MediaVerse

A universe of media assets
and co-creation opportunities

D5.4

Content Moderation Toolset

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	Content Moderation Toolset
Deliverable number	D5.4
Deliverable version	V1.0
Previous version(s)	N/A
Contractual Date of delivery	30.09.2022
Actual Date of delivery	07.10.2022
Nature of deliverable	Other
Dissemination level	Public
Partner Responsible	CERTH
Author(s)	Olga Papadopoulou, Manos Schinas, Dimitris Karageorgiou, Yiannis Sarridis, Symeon Papadopoulos (CERTH), Spyros Papafragkos, Tasos Lampropoulos, Nikolaos Latzonis (ATC)
Reviewer(s)	Nicolas Patz, Kay Macquarrie (DW), Florian Maillard (STXT)
EC Project Officer	Luis Eduardo Martinez Lafuente

Abstract	This deliverable provides documentation on the Content moderation toolset. It presents the individual services that constitute the toolset, several experimental results on the underlying AI models, and the moderation UI and its integration within MediaVerse.

Keywords	Moderation, Misinformation, Disturbing content, Not Safe for Work, DeepFakes, Verification

Copyright

© Copyright 2022 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	16/06/2022	Olga Papadopoulou (CERTH)	First Draft Table of Content
V0.2	24/06/2022	Manos Schinas, Symeon Papadopoulos (CERTH)	Final Table of Content
V0.3	01/09/2022	Yiannis Sarridis (CERTH)	Disturbing and NSFW content detection
V0.4	09/09/2022	Olga Papadopoulou, Dimitris Karageorgiou (CERTH)	Deepfake and image forensics detection
V0.5	10/09/2022	Tasos Lampropoulos, Nikolaos Latzonis (ATC)	Content moderation User Interface and Integration
V0.6	13/09/2022	Spyros Papafragkos (ATC)	Hate Speech and Content moderation User Interface
V0.7	15/09/2022	Manos Schinas, Symeon Papadopoulos (CERTH)	Final Draft Review
V0.8	21/09/2022	Nicolas Patz, Kay Macquarrie (DW), Florian Maillard (STXT)	Review
V0.9	30/09/2022	Olga Papadopoulou, Manos Schinas (CERTH)	Revised version
V1.0	07/10/2022	Evangelia Kartsounidou, Symeon Papadopoulos (CERTH)	Final Document

Glossary

Abbreviation	Meaning
AI	Artificial Intelligence
ANNs	Artificial Neural Networks
BOF	Bag-Of-Features
CNNs	Convolutional Neural Networks
CONAN	COunter NArratives through Nichesourcing
DID	Disturbing Image Dataset
GANs	Generative Adversarial Networks
HKUST-MLMA	Multilingual and Multi-Aspect Hate Speech Analysis
HS	Hate Speech
MLP	Multilayer Perceptron
MV	MediaVerse
NSFW	Not Safe for Work
SGD	Stochastic Gradient Descent
SFW	Safe for Work
UGC	User Generated Content
UI	User Interface
ViTs	Visual Transformers
WP	Work Package

Table of Contents

Index of Figures

Index of Tables

Executive Summary

Content moderation is the task of monitoring user-generated content based on platform-specific rules and guidelines to determine if the content should be published on the online platform or not. Platforms adopt two types of moderation: soft moderation, where warning labels to questionable or harmful content are applied while content remains accessible and hard moderation, where content is removed and accounts are suspended.

This report explains how content moderation is applied in MediaVerse and presents the methods that constitute the content moderation toolset. Content moderation in the MediaVerse node comprises automatic filters, which flag content that potentially violates the local policies of the node. The toolset contains models for identifying the following types of inappropriate content:

- Disturbing content including violent and gruesome scenes
- Nudity, profanity and other types of NSFW (not safe for work) content
- Manipulated (photoshoppped) images and deepfake/synthetic media
- Content associated with hate speech and cyberbullying

For manipulated or synthetic media, users will be able to verify content on demand by applying image forensics methods for image assets and deepfake detection for image and video assets. For the other media types, we developed tools that apply moderation annotations on the entire incoming assets and provide labels such as disturbing, NSFW, hate speech, along with a confidence score.

The MediaVerse administrator is responsible for setting up the moderation rules to filter assets. In the future, a feedback mechanism will be developed that will result in new annotated items being available to the system to retrain underlying models and fine-tune them to the MediaVerse node needs.

The report also contains the initial design of a simple and easy-to-use User Interface that will provide the content moderation functionalities.

1 Introduction

The ever-growing amounts of online User Generated Content (UGC) made clear the need for content moderation technologies to protect the audience of digital platforms from content that can cause feelings of worry, concern, or anxiety. The type of moderation varies from platform to platform, as rules around what UGC is considered appropriate are often set at a platform level and are typically implemented through non-transparent systems through a combination of Artificial Intelligence (AI) systems and crowd workers. The overwhelming amount of information combined with the coronavirus pandemic has forced platforms to rely more on automatic content moderation methods. Specifically, Facebook, Google and Twitter turned to machine-learning tools and artificial intelligence — not humans — to control posts on their platforms. However, a POLITICO[1] post entitled 'What happened when humans stopped managing social media content'[2] states, 'Nobody appreciated the content moderators until they were gone.' Automatic content moderation has become necessary; however, the accuracy and bias of the AI-based models remain significant challenges. Below we list two prominent 'mistakes' affecting Facebook as a result of their automatic content moderation methods.

A black activist and writer says Facebook censored her for calling out racism[3]. Ijeoma Oluo was on a road trip with her children when she decided to stop at a Cracker Barrel. While at the restaurant, which has paid millions to settle lawsuits over racial discrimination against black employees and diners, she joked on Twitter: "At Cracker Barrel 4 the 1st time. Looking at the sea of white folk in cowboy hats & wondering 'will they let my black ass walk out of here?'" (Figure 1 left). After her post, she received racist hate and violent threats from people and as she said she would be nervous to be surrounded by such angry people. Facebook reacted to her post by suspending her account instead of suspending the accounts that abused her and removing the hate posts against her. The second example concerns an Associated Press photographer who took a photo of children, during the Vietnam War, as they fled a napalm attack (Figure 1 right). One of them, a 9-year-old girl, is naked. Facebook deleted the post on the grounds that it contained nudity.

Figure 1: Examples of automatic content moderation mistakes in Facebook

[1] https://www.politico.eu/

[2] https://www.politico.eu/article/facebook-content-moderation-automation/

[3] https://eu.usatoday.com/story/tech/2017/08/03/facebook-ijeoma-oluo-hate-speech/537682001

These are cases of unfair moderation that harmed individuals and that platforms did not offer any way to amend this. This kind of issues is what motivates MediaVerse, because moderation rules can be adapted by local communities/organizations and not centrally set and enforced.

An overarching question about content moderation refers to the kind of content is inappropriate and should be moderated. The moderation rules are platform-specific, and we need to define how MediaVerse nodes will deal with uploaded content. MediaVerse nodes define moderation rules in a transparent way, e.g., users can see the rules defined by the organization in the deployed node. Additionally, users can define their own additional moderation rules without however disregarding the node-level rules. Finally, users or organisations can deploy their own instance and define their own rules if the available nodes do not fit their needs.

Considering the MediaVerse requirements, we have focused on the following types of inappropriate content:

- **Disturbing content including violent and gruesome scenes** by developing a disturbing content detector that filters images and videos depicting humans or animals subjected to violence, harm, and suffering that can cause feelings of worry, concern, or anxiety to the viewer,
- **Nudity, profanity and other types of NSFW (not safe for work) content**,
- **Misleading, inaccurate or simply false media** by integrating image forensics and image/video deepfake detection tools, and
- **Content associated with hate speech and cyberbullying** by developing a hate speech detection service.

State-of-the-art deep learning models are applied on text and visual content, and trained on reference datasets to build the models that support the respective services.

Through this work, we have identified numerous challenges in automatic content moderation and concluded that human contribution is essential for accurate results. To this end, we first focused on developing and designing the content moderation toolset with the best possible performance in terms of the developed methods and in some particularly challenging cases, we have performed iterations incorporating human feedback to improve the models. As a next step in the final year of the project, we aim to investigate and develop a streamlined feedback mechanism to incorporate user annotations on new media assets in order to fine-tune the underlying models to the particular node needs. Additionally, the administrator of the MediaVerse node will be given the option of setting the "strictness" level of the filter (e.g., strict, relaxed, inactive) and use more advanced settings to configure the behaviour of the moderation toolset. The toolset will support this adaptation process without requiring any AI or technical expertise from the MediaVerse administrator.

Figure 2 illustrates a high-level diagram of the content moderation toolset. The toolset includes four components, each of them serving a different purpose. The automatic detection models are used directly by the DAM for every uploaded asset to identify disturbing and NSFW content. In addition, content discovered in social media and imported in MV is also automatically tagged by the hate speech model. The on-demand analysis components can be used for selected assets to flag misleading or false content with the help of user inspection. This pertains to deepfake images and videos and tampered images. A fine-tuning component is also foreseen to provide retraining of the models based on user feedback. Finally, the moderation UI provides a way to define the moderation rules at node or user level. This can be used by the administrators of a MV node to specify a level of moderation for content uploaded to or retrieved by the node, but also by individual users to restrict the content discovered during federated search.

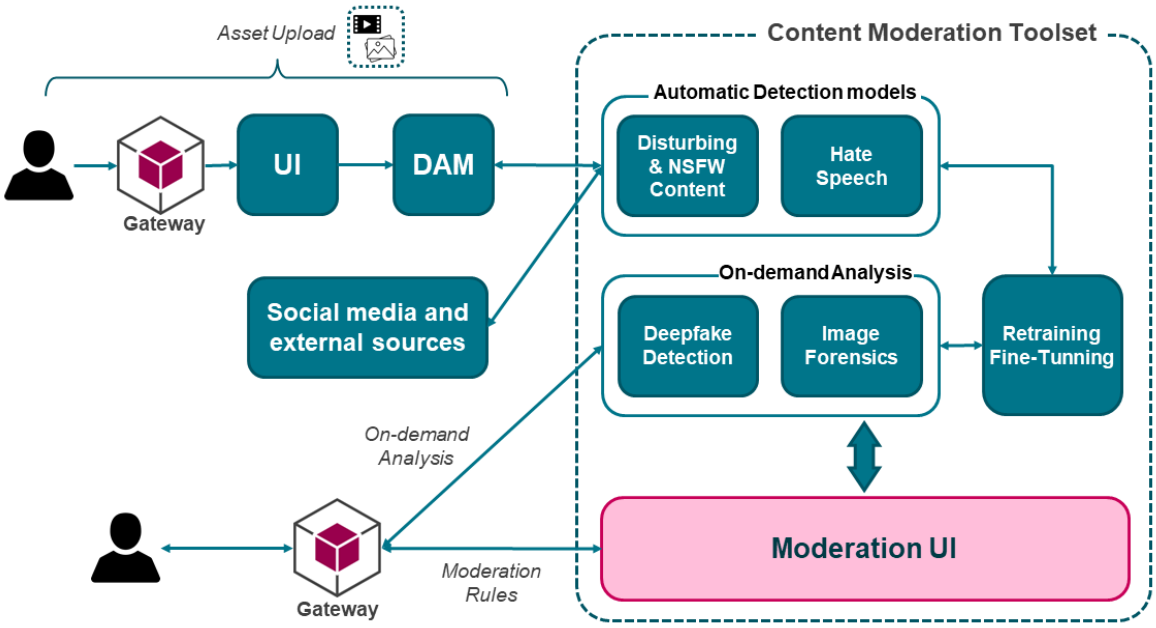


Figure 2: High-level architecture of Content Moderation Toolset

1.1 Purpose of the Deliverable

This document describes the Content Moderation Toolset of WP5 / T5.1. The report is structured as follows:

- Section 2 describes the disturbing content detection approach developed by CERTH.
- Section 3 describes the detection approach for Not Safe For Work (NSFW) content developed by CERTH.
- Section 4 describes the deepfake detection and image forensics approaches that have been integrated by CERTH in the Content Moderation Toolset.
- Section 5 presents the hate speech detector developed by ATC that is able to detect hate speech content on text content from social media sources like Twitter.
- Section 6 presents the design of the Content Moderation User Interface and the integration of the presented components.

1.2 Relation with Other Activities in MediaVerse

The Content Moderation toolset can offer valuable support to the MediaVerse Use Cases:

- In **Use Case 1: Citizen Journalism**, the verification components of the Content Moderation toolset could be used by journalists to verify the content uploaded by citizens, while the disturbing content detection could be used to blur/hide such content and reduce the distress of content moderators.
- In **Use Case 2: Co-creation of New Media Formats**, the toolset aims to protect vulnerable users (e.g., young immigrants, young students) from inappropriate or harmful content, including violent scenes, NSFW and hate speech.
- In **Use Case 3: Hybrid Intelligence Experimental Artwork Series**, the Content Moderation toolset could contribute to the experiments addressing the notion of truth in social media by providing verification labels to the content.

2 Disturbing Content Detection

Disturbing content refers to visual depictions of humans or animals subjected to violence, harm, and suffering that can cause feelings of worry, concern, or anxiety to the viewer. In many professions, such as journalism, employees are often exposed to content generated by users, and inevitably, they are also exposed to disturbing content that could cause emotional trauma. For instance, the recent Russian invasion of Ukraine resulted in the dissemination of a large amount of disturbing content that journalists have to face daily. The main challenge in disturbing content detection is that different contents affect people differently. It depends on the respective "frame of mind" of people exposed to potentially disturbing material. Here, we focus on the development of a disturbing detector that considers what we call "obvious graphic imagery", meaning images depicting visible bodily harm or injury and such. Other forms of disturbing content that are less obvious or subliminal are beyond the scope of this work.

The major challenge of developing automatic approaches to detecting disturbing content is the lack of datasets for such a task. This is expected once we consider how psychologically demanding the annotation procedure is for such content. In our previous work, we had generated such an annotated dataset, which we further describe in Section 2.1. After presenting the reference dataset of disturbing content in Section 2.1, we present the first fine-tuning experiments on a variety of deep learning architectures (Section 2.2) and further iterations on the disturbing content detection model with the aim of incorporating more annotation feedback focusing on a set of "hard" examples (Section 2.3). We also provide some support of the explainability of the developed models (Section 2.4) and describe their integration in the media annotation framework for videos (Section 2.5).

2.1 Dataset

To the best of our knowledge, the Disturbing Image Dataset (DID) (Zampoglou et al., 2017) is the only existing dataset of disturbing visual content in the literature[4]. DID consists of 5401 images, of which 2043 are annotated as disturbing, and the rest 3358 as non-disturbing. The ground truth is noisy as the images were annotated in a semi-automatic way. The categories included in DID include teratogenesis, hanging of persons, emaciated bodies (e.g., death or suffering caused by malnutrition), blood (without involving any human/animal body - only blood) and dead bodies. Figure 3 depicts 16 sample images of the DID dataset. The three disturbing examples have been processed to obfuscate the disturbing parts.

[4] There are likely several proprietary datasets used by digital platforms and by big services providers – however, these are not available to researchers. For instance, Amazon Rekognition, a popular cloud service used to identify objects, people, text, scenes, and activities, can also detect inappropriate or offensive content in images using a two-level taxonomy of inappropriate or offensive content. For visually disturbing content, they define five categories: emaciated bodies, corpses, hanging, air crashes, explosions and blasts.

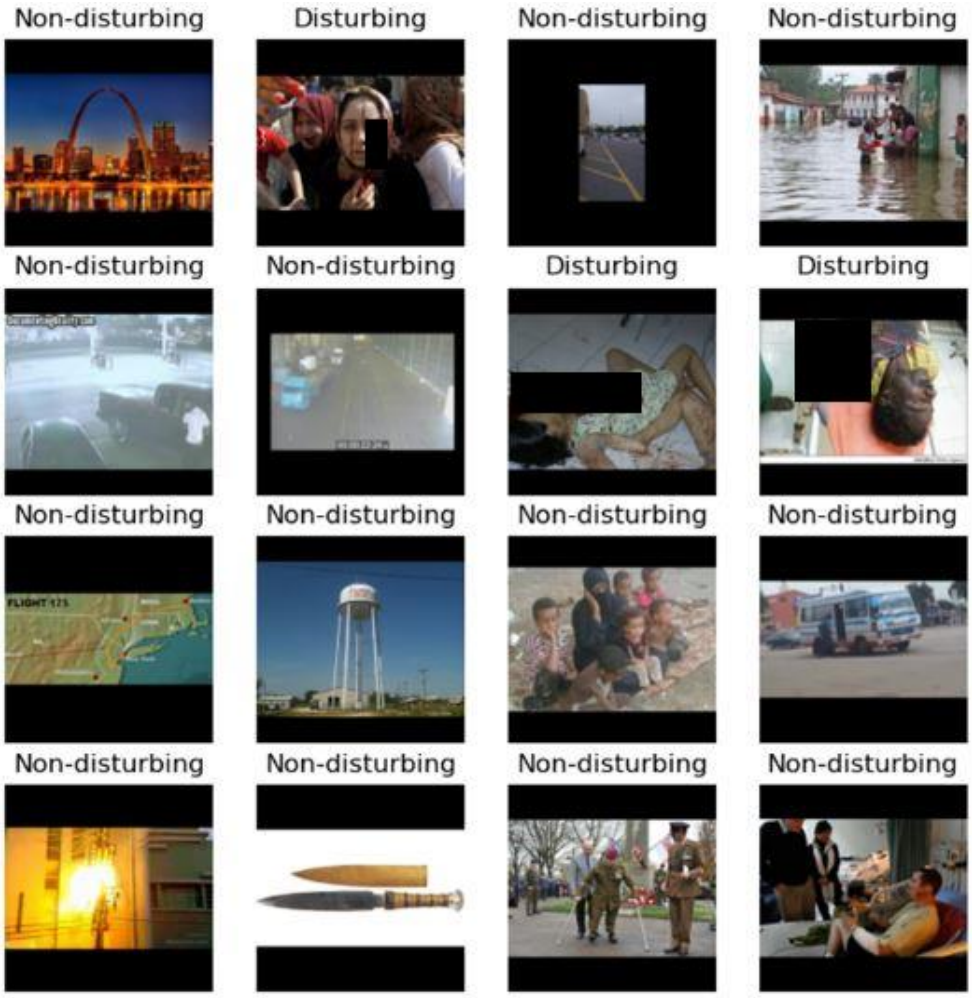


Figure 3: Sample images of the DID

2.2 Fine-tuning of Pre-trained Models

Training from scratch state-of-the-art models on image classification requires very large amounts of training data, which is not the case for the DID. For that reason, we opted for fine-tuning pre-trained state-of-the-art models. The two major models we used for this purpose are Convolutional Neural Networks (CNNs) and Visual Transformers (ViTs). In particular, we conducted experiments using the following architectures:

- **ResNet** (He et al., 2016): A deep residual learning framework that addresses the degradation problem (i.e., increasing the network depth, accuracy saturates and degrades.)
- **MobileNet** (Howard et al., 2017): A depth-wise convolution-based architecture that allows for building efficient models.
- **EfficientNet** (Tan & Le, 2019): Scaling networks along the width, resolution and depth simultaneously.
- **ConvNeXt** (Liu et al., 2022): Optimizes the key components that contribute to the performance of CNNs.
- **ViT** (Dosovitskiy et al., 2021): Leverages transformers for addressing image classification task.
- **gMPL** (Liu et al., 2021): A simple gated MLP approach that includes no self-attention mechanism.
- **ResMLP** (Touvron et al., 2021): A residual based architecture built entirely upon multi-layer perceptrons.
- **XCiT** (Ali et al., 2021): A version of self-attention operating across feature channels rather than tokens.

Having split the DID into 80%/20% training/test sets, we conducted several experiments to evaluate the above models using various hyperparameters and training techniques. Specifically, we tried Adam, AdamW, and SGD (with 0.9 momentum) optimizers with a learning rate from 1e-1 to 1e-6 with or without a scheduler (StepLR, CosineAnnealingLR, and LambdaLR). The batch size was 64, and the models were trained for 50 epochs. Table 1 summarises the best performance for each model type. The Efficientnet-b1 model achieved the best accuracy.

Table 1: Performance of various fine-tuned networks on DID

Model	Image size	Optimizer	Learning rate	Scheduler	Warm up	Accuracy (%)
resnet-50	256	SGD	1e-3	-	-	91.67
efficientnet-b1	256	Adam	1e-4	Cosine	-	92.31
efficientnet-b1	256	Adam	1e-4	-	-	**92.50**
efficientnet-v2 b1	256	Adam	1e-4	Cosine	-	89.35
efficientnet-v2 b1	256	SGD	6e-4	-	-	90.09
convnext tiny	256	SGD	1e-3	-	-	90.93
convnext tiny	256	Adam	1e-4	Cosine	-	92.31
gmlp_s16	224	SGD	1e-3	-	-	91.48
gmlp_s16	224	Adam	1e-5	-	-	90.65
gmlp_s16	224	SGD	1e-3	Cosine	Exponential	91.20
resmlp_24	224	SGD	1e-3	Cosine	-	91.30
ViT-B/16	224	Adam	1e-4	Cosine	Exponential	91.57
Xcit small	256	Adam	1e-4	-	-	91.94
Xcit small	256	Adam	5e-5	Cosine	-	92.05

2.3 Results Analysis and Model Improvement

Having evaluated the performance of several state-of-the-art models, we proceeded with a more in-depth study of the false positives and negatives of the best-performing model (i.e., Efficientnet-b1) and noticed that several images were wrongly labelled. Therefore, we re-annotated the corresponding samples in a semi-automatic manner. First, we split the dataset into eight folds and applied cross-validation using the Efficientnet-b1 network. After extracting the predictions for each fold, we manually inspected the labels of the false predictions, and we corrected the labels of the wrong images. In total, 146 were not correctly annotated, of which 66 were wrongly labelled (45 false negatives, 21 false positives), and the remaining 80 were difficult to classify due to their low resolution or border cases. We only changed the label of clearly wrongly labelled samples. After updating the ground truth, we re-trained the Efficientnet-b1, and its accuracy improved from 92.50% to 93.06%.

Although the achieved accuracy on DID is high, the dataset size (5401 images) poses a limitation to the model's effectiveness on real-world cases. In particular, models trained for such a task are prone to false positives, as the negative samples used during training do not have the necessary diversity to cover as many as possible real-world scenarios. Therefore, the generalization capability of these models is limited. For instance, images that depict meat (either cooked or raw) or a human sleeping can confuse a model, as it has not been trained on such negative samples. This limitation can be addressed by enriching both the negative class and the positive class (to maintain the balance between the classes) with new data. To this end, we tested the model on the YFCC100m dataset that consists of 99.2 million Flickr images to discover "confusing" samples that the model classifies as disturbing with high confidence. This experiment resulted in 281,458 false-positive images. Figure 4 presents

some difficult false positive examples that confirm the need to enrich the negative class of the training data. It is noteworthy that while inspecting these images we noticed that some of them are indeed disturbing, which is another indication of the difficulty and subjective nature of this task. We manually selected 802 positive and 2681 negative samples to fine-tune our model. Although the fine-tuned model reduced the YFCC100m false positives to 6,809 samples (from which 1,122 were indeed found to be disturbing), its accuracy on the DID test set dropped by 1.67% due to the imbalance of the training data (i.e., the negative samples are considerably more than the positive ones). As the balance between the classes is crucial, we populated the positive class of the training data with the 1,122 disturbing samples found above. Following this procedure, the training set more than doubled (106.65% increase) compared to the original training set of DID. Using this augmented version of the dataset, we re-trained the Efficientnet-b1 using the same hyperparameters. The final model achieves 93.79% accuracy on DID (i.e., 0.73% increase) and 91.38% on the 281,458 images of YFCC100m.

Figure 4: False-positive samples of the YFCC100m dataset

2.4 Class Activation Maps

Visualizing the class activation maps of the best performing model (i.e., Efficientnet-b1) can help us understand in which region of an image the model pays attention in order to judge an image as disturbing or not disturbing. To this end, we used the widely adopted Grad-CAM (Selvaraju, 2017) approach for extracting the activation maps. Figure 5 presents the class activation maps for four disturbing images. As illustrated, the model accurately focuses on the regions of the images that can cause anxiety to the viewer. On the contrary, only one small false activation is noticed for the negative samples, as depicted in Figure 6.

Figure 5: Class activation maps for disturbing images

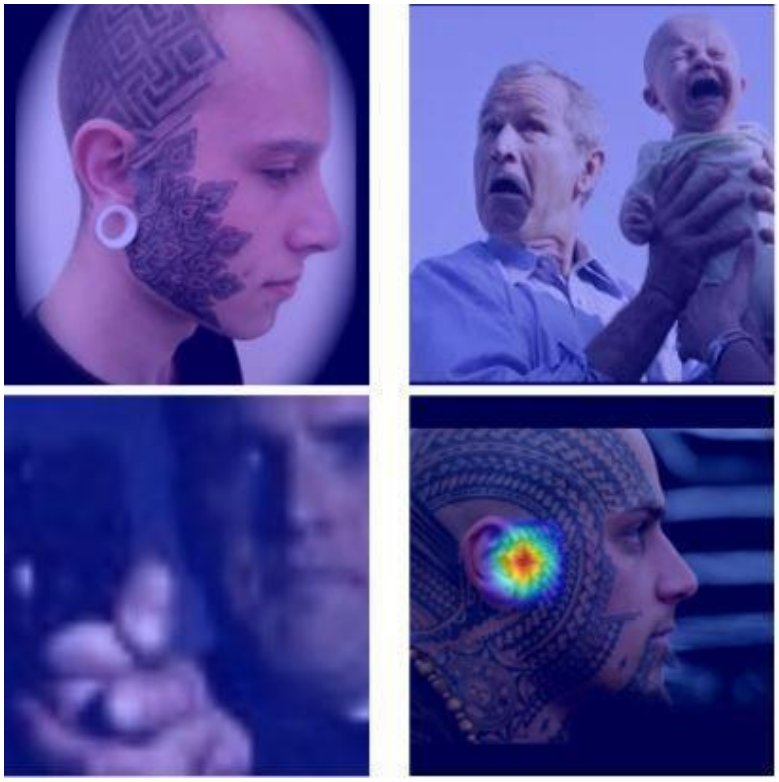


Figure 6: Class activation maps for non-disturbing images

2.5 Disturbing Content Detection in Videos

The trained model for disturbing detection developed for images is applied to videos by utilising the framework proposed in D3.2 - Content Discovery and Recommendation, Annotation and Adaptation Framework[5], section 1.2.1.3. Notably, scenes are extracted from each video and one frame is randomly selected as the representative frame of the scene. The representative frame is classified as disturbing or not, providing a label to the entire scene. For each scene, the provided information comprises the time interval of the scene, i.e., the scene start and end, and the corresponding disturbing confidence score.

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

3 Not Safe For Work Detection (NSFW)

NSFW content is mainly adult content, typically nudity and pornography. Making the Internet a safe place for people of all ages requires minimising the possibility of exposure to such content, especially by children. Other than that, pornography detection can be a valuable tool for enforcing legal obligations of digital service providers, especially in relation to child pornography and revenge porn. This section first provides a short overview of related scientific work in the area (Section 3.1) and then presents the datasets used as basis for our work and their pre-processing (Section 3.2). Section 3.3 describes the architecture selection, training and hyperparameter optimization of the NSFW detection models, while Section 3.4 presents the experimental results and an iterative improvement by incorporating semi-automatically generated annotations on a very large image dataset. Section 3.5 provides a comparison of the MediaVerse NSFW detection model with commercial alternatives, and Section 3.6 gives some evidence of the model's explainability. The application of the model to video content has been done in exactly the same manner as described in Section 2.5 for the disturbing content detection model.

3.1 Related Work

The early efforts to tackle the problem of pornography detection were mainly based on human skin features. Ap-Apid (2005) proposed a method based on colour spaces to indicate skin regions in an image and a classification model based on the size of the skin regions and their relative distances. Similarly, Ruiz-del-Solar et al. (2005) proposed a skin region detection method and a 28 skin-related feature extraction (orientation, size, solidity, position, etc.). Santos et al. (2012) argued that pornographic content tends to cover the middle region of an image, and proposed a method that splits the images into zones before extracting the skin features. Moreover, a forensic tool, named NuDetective was introduced in (de Castro Polastro et al., 2010). Lopes et al. (2009a) introduced a Bag-Of-Features (BOF) approach based on Hue-SIFT descriptors (Van De Sande et al. 2009) that achieved similar performance without including any shape or geometric modelling. Inspired by this idea, Lopes et al. (2009b) applied a BOF approach for detecting nudity in videos instead of images.

The discussed traditional approaches share the same shortcomings. First, the false positive rates are very high, as they are based on the exposed skin. Many types of Safe for Work (SFW) contexts include exposed skin, such as an image depicting people at a beach or a wrestling match. Second, the existence of pornographic content where the ratio of exposed skin is low makes those methods unreliable for positive sample detection.

The exceptional performance of Convolutional neural networks (CNNs) on visual tasks led to their use for the task of pornography detection. Moustafa (2015) presented the first CNN approach, where AlexNet (Krizhevsky et al., 2012) and GoogleNet (Szegedy et al., 2015) were fine-tuned on pornography data. Furthermore, Perez et al. (2017) introduced an approach for detecting pornographic content in videos by leveraging both static and motion information. Finally, Geremias et al. (2022) presented an effort to tackle the child sexual abuse detection problem, using a framework with two modules: one for pornography detection, and a second for age estimation.

3.2 Datasets and Pre-processing

The pornography-2k dataset (Perez et al., 2017) constitutes the most representative available dataset in the literature. It consists of 140 hours of 1,000 pornographic and 1,000 safe videos. Specifically, Pornography-2k is an extension of the Pornography-800 (or NPDI) dataset (Avila et al., 2013). The positive class includes both professional and amateur content, many genres of pornography, several races, and animated content. The negative class is of the greatest importance and comprises many videos associated with skin exposure. Apart

from the pornography-2k, the NudeNetData (Nudenet, 2018) constitutes an image NSFW dataset that consists of 483,495 NSFW, 191,998 SFW, and 27,059 "sexy" images.

Pornography-2k comprises videos, so pre-processing is necessary to extract video frames. This section focuses on models for NSFW detection in images; thus, we do not leverage any motion/time information. Taking into account that in many pornographic videos, there are parts that do not depict any NSFW content, we centre-cropped the videos so that 80% of their total duration remains. Then, one frame per five seconds was extracted for each video. This procedure resulted in 146,028 SFW and 114,368 NSFW images. Although the centre cropping contributed to labelling frames properly, we noticed that there were still samples labelled as NSFW that do not depict any NSFW content. For that reason, we applied a 50%-50% split cross-validation, manually inspected the false predictions, and corrected the wrong labelled samples. This re-annotation procedure resulted in identifying 11,150 samples wrongly labelled as NSFW.

Regarding the NudeNetData, we opted not to involve samples from the SFW class as they are easy samples to classify, while our purpose was to enrich the data with diverse and challenging samples. Taking that into account, we opted for the sexy class to enrich the negative class. This class is quite noisy as it consists of many images that depict nudity. Thus, we manually selected 15,000 images that do not involve nudity to populate the negative class of our data. Regarding the NSFW class, we randomly selected 15,000 images to increase the data diversity.

3.3 Model Architecture Selection and Hyperparameter Optimization

Selecting a proper CNN architecture for a specific task and dataset is crucial. Here, we opted for the EfficientNet (Tan, 2019) models, as they have been proven to be, one of the most effective CNN architectures on various visual tasks (as was also shown in Section 2). EfficientNets allow for scaling their width, depth, and input resolution. This enables selecting a variant that fits a given task and dataset. Although the computationally heavy variants (e.g., EfficientNet-b7) outperform the rest variants on big datasets, such as ImageNet, with many classes, smaller variants are recommended for either smaller datasets or few class tasks. Taking that into account, we expect that EfficientNet-{b0-b4} would provide the highest performance on the pornography detection task as (i) it is a binary task and (ii) the training data are <1 million.

We conducted several experiments to select the optimal EfficientNet variant. Pre-trained weights on the ImageNet dataset were used as the initial weights for all the conducted experiments. For a fair comparison between the different variants, several hyperparameter sets were used. Both Adam and Stochastic Gradient Descent (SGD) optimizers used initial learning rates between 0.0001-0.01, batch sizes 32 and 64, and the learning rate schedule proposed in BiG-Transfer (Kolesnikov et al., 2020). Table 2 presents the best results for each of the EfficientNet variants. As expected, smaller models tend to perform better on this task. Specifically, EfficientNet-b1 outperforms the rest of the variants on the pornography-2k frame classification task.

Table 2: Classification accuracy of the EfficientNet variants on the frames of Pornography-2k dataset

EfficientNet Variant	Optimizer	Learning Rate	#Steps LR {warm up, LR, LR/10, LR/100, LR/1000}	Batch Size	Accuracy
b0	SGD	0.001	{500, 2500, 3000, 3000, 1000}	64	93.16
b1	SGD	0.0005	{1000, 5000, 6000, 6000, 2000}	64	**93.78**
b2	SGD	0.0005	{1000, 5000, 6000, 6000, 2000}	32	93.56
b3	SGD	0.0005	{500, 2500, 3000, 3000, 1000}	32	93.14
b4	SGD	0.0003	{500, 2500, 3000, 3000, 1000}	32	91.37

Having selected the EfficientNet-b1 as the most suitable architecture for our purpose, we proceed to the hyperparameter tuning to find the most effective combination of the batch size and learning rate values. For this purpose, we used the Ray-Tune Python library. Specifically, we conducted 25 experiments with learning rate values between 0.1 and 0.0001, batch size in {64, 128, 256, 512}, and Adam or SGD as optimizers. Table 3 presents the top-5 models in terms of accuracy.

Table 3: Pornography-2k frames: Hyperparameter optimization of Efficientnet-b1 using the Ray-Tune library

Optimizer	Learning Rate	Batch Size	Accuracy
SGD	0.001064	128	**94.35**
SGD	0.004138	256	94.09
Adam	0.000318	64	93.93
SGD	0.001603	256	93.90
Adam	0.000128	32	93.89

3.4 Results Analysis and Model Improvement

After carefully selecting the model's architecture and optimising the training hyperparameters, we evaluated the trained model on the videos of the Pornography-2k dataset. The dataset providers offer five official 50%/50% train/test splits for evaluation purposes. Using the EfficientNet-b1 with the hyperparameters we found in Section 3.3, the mean video accuracy of the Pornography-2k dataset is 96.38%, higher than the best-achieved performance in the literature, which is 96% (Perez et al., 2017) when leveraging static and motion information. The remaining question is whether the 96.38% accuracy on the pornography-2k dataset makes the model accurate on images in the wild. Surprisingly the model's accuracy on the NudeNetData is 78.36%. This led us to extend the training data further, inspect them to correct the wrongly labelled ones, and further inspected the wrong predictions to better understand when the model fails.

Considering this model as a baseline, we populated the training data with 30,000 images from the NudeNetData. Training the Efficientnet-b1 with the same hyperparameters, the model exhibits increased performance on the NudeNetData test set and outperforms the previous model on the pornography-2k test data. In particular, the new model achieves 95.89% accuracy on frames and 97.7% accuracy on videos of the Pornography-2k and 90.72% on the NudeNetData test set. Afterwards, considering that our goal is not only to achieve the highest performance on the pornography-2k dataset but also to train a model that can effectively generalize (i.e., high performance on other datasets and real-world samples), we opted for modifying the train/test splits from 50%/50% to 90%/10%. Adopting this setup, the trained model achieves 99% accuracy on the videos, 96.41% on the frames of the pornography-2k dataset, and 92.12% on the NudeNetData test samples. Although these results are not comparable with the original 50%/50% splits, the increased performance on the NudeNetData indicates the enhanced generalization ability of the new model. Table 4 summarizes our experimental results.

In a real-world scenario where users upload images to a social media platform, the highest proportion of test images will be SFW, and only a very small proportion will be NSFW. Although the creators of the Pornography-2k dataset considered the importance of diverse and challenging SFW data, we should also consider the question of whether human and AI models judge the same examples as hard to classify. For instance, a human can characterise an image that depicts people wrestling as a hard case to classify, but other cases are extremely easy for humans while AI models fail in them. Figure 7 presents such an example. Several NSFW detection models fail to classify SFW as a basket of eggs, as the image colour and edges are similar to certain human parts. Specifically,

our model predicts the image of Figure 7 as NSFW with a probability of 81.22%, while the Vision AI API provided by Google predicts that this image is NSFW with the highest confidence.

Table 4: The results of the conducted experiments using the pornography-2k dataset and NudeNetData

METHOD	NUDENETDATA (TRAINING DATA)	ACCURACY		
		PORNOGRAPHY-2K		NUDENETDATA (TEST SET)
		FRAMES	VIDEOS	
Static+Motion (Perez et al., 2017)		-	96.4%	-
MediaVerse NSFW		92.84%	96.38%	78.36%
MediaVerse NSFW	✓	96.89%	97.7%	90.72%
MediaVerse NSFW	✓	**96.41**%	**99**%	**92.12**%

Figure 7: A sample that is easy for humans but confusing for an AI model

Motivated by the above observation, we tested our best performing model on the YFCC-100m dataset, which consists of approximately 100 million Flickr images in order to (i) evaluate the false-positive rate of the model on a wide variety of real images and (ii) further fine-tune the model on the derived false-positives.

We considered as false positives the images with NSFW scores higher than 0.8, as the target was to find SFW samples that the model classifies wrongly with high confidence. After feeding all YFCC100m images into the model, 65,275 images (i.e., 0.06%) were classified as NSFW with confidence greater than 0.8. These were split into 80%/20% training/test sets for further fine-tuning and evaluating our model. Table 5 presents the results. For the pornography-2k dataset, the final model accuracy on frames increased by 0.38%, while no improvement was noticed for videos (i.e., 99%). Regarding the NudeNetData test set, the accuracy increased by 0.51% (i.e., 92.63%), and as regards the YFCC100m test set, the final model achieves 98.99%.

Table 5: The model's performance before and after the fine-tuning on YFCC100m data

YFCC100M	ACCURACY			
	PORNOGRAPHY-2K		NUDENETDATA	YFCC100M
	FRAMES	VIDEOS		
	96.41%	**99%**	92.12%	0%
✓	**96.79%**	**99%**	**92.63%**	**98.99%**

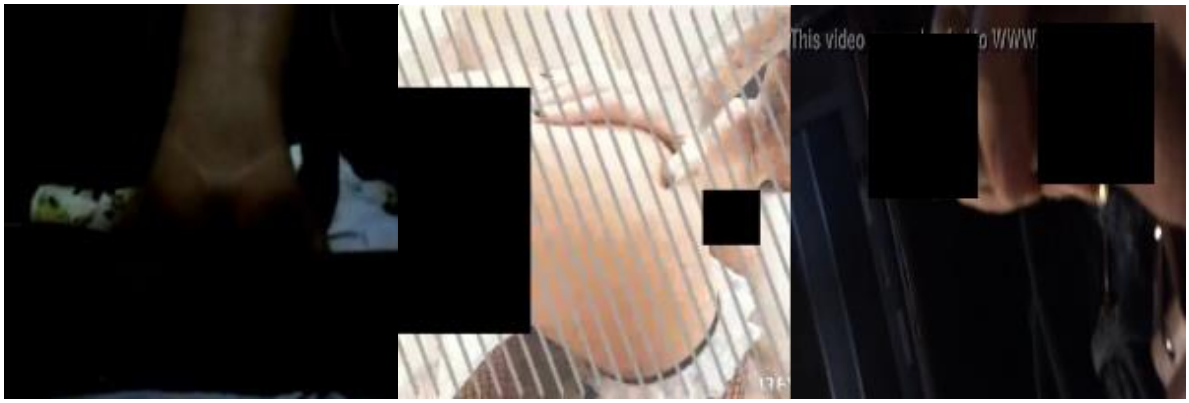
3.5 Comparison with Other Services

Here, we compare our final model predictions to those of other popular NSFW detection services. For this purpose, we selected some confusing examples to evaluate the effectiveness of each service. The services we used for the comparison are Google's Vision AI and the DeepAI NSFW Detector. Figure 8 presents the selected samples and the corresponding predictions of the services (3 NSFW and 3 SFW). For the SFW class, we opted for an image with much skin exposed, a wrestling image, and a breastfeeding image. Observing the results, both Google Vision AI and DeepAI APIs classify them wrongly as NSFW samples, while our model predicts the correct label. For the NSFW class, we selected images with low brightness and an image that has been intentionally modified to confuse the detectors. The DeepAI API fails to detect all the NSFW samples, while our model and the Google Vision AI fail to detect only the image that was adversarially modified to confuse the models.

(a) DeepAI: 0.7466 (NSFW), Google: 5/5 (NSFW), Ours: 0.1206 (SFW)

(b) DeepAI: 0.9417 (NSFW), Google: 3/5 (NSFW), Ours: 10^{-5} (SFW)

(c) DeepAI: 0.9532 (NSFW), Google: 5/5 (NSFW), Ours: 0.1248 (SFW)



(d) Low brightness example. DeepAI: 0.0431 (SFW), Google: 4/5 (NSFW), Ours: 0.6845 (NSFW)

(e) Sample edited to confuse models. DeepAI: 0.0138 (SFW), Google: 1/5 (SFW), Ours: 0.3453 (SFW)

(f) Sample depicting male genitals. DeepAI: 0.3484 (SFW), Google: 3/5 (NSFW), Ours: 0.8513 (NSFW)

Figure 8: The predictions of the proposed model, Google Vision and DeepAI NSFW Detector

3.6 Class Activation Maps

To demonstrate the capability of the final model to focus on the regions of interest in NSFW images, we visualise the class activation maps for both NSFW and SFW classes using the Grad-CAM (Selvaraju, 2017). In Figure 9, we notice that although the selected samples are challenging, the model does not confuse them for NSFW, while in Figure 10 the model accurately focuses on either male or female genitals and breasts in the NSFW images.

Figure 9: Class activation maps for SFW samples

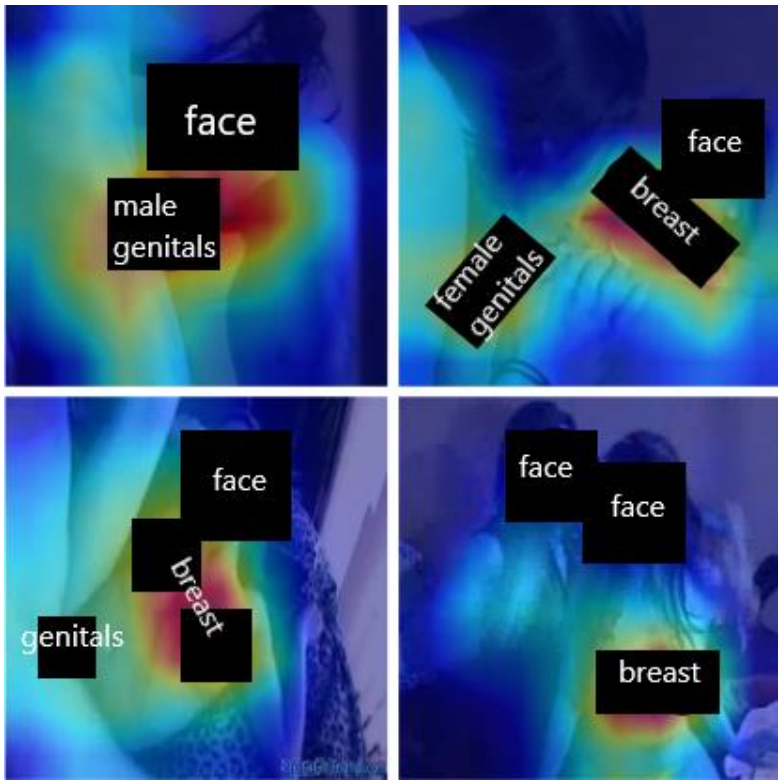


Figure 10: Class activation maps for NSFW samples

4 Deepfake Detection and Image Forensics

As part of our work in the content moderation toolset, we also refined and integrated in the toolset two media verification components from our previous work, namely the MeVer deepfake detection service (Baxevanakis et al., 2022) and the Image Verification Assistant (Zampoglou et al., 2015).

Deepfake detection applies to image and video assets, providing a deepfake score that indicates whether the image or video depicts deepfake manipulated faces. A famous example of deepfakes disseminated online is a video of President Vladimir Putin announcing, in March 2022, the end of Russia's war with Ukraine. A tweet sharing this video was analyzed by our deepfake detection component resulting in a deepfake score of 70%, labelling the video as a deepfake (Figure 11).

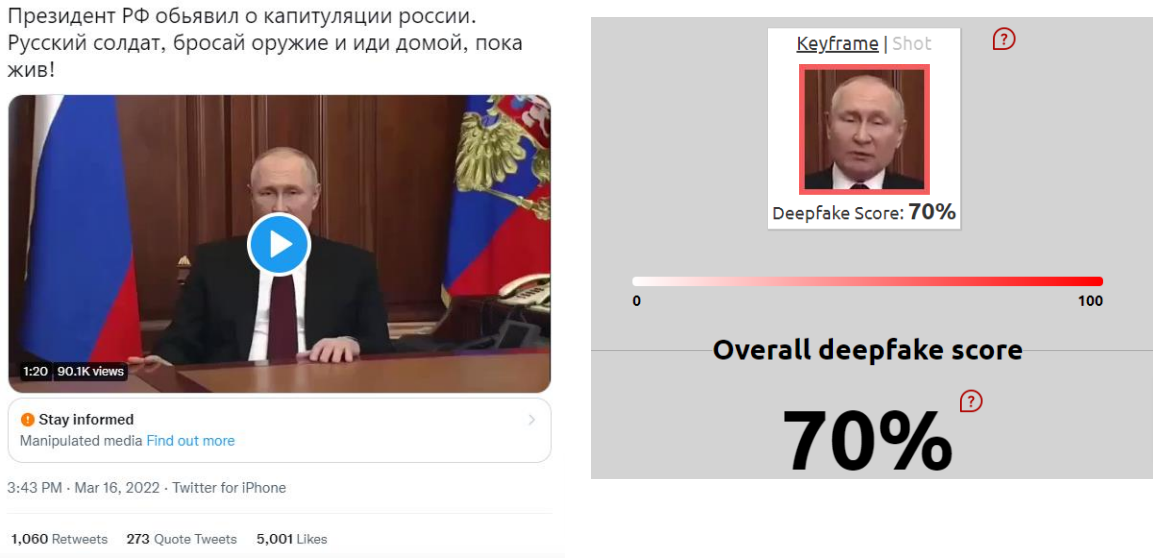


Figure 11: Example of using the deepfake detection component on a deepfake video shared through Twitter

Similarly, the image forensics component provides cues for traces of forgeries in images. Figure 12 presents an example of a manipulated image: the image shared on Twitter during the 2017 Catalan referendum contains a Catalan flag that was added to the photo using editing tools. At the right, the image forensics component highlights the added area in the image, making clear the manipulation to the end user.

Figure 12: Example of image forensics component detection

Differently from the disturbing and NSFW content detection components of Sections 2 and 3, the deepfake detection and image forensics tools are meant to be used "on demand" by MediaVerse users. This is due to the following reasons: a) the tools do not provide a definitive classification, but are meant to assist end users in their classification, b) deepfake and manipulated content is in most cases not disturbing or stressful for the viewer, and c) deepfake video detection is computationally intensive and it would be inefficient to apply it to the full set of media that is uploaded to a MV node.

4.1 Extensions of MeVer Deepfake Detection in MediaVerse

The MeVer deepfake detection service has been improved in terms of computational speed and efficiency. Specifically, we re-implemented the video segmentation process to calculate directly an overall deepfake score for a video, in contrast to the previous approach, which segmented the video and processed each segment individually to get a final score. The re-implementation of the video segmentation process offers a noteworthy speed-up and flexibility regarding the segments that the service and/or the user choose(s) to use.

In terms of efficiency, we first modified the face clustering component. The face clustering component helps reduce data noise as well as reject unimportant background persons that may be present in the video. The new procedure rejects any person(s) that do not comprise at least 10% of the total faces detected, whereas the previous one would reject persons if they do not appear in at least 20% of video frames. This change ensures that the service does not dismiss all detected faces even in videos where faces appear only for brief moments.

Figure 13 presents an example of the updated face clustering component compared to the existing. We used as query a lengthy video with few faces. On the left, the old methodology rejects all the identities found since their ratio of appearance is too low when compared with the much higher number of frames. On the right, the new procedure accepts the most frequent identity.

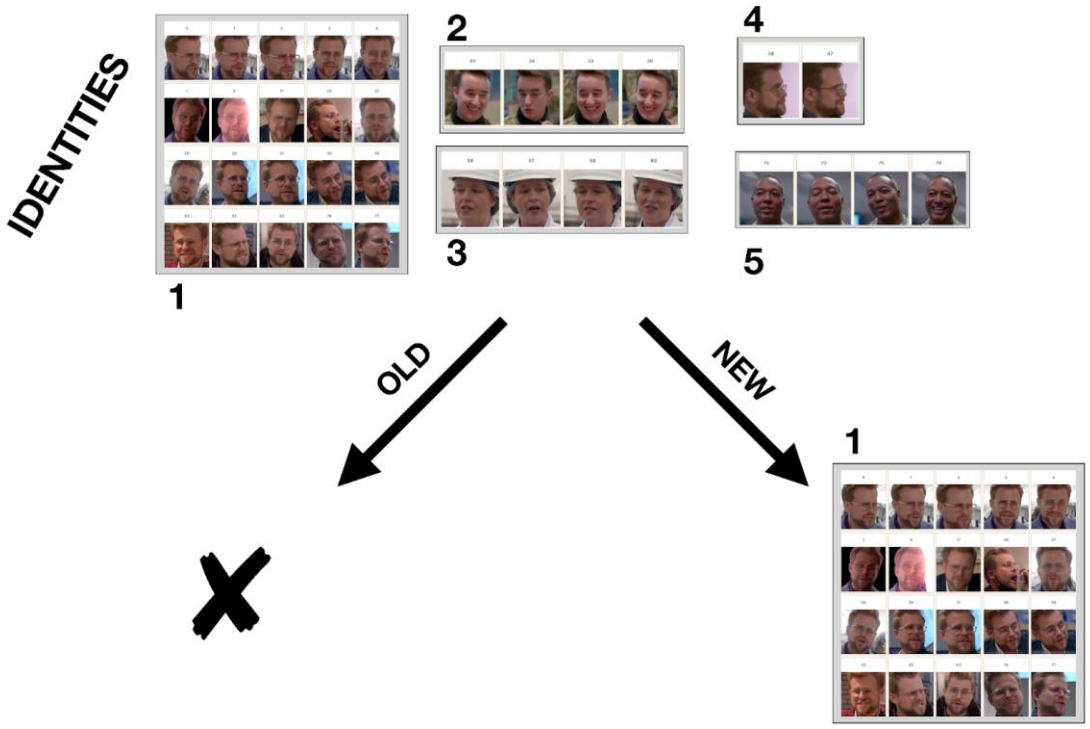


Figure 13: Example of the new face clustering component on a TV show trailer with very few faces

Besides the above changes, we chose to investigate the way the service aggregates face scores from the whole video - also known as an aggregation method - in order to output a final video prediction. First, the method checks how many confident real and fake predictions are in the video, with confident predictions defined as scores that are less than 0.2 for real and greater than 0.8 for fake predictions. Then, for any given person, if the number of confident face predictions for one category are at least twice as many as the other category's confident predictions, we set the person's deepfake score as the mean of that category; otherwise, the person's score is set as the mean of all its face predictions. Finally, after each person has a deepfake score, we define the total video prediction as the most confident person's score, in the case where there is a confident score, or as the mean of all persons' scores.

Finally, we performed extensive hyperparameter experiments to find values for each step in the deepfake detection methodology. This led to the discovery that lowering the similarity threshold of the face-clustering algorithm yielded much better results in terms of prediction confidence and accuracy. The previous threshold of 0.8 would result in fragmented person identities, meaning that a person's face detections would not be grouped up to a single identity but rather broken up to multiple smaller identities. Conversely, lowering the threshold to 0.65 results in proper person identification despite wide-ranging lightning and face angle conditions.

Additionally, we integrated a new functionality for query images motivated by the rise of synthetic generative models (Karras et al., 2020; 2021). Generative Adversarial Networks (GANs) have recently shown impressive ability in image synthesis (Goodfellow et al., 2014), introducing a challenging type of fake images, which makes it difficult for non-expert users to identify their authenticity. We expanded the tool to include a GAN-based generated image detector. Thus, for a query image the service provides the per-face and overall deepfake scores as well as a score that indicates whether the image is a GAN-synthesized image. To demonstrate the usefulness of the newly added functionality in assisting users to detect different kinds of manipulations, we used the 'this-person-does-not-exist' Random Face Generator[6], a well-known website that provides such images. Table 6 shows the scores for a synthetic image example calculated by the deepfake detector and the GAN-based generated image detector.

Table 6: Example of a synthetic image analysed by the deepfake and GAN-based generated image detectors

QUERY IMAGE	GAN-BASED SCORE	DEEPFAKE SCORE
	**99%**	**43%**
	**99%**	**6%**

[6] https://this-person-does-not-exist.com/en

4.2 Extensions of Image Verification Assistant in MediaVerse

The field of image forensics is a fast evolving area, where new image manipulation techniques and tools constantly emerge, requiring new and more sophisticated algorithms (Nabi et al., 2022). In that sense, we have expanded the set of algorithms included in the Image Verification Assistant with two approaches, allowing it to detect a wider range of manipulated images.

The first algorithm is the Noiseprint algorithm (Cozzolino et al., 2019), which extracts and models the noise pattern of an image. The extraction process is based on a Siamese CNN, to be followed by the modelling of noise patterns using Gaussian Mixture Models. Reporting the discrepancies in the noise pattern provides a powerful tool for detecting spliced images, i.e., images whose content was captured using multiple sources. This method improves over previous ones that utilise the noise pattern for forensics analysis, like the Splicebuster algorithm (Cozzolino et al., 2015). Figure 14 provides a case where the Splicebuster failed to localize the manipulation, while the newly introduced algorithm shows accurate results.

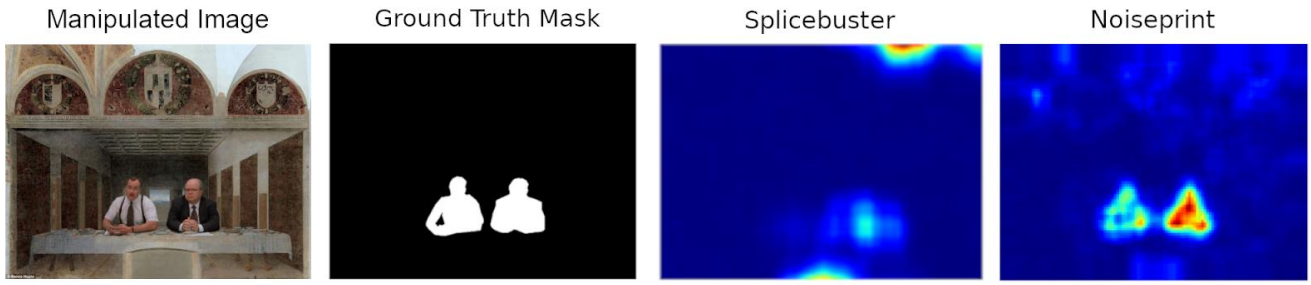


Figure 14: Image forensics: Noiseprint algorithm generated a more accurate map compared to Splicebuster

The second algorithm integrated in the image forensics service is the SPAN algorithm (Hu et al., 2020), which is an end-to-end deep-learning method for localizing the manipulated area of an image. Compared to previous deep-learning approaches, like Mantranet (Wu et al., 2019), it can capture the spatial relationships between different image patches by introducing a self-attention module, namely the Spatial Pyramid Attention Block. This addition enhances the detection capability of the forensics tool by detecting cases that previous methods could not spot. Figure 15 presents an example that compares Mantranet and SPAN on a query image.

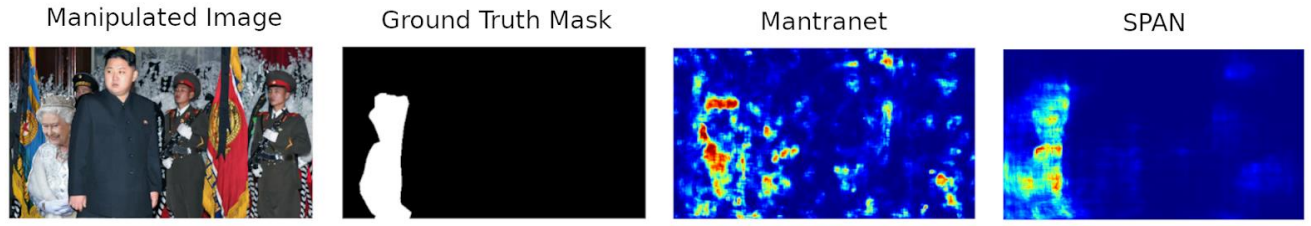


Figure 15: Image forensics: SPAN algorithm accurately detected the spliced area against Mantranet

The integration of the two algorithms enhances the capabilities of the image forensics tool but also increases the computational resources required to perform image analysis. Approaches that are based on deep neural networks require a large set of parameters and complex computations, resulting in the need for more processing power to analyse an image. To deal with this and avoid significant delays, we performed architectural improvements and efficiency optimizations in multiple algorithms.

To begin with, we developed a unified benchmarking tool to measure various properties related to the execution time and the detection capability of the forensics algorithms while automatically generating comparative charts. Given that all the implementations were containerized, we made the tool able to automatically manage the lifecycle and the environment of these Docker containers. Then, using that tool, we benchmarked the code of all the algorithms included in the forensics service and came up with a list of the most resource-intensive ones. After code reviews and extensive profiling, we spotted the most prominent directions for improvement and devoted effort to improve their efficiency.

Summarising the most important changes per algorithm, first of all, we dropped all legacy Java code included in our forensics service, namely the CAGI (Iakovidou et al., 2018), Median filtering (Justusson et al., 1981) and the Wavelet (Mahdian & Saic, 2009) microservices, and migrated them to high-performance implementations in Python. To achieve high computational gains, we paid great attention to the vectorization of the computationally intensive parts, using efficient libraries for vectorized numerical computations, such as NumPy. In that way, we achieved speedups compared to the previous implementations of 9 times for CAGI, 2 times for Median and 200 times for Wavelet. We also improved the implementation of the BLK (Li et al., 2009) algorithm through more efficient implementations of its internal median filtering, achieving a 4.5x speedup. Regarding the Mantranet (Wu et al., 2019) microservice, the initial implementation was based on a legacy and unsupported version of TensorFlow, hindering us from accelerating its execution on modern GPUs due to the requirement for outdated CUDA environments. Migrating to a PyTorch implementation, along with additional data handling improvements, led to a 35x speedup when executed on a modern GPU compared to the previous version executed on a 24-core CPU. Finally, we paid significant attention to the Splicebuster (Cozzolino et al., 2015) algorithm, improving its execution time by applying a set of both algorithmic and implementation optimizations. We heavily vectorised its feature extraction step and tuned the number of the Gaussian Mixture Models involved into its final stage after conducting an extensive study on its effect on the detection capability, achieving a speedup of 3.5 times while maintaining the detection capability of the original algorithm. Table 7 summarises the relative speedup over their previous implementations for all the optimized algorithms.

Table 7: Speedup of the image forensics over their previous implementations

ALGORITHM	SPEEDUP	ALGORITHM	SPEEDUP
BLK	4.5	Median	2
CAGI	9	Wavelet	200
Mantranet	35	Splicebuster	3.5

Along with the above implementation improvements to the algorithms, we also applied extensive architectural optimizations to the backend of the image forensics service. We re-designed all the microservices that serve the publicly provided HTTP endpoints of the service to enable them to efficiently exploit the parallel processing capabilities of the modern CPUs while reducing multiple unwanted delays caused by the previous IO pattern. These optimizations reduced response time when requesting the output of the analysis by more than 80% on average. Also, we improved the way the intermediate outputs from the various forensics algorithms were internally stored and cached, by switching from generic numerical representations, like serialized NumPy arrays[7], to data-specific representations, like standard image formats for visual data, that provide a much higher compression ratio, reducing the required storage by more than 50 times. Thus, after all the optimizations described above, we provide the users with a much more responsive tool, supporting many more concurrent users by scaling better to parallel hardware architectures and requiring fewer resources per image analysis, effectively decreasing its environmental impact by not requiring the deployment of more hardware.

[7] https://numpy.org/doc/stable/reference/generated/numpy.lib.format.html#module-numpy.lib.format

5 Hate Speech Detection

The extensive popularity of the Internet and social media has empowered the fast and anonymous spread of hate speech on platforms such as Twitter and Facebook. The existing regulation against hate speech, combined with the large amount of data on digital platforms, raise the need for tools that can detect Hate Speech (HS). Assigning the task only to people is not a practical option. A common alternative is to rely on user reports in order to review only the reported posts and comments. This is also inefficient since it relies on the users' subjective view, as well as their ability to thoroughly track and flag such content. Thus, the development of automated tools to detect HS content is necessary. The scope of our work in MediaVerse has been to study different text representations and classification algorithms in the context of HS detection.

A major challenge and important first step is to provide a clear and comprehensive definition for HS. It is a crucial process, especially during the manual compilation of HS datasets, where human annotators are involved, since HS detection is a highly subjective process and depends on the background of the annotator, such as education, culture and race. Therefore, a concise definition is required to minimize personal bias in the annotation process.

Following the definition by Brown (2017): "it covers all forms of expressions that spread, incite, promote or justify racial hatred, xenophobia, antisemitism or other forms of hatred based on intolerance. In addition, it can be insulting, degrading, defaming, negatively stereotyping or inciting hatred, discrimination or violence against people in virtue of their race, ethnicity, nationality, religion, sexual orientation, disability, gender identity". HS can also be expressed by statements promoting superiority of one group of people against another or by expressing stereotypes against a group of people.

The HS detection task can be formulated using a machine learning perspective, as a classification problem: given a dataset D consisting of N tuples (t_i, l_i), $i = [1 \ldots N]$, where t_i are short pieces of text from social media and l_i are annotations/labels, the task is, given an input text T, to output True, if T contains HS and False otherwise. Modelling the task as a binary classification problem, the detector is trained and evaluated using the standard supervised learning protocol. Furthermore, multiple categories can be supported (as in our case) to enrich the above simpler model and a multi-class model can be used (example classes: racism, sexism, etc.).

In the multi-class setting, classes correspond to hate speech categories. We seek a classifier $F(\cdot)$ to learn to assign short texts with HS tags, i.e. $F(t_i) = l_i$, with for $i = [1,\ldots,N]$. To achieve robust performance for application in real-world settings, the adopted solution should:

- Take advantage and consider diverse pieces of text within the scope of the task, to be able to achieve good generalization.
- Be easily extensible and configurable, to be able to adapt to changing and evolving needs in the industry.
- Be fine-tuned to optimal hyperparameters to achieve the best possible performance. Additionally, the fine-tuning process should be easy to monitor and track, as well as be interpretable by non-technical personnel to aid decision-making.

In this work, we focus on user-generated texts from Twitter. The target classes include:

- Racism (i.e., related to the race of an individual or a group)
- Sexism (related to expressions of misogyny / misandry)
- Sexual orientation (connected to sexual orientation and gender identity)
- Religious chauvinism (referring to the religion of an individual or a group)
- None

More specifically, the combined dataset is based on two separate datasets in the context of hate speech, namely CONAN[8] (COunter NArratives through Nichesourcing) and HKUST-MLMA[9] (Multilingual and Multi-Aspect Hate Speech Analysis) dataset.

CONAN is a multilingual and expert-based dataset of hate speech/counter-narrative pairs for English, French and Italian. It contains various types of metadata like expert demographics, counter-narrative types and hate speech topics. HKUST-MLMA is composed of a pilot corpus of 100 tweets per language, and comparable corpora of 5,647 English tweets, 4,014 French tweets, and 3,353 Arabic tweets. It contains information about the hostility type, the hostility directness (direct vs indirect), the target attribute (i.e., sexual orientation, disability), the target group (i.e., gay, immigrants, women) and the annotator's sentiment (i.e., shock_disgust). It should be noted that only the English subset from both datasets has been used in our analysis.

To build our dataset, we use the above public datasets annotated with HS labels and consisting of texts predominantly from Twitter. Additionally, we obtain data via the Twitter Python API, using lists of relevant post IDs. We subsequently merge partial results from all the different sources via a conversion and curation proess, applying data cleaning, label mapping and relevant information extraction. After this process, we arrive at the five labels of interest and a dataset of 15,680 instances. Table 8 provides some basic dataset statistics.

Table 8: Details of the derived dataset

LABEL	TRAINING		TEST	
	#instances	mean # words	#instances	mean # words
racism	2448	14.22	15	14.0
sexism	4213	15.57	15	17.53
orientation	677	12.78	15	12.47
religion	581	19.18	15	20.0
none	7761	13.99	15	16.53
overall	15680	14.59	75	16.11

The initial combined dataset was imbalanced in terms of number of instances per HS type. Techniques for under- and over-sampling have been applied to address this issue but with no significant performance improvement.

The above datasets were fed into a sklearn pipeline that implements the embedding process of the labelled sentences, which are then used in combination with the labels for the classifier training. The embedding is based on two different vectorization techniques: The first is based on the GloVe[10] 27.B model that contains pretrained 200d vector representations for 977K words. Each word in the sentence is replaced by the corresponding vectors and the sentence is represented by a combination of those vectors. The second embedding is based on the bag of words model using a word dataset containing 1545 English HS terms. The two embeddings are concatenated in a single vector that represents the sentence containing both HS and general semantic content. In this first version, we use two different classification models, as we describe below see also Figure 16.

As a baseline method, we employed the Logistic Regression, a statistical model commonly applied in binary text classification tasks. It produces a prediction via a linear combination of the input with a set of weights, passed through a logistic function, which squeezes scores in the range between 0 and 1, i.e., thus producing binary classification labels. Training the model involves discovering optimal values for the weights, usually acquired

[8] https://github.com/marcoguerini/CONAN#Multi-hate-target-knowledge-grounded-hate-countering-dataset
[9] https://github.com/HKUST-KnowComp/MLMA_hate_speech/blob/master/hate_speech_mlma.zip
[10] http://nlp.stanford.edu/data/glove.twitter.27B.zip

through a maximum likelihood estimation process. In our case, we used the expanded version of the logistic regression that maps the outcomes to multiple classes (5 in our case).

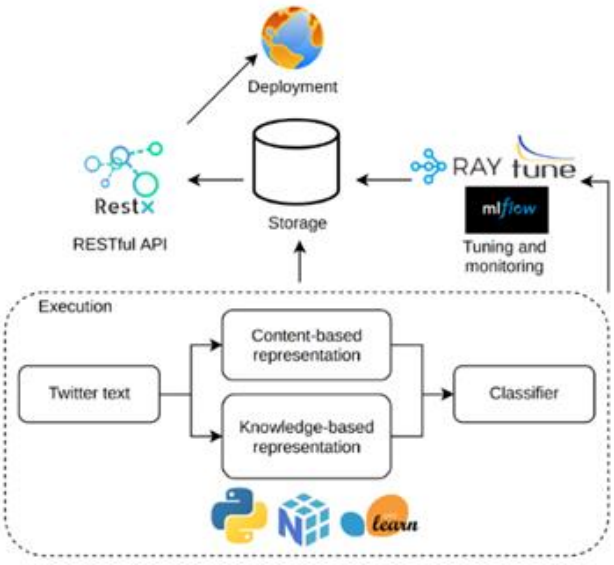


Figure 16: Overall architecture

As a second model, we used the multilayer perceptron (MLP), a feedforward artificial neural network model that maps input data sets to a set of appropriate outputs. An MLP consists of multiple layers and each layer is fully connected to the following. The nodes of the layers are neurons with nonlinear activation functions, except for the nodes of the input layer. Between the input and the output layer there may be one or more nonlinear hidden layers. In particular, we used an MLP composed of four hidden layers that contain 512, 256, 128 and 64 neurons respectively. The network was trained for 50 epochs using a batch size of 200 instances and a learning rate of 0.001 with the Adam optimizer. Table 9 summarises the obtained results. The MLP model performs considerably better than the LR model with minor differences in the performance per class.

Table 9: Results of the two algorithms per class

MODEL	FI-SCORE	NONE	SEXUAL-ORIENTATION	RACISM	SEXISM	RELIGIOUS
LR	0.77	0.845	0.849	0.785	0.607	0.774
MLP	0.916	0.956	0.925	0.915	0.883	0.899

For the tuning process of the models, we use the Ray framework[11], a Python library for experiment execution and hyperparameter tuning at any scale. We use Ray to increase our model performance by leveraging a variety of optimization algorithms, reducing the cost of tuning by exploiting the early stopping method, choosing better parameters to evaluate, and changing the hyperparameters during training using optimized schedulers. For workflow management, organization, performance monitoring and model storage, we used MLflow[12].

[11] https://docs.ray.io/en/master/tune/index.html
[12] https://mlflow.org/

6 Content Moderation User Interface and Integration

A new User Interface (UI) was implemented for the administration of the moderation rules of each node and user. At the moment, this UI is different from the dashboard of the MediaVerse node, which offers all the core user experience of the MediaVerse platform. The main functionality of the UI is the definition of the node and user moderation rules that will be applied during the upload and search of media content.

The individual models and services that constitute the content moderation toolset are integrated either as REST APIs or through the media annotation service presented in D3.2 - Content Discovery and Recommendation, Annotation and Adaptation Framework[13]. Specifically, the deepfakes and image forensics tools are implemented as REST APIs while the disturbing and NSFW detection models are exposed through the media annotation service.

6.1 Module Integration

The moderation UI is built on the React[14] v.18.2.0 library. For the application's state management redux[15] v.4.2.0 is used. The design of the components is based on rsuite[16] v.5.1.0 theme. For styling the whole app, we make use of bootstrap[17] v.5.1.3 along with the sass[18] v.1.52.3 CSS preprocessor.

For integrating the UI with the rest of the modules of the MediaVerse node a dockerized version of the UI was created and pushed to the respective docker hub repository. Figure 17 presents the dockerization instructions. The Dockerfile uses a Node.js[19] docker image to build the application and an Nginx[20] docker image to serve the produced static files. The most important characteristic of the app regarding the module integration is the runtime-env-cra[21] v.0.2.4 library. The package is meant to be used in Docker or VM based environments, where the administrator has full control over how the application will start, meaning that it enforces reconfiguration during runtime, therefore no build per environment is required.

The *default.conf* file (default configuration) for Nginx was modified, as shown in Figure 18. We defined the /moderation-ui path as the starting point on which the Nginx will serve all the dist files that came from the app building process. The reason behind the subpath configuration is that any module in the MediaVerse node sits behind a Kong[22] Gateway application, which acts as a single entry point of the node and is configured to perform a subpath-based routing functionality, as shown in Figure 19.

[13] https://mediaverse-project.eu/wp-content/uploads/2022/07/D3.2-V1.0.pdf
[14] https://reactjs.org/
[15] https://redux.js.org/
[16] https://rsuitejs.com/
[17] https://getbootstrap.com/
[18] https://sass-lang.com/
[19] https://nodejs.org/en/
[20] https://www.nginx.com/
[21] https://github.com/kHRISl33t/runtime-env-cra
[22] https://konghq.com/

Dockerfile

```
# Stage 1 as builder
FROM node:14 AS react-build
# Make the folder
WORKDIR /usr/src/app
# Copy the package.json and copy the files
COPY package* ./
COPY . .
# Install dependencies and build the project
RUN npm i
RUN npm run build
# Production Env
ENV NODE_ENV=production

# Stage 2 - the production environment
FROM nginx:alpine
# Copies static resources from builder stage
COPY --from=react-build /usr/src/app/build /usr/share/nginx/html
COPY --from=react-build /usr/src/app/.env-production /usr/share/nginx/html/.env
COPY --from=react-build /usr/src/app/ngnix/default.conf /etc/nginx/conf.d/default.conf

RUN apk add --update nodejs
RUN apk add --update npm
RUN npm install -g runtime-env-cra

WORKDIR /usr/share/nginx/html
EXPOSE 80

# Start Nginx server
CMD ["/bin/sh", "-c", "runtime-env-cra && nginx -g \"daemon off;\""]

# docker build -t moderator-mediaverse .
# docker run --name moderator-mediaverse -p 3000:80 --env-file=./.env moderator-mediaverse
```

Figure 17: Dockerfile for moderator UI

ngnix > default.conf

```
server {

  listen 80;

  location / {
    root   /usr/share/nginx/html;
    index  index.html index.htm;
    try_files $uri $uri/ /index.html;
  }
  location ^~ /moderation-ui {
      rewrite ^/moderation-ui(.*)$ $1 last;
  }
}
```

Figure 18: Configuration file for nginx

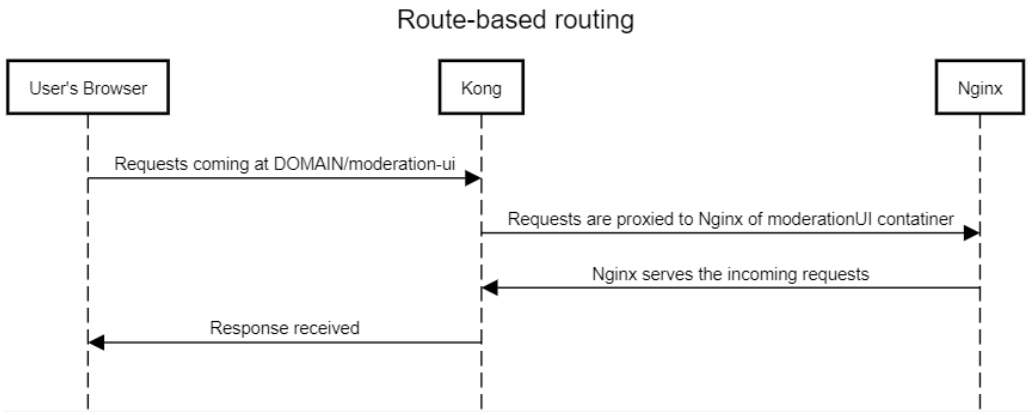


Figure 19: Routing sequence diagram

The dockerization process was automated using a Jenkins[23] instance that automatically builds the moderation UI image and pushes it to the docker hub repository. Figure 20 shows the Jenkinsfile holding the automation script.

Jenkinsfile

```
node{
    def appName = 'moderatorui'
    def imageVersion = '1.0'
    def patchVersion = ("${env.BUILD_NUMBER}".toInteger())
    def imageTag = "mediaverseeu/${appName}:${imageVersion}.${patchVersion}"
    def app

    checkout scm

    stage('Build image') {
        app = docker.build("${imageTag}")
    }

    stage('Login to Docker hub & push image to registry') {
        docker.withRegistry('', 'mediaverse-dockerhub') {
            app.push()
            app.push('latest')
        }
    }


}
```

Figure 20: Jenkins file for deploying moderator UI image

[23] https://www.jenkins.io/

6.2 User Experience

The moderation UI is an external tool with its own authentication system to authenticate MediaVerse users, as shown in Figure 21. That means that the same user can log in on both moderator and MediaVerse apps because both apps authenticate the users using the DAM API. An additional feature to the authentication flow is a controller that detects if a user is already logged in the MediaVerse app. In that case, the routing system makes sure to skip the login page and to navigate the user directly to the rules page. For a successful login, the UI asks from the DAM API if the credentials match a valid MediaVerse user. That user could be either admin or a creator. A creator user has restricted access to the moderator UI, in contrast with an admin user of the node. If the login action is successful, the app navigates the user to the rules page; otherwise, an error message is displayed.

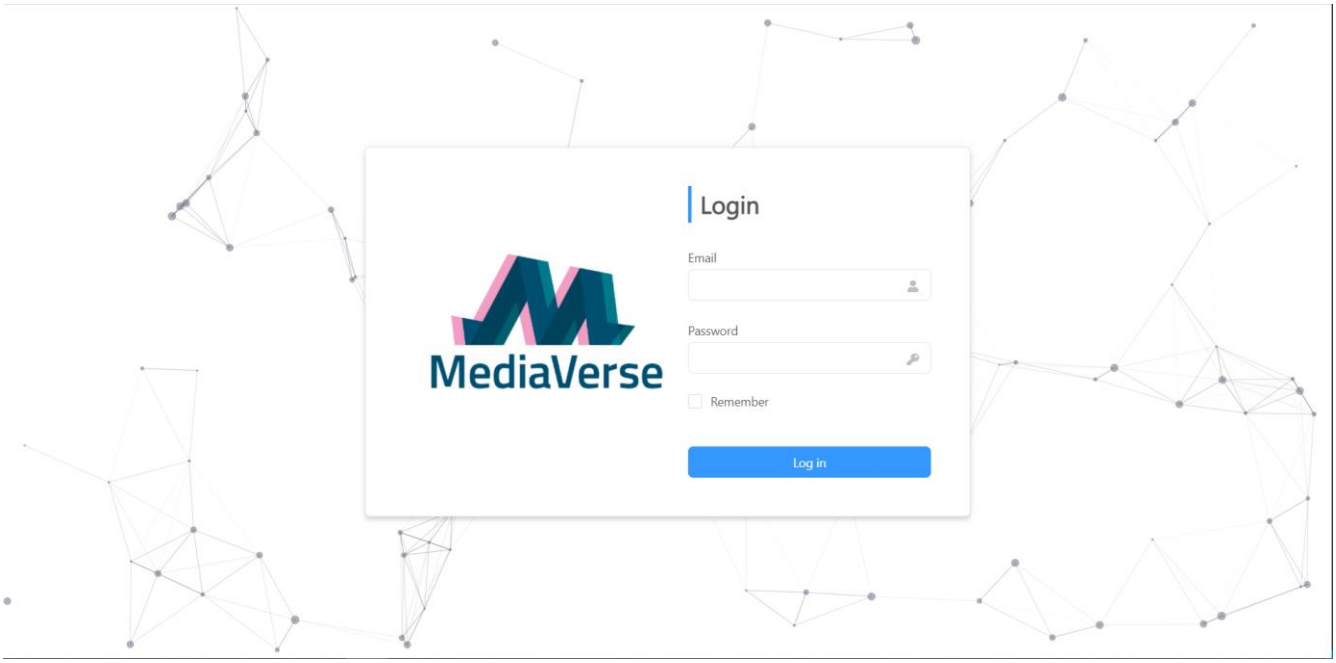


Figure 21: Login screen of the moderation UI

The user is able to modify the rules for them or the node by selecting the second tab on the left sidebar. By default, the node rules section opens, user rules are closed, and the total number of active rules is displayed. To edit the node rules, the user must be an admin of that node. Otherwise, he/she has only view access, see Figure 27. In the beginning, each section includes a simple selecting card for adding new rules, as shown in Figure 22. The user can select by clicking or searching and clicking one of the available rules (Figure 23). The user is not able to add two rules with the same name. By clicking the add button (Figure 24), a new rule is pushed to the current directory, and by default the confidence of that rule is set to 90%. To remove the rule, the user can click the checkbox on the top right of each rule card. The user is able to set their own strictness level for the rule by dragging the slider (from 0-100%) or by setting a default tag confidence level as shown in Figure 25. The default predefined confidence levels are Extremely Low (20%), Low (40%), Normal (60%), High (80%) and Extremely High (100%). Each rule card has a shadow effect of informing the user that the confidence level has been edited. After editing the rules, the user can either submit his/her changes by clicking the save button or discarding his/her changes by clicking the refresh button. If the save action is successful, a success message is popped up, or an error message is displayed in case there is any error status alongside with the error message (Figure 26).

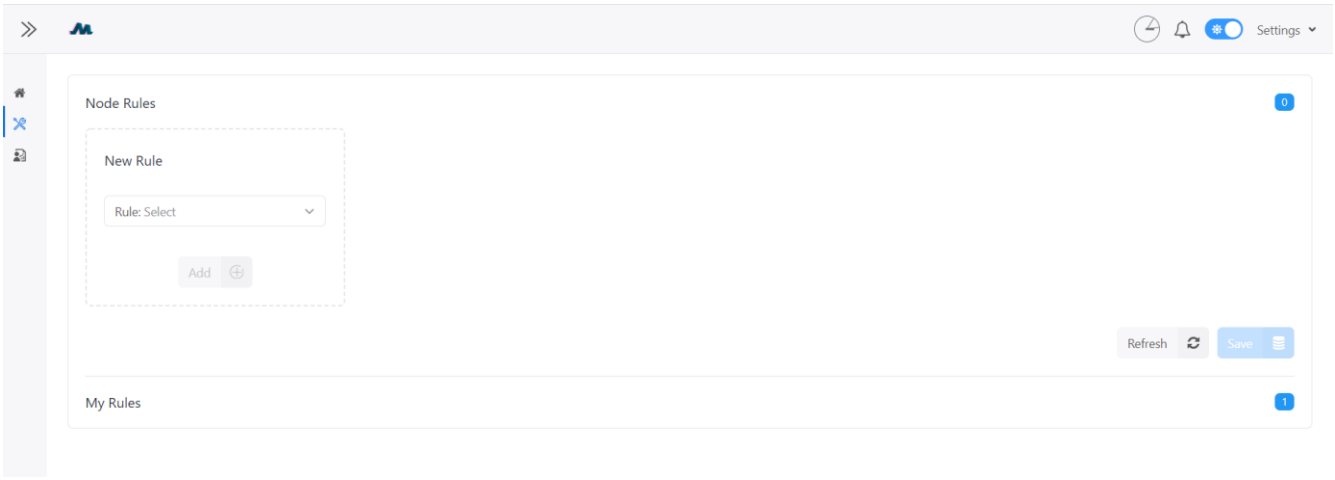


Figure 22: Rules page

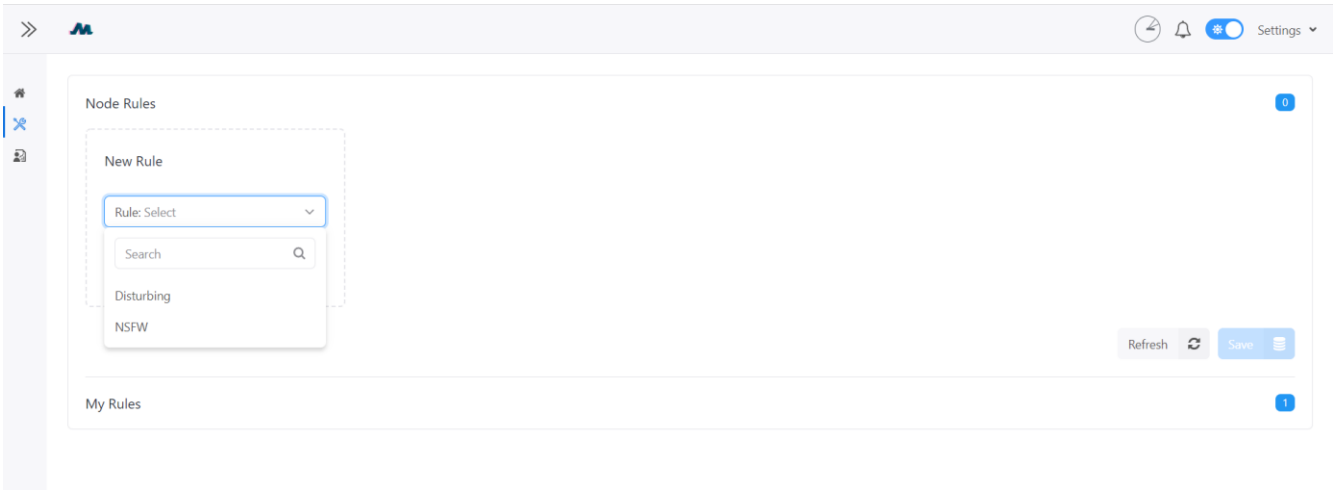


Figure 23: Choosing an available rule

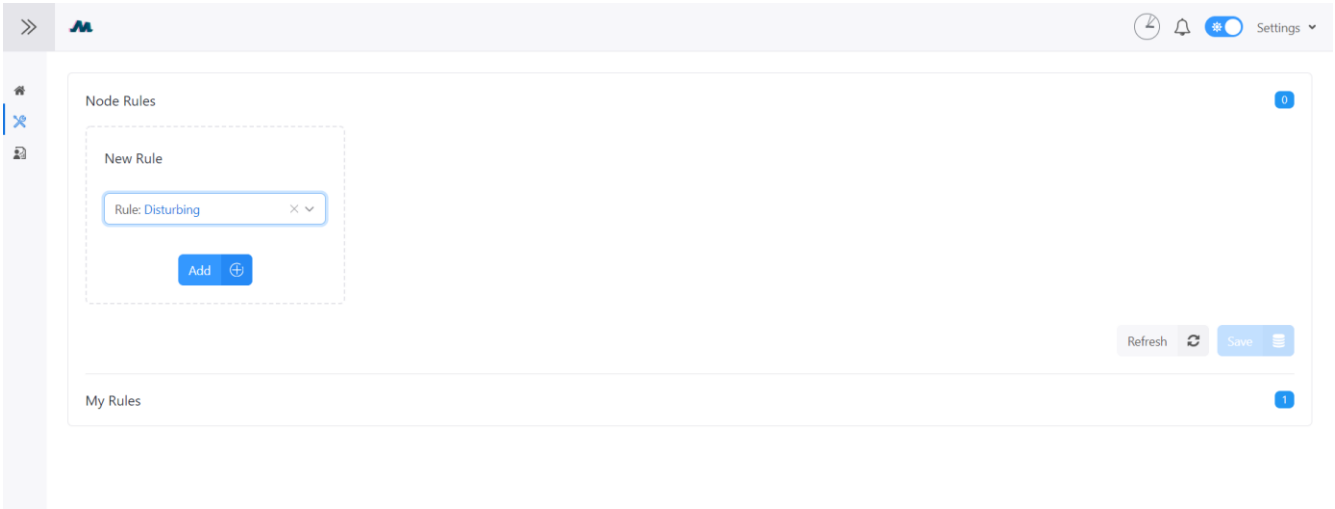


Figure 24: Adding a selected rule

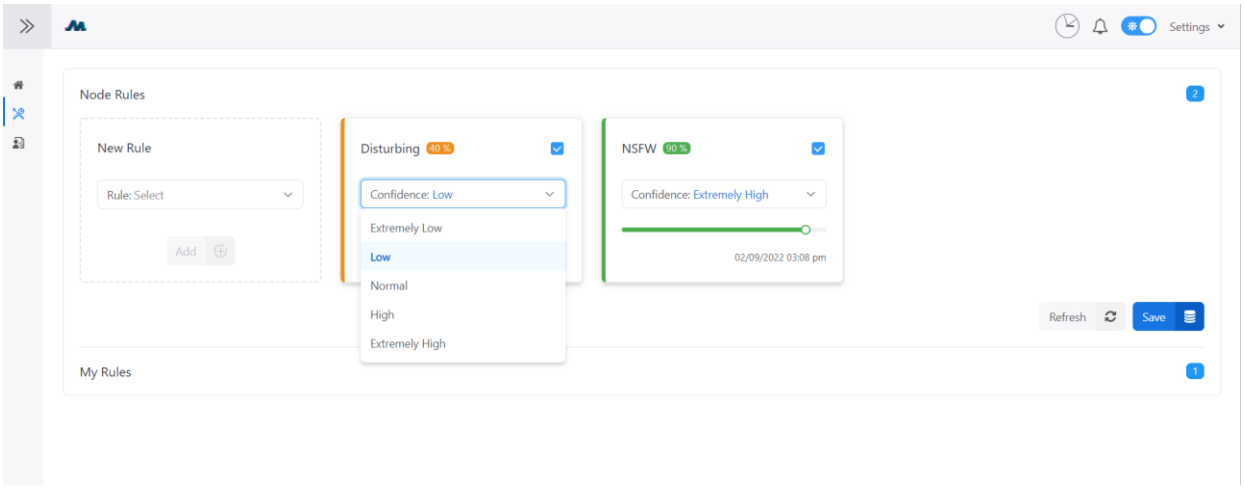


Figure 25: Selecting a predefined confidence level

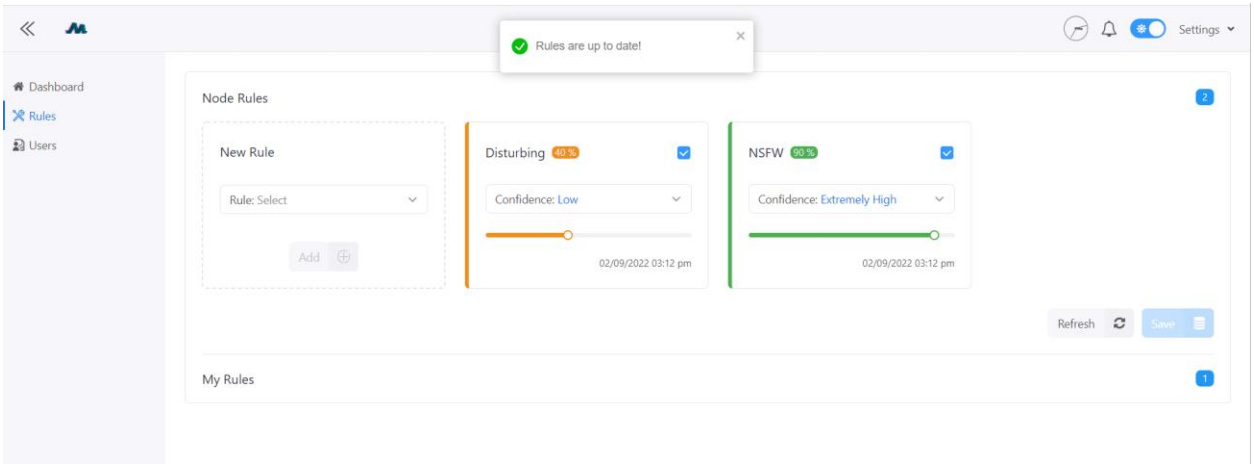


Figure 26: A successfully saving action for node rules directory

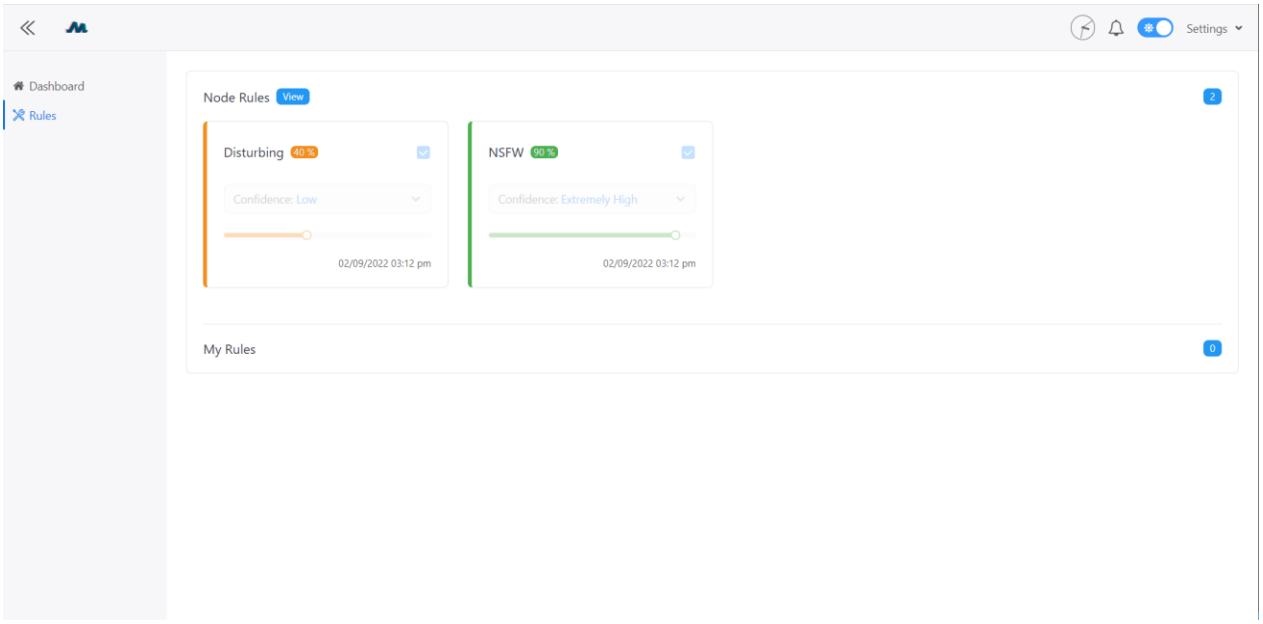


Figure 27: View mode for node rules for a MediaVerse user

7 Next Steps

During the final year of the project, we will work on the improvement and optimization of the presented components. Specifically, for the disturbing content detector, we will investigate methods and tools to alleviate the feeling of worry, concern, or anxiety to the viewer. Regarding the UI, a feedback mechanism will be developed so the node administrator or selected users are able to provide new annotated items and fine-tune the underlying models in a streamlined fashion. Finally, we will evaluate and optimize the existing moderation rules so that no technical expertise is required by the node administrator in order to define and modify them.

8 References

Ali, A., Touvron, H., Caron, M., Bojanowski, P., Douze, M., Joulin, A., ... & Jegou, H. (2021). Xcit: Cross-covariance image transformers. Advances in neural information processing systems, 34.

Avila, S., Thome, N., Cord, M., Valle, E., & AraúJo, A. D. A. (2013). Pooling in image representation: The visual codeword point of view. Computer Vision and Image Understanding, 117(5), 453-465.

Ap-Apid, R. (2005, March). An algorithm for nudity detection. In 5th Philippine Computing Science Congress (pp. 201-205).

Baxevanakis, S., Kordopatis-Zilos, G., Galopoulos, P., Apostolidis, L., Levacher, K., Baris Schlicht, I., ... & Papadopoulos, S. (2022, June). The MeVer DeepFake Detection Service: Lessons Learnt from Developing and Deploying in the Wild. In Proceedings of the 1st International Workshop on Multimedia AI against Disinformation (pp. 59-68).

Brown, A. (2017). What is hate speech? Part 1: The myth of hate. Law and Philosophy, 36(4), 419-468.

Cozzolino, D., Poggi, G., & Verdoliva, L. (2015, November). Splicebuster: A new blind image splicing detector. In 2015 IEEE International Workshop on Information Forensics and Security (WIFS) (pp. 1-6). IEEE.

Cozzolino, D., & Verdoliva, L. (2019). Noiseprint: a CNN-based camera model fingerprint. IEEE Transactions on Information Forensics and Security, 15, 144-159.

de Castro Polastro, M., & da Silva Eleuterio, P. M. (2010, August). Nudetective: A forensic tool to help combat child pornography through automatic nudity detection. In 2010 Workshops on Database and Expert Systems Applications (pp. 349-353). IEEE.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020, September). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In International Conference on Learning Representations.

Geremias, J., Viegas, E. K., Britto Jr, A. S., & Santin, A. O. (2022, April). A motion-based approach for real-time detection of pornographic content in videos. In Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing (pp. 1066-1073).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139-144.

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

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Hu, X., Zhang, Z., Jiang, Z., Chaudhuri, S., Yang, Z., & Nevatia, R. (2020, August). SPAN: Spatial pyramid attention network for image manipulation localization. In European conference on computer vision (pp. 312-328). Springer, Cham.

Iakovidou, C., Zampoglou, M., Papadopoulos, S., & Kompatsiaris, Y. (2018). Content-aware detection of JPEG grid inconsistencies for intuitive image forensics. Journal of Visual Communication and Image Representation, 54, 155-170.

Justusson, B. I. (1981). Median filtering: Statistical properties. Two-Dimensional Digital Signal Prcessing II, 161-196.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and improving the image quality of stylegan. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 8110-8119).

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., & Aila, T. (2021). Alias-free generative adversarial networks. Advances in Neural Information Processing Systems, 34, 852-863.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., & Houlsby, N. (2020, August). Big transfer (bit): General visual representation learning. In European conference on computer vision (pp. 491-507). Springer, Cham.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

Li, W., Yuan, Y., & Yu, N. (2009). Passive detection of doctored JPEG image via block artifact grid extraction. Signal Processing, 89(9), 1821-1829.

Liu, H., Dai, Z., So, D., & Le, Q. V. (2021). Pay attention to mlps. Advances in Neural Information Processing Systems, 34, 9204-9215.

Liu, Z., Mao, H., Wu, C. Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A ConvNet for the 2020s. arXiv preprint arXiv:2201.03545.

Lopes, A. P., de Avila, S. E., Peixoto, A. N., Oliveira, R. S., & Araújo, A. D. A. (2009a, August). A bag-of-features approach based on hue-sift descriptor for nude detection. In 2009 17th European Signal Processing Conference (pp. 1552-1556). IEEE.

Lopes, A. P. B., de Avila, S. E., Peixoto, A. N., Oliveira, R. S., Coelho, M. D. M., & Araújo, A. D. A. (2009b, October). Nude detection in video using bag-of-visual-features. In 2009 XXII Brazilian Symposium on Computer Graphics and Image Processing (pp. 224-231). IEEE.

Mahdian, B., & Saic, S. (2009). Using noise inconsistencies for blind image forensics. Image and Vision Computing, 27(10), 1497-1503.

Moustafa, M. (2015). Applying deep learning to classify pornographic images and videos. arXiv preprint arXiv:1511.08899.

Nabi, S. T., Kumar, M., Singh, P., Aggarwal, N., & Kumar, K. (2022). A comprehensive survey of image and video forgery techniques: variants, challenges, and future directions. Multimedia Systems, 1-54.

NudeNet dataset. (2019, June 18). Archive. https://archive.org/details/NudeNet_classifier_dataset_v1

Perez, M., Avila, S., Moreira, D., Moraes, D., Testoni, V., Valle, E., ... & Rocha, A. (2017). Video pornography detection through deep learning techniques and motion information. Neurocomputing, 230, 279-293.

Ruiz-del-Solar, J., Castañeda, V., Verschae, R., Baeza-Yates, R., & Ortiz, F. (2005). Characterizing objectionable image content (pornography and nude images) of specific web segments: Chile as a case study. In Third Latin American Web Congress (LA-WEB'2005) (pp. 10-pp). IEEE.

Santos, C., dos Santos, E. M., & Souto, E. (2012, July). Nudity detection based on image zoning. In 2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA) (pp. 1098-1103). IEEE.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618-626).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.

Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., ... & Jégou, H. (2021). Resmlp: Feedforward networks for image classification with data-efficient training. arXiv preprint arXiv:2105.03404.

Van De Sande, K., Gevers, T., & Snoek, C. (2009). Evaluating color descriptors for object and scene recognition. IEEE transactions on pattern analysis and machine intelligence, 32(9), 1582-1596.

Wu, Y., AbdAlmageed, W., & Natarajan, P. (2019). Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 9543-9552).

Zampoglou, M., Papadopoulos, S., & Kompatsiaris, Y. (2015, June). Detecting image splicing in the wild (web). In 2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW) (pp. 1-6). IEEE.

Zampoglou, M., Papadopoulos, S., Kompatsiaris, Y., & Spangenberg, J. (2017, January). A Web-Based Service for Disturbing Image Detection. In International Conference on Multimedia Modeling (pp. 438-441). Springer, Cham.

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.