for image encoding being a hyperparameter. Finally, a graph-based approach is presented in (Diao et al, 2021) where self-attention with average vector query is employed for final feature vector computation on both modalities. Global (between the final feature vectors) and local (between each word feature and the corresponding context vector from region features) similarity vectors between image and text are extracted and the two modules, (1) similarity graph reasoning and (2) similarity attention filtration compute the final similarity vectors which pass through fully connected layers to compute similarity score between image and text.

Experiments

The objective here is to build a model that simultaneously encodes images and text with the aim to embed them in a new shared vector space, in which the positive image/text pairs have similar vectors while the negative image/text pairs have dissimilar vectors. This will provide vector annotations to images uploaded to MV

instances, which then can be retrieved based on the similarity with the search query. In Figure 9, we exemplify one image with the corresponding text description from the MSCOCO dataset.



Figure 9: Example image from the MSCOCO dataset. Target description: “black honda motorcycle parked in front of garage”

In order to achieve this, we first utilize the MSCOCO dataset that contains 82,783 images for training and 40,504 images for validation. Each image is annotated with at least 5 free text descriptions, like the one presented in Figure 9, summing up to 616,767 image/text positive pairs. For our purposes we do not keep the pre-defined split in training and validation sets as we want to leverage as much information as possible. Thus, we split this corpus as illustrated in Table 19.

Table 19: Positive image/text pairs of MSCOCO dataset in training and validation sets

| | TRAINING (95%) | VALIDATION (5%) | TOTAL |
|-------------------------|----------------|-----------------|---------|
| Image/text pairs MSCOCO | 585,927 | 30,840 | 616,767 |

The image/text pairs should now be used to train a dual encoder that aligns the produced vector representations. To this end, we consider a variation of the recently proposed from OpenAI, Contrastive Language–Image Pre-training model (CLIP)¹⁶ (Radford et al, 2021), which originally was created for zero-shot classification scenarios. Actually, we consider the contrastive learning framework proposed there and the vision/text encoders proposed for the ALIGN model (Jia et al, 2021) (EfficientNet (Tan & Le, 2019) and BERT (Devlin et al, 2019) respectively), and repurpose it for cross-modal retrieval. In Figure 10, we present a high-level graphical representation of the part of the CLIP model (taken from openai.com) that we consider for image/text alignment.

¹⁶ <https://openai.com/blog/clip/>

1. Contrastive pre-training

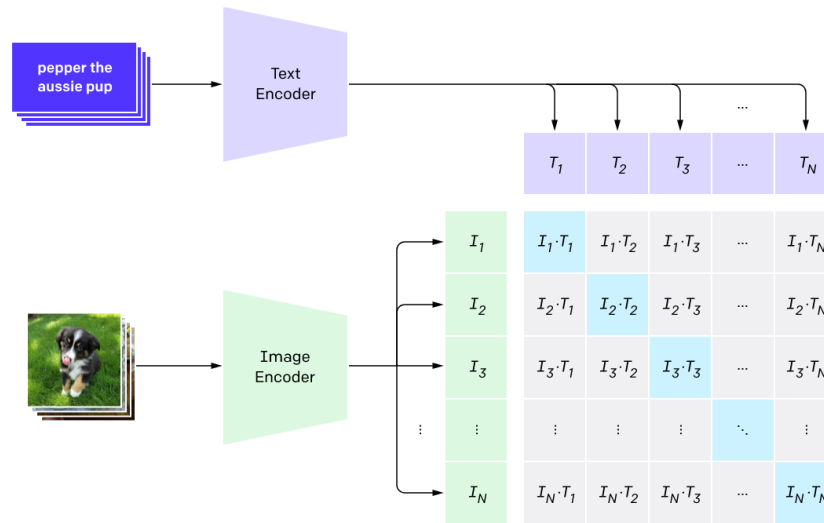


Figure 10: The contrastive learning framework. A batch of image/text pairs is passed through CLIP which is optimized in order to maximize the similarity of the positive pairs and minimize the similarity of all other pairs considered here as negative.

In our first experiment, the text encoder is a rather small version of pre-trained BERT (Devlin et al, 2019) with $L=4$ transformer stacks, $H=512$ embedding dimension and $A=8$ heads of attention, and the image encoder is EfficientNetB3 (Tan & Le, 2019) pre-trained on ImageNet, both without trainable weights to obtain the pre-trained vector representations.

On top of each encoder, we add 4 fully-connected trainable layers of size 256 and optimize the model with adamW (learning rate $1e-3$, batch size 64) to minimize the categorical cross-entropy loss on both axes of the table produced in Figure 8. Then, in order to assess the efficacy of the produced representations we apply recall at k ($R@k$) metrics on the validation set (6,165 images, 30,840 captions) and present the results in the two columns of Table 20 for text-to-image and image-to-text retrieval respectively. Additionally, for quality assessment of the retrieved content we produced examples with random queries and the corresponding 9 most similar images in the validation set (unseen texts, unseen images), in Figure 11, we present one of them. Despite the bad results in terms of $recall@k$, most of the retrieved images are very relevant to the query except the ones from the “police car” query. We investigated further this example and found that the word “police” is present 65 times in captions of this corpus and the word “cop” 14 times. After visual inspection of all the images of the validation set whose description contains either “police” or “cop” it can be concluded that it might be a hard task for a machine to discriminate between police and regular bike riders, horse riders, groups of people etc with very few examples.

Table 20: Evaluation metrics for text to image (T2I) and image to text (I2T) retrieval on MSCOCO validation set

| | T2I | I2T |
|--------|--------|--------|
| R@1 | 0.0282 | 0.0330 |
| R@5 | 0.1046 | 0.1166 |
| R@10 | 0.1727 | 0.1930 |
| R@50 | 0.4720 | 0.4929 |
| R@100 | 0.6649 | 0.6507 |
| R@200 | 0.8293 | 0.7954 |
| R@500 | 0.9413 | 0.9265 |
| R@1000 | 0.9776 | 0.9732 |

query: football



Figure 11: Retrieved images for query “football”

Another more formal check we did is to retrieve images using words of average frequency in the validation set as queries. We opted for average frequency words so the queries are neither very general nor very niche concepts. The average word frequency in the validation set is 41.73 thus we selected words with frequency $\text{avg} \pm 2$, namely the 50 words illustrated in Table 21. It seems that in many cases like beds, dock and bathtub the model retrieves relevant content while also there are queries that are not related to the retrieved content such as boarder, couches and desert.

Table 21: Words of average frequency in the validation set of MSCOCO dataset

| | | | | |
|------------------|-----------------|-------------------|--------------------|--------------------|
| ('desert', 43) | ('neck', 42) | ('suitcases', 43) | ('assortment', 41) | ('serve', 40) |
| ('bushes', 40) | ('guys', 40) | ('she', 42) | ('go', 41) | ('vintage', 41) |
| ('sticking', 41) | ('docked', 41) | ('fly', 40) | ('nearby', 41) | ('toothbrush', 40) |
| ('trail', 41) | ('couches', 43) | ('sinks', 42) | ('stall', 43) | ('including', 43) |
| ('shore', 42) | ('fish', 42) | ('facing', 40) | ('equipment', 40) | ('style', 41) |
| ('bathtub', 40) | ('broken', 43) | ('adults', 40) | ('tomatoes', 42) | ('plant', 42) |
| ('produce', 41) | ('church', 43) | ('beds', 40) | ('shop', 42) | ('boarder', 41) |
| ('same', 41) | ('sides', 41) | ('woods', 43) | ('french', 40) | ('reaching', 41) |
| ('curb', 43) | ('boxes', 40) | ('both', 42) | ('cluttered', 40) | ('bowls', 43) |
| ('bicycles', 43) | ('wire', 40) | ('shorts', 43) | ('well', 41) | ('dock', 43) |

To further investigate improvement options, we conducted a series of experiments with unfreezed weights of both encoders and parameter tweaks. The corresponding comparative results are presented in Table 22 and it seems that the performance is stuck way below state of the art as Jia et al (2021) present in Table 9 therein, where the same models are trained from scratch on Conceptual Captions dataset of ~3M image/text pairs. We rely on pre-trained models hence we get similar recall@k, otherwise we believe it would be even worse. Hence, it is concluded that to achieve state of the art retrieval in the MediaVerse network we should currently rely on publicly available pre-trained models like CLIP or Oscar (Li et al, 2020). To build a model that surpasses current state of the art or at least perform such as other non pre-trained models we should investigate methodological improvements in the modelling part, as the computational cost of large-scale vision-language pre-training cannot be afforded by our equipment.

Table 22: Comparative study for image and text retrieval on MSCOCO validation set

| MODEL/METRIC | IMAGE RETRIEVAL | | | |
|--|-----------------|--------|--------|--------|
| | R@1 | R@5 | R@10 | R@50 |
| freezed EfficientNetB3, freezed BERT (L=4, H=512, A=8), 4x256 projection layers, learning rate 1e-3, batch size 64 (initial), dropout 0.0 | 0.0282 | 0.1046 | 0.1727 | 0.4720 |
| EfficientNetB2 (20 unfreezed layers), BERT (L=2, H=128, A=2) unfreezed, 1x256 projection layers, learning rate 5e-4, batch size 128, dropout 0.3 | 0.1572 | 0.4084 | 0.5496 | 0.8560 |
| EfficientNetB3 (40 unfreezed layers), BERT (L=4, H=512, A=8) unfreezed, 1x256 projection layers, learning rate 5e-5, batch size 64, dropout 0.3 | 0.1626 | 0.4187 | 0.5644 | 0.8780 |
| EfficientNetB4 (80 unfreezed layers), BERT (L=4, H=512, A=8) unfreezed, 1x256 projection layers, learning rate 5e-5, batch size 64, dropout 0.3 | 0.1606 | 0.4122 | 0.5578 | 0.8783 |
| EfficientNetB4 (40 unfreezed layers), BERT (L=12, H=768, A=12) unfreezed, 1x256 projection layers, learning rate 5e-5, batch size 32, dropout 0.3 | 0.1775 | 0.4402 | 0.5834 | 0.8890 |
| EfficientNetB4 (100 unfreezed layers), BERT (L=12, H=768, A=12/3) unfreezed, 1x256 projection layers, learning rate 5e-5, batch size 32, dropout 0.3 | 0.1793 | 0.4417 | 0.5888 | 0.8915 |

| | | | | |
|---|----------------|---------------|---------------|---------------|
| EfficientNetB5 (60 unfreezed layers), BERT (L=12, H=768, A=12/3) unfreezed, 1x256 projection layers, learning rate 5e-5, batch size 32, dropout 0.3 | 0.1845 | 0.4506 | 0.5959 | 0.8967 |
| EfficientNetB5 (120 unfreezed layers), BERT (L=12, H=768, A=12/3) unfreezed, 1x512 projection layers, learning rate 1e-5, batch size 32, dropout 0.5 | 0.1490 | 0.3908 | 0.5354 | 0.8644 |
| EfficientNetB5 (120 unfreezed layers), BERT (L=12, H=768, A=12/3) unfreezed, 1x1024 projection layers, learning rate 1e-5, batch size 16, dropout 0.5 | 0.1475 | 0.3829 | 0.5262 | 0.8617 |
| | TEXT RETRIEVAL | | | |
| frozen EfficientNetB3, frozen BERT (L=4, H=512, A=8), 4x256 projection layers, learning rate 1e-3, batch size 64 (initial), dropout 0.0 | 0.0330 | 0.1166 | 0.1930 | 0.4929 |
| EfficientNetB2 (20 unfreezed layers), BERT (L=2, H=128, A=2) unfreezed, 1x256 projection layers, learning rate 5e-4, batch size 128, dropout 0.3 | 0.2068 | 0.4731 | 0.6061 | 0.8778 |
| EfficientNetB3 (40 unfreezed layers), BERT (L=4, H=512, A=8) unfreezed, 1x256 projection layers, learning rate 5e-5, batch size 64, dropout 0.3 | 0.2077 | 0.4674 | 0.6111 | 0.8884 |
| EfficientNetB4 (80 unfreezed layers), BERT (L=4, H=512, A=8) unfreezed, 1x256 projection layers, learning rate 5e-5, batch size 64, dropout 0.3 | 0.2123 | 0.4661 | 0.6047 | 0.8836 |
| EfficientNetB4 (40 unfreezed layers), BERT (L=12, H=768, A=12) unfreezed, 1x256 projection layers, learning rate 5e-5, batch size 32, dropout 0.3 | 0.2306 | 0.5057 | 0.6428 | 0.9042 |
| EfficientNetB4 (100 unfreezed layers), BERT (L=12, H=768, A=12/3) unfreezed, 1x256 projection layers, learning rate 5e-5, batch size 32, dropout 0.3 | 0.2262 | 0.4974 | 0.6387 | 0.9062 |
| EfficientNetB5 (60 unfreezed layers), BERT (L=12, H=768, A=12/3) unfreezed, 1x256 projection layers, learning rate 5e-5, batch size 32, dropout 0.3 | 0.2446 | 0.5265 | 0.6652 | 0.9153 |
| EfficientNetB5 (120 unfreezed layers), BERT (L=12, H=768, A=12/3) unfreezed, 1x512 projection layers, learning rate 1e-5, batch size 32, dropout 0.5 | 0.1865 | 0.4319 | 0.5755 | 0.8793 |
| EfficientNetB5 (120 unfreezed layers), BERT (L=12, H=768, A=12/3) unfreezed, 1x1024 projection layers, learning rate 1e-5, batch size 16, dropout 0.5 | 0.1842 | 0.4184 | 0.5542 | 0.8608 |

The best performing dual encoder in terms of recall@k on MSCOCO validation set, is EfficientNetB5 (60 unfreezed layers), BERT (L=12, H=768, A=12/3) unfreezed, 1x256 projection layers, learning rate 5e-5, batch size 32, dropout 0.3. This model exhibits an improvement compared to the initial one, with respect to the evaluation metric recall@k (for all k=1, 5, 10, 50) but it is still far from the current state-of-the-art in image-text retrieval briefly presented in the next paragraph. We also investigated retrieved images using the current best model given the same random queries and the same average frequency queries, as in the initial analysis. It seems that the retrieved images are still very relevant and, in some cases, even more relevant (like “car”, “police car”, “green field”, “people on the beach”, “fruit”, “vegetables” in random queries and “bathtub”, “beds”, “bicycles”, “boarder”, “bowls”, “couches”, “curb”, “equipment”, “neck”, “she”, “toothbrush”, “vintage”, “wire” in average frequency queries). Finally, we also train and evaluate the best performing model so far on a comparable split of

MSCOCO, namely we use the Karpathy split¹⁷ of 113,287 training, 5,000 validation and 5,000 test images which is common in most studies e.g. (Wei et al, 2020) and the corresponding results are: $R@1=0.189$, $R@5=0.459$, $R@10=0.597$ for image retrieval and $R@1=0.235$, $R@5=0.527$, $R@10=0.666$ for text retrieval.

As an additional step to our analysis, we also retrieved images given image queries (20 images from the MSCOCO validation set are randomly selected as queries). In Figure 12, we present one example. The retrieved images are very relevant to the query images. This kind of retrieval could potentially be useful in many ways, one of which is retrieving image memes involving a certain person or theme. For instance, if someone wants to retrieve “Bernie Sanders” image memes, this can be done by using this module with an input image similar to Figure 13, and then filter the retrieved results with the Meme Detector’s output to only return image memes and not regular images.

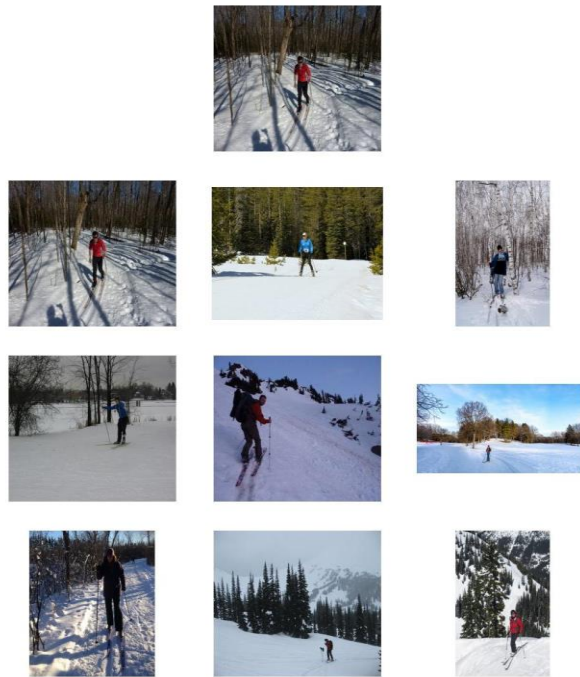


Figure 12: Image retrieval from image query (top image)



Figure 13: Image relevant to the “Bernie Sanders” meme. Source: dw.com

¹⁷ <https://www.kaggle.com/shtvkumar/karpathy-splits>

Pre-trained models

The pre-trained CLIP model is available online for inference and fine-tuning purposes¹⁸. However, the publicly available versions do not include the best performing ones as reported in the paper. Thus, we opt for the best among the available ones which is CLIP with ViT-B/32 vision encoder. We have downloaded and zero-shot evaluated this CLIP version on MSCOCO 5K test set giving R@1=**0.29**, R@5=**0.54** and R@10=**0.65** for image retrieval and R@1=**0.48**, R@5=**0.73** and R@10=**0.81** for text retrieval.

Other state of the art models like the Oscar pre-trained model which has been fine-tuned on the downstream task of image-text retrieval (MSCOCO captions dataset) are available online for inference purposes¹⁹. However, while the Oscar model outperforms CLIP on MSCOCO captions, the latter one is potentially a better choice for our annotation API as it has been trained on 400M image text pairs meaning that it recognizes many more than the MSCOCO concepts in visual and textual content and probably better fits MediaVerse's needs.

Multi-head cross-modal attention

We have also made attempts for methodological improvements of existing models in the same context, using transformer-like self-attention and cross-attention mechanisms on top of visual and textual encoders, either pre-trained or trained from scratch together with the attention parts. Results on the MSCOCO 5K test set (Karpathy split) are presented in Table 23. It seems that with this setting we have not surpassed the state of the art but we are still exploring improvement options. However, we also came across (Wei et al, 2020) that implements a very similar approach, that provides state of the art results, rendering this work less likely to be adapted by the annotation service.

Table 23: Performance of variants of multi-head cross-modal attention method on the MSCOCO 5K test set

| encoder | TEXT RETRIEVAL | | | IMAGE RETRIEVAL | | |
|---|----------------|-------|-------|-----------------|-------|-------|
| | 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 |

Celebrity detection

State of the art

The application of deep learning and representation learning to the domain of facial recognition has been the driving force behind recent advances in the state-of-the-art. The most accurate techniques of today leverage datasets of increasing size in conjunction with convolutional neural networks. The standard pipeline for facial recognition has changed drastically over the past few years. We've seen a transition from hand engineered features to learned features, a transition from face-specific alignment to rough alignment and centering, and a transition from datasets with tens of thousands of images to datasets with hundreds of millions of images. The

¹⁸ <https://github.com/openai/CLIP>

¹⁹ <https://github.com/microsoft/Oscar>

first application of learned features in face recognition was done by (Taigmain et al., 2014) which was based on the work of (Le et al., 2012) training a sparse AutoEncoder for large scale feature learning. This pattern continued, where deep Face Recognition architectures have followed those of deep object classification and evolved from AlexNet to SENet as shown in . There is also research that utilizes modified Transformer architecture for Face Recognition tasks (Bhagavatula et al., 2017; Wu et al., 2017; Chen et al., 2016). In order to train those datasets, there is a need for millions of images (Schroff et al., 2015). Researchers tried to enrich the information for each subject by increasing the number of images while concurrently introducing multiple poses, ages and illumination to create an invariant representation for every identity (Cao et al., 2018), (Best-Rowden et al., 2014). There are companies that have succeeded in creating great datasets with many thousands up to millions of identities (Schroff et al., 2015; Zhou et al., 2015) but kept private. To combat that issue, the academic community makes an effort to design effective loss functions and adopts efficient architectures to make deep features more discriminative using relatively small training datasets (Deng et al., 2018; Liu et al., 2017).

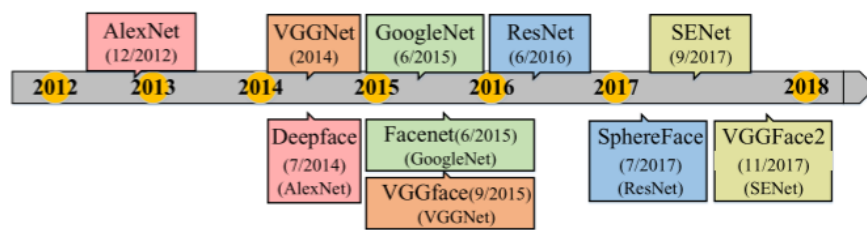


Figure 14: Deep Face recognition architecture evolution. On the top side are the architectures in object classification and on the bottom side the Face Recognition architectures in color correspondence (Wang et al., 2021)

Experiments

The main objective of this task is to recognize celebrity identities in an image. To achieve that we must train a classifier on a labelled dataset that contains images of the celebrities as inputs and their corresponding names as labels. The ideal dataset can provide useful facial features for training combined with a high number of images per subject. This can be done by gathering images with high variety in pose, age and illumination.

We were able to gather a list of the available datasets based on face recognition and compared them based on the above plus the availability of correspondence lists (image to identity), metadata, low error rate in labels as well as a manageable size to be able to train a model in reasonable time. Lastly, since MediaVerse is a European project, the ideal dataset would contain as many European celebrities as possible. Following the results in Table 24, the best candidates were Celeb-A, YouTube Faces Database and VGG Face2. They provide the best combination of average number of celebrities and size to examine if the training models generalize well. We are going to test each individual dataset with multiple architectures to examine the role of each attribute of the datasets. The other datasets were either too big to efficiently assess them on multiple architectures (MS-Celeb, UMD Faces) or they exhibit a very low number of images per subject (IMDB-WIKI, LFW).

CelebFaces Attributes Dataset (CelebA): It is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including - 10,177 number of identities, - 202,599 number of face images, and - 5 landmark locations, 40 binary attribute annotations per image. The images were obtained from the Internet and the authors chose to not reveal the real identities publicly, although they are available upon request.²⁰

²⁰ Celeb-A dataset is available at <https://drive.google.com/drive/folders/0B7EVK8r0v71pTUZsaXdaSnZBZzg>

YouTube Faces Database: Is a database of face videos designed for studying the problem of unconstrained face recognition in videos. The data set contains 3,425 videos of 1,595 different people. All the videos were downloaded from YouTube. An average of 2.15 videos are available for each subject. The shortest clip duration is 48 frames, the longest clip is 6,070 frames, and the average length of a video clip is 181.3 frames. For our analysis we used the aligned images and since the identities were available, we could produce country distributions. For the country information we used the wptools python package. The choropleth map and distribution plot are shown in Figure 15 and Figure 17.²¹ The majority of the subjects are American people (637), followed by English (101), Canadian (48) and Italian (31).

VGGFace2 (Cao et al., 2018): Consists of 3.31 million images of 9131 subjects, with an average of 362.6 images per subject. It includes human-verified bounding boxes around faces and five face landmarks. In addition, pose (yaw, pitch and roll) and apparent age information are estimated by pre-trained pose and age classifiers as opposed to ground truth annotations. Celebrity Images from the Internet, the images were downloaded from Google Image Search. We were able to get the birthplaces of 8045 celebrities. The choropleth map and distribution plot per country is shown in Figure 16 and Figure 18.²²

Table 24: Dataset research

| DATASET | NUMBER OF IDENTITIES | AVERAGE NUMBER OF IMAGES /SUBJECT | HIGH VARIETY IN POSE, AGE AND ILLUMINATION | CORRESPONDENCE LIST | METADATA | ERROR RATE | SIZE |
|-------------------------|----------------------|-----------------------------------|--|---------------------|------------|------------|-----------------|
| Celeb-A | 10177 | 19.90 | Yes | Yes | Yes | Low | 1.4 GB |
| MS-Celeb ²³ | 100K | 100 | Yes | Yes | No | Low | ~250 GB |
| VGG Face 2 | 9131 | 362.50 | Yes | Yes | Yes | Low | 40.25 GB |
| UMD Faces ²⁴ | 11377 | 44.44 | Yes | Yes | Yes | Low | 250 GB |
| YouTube Faces DB | 1595 | 389.42 | No | Yes | Yes | Low | 12.6GB |
| IMDB-WIKI ²⁵ | 100K | 5.23 | No | No | Yes | Low | 269 GB |
| LFW ²⁶ | 5749 | 2.26 | Yes | Yes | Yes | Avg | 173 MB |

²¹ YouTube Faces Database Dataset is available at <http://www.cslab.openu.ac.il/download/>

²² VGG Face 2 dataset is available at <https://academictorrents.com/details/535113b8395832f09121bc53ac85d7bc8ef6fa5b>

²³ <https://github.com/EB-Dodo/C-MS-Celeb>

²⁴ <http://umdfaces.io/>

²⁵ <https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/>

²⁶ <http://vis-www.cs.umass.edu/lfw/>

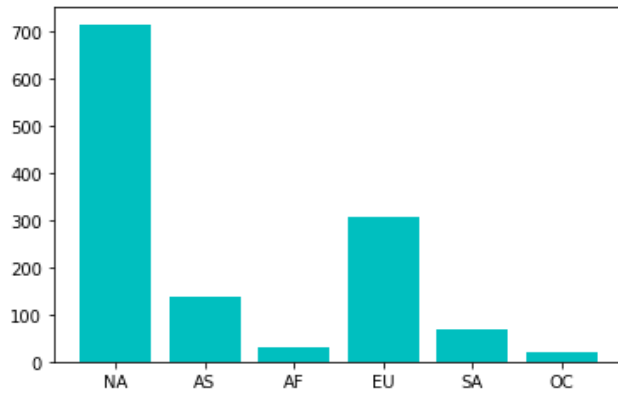
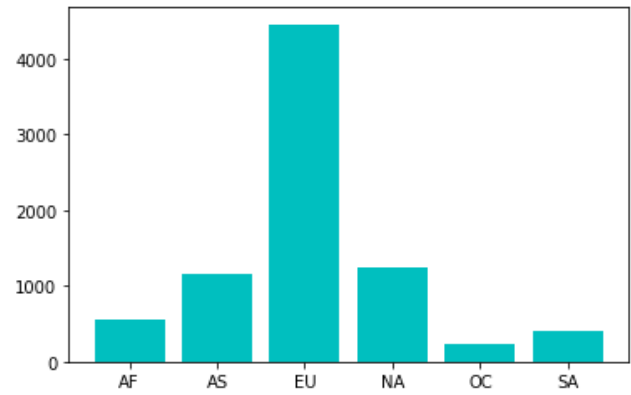
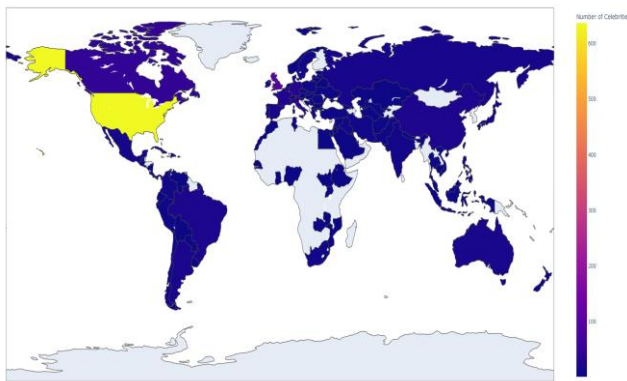
Figure 15: Distribution of continents for YouTube dataset²⁷Figure 16: Distribution of continents for YouTube dataset²⁸

Figure 17: Choropleth map for YouTube dataset

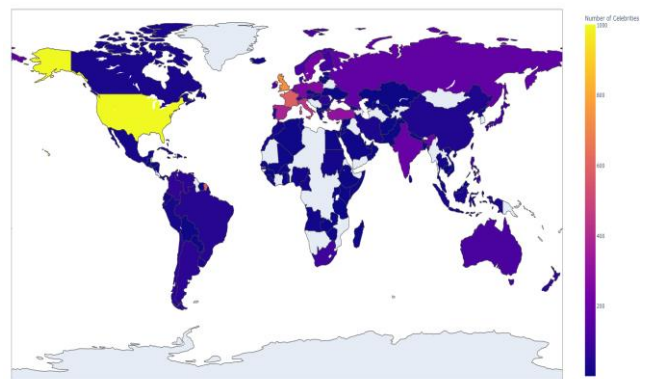


Figure 18: Choropleth map for VGG Face 2 dataset

Training

Given the available datasets, we need to use a good pre-trained network in order to decide which dataset is most suitable for the MediaVerse scope, which later will be used as a base to increase the number of identities. In general, the face recognition pipeline consists of four layers, loading, pre-processing, training and postprocessing. For dataset labelling we either use the folder structure or a correspondence file.

We ran several experiments using transfer learning. The models we used were MTCNN, FaceNet, Inception ResNet V2 and VGG Face 2 networks.

Multitask Cascade Convolutional Neural Network (MTCNN) (Zhang et al., 2016): Is primarily used for face detection. It uses a cascade structure with three networks; first the image is rescaled to a range of different sizes (called an image pyramid), then the first model (Proposal Network or P-Net) proposes candidate facial regions, the second model (Refine Network or R-Net) filters the bounding boxes, and the third model (Output Network or O-Net) proposes facial landmarks. The three models are not connected directly; instead, outputs of the previous stage are fed as input to the next stage. This allows additional processing to be performed between stages; for example, non-maximum suppression (NMS) is used to filter the candidate bounding boxes proposed by the first-stage P-Net prior to providing them to the second stage R-Net model.

²⁷ Where AF: Africa, AS: Asia, EU: Europe, NA: North America, OC: Oceania, SA: South America

²⁸ Where AF: Africa, AS: Asia, EU: Europe, NA: North America, OC: Oceania, SA: South America

FaceNet (Schroff et al., 2015): Is a face recognition system designed in 2015 by researchers at Google that achieved state-of-the-art results on a range of face recognition benchmark datasets. Given a picture of a face, the model will extract high-quality features from the face and predict a 128 element vector representation of these features. The model is a deep convolutional neural network trained via a triplet loss function that encourages vectors for the same identity to become more similar (smaller distance), whereas vectors for different identities are expected to become less similar (larger distance). The focus on training a model to create embeddings directly (rather than extracting them from an intermediate layer of a model) was an important innovation in this work.

Inception ResNet V2 (Szegedy et al., 2016): Is a hybrid network that combines the power of Inception networks with residual connections. This addition significantly improved face recognition performance and training times. They were introduced alongside Inception v4 and Inception ResNet V1. The main differences with the pure Inception networks are in the stem units and the pooling operation inside the main inception modules, which were replaced in favour of the residual connections. However, those operations are still in the reduction blocks.

VGG Face 2: Is essentially a ResNet-50 network with Squeeze and Excitations blocks. These blocks adaptively recalibrate channel-wise feature responses by explicitly modelling interdependencies between channels. They can be added to any network. The basis of the mechanism is getting a global understanding of each channel by squeezing the feature maps to a single numeric value. This results in a vector of size n , where n is equal to the number of convolutional channels. Afterwards, it is fed through a two-layer neural network, which outputs a vector of the same size. These n values can now be used as weights on the original features maps, scaling each channel based on its importance.

For the first batch of training only the YouTube and Celeb-A datasets were used. We splitted the dataset by subject, using an 80-10-10 split. Specifically, for the Celeb-A due to celebrities with a low number of available images, we chose to use only those with sixteen (16) images or higher for better representation per subject. The dataset statistics are shown in Table 25. Labels were one-hot encoded; we chose Adam as optimizer, Categorical Cross Entropy for loss, accuracy as the metric and the following grid search to find the optimal combination:

- Fine tune layers of the base model: [0, 10]
- learning rate: [1e-3, 1e-4]
- epochs: [10, 20, 30]
- batch size: [32, 64]

We used MTCNN for face detection on the YouTube dataset to extract the face from every image. The process was computationally intensive, taking over a week to complete on our lab server. The Celeb-A dataset was used as is, due to proper alignment done by the authors. Finally, the images were resized to the indicated size for each network and standardized or passed through a dedicated pre-process layer (i.e., for Inception ResNet V2). No data augmentation was used.

The results of the experiments were very poor using FaceNet as a base model on both Celeb A and YouTube datasets. Inception ResNet V2 exhibits great performance in the validation and test phases for the YouTube dataset (on average, loss < 0.1 and accuracy > 98%), however it appears that the network was overfitting as its performance on a small novel dataset with images gathered from the internet had very bad accuracy. This probably happened because the dataset has several images cropped from one to five videos for each celebrity, so the diversity of images is very limited and the training and test images are actually very similar. Also, the image quality was very low which might be another factor that caused the network to be unable to correctly recognize

new faces. For the same configuration on the Celeb-A dataset, the network could not be trained. We present a portion of the results on Table 26.

Table 25: Statistics for Celeb-A and YouTube datasets

| DATASET | MIN NUMBER OF IMAGES/CELEB | | | AVG NUMBER OF IMAGES/CELEB | | | MAX NUMBER OF IMAGES/CELEB | | | TOTAL NUMBER OF IMAGES | | |
|---------|----------------------------|------------|------|----------------------------|------------|-------|----------------------------|------------|------|------------------------|------------|-------|
| | train | validation | test | train | validation | test | train | validation | test | train | validation | test |
| Celeb-A | 16 | 2 | 3 | 21.31 | 2.44 | 3.18 | 28 | 3 | 4 | 113032 | 12698 | 16869 |
| YouTube | 34 | 1 | 1 | 311.53 | 38.94 | 38.94 | 4847 | 620 | 603 | 496900 | 62112 | 62114 |

Table 26: First training batch using pretrained networks

| DATASET | BASE MODEL | LOSS ON VALIDATION SET (LAST) | ACCURACY ON VALIDATION SET (LAST) | LOSS ON TEST SET | ACCURACY ON TEST SET |
|---------|---------------------|-------------------------------|-----------------------------------|------------------|----------------------|
| Celeb A | FaceNet | 3.7728 | 0.4226 | 3.7579 | 0.4226 |
| Celeb A | FaceNet | 3.2655 | 0.4568 | 3.3497 | 0.4524 |
| Celeb A | FaceNet | 3.3838 | 0.4511 | 3.3474 | 0.4558 |
| Celeb A | FaceNet | 4.2088 | 0.4010 | 4.1848 | 0.4035 |
| YouTube | FaceNet | 13.8962 | 0.0076 | 10.6628 | 0.0000 |
| Celeb A | FaceNet | 3.3744 | 0.4466 | 3.3682 | 0.4504 |
| Celeb A | FaceNet | 3.3847 | 0.4479 | 3.3726 | 0.4496 |
| YouTube | FaceNet | 12.2910 | 0.0000 | 10.4588 | 0.0000 |
| Celeb A | FaceNet | 3.3806 | 0.4441 | 3.3706 | 0.4502 |
| YouTube | Inception ResNet v2 | 0.0843 | 0.9887 | 0.0828 | 0.9896 |

For the second batch we did transfer learning on VGGFace2 using Celeb-A and VGGFace2 datasets. For the Celeb-A dataset we chose to use VGGFace2 as a feature extractor and then feed the features to an SVM classifier. The normalization for the VGGFace2 network is the subtraction of mean from each channel. The embeddings from VGGFace2 were L2 normalized before passing to the SVM classifier. The dataset was split by celebrity in a 80-10-10 way. This was done to ensure that the model has a good representation from every subject as every celebrity has a different number of available images. A subset of the dataset was selected from the celebrities containing the highest number of images. We ran four experiments for 10, 100, 500 and 1000 celebrities. The results for the experiments and image statistics are shown on Table 27.

Table 27: SVM training using VGG Face 2 and dataset statistics

| NUM OF CELEBRITIES | ACCURACY | MIN NUMBER OF IMAGES/CELEB | | AVG NUMBER OF IMAGES/CELEB | | MAX NUMBER OF IMAGES/CELEB | | TOTAL NUMBER OF IMAGES | |
|--------------------|----------|----------------------------|------|----------------------------|------|----------------------------|------|------------------------|------|
| | | TRAIN | TEST | TRAIN | TEST | TRAIN | TEST | TRAIN | TEST |
| 10 | 98.57% | 24 | 7 | 26.1 | 7 | 28 | 7 | 261 | 70 |
| 100 | 97.89% | 24 | 6 | 24.21 | 6.17 | 28 | 7 | 2421 | 617 |
| 500 | 95.59% | 24 | 6 | 24.04 | 6.03 | 28 | 7 | 12021 | 3017 |
| 1000 | 93.91% | 24 | 6 | 24.02 | 6.01 | 28 | 7 | 24021 | 6017 |

For the last batch of experiments, we used the network and data from the VGG Face 2. The network, which was trained for 8631 subjects, was used as the base model and a SoftMax layer was added at the end. We evaluated it on the trained set and got **79.83%** accuracy. The training set for this configuration was the test portion of the VGG Face 2 dataset. This set contains 500 subjects with an average of 338.8 images per subject. We chose 50 random identities for the test set and the remaining images for training. The pre-processing is the same as the previous batch. The main aspect was to replicate the results of VGG Face 2 and its' performance in face recognition tasks. Since they did not use a validation set, we follow the same dataset splitting procedure.

We did a grid search keeping the loss (Categorical Cross Entropy) and optimizer (Adam) fixed. The results are shown in Table 28. The combination of 30 epochs and 1e-04 learning rate gave us the best results. As a second step we explore some combinations of learning rates and fine tune layers for the 20 epochs. The results are shown on Table 29. Since we did not use a validation set and had set up a checkpoint for accuracy based on the training set, we observed that the training of some combinations was stopped before completing all the epochs, so there is no need for further investigation for a greater number of epochs. The best combination achieved **96.63%**, which is **0.5%** better than the best model of VGG Face 2 paper.

Table 28: Transfer learning on VGG Face 2 using VGG Face 2 dataset

| EPOCHS | LEARNING RATE | LOSS ON THE TEST SET | ACCURACY ON THE TEST SET |
|--------|---------------|----------------------|--------------------------|
| 10 | 0.001 | 2.0943 | 92.05% |
| 10 | 0.0005 | 1.1908 | 91.97% |
| 10 | 0.0003 | 0.8185 | 92.46% |
| 10 | 0.0001 | 0.4910 | 93.50% |
| 20 | 0.001 | 2.5310 | 92.52% |
| 20 | 0.0005 | 1.3349 | 92.96% |
| 20 | 0.0003 | 0.9168 | 92.96% |

| | | | |
|-----------|---------------|---------------|---------------|
| 20 | 0.0001 | 0.5391 | 93.77% |
| 30 | 0.001 | 2.6268 | 92.99% |
| 30 | 0.0005 | 1.4928 | 93.06% |
| 30 | 0.0003 | 0.9977 | 93.19% |
| 30 | 0.0001 | 0.5626 | 94.04% |

Table 29: Fine tuning of VGG Face 2

| EPOCHS | LEARNING RATE | FINE TUNE LAYERS | LOSS ON THE TEST SET | ACCURACY ON THE TEST SET |
|---------------|---------------|------------------|----------------------|--------------------------|
| 20 | 0.0001 | 0 | 0.5241 | 93.90% |
| 20 | 0.0001 | 5 | 0.5317 | 93.89% |
| 20 | 0.0001 | 10 | 0.4833 | 94.17% |
| 20 | 0.0001 | 20 | 0.4949 | 93.73% |
| 20(19) | 1e-05 | 0 | 0.4604 | 94.36% |
| 20 | 1e-05 | 5 | 0.4642 | 94.57% |
| 20(17) | 1e-05 | 10 | 0.5263 | 94.01% |
| 20(14) | 1e-05 | 20 | 0.5651 | 96.62% |
| 20(15) | 3e-05 | 0 | 0.4659 | 94.73% |
| 20(11) | 3e-05 | 5 | 0.4482 | 94.61% |
| 20 | 3e-05 | 10 | 0.5036 | 94.25% |
| 20(18) | 3e-05 | 20 | 0.5556 | 96.63% |
| 20(16) | 5e-05 | 0 | 0.4557 | 95.03% |
| 20(17) | 5e-05 | 5 | 0.4597 | 95.04% |
| 20(13) | 5e-05 | 10 | 0.5295 | 93.80% |
| 20 | 5e-05 | 20 | 0.5269 | 93.76% |

Video understanding and annotation

In this part we analyse the current literature state regarding information extraction from videos. We explore the parts of video classification, captioning, objection detection, face recognition and present a plan on which our prospective research will be based.

State of the art

Although there are well-established methods for static images, their application to video data on a frame by frame basis faces two shortcomings: (i) lack of computational efficiency due to redundancy across image frames or by not using a temporal and spatial correlation of features across image frames, and (ii) lack of robustness to real-world conditions such as motion blur and occlusion. To combat that issue, researchers attempted to introduce sequential neural network models, initially proposed in the context of NLP, namely Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTMs) networks and attention-based networks. In the following paragraphs we are presenting the state of the art in aforementioned parts. (Zhu et al., 2020)

Classification

In video classification there are two types of models that have been proposed; two-stream convolutional neural networks and 3D convolutional neural networks. The two-stream convolutional neural network was first proposed in 2014 by Simonyan et al. (2014). It models the dynamic and static information of video data separately. The static features are extracted from the video frames, and the dynamic features are extracted from the optical flow data. Wang et al. (2017) proposed a temporal segment network (TSN). Based on the two-stream convolutional neural network, TSN divides the video into several segments, and each segment is sparsely sampled and classified; the final classification results are obtained by merging the segment classification scores while improving the computational efficiency. As different parts of a video can have a variety of impact on classification results, Peng et al. (2017) introduced the attention mechanism into the two-stream convolutional neural network.

Although optical flow is a suitable method to find relationships in adjacent frames, it is computationally intensive and some researchers consider other methods. For example, Sun et al. (2017) directly calculate the pixel-level spatial-temporal gradient on the feature map, so that the convolutional neural network can obtain motion information directly from the image frame. The optical flow guided feature model proposed by Sun is 15 times faster than the two-stream convolutional neural network.

Inspired by the successful application of recurrent neural networks in NLP, Ng et al. (2015) proposed a method that puts the output of the convolutional neural network to the LSTM resulting in very good accuracies. Carreira et al. (2017) combined the two-stream method with the 3D convolutional neural network and proposed an I3D model to improve performance. Tian et al. (2018) proposed a multimodal deep learning framework using different sources of information including visual, audio and textual data. It solves the problem that many modal data in multimedia resources cannot be fully utilized.

For unsupervised methods, in Zhang and Peng (2018a), the feature composition maps obtained from multiple video-related self-supervised learning tasks are used to measure similarity, and the result guides the training procedure of a lighter classification network. Compared with the features obtained by the supervised learning method, the features obtained by an unsupervised method are weaker, but can make full use of various video data. Gan et al. (2016) and Zhang et al. (2018b) implemented zero-shot classification in order to alleviate the problem of generalization of supervised algorithms.

Another aspect of research is to construct models as compact as possible without compromising the performance. The compact model can be achieved by repeatedly applying small weight matrix operations to all frames in the video (Joonseok et al., 2018). In this way models will take up very little in memory and the floating point operations (FLOPs) are still large. Bhardwaj et al. (2019) used the teacher-student model to make a compact model by training a see-very-little student model with a see-it-all teacher model. The student model requires only a small amount of memory and FLOPs. It also maintains high classification accuracy. In Shweta's work, the student network can reduce inference time by 30%, FLOP by about 90%, and performance degradation is negligible.

Object detection

For object detection in videos, based on the utilization of the temporal information and the aggregation of features extracted from video snippets, video object detectors can be divided into flow-based (Zhu et al., 2017; Zhu et al., 2017a; Zhu et al., 2018), LSTM-based (Liu et al. 2019; Liu & Zhu, 2018; Zhang & Kim, 2019), attention-based (Chen et al., 2020; Guo et al., 2017; Wu et al., 2019), tracking-based (Yang et al., 2019; Feichtenhofer et al., 2017) and hybrid methods (Bertasius et al., 2018; Xiao et al., 2017; Wang et al., 2019a). The methods are presented in Figure 17. The majority of approaches uses ImageNet VID dataset for performance evaluation.

Flow-guided groups use CNN-based architectures like ResNet-101, MobileNet and combinations of those to propagate and align the feature maps according to the flow field obtained by optical flow. For example, Deep Feature Flow (DFF) which uses optical flow to propagate features from key frames to non-key frames, provides high computational efficiency and achieves a runtime of 20.25 frames-per-second (fps) using a Titan K40 GPU. Flow-Guided Feature Aggregation (FGFA) method was proposed to improve the detection accuracy due to motion blur, rare poses, video defocus, etc. It achieves a high accuracy producing 76.3% mean Average Precision (mAP) with 1.36 fps. Obviously, DFF is faster than FGFA. Flow-guided methods are intuitive and well understood to propagate features. Optical flow is deemed suitable for small movement estimation. In addition, since optical flow reflects pixel level displacement, it has difficulties when it is applied to high-level feature maps. One pixel movement on feature maps may correspond to 10 to 20 pixels movement. In the LSTM group, Flow&LSTM (Zhang et al., 2019) achieved the highest accuracy of 75.5%. Looking Fast and Slow (Liu et al., 2019) generated high speed but with low accuracy. Attention-based methods also show the ability to perform video object detection effectively. In the attention-related group, MEGA (Cheng et al., 2020) with ResNeXt-101 as backbone achieved the highest accuracy of 84.1% mAP. It achieved a very high accuracy with a relatively fast speed. Attention-based methods aggregate the features within proposals that are generated. This decreases the computation time. Because of only using the features within the proposals, the performance relies on the effect of RPN to a certain extent. In the tracking-based group, methods have been developed to detect objects on fixed interval frames and track them in frames in between. D&T loss (Feichtenhofer et al. 2017) achieved 75.8% mAP. Tracking is an efficient method to employ the temporal information with a detector assisted by a tracker. However, it cannot solve the problems created by motion blur and video defocus directly. As the detection performance relies on the tracking performance, the detector part suffers from tracking errors. There are also other standalone methods including TCNN (Kang et al., 2018), STSN (Bertasius et al., 2018) and STMN (Xiao & Lee, 2017). In order to further improve the performance in terms of detection accuracy, post-processing can be added to the above methods. With post-processing, the accuracy is noticeably improved. For example, the accuracy of MEGA is improved from 84.1% to 85.4% mAP (Zhu et al., 2020).

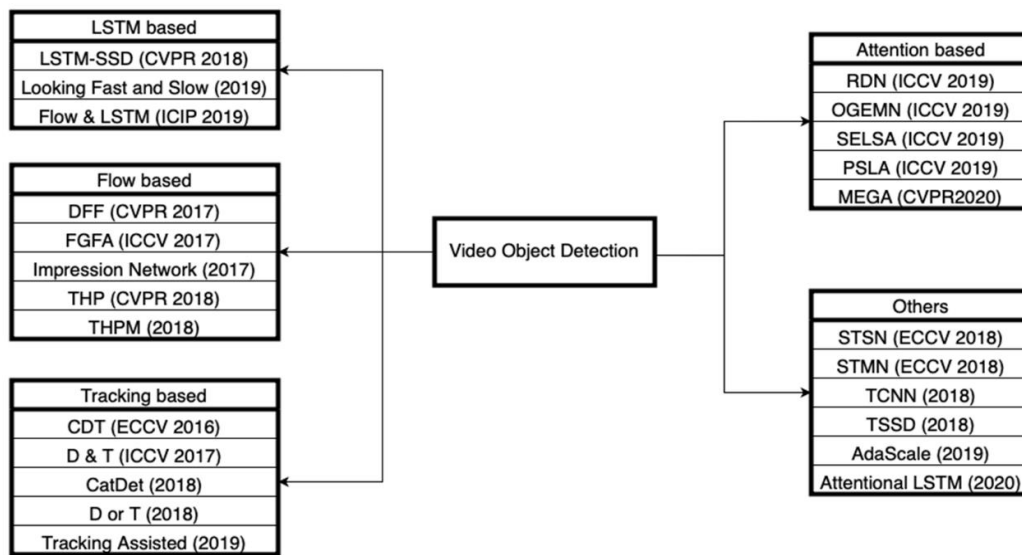


Figure 19: Video object detection methods sorted in different groups (Zhu et al., 2020)

Captioning

The approaches in video captioning can be divided in two categories: CNN-RNN and RNN-RNN video captioning. CNN-RNN methods typically utilize 2D or 3D convolutional networks for feature extraction and the feature vectors are fed into the RNN architecture via a Fully-Connected (FC) linear layer to generate word sequence (Shen et al., 2017; Hou et al., 2019; Zhang and Peng, 2019; Zhang et al., 2020;). In the RNN-RNN branch, models have RNN for both feature extraction and language generation purposes (Pasunuru and Bansal, 2017; Wang et al., 2018; Yang et al., 2018). The encoder portion uses a combination of CNN and a variant of RNN known as LSTM, and the decoder uses LSTM for language generation. Using an Attention model enhances the performance of the decoder. A general overview for both approaches is presented in Figure 20 and Figure 21. A combination of different models has been used as a feature extractor and description generator for video captioning at different times. The performance of a video captioning system also depends on datasets, with the most used dataset being the Microsoft Video Definition (MSVD) (Chen & Dolan, 2011). In 2017, Pasunuru and Bansal (2017) had performed probably best with the MSVD dataset in all the evaluation metrics. In the same year Shen et al. (2017) brings out quite similar results in all evaluation metrics using the MSR-VTT dataset. With the improvement of LSTM (two layers), Hou et al. (2019) and Zhang and Peng (2019), show some impressive results with the MSVD dataset in 2019. Zhang et al. (2020) shows by far the best results in BLEU@4, CIDEr, and METEOR outperforming all the previous video description models. In 2020 (till July), Chen et al. (2020) showed possibly the best results of all time with the MSVD dataset. Many other datasets have been used. Rather than MSVD, Xiao et al. (2020) shows conceivably the best result using the MSR-VTT dataset.

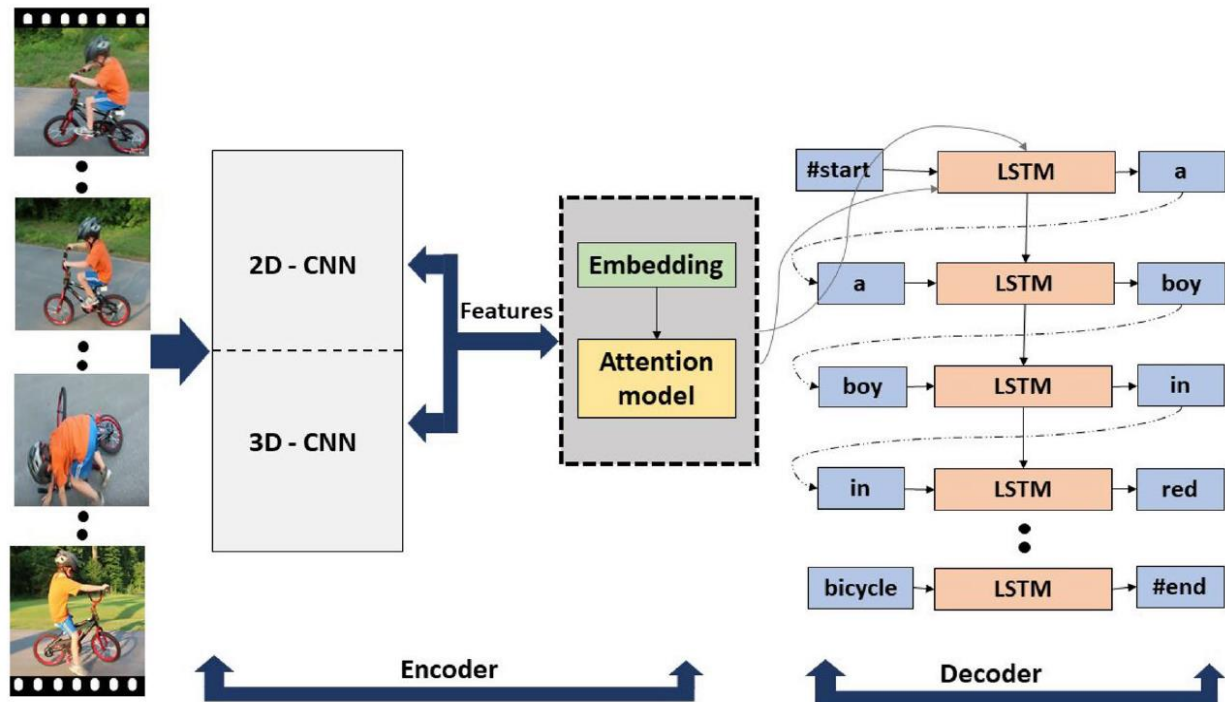


Figure 20: General overview of CNN-RNN video caption generation method (Islam et al., 2021)

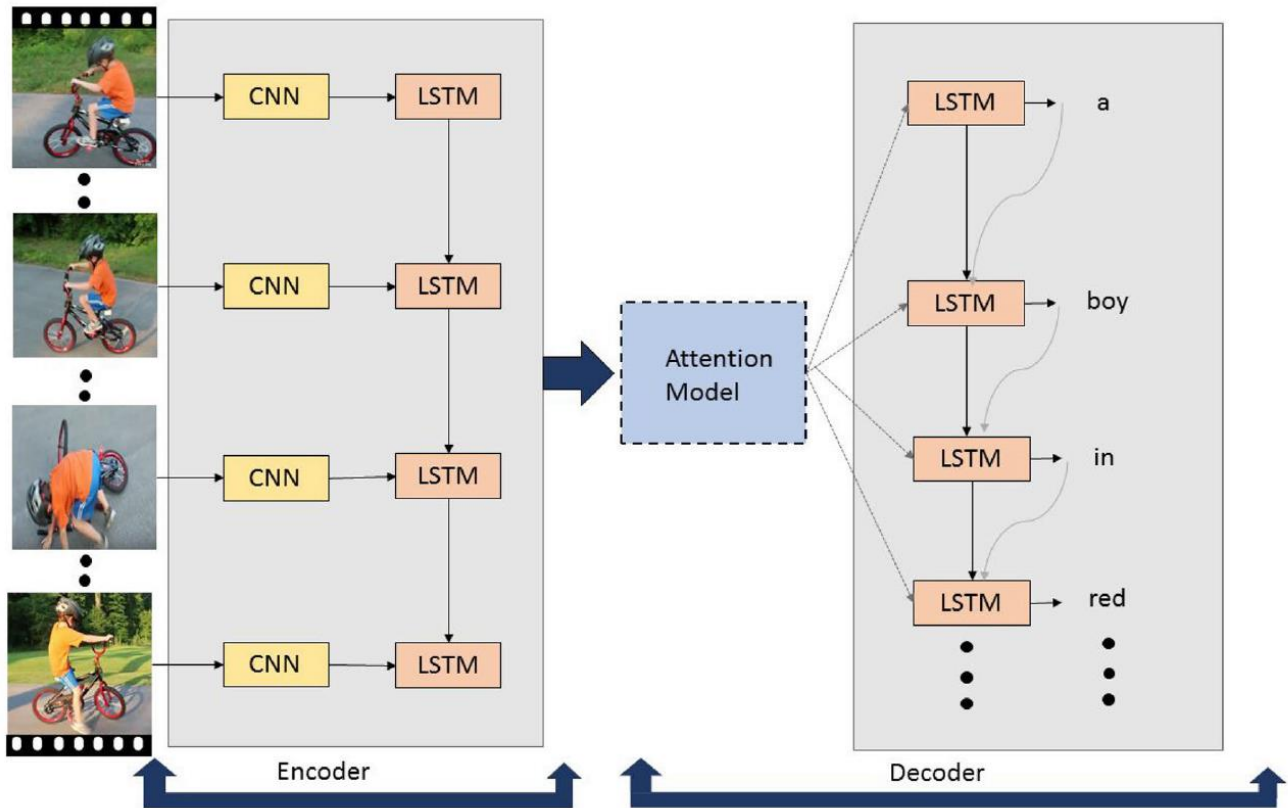


Figure 21: General overview of video caption generation method where RNN is used in Encoding stage (Islam et al., 2021)

Face recognition

Finally, in the face recognition branch, the objective is the construction of a method that accurately detects and keeps track of multiple faces throughout the video duration. Zheng and Xu (2021) proposed a SENResNet -which is similar to the VGGFace2 implementation, as it utilizes Squeeze and excitation blocks- for face detection and a Regression Network-based Face Tracking (RNFT) model (RFNT) for facial feature extraction from adjacent frames and prediction of the the position of the target face in the next frame. The reported results exceed the state-of-the-art in terms of accuracy and precision. Lin et al. (2020) presented a multi-trajectory incremental learning (MTIL) algorithm, which categorizes trajectories using a Euclidean distance-based greedy algorithm and estimates the most likely labels for each trajectory by incremental learning to correct their classification and improve the accuracy of recognition. Furthermore, this study proposes an enhanced detection method that combines face detection with a robust tracking-learning-detection (TLD) algorithm to improve the performance of face detection in video. Huo and Van Zyl (2020) compared the use of the Siamese network with contrastive loss and Triplet Network with triplet loss implementing the following architectures: Google/Inception architecture, 3D Convolutional Network (C3D), and a 2-D Long short-term memory (LSTM) Recurrent Neural Network. The dataset used was the YouTube Face Database designed for investigating the problem of face recognition in videos. The reported results showed that the 3-D Convolution networks and 2-D LSTM with triplet loss outperform the Google/Inception with triplet loss in top-n rank face retrievals on the dataset and a Support Vector Machine (SVM) was used in conjunction with the CNNs' learned feature representations for facial identification. Zheng et al. (2019) proposed a multi-scale single-shot face detectors (SSD) to efficiently localize faces in videos. The detected faces are then grouped respectively through carefully designed face association methods, especially for multi-shot videos. Finally, the faces are recognized by the proposed face matcher based on an unsupervised subspace learning approach and a subspace-to-subspace similarity metric. Extensive experiments on challenging video datasets, such as Multiple Biometric Grand Challenge (MBGC), Face and Ocular Challenge Series (FOCS), IARPA Janus Surveillance Video Benchmark (IJB-S) for low-quality surveillance videos and IARPA JANUS Benchmark B (IJB-B) for multiple-shot videos, demonstrate that the proposed system can accurately detect and associate faces from unconstrained videos and effectively learn robust and discriminative features for recognition. To battle computationally heavy probabilistic models which models on a frame-by-frame basis to identify faces Nagendra et al. (2015) proposed regularized sparse representation classification (RSRC) algorithm uses ℓ^2 minimization approach instead of conventional ℓ^1 minimization method and obtains a single coefficient vector for all frames. Since second order minimization is used, more sparsity ratios are achieved and the residual error over the frames are reduced. The proposed algorithm is compared with the existing methods and the experimental results prove that, due to minimal error better classification accuracy and high confidence value are achieved.

Plan

Based on our review so far, it is evident that there is a plethora of research done in video understanding and annotation and many of the architectures (CNNs, LSTMs, attention mechanisms, etc.) are used by multiple branches which can be advantageous in the development stage. Our plan is to initially create baseline models for each part with the existing collection of publicly available datasets while trying to alleviate the known issues for each of them with the objective of creating more robust methods. Afterwards, we will investigate improvement options over the current state of the art.

2.2.2 3-Dimensional Data

One of the new media types addressed in T3.1 is 3D content, which is particularly used in XR/VR environments. Annotating and understanding this media type is one of the objectives of T3.1 towards an indexing service for 3D content. While there are multiple tasks in the 3D data processing literature (i.e., 3D single object classification, 3D Object detection, 3D semantic/instance/part segmentation to name a few), in T3.1, we will consider the ones that operate on entire scenes containing multiple 3D objects rather than single 3D objects. This is due to the fact that our objective is to annotate the MediaVerse content, which in the case of 3D are entire VR scenes.

There are several tasks that may be good candidates for MV's annotation service, but during these first months of the project, we mainly focus on semantic segmentation (SS), whose target is to classify each point to a class (similar to classification but for all points in the 3D scene). The reason for that is because SS has the two main features needed for MV's content annotation. First of all, its input is a 3D scene, which is the same as the user input in MV, and secondly, it can provide multi-object annotations in one scene (which is quite important for content retrieval). This is in contrast to other tasks, such as 3D single-object classification, which can only provide annotations for a single concept describing the 3D data. Single object detection is one of the first tasks in the 3D content processing literature, including the task of part segmentation. Various datasets and methods have been developed in this domain (ModelNet²⁹, ShapeNet³⁰, etc.). However, single object classification is not in line with the content that is available in MediaVerse, which is typically entire scenes with multiple objects. Moreover, in 3D object detection, in most cases the input is in RGB-D (RGB + depth images) format and the output is 3D bounding boxes. RGB-D format is not considered full 3 dimensional data (instead it is typically referred to as 2.5D), and it does not take advantage of the 3D space. Therefore, any task (or network) that operates with RGB-D data, is out of the scope of this task. In the next sections, we will discuss in detail the task of SS.

All the above tasks were easy to reject given that they do not operate directly in a VR scene. When it comes to tasks related to 3D scenes, we prioritize semantic segmentation over the instance segmentation task because we are not interested in a separate annotation of the same object (chair one and chair two), rather than a general annotation if the model (chair) is part of the scene. Another interesting task is 3D scene understanding (i.e. provide high-level description of the scene such as office, living room or class). However, most of the available works depend on RGB-D input, which is not the case of the available data in MediaVerse, as discussed earlier, and additionally there is only one public dataset (ScanNet³¹) with high-level annotations but with only 1500 observations (Huang et al. 2020). For the aforementioned reasons, we decided not to proceed with this task as a first step of the annotation service but given the limited research interest for this task and the fact that it aligns with T3.1 objective, 3D scene understanding may be considered for future work.

In this part of the deliverable, we will present the state of the art for the task of semantic segmentation, related to 3D scene analysis, aligned with the different 3D data representation types. In the following subsections, we present the work that has been done during the first months of the project for 3D data processing. Firstly, a small summary of the different 3D data representations is presented. Then, the focus is given in the semantic segmentation task. Datasets, relevant evaluation metrics and state of the art networks are presented. Finally, the implementation details and the results are presented and discussed both for Semantic Segmentation and annotation.

²⁹ <https://modelnet.cs.princeton.edu/>

³⁰ <https://shapenet.org/>

³¹ <http://www.scan-net.org/>

3D Data Representation

A major difference between 2D and 3D content is the data type and the representation of the content. While in 2D, the data type is an image and the representation of an image is a 2D matrix with the corresponding pixel values, in 3D there is a plethora of 3D data types and their corresponding representations, besides the 3D voxels equivalent to the 2D pixels. In Figure 22, a high-level description of the different 3D data representations can be found. The development of 3D data processing methodologies depends heavily on the data representation and it will be discussed thoroughly later in this deliverable.

Each one of the representations has its own advantages and disadvantages. That is there is not any “best” representation, but there is a trade-off between the effectiveness and the complexity of each representation. For example, voxels are able to capture the 3D geometry efficiently, however they suffer from some constraining limitations. They represent both occupied and non-occupied parts of the scene, which leads to a huge need for computer storage and therefore, they cannot be used for high-resolution data. On the other hand, point clouds are irregular, unstructured and unordered, which make their processing with conventional deep learning (DL) approaches, such as CNNs, challenging, but they are significantly less complex than voxels.

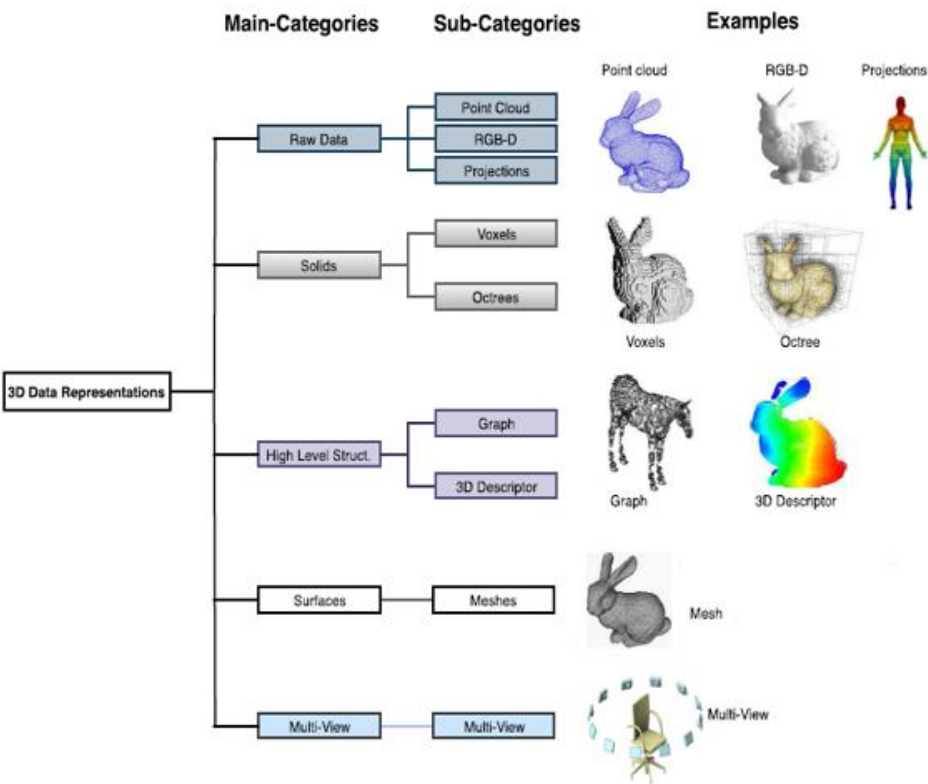


Figure 22: A representation of different 3D data types Source: A Review on Deep Learning Approaches for 3D Data Representations in retrieval and Classifications

3D Semantic Segmentation State-of-the-art

As already discussed in the introduction, the main task of T3.1 is the annotation of different data contents. In order to achieve that, for 3D representations, we will focus on Semantic Segmentation of 3D scenes. The target of semantic segmentation, which is a generalization of the classification task, is to classify each point to a specific class. On its own, SS cannot provide annotations for the 3D scene. In order to extract these annotations from SS results, another step is required. That is, to create a list by grouping all the predictions for every single point. That list contains all the annotations of the specific 3D scene and it will be used as tags for that scene.

Datasets for Semantic Segmentation

A summary of the most common benchmark datasets for scene segmentation can be found in Table 30. It can be easily observed that not only is there a limited amount of datasets with 3D scenes, but their context is also quite limited. Specifically, the existing datasets consist of houses and offices (when it comes to indoor scenes) and roads, farms and other common outdoor scenes. Finally, note that Table 30 contains only datasets for full 3D representations without taking into account RGB-D representation, which is considered to be 2.5D.

Table 30: Datasets for 3D scenes. SS stands for Semantic Segmentation and IS for Instance Segmentation. S and R represent Synthetic or Real dataset

| DATASET | YEAR | TYPE | CLASSES TYPE | DESCRIPTION | NUMBER OF CLASSES | REPRESENTATION | TASKS |
|---|------|------|--------------|---|-------------------|--|--------|
| SceneNN (Hua et al., 2016) | 2016 | S | Outdoor | 101 indoor scenes offices, bedrooms, living rooms | 40 | Mesh reconstruction from RGB-D | SS |
| S3DIS (Armeni et al., 2017) | 2017 | R | Indoor | 271 rooms of 6 scenes from 3 different buildings, offices, classes etc. | 13 | Points: 273M Mesh reconstruction without mesh annotation Only per point annotation | SS, IS |
| ScanNet (Dai et al., 2017) | 2017 | R | Indoor | 707 environments Bathroom, living room etc. | 20 | Labelled voxelized Mesh reconstruction Points available after sampling | SS, IS |
| Semantic3D (Hackel et al. 2017) | 2017 | R | Outdoor | Urban and rural Farms, churches, streets etc. | 8 | Points + RGB + Intensity | SS |
| Matterport3D (Chang et al. 2017) | 2017 | R | Indoor | 90 indoor building-scale scenes | 40 | Points Mesh reconstruction | SS, IS |
| Paris-Lille (Roynard et al., 2018) | 2018 | R | Outdoor | Road, cars, pedestrians | 10 | Points | SS |
| SemanticKITTI (Behley et al., 2019) | 2019 | R | Outdoor | Road, car, building etc. | 25 | Sequential point clouds | SS |

Evaluation Metrics

To assess the performance of the models, we evaluate the results at two granularity levels; first at the pixel level (i.e., how accurately each point in the point cloud is classified) and second at the scene level (i.e., how accurately the model predicts the classes in the scene). For the first one, we directly evaluate the output of the SS, which is a point wise classification, with the provided ground truth of the dataset. For the second, we aggregate all the point-wise predictions of the network as a list. Therefore, that list contains all the objects that have been predicted to occur in the 3D scene. Then we compare that list with the ground truth (i.e., another list which contains all the objects that actually occur in the 3D scene). In summary, we use 6 evaluation metrics (3 for point-level annotations and 3 for scene-level annotations):

Metrics for point-level annotation

- **Overall accuracy (OA):** The overall accuracy of the network among all points.
- **Mean Accuracy (mAcc):** An extension of OA by computing the OA per-class and then averaging over the total number of classes.
- **Mean Intersection over Union (mIoU):** The intersection ratio between the ground truths and predicted values, normalized by the total number of classes.

Metrics for scene-level annotation

- **Precision (Pr):** The ratio of correctly predicted positive examples (objects that are part of the scene) divided by the sum of true positives (correctly classified in the scene) and false positives (predicted in the scene but they are not part of it). It is used to determine how many of the retrieved objects are relevant (part of the scene).
- **Recall (R) :** The ratio of correctly predicted positive examples (same as precision) divided by the sum of true positives and false negatives (objects that are part of the scene but were not predicted as such). It is used to determine if all of the relevant objects in the scene have been retrieved.
- **F1-score (F1) :** The harmonic mean of precision and Recall. The ratio between the multiplication and summation of precision and recall, multiplied by 2.

Literature

While there is a plethora of works in 2D, the research interest in processing 3D content has mostly emerged in the last decade. The availability of low-cost devices for the collection of 3D data, made the researchers focus on this field by trying to apply their knowledge from 2D to 3D. In the literature, there are several ways to categorize methodologies for 3D semantic segmentation. The most common categorization depends on the input data representation. Therefore, we can classify the 3D semantic segmentation methodologies into (He et al. 2021): a) RGB-D based, b) Projection based, c) Voxel based, d) Point cloud based, and e) Mesh based.

We take into consideration only works, which directly process 3D data. Therefore, we do not present any research on RGB-D data, which is considered to be 2.5 D, and projection based approaches, where the 3D is converted into 2D multi-view or spherical images. Moreover, the voxel based architectures are constructed based on the point cloud representation. Specifically, the raw point clouds are converted to voxels, which is the input of the network. The Point based category consists of networks that directly process raw point clouds without any transformation. Finally, the mesh-based approaches directly process the 3D meshes.

Voxel based: In the early research stage in point cloud representations, the community tried to transfer the deep learning techniques from 2D to 3D. That is to use Convolutional Neural Networks (CNN) directly in 3D. For that to happen, the raw point clouds were converted to a grid representation. Therefore, as a first step of the pipeline, the raw point clouds are voxelized. This procedure surpasses the limitation of the points, being unordered and unstructured, and the final data can be the input of a CNN similar to 2D. On the other hand, their high memory consumption leads to low resolution inputs, which is a quite important limitation. In this category, we can group the methods based on the way that voxels are distributed within the 3D space; *uniform* and *non-uniform*.

Uniform (Dense) voxelization: One of the limitations of voxel representation is that within a voxel there is no information regarding the distribution of the points, but only a Boolean status of being or not occupied. To address that, Meng et al. (2019) proposed Voxel VAE Net (VV-Net), a novel network which is using a Variational Autoencoder (VAE). Beyond that, a radial basis function (RBF) is used instead of typical sampling techniques to

encode the local geometries and capture the distribution of the cloud points for each voxel. It is known that 3D scans, most times, suffer from sensor occlusion, which leads to incomplete 3D scenes. ScanComplete (Dai et al., 2018), works in a hierarchical coarse-to-fine manner, where each level takes as input a partial voxelized 3D scan and outputs not only the complete 3D scene, but also semantics for each voxel. Moreover, the output of each level is the input in the next level. In this way, the input resolution is gradually increasing, which leads to fine detailed output.

Nonuniform voxelization: For voxels, higher resolution leads to higher complexity. Also, 3D scenes are often sparse and, hence, dense convolutions add undesirable complexity. To surpass this limitation, Riegler et al. (2017) proposed OctNet, a network which makes use of the sparsity property. Specifically, the 3D space is divided into nonuniform voxels by using unbalanced octrees. In this way, deeper networks can be used without the need of lower resolution. On the other hand, empty space still consumes memory, even if it is less than the uniform grid approaches. In order to tackle this limitation Graham et al. (2017) proposed a submanifold sparse convolutional networks (SSCNs), which in contrast to OctNets, do not lead to computational or memory overhead.

Point cloud based: Point cloud is the simplest form of 3D representation, including either just the XYZ coordinates in the 3D space or extra features such as colour and normals. Their simplicity compared to other representations comes with a cost. Specifically, their main disadvantage, as described in (Bello et al., 2020) is the fact that point clouds are irregular, unstructured and unordered. The availability of (mostly) LIDAR devices such as Kinect V2, Tele-15 etc, made point clouds accessible and the centre of attention for the researchers in the 3D domain (Bello et al., 2020). The point based methodologies, which directly perform in *raw data*, can be classified as (He et al., 2021) point-wise MLP, Point Convolution based and Graph convolution based. Specifically, Point-wise MLP approaches are using MLP to extract features directly from the raw point clouds. Point Convolution based networks perform convolution operations on the raw point clouds. Finally, in Graph Convolution based approaches, the raw point clouds are represented in a graph manner and each point cloud is represented by a node. Features are extracted by applying convolutions on the points, which in this case are connected with a graph structure. PointNet (Qi et al., 2017) was the pioneer work when it comes to **raw point clouds**. It can be used directly for classification and semantic segmentation, but it can also be part, as feature extractor, for other tasks such as point cloud completion. However, it can only capture global features, and therefore, it fails to take advantage of local features and the correlation between close-by points. In order to overcome the limitation of PointNet (Qi et al., 2017), Qi, Yi et al. (2017) proposed PointNet++ (Figure 23), a hierarchical network, which uses PointNet in a recursive manner. Similar to PointNet, PointNet++ can be used for classification and segmentation tasks. Most works on point cloud segmentation in this category rely on PointNet++.

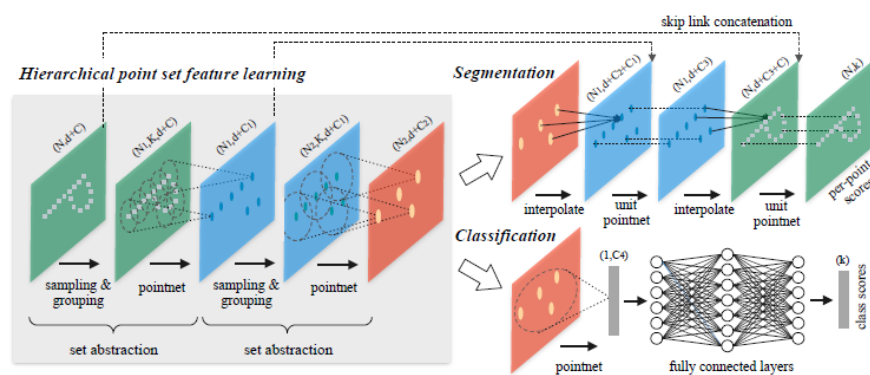


Figure 23: PointNet++ architecture which can be used for classification and segmentation. As a sampling algorithm, it uses farthest point sampling (FPS). Moreover, ball query is used as grouping algorithm

Mesh based

Mesh is another, more complex than point clouds, 3D representation. This kind of method works with triangular or quadrilateral meshes. Unlike point clouds, the connection between the faces of the mesh can provide information regarding the neighbours of the vertices and make the computation of normal vectors easier. Huang et al. (2019) proposed **TextureNet**, which has 3D textured mesh as input. It is based on a 4-rotational symmetric field (4-RoSy) for feature extraction on the surface of a mesh. Specifically, the network is able to learn features from the 3D mesh by computing local geodesic neighborhoods. The output is a set of features that can be used for SS and other tasks. Schult et al. (2020) proposed DualConvMesh-Net. DualConvMesh-Net gets meshes as input and uses two different types of convolutional kernel (one in the Euclidean space and one in the Geodesic space). Their proposed (hierarchical) architecture consists of several of these dual convolutional kernels together with skip connections. As an input mesh, they use not only the position, but also the color and normal of each vertex. Finally, they train in the cropped mesh while evaluating on the entire surface. Their network achieved state of the art results when it comes to graph methods.

Even though there is tremendous progress in 3D SS, and in 3D processing in general, during the last years, each one of the methods discussed in the previous sections have their own drawbacks. For example, voxel based methods are able to capture more of the geometric information, but due to their complexity, they can only operate in low resolution. On the other hand, multi-view images can capture more of the semantic information, but they are limited to the geometric information given their 2D format. In order to surpass these drawbacks, researchers are trying to get the best out of each representation by combining them (hybrid representation). In this category we can find 3DMV (Dai and Nießner, 2018), where the 3D voxel representation is combined with multi-view images. Specifically, the multi-view RGB images are passed through a 2D network for feature extraction. These features are then projected in the 3D space and are passed as input in a 3D CNN. The same happens for the 3D voxel representation. Finally, the outputs of the 2 3D CNNs are combined in another 3D CNN, which outputs per-voxel semantic segmentation predictions. Several other works, such as (Chiang et al., 2019), (Jaritz et al., 2019), (Meyer et al., 2019) are following a similar methodology by combining 2D and 3D space. A combination of volumetric and point cloud input proposed in (Rethage et al., 2018). Liu, Tang et al. (2019) proposed Point-Voxel CNN (PVCNN), a network, which combines both points and voxel representations. In order to guarantee less memory consumption, the input of the 3D network is point clouds, while the convolutions are performed in the voxel representation.

Selection of network for deployment

As already discussed in the earlier sections, during these first months, we focused on the task of SS for 3D data. In order for the SS results to be used as annotations, we need one more step on the output of the network. Specifically, the result of any SS network is a point-wise classification. This cannot be directly used for annotation. For that reason, the final step of the pipeline is to group by all the predicted points per class and produce a list of objects existing in the 3D scene.

Finally, despite the fact that the users' input in MV will be a 3D scene in mesh representation, we chose to use a network with point cloud as input. There are several reasons behind this choice. First of all, as discussed in the previous section, the works based on mesh representation are limited compared to point clouds. Moreover, point clouds do not make use of 3D coordinates only, but they can also include colors, normals etc (similar to meshes). Therefore, despite their drawbacks, recent works with point clouds are able to provide competitive results. The aforementioned reasons along with the fact that point clouds are less computationally expensive

compared to meshes, make point clouds the most widely 3D data representation used and best candidate for MV's annotation pipeline. As a semantic segmentation network, we make use of (Qiu et al. 2021), a state of the art network, which works with point clouds as input.

Network Details

Recently, Qiu et al. (2021) proposed the Bilateral Augmentation and Adaptive Fusion ([BAAF-NET](#)), which can be used for semantic segmentation. The network leverages all available information as input. That is not only the XYZ coordinates, but also other related information such as color, normals etc can be used. As a first step, MLPs are used in order to extract features from the input (not the XYZ coordinates, only the extra features). Later, the feature and the 3D scene with the XYZ coordinates are passed through a Bilateral Context Module, which is using several blocks to downsample the point clouds and explore lower resolutions of the same point cloud. This process, similar to 2D CNN, is used for downsampling the input, but upsampling the output dimensions as well. Moreover, an Adaptive Fusion module is used to aggregate the outputs of all the different resolutions from the previous blocks. Finally, a FC network is used in order to provide the final semantics for each point. Cross-entropy loss L_{CE} is used along with point-level augmentation losses $L(p_i)$, which is defined as the l_2 distance between a centroid point p_i and its neighbors.

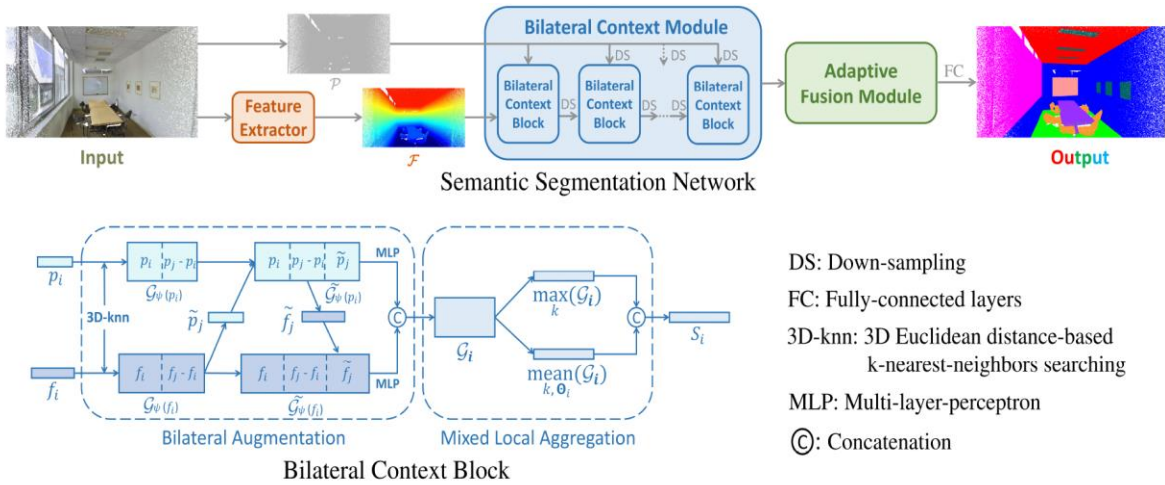


Figure 24: The entire BAAF-Net architecture (Qiu et al. 2021)

In terms of a Bilateral Context Block processing N_m points, the total augmentation loss regarding N_m points would be $L_m = \sum_{i=1}^{N_m} L(p_i)$. Therefore, the overall loss is:

$$L_{all} = L_{CE} + \sum_{m=1}^M w_m \cdot L_m, \quad (1)$$

where w_m is a hyper-parameter of weight for each Bilateral Context Block. They tested their approach in many 3D scene datasets and they achieved state of the art results. Specifically, for [S3DIS](#) dataset they achieved the lower mIoU, while for [Semantic3D](#) and [Semantickitti](#), their results were competitive with the state-of-the-art.

Hyperparameter Details

The implementation of (Qiu et al. 2021) was in Python 3.6. Regarding specific frameworks and libraries, Tensorflow (Abadi et al. 2015) 1.13.1 was used for building the network, while Open3D (Zhou et al. 2018) was the main library used for pre-processing of the 3D data.

The models have been trained for 100 epochs with a batch size of 4. Moreover, the Adam (Kingma et al., 2014) optimizer is used for minimizing the overall loss in Equation 1. The learning rate started from 0.01 and decays with a rate of 0.5 after every 10 epochs.

Experiments

In this section, we provide the results when using BAAF-Net. Specifically, we make use of the pre-trained models of BAAF-NET trained in the S3DIS dataset (Armeni et al., 2017). We evaluate the model in a test split of the same dataset. More specifically, S3DIS is collected in 6 large-scale indoor areas that originate from 3 different buildings of mainly educational and office use. During the experiments we make use of 5 areas for training and one area for testing. For each area, all modalities are registered in the same reference system, yielding pixel to pixel correspondences among them. It contains in total 271 rooms distributed in these 6 areas. Each 3D scene has between 0.5 and 2.5 million points. There are a total of 13 classes of indoor scenes mostly in working environments such as offices, conference rooms, etc. The 13 classes are the following: Ceiling, floor, wall, beam, column, window, door, table, chair, sofa, bookcase, board, clutter.

Comparison with state of the art

In Table 31, one can see the results in three evaluation metrics for several state of the art approaches. The results for all networks except BAAF-Net, are presented as they are in the official papers. First of all, BAAF-NET performs better in all three metrics compared to a lot of similar state of the art networks. Moreover, while the OAs of the networks are not that different, this could be misleading for the SS ability given the imbalance of the dataset (i.e. the number of points for walls, ceiling and floor points exceeds the rest of the categories). The other two metrics (mACC and mIoU) are more appropriate for comparing the networks and as can be seen in Table 31, BAAF-NET outperforms.

Table 31: Semantic Segmentation (6-fold cross-validation) results on the S3DIS dataset (mAcc : average class accuracy, OA: overall accuracy, mIoU: mean Intersection-over-Union)

| YEAR | METHOD | mACC | OA | mIoU |
|------|-----------------|-------------|-------------|--------------|
| 2017 | PointNet | 66.2 | 78.6 | 47.6 |
| | PointNet++ | 67.1 | 81.0 | 54.5 |
| 2018 | A-SCN | - | 81.6 | 52.7 |
| | PointCNN | 75.6 | 88.1 | 65.4 |
| | SPG | 73 | 85.5 | 62.1 |
| 2019 | DGCNN | - | 84.1 | 56.1 |
| | KP-Conv | 79.1 | - | 70.6 |
| | ShellNet | - | 87.1 | 66.8 |
| | PointWeb | 76.2 | 87.3 | 66.7 |
| | SSP+SPG | 78.3 | 87.9 | 68.4 |
| 2020 | Seg-GCN | 77.1 | 87.8 | 68.5 |
| | PointASNL | 79 | 88.8 | 68.7 |
| | RandLA-Net | 82 | 88.0 | 70.0 |
| | MPNet | - | 86.8 | 61.3 |
| | InsSem-SP | 74.3 | 88.5 | 64.1 |
| | BAAF-Net | 83.1 | 88.9 | 72.15 |

Semantic Segmentation results

Table 32 presents the results for each one of the 6 areas. As mentioned above, we use the pre-trained models from BAAF-Net and evaluate the model in each test area. For each area, the training set consists of all the other areas. That is, for instance, train the model in all areas except 5 and test in area 5. It can be seen that the results vary across different testing areas. In all areas the network achieves high IoU for “Ceiling”, “Floor” and “Wall” categories. On other hand, in most areas the IoU is low for “Beam” and “Column”. One interesting observation is the IoU for object “beam” in Area 5, whose value is 0. This can be explained by the fact that for all 68 scenes (with millions of points at each scene) in this area, only approximately 20000 points belong to the class “beam”, but none of them has been classified correctly as “beam”. Therefore, the value of IoU is zero. Moreover, the IoU for the “column” class is also low. One explanation of that could be that for that particular area, the model cannot separate the two objects, which is normal given the similarities that a column and a beam have.

Table 32: Results using pre-trained models from BAAF-NET in S3DIS dataset. All models have been trained in all areas except for the testing area

| TESTING AREA | mIoU | CEILING | FLOOR | WALL | BEAM | COLUMN | WINDOW | DOOR | TABLE | CHAIR | SOFA | BOOKCASE | BOARD | CLUTTER |
|--------------|-------|---------|-------|-------|-------|--------|--------|-------|-------|-------|-------|----------|-------|---------|
| Area 1 | 76.23 | 96.51 | 95.41 | 80.22 | 65.23 | 58.70 | 77.91 | 84.28 | 70.57 | 82.44 | 77.78 | 61.20 | 72.98 | 67.80 |
| Area 2 | 57.78 | 87.09 | 95.12 | 80.04 | 19.50 | 32.74 | 47.67 | 69.58 | 45.31 | 83.57 | 52.20 | 50.90 | 32.83 | 54.61 |
| Area 3 | 79.87 | 95.82 | 98.19 | 83.21 | 74.06 | 39.92 | 86.34 | 88.62 | 74.38 | 83.54 | 77.91 | 73.24 | 89.21 | 73.86 |
| Area 4 | 64.20 | 94.81 | 97.12 | 78.78 | 50.95 | 49.11 | 30.53 | 60.77 | 67.69 | 76.67 | 70.37 | 51.12 | 45.11 | 61.58 |
| Area 5 | 65.33 | 93.10 | 98.03 | 82.18 | 0 | 22.84 | 65.13 | 65.35 | 78.63 | 87.43 | 60.37 | 70.74 | 68.49 | 57.06 |
| Area 6 | 81.77 | 96.40 | 97.45 | 86.27 | 79.76 | 80.81 | 78.49 | 90.00 | 77.12 | 87.94 | 65.16 | 72.40 | 80.04 | 71.20 |
| 6-fold CV | 72.15 | 93.4 | 96.83 | 81.55 | 61.63 | 49.45 | 65.47 | 73.37 | 72 | 83.74 | 67.04 | 64.26 | 66.92 | 62.32 |

Except for the quantitative evaluation, the visualization of the results is also important in the task of Semantic Segmentation. As can be seen in Figure 25, the model is able to correctly capture the 3D scene at a high level. Specifically, in all rows it captures the objects correctly, while it misses the details close to the borders of each object (for example the board in the wall or the objects in the table). As we mentioned above though, we are not really interested in the Semantic Segmentation itself rather than the list of the classified objects in the 3D scene. For example, in the second row, our model does not capture the door in the middle of the scene. This would lead to worse SS results, but it will not affect the annotation results because the object “door” is in the final predictions.

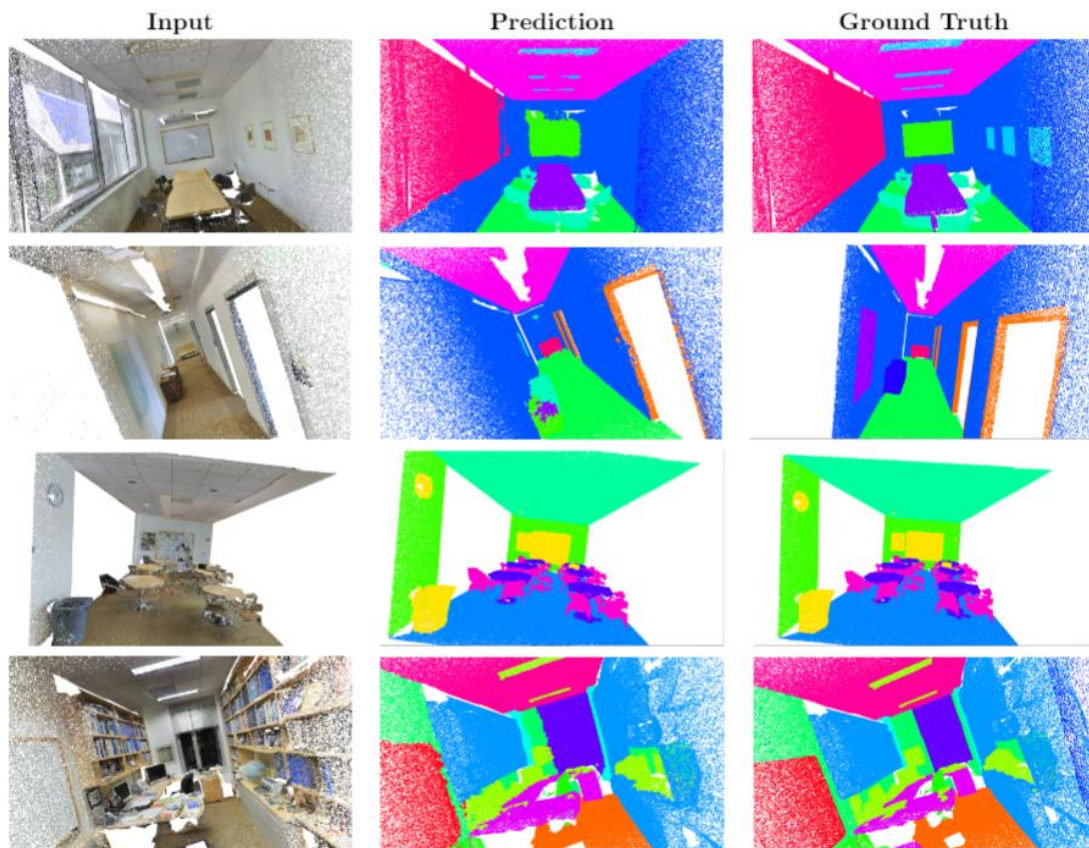


Figure 25: Semantic Segmentation results in S3DIS dataset

Annotation Results

Given that the final goal is not the SS results, but the annotations of each scene, our ultimate goal is to evaluate the network in an annotation level. Therefore, as a final step, we combine all the predictions of the SS as a list, which contains the objects that are predicted to occur in the scene. For instance, if there are points that have been classified as “chair”, then the object “chair” is predicted to be part of the scene. Further, we evaluate the two lists (i.e. the one with the predicted objects and the one with the objects that are actually part of the scene) in a multi-label classification manner. Annotation results can be found in Table 33. Specifically, we present the Precision, Recall and F1-score for each testing area and each object. For “Ceiling”, “Floor” and “Wall” categories, both the precision and recall is 1 in all Areas. That means that, in Area5 for instance, for all the scenes the two aforementioned lists were identical. Therefore, all the objects that are actually part of the scene are also part of the prediction list (recall equals to 1), but also all the objects in the prediction list are part of the scene (precision equals to 1). Moreover, the recall in almost all areas and objects is quite high, which means that all the relevant objects have been retrieved for the scene. Additionally, the precision is also high in most of the areas except Area 2. There, despite the fact that the recall for each object is high, the precision is not. That means that the model manages to retrieve the relevant context, but on the other hand not everything that has been retrieved was relevant (i.e., objects in the prediction list are not actually part of the scene). When it comes to the annotation for MV, it is important to have a good trade-off between precision and recall. On the one hand, we need a high recall so that the user will retrieve all the relevant content. On the other hand, a low precision means that a lot of unrelated content will be presented during the retrieval, which may confuse the user. In any case, both of the metrics are high when using pre-trained models from BAAF-Net.

Table 33: Annotation results for all areas of S3DIS

| TEST AREA | METRIC | CEILING | FLOOR | WALL | BEAM | COLUMN | WINDOW | DOOR | TABLE | CHAIR | SOFA | BOOKCASE | BOARD | CLUTTER |
|-----------|-----------|---------|-------|------|-------|--------|--------|-------|-------|-------|-------|----------|-------|---------|
| Area 1 | Precision | 1 | 1 | 1 | 0.87 | 0.867 | 0.848 | 0.977 | 0.944 | 0.89 | 0.714 | 0.89 | 0.846 | 1 |
| | Recall | 1 | 1 | 1 | 1 | 0.929 | 1 | 1 | 1 | 1 | 1 | 0.97 | 1 | 1 |
| | F1-score | 1 | 1 | 1 | 0.931 | 0.897 | 0.918 | 0.989 | 0.971 | 0.94 | 0.833 | 0.928 | 0.917 | 1 |
| Area 2 | Precision | 1 | 1 | 1 | 0.41 | 0.294 | 0.25 | 0.949 | 0.786 | 0.759 | 0.25 | 0.53 | 0.571 | 1 |
| | Recall | 1 | 1 | 1 | 0.9 | 0.714 | 1 | 1 | 1 | 1 | 1 | 1 | 0.89 | 1 |
| | F1-score | 1 | 1 | 1 | 0.563 | 0.417 | 0.4 | 0.974 | 0.88 | 0.863 | 0.4 | 0.694 | 0.696 | 1 |
| Area 3 | Precision | 1 | 1 | 1 | 0.684 | 0.462 | 0.89 | 0.95 | 0.87 | 1 | 0.75 | 0.867 | 1 | 0.957 |
| | Recall | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | F1-score | 1 | 1 | 1 | 0.812 | 0.632 | 0.94 | 0.974 | 0.929 | 1 | 0.857 | 0.929 | 1 | 0.978 |
| Area 4 | Precision | 1 | 1 | 1 | 0.125 | 0.615 | 0.75 | 0.938 | 0.871 | 0.97 | 0.6 | 0.74 | 0.32 | 1 |
| | Recall | 1 | 1 | 1 | 1 | 0.889 | 0.923 | 0.978 | 1 | 0.97 | 0.75 | 1 | 0.78 | 1 |
| | F1-score | 1 | 1 | 1 | 0.22 | 0.73 | 0.828 | 0.957 | 0.931 | 0.97 | 0.67 | 0.853 | 0.45 | 1 |
| Area 5 | Precision | 1 | 1 | 1 | 0.048 | 0.766 | 0.761 | 0.97 | 0.839 | 0.94 | 0.857 | 0.828 | 0.833 | 0.985 |
| | Recall | 1 | 1 | 1 | 0.5 | 0.947 | 1 | 1 | 1 | 1 | 0.857 | 1 | 0.972 | 1 |
| | F1-score | 1 | 1 | 1 | 0.087 | 0.847 | 0.864 | 0.985 | 0.913 | 0.968 | 0.857 | 0.906 | 0.897 | 0.993 |
| Area 6 | Precision | 1 | 1 | 1 | 1 | 0.926 | 0.87 | 0.937 | 1 | 1 | 0.6 | 0.83 | 0.848 | 1 |
| | Recall | 1 | 1 | 1 | 0.978 | 0.926 | 0.964 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | F1-score | 1 | 1 | 1 | 0.989 | 0.926 | 0.915 | 0.968 | 1 | 1 | 0.75 | 0.907 | 0.915 | 1 |

In order to analyse the results in Table 33 in more depth, we went through the scenes in which there was a misalignment between the predicted classes and the ground truths. The number of misclassified points (which lead to a wrong annotation prediction) differs from scene to scene. In most cases the number of points is very low (less than 1000), while in very few cases this number is higher than 10000. To be clear, when we mention misclassified points here, we only refer to the points belonging to an object that was wrongly classified during the annotation step. It should be also stated that most of the misalignments were False Positives (labels that were not part of the scene but classified as they were). Given that each scene consists of millions of points, in most cases, it is even hard to visually understand where this mismatch exists. Given the analysis above, these False Positive predictions could be reduced by adding a threshold to each category. Specifically, given the amount of points for each scene, it is highly unlikely that an object will consist of less than, for example, 5000 points. Therefore, a threshold close to that could provide even more accurate results.

Following that idea, we conducted one more experiment, in which we used a threshold equal to 5000 in order to discard the misclassified annotations with low number of points. For example, if in the predictions the object “chair” wrongly exists, we count the number of misclassified points and we discard this annotation if this number is less than 5000. As can be seen in Table 34 by using this heuristic approach, the results have improved.

Table 34: Annotation results for all areas of S3DIS dataset using 5000 threshold

| TEST AREA | METRIC | CEILING | FLOOR | WALL | BEAM | COLUMN | WINDOW | DOOR | TABLE | CHAIR | SOFA | BOOKCASE | BOARD | CLUTTER |
|-----------|-----------|---------|-------|------|-------|--------|--------|-------|-------|-------|-------|----------|-------|---------|
| Area 1 | Precision | 1 | 1 | 1 | 0.895 | 0.867 | 0.966 | 1 | 1 | 1 | 1 | 0.97 | 1 | 1 |
| | Recall | 1 | 1 | 1 | 1 | 0.929 | 1 | 1 | 1 | 1 | 1 | 0.97 | 1 | 1 |
| | F1-score | 1 | 1 | 1 | 0.944 | 0.897 | 0.983 | 1 | 1 | 1 | 1 | 0.97 | 1 | 1 |
| Area 2 | Precision | 1 | 1 | 1 | 0.47 | 0.5 | 0.6 | 0.974 | 0.917 | 0.957 | 0.435 | 0.74 | 1 | 1 |
| | Recall | 1 | 1 | 1 | 0.9 | 0.714 | 1 | 1 | 1 | 1 | 1 | 1 | 0.89 | 1 |
| | F1-score | 1 | 1 | 1 | 0.62 | 0.588 | 0.75 | 0.987 | 0.957 | 0.978 | 0.6 | 0.85 | 0.94 | 1 |
| Area 3 | Precision | 1 | 1 | 1 | 0.72 | 0.6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Recall | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | F1-score | 1 | 1 | 1 | 0.84 | 0.75 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Area 4 | Precision | 1 | 1 | 1 | 0.5 | 0.727 | 0.923 | 0.978 | 0.9 | 1 | 0.857 | 0.85 | 0.875 | 1 |
| | Recall | 1 | 1 | 1 | 1 | 0.889 | 0.923 | 0.978 | 1 | 0.97 | 0.75 | 1 | 0.78 | 1 |
| | F1-score | 1 | 1 | 1 | 0.67 | 0.8 | 0.923 | 0.978 | 0.947 | 0.985 | 0.8 | 0.921 | 0.82 | 1 |
| Area 5 | Precision | 1 | 1 | 1 | 0.143 | 1 | 0.946 | 1 | 1 | 0.979 | 1 | 0.94 | 0.972 | 1 |
| | Recall | 1 | 1 | 1 | 0.5 | 0.947 | 1 | 1 | 1 | 1 | 0.86 | 1 | 0.972 | 1 |
| | F1-score | 1 | 1 | 1 | 0.22 | 0.972 | 0.97 | 1 | 1 | 0.99 | 0.923 | 0.97 | 0.972 | 1 |
| Area 6 | Precision | 1 | 1 | 1 | 1 | 0.96 | 0.931 | 0.978 | 1 | 1 | 0.6 | 0.92 | 0.964 | 1 |
| | Recall | 1 | 1 | 1 | 0.978 | 0.926 | 0.964 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | F1-score | 1 | 1 | 1 | 0.989 | 0.943 | 0.947 | 0.989 | 1 | 1 | 0.75 | 0.958 | 0.982 | 1 |

In Figure 26, several examples of annotation mismatching can be found. In the first row, we can see an example of the scene for which 2720 points were misclassified as columns instead of ceiling, wall and bookcase. It goes without saying that it is quite normal for the network to classify the area in the bounding box as column instead of (mostly) wall as it should have done. In the second row, we can see an example, for which 727 points are misclassified as beam instead of wall. Looking at the input though, we can comment that there is something like a fuse box, which in the ground truth is classified as a wall. Therefore, it is normal that the network classifies it as beam and not wall. In the third row, most of the points of the table inside the bounding box were classified as “bookcase”. In the fourth row, very few points with “clutter” and “bookcase” as ground truth, were classified as “beam”. Finally, in the last row some points were wrongly classified as “sofa” instead of “floor” and “clutter”.

The analysis above is related to the scene-level annotation. Regarding the Semantic segmentation results (point-wise) in the same Figure, it is obvious that there is a lot of mismatching in the point classification results. For example, in the first row, the entire wall close to the window is wrongly classified as a window. As we have mentioned before though, the scope of this pipeline is the final annotation results rather than the implicit semantics.

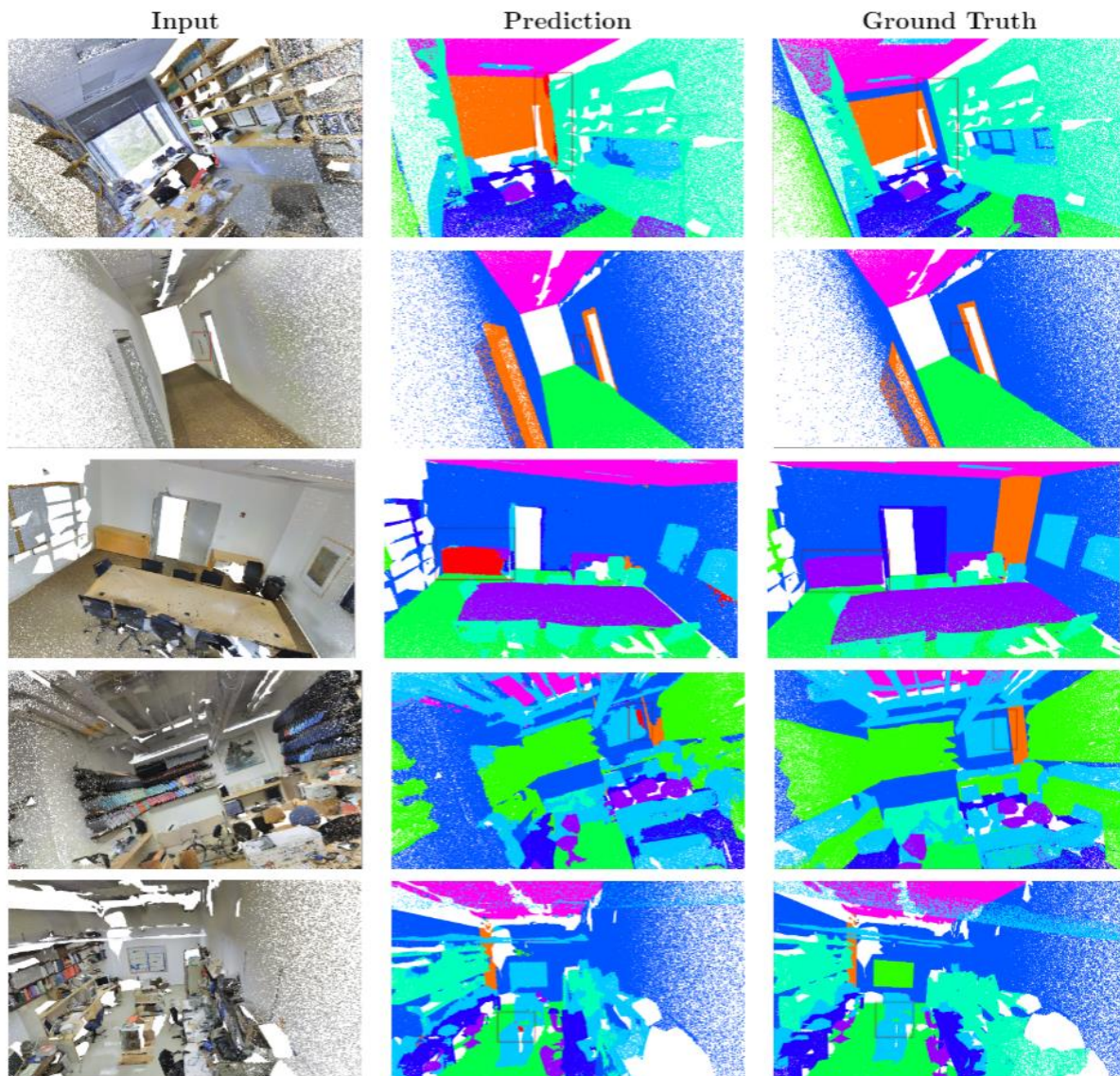


Figure 26: Results of annotation mismatch

2.3 Technical Implementation & API

For the deployment of the models, we used Nvidia Triton Inference Server³². Triton runs models concurrently on GPUs to maximize utilization, supports CPU-based inferencing, and offers advanced features like model ensemble and streaming inferencing. Supports all major frameworks like TensorFlow, TensorRT, PyTorch, ONNX Runtime, and even custom framework backends. Also, it can load models from local storage or cloud platforms. As models are retrained with new data, developers can easily make updates without restarting the inference server or disrupting the application. To load the models, a config file must be made in .pbtxt format. This would contain the platform that is being used (i.e. TensorFlow, PyTorch etc), max batch size and the Input / output schema for the data, along with the names of first and last layers. An example is given in Table 35. Then, the config file along with the saved_model will be loaded by the Triton Server using Docker.

³² <https://developer.nvidia.com/nvidia-triton-inference-server>

Table 35: Example of a config.pbtxt file

```
platform: "tensorflow_savedmodel"
max_batch_size: 8
input [
  {
    name: "base_input"
    data_type: TYPE_FP32
    dims: [224, 224, 3]
  }
]
output [
  {
    name: "classifier_low_dim"
    data_type: TYPE_FP32
    dims: [8631]
  }
]
```

To communicate with the server, Nvidia implements HTTP (REST) and gRPC protocols. We used the gRPC protocol as it implements client libraries for most languages, has easy support for streams and it is much faster than a RESTful service. Its' main notion is like calling a method on a server application on a different machine as if it were a local object. As in many RPC systems, gRPC is based around the idea of defining a service, specifying the methods that can be called remotely with their parameters and return types. On the server side, the server implements this interface and runs a gRPC server to handle client calls. On the client side, the client has a stub (referred to as just a client in some languages) that provides the same methods as the server.

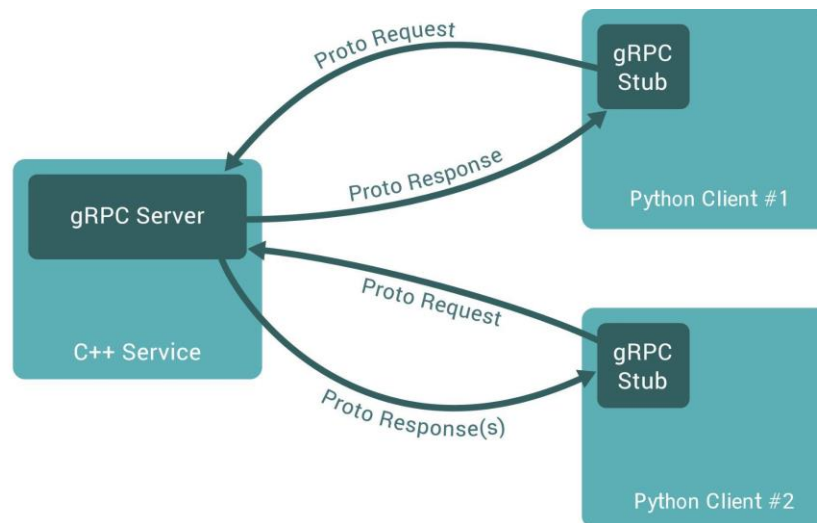


Figure 27: gRPC communication flow (Google. 2021)

In order to serve a model using gRPC we must define the schema of the requests and responses. That schema is written inside a Protocol buffer file (.proto). Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data (Google, 2001). Protocol buffer data is structured as messages, where each message is a small logical record of information containing a series of name-value pairs called fields. An example is shown in Table 36.

Table 36: Example of Protobuf file

```

syntax = "proto3";

package face_recognition;

message FaceRecognitionResponse {
    // Value of 0 indicated failure and 1 success
    int32 success = 1;

    // If set, success will be 0 and this field will describe the error that happened.
    string error = 2;

    // Number of faces identified
    int32 faces_identified = 3;

    // Number of celebrities
    int32 celebrities = 4;

    // Unidentified persons
    int32 unidentified = 5;

    // A list of the face recognised
    repeated FaceRecognitionResult identities = 6;
}

message FaceRecognitionResult {

    // 0 means the image contains a celebrity, 1 contains a face but the identity is
    // unknown
    int32 is_celeb = 1;

    // Name of the celebrity identified
    string celeb_name = 2;

    // The confidence of the classification.
    float confidence = 3;
}

```

Once the data structures are defined, the protoc compiler will be used to generate the data access classes in the preferred language(s) from the proto definition. These provide simple accessors for each field, like name() and set_name(), as well as methods to serialize/parse the whole structure to/from raw bytes. So, for instance, if your chosen language is Python, you must install the grpcio-tools package and run the compiler with the command on Table 37. For this example, it will generate the classes FaceRecognitionResponse and FaceRecognitionResult. You can then use these classes in your application to populate, serialize, and retrieve the corresponding protocol buffer messages.

Table 37: Command to run protoc compiler

```
python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. <filename>.proto
```

Every model needs to implement a handler that takes three arguments as shown in Table 38.

- buff: An IO buffer that contains the raw bytes read from the downloaded asset.
- triton_client: An instance of a grpc triton client, to be used to query the triton inference server where the model is loaded.
- metadata: A metadata object that is passed intact from the request and the handler is expected to pass it back in the response.

Table 38: Function example of the buffer

```
def new_model_name_handler(
    buff: IO,
    triton_client: grpcclient.InferenceServerClient,
    metadata: Dict[str, str]
):
    # code
```

The handler wraps the logic for each model and connects to the endpoint of the inference server. In Figure 28, the general flow of our system is shown. The general process is very straightforward and easily reusable. Updating a model requires only a new config.pbtxt file and the corresponding saved model. You can find details for each service in Annex I: Implementation details of API

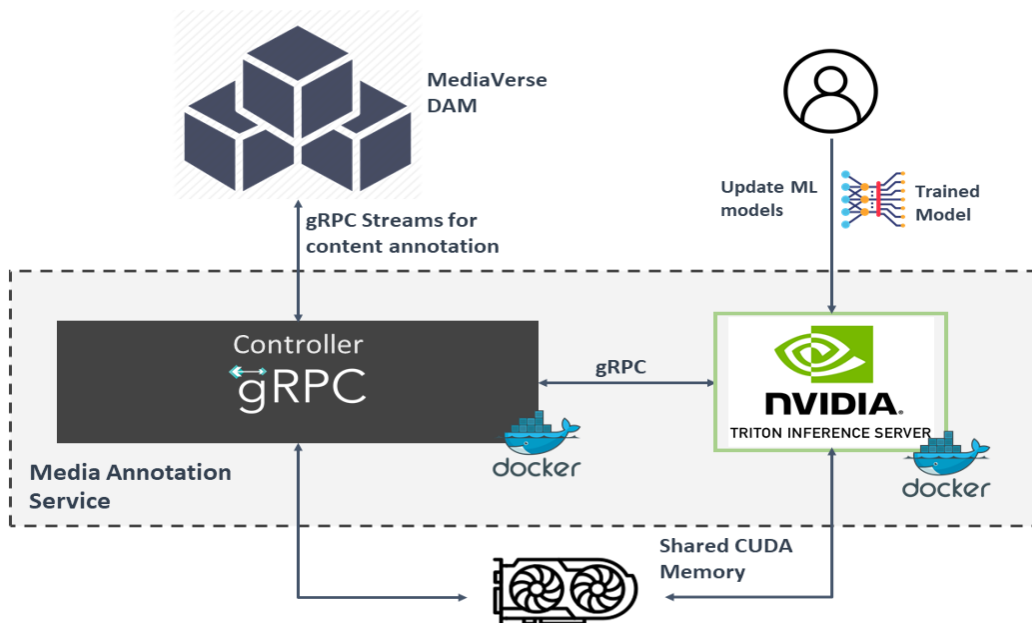


Figure 28: Flow diagram of MV-annotation service

3 Multilingual Annotation Tool

As part of contribution for T3.1, LINKS provides a multilingual annotation tool called *Transner* to be applied on social media posts. This tool is able to extract named entities from text in different languages. *Transner* is applied on both T3.1 and T5.2 with different aims. For the sake of T3.1, the tool is meant to be applied on social media posts in order to extract useful information for textual content understanding. In the following subsections we describe what is Named Entity Recognition, which features our tool includes, and how to use the provided service via REST API interface.

3.1 Named Entity Recognition

Named Entity Recognition (NER) is an information retrieval technique that aims to locate and classify named entities in natural language text documents. A named entity is a real-world object (physical or abstract) that can be identified with a proper name and that belongs to a certain category. Examples of named entities are Isaac Newton, Google, the Sahara Desert that represent a person, an organization, and a location respectively.

A named entity has a type and a value. The value represents the actual instantiation of that entity type. For instance, in the sentence “Albert Einstein was a famous physicist”, Albert Einstein belongs to the class PERSON, thus representing an instance of PERSON entity type having Albert Einstein as a value.

A comprehensive and standardized schema of all the entity types has not been released as far as we know and its definition highly depends on the application scenario. Some examples of entity types commonly used are PERSON, ORGANIZATION, LOCATION, DATES, MEDICAL CODES, and PERCENTAGES.

In a NER scenario we aim to achieve two goals:

1. The named entity has to be localized in the text (i.e., text span)
2. The named entity has to be classified based on the predefined schema of categories

The solution to this task is still discussed today in the research community. Software solutions typically rely on carefully annotated data and custom deep learning models.

NER is widely used in information retrieval pipeline, typically following Part-Of-Speech Tagging³³, and its results are used to perform Named Entity Linking (NEL)³⁴ that links the natural language span in the text, located and classified by NER, to a URI (i.e., Wikipedia URI) that uniquely identifies the entity.

In a full information retrieval pipeline, the last step is Relation Extraction³⁵ that has the final aim to build a Knowledge Graph (KG)³⁶. A Knowledge Graph is a data structure used to store factual data about the world where each node is an entity and each edge is a relation that connects different entities together (such as the entities “Barack Obama” and “Honolulu” are connected through the “has birthplace” relation).

³³ Neunerdt, Melanie, Bianka Trevisan, Michael Reyer, and Rudolf Mathar. "Part-of-speech tagging for social media texts." In *Language Processing and Knowledge in the Web*, pp. 139-150. Springer, Berlin, Heidelberg, 2013.

³⁴ Shen, Wei, Jianyong Wang, Ping Luo, and Min Wang. "Linden: linking named entities with knowledge base via semantic knowledge." In *Proceedings of the 21st international conference on World Wide Web*, pp. 449-458. 2012.

³⁵ Bach, Nguyen, and Sameer Badaskar. "A review of relation extraction." *Literature review for Language and Statistics II 2* (2007): 1-15.

³⁶ Ji, Shaoxiong, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. "A survey on knowledge graphs: Representation, acquisition and applications." *arXiv preprint arXiv:2002.00388* (2020).

NER can also be used in modern companies to extract customer names, contact information, and other related things from customers' support tickets and customer feedback, providing a faster customer care process. Another important application is in web search engines. Search engines heavily rely on the detection of entities inside a user query to retrieve the main resources about that entity. Entities need to be disambiguated and linked to a specific ID (i.e., URI) that uniquely identifies them on the Web (or on a sub-domain). For instance, Wikipedia provides URI for each resource it contains to distinguish among homonyms entities (i.e., different people with the same name). A visual example of NER applied on a piece of text from Wikipedia is shown in Figure 29.

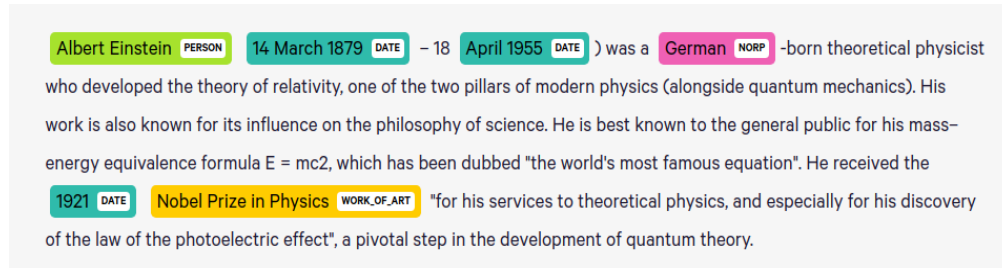


Figure 29: Results of NER applied on a piece of a Wikipedia article with the tool <https://explosion.ai/demos/displacy-ent>

3.2 Description of the Tool

This subsection describes the tool we released together with the list of recognized named entities, supported languages, and qualitative results on a Wikipedia-based dataset.

3.2.1 Transner

Transner is a model for Named Entity Recognition that uses both deep learning and rule-based methodologies. The deep learning of *Transner* is represented by a transformer³⁷ architecture that is used to create a deep learning model from an annotated dataset to correctly detect and classify named entities in a text. The rule-based side instead is used to create a rule that can capture the standard shape of a particular entity type (regular expression) or query a list of all possible values that a particular entity type can assume (Gazetteers).

Transner is able to detect location nested-entities by using a Gazetteer of worldwide cities and countries. An example of a location nested entity is provided in Figure 30.

The deep learning model was trained with multilingual BERT³⁸ that represents the state-of-the-art Transformer model for multilingual NER. The output of the Transformer model is then fed to a linear layer with a softmax activation function to output the probability distribution of each word to be part of one of four entity types: PERSON, ORGANIZATION, LOCATION, MISCELLANEOUS. The other entity types are handled with two rule-based approaches: regular expressions and Gazetteers. Regular expressions are used for entities with standard shapes, such as SSID or ipv4, and they are integrated with the result of the deep learning approach.

Gazetteers instead are queried after regular expressions to detect religion entity type and location sub-entities (i.e., nested entities). Sub-entities are searched inside miscellaneous entities only to add a fine-grained classification of this macro class. The search of sub-entities is performed by querying the Gazetteer with all the substrings inside a miscellaneous entity to find matches.

³⁷ Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

³⁸ Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

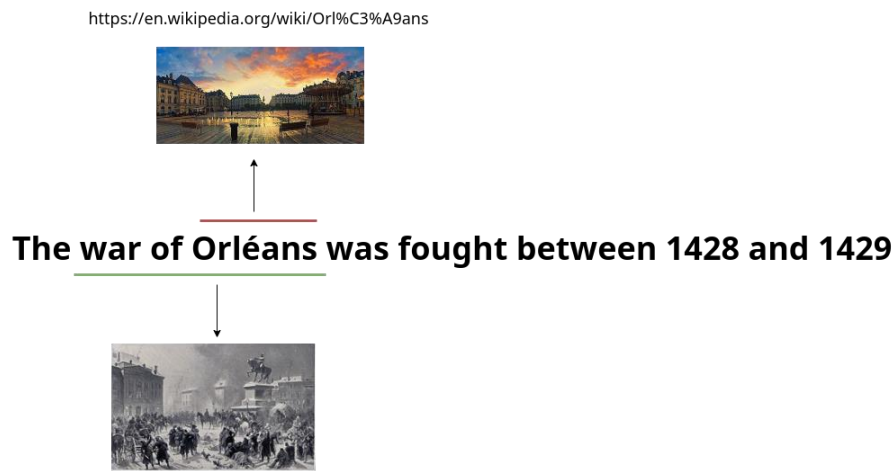


Figure 30: Example of nested entities where the word Orléans represents both a historical event and to location. The task of nested entities is to recognize the whole entity “war of Orléans” as a historical event and the sub-entity “Orléans” as a real LOCATION.

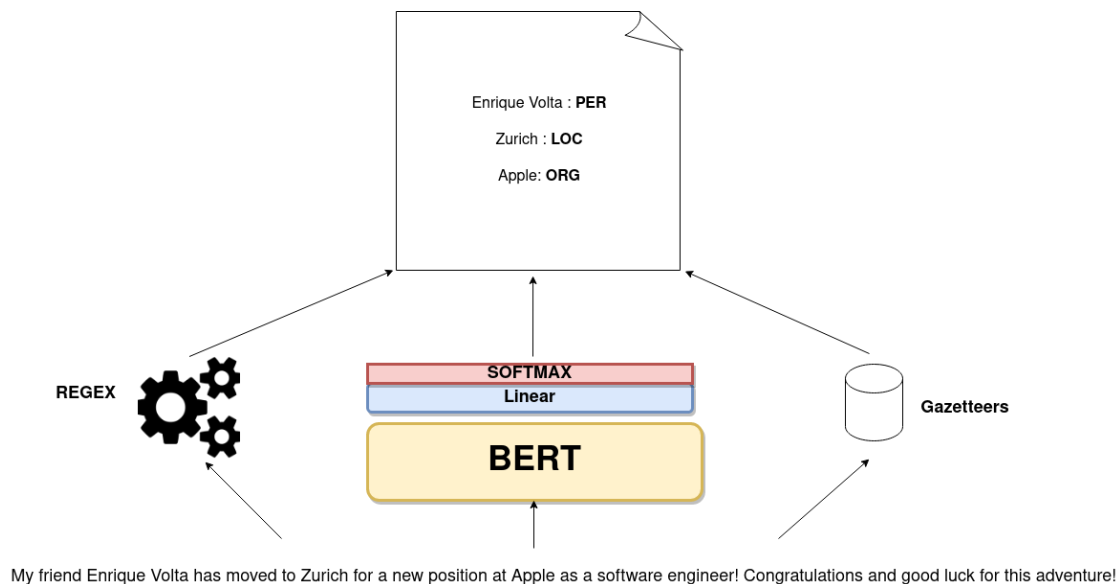


Figure 31: Transner main components

As depicted in Figure 31 Transner is composed of three main components: Transformer for NER, regular expressions, and Gazetteers. The left side of the figure represents the rule-based component made of regular expressions. Regular expressions are used to classify those entities that have a fixed shape (i.e., fiscal code, ipv4). The central branch represented in this figure is the Transformer with a linear classifier on top. The linear classifier receives as input the contextual embeddings produced by the Transformer (BERT) and outputs logits. The logits are then fed to a softmax activation function that outputs the probability distribution of each token to belong to a specific entity type. The right side of the figure represents the rule-based component made of Gazetteers. The Gazetteers are used to classify those entities that can assume only a finite set of values. Transner uses Gazetteers to classify location sub-entities and religions.

3.2.2 List of Recognized Entities

- PERSON
- LOCATION
- ORGANIZATION
- IT_FISCAL_CODE
- EU_IBAN
- NL_CITIZEN_SERVICE_NUMBER
- UK_NATIONAL_ID_NUMBER
- EU_PHONE_NUMBER
- EMAIL_ADDRESS
- IPV4_ADDRESS
- RELIGION
- MISCELLANEOUS (entities belonging to a category not present in the schema)

3.2.3 List of Supported Languages

- Italian
- English
- Dutch

3.2.4 Quantitative Results

Here, we report the qualitative results over precision, recall, and F1 on WikiNER dataset³⁹. The qualitative analysis is computed on the Deep Learning component of *Transner* only, thus on PER, LOC, ORG, and MISC named entities. WikiNER is a multilingual dataset containing named entity annotations for the four categories reported before. In Table 39, we can have a look at the dimension of the dataset, together with the number of entities for each different category. In Table 40, we report the qualitative results over the mentioned metrics.

Table 39: Statistics for the WikiNER portion of data we used to assess the quality of the annotation tool

| LANGUAGE | NO. TOKENS | NO. PER | NO. LOC | NO. ORG | NO. MISC | NO. SENTENCES | AVG. SENTENCES |
|----------|------------|---------|---------|---------|----------|---------------|----------------|
| IT | 45,411 | 12,699 | 14,571 | 5,689 | 6,572 | 18,793 | 28.49 |
| EN | 40,504 | 13,494 | 13,472 | 7,638 | 9,705 | 21,226 | 24.25 |
| NL | 47,537 | 13,570 | 22,263 | 5,488 | 8,608 | 26,934 | 18.99 |
| AVG | 133,452 | 39,763 | 50,306 | 18,815 | 24,885 | 66,953 | - |

³⁹ Nothman, Joel, et al. "Learning multilingual named entity recognition from Wikipedia." *Artificial Intelligence* 194 (2013): 151-175.

Table 40: Quantitative results over precision, recall, and F1 metrics

| ENTITIES | PRECISION | RECALL | F1 |
|------------|-----------|--------|------|
| PER | 0.94 | 0.94 | 0.94 |
| ORG | 0.87 | 0.82 | 0.85 |
| LOC | 0.91 | 0.92 | 0.91 |
| MISC | 0.76 | 0.82 | 0.79 |
| micro avg. | 0.76 | 0.82 | 0.79 |

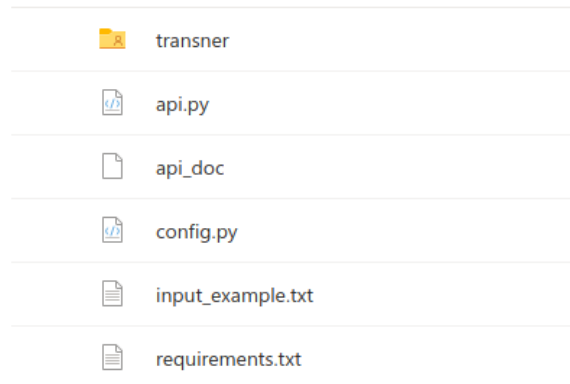
3.3 API Description

Transner is exposed via REST API interface to allow a flexible integration in the ecosystem. In this subsection we describe more pragmatically how to use the annotation tool. We describe how to download the tool and its project folder, how to correctly setup the environment, and how to use the tool for text annotation.

3.3.1 Project folder

The folder of *Transner* available at <https://bit.ly/3hJvhqb> contains a folder, a series of python files and some additional text files to provide examples of usage.

- “api.py” is the Python file containing the REST API interface.
- “api_doc” contains an example of input and output for *Transner* REST API.
- “config.py” is a file containing some configurations.
- “input_example.txt” provides some examples of input structure for the REST API.
- “requirements.txt” contains the Python packages to download for the setup of the virtual environment (see section Installation and environment setup).









| | |
|---|-------------------|
|  | transner |
|  | api.py |
|  | api_doc |
|  | config.py |
|  | input_example.txt |
|  | requirements.txt |

Figure 32: Folders structure of *Transner*

The *Transner* folder contains the source code of the module and Gazetteers for religions and cities in the world.

- “multilang_uncased” folder contains the model binaries.
- “utils” contains a Python module with preprocessing tools used by *Transner*.
- “worldcities” is a folder containing the Gazetteers for the worldwide locations used for nested entities.
- “benchmarks.json” contains the links to the .csv containing the scores of *Transner*.
- “ner.py” is the Python source code of *Transner*.
- “religions.txt” is the Gazetteer for religions in the world (Italian with v0.4).

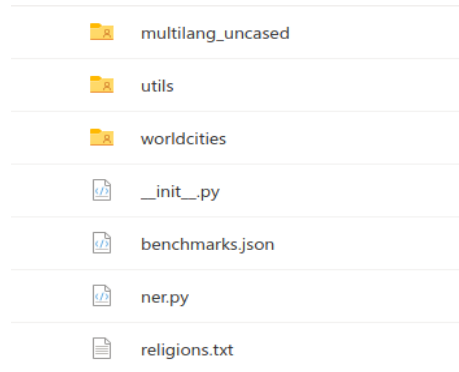


Figure 33: Structure of the folder called *Transner*

3.3.2 Installation and Environment Setup

Transner is exposed via REST API through the Flask library. In this section, we describe how to download the source code and setup the environment for the correct use of *Transner* on a Linux machine.

1. Go to <https://bit.ly/3hJvhqb> a page like this should appear. Then click “Download”.

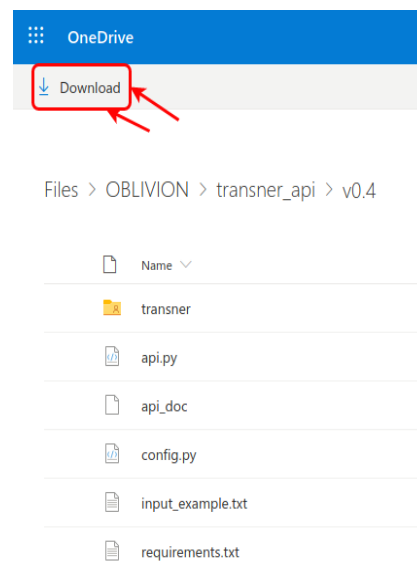


Figure 34: OneDrive page for project download

2. A file named “v0.4.zip” should have been downloaded. Go into your Download folder and extract the zip. We use 7zip on a Linux machine.

```
[matte@matte-arch: Downloads]$ 7z x v0.4.zip

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016
-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on
,64 bits,16 CPUs AMD Ryzen 7 3700X 8-Core Processor
(870F10),ASM,AES-NI)

Scanning the drive for archives:
1 file, 2006324639 bytes (1914 MiB)

Extracting archive: v0.4.zip
--
Path = v0.4.zip
Type = zip
Physical Size = 2006324639

Everything is Ok

Files: 22
Size:      2006321089
Compressed: 2006324639
```

Figure 35: Project unzip with 7zip

- Go into the folder “v0.4/” and create a python virtual environment with Python 3.7+. We will show the procedure setup a virtual environment with Pipenv which is the one suggested in the Python documentation (you can see more here <https://pipenv.pypa.io/en/latest/>).

```
[matte@matte-arch: Downloads]$ cd v0.4
[matte@matte-arch: v0.4]$ pipenv shell --python 3.7
Creating a virtualenv for this project...
Pipfile: /home/matte/Downloads/v0.4/Pipfile
Using /usr/bin/python3.7 (3.7.7) to create virtualenv...
: Creating virtual environment...created virtual environment
CPython3.7.7.final.0-64 in 878ms
creator CPython3Posix(dest=/home/matte/.local/share/virtual
envs/v0.4-HA4Z0e8_, clear=False, global=False)
seedler FromAppData(download=False, pip=latest, setuptools=l
atest, wheel=latest, via=copy, app_data_dir=/home/matte/.loca
l/share/virtualenv/seed-app-data/v1.0.1)
activators BashActivator,CShellActivator,FishActivator,Power
ShellActivator,PythonActivator,XonshActivator

✓Successfully created virtual environment!
Virtualenv location: /home/matte/.local/share/virtualenvs/v0.
4-HA4Z0e8_
requirements.txt found, instead of Pipfile! Converting..
✓Success!
Warning: Your Pipfile now contains pinned versions, if your r
equirements.txt did.
We recommend updating your Pipfile to specify the "*" version
, instead.
Launching subshell in virtual environment...
. /home/matte/.local/share/virtualenvs/v0.4-HA4Z0e8_/bin/act
ivate
[matte@matte-arch: v0.4]$ . /home/matte/.local/share/virtual
envs/v0.4-HA4Z0e8_/bin/activate
(v0.4) [matte@matte-arch: v0.4]$
```

Figure 36: Creation of the Python virtual environment for the project with pipenv

- Now that your virtual environment is created you can set up the environment by installing the “requirements.txt” via pip.
- If everything succeeded, you now have a working Python environment for using the *Transner* REST API.


```
(v0.4) [matte@matte-arch: v0.4]$ pip install -r requirements.txt
Collecting boto3==1.12.21
  Using cached boto3-1.12.21-py2.py3-none-any.whl (128 kB)
Collecting botocore==1.15.21
  Using cached botocore-1.15.21-py2.py3-none-any.whl (6.0 MB)
Collecting certifi==2019.11.28
  Using cached certifi-2019.11.28-py2.py3-none-any.whl (156 kB)
Collecting chardet==3.0.4
  Using cached chardet-3.0.4-py2.py3-none-any.whl (133 kB)
Collecting click==7.1.1
  Using cached click-7.1.1-py2.py3-none-any.whl (82 kB)
Collecting docutils==0.15.2
  Using cached docutils-0.15.2-py3-none-any.whl (547 kB)
Collecting filelock==3.0.12
  Using cached filelock-3.0.12-py3-none-any.whl (7.6 kB)
Collecting Flask==1.1.1
  Using cached Flask-1.1.1-py2.py3-none-any.whl (94 kB)
Collecting Flask-Cors==3.0.8
  Using cached Flask_Cors-3.0.8-py2.py3-none-any.whl (14 kB)
Collecting h5py==2.10.0
  Using cached h5py-2.10.0-cp37-cp37m-manylinux1_x86_64.whl (2.9 MB)
Collecting idna==2.9
```

Figure 37: Installation of the requirement for the project

3.3.3 How to Use

In this section, we describe how to run and use the *Transner* service exposed via REST API.

1. Go inside the source folder of *Transner* and activate the Python environment if not yet done.

```
[matte@matte-arch: v0.4]$ pipenv shell
Launching subshell in virtual environment...
. /home/matte/.local/share/virtualenvs/v0.4-HA4Z0e8_/bin/activate
[matte@matte-arch: v0.4]$ . /home/matte/.local/share/virtualenvs/v0.4-HA4Z0e8_/bin/activate
(v0.4) [matte@matte-arch: v0.4]$
```

Figure 38: Figure 38: Activation of the Python virtual environment with pipenv

2. Run the REST API inside the “api.py” file. The API is exposed on port 5000, it must be free to allow *Transner* to run correctly.

```
(v0.4) [matte@matte-arch: v0.4]$ python api.py
* Serving Flask app "api" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a
  production deployment.
  Use a production WSGI server instead.
* Debug mode: on
```

Figure 39: Figure 39: Running of Transner with Python

3. You can query the API on port 5000 by sending an input that is a JSON array of strings. Each string represents a sentence you want *Transner* to annotate. Some input examples are present in the “input_example.txt” file in the *Transner* folder. We show how to send a request to the API through curl command available for Linux machines (for reference <https://linux.die.net/man/1/curl>)

```
[matte@matte-arch: ~]$ curl -i -H "Content-Type: application/
json" -X POST -d '{"strings": ["Maria Santos è nata a Cardena
s il 13/08/1983", "The following documents were signed by Joh
n Stewart at Berlin headquarters of Deutsche Bank", "Bevestig
ing van betaling aan ABN AMRO door dhr. Rutger Verhoeven."]}'
http://localhost:5000/transner/v0.3/ner
```

Figure 40: Making the call to Transner REST API

- The response of *Transner* is a JSON list of results (one for each input sentence). Each element of the object is composed of the original sentence and an array of entity objects. Each entity object consists of three fields: value, type, and offset. The value is the current substring in the text representing the entity. The type represents the type of the entity (see section Entity Types to know the supported entity types). Offset is the position in the original sentence where the entity is located.

```
{
  "entities": [
    {
      "offset": 0,
      "type": "PERSON",
      "value": "Maria Santos"
    },
    {
      "offset": 22,
      "type": "LOCATION",
      "value": "Cardenas"
    }
  ],
  "sentence": "Maria Santos è nata a Cardenas il 13/08/1983"
},
{
  "entities": [
    {
      "offset": 39,
      "type": "PERSON",
      "value": "John Stewart"
    },
    {
      "offset": 55,
      "type": "LOCATION",
      "value": "Berlin"
    },
    {
      "offset": 78,
      "type": "ORGANIZATION",
      "value": "Deutsche Bank"
    }
  ],
  "sentence": "The following documents were signed by John Stewart at Berlin headquarters of Deutsche Bank"
},
{
  "entities": [
    {
      "offset": 29,
      "type": "ORGANIZATION",
      "value": "ABN AMRO"
    },
    {
      "offset": 48,
      "type": "PERSON",
      "value": "Rutger Verhoeven"
    }
  ],
  "sentence": "Bevestiging van betaling aan ABN AMRO door dhr. Rutger Verhoeven."
}
]
```

Figure 41: Result of the Transner REST API

4 Content Discovery and Interlinking services

4.1 Introduction

Content discovery and interlinking are one of the most important features of the social media platforms web companies have built in the last 10 years. Platforms like YouTube and Instagram are heavily based on interlinking of contents, and without them, content creator's discovery would have been difficult and frustrating from both the user and creator perspectives. On YouTube, new recommendations of videos are provided to the user based on his/her history. For instance, a user watching videos about football will probably get recommendations of other videos regarding the same topic. Using this approach, YouTube is able to recommend contents based on the preferences of the final user. Differently, Instagram and Twitter use the concept of hashtag, with which it is possible to interlink different posts and tweets together. The insertion of a hashtag is delegated to the user that creates the content. By grouping together contents sharing the same hashtags, these platforms give the possibility of discovering similar content whenever the user searches for them.

In MediaVerse we start with the idea of interlinking content based on deeper relationships present in the content itself. For instance, if two pictures both represents a giraffe but their hashtags are misaligned (i.e., #zoo and #sunnyday, #animal) we still want to create a connection between them. This connection can be created using the characteristics of the content itself and not the description inserted by the user, thus exploiting the semantics of the content. Interlinking contents at the semantic level allows to build more reliable and richer search and recommendation infrastructure that can be properly used to prepare ready-to-be-consumed content packages for the final user.

Additionally, we also want to address the interlinking challenges in a way that can overcome the limitations that can arise due to the presence of different types of content. For instance, using again the example of the giraffe, we do not want to limit the connection between two images containing a giraffe, instead we also want to build a connection between an image and a textual post that deals with giraffes. In order to build an interlinking service able to go beyond the single modality (i.e., vision, language) we require to have specific modules for dealing with visual contents. In this document, we will refer to this feature as “cross-modality”.

In T3.2 we build the services to provide the functionality of cross-modal content linking based on their semantics. For the sake of this work, we define two different ways of preparing ready-to-be-consumed content packages from the user's perspective: a proactive way and a passive way. The proactive way consists in the user searching for content in MediaVerse using a query, we will refer to it as “retrieval”. The passive way instead is the typical recommendation scenario, where the user is recommended with content from the platform itself. We will refer to this last way as “recommendation”. To address both, in T3.2 we implement a tool that:

1. does cross-modal retrieval of content within the MediaVerse network
2. does content recommendation

In the following chapters, we describe the usage scenario of T3.2, the methodology we used to address the challenges, the technical choice from the implementation perspective, and, finally, the description of the REST API we developed to expose the service.

4.2 Usage Scenario and Requirements

As described in Section 4.1, we envision two main ways the interlinking services must work in the MediaVerse platform that are the retrieval and recommendation of contents. For this reason, we describe two different usage scenarios. The first scenario is related to content retrieval; thus, the user is proactive in content searching. The second scenario addresses the recommendation case, where the user is a passive actor that is provided with the contents chosen by MediaVerse interlinking services. The two usage scenarios are reported in the first subsections below. The last subsection is instead devoted to the description of the requirements for T3.2.

4.2.1 Retrieval Scenario

John is an English student of photography interested in urban environments. He is interested in learning by watching new urban photos posted by worldwide photographers with more experience than him. John uses MediaVerse to share his photos, receive feedback, and search for other content that contains certain subjects and certain urban objects in them. One day, John sees a new spot to shoot his photo that seems, at the same time, challenging and inspiring. The spot contains a big clock at the center of a town square with a lot of people passing by. John, who is still a student, wants to see the works of others before deciding which types of settings to use. Thus, he opens the MediaVerse platform and searches for “photographs with a clock”. The MediaVerse engine searches and ranks different photos on the network based on their similarities with the search query. Finally, the photos appear on the screen ranked by similarity score. The photos contain clocks in indoor environments or clock towers. John is not yet satisfied since the environment is different from the one, he was looking for. Thus, he enriches the query by adding information about the square and the people walking by. He entered “a clock in the center of a square with people” as a query and press enter. The final results fit better with the needs of John. Now he can look for photos he likes and try to reproduce them by himself.

4.2.2 Recommendation Scenario

Giorgio is an Italian journalist interested in actuality. He is working on new content to tell the events that are happening during the last Covid demonstration in Rome. For this purpose, he is trying to create a 360 story using Fader. He has collected some photographs but he needs more material to compose a useful story. By uploading the images on MediaVerse, the MediaVerse recommendation engine can suggest, based on the contents of Giorgio’s photos, relevant content from other users that participated in the demonstration and uploaded their content on MV. Based on the initial photographs and the suggested ones, Giorgio can create an immersive story that describes the demonstration and the atmosphere that characterizes it.

4.2.3 Requirements

For the development of this section, we decided to divide the requirements of T3.2 (for a complete list of MediaVerse requirements please refer to D2.1) in two different tables.

Table 41 contains the SRCH-07 MediaVerse project requirement satisfied by our work on T3.2. More specifically, this requirement states that the search results on the MediaVerse platform can be ordered and down ranked according to certain criteria. As described in the Section 4.3 (i.e., Methodology), the interlinking service we provide returns a score for each content by taking in consideration the search query. Thus, the results can be easily ordered according to the value of the score.

Table 42 instead contains the requirements of T3.2 itself, thus highlighting the basic functionalities that our service needs in order to work properly. The first requirement is SRCH-01, a fundamental requirement stating

that the content can be searched across nodes. This is a critical functionality for a decentralized platform, without it the service we provide for T3.2 is limited to a single node, thus severely limiting its usefulness. The second requirement is ANNO-01, stating that the MediaVerse system applies tags to each new content. To better understand this requirement there is the need to refer to Section 4.3 (i.e., Methodology) where we describe how our service uses the annotations made in T3.1 to represent visual content. Without this functionality, the service will be limited to work on single modality (i.e., only text-based content) because it cannot interlink the query with items belonging to different modalities. Finally, the last requirement we list is PUBL-01 that allows creators to publish their content in the MediaVerse network.

Table 41: Requirement satisfied by T3.2

| REQUIREMENT | DESCRIPTION |
|----------------|--|
| SRCH-07 | Search results can be ordered/down ranked according to certain aspects |

Table 42: Requirements necessary for T3.2

| REQUIREMENT | DESCRIPTION |
|----------------|--|
| SRCH-01 | Content can be searched across nodes |
| ANNO-01 | The system will add tags to each new asset |
| PUBL-01 | Content creators can publish their content in the MediaVerse |

4.3 Methodology

Modern Artificial Intelligence is heavily focused on the development of mathematical models, extracted from data patterns, for solving particular tasks like classification or regression. This new paradigm goes under the name of Machine Learning. The mathematical model is derived directly from data using different algorithms; thus the process is completely data-driven. In the last decade, the attention of the AI community shifted from traditional techniques⁴⁰ to modern techniques based on deep neural networks⁴¹, creating a new subdomain that takes the name of Deep Learning. The idea behind Deep Learning is to learn a model that approximates probability distribution hidden in data patterns. Handling probability is possible to achieve state-of-the-art performance on tasks requiring visual understanding or language comprehension.

In T3.2, we use a pure data-driven approach exploiting modern techniques based on Deep Learning. In particular, we developed a model that is able to give a similarity score between a textual query and an image. This model is the core of the interlinking services developed in T3.2.

⁴⁰ Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik. "A training algorithm for optimal margin classifiers." *Proceedings of the fifth annual workshop on Computational learning theory*. 1992.

⁴¹ Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. *Learning internal representations by error propagation*. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

4.3.1 Training Procedure

The training procedure in Deep Learning refers to the operations performed to extract the mathematical model from the data. During this phase, we define the following components:

- a neural network architecture
- a dataset
- a loss function
- an algorithm to optimize the loss function

The neural network defines the aspect of our model (raw set of parameters and operations that links them), the dataset is the source of data we want to extract the model from, the loss function is a measure of how good our model performs on the training set. The final goal of the training procedure is to find a model (i.e., set of parameters) that minimizes the loss value, thus we need to choose an algorithm to optimize the loss we chose.

For the sake of T3.2, we opted for a training procedure specific for the development of a retrieval model. A retrieval model is a model able to understand if the selection of a particular item is appropriate or not under certain constraints. In our work, the query represents the constraint we use for deciding if an item is appropriate or not. This setup is still valid when we consider the recommendation scenario. In that case, the query is represented by the content the user is watching and based on this, we evaluate the appropriateness of recommending other particular contents.

The training setup we used is characterized as follows: a query Q , a set of items $E=\{e_1, \dots, e_n\}$ (i.e., our contents), and a scorer model. The scorer has to analyze Q and, based on that, assign a score to each e_i in E . The higher the score, the higher the appropriateness of the item given Q . We proceed to compose the series of pairs $S=\{(Q, e_1), (Q, e_2), \dots, (Q, e_n)\}$ characterized by the same query Q paired with all the different contents available. Among the contents, only some of them are the most appropriate to retrieve given Q , while the other contents have to be considered not relevant. Thus, we can distinguish two types of pair, a positive pair A^+ , that is the one with the relevant content, and a negative pair A^- containing the non-relevant content. Having these definitions in mind, we can assert the series S contains only one positive pair. During training, we build each pair in S with the following approach:

- 50% of times, randomly replace one of the element of a positive pair with a random one
- 50% of times, leave the positive pair as is

With this setup, we finally train our model to learn how to recognize the positive pairs within S .

While, at a first glance, this procedure seems not to be related to the task of assigning an appropriateness score to each content, we want to highlight that in our setup recognizing a positive pair means to assign a high score to that pair. Thus, we are teaching our model to assign the highest scores to pairs that have to be considered as positives. From a technical perspective, we set up the training as a classification task. Given an input pair $A=(Q, e_i)$ we use the model to approximate the probability distribution $P(A^+ | A)$, which is the probability that the pair A belongs to the set of positive pairs. If the produced probability is greater than 0.5, then we consider the model prediction as positive, otherwise we consider it as negative.

To also deal with the cross-modality feature, we build S to contain multi-modal pairs, which means that each pair $A=(Q, e_i)$ is composed of a textual component and a visual one. We do not impose any type of constraint on which is which, thus the query Q can be both a textual query or a visual one and the same for the item e_i .

Concerning the visual modality, we decide to avoid content-specific models that means to build a specific model for each possible content type in MediaVerse (i.e., images, video, 3D models) and we proceed to build one single model that can handle different contents. To build such a model we exploit the output of T3.1. In MediaVerse, each time a new content is uploaded on the network, this has to undergo the annotation services that extract useful metadata from it. Among these metadata, we decide to use the ones corresponding to the objects present in the content. Practically speaking, we replace each content different from text with its objects-related metadata. Using this approach, we turn the complex cross-modal scenario into a single-modality scenario where the only modality taken in consideration by the scorer model is text.

The architecture we select is composed of several pieces.

We used two bidirectional LSTM⁴² to encode the query and the content. The bidirectional LSTM is composed of two LSTMs, one encodes the input following a left-to-right direction, the other encodes right-to-left. For each LSTM, the left-to-right and right-to-left last hidden states are concatenated to form a dense representation. At this point, the two dense representations are concatenated together and fed to a fully connected neural network (FFN). This network applies affine transformations to produce a representation that contains the most important feature to understand if the two items in the pair are interlinked or not, thus if the pair belongs to the positive or to the negative class. The FFN contains affine layers followed by ReLU⁴³ non-linearities and normalization layers. After the FFN, a last affine transformation with softmax function is used to produce the probability distribution over the positive and negative classes.

The hidden size of the model was chosen to be 768 and we use Glove⁴⁴ pre-trained embeddings to provide a good word representation to the LSTMs. Since we modelled our task as a classification task, we chose the cross-entropy loss and we trained the model for 15 epochs and a batch size of 32. As an optimization algorithm we decided to use Adam optimizer⁴⁵ with a learning rate of 1e-6.

The architecture we used is depicted in Figure 42. The two sequences of words (text and annotations) are fed to two bidirectional LSTMs. The final representations are concatenated and fed to a final FFN that produces, thanks to a Softmax function, the probability distribution over the positive and negative class.

⁴² Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

⁴³ Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." *ICML*. 2010.

⁴⁴ Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.

⁴⁵ Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

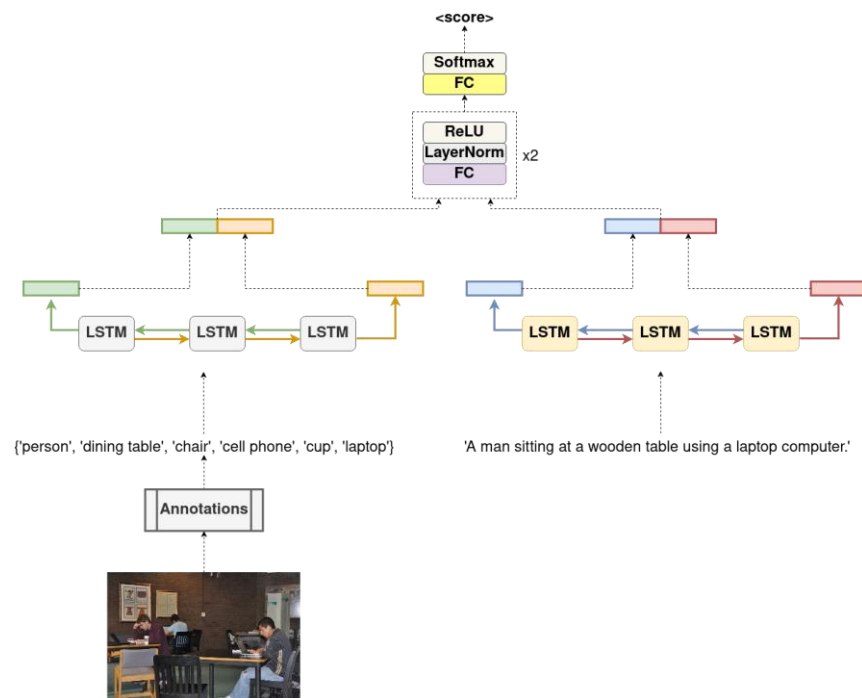


Figure 42: Neural Network architecture

4.3.2 Training Data

As training data, we used Microsoft COCO⁴⁶ (MSCOCO), a publicly available dataset for computer vision containing several annotations like objects bounding boxes and segmentation masks. We decided to choose this dataset for two main reasons:

1. cross-modality: each image in MSCOCO is annotated with up to 4 different textual captions.
2. objects metadata: each image in MSCOCO is annotated with objects belonging to a schema of 91 different categories.

The cross-modality nature of MSCOCO makes possible the creation of pairs of items belonging to different categories, as explained in 4.3.1. At the same time, the presence of object annotations makes it possible to simulate the MediaVerse platform in which each uploaded image is annotated with different tags. Furthermore, the annotations schema used in MSCOCO is the same used in MediaVerse (i.e., same object categories), thus representing an ideal set up for T3.2.

Each image in MSCOCO dataset contains an average of 7.27 annotated objects belonging to a set of 91 different categories. Concerning the textual captions, the average number of tokens for each caption, using the NLTK WordPiece tokenizer (please refer to 4.4), is 11.3.

As data pre-processing, we proceed to crop captions to a maximum size of 15 tokens and to substitute each image with the list of its object tags. Then, we proceed to build the positive and negative pairs. The positive pairs are formed by selecting the image and one of its four related captions. The negative pairs instead are obtained by taking a positive pair and applying a random substitution 50% of the time, as explained in 4.3.1. We substitute with equal probability the image or the caption in the pair with a random one inside the dataset.

⁴⁶ Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." *European conference on computer vision*. Springer, Cham, 2014.

For training, we used the COCO train split of 2017 and the respective validation split as a development set (i.e., to use for hyperparameters tuning). Since the annotations made in T3.1 were not yet available at the time of training, we trained our model using the ground truth objects annotated in MSCOCO dataset. Having made this decision does not allow us to use the official test set to report the results since it does not contain the ground truth object annotations. In order to report qualitative results, we divided the validation split into two parts, one part has to be used as a development set, and the other part as a test set for the qualitative metrics.

4.3.3 Results

After the model training, we assessed the quality of the tool using 4 different metrics defined in the literature: accuracy, precision, recall, and f1. The formal definition of these metrics is defined below using the notation from the confusion matrix.

$$Accuracy = \frac{TP + FN}{TP + FP + FN + TN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision}$$

In order to compute the results, we use the test split as described in the previous section and we feed the model with positive and negative pairs. We then compute the quality of the tool in discriminating between the positive and the negative. Since the negative pairs are obtained by a random substitution, we compute the average of the results over 10 different runs. The results are reported in Table 43.

We can see that the model obtains good results in the setup we created. The most informative metric is the f1 that is the harmonic mean of precision and recall. An f1 score of 0.9437 highlights the ability of the model in both precision and recall.

Additionally, the scores are stable given that the standard deviation is 2 magnitudes smaller than the mean value.

Table 43: Results on the test split averaged over 10 runs

| ACCURACY [M (̂)] | PRECISION [M (̂)] | RECALL [M (̂)] | F1 [M (̂)] |
|---------------------|----------------------|--------------------|--------------------|
| 0.9422 (0.0014) | 0.9690 (0.0011) | 0.9197 (0.0019) | 0.9437 (0.0014) |

4.3.4 Next Steps

In this section, we have described the current status of T3.2. To conclude the methodology part, we decide to list the limitations of the current approach together with the improvements we are introducing in the retrieval system to overcome them.

We foresee two major limitations of this approach:

1. Scalability: the neural network cannot be used to retrieve content from the entire MediaVerse network because of its computational cost.
2. Visual blindness: the retrieval system we presented is partially “blind” when dealing with visual content. Its blindness is due to considering only the tags of the objects contained in the image. This design choice results in the model’s inability to distinguish two images containing the same objects, thus limiting the precision of the tool.

Thanks to the metadata now available from T3.1, we can enrich the retrieval system with additional information for the different types of contents present in MediaVerse. Currently, we are proceeding in two directions:

1. usage of the traditional approaches for pre-filtering. Traditional approaches are more scalable and can be used to produce an initial list of candidates that will then be re-ranked using a neural network.
2. exploit of cross-modal representations provided by T3.1 to overcome the blindness of the ranker when dealing with visual content.

With these two directions in mind, we will measure the impact of these new features compared to the current method. Additionally, we are creating a new experimental setup similar to other approaches existing in literature to have a good quantitative understanding of the general performance of the retrieval system.

4.4 Implementation

As the programming language to use for the development of the service we chose Python. This choice was driven by the multitude of Machine Learning libraries available for this language. We used PyTorch⁴⁷ for the implementation of the neural network, a Deep Learning library that offers automatic differentiation, basic blocks for neural networks, GPU capabilities, and other features that allow us to produce complex architectures using only the Python language.

For text processing we used the NLTK library⁴⁸, a library that provides several features to process and clean text. In particular, we used the *WordTokenizer* to tokenize textual data.

To easily integrate the Python service within the MediaVerse DAM, we decided to expose the service via REST API. The service is provided using the Flask library available for Python. A block diagram showing the operation performed by our service is available in Figure 43.

In the following subsections, we describe more practically how to download and use the service developed for this work.

⁴⁷ Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." *Advances in neural information processing systems* 32 (2019): 8026-8037.

⁴⁸ Loper, Edward, and Steven Bird. "Nltk: The natural language toolkit." *arXiv preprint cs/0205028* (2002).

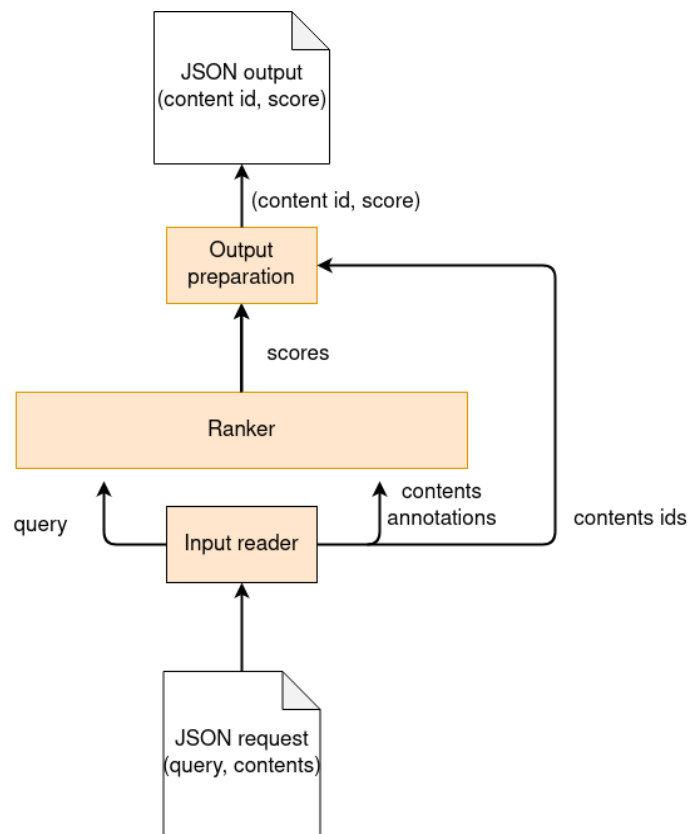


Figure 43: Diagram block showing the inner workings of the developed service

4.4.1 Project folder

The project folder is available at <https://bit.ly/2WKfLCm> and contains the following files

- “api.py” is the Python file containing the REST API interface.
- “best_model.pth” is a pickle file containing the parameters of the neural network
- “call.sh” is a bash file containing the instructions to call the service (with an example of input)
- “model.py” is the Python file containing the implementation of the neural network.
- “requirements.txt” is a text file containing the packages to set up the environment.
- “word2id.json” is a JSON file containing the vocabulary of the model.

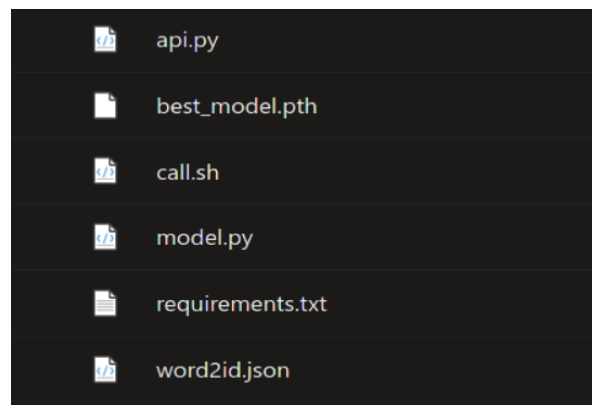


Figure 44: Project folder

4.4.2 Installation and Environment Setup

In this section, we describe how to download the source code and set up the environment for the correct use of the service on a Linux machine.

1. Go to <https://bit.ly/2WKfLCm>, a page like this should appear. Then click “Download”.

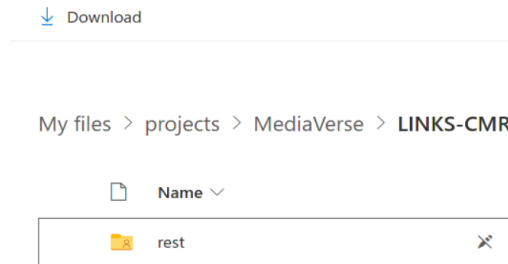


Figure 45: Download page

2. A file named “LINKS-CMR.zip” should have been downloaded. Go into your Download folder and extract the zip. We use 7zip on a Linux machine.

```
7-Zip [64] 17.04 : Copyright (c) 1999-2021 Igor Pavlov : 2017-08-28
p7zip Version 17.04 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,8 CPUs x64)

Scanning the drive for archives:
1 file, 677274985 bytes (646 MiB)

Extracting archive: LINKS-CMR.zip
--
Path = LINKS-CMR.zip
Type = zip
Physical Size = 677274985

Everything is Ok

Files: 6
Size: 677274105
Compressed: 677274985
[matte@MATTE-WORK: Downloads]$
```

Figure 46: Unzipping of the downloaded project archive

Go into the “LINKS-CMR” folder. You should find a subfolder called “rest-vX.Z”, where X.Z is the current version of the tool (i.e. rest-v0.1). Go into the subfolder, create the Python virtual environment and activate it.

```
[matte@MATTE-WORK: rest-v0.1]$ python -m venv venv_cmr
[matte@MATTE-WORK: rest-v0.1]$ source venv_cmr/bin/activate
(venv_cmr) [matte@MATTE-WORK: rest-v0.1]$
```

Figure 47: Installation of the Python environment

3. Set up the environment by installing the packages listed in “requirements.txt”.
4. If everything succeeded, you now have a working Python environment for the usage of the REST API.


```
(venv_cmr) [matte@MATTE-WORK: rest-v0.1]$ pip install -r requirements.txt
Collecting click==8.0.1
  Using cached click-8.0.1-py3-none-any.whl (97 kB)
Collecting Flask==2.0.1
  Downloading Flask-2.0.1-py3-none-any.whl (94 kB)
    | 94 kB 546 kB/s
Collecting itsdangerous==2.0.1
  Downloading itsdangerous-2.0.1-py3-none-any.whl (18 kB)
Collecting Jinja2==3.0.1
  Downloading Jinja2-3.0.1-py3-none-any.whl (133 kB)
    | 133 kB 231 kB/s
Collecting MarkupSafe==2.0.1
  Downloading MarkupSafe-2.0.1-cp39-cp39-manylinux2010_x86_64.whl (30 kB)
Collecting torch==1.9.0
  Downloading torch-1.9.0-cp39-cp39-manylinux1_x86_64.whl (831.4 MB)
    | 146.2 MB 2.7 MB/s eta 0:04:12^C
```

Figure 48: Installation of the required packages

4.4.3 Usage

In this section, we describe how to run and use the service exposed via REST API.

1. Go inside the rest folder and activate the Python environment if not yet done.
2. Run the REST API inside the “api.py” file. You need to insert a free port through the “--port” command line parameter.

```
(venv_rest_cmr) matte@venus:~/WORK/MV/MediaVerse-CMR/rest$ python api.py --port 6000
* Serving Flask app 'api' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
```

Figure 49: How to run the service. Remember to set the --port parameter

3. You can now query the API on port 5000 by sending a JSON input. The input is composed of a “query” key having as value the textual query, and the “items” key with an array of items. Each item is an object having an id (i.e., the id of the content in the MediaVerse network) and the list of annotations for that item. In the image below we can see an example of a request.

```
curl -i -H "Content-Type: application/json" -X POST -d\
  '{"query": "A man on a bicycle",
   "items": [{"id": 721, "annotations": ["person", "car", "bicycle"]},
              {"id": 199, "annotations": ["boat", "dog", "car"]}
  ]'\
  http://localhost:6000/cmr/v0.1/scorer
```

Figure 50: Example of request for the service

4. The response made by the service is a JSON list of objects (one per item in the input request). Each object contains two fields (id and score). The score is a value in [0, 1] for the item having that particular id. Higher the score, higher the similarity with the query. An example of output is shown in the image below.

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 54
Access-Control-Allow-Origin: *
Server: Werkzeug/2.0.1 Python/3.8.3
Date: Thu, 29 Jul 2021 07:08:25 GMT

[{"id": 721, "score": 0.9942}, {"id": 199, "score": 0.0809}]
```

Figure 51: Example of service response

5 Content Adaptation Pipeline

5.1 Introduction

Task 3.3 takes care of developing resources for the automatic adaptation of content (pictures, audio, flat video, and 360 videos). Ideally, the content uploaded by the creators will be of top quality, which means big files that will be hard to play on any kind of device or to be delivered in a fast way. This is where it is needed to consider technologies such as adaptive streaming, progressive downloads, and 360 video optimization.

Figure 52 shows how high-quality input content can be made available to any kind of device. Making several versions of that content and adapting the quality of the content as needed when the content is played.

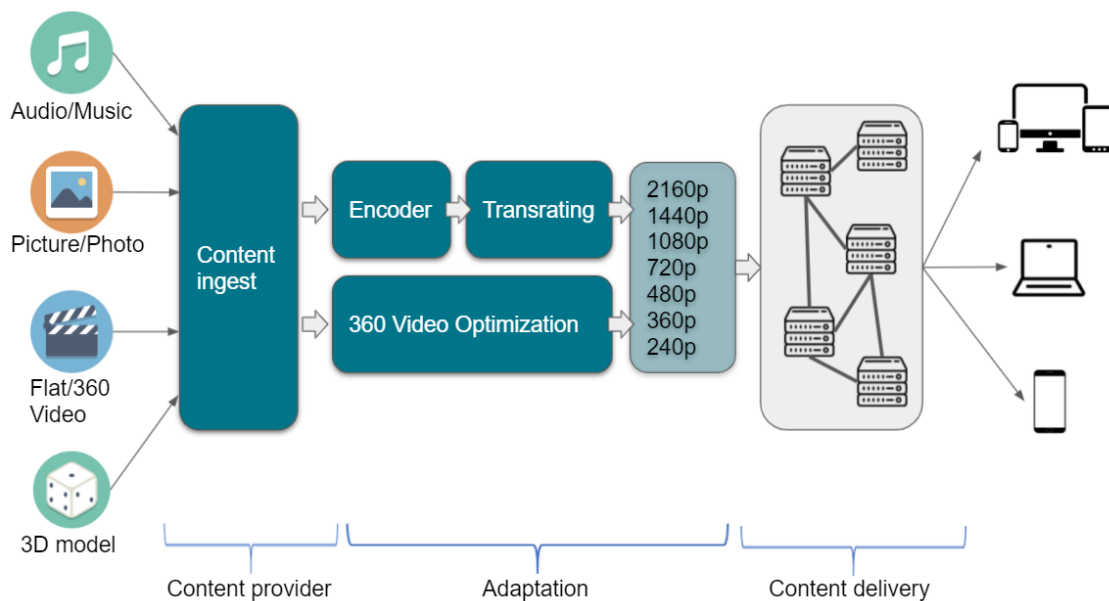


Figure 52: MediaVerse Content adaptation pipeline

5.2 Content Provider

The content ingest enables the insertion and storage of content in MediaVerse. This service and the storage will work as an asset repository for the adaptive streaming component, and also be accessed by other components of the project.

This Digital Asset Management system of MV nodes (DAM) is being created by ATC and offers a REST API that enables the use of functionalities, like insertion, delete, and fetch of contents.

5.3 MediaVerse Content Delivery

The MediaVerse Content Delivery service will deliver to the end-users (professional and general users) media contents mentioned before (Audio, Picture, Video). The service is going to implement an adaptive and immersive end-to-end media service that relies on Content Delivery Network (CDN) technologies and novel adapting techniques to maximise the end-user Quality of Experience (QoE).

The service extends enabling Media technologies to implement, deliver and govern the Video Distribution Service chain leveraging the benefit of the Cloud paradigm and technologies. So, the solution will rely on virtualized

technologies to deploy and configure on-the-fly and govern CDNs' modules (Transcoding, transrating, streaming engine, caches, etc.) as well as CMS (in addition to the streaming synchronization service) considering network load, user preferences, and context, etc.

The use of virtualized instances gives the ability to deploy the streaming service components enables very versatile and flexible deployments of the service chain, including edge placement, approaching in this way contents and final viewers.

5.3.1 Video Encoding and Transcoding

Video encoding is the process of converting raw uncompressed video into a compatible format with most devices. Content distributors can compress this video to save bandwidth while delivering similar quality.

Transcoding is the same conversion but from an already compressed video file to another type or file format.

MediaVerse will optimize any video uploaded by the creators converting it to the chosen codecs and will store several qualities of it.

Introduction to codecs

Codecs are a key element when it comes to the creation and usage of media, one of the main challenges in MediaVerse.

In the context of a video archive with an extension (mp4, for example) this is a kind of container with other types of data inside. Video stream, audio stream, and metadata are inside of a container.

To compress the content data lighter and faster is a priority in the multimedia world. The content creators don't want to wait hours to upload audiovisual content, and users want to consume movies or songs as quickly as possible. Compression helps with faster delivery and also allows the companies to save bandwidth, so it is key to choose the ones that fit the best with any project.

Figure 53 illustrates the power of compression: With the right codec, it is possible to deliver the content within a much shorter time.

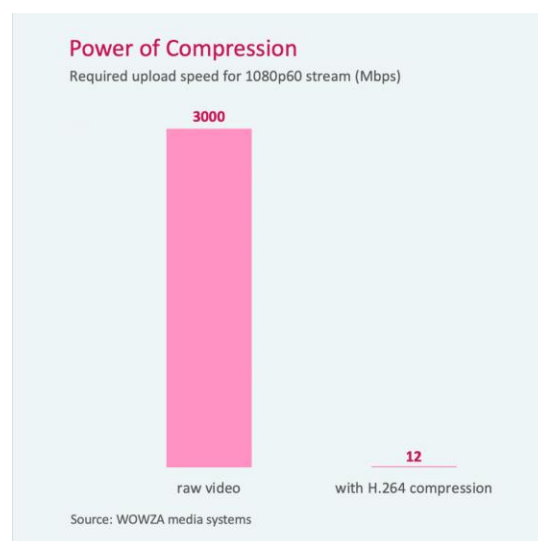


Figure 53: Power of Compression

The MediaVerse codecs election

In the scope of MediaVerse, project partners are building a system to support using media files (images, video, audio, etc.) over the Internet. Adaption is not only based on software or architecture but also based on data. The following requirements have to be met:

- The formats, type containers, and codecs must be analyzed to see which fits better with the MV system.
- Working over the internet the best option needs to be flexible for playing media files. The device that plays media can be portable or not, for example, it is necessary to adapt codecs when a phone is a device but not when it is a laptop because the mobile phone doesn't have the same processing power and the same screen if it is compared against a laptop with a bigger screen and processor.
- The hardware and software need to be compatible; to observe which are the correct codecs and try to include all software and hardware versions as much as possible.
- Codecs with potential and future. Avoiding choosing deprecated or obsolete codecs.
- Prioritizing open-source and royalty-free codecs.

Video Codecs

Here we present the chosen codecs for MediaVerse videos. Any video uploaded into a node will be encoded to one of several of these codecs.

VP9

The first chosen codec is VP9. Youtube created it in 2013, as an improvement of VP8, reducing its bit rate by 50% retaining the same visual quality. VP9 prioritizes sharpest images and some calculations are performed to keep these images in some scenes that are difficult to manage, such as images with a vast quantity of colours and a big number of different shapes.

The encoding algorithm turns the larger pixels into a higher output resolution. Each frame is split into a pixel block, these blocks are analyzed for image redundancies, for instance, the areas that are not changed. The blocks are encoded via motion vectors that predict the qualities of the given block on the next frame. Additional information is encoded using binary compression.

The companies that are using VP9 are Google, Amazon, Bitmovin, Wowza, among others.

VP9 is compatible with Chrome, Edge, Firefox, Explorer, and Opera.

- Open-source and free royalties.
- VP9 has good times for decoding and encoding

H.264

Advanced Video Coding (AVC) also known as H.264 or MPEG-4 Part 10 AVC was created in 2003 by ITU-T, ISO, and IEC.

It is the most used video codec by far, with the disadvantage of being a paid codec for high volume use, so not royalty free. It is compatible with a high range of containers and uses different profiles and adaptive techniques for compression, codification, and decodification. Also, most devices have optimized hardware for decoding it.

H.264 is used by more than 90% of the video industry. Companies that use it are, among others, Netflix, Hulu, Amazon, Vimeo, YouTube, and the iTunes Store.

AV1

There is a third considered option: AV1, which brings a good potential for the future (HDR, HD, 8K, 4K UHD, WCG etc.). It is an open-source and free-royalty codec. The encoding is slower than VP9 but is a codec that has been accepted by all major players of the industry for the first time, and it does not ask for royalties.

Netflix and YouTube have started streaming AV1. YouTube in low-resolution videos and 8K videos. Netflix has started to use it for Android devices.

Video Transcoding, transrating, and content conditioning

This service will process video media files to make them ready for any kind of device and connectivity. Transcoding means encoding to another media format (like moving from AVC video to VP9 video). On the other hand, transrating changes the bitrate of the media file. The objective of these two processes is to reduce the bitrate of the video to have different quality versions of the same video.

The third concept, conditioning, is necessary to provide adaptive streaming on demand. This process implies the alignment of the chunks inside the media files that later will allow the player to switch between qualities as is needed in real-time. Figure 54 shows the pipeline flows.

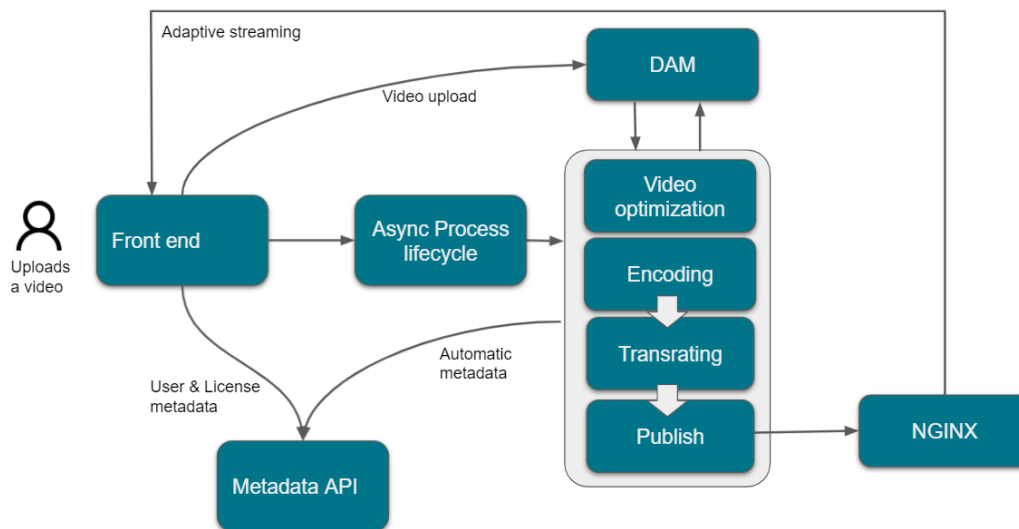


Figure 54: MediaVerse Content adaptation workflow

The content is uploaded by the user directly from the front end and sent to the storage module (DAM).

The transcoding, transrating, and content conditioning services will retrieve the asset, process it, and store the new files in the DAM. The implementation of these services will be based on FFmpeg, which is an open-source project that provides tools to all of these operations.

The metadata database will keep the information associating all these files between them.

After the conditioning process finishes. The user will be able to watch the media file from the front end and it will be served immediately, without needing to download the complete file before.

5.3.2 Audio Transcoding

Audio can be compressed using codecs too. Very popular formats such as mp3 revolutionized the way of using or storing music and nowadays these codecs have been replaced by newer ones that provide the same quality with less file size.

Opus

For audio, the Opus audio codec has been chosen for MediaVerse because it has better performance over the internet (lower latency) than others, is free, open, and fully compatible. Opus has a good future, too, being compatible with VR and spatial sounds (ambisonic or positional 3D). Telegram and WhatsApp are using this audio codec for voice recordings functionality, and most youtube videos are using this codec for the audio. Fader is using the opus codec with Ogg⁴⁹ as a container format delivered via Dash/HLS. Media analysis is done using ffprobe. The transcoding is done using ffmpeg and AAC.

5.3.3 Picture Optimization

There are many techniques to optimize pictures. MediaVerse will focus on three of them:

Converting the input pictures into the right formats

MediaVerse should keep JPEG and PNG formats for compatibility reasons, as they are the most widely used formats. But also, will include WEBP or AVIF as they improve the compression of JPEG by 25%-34% at equivalent perceived quality, and also include transparency.

Compression rate

Choosing a compression rate where there isn't much quality loss but still losing file size

Resizing

Creating several image sizes so the final user can choose the size closest to the one they need.

5.3.4 Streaming Engine

[Nginx](#) is the chosen Streaming engine. Is a free and open-source software that works mainly as a web server but also can be used as a reverse proxy, load balancer, mail proxy, and HTTP cache. It is released under the terms of the [2-clause BSD](#) license, pretty similar to MIT.

Nginx can be configured to run as an RTMP (Real-Time Messaging Protocol) streaming server. According to Netcraft, Nginx served or proxied 22.38% of the busiest sites in August 2021 and is used by online video leaders like Netflix and Hulu.

Adaptive video streaming

Adaptive video streaming takes into account the network capabilities to adapt the transmitted bitrate automatically for the player, optimising the content quality in real-time.

⁴⁹ <https://ftp.osuosl.org/pub/xiph/releases/ogg/>

For video on demand, the media files must be first generated at different qualities and segmented in chunks in the encoding process. For the implementation of this, it will be used Nginx + FFmpeg. For having encoding capabilities and a streaming engine as well.

Among the technologies to achieve adaptive streaming, there is MPEG-DASH and HLS. The first one is a standard created by MPEG, and the second is a proprietary specification specified by Apple. Both of these technologies are similar and define a file format to list the available qualities of a video and allow to choose between the different streams. This will allow the player to adapt to different situations.

The next figure shows a video-on-demand test. The player shows the video content along with data about the performance, like bitrate, qualities, etc.

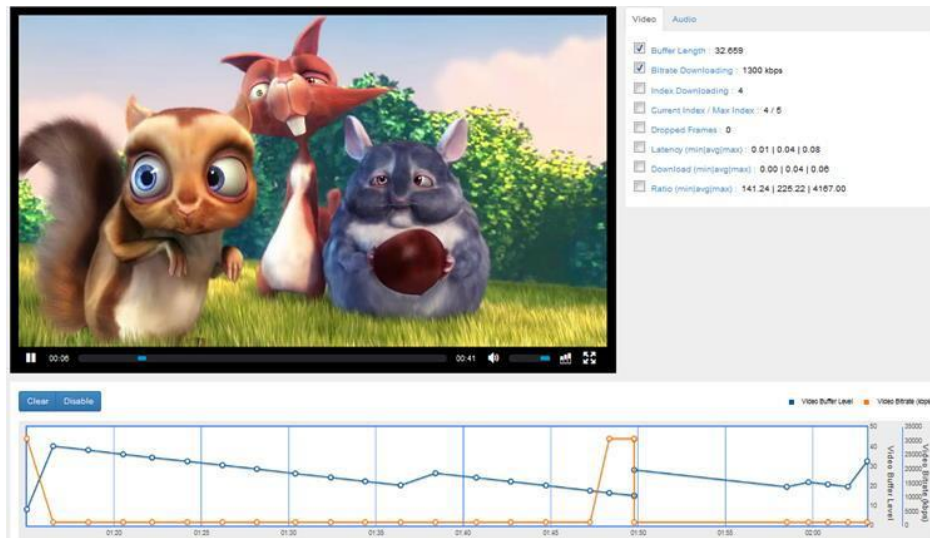


Figure 55: Video-on-Demand test

FoVeated optimisation and segmentation of 360-degree content

One of the major challenges in providing a good quality of service in immersive experiences is the provision of immersive video. Since a 360-degree camera captures the whole sphere, it also needs to be presented to the viewers in full dimensions and resolutions. This requires high throughput networks, as for simple streaming in equirectangular formats, the base resolution needs to be at least 1440p.

An approach to reduce the massive bandwidth requirements is to only stream the current viewport in high quality and provide the other directions at a much lower resolution and switch viewports on demand, e.g., when viewers change their view direction. 360-degree media is tiled and processed in many view dependent variants and the buffer latency is far more reduced compared to 2D video streaming, as swift changes in view direction must be accounted for. Several implementations have been developed over the past years, among which we have chosen some candidates to be provided within MediaVerse.

Selection of a baseline solution

Transform360: During the initial research, a potential candidate was to base the solution on the encoding techniques presented by the Facebook engineering team. They suggest changing the layout to a cubemap format would reduce the file size to be transferred by 25% due to reducing redundancy at the poles. They even suggested a pyramid and dodecahedron geometry, which would reduce the file size by even 80%, but later retracted this as they found offset cubemaps superior.

ISO/IEC 23090-2 Omnidirectional Media Format (OMAF)^{50 51}: This is an MPEG standard for the storage and delivery format of 360-degree content. It features a tile-based High Efficiency Video Coding (HEVC) approach. An equirectangular or cube map projection 360-degree video is further split into viewport-dependent tiles of different resolutions. This ensures that the main view direction gets streamed in a high resolution without wasting too much bandwidth on the non-visible parts. We chose to use the OMAF-standard and reference implementation by Fraunhofer HHI⁵², which is an extension of the Nokia OMAF implementation⁵³ as shown in Figure 56, which is taken from the Fraunhofer MPEG-OMAF technologies site.

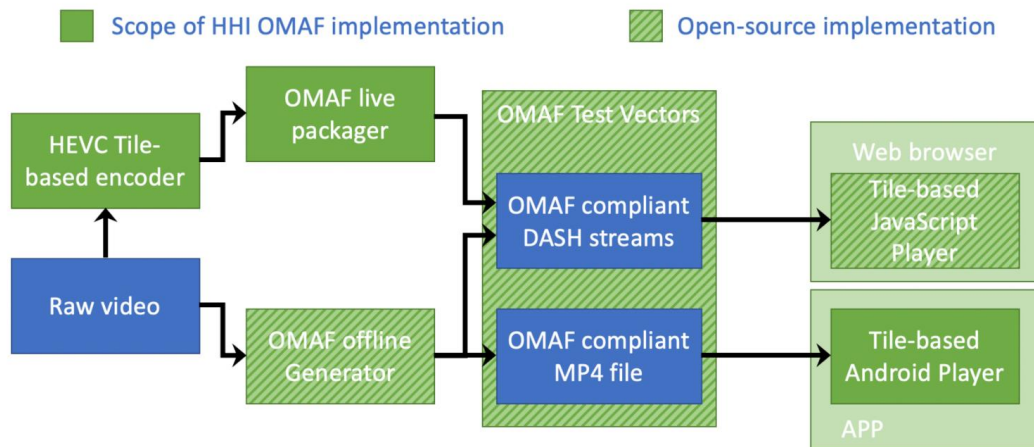


Figure 56: Overview on the scope of Fraunhofer OMAF generator and player implementation⁵⁴

OMAF generator

The OMAF generator within MediaVerse will be based on the open-source OMAF offline generator. It shall be designed as a microservice that allows submitting or linking the source of a 360-degree video file and creating the OMAF compliant file. It will have a job queue to submit a transcoding job, check its status, and receive the finished job result via an API. It takes an equirectangular projection (ERP) 360-degree video file and produces a cubemap projection (CMP) with

- 24 high (768x768 pixel) and 24 low (384x384 pixel) resolution tiles
- 24 viewport streams

The set of output files is packaged in a specific file structure to be consumed by an OMAF compliant player. Currently, there is an open-source player for web browsers, which will be the means of replaying OMAF formats in MediaVerse.

⁵⁰ ISO/IEC 23090-2:2021 Information technology — Coded representation of immersive media — Part 2: Omnidirectional media format <https://www.iso.org/standard/79881.html>

⁵¹ Podborski, D., Jangwoo Son, HTML5 MSE playback of MPEG 360 VR tiled streaming: JavaScript implementation of MPEG-OMAF viewport-dependent video profile with HEVC tiles, MMSys '19: Proceedings of the 10th ACM Multimedia Systems Conference June 2019 Pages 324–327, <https://dl.acm.org/doi/10.1145/3304109.3323835>

⁵² <https://github.com/fraunhoferhhi/omaf.js> (accessed 2021-09-12)

⁵³ <https://github.com/nokiatech/omaf> (accessed 2021-09-12)

⁵⁴ <https://www.hhi.fraunhofer.de/en/departments/vca/technologies-and-solutions/mpeg-omaf.html> (accessed 2021-09-12)

OMAF player in MediaVerse

The DAM within MediaVerse needs preview functionality, also for 360-degree media. A reference implementation of a tile-based OMAF JavaScript player is provided and will be made available in the DAM front-end by ATOS. The functionality is described in Figure 57, which is taken from the Fraunhofer MPEG-OMAF technologies site.

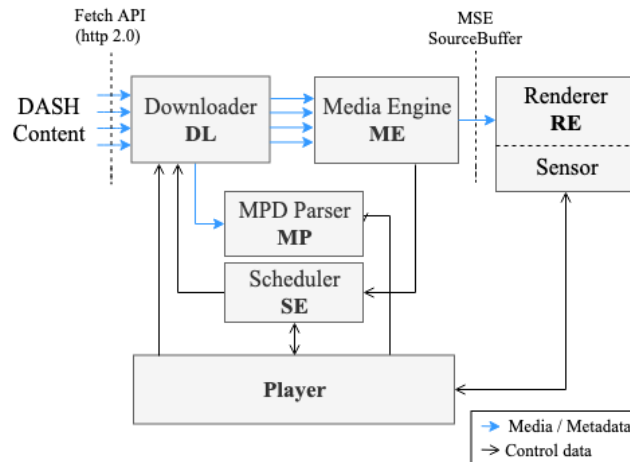


Figure 57: OMAF JavaScript player for web browsers⁵⁵

OMAF player in authoring tools

The authoring tools in MediaVerse, specifically Fader, have to be adapted to support playing OMAF compliant media. The reference OMAF JavaScript player (see Figure 57) for web browsers will be used as a baseline. It is still unclear how advanced features like VR overlays can be used for subtitles, for example. This will have to be investigated during the implementation phase.

Implementation Plan

The purpose of this task within T3.3 is to create an OMAF-compliant file creation service for MediaVerse. It should enable any MediaVerse node to deploy and run this component and make it available as a media service to users.

OMAF4CLOUD⁵⁶ suggests a reference architecture for complex multimedia services that processes 360-degree video files from source to sink via a Media Processing Entity (MPE, the microservice to be implemented) that is configured and controlled by a workflow manager. To address the complexity and fragmentation of approaches for assigning resources to the MPE on multiple cloud environments, the Network-based Media Processing (NBMP) standard was published as ISO/IEC 23090-8:2020. Processing tasks can be templated using a Workflow Description Document (WDD). The MPE will be a containerized (Docker) web application based on the Phoenix framework⁵⁷. It will expose an API to facilitate uploading media, and to retrieving status information and the processing result. A job queue and scheduler will be implemented to manage the use of resources. A current candidate for the job processing queue is Que⁵⁸ to facilitate workflow management. It will be investigated whether the approach suggested by OMAF4CLOUD (Figure 58 as presented by them) can be implemented within MediaVerse. As such, it should be a blue-print for any Media Processing Entity.

⁵⁵ <https://www.hhi.fraunhofer.de/en/departments/vca/technologies-and-solutions/mpeg-omaf.html> (accessed 2021-09-12)

⁵⁶ https://www.researchgate.net/publication/335991664_OMAF4CLOUD_STANDARDS-ENABLED_360_VIDEO_CREATION_AS_A_SERVICE (accessed 2021-09-12)

⁵⁷ Phoenix Framework, <https://phoenixframework.org/> (accessed 2021-09-12)

⁵⁸ Que background job processing library, <https://hexdocs.pm/que/Que.html> (accessed 2021-09-24)

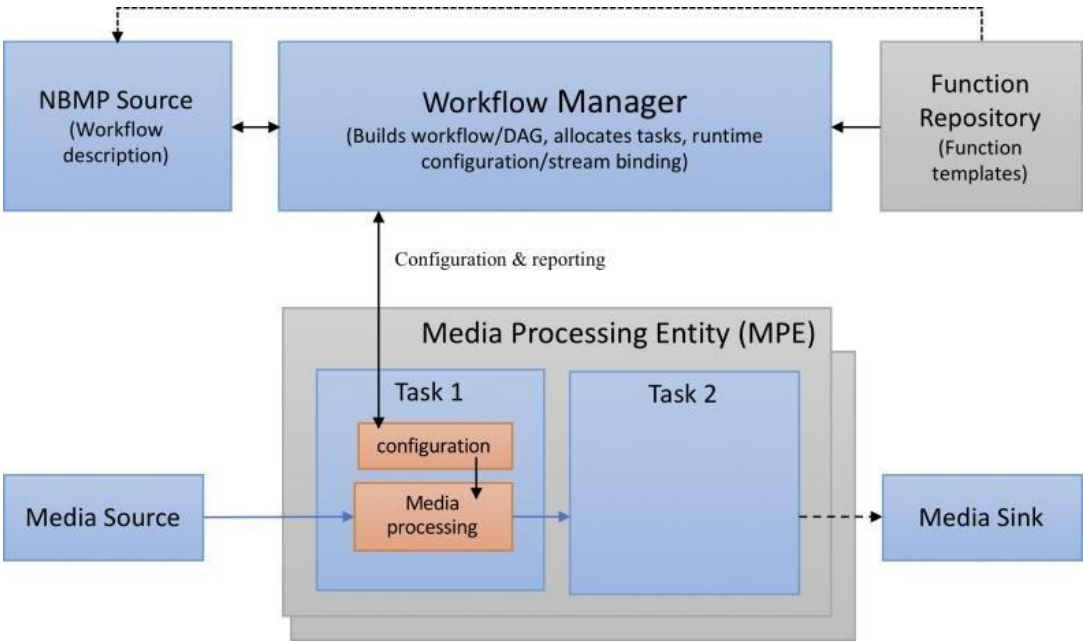


Figure 58: NBMP reference architecture [OMAF4CLOUD]

6 MediaVerse Data Model for content description

6.1 Introduction

In MediaVerse there is a need for a data model able to work with any sort of media file (video, pictures, audio, or any other media file) while storing all the needed aspects of information of the project (user data, licenses, annotation data, technical data, etc.).

6.2 MediaVerse Metadata

The metadata represents "Data talking about data", is structured reference data, necessary for finding those archives, sorting them, or just using them. For Google, YouTube, Netflix and other big companies metadata is crucial to the performance of information systems to classify and categorise data.

When choosing a metadata model, there are a few points to take into account:

- Extensible: The model must allow the users to add their assets.
- Scalable: a scalable solution to the mapping of different metadata into a common format and in a single repository.
- Flexible: the model can be updated if needed.
- A well-defined syntax: An ontology well formed.
- Easily exported to JSON: a simple mapping of metadata from XML to JSON.
- Constantly evolving: EBU community is constantly meeting.
- Open: examples, core and multitude of software extensions are public and are open in repositories.

The metadata of every content present in every node of MediaVerse is stored in a database to make it easier to use and to improve the performance, especially the access speed. Metadata store is twofold, the first one is embedding metadata inside the own archives, this way is slow and difficult to do because the archive headers need to be added or modified in hexadecimal. There is not a unified software to do it in a fast way. The second way is to save metadata in a database and link each archive with its metadata. Choosing the second way is easier to manage and faster to access this metadata.

During the first phase of the project part of the activity carried has been to investigate the current state of the art in terms of metadata and finally, has been selected the format that better supports the MediaVerse approach.

6.2.1 Other Specifications

To make sure we choose a suitable model for the project we have investigated several models including some specific solutions from companies and also more generic standards.

MovieLabs

MovieLabs⁵⁹ was born in 2006 in the United States by the hand of Disney, Paramount, Century Fox, Sony Pictures, Universal, and Warner Bros to research and develop the distribution of movie and series content. One of their projects is the Development of a standard metadata model for this type of content.

⁵⁹ <https://movielabs.com/>

MovieLabs offers some interesting strengths:

- Good ontology
- Pretty well-formed documentation and video tutorials about its models.
- An open API for digital content distribution.
- A project facing the new and upcoming technologies: “2030 Vision” project⁶⁰
- Works with various non-profit organizations to develop public standards such as W3C and SMTPE.

On the cons, it is observed that the model does not contemplate all types of multimedia content such as radio broadcasting, distribution of content in the form of music (music album, discographies), images (museums), or distribution of books.

Dublin Core Metadata Initiative

Another important model to mention is the model Dublin Core⁶¹ (DC), this initiative was born in the USA in 1995 in a meeting of a library consortium based in Dublin (Ohio). The first version was finalized in 1999 and was written in HTML. It was created for resources in the Web world but is currently used by government institutions, research, web pages’ authors, public organizations, etc.

It consists of a fixed base of 15 elements divided into 3 groups (Figure 59):

- **Content metadata:** title, description, subject, source, type, relation and coverage.
- **Intellectual Property metadata:** creator, publisher, rights and contributor.
- **Instantiation metadata:** date, format, identifier and language.

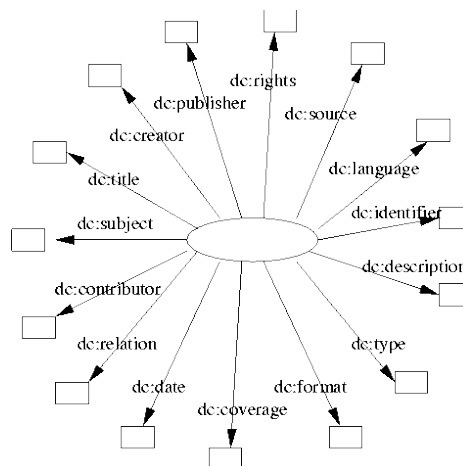


Figure 59: Dublin Core conceptual diagram

Its strengths:

- Easy to extend.
- Easy to understand.
- Open and public.

The advantage of being simple for the MediaVerse project falls short since we need a more complex model that covers more types of elements and relationships between them focused on the audio-visual part.

⁶⁰ https://movielabs.com/prodtech/ML_2030_Vision.pdf

⁶¹ <https://dublincore.org/>

6.2.2 Handmade Specification

There is the option of creating a new metadata model, fully tailored to the needs of MediaVerse. Creating a handmade model has its pros and cons, compared with choosing from the more than 30 standards that exist to manage metadata of multimedia content.

Pros:

- Easily extensible.
- Semantical adaptation to the Project.
- Rapid integration into the Project by the partners.
- Better control over the model.

Cons:

- Development time from the specification to the creation of the practical model in XML-JSON is much longer: create semantics (ontology), implementation in RDF, XML, JSON.
- To create something that is already invented and analysed could be a waste of time.
- Bigger error chances.
- Harder to maintain.
- Harder to adopt.

Netflix

Netflix uses the Media Document model⁶² to represent both dynamic and static metadata for different types of media. It is designed to be adaptable and represent a wide variety of different documents, ranging from the documents resulting from the analysis of the encoded video streaming.

To satisfy all use cases, the Media Document model is built around a few basic principles detailed below:

- Timing Model.
- Spatial Model.
- Nested Structure.

To carry out this cycle in the Netflix environment, they use a “schema-on-write”. For this approach, each Media Document type is associated with a schema. The Netflix Media DataBase also serves as a validation of the media element through the model. The scheme of the cycle is in Figure 60.

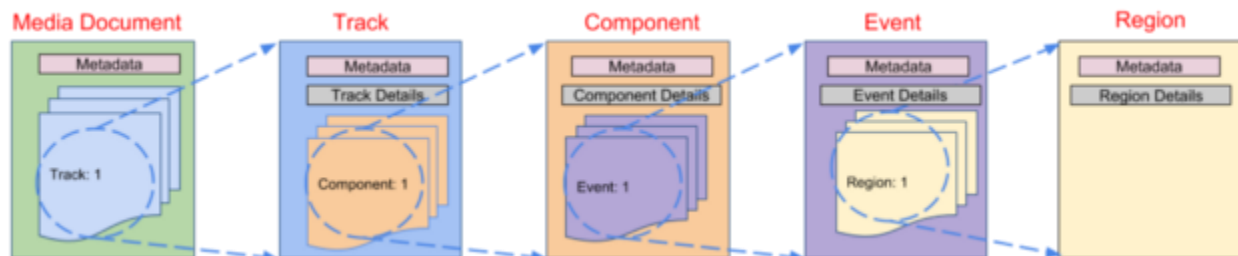


Figure 60: Netflix metadata environment cycle

⁶² <https://netflixtechblog.com/netflix-mediadatabase-media-timeline-data-model-4e657e6ffe93>

Netflix VS Other Standards

The model used by Netflix, as with Movielabs, is an ad-hoc metadata model made exclusively for the Platform. They do not have any type of standardization or seek homogenization with other models.

To adapt this model to the MediaVerse needs would be difficult as, once again, this model focuses too much on series, movies, shorts, etc., while it does not cover music or subtitled audiobooks, between others.

6.2.3 EBUCore

EBUCore⁶³ has been designed to describe the audio, video, and other resources for a wide range of broadcasting applications including archives, exchange, and production in the context of a Service-Oriented Architecture. EBUCore is based on the Dublin Core to maximize interoperability with the community of Dublin Core users such as the European Digital Library 'Europeana'.

```
<ebuCoreMain xmlns:...>
  <coreMetadata>
    <part partId="season1" partName="season a" typeLabel="Season">
      <part partId="e1" partName="episode 1" typeLabel="Episode">
        <title>The New Beginning</title>
        <description>...</description>
      </part>
      <part partId="e2" partName="episode 2" typeLabel="Episode">
        <title>Vendetta</title>
        <description>...</description>
      </part>
      <!-- Etc.-->
    </part>
    <!-- Etc.-->
  </coreMetadata>
</ebuCoreMain>
```

Figure 61: Example of EBUCore

6.2.4 Metadata Section.

Why EBUCore?

- Among the standards, the one supporting the most media file types (images, audios, videos, 3D, 360).
- The biggest core, a big ontology and good semantic.
- Flexible, scalable, customizable.
- Easy conversion to JSON.
- European public vision with present and future.
- A vast quantity of examples (repositories, XML examples) and a friendly community behind EBUCore.
- Most European countries are using EBUCore on their public TVs.

⁶³ <https://tech.ebu.ch/MetadataEbuCore>

After analysing all the pros and the cons, EBUCore has been chosen as the model for the metadata in MediaVerse. EBUCore covers 90% of the user standards. Used by public tv and radio stations like RTVE⁶⁴, RAI, and BBC, used in Eurovision⁶⁵ or public projects like EUScreen⁶⁶ and Europeana⁶⁷.

EBUCore offers a huge cover for any type of multimedia file as metadata. It's based on a very mature metadata model called Dublin Core, but EBUCore expands it and focuses on the multimedia side while trying to harmonize with other metadata models.

After choosing EBUCore as the baseline for the MediaVerse Metadata, it has been needed to customize the specifications to address the special needs of the project, extending some parts to fit better with MediaVerse, such as the technical and descriptive metadata and the metadata for the MediaVerse annotation system. But there are use cases to be implemented yet, such as the metadata for 360 videos and the metadata for the licensing system (WP4).

It is currently under development to perform the functions described in D2.1 for the other use cases.

EBUCore works mainly for its semantics in XML format but it has been achieved through some of the functionalities of the developed API to transform all XML models and semantics into JSON.

The next two tables compare between EBUCore and Movielabs, with aspects that could impact in MediaVerse.

Table 44: EBUCore

| EBUCORE | | |
|---|---|--|
| Positive | Neutral | Negative |
| Extensive (bigger than Movielabs). Flexible, scalable, customizable. Open repository in GitHub. Working with W3C, EUScreen, SMPTE. Well Defined Semantics JSON alternative representation formats for data. EBUCore last change in 2020. 3D image compatibility. Good quantity of examples. | Eurovision Contest, Euroradio, 115 organizations (Europe+) First Publication in 2000. Based on Dublin Core. EBUCore uses EIDR: an universal unique identifier for movie and television assets. | Older than Movielabs. Maybe too complex. Rating Classification Scheme Deprecated. |

⁶⁴ <https://www.rtve.es/>

⁶⁵ https://www.ebu.ch/metadata/schemas/EBUCoreXmlExamples/Eurovision%20Song%20Contest%202015%20Grand%20Final_ebucorexml.xml

⁶⁶ https://pro.europeana.eu/files/Europeana_Professional/Projects/Project_list/EUScreen/Deliverables/D4.5%20E2%80%9393%20The%20Interoperability%20Guidelines.pdf

⁶⁷ <https://www.europeana.eu/en>

Table 45: MovieLabs

| MOVIELABS | | |
|--|--|---|
| Positive | Neutral | Negative |
| <p>Good ontology.</p> <p>Open Repository on GitHub.</p> <p>MovieLabs has an open API for digital distribution.</p> <p>Video tutorials.</p> <p>Working on progress: MovieLabs “2030 Vision” project (Cloud, security & access, software defined workflows).</p> <p>Working with W3C, SMPTE.</p> <p>Recent changes in CM, MMC, MEC (2019-2020).</p> <p>3D image compatibility.</p> <p>Examples of metadata.</p> <p>Good documentation.</p> | <p>Amazon, Sony Pictures, Paramount, Universal Studios, Warner Bros (USA) use MovieLabs.</p> <p>Built in 2006.</p> <p>It uses EIDR: universal unique identifier for movie and television assets.</p> | <p>It’s not based on radio media.</p> <p>A few developed tools.</p> |

Figure 62 depicts an example of an EBUCore asset extended and adapted to MediaVerse to create the annotations metadata, while Figure 63 presents the equivalent JSON representation.

```

<?xml version="1.0" encoding="UTF-8"?>
<ebuCore:ebuCoreMain xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:ebucore="urn:ebu:metadata-schema:ebucore">
  <ebucore:coreMetadata>
    <ebucore:annotationMV annotationId="125">
      <ebucore:name>Man on a bike</ebucore:name>
      <ebucore:name>Beer</ebucore:name>
      <ebucore:score>0.8588 </ebucore:score>
      <ebucore:score>0.7866 </ebucore:score>
      <ebucore:id>354</ebucore:id>
      <ebucore:id>5678</ebucore:id>
    </ebucore:annotationMV>
  </ebucore:coreMetadata>
</ebuCore:ebuCoreMain>

```

Figure 62: EBUCore asset extended and adapted to MediaVerse

```

{
  "name": {
    "namespaceURI": "urn:ebu:metadata-schema:ebucore",
    "localPart": "ebuCoreMain",
    "prefix": "ebucore",
    "key": "{urn:ebu:metadata-schema:ebucore}ebuCoreMain",
    "string": "{urn:ebu:metadata-schema:ebucore}ebucore:ebuCoreMain"
  },
  "value": {
    "TYPE_NAME": "PO.EbuCoreMainType",
    "coreMetadata": {
      "TYPE_NAME": "PO.CoreMetadataType",
      "titleOrAlternativeTitleOrCreator": [
        {
          "name": {
            "namespaceURI": "urn:ebu:metadata-schema:ebucore",
            "localPart": "annotationMV",
            "prefix": "ebucore",
            "key": "{urn:ebu:metadata-schema:ebucore}annotationMV",
            "string": "{urn:ebu:metadata-schema:ebucore}ebucore:annotationMV"
          },
          "value": {
            "TYPE_NAME": "PO.AnnotationMVType",
            "annotationId": 125,
            "name": [
              {
                "TYPE_NAME": "PO.ElementType",
                "value": "Man on a bike"
              },
              {
                "TYPE_NAME": "PO.ElementType",
                "value": "Beer"
              }
            ],
            "score": [
              0.8588,
              0.7866
            ],
            "id": [
              354,
              5678
            ]
          }
        }
      ]
    }
  }
}

```

Figure 63: EBUCore JSON representation

Figure 64 shows an extract of the basic metadata template that has been created for any type of media file. This template will be adapted to the future needs of the project.

| 1 | Metadata Field | Type | Definition / Semantics |
|----|-----------------------|-------------|--|
| 2 | version | main | Version of metadata |
| 3 | identifier | main | The standard for file identification |
| 4 | persons | description | Persons involved (acting, reporting, speaking, etc.) in the file. |
| 5 | genre | description | Style or category of music, movie, audio. |
| 6 | title | main | The title of an item contained in a dataObject. |
| 7 | fileName | format | Full name of the file with extension |
| 8 | fileSize | format | Enter the numerical measurement indicating the size of the digital asset's file(s), in KB, MB, GB, or TB. (Total size of Item) |
| 9 | dateCreation | format | When this file was created, exam: 2020-06-15 00:00:00 |
| 10 | dateLastModification | format | The last time was modified |
| 11 | duration | format | Duration of the overall content (from the first to the last frame) |
| 12 | license | description | Name of the license for this media (copyright, copyleft...) |
| 13 | classification (PEGI) | description | Provides a data structure for age classification related to a programme. |
| 14 | videoFormat | format | Format information of the video file. |
| 15 | audioFormat | format | Format information about the file containing the video or audio or subtitle data and about. |
| 16 | imageFormat | format | Format information about the file containing the image. |
| 17 | containerFormat | format | Format information about file containing the video or audio or subtitle data and about |
| 18 | organization | description | The company that created this file. |
| 19 | programme | description | If this a part of a bigger part. For example, an episode belongs to a programme. |
| 20 | aspectratio | format | 16:9, 4:3 format |
| 21 | overallbitRate | format | The total bit rate of all media streams in this file. |
| 22 | languages | format | Defines languages present in the file by its name and/or code. |
| 23 | types | format | What kind of media contains this file: video, image, audio. |
| 24 | countryOrigin | description | Code or name of the country where was created. |
| 25 | is data object of | relation | Relation with other/s file. |
| 26 | resolution | format | The ratio of width and height of the displayed image. (Width and height does not require to be in pixels.) |
| 27 | author | description | Persons who created file and is the intellectual author. |
| 28 | subtitles | format | If this media contains subtitles here are the files. |
| 29 | blockchainUserID | relation | Identifier of the MV User on the Blockchain (Reference T4.2) |

Figure 64: EBUCore basic metadata template

6.3 API to Manage the MediaVerse Metadata

Atos has created a service to manage the metadata database and to make its functionality available to the rest of the MediaVerse ecosystem. Its API provides endpoints to create, update and delete the metadata in every media content in the project.

6.3.1 Metadata API

This service will take care of any CRUD operations for any assets uploaded into a MediaVerse node. Other services related to metadata like the annotation service, and many others, will add or update fields of metadata that this service will consolidate in just one place.

The other purpose of the metadata database is to store data related to media assets and make it available to any services that request metadata across the platform when needed.

The current implementation is based on MongoDB using a structure very close to the one used by EBUCore for storing the metadata object. The service is being developed using NodeJS with Express and JWT for managing the authentication.

Metadata API service architecture

Figure 65 shows how the API service is organized internally.

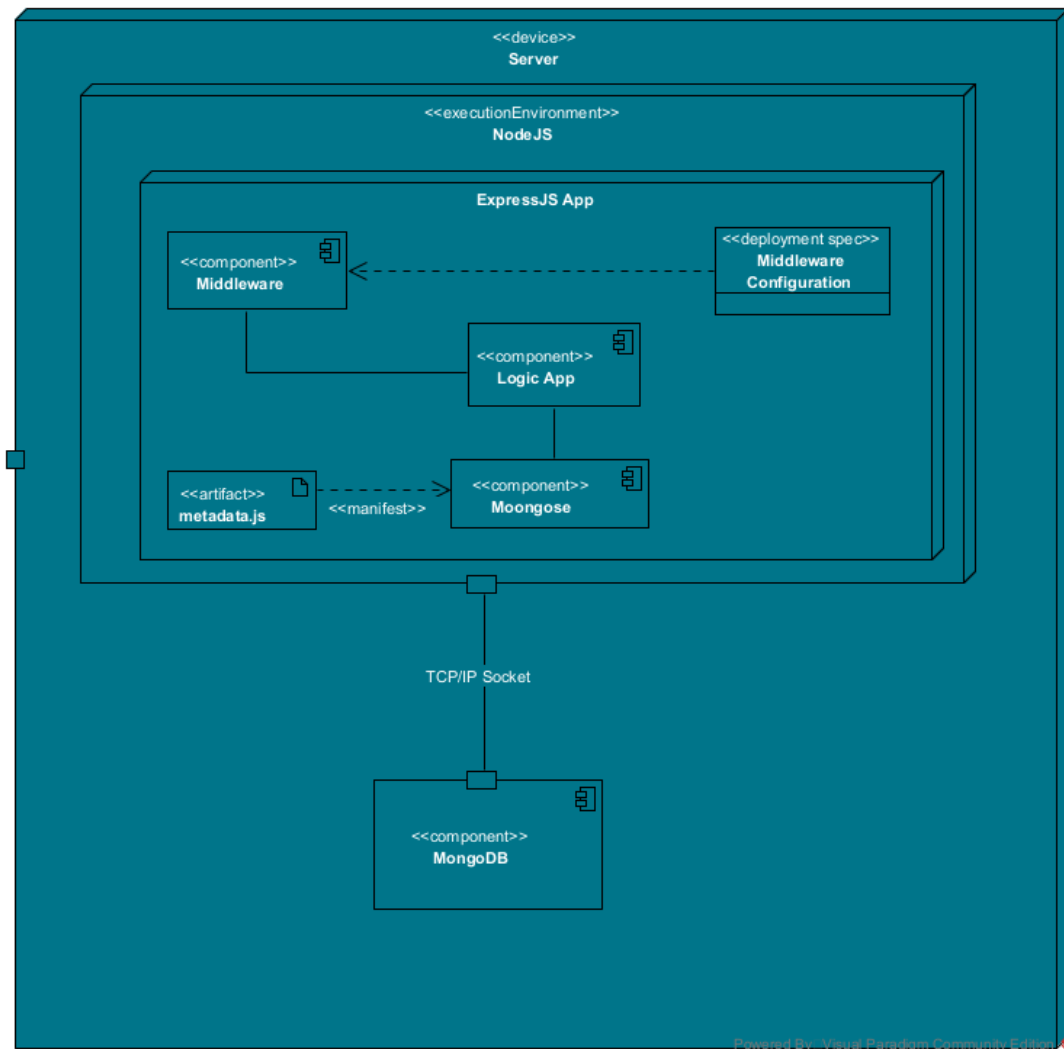


Figure 65: MediaVerse metadata template API service

Endpoints already available

There is an endpoint to upload any media file. A tool closely linked to EBUCore called MediaInfo, then extracts technical and descriptive metadata of the file, and that will be transformed into JSON and stored in the database.

Extending EBUCore will create an ad-hoc domain model that will allow CRUD operations with the different elements (assets created for MediaVerse, annotations, license, 360 video info, etc). Some of the metadata fields will remain empty with the possibility of being added to the metadata in the future like if there are additions to the MediaVerse metadata extended core.

Update and delete endpoints are also available and will allow updating not only the metadata of the stored file but also any previously ad-hoc metadata created.

Archive:

- **getArchive:** This endpoint returns the file with the id specified in the parameter, returning its previously-stored metadata.

```

{
  "success": true,
  "archive": {
    "annotations": [...],
    "_id": "612dd693d1e2c35234dfe096",
    "__v": 0,
    "metadata": {...},
    "name": "file_example_MP3_700KB.mp3"
  }
}

```

Figure 66: getArchive example

- **deleteArchive:** This endpoint deletes the metadata with the id specified in the param of the URL. It returns true if it has been deleted successfully or false otherwise.
- **updateAddArchive:** This endpoint receives a file in the body and generates from it its technical and descriptive metadata, if the metadata already existed then it updates it.

```

{
  "body": {
    "mode": "formdata",
    "formdata": [
      {
        "key": "file",
        "type": "file",
        "src": "/C:/Users/a817692/Downloads/MediaArea_examples/file_example_MP3_700KB.mp3"
      }
    ]
  }
}

```

Figure 67: updateAddArchive example

Returns a Boolean if the operation was successful.

Annotation:

- **updateAddAnnotation:** This endpoint gets a field in the body called "metadata" that will be the XML or JSON including annotation data, and the id of the content that will be updated.

Body request example:

```

{
  "body": {
    "mode": "urlencoded",
    "urlencoded": [
      {
        "key": "metadata",
        "value": "<?xml version='1.0' encoding='UTF-8'"
        "type": "text"
      },
      {
        "key": "archive",
        "value": "file_example_MP3_700KB.mp3",
        "type": "text"
      }
    ]
  }
}

```

Figure 68: updateAddAnnotation example

- **deleteAnnotation:** It deletes the annotation section of the metadata specified by id.

```

{
  "body": {
    "mode": "urlencoded",
    "urlencoded": [
      {
        "key": "archive",
        "value": "file_example_MP3_700KB.mp3",
        "type": "text"
      },
      {
        "key": "annotationId",
        "value": "125",
        "type": "text"
      }
    ]
  }
}

```

Figure 69: deleteAnnotation example

References

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.
- Armeni, I., Sax, S., Zamir, A. R., & Savarese, S. (2017). Joint 2d-3d-semantic data for indoor scene understanding. arXiv preprint arXiv:1702.01105.
- Bach, Nguyen, and Sameer Badaskar. "A review of relation extraction." Literature review for Language and Statistics II 2 (2007): 1-15.
- Bahdanau, D., Cho, K. & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. ICLR.
- Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., & Gall, J. (2019). Semantickitti: A dataset for semantic scene understanding of lidar sequences. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 9297-9307).
- Bello, S. A., Yu, S., Wang, C., Adam, J. M., & Li, J. (2020). Deep learning on 3D point clouds. Remote Sensing, 12(11), 1729.
- Best-Rowden, L., Han, H., Otto, C., Klare, B. F., & Jain, A. K. (2014). Unconstrained Face Recognition: Identifying a Person of Interest from a Media Collection. IEEE Transactions on Information Forensics and Security, 9(12), 2144–2157. <https://doi.org/10.1109/tifs.2014.2359577>
- Bernardi, R., Cakici, R., Elliott, D., Erdem, A., Erdem, E., Ikizler-Cinbis, N., Keller, F., Muscat, A. & Plank, B. (2017). Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures, IJCAI.
- Bertasius, G., Torresani, L., Shi, J. (2018). Object Detection in Video with Spatiotemporal Sampling Networks. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany.
- Bhagavatula, C., Zhu, C., Luu, K., & Savvides, M. (2017). Faster Than Real-time Facial Alignment: A 3D Spatial Transformer. . . ArXiv.Org. <https://arxiv.org/abs/1707.05653>
- Bhardwaj, S., Srinivasan, M., & Khapra, M. M. (2019). Efficient Video Classification Using Fewer Frames. ArXiv.Org. <https://arxiv.org/abs/1902.10640>
- Boscaini, D., Masci, J., Rodolà, E., & Bronstein, M. (2016). Learning shape correspondence with anisotropic convolutional neural networks. Advances in neural information processing systems, 29
- Boscaini, D., Masci, J., Rodolà, E., Bronstein, M. M., & Cremers, D. (2016, May). Anisotropic diffusion descriptors. In Computer Graphics Forum (Vol. 35, No. 2, pp. 431-441).
- Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik. "A training algorithm for optimal margin classifiers." *Proceedings of the fifth annual workshop on Computational learning theory*. 1992.
- Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018). VGGFace2: A Dataset for Recognising Faces across Pose and Age. 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018). Published. <https://doi.org/10.1109/fg.2018.00020>
- Carreira, J., & Zimmermann, A. (2017). Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. ArXiv.Org. <https://arxiv.org/abs/1705.07750>

- Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., ... & Zhang, Y. (2017). Matterport3d: Learning from rgb-d data in indoor environments. arXiv preprint arXiv:1709.06158.
- Chen, Y., Cao, Y., Hu, H., Wang, L. (2020). Memory Enhanced Global-Local Aggregation for Video Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.
- Chen, D., & Dolan, W. (2011). Collecting Highly Parallel Data for Paraphrase Evaluation. ACL Anthology. <https://aclanthology.org/P11-1020/>
- Chen, D., Hua, G., Wen, F., & Sun, J. (2016). Supervised Transformer Network for Efficient Face Detection. ArXiv.Org. <https://arxiv.org/abs/1607.05477>
- Chen, H., Li, J., Hu, X. (2020). Delving deeper into the decoder for video captioning. arXiv preprint arXiv:200105614
- Chen, Y.C., Li, L., Yu, L., El Kholy, A., Ahmed, F., Gan, Z., Cheng, Y. & Liu, J. (2020b). UNITER: UNiversal Image-Text Representation Learning. In: Vedaldi A., Bischof H., Brox T., Frahm J.M. (eds) Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science, vol 12375. Springer
- Chen, J., Zhang, L., Bai, C. & Kpalma, K. (2020a). Review of Recent Deep Learning Based Methods for Image-Text Retrieval. IEEE 3rd International Conference on Multimedia Information Processing and Retrieval.
- Chiang, H. Y., Lin, Y. L., Liu, Y. C., & Hsu, W. H. (2019, September). A unified point-based framework for 3d segmentation. In 2019 International Conference on 3D Vision (3DV) (pp. 155-163). IEEE.
- Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., & Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5828-5839).
- Dai, A., & Nießner, M. (2018). 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 452-468).
- Dai, A., Ritchie, D., Bokeloh, M., Reed, S., Sturm, J., & Nießner, M. (2018). Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4578-4587).
- Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). ArcFace: Additive Angular Margin Loss for Deep Face Recognition. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).
- Devlin, J., Chang, M.W., Lee, K. & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT, (1), 4171-4186.
- Diao, H., Zhang, Y., Ma, L. & Lu, H. (2021). Similarity Reasoning and Filtration for Image-Text Matching. AAAI.
- Engelmann, F., Kontogianni, T., Hermans, A., & Leibe, B. (2017). Exploring spatial context for 3d semantic segmentation of point clouds. In Proceedings of the IEEE international conference on computer vision workshops (pp. 716-724).
- Engelmann, F., Kontogianni, T., Schult, J., & Leibe, B. (2018). Know what your neighbors do: 3D semantic segmentation of point clouds. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops

- Engelmann, F., Kontogianni, T., & Leibe, B. (2020, May). Dilated point convolutions: On the receptive field size of point convolutions on 3d point clouds. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 9463-9469). IEEE.
- Fiorucci, S. (2020). SNK @ DANKMEMES: Leveraging Pretrained Embeddings for Multimodal Meme Detection. EVALITA.
- Feichtenhofer, C., Pinz, A., Zisserman, A. (2017). Detect to Track and Track to Detect. In Proceedings of the 2017 IEEE International Conference on Computer Vision, Venice, Italy, (pp. 3057–3065).
- Gan C., Yang Y., Zhu L., et al. (2016) Recognizing an Action Using Its Name: A Knowledge- Based Approach. International Journal of Computer Vision, 120(1):61-77.
- Garland, M., & Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques (pp. 209-216).
- Girshick, R. (2015). Fast R-CNN. ICCV.
- Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR.
- Graham, B., Engelcke, M., & Van Der Maaten, L. (2018). 3d semantic segmentation with submanifold sparse convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 9224-9232).
- Google. (2001). Protocol Buffers | . Google Developers. <https://developers.google.com/protocol-buffers>
- Google. (2021). Introduction to gRPC. GRPC. <https://www.grpc.io/docs/what-is-grpc/introduction/>
- Guo, C., Fan, B., Gu, J., Zhang, Q., Xiang, S., Prinet, V., Pan, C. (2019). Progressive Sparse Local Attention for Video object detection. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Korea.
- Guo, L., Liu, J., Zhu, X., Yao, P., Lu, S. & Lu, H. (2020a). Normalized and Geometry-Aware Self-Attention Network for Image Captioning, CVPR.
- Guo, Y., Yuan, H. & Zhang, K. (2020b). Associating Images with Sentences Using Recurrent Canonical Correlation Analysis. Applied Sciences, 10, 5516; doi:10.3390/app10165516.
- Hackel, T., Savinov, N., Ladicky, L., Wegner, J. D., Schindler, K., & Pollefeys, M. (2017). Semantic3d. net: A new large-scale point cloud classification benchmark. arXiv preprint arXiv:1704.03847.
- Han, W., Wen, C., Wang, C., Li, X., & Li, Q. (2020, April). Point2Node: Correlation learning of dynamic-node for point cloud feature modeling. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 07, pp. 10925-10932).
- Hanocka, R., Hertz, A., Fish, N., Giryas, R., Fleishman, S., & Cohen-Or, D. (2019). Meshcnn: a network with an edge. ACM Transactions on Graphics (TOG), 38(4), 1-12.
- Hänsch, R., Weber, T., & Hellwich, O. (2014). Comparison of 3D interest point detectors and descriptors for point cloud fusion. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2(3), 57.
- He, K., Zhang, X., Ren, S. & Sun, J. (2015). Deep Residual Learning for Image Recognition. CVPR, 2016.

- He, S., Yang, H., Zheng, X., Wang, B., Zhou, Y., Xiong, Y. & Zeng, D. (2019). Massive Meme Identification and Popularity Analysis in Geopolitics. IEEE International Conference on Intelligence and Security Informatics (ISI).
- He, Y., Yu, H., Liu, X., Yang, Z., Sun, W., Wang, Y., ... & Mian, A. (2021). Deep Learning based 3D Segmentation: A Survey. arXiv preprint arXiv:2103.05423.
- Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.
- Hodosh, M., Young, P. & Hockenmaier, J. (2013). Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. *Journal of Artificial Intelligence Research*, 47, 853-899.
- Hou, J., Wu, X., Zhao, W., Luo, J., Jia, Y. (2019). Joint syntax representation learning and visual cue translation for video captioning. In: *Proceedings of the IEEE International Conference on Computer Vision* (pp. 8918–8927).
- Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., ... & Markham, A. (2020). Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 11108-11117).
- Hua, B. S., Pham, Q. H., Nguyen, D. T., Tran, M. K., Yu, L. F., & Yeung, S. K. (2016, October). Scenenn: A scene meshes dataset with annotations. In *2016 Fourth International Conference on 3D Vision (3DV)* (pp. 92-101). IEEE.
- Huang, J., Zhang, H., Yi, L., Funkhouser, T., Nießner, M., & Guibas, L. J. (2019). Texturenet: Consistent local parametrizations for learning from high-resolution signals on meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4440-4449).
- Huo, J., & Van Zyl, T.L. (2020, June 10). Unique Faces Recognition in Videos. ArXiv.Org. <https://arxiv.org/abs/2006.05713>
- Islam, S., Dash, A., Seum, A. et al. Exploring Video Captioning Techniques: A Comprehensive Survey on Deep Learning Methods. *SN COMPUT. SCI.* 2, 120 (2021). <https://doi.org/10.1007/s42979-021-00487-x>
- Jaritz, M., Gu, J., & Su, H. (2019). Multi-view pointnet for 3d scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops* (pp. 0-0).
- Ji, Shaoxiong, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. "A survey on knowledge graphs: Representation, acquisition and applications." arXiv preprint arXiv:2002.00388 (2020).
- Jia, C., Yang, Y., Xia, Y., Chen, Y.T., Parekh, Z., Pham, H., Le, Q.V., Sung, Y., Li, Z. & Duerig, T. (2021). Scaling Up Visual and Vision-Language Representation Learning with Noisy Text Supervision. *Proceedings of the 38 th International Conference on Machine Learning*, PMLR 139.
- Jiang, M., Wu, Y., Zhao, T., Zhao, Z., & Lu, C. (2018). Pointsift: A sift-like network module for 3d point cloud semantic segmentation. arXiv preprint arXiv:1807.00652.
- Jing Huang and Suyu You, "Point cloud labeling using 3D Convolutional Neural Network," 2016 23rd International Conference on Pattern Recognition (ICPR), 2016, pp. 2670-2675, doi: 10.1109/ICPR.2016.7900038.
- Joonseok, L., Natsev, A., Walter, R., Rahul, S., & George, T. (2018). The 2nd youtube-8m large-scale video understanding challenge. In: *Proc. of the 2nd Workshop on YouTube-8M Large-Scale Video Understanding (ECCV2018)*.
- Kang, K., Li, H., Yan, J., Zeng, X., Yang, B., Xiao, T., Zhang, C., Wang, Z., Wang, R., Wang, X., et al. (2018). T-CNN:

Tubelets With Convolutional Neural Networks for Object Detection from Videos. *IEEE Trans. Circuits Syst. Video Technol* (pp. 2896–2907).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kim, H.U., Kim, C.S. (2016). CDT: Cooperative Detection and Tracking for Tracing Multiple Objects in Video Sequences. In *Computer Vision—Eccv 2016; Part VI*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland (pp. 851–867).

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R.S., Torralba, A., Urtasun, R. & Fidler, S. (2015). Skip-Thought Vectors. *NIPS*.

Landrieu, L., & Simonovsky, M. (2018). Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4558-4567).

Landrieu, L., & Boussaha, M. (2019). Point cloud oversegmentation with graph-structured deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7440-7449).

Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J., and Ng, A. (2012). Building high-level features using large scale unsupervised learning, *International Conference in Machine Learning*.

Lee, K.H., Chen, X., Hua, G., Hu, H. & He, X. (2018). Stacked Cross Attention for Image-Text Matching. *ECCV*.

Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., Wang, L., Hu, H., Dong, L., Wei, F., Choi, Y. & Gao, J. (2020) Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks. *ECCV*.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B. (2018). Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 820-830.

Lin T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) *Computer Vision – ECCV 2014*. *ECCV 2014. Lecture Notes in Computer Science*, vol 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1_48

Lin, J., Xiao, L., & Wu, T. (2020). Multi-target video-based face recognition and gesture recognition based on enhanced detection and multi-trajectory incremental learning. *Technology and Health Care*, 28, 25–35. <https://doi.org/10.3233/thc-209004>

Liu, F., Li, S., Zhang, L., Zhou, C., Ye, R., Wang, Y., & Lu, J. (2017). 3DCNN-DQN-RNN: A deep reinforcement learning framework for semantic parsing of large-scale 3D point clouds. In *Proceedings of the IEEE international conference on computer vision* (pp. 5678-5687).

Liu, J., Ni, B., Li, C., Yang, J., & Tian, Q. (2019). Dynamic points agglomeration for hierarchical point sets learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 7546-7555).

Liu, M., Zhu, M., White, M., Li, Y., Kalenichenko, D. (2019). Looking Fast and Slow: Memory-Guided Mobile Video Object Detection. *arXiv*, arXiv:1903.10172.

Liu, Z., Tang, H., Lin, Y., & Han, S. (2019). Point-voxel cnn for efficient 3d deep learning. *arXiv preprint arXiv:1907.03739*.

Liu, M., Zhu, M. (2018). Mobile Video Object Detection with Temporally-Aware Feature Maps. In *Proceedings of the 2018 IEEE/Cvf Conference on Computer Vision and Pattern Recognition* (pp. 5686–5695).

- Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., & Song, L. (2017). SphereFace: Deep Hypersphere Embedding for Face Recognition. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
- Loper, Edward, and Steven Bird. "Nltk: The natural language toolkit." *arXiv preprint cs/0205028* (2002).
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 91-110.
- Lu, J., Batra, D., Parikh, D. & Lee, S. (2019). ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. NIPS.
- Ma, Y., Guo, Y., Liu, H., Lei, Y., & Wen, G. (2020). Global context reasoning for semantic segmentation of 3D point clouds. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 2931-2940).
- Meng, H. Y., Gao, L., Lai, Y. K., & Manocha, D. (2019). Vv-net: Voxel vae net with group convolutions for point cloud segmentation. In *Proceedings of the IEEE/CVF Int. Conference on Computer Vision* (pp. 8500-8508).
- Meyer, G. P., Charland, J., Hegde, D., Laddha, A., & Vallespi-Gonzalez, C. (2019). Sensor fusion for joint 3d object detection and semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (pp. 0-0).
- Nagendra, S., Baskaran, R., & Abirami, S. (2015). Video-Based Face Recognition and Face-Tracking using Sparse Representation Based Categorization. *Procedia Computer Science*, 54, 746–755. <https://doi.org/10.1016/j.procs.2015.06.088>
- Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." *Icml*. 2010.
- Neunerdt, Melanie, Bianka Trevisan, Michael Reyer, and Rudolf Mathar. "Part-of-speech tagging for social media texts." In *Language Processing and Knowledge in the Web*, pp. 139-150. Springer, Berlin, Heidelberg, 2013.
- Ng, J. Y., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., & Toderici, G. (2015). Beyond Short Snippets: Deep Networks for Video Classification. ArXiv.Org. <https://arxiv.org/abs/1503.08909>
- Nothman, Joel, et al. "Learning multilingual named entity recognition from Wikipedia." *Artificial Intelligence* 194 (2013): 151-175.
- Papineni, K., Roukos, S., Ward, T. & Zhu, W.J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, pp. 311-318.
- Pasunuru, R., Bansal, M. (2017) Multi-task video captioning with video and entailment generation. arXiv preprint arXiv:170407489
- Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." *Advances in neural information processing systems* 32 (2019): 8026-8037.
- Peng, Y., Zhao, Y., & Zhang, J. (2017). Two-stream Collaborative Learning with Spatial-Temporal Attention. ArXiv preprint <https://arxiv.org/abs/1711.03273>
- Pennington, J., Socher, R. & Manning, C.D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

- Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 652-660).
- Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413.
- Qiu, S., Anwar, S., & Barnes, N. (2021). Semantic Segmentation for Real Point Cloud Scenes via Bilateral Augmentation and Adaptive Fusion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 1757-1767).
- Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Aspell, A., Mishkin, P., Clark, J., Krueger, G. & Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020.
- Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640.
- Redmon, J. & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. arXiv:1612.08242.
- Redondo, R., & Gibert, J. (2020, June 24). Extended Labeled Faces in-the-Wild (ELFW): Augmenting Classes for Face Segmentation. arXiv:2006.13980
- Ren, S., He, K., Girshick, R. & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497.
- Rethage, D., Wald, J., Sturm, J., Navab, N., & Tombari, F. (2018). Fully-convolutional point networks for large-scale point clouds. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 596-611).
- Riegler, G., Osman Ulusoy, A., & Geiger, A. (2017). Octnet: Learning deep 3d representations at high resolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3577-3586).
- Rossignac, J., & Borrel, P. (1993). Multi-resolution 3D approximations for rendering complex scenes. In Modeling in computer graphics (pp. 455-465). Springer, Berlin, Heidelberg.
- Roynard, X., Deschaut, J. E., & Goulette, F. (2018). Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. The International Journal of Robotics Research, 37(6), 545-557.
- Rosu, R. A., Schütt, P., Quenzel, J., & Behnke, S. (2019). Latticenet: Fast point cloud segmentation using permutohedral lattices. arXiv preprint arXiv:1912.05905.
- Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. *Learning internal representations by error propagation*. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- Schroff, F., Kalenichenko, D. & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. CVPR.
- Schult, J., Engelmann, F., Kontogianni, T., & Leibe, B. (2020). Dualconvmesh-net: Joint geodesic and euclidean convolutions on 3d meshes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8612-8622).
- Setpal, J. & Sarti, G. (2020). ArchiMeDe @ DANKMEMES: A New Model Architecture for Meme Detection, EVALITA.

- Shen, Wei, Jianyong Wang, Ping Luo, and Min Wang. "Linden: linking named entities with knowledge base via semantic knowledge." In Proceedings of the 21st int. conference on World Wide Web, pp. 449-458. 2012.
- Shen, Z., Li, J., Su, Z., Li, M., Chen, Y., Jiang, YG., Xue, X. (2017) Weakly supervised dense video captioning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1916–1924.
- Shengyu Huang, Mikhail Usvyatsov and Konrad Schindler Indoor Scene Recognition in 3D, 2020
- Simonyan, K., & Zisserman, A. (2014). Two-Stream Convolutional Networks for Action Recognition in Videos. NeurIPS Proceedings.
- Simonyan, K. & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR.
- Song, L., Liu, J., Qian, B. & Chen, Y. (2019). Connecting Language to Images: A Progressive Attention-Guided Network for Simultaneous Image Captioning and Language Grounding. AAAI.
- Su, W., Zhu, X., Cao, Y., Li, B., Lu, L., Wei, F. & Dai, J. (2020). VL-BERT: Pre-training of Generic Visual-Linguistic Representations. ICLR.
- Sun, S., Kuang, Z., Ouyang, W., Sheng, L., & Zhang, W. (2017). Optical Flow Guided Feature: A Fast and Robust Motion. . . ArXiv.Org. <https://arxiv.org/abs/1711.11152>
- Szegedy, C., Ioffe, S., Vanhoucke, V. & Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. arXiv:1602.07261.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015). Going Deeper with Convolutions. CVPR.
- Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification, Conference on Computer Vision and Pattern Recognition. CVPR.
- Tan, M. & Le, Q.V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Proceedings of the 36th International Conference on Machine Learning, PMLR 97.
- Tan, M., Pang, R. & Le, Q.V. (2020). EfficientDet: Scalable and Efficient Object Detection. CVPR.
- Tatarchenko, M., Park, J., Koltun, V., & Zhou, Q. Y. (2018). Tangent convolutions for dense prediction in 3d. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3887-3896).
- Thomas, H., Qi, C. R., Deschaud, J. E., Marcotegui, B., Goulette, F., & Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 6411-6420).
- Tian, H., Tao, Y., Pouyanfar, S., Chen, S. C., & Shyu, M. L. (2018). Multimodal deep representation learning for video classification. World Wide Web, 22(3), 1325–1341. <https://doi.org/10.1007/s11280-018-0548-3>
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In Advances in neural information processing systems, pp. 5998-6008. 2017.
- Vinyals, O., Toshev, A., Bengio, A. & Erhan, D. (2015). Show and Tell: A Neural Image Caption Generator. CVPR.

- Vlad, G.A., Zaharia, G.E., Cercel, D.C. & Dascalu, M. (2020). UPB @ DANKMEMES: Italian Memes Analysis - Employing Visual Models and Graph Convolutional Networks for Meme Identification and Hate Speech Detection. EVALITA.
- Wang, C.Y., Bochkovskiy, A. & Liao, H.Y.M. (2020). Scaled-YOLOv4: Scaling Cross Stage Partial Network. arXiv:2011.08036.
- Wang, W., Chen, Z. & Hu, H. (2019). Hierarchical Attention Network for Image Captioning. AAAI.
- Wang, M., & Deng, W. (2021). Deep face recognition: A survey. Neurocomputing, 429, 215–244. <https://doi.org/10.1016/j.neucom.2020.10.081>
- Wang, J., Jiang, W., Ma, L., Liu, W., Xu, Y. (2018). Bidirectional attentive fusion with context gating for dense video captioning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 7190–7198).
- Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., & Gool, L. V. (2017). Temporal Segment Networks for Action Recognition in Videos. ArXiv.Org. <https://arxiv.org/abs/1705.02953>
- Wang, T., Xiong, J., Xu, X., Shi, Y. (2019a) SCNN: A General Distribution Based Statistical Convolutional Neural Network with Application to Video Object Detection. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA (pp. 5321–5328).
- Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog), 38(5), 1-12
- Wei, X., Zhang, T., Li, Y., Zhang, Y. & Wu, F. (2020). Multi-Modality Cross Attention Network for Image and Sentence Matching. CVPR.
- Wu, H., Chen, Y., Wang, N., Zhang, Z. (2019). Sequence Level Semantics Aggregation for Video Object Detection. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV).
- Wu, W., Kan, M., Liu, X., Yang, Y., Shan, S., & Chen, X. (2017). Recursive Spatial Transformer (ReST) for Alignment-Free Face Recognition. 2017 IEEE International Conference on Computer Vision (ICCV). Published. <https://doi.org/10.1109/iccv.2017.407>
- Xiao, F., Lee, J. (2017). Y. Video Object Detection with an Aligned Spatial-Temporal Memory. arXiv 2017, arXiv:1712.06317.
- Xiao, H., Shi, J. (2020). Video captioning with text-based dynamic attention and step-by-step learning. Pattern Recognition Letters.
- Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. & Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:2048-2057.
- Xu, H., Dong, M., & Zhong, Z. (2017). Directionally convolutional networks for 3d shape segmentation. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2698-2707).
- Yan, X., Zheng, C., Li, Z., Wang, S., & Cui, S. (2020). Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5589-5598).

- Yang, W., Liu, B., Li, W., Yu, N. (2019). Tracking Assisted Faster Video Object Detection. In Proceedings of the 2019 IEEE International Conference on Multimedia and Expo, Shanghai, China (pp. 1750–1755).
- Yang, Y., Zhou, J., Ai, J., Bin, Y., Hanjalic, A., Shen, H.T., Ji, Y. (2018) Video captioning by adversarial lstm. IEEE Transactions on Image Processing (pp. 5600–11).
- Young, P., Lai, A., Hodosh, M. & Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. Transactions of the Association for Computational Linguistics.
- Zhang, C., Kim, J. (2019). Modeling Long—And Short-Term Temporal Context for Video Object Detection. In Proceedings of the 2019 IEEE International Conference on Image Processing, Taipei, Taiwan (pp. 71–75).
- Zhang, Q., Lei, Z., Zhang, Z. & Li, S.Z. (2020). Context-Aware Attention Network for Image-Text Retrieval. CVPR.
- Zhang, C., & Peng, Y. (2018a). Better and Faster: Knowledge Transfer from Multiple. Arxiv preprint. <https://arxiv.org/abs/1804.10069>
- Zhang, C., & Peng, Y. (2018b, April 26). Visual Data Synthesis via GAN for Zero-Shot Video Classification. Arxiv preprint. <https://arxiv.org/abs/1804.10073>
- Zhang, J., Peng, Y. (2019). Hierarchical vision-language alignment for video captioning. In: International Conference on Multimedia Modeling, Springer (pp. 42–54).
- Zhang, Z., Shi, Y., Yuan, C., Li, B., Wang, P., Hu, W., Zha, Z.J.. (2020). Object relational graph with teacher-recommended learning for video captioning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, (pp. 13278–13288).
- Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. IEEE Signal Processing Letters, 23(10), 1499–1503.
- Zhao, H., Jiang, L., Fu, C. W., & Jia, J. (2019). Pointweb: Enhancing local neighborhood features for point cloud processing. In Proceedings of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (pp. 5565–5573).
- Zheng, J., Ranjan, R., Chen, C. H., Chen, J. C., Castillo, C. D., & Chellappa, R. (2020). An Automatic System for Unconstrained Video-Based Face Recognition. IEEE Transactions on Biometrics, Behavior, and Identity Science, 2(3), 194–209. <https://doi.org/10.1109/tbiom.2020.2973504>
- Zheng, G., & Xu, Y. (2021). Efficient face detection and tracking in video sequences based on deep learning. Information Sciences, 568, 265–285. <https://doi.org/10.1016/j.ins.2021.03.027>
- Zhou, E., Cao, Z., & Yin, Q. (2015). Naive-Deep Face Recognition: Touching the Limit of LFW Benchmark or Not?. arxiv:1501.04690
- Zhou, Q. Y., Park, J., & Koltun, V. (2018). Open3D: A modern library for 3D data processing. arXiv preprint arXiv:1801.09847
- Zhu, H., Wei, H., Li, B., Yuan, X., & Kehtarnavaz, N. (2020). A Review of Video Object Detection: Datasets, Metrics and Methods. Applied Sciences, 10(21), 7834. <https://doi.org/10.3390/app10217834>
- Zhu, X., Dai, J., Yuan, L., Wei, Y. (2018). Towards High Performance Video Object Detection. In Proceedings of the 2018 IEEE/Cvf Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA (pp. 7210–7218).

Zhu, X., Wang, Y., Dai, J., Yuan, L., Wei, Y. (2017). Flow-Guided Feature Aggregation for Video Object Detection. In Proceedings of the 2017 IEEE International Conference on Computer Vision, Venice, Italy, (pp. 408–417).

Zhu, X., Xiong, Y., Dai, J., Yuan, L., Wei, Y. (2017a) Deep Feature Flow for Video Recognition. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, Venice, Italy, 22–29 October 2017; (pp. 4141–4150).

Zhu, H., Yan, X., Tang, H., Chang, Y., Li, B., Yuan, X. (2020). Moving Object Detection with Deep CNNs. IEEE Access (pp. 29729–29741).

Hart, C. (2021). Opus Codec: The Audio Format Explained. Retrieved 28 September 2021, from <https://www.wowza.com/blog/opus-codec-the-audio-format-explained>

Ruether, T. (2021). Video Codecs and Encoding: Everything You Should Know. Retrieved 28 September 2021, from <https://www.wowza.com/blog/video-codecs-encoding>

VP9 encoding/decoding performance vs. HEVC/H.264 | Ronald S. Bultje. (2021). Retrieved 28 September 2021, from <https://blogs.gnome.org/rbultje/2015/09/28/vp9-encodingdecoding-performance-vs-hevc-h-264/>

Best audio codec, Anom (2021). Retrieved 28 September 2021, from <https://www.dacast.com/blog/best-audio-codec/>

VP9 Video Codec | Encoding.com. (2021). Retrieved 28 September 2021, from <https://www.encoding.com/vp9/>

H.264 Video Compression | Video Streaming Definition. (2021). Retrieved 28 September 2021, from <https://www.haivision.com/resources/streaming-video-definitions/h-264/>

How to choose the right video codec - Comparison of AV1 | HEVC | VP9. (2021). Retrieved 28 September 2021, from <https://www.muvi.com/blogs/best-video-codec-for-streaming.html>

MMC—MovieLabs. (2021). Retrieved 28 September 2021, from <https://movielabs.com/md/mmc/>

What is an RDF Triplestore?. (2021). Retrieved 28 September 2021, from <https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf-triplestore/>

Netflix MediaDatabase — Media Timeline Data Model. (2021). Retrieved 28 September 2021, from <https://netflixtechblog.com/netflix-mediadatabase-media-timeline-data-model-4e657e6ffe93>

About DCMI. (2021). Retrieved 28 September 2021, from <https://dublincore.org/about/>

HBO - Media Manifest Metadata | Ashton Brown | MBA. (2021). Retrieved 28 September 2021, from <https://ashtongbrown.com/hbo-mmc>

EbuCore metadata (2021). Retrieved 28 September 2021, from <https://tech.ebu.ch/docs/tech/tech3293.pdf>

Standards and Interoperability | EIDR. (2021). Retrieved 28 September 2021, from <https://www.eidr.org/standards-and-interoperability/>

MediaInfo. (2021). Retrieved 28 September 2021, from <https://mediaarea.net/es/MediaInfo>

Support – Video Central. (2021). Retrieved 28 September 2021, from <https://videocentral.amazon.com/home/help?topicId=GYBC754G7J885746>

Annex I: Implementation Details of API

Face recognition

The face recognition service takes a single image or batch as input. After loading, it uses the MTCNN network to detect the face/s and enlarge the bounding box by 30%, subtracts the mean from every channel and sends the pre-processed image in the inference server to be passed to the trained Face Recognition model. After the prediction, the server sends back the response to the client as a list of probabilities. We retrieve the highest probability from the list and if the probability is higher than 80%, the index is mapped to the corresponding celebrity name and gets returned as the response along with the number of faces identified, how many of them were celebrities and how many of them are unknown, else it just returns the number of faces the MTCNN predicted. We present four examples where images are queried to the service. There are four categories, novel image where the identity is contained in the training set, novel image with multiple faces whose identities are contained in the training set, novel image where the identity is not in the training set, novel image without any face (i.e. a building).

Table 46: Example of the face recognition service with one known face


| EXAMPLE 001 | |
|-----------------|--|
| Scenario | User queries a celebrity image (identity contained in dataset) to the face recognition service. This image contains a portrait of Brad Pitt. |
| Request payload |  |
| Response | <pre>success: 1 faces_identified: 1 celebrities: 1 identities { is_celeb: 1 celeb_name: "Brad_Pitt" confidence: 0.9324162 }</pre> |

Table 47: Example of the face recognition service with multiple known faces


| EXAMPLE 002 | |
|-----------------|--|
| Scenario | <p>User queries an image with multiple celebrities (identities contained in dataset) to the face recognition service. The image contains multiple resolution samples to prove that the system works well with a variety of inputs. The identities are:</p> <ol style="list-style-type: none">1. Hank von Helvete2. Maja Ostaszewska3. Lindsay Whalen4. Alex Atala5. Luca Toni6. Amrinder Gill |
| Request payload |  |
| Response | <pre>success: 1 faces_identified: 6 celebrities: 6 identities { is_celeb: 1 celeb_name: "Hank_von_Helvete" confidence: 0.9999939203262329 } identities { is_celeb: 1 celeb_name: "Luca_Toni" confidence: 0.9994907379150391 } identities { is_celeb: 1 celeb_name: "Maja_Ostaszewska" confidence: 0.9999998807907104 }</pre> |

| | |
|--|--|
| | <pre>identities { is_celeb: 1 celeb_name: "Amrinder_Gill" confidence: 0.9998732805252075 } identities { is_celeb: 1 celeb_name: "Lindsay_Whalen" confidence: 0.9856782555580139 } identities { is_celeb: 1 celeb_name: "Alex_Atala" confidence: 0.9994114637374878 } {'id': '1'}</pre> |
|--|--|

Table 48: Example of the face recognition service with an image with an unknown face

| EXAMPLE 003 | |
|-----------------|---|
| Scenario | User queries a celebrity image (identity not contained in dataset) to the face recognition service. |
| Request payload |  |
| Response | <pre>success: 1 faces_identified: 1 unidentified: 1 identities { } {'id': '1'}</pre> |


Table 49: Example of the face recognition service with an image without faces

| EXAMPLE 004 | |
|-----------------|--|
| Scenario | User queries an image without faces to the face recognition service. |
| Request payload |  |
| Response | success: 0 error: "The image contains no faces" |

Object detection


The object detection service takes a single image or batch as input. After loading, the image dimensions are expanded by one and sent to the trained Object Detection model. The predictions then are parsed and divided in detected_boxes, detection_classes, detection_scores and num_detections. Then each bounding box, with the object name and confidence are returned to the user. If there is a problem, the service returns the appropriate message to inform the user. We present two examples where images are queried to the service. There are two categories, novel image with one object and novel image with multiple objects.

Table 50: First example of the object detection service

| EXAMPLE 005 | |
|-----------------|--|
| Scenario | User queries an image with a single object |
| Request payload |  |
| Response | success: 1 |

| | |
|--|---|
| | <pre>detections { object: "car" confidence: 0.9808660745620728 bbox { top_left_x: 0.09656491875648499 top_left_y: 0.053043484687805176 width: 0.8666345477104187 height: 0.9994558095932007 } } detections { object: "car" confidence: 0.9025916457176208 bbox { top_left_x: 0.08079338073730469 top_left_y: 0.014848381280899048 width: 0.9251905679702759 height: 0.5852 456092834473 } } {'id': '1'}</pre> |
|--|---|

Table 51: Second example of the image annotation service

| EXAMPLE 006 | |
|-----------------|--|
| Scenario | User queries an image with multiple objects. |
| Request payload |  |
| Response | <pre>success: 1 detections { object: "chair" confidence: 0.9889440536499023 bbox {</pre> |

```

top_left_x: 0.5380624532699585
top_left_y: 0.18489040434360504
width: 0.9687177538871765
height: 0.44020676612854004
}
}
detections {
  object: "potted plant"
  confidence: 0.9581874012947083
  bbox {
    top_left_x: 0.11374226957559586
    top_left_y: 0.9172728061676025
    width: 0.2543990910053253
    height: 0.9988775253295898
  }
}
detections {
  object: "laptop"
  confidence: 0.9419235587120056
  bbox {
    top_left_x: 0.4484875500202179
    top_left_y: 0.5046278834342957
    width: 0.5826987624168396
    height: 0.6446022391319275
  }
}
detections {
  object: "cup"
  confidence: 0.9123003482818604
  bbox {
    top_left_x: 0.4460694491863251
    top_left_y: 0.3766803741455078
    width: 0.521438717842102
    height: 0.41273224353790283
  }
}
detections {
  object: "potted plant"
  confidence: 0.8860528469085693
  bbox {
    top_left_x: 0.13246750831604004
    width: 0.3462074398994446
    height: 0.08883935958147049
  }
}
detections {
  object: "potted plant"
  confidence: 0.8571756482124329


```


| | |
|--|--|
| | <pre>bbox { top_left_x: 0.001121196779422462 top_left_y: 0.002717442810535431 width: 0.11723197996616364 height: 0.07148905098438263 } } {'id': '1'}</pre> |
|--|--|

Image captioning

The image captioning service takes a single image or batch as input. After loading, the image/s is sent to the image captioning model for prediction. The predictions then are passed from the tokenizer and are returned to the user. If there is a problem, the service returns the appropriate message to inform the user. We present one example where images are queried to the image captioning service.

Table 52: Example of the image annotation service

| EXAMPLE 007 | |
|-----------------|--|
| Scenario | User queries an image to be annotated. |
| Request payload |  |
| Response | <pre>success: 1 caption: "bunch of fruits and vegetables are on table" {'id': '1'}</pre> |

Meme detection

The meme detection service takes a single image or batch as input. After loading, the image is passed to the meme detection model for prediction. If the probability is greater than 50% then the image is classified as a meme else as a regular image. The results then are returned to the user. If there is a problem, the service returns the appropriate message to inform the user. We present two examples where images are queried to the service. We use a meme type image (with text) and one regular and present the responses from the API.

Table 53: First example of meme detection with a meme type image



| EXAMPLE 008 | |
|-----------------|--|
| Scenario | Meme type image (with text) |
| Request payload |  |
| Response | <pre> success: 1 result { is_meme: "yes" confidence: 0.992141842842102 } {'id': '1'}</pre> |

Table 54: Second example of meme detection with a regular image

| EXAMPLE 009 | |
|-----------------|--|
| Scenario | Regular Image |
| Request payload |  |
| Response | <pre> success: 1 result { is_meme: "no" confidence: 0.9998974800109863 } {'id': '1'}</pre> |



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.