Universität Osnabrück
Fachbereich Humanwissenschaften
Institute of Cognitive Science

Study Project

Protect Your Privacy

Manar Ali
Janosch Bajorath
Christopher Bröcker
Léon Dankert
Reem Farah
Pelin Kömürlüoglu
Vera Proiss
Muhammad Faraz Rajput
Marvin Raspini
Milad Rouygari
Helena Sanna
Argha Sarker
Hendrik Timm

Cognitive Science Master Program
October 2021 - September 2022

Supervisors: Ph.D. Ulf Krumnack
M.Sc. Ann-Christin Meisener

Abstract

This document summarizes the ideation and development process of the GDPR-compliant image anonymization platform, carried out as a study project collaboration between the University of Osnabrück and LMIS, a company focused on industry-related research and development in the field of artificial intelligence. Within a year, we developed a GDPR-compliant product achieving the project's goal: anonymization of personally identifiable information in images. Specifically, the product can detect and de-identify persons, faces, tattoos, vehicles, license plates, text and computer screens. Six detection techniques (person, face, vehicle, license plate, text and computer screen) and four anonymization techniques (masking, blurring, face and tattoo anonymization) are combined to achieve these results.

Contents

1. Introduction

The data acquiring, storing, processing, and distributing procedures may lack transparency and become a liability to someone sharing their private information. To protect persons and prevent violation of their right to secure personal data, the European Union implemented the General Data Protection Regulation (GDPR) in May 2018, which provided a framework for data anonymization by-laws. This regulation mainly affected how companies were allowed to process and store the personal information of individuals, resulting in programs that work in line with the GDPR. To stay GDPR-compliant, a company has to adhere to the following criteria: irreversibility; impossibility to identify a person; impossibility to derive insights or discrete information, even by the party responsible for anonymization. According to GDPR, personal data is "any information related to an identified or identifiable natural person" [1]. Your name, location, or phone number are good examples. For instance, when ensuring the data is GDPR-compliant, companies applying computer vision techniques may use a numerical representation of an image to generate meaningful information and store the data irreversible to its original to prevent re-identification. Different techniques exist to de-identify data that fall under two categories: anonymization and pseudonymization. The anonymization removes the identifiable information entirely using methods such as masking and deep anonymization. The pseudonymization, too, removes identifiable data. However, this process is reversible. For instance, identifiable information can be replaced by a reversible process such as blurring, which implies that pseudonymized data is still subject to data protection regulations. In our study project, we aim to develop a product that can provide a solution to protect personally identifiable information in images. Specifically, it needs to detect and anonymize humans, faces, tattoos, license plates, texts, and computer screens. The product can apply masking, blurring, and deep anonymization techniques to the regions of interest (RoI) and preserve the rest of the image content. Last but not least, we named our product SHIELD, which stands for Secure Human Information processing for Ethical & Legal Data management, and carried out as a collaboration between the University of Osnabrück and LMIS, a company focused on industry-related research and development in AI.

2. Theoretical Background

The LMIS' proposed product idea was to develop an application capable of detecting and removing private information from the images 1. In the initial stages of our study project, we gained an overview of the current research in Computer Vision and GDPR-compliant anonymization techniques. Additionally, our research focused on collecting information on available training datasets and performing market research. We outline the results of our literature review in the following sub-sections.

2.1. Market Analysis

The market analysis was conducted in two steps: market research and SWOT analysis. The market research focused on the products that are similar to the product we aimed to develop. The SWOT is an abbreviation for Strengths, Weaknesses, Opportunities, and Threats. For our product, it meant identifying its competitive advantage or the areas in which it could perform well.

2.1.1. Market Research

The market research started with a brief literature review where we aimed to find out the latest improvements in the image anonymization field. This allowed us to filter out the old-technology products and focus on the more advanced services and products available on the market. Below are a few examples of keywords that were used for the product search: "image anonymization", "deep anonymization", "image privacy", and "image blurring".

The market research findings are shown in Table 2.1. In general, many individual tools aim to accomplish a specific task, such as anonymizing license plates, faces, or text. But no product could be used in a more complex situation, where different features within an image should be anonymized. Besides this, we could not find a product that facilitated the registration of individual anonymization preferences in a predefined database.

Table 2.1.: The products on the market

Name of product	**Product features**	**Used methods**	**Targeted costumers & areas of use**
Pixelate	Video redaction, CCTV	Blurring, pixelation	Surveillance
DENKNetze	Image or video pixelation	Blurring	Surveillance
BrighterAI	Image and video anonymization	Precision blur, deep anonymization	Public sector
UAI Anonymizer	Face and license plate anonymization	Blurring	Street scene data
Celantur	Bodies, License Plates, Faces, Vehicles	Blurring	Geospatial
BlurIt	Faces, Bodies, License Plates	Blurring	Automative
SightEngine	API for Anonymization	Detection, moderation, anonymization	General public
Playment	Anonymization	Blurring	General public
PlaygroundAI	Mobile anonymization	Blurring	iOS users
Image Scrubber	Mobile tool on Github	Blurring, painting	General public

2.1.2. SWOT Analysis

The table of the products on the market was the subject of the presentation on "market research" 2.1. The presentation informed us about the different solutions to similar products that already exist on to the market and showed their advantages and disadvantages. At the end of the presentation, a brief list of what is missing in the market was listed to give an idea of what we can include in our product to make it significantly better than the competitors in the market.

The market research and product planning process continued with the SWOT analysis. This helped to get a structured idea of the product to be developed, especially not only about what is included as a feature and how it makes a difference in the market, but also what should be avoided that could pose a threat and jeopardize the success of the product. However, as it was part of the design, the analysis was done on a hypothetical product. This hypothetical product served to establish an end goal for the final product. Since this product was somehow imaginary, the opportunities and threats could not be determined, so the

SWOT analysis focused only on strengths and weaknesses. The results of the SWOT analysis are explained in a paragraph for each section below. As a guide, a comprehensive list of questions was answered. The questions can be found online[1].

Strengths. Our product uses Computer Vision methods (AI and other knowledge-based systems) as key features to solve GDPR-related issues in video and image data. Our product provides users with customization options, making it different from the existing products, which are not flexible enough to give the users the freedom to tailor the system to their specific needs. The client can choose what to anonymize in their images. Therefore, our unique selling point is adjustability. Further, we implement features that make our product stand out as a valuable product. One of such features is that we take semantics and/or the context of the scenery into account to achieve privacy in different conditions. We believe our product is a great help, especially in the logistics, production, and maintenance sectors.

Weaknesses. As it is the nature of developing a product, ours also have some missing features. The most important one is that our product focuses only on the GDPR, while some other products focus on different privacy laws in other regions and countries. We also do not introduce non-commercial or personal use, such as posting images on social media accounts and websites. Even though we strive for customizability, the diversity of our solutions may lead to instability. The product cannot perform at its best if many features require more time than usual to achieve the desired result. It also makes it challenging to maintain the high quality of the product. Furthermore, the product is not competitive because the market is already saturated with solutions, and the customers might not want to change the product they are familiar with. These weaknesses matter because if the performance degrades, the GDPR agreement cannot be guaranteed, and the customer can get into legal trouble.

2.2. Datasets

Datasets are a crucial part of machine learning models as algorithms learn patterns found in the dataset to predict future events. In our project, we looked into large datasets with a wide variability of topics. To ensure the high quality of our datasets, we needed to verify their completeness, consistency, uniformity, and constraints. With this in mind, the dataset investigation was split into two parts; one focused on the license restrictions, and the other on usability for potential product features. We explored several common license types for datasets: commercial and non-commercial.

[1] https://www.swotanalysis.com/blog/swot-analysis-on-a-product-or-service

2.2.1. Text Detection

Text can appear in various contexts carrying sensitive data. In each context, text detection holds some unique challenges. For instance, legal documents may be in multi-languages [2] or handwritten [3]. While detecting the text of license plates presents other challenges, such as the different layouts [4]. Research on text detection faces several challenges emerging in all contexts, such as multi-oriented text, meaning it appears anywhere on the image. It is also primarily focused on English, so there are a large number of datasets available. It explains the lack of multilingual datasets, which is a major problem for detecting multilingual text [5]. Thus, we choose RRC-MLT-20 [2] since it includes 20,000 natural scene images with embedded text in 10 languages, such as street signs, passing vehicles, and shop names. It also contains a synthetic dataset of 277,000 generated images with synthetic text of multiple scripts embedded in the background, as shown in Figure 2.1.

(a) Arabic Scene Text (b) Bangla Scene Text (c) Chinese Scene Text

(d) Japanese Scene Text (e) Korean Scene Text (f) Latin Scene Text

Figure 2.1.: Samples of the synthetic training dataset of RRC-MLT-20 2

Figure 2.2.: Samples of LPDC2020 [4]

[2] https://rrc.cvc.uab.es/?ch=15&com=downloads

The synthetic dataset helps with the training because the set of real scene images does not contain many images [6]. For multinational license plate detection, we selected LPDC2020 [4] since this dataset contains 29030 images of license plates in different languages with either single-line or double-line layouts as shown in Figure 2.2. The dataset contains license plates from the following countries: Turkey, Europe, USA, KSA, and UAE.

2.2.2. Logo Detection

Logo detection (LD) localizes and identifies logos in images, as shown in Figure 2.3. Some practical applications of LD are brand protection, brand-aware product search, recommendation, copyright infringement detection, and contextual content placement [7] [8].

Figure 2.3.: Detected logo

LD is a complex problem for Computer Vision. Logo comes in many variations and shapes, some of which may have synthetic bodies and others may have rigid bodies. Moreover, logo detection can be highly challenging due to deformation, background clutters, occlusion, low resolution, text logos, and different layouts of the same brand [9] 2.4. Considering these challenges, we set out to find an open-source dataset that is resourceful and ideal for our project. Traditionally, LD is treated as a closed-set object detection problem, in which the system is trained for a predefined set of logo classes and can only recognize these classes at test time [8].

During the research for logo detection and with the project's goal in mind, we came across the FlickrLogos-32 dataset [10]. FlickrLogos-32 was developed for logo retrieval, multi-class logo detection, and object recognition. It is one of the publicly available and widely used datasets. It is monitored at the object level with 32 classes and 8240 images. Figure 2.5 shows some examples of logos taken from the dataset. This dataset also contains pixel-level annotations, i.e., binary masks and bounding boxes that mark the position of the logo in each image [10].

Figure 2.4.: Challenges of logo detection

(a) ALDI

(b) Corona Extra

(c) DHL

Figure 2.5.: Examples from FlickrLogos-32

The images of the dataset have been divided into three subsets: P1 (training set), P2 (validation set), and P3 (test set). The training set consists of hand-selected images showing a single logo against different backgrounds. The test set was created with non-logo images to ensure better accuracy for the trained model. Given the properties and the design of the dataset, FlickrLogos-32 is an ideal choice for our project's potential use.

2.2.3. Biometrics Detection

In biometrics, we measure the statistical analysis of unique physical and behavioral characteristics of individuals [11]. There are three types of modalities in biometrics: hard, soft, and hidden. Hidden biometrics are based on medical data such as MRI or X-Ray images.

Hard biometrics describe features such as faces, signatures, and fingerprints. Soft biometrics include height, weight, skin color, hair color, facial measurements, tattoos, hats, and glasses [12]. In our research, we decided to focus on hard and soft biometrics and looked into datasets that contain faces, signatures, tattoos, and other soft biometrics.

Figure 2.6.: WIDERFACE dataset [13]

Faces can appear in different scales, genders, and ages, to only name a few variabilities in the facial images. Also, the image itself can be of high or low quality. In face detection, multiple datasets try to tackle the mentioned problems and many more. Some of the datasets designed for face detection are as following. WIDERFACE [3] to capture different poses, occlusion, expressions, makeup and illuminations in images. FDDB [4] for low resolution images. 4k face [5] for high resolution images. MAFA [6] for occlusion. Adience [7] for different ages and genders. AFW [8] contains images from various appearances. UFDD [9] for different weather. Wildest faces[10] for violence or fight images. Pascal face [11] for large faces. MALF [12] for multiple facial attributes.

All of these datasets are for academic research purposes only and do not allow commercial use except for the 4k face dataset. However, the images of the 4k dataset were collected from the internet with a keyword search, and from personal experience, these images are rarely checked for licenses [14].

[3] http://shuoyang1213.me/WIDERFACE/
[4] http://vis-www.cs.umass.edu/fddb/
[5] https://arxiv.org/abs/1804.06559
[6] https://imsg.ac.cn/research/maskedface.html
[7] https://talhassner.github.io/home/projects/Adience/Adience-data.html
[8] https://docs.activeloop.ai/datasets/afw-dataset
[9] https://ufdd.info/
[10] https://arxiv.org/abs/1805.07566
[11] https://arxiv.org/abs/1804.10275
[12] http://www.cbsr.ia.ac.cn/faceevaluation/

The most prominent dataset in Figure 2.7 is the WIDERFACE dataset (2.6), which contains a very large number of faces and images compared to other datasets (2.8) [13].

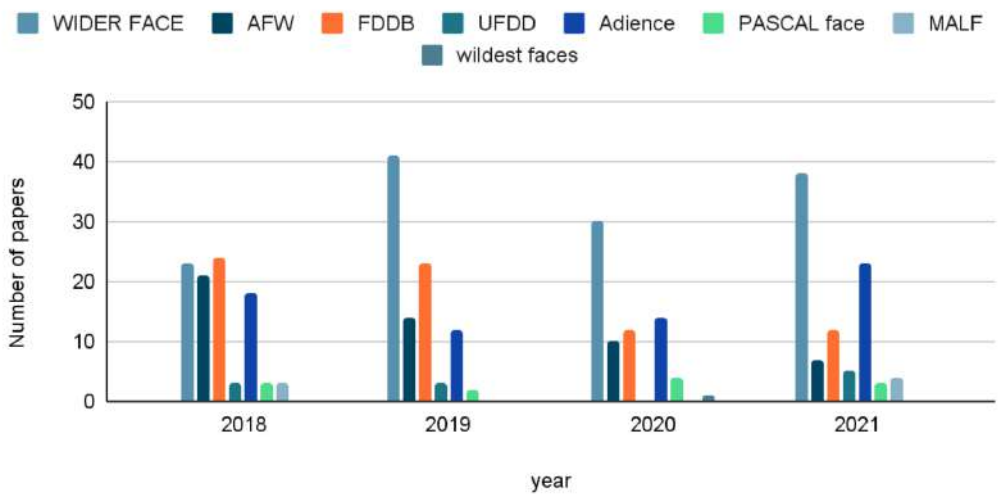

Figure 2.7.: Published papers in the last 4 years of different face detection datasets; data from [15]

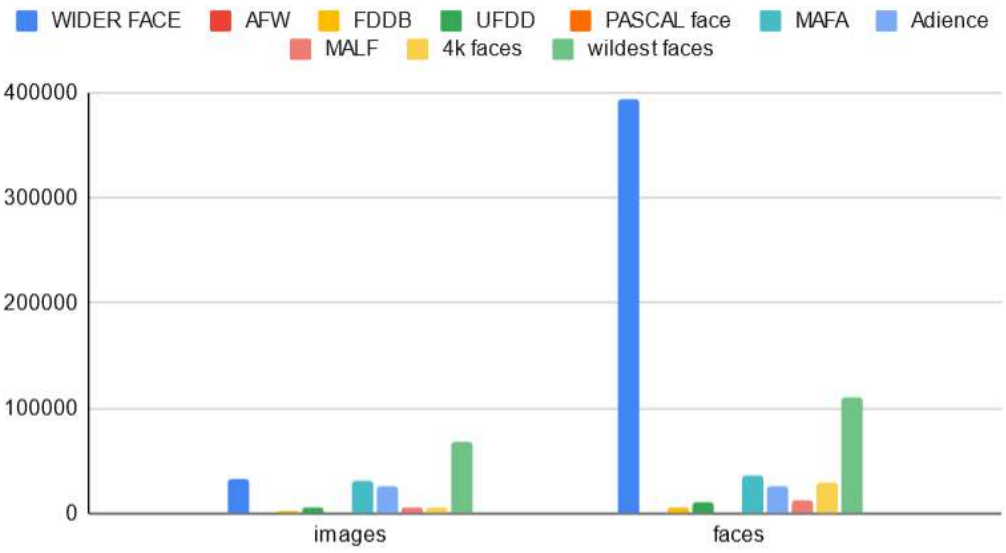

Figure 2.8.: Number of faces and images in different face detection datasets; data from [15]

2.2.4. Person Detection

Removing people from images an essential anonymization when it comes to anonymizing personal data. In the past, there has been a lot of research to create ever larger datasets for object classification. In 2014, Microsoft Research published the COCO dataset [16]. The main problems it aims to solve are the detecting of non-iconic views of objects, contextual reasoning between objects, and the precise 2D localization of objects.

This is achieved by applying individual object instance segmentation. Most object classification datasets are labeled only by bounding boxes, but the COCO dataset labels each image with individual instance segmentation. For the project, this is helpful because we want to detect the person as accurately as possible, so bounding boxes might not be sufficient. If a specific anonymization method requires bounding boxes (e.g., the kernel for blurring), we can simply calculate the bounding box from the segmentation mask.

Table 2.2.: Person detection dataset comparison [17]

	Caltech	KITTI	CityPersons	COCOPersons	CrowdHuman
# images	42,782	3,712	2,975	**64,115**	15,000
# persons	13,674	2,322	19,238	257,252	**339,565**
# ignore regions	50,363	45	6,768	5,206	**99,227**
# person/image	0.32	0.63	6.47	4.01	**22.64**
# unique persons	1,273	$< 2,322$	19,238	257,252	**339,565**

In Figure 2.2, we can see the comparison between different human detection datasets. The COCO dataset is the largest dataset with the most images. However, another interesting dataset is the CrowdHuman dataset [17]. It does not specialize in the sheer amount of images but tries to increase the number of people per image. There are 4.5 times more people per image than in the COCO dataset. One disadvantage of the CrowdHuman dataset is that it only provides bounding boxes and no segmentations.

The terms of use of the COCO dataset state that the annotations are licensed under *Creative Commons 4.0*. The images are from Flickr [18], so users must follow the Flickr terms of use. For the CrowdHuman dataset, there is no publicly available information on how the licensing works.

2.3. Image Segmentation for Text Detection & Character Recognition

Some of the end goals of this project are to remove written information from a given image, such as sensitive financial information (e.g., bank accounts, transition overviews, customer details), signatures, confidential statements, declassification of official documents, and more. To achieve this, text detection and localization, and text recognition are required.

During the literature review on image segmentation for text detection and character recognition, several research questions were considered to provide a better understanding and a way to achieve our goals. For example, what are the unique challenges of text detection and character recognition, what are the existing methods that are specifically adapted to this topic, and what are the general image segmentation methods that are suitable for our purpose.

Natural scene text detection and recognition aim to detect and locate texts [19] [20]. Scene text detection and recognition consist mostly of three tasks: text detection and localization, text recognition, and end-to-end text recognition system [19] [20].

The goal of text detection and localization is to determine whether the text is present in the given image or video and localize it. Existing methods for text detection and localization can be broadly categorized into three major groups: connected component (CC)-based methods, texture-based methods, and Deep Learning-based methods.

Text recognition, which may not be so relevant for the project, focuses on recognizing the text in a text candidate under the assumption that the text has been successfully detected [21]. Text recognition methods fall broadly into three categories: character-based methods, word-based methods, and sequence-based methods. For more information about these methods, see Table 2.3.

An end-to-end text recognition system is a combination of two tasks: text detection and text recognition [21]. A complete framework consists of input of the target image, then text detection and localization, text recognition, and in the end, the output of the result.

There are several challenges with text detection methods that can affect the detection process. For example, texts can be often scattered in the scene image, and there is no prior information about their position. In addition, scene texts may often have a variety of sizes, fonts, and orientations. Not to mention, the quality of scene images acquired by digital

devices may be poor and many character-like patterns (non-character) in scene images can be mistaken for text. [19] Figure 2.9.

Fig. 1 Examples of challenges of scene text detection. **a** Different orientations; **b** different languages; **c** different colors; **d** different size; **e** complex background; **f** occlusion; **g** blur; **h** noise; **i** non-uniform illumination

Figure 2.9.: Challenges of text detection and recognition [19]

The state-of-the-art techniques currently used for scene text detection and recognition methods are mostly Deep Learning-based method which has been widely used in semantic segmentation and general object detection [20]. Semantic segmentation-based detectors extract text blocks from the segmentation map generated by a Fully Convolutional Network (FCN). In further steps, the bounding boxes of text are obtained by complex post-processing. General object detectors, on the other hand, predict candidate bounding boxes directly by considering texts as objects. Unlike common objects, texts have a clear definition of orientation that should be predicted in addition to the axis-aligned by bounding box information. Some

examples of Deep Neural networks used in semantic segmentation and object detection can be: FCNs [22], *You Only Look Once* [23], and FASTER R-CNN (Region-Based Convolutional Neural Networks) [24]. However, there are other methods for text detection and localization such as the connected component (CC)-based, texture-based, sliding window method. The first method can find small components and combine them into one large component, then filter out non-text components by the classifier, and finally extract text from the image and combine it into the text [25] [26] [27] [28]. The second method is based on the idea that text in an image has distinct textural properties that can distinguish it from the background [29] [30]. The third method first detects text information by moving a multiscale sub-window through all possible locations in an image and then uses a pre-trained classifier to determine if any text is contained within the sub-window [22]. There are two representative methods, namely stroke width transform (SWT) and maximally stable extremal regions (MSER).

Table 2.3.: Text recognition methods [20]

Method	**Strength**	**Weakness**
Character classification based [31] [32]	Be insensitive to font variation, noise, blur and orientation	Rely on complex heuristic rules or language models
Word classification based [33]	Can effectively recognize words in scene image with a large number of class labels	Rely on a lexicon and hardly to handle long word with deformation
Sequence based Lee & Osindero [34] [22] [31]	Do not rely on the precision of text segmentation, and can process arbitrary strings	Need to design proper objective function to optimize the network parameter

2.4. Image Segmentation for Person Detection

2.4.1. SeMask

We began our search for background information on segmentation by searching for articles using the keywords "semantic segmentation", "segmentation", "person segmentation", and "deep learning". In the results, we skimmed some articles, selecting them by publication date and citations, and looked for potentially interesting cited sources in the skimmed articles. Finally, we settled on "Panoptic Segmentation"[35], in which the authors divide segmentation into: semantic segmentation (class label per pixel), instance segmentation (mask per object and class label per pixel), and panoptic segmentation (class and instance

labels per pixel). In further research, however, the newly introduced panoptic segmentation is usually treated under the term semantic segmentation.

In the aforementioned paper [35], several datasets for panoptic segmentation were presented, namely Cityscapes, ADE20k, Mapillary Vistas, and COCO. Since all datasets contain the class "Person" in a sufficient number of samples, we selected the dataset based on its difficulty level and current research interest. Since semantic segmentation is the category with the most contributions on Papers With Code [13] website, which can be used as a source for this task. We looked at the benchmarks and found that ADE20K is the most challenging dataset with recent improvements.

By skimming the best results for the dataset, SeMask[36] has been shown to produce the best results, using a promising approach that attempts to compensate for lost contextual information from the transformer. They also use licenses that allow use in commercial products and checkpoints for models trained on the ADE20K, Cityscapes, and COCO-stuff datasets.

2.4.2. SEG-YOLO

We decided to look for articles published in 2019 or later. Specifically, we looked at Papers With Code[13] website and Google Scholar. Our goal was to find the best model that fits our project. We found SEG-YOLO [37] is a highly efficient and lightweight model. It is faster than the Mask R-CNN model[38]. The model also can deal with poor lighting conditions and shadows. According to the author [37], speed and accuracy are very important aspects of image segmentation. Therefore, they used YOLO [39] with MOG2[40] and KNN [41] as background removal techniques. The latter two algorithms, although fast, produce only low-quality masks and the accuracy of the mask is unreliable. SEG-YOLO, on the other hand, produces highly accurate masks, and its speed also allows real-time usage.

2.4.3. CBNetV2

When looking for a model for human segmentation, we searched for review articles on image segmentation via Google Scholar using keywords such as "image segmentation survey" or "image segmentation overview" with the filter set to only show papers from 2019 or newer, as the state-of-the-art in popular tasks (like image segmentation) changes rapidly.

[13] `https://paperswithcode.com/`

Google Scholar search revealed that the paper of interest was "Image Segmentation in Deep Learning: A Survey" [42] as the other papers were about specific contexts (e.g., medical field). We then double-checked for other review papers by using Connected Papers[14] website with the mentioned article's title as the search term, and there again we could not find any other relevant survey papers more recent than 2019. Thus, we concluded our search for overview papers and looked at the critical parts of [42], i.e., the part listing the performances of the surveyed models. For each of the datasets in which performances were reported, we looked at the best models of the dataset's leaderboard on Papers With Code[13] website to see if the models in [42] were already outdated or not. As it turned out that even the best models from Minaee *et al.* [42] reported significantly worse performance than the best models from paperswithcode.com, we discarded the content of Minaee *et al.* [42] as being outdated and decided to consider only the models from the Papers With Code[13] website's leaderboards.

However, since models not tested on the same dataset are difficult to compare, we decided to look at and select a model from the leaderboard from only one of the datasets for (human) image segmentation. The criteria for selecting such a model were (in descending order of priority): size of the dataset (the bigger, the better, and especially the number of images with human masks was important); popularity of the dataset as a benchmark; diversity in the data so good performance on the test set can be expected which in turn lead to similarly good performance in real-world use.

Fortunately, there was one dataset that was a clear winner based on the mentioned criteria: The COCO dataset. On Papers With Code[13] website, there were further two more leaderboards for the Instance Segmentation task (the relevant task for our interest) for COCO: minival and test-dev. After reading the explanation on these two, we decided to look at the test-dev leaderboard because (as the name implies) it reported the test performance of models. From the leaderboard, we chose the CBNetV2 models as our model of choice for the following reason: It was ranked third on the leaderboard, but had the best performance with a trained model available for download and the license allowed commercial use of the model.

2.5. Removing Biometrics in Images

People have many unique characteristics that set them apart from others. Personal identifiers such as faces, voice, and gender are some of these features that distinguish a person.

[14] `http://connectedpapers.com`

Figure 2.10 shows different biometric (hard and soft) and non-biometric identifiers. The focus of this section was predominantly physiological biometrics (mostly faces) and soft-biometrics like tattoos. We examined these various characteristics and showed how the existing literature on biometric privacy address them.

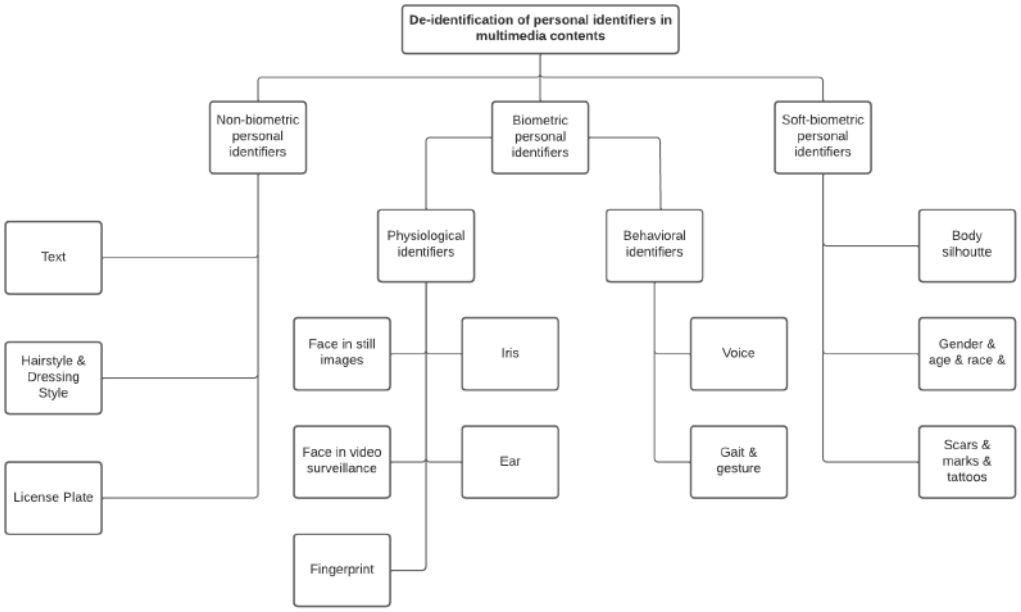

Figure 2.10.: Biometric and non-biometric identifiers [43]

Before we take a closer look into facial biometrics and its different methods for anonymizing, we need to look at the characteristics of biometric privacy enhancement techniques (B-PETs) and identify possible points where the enhancement techniques must be applied. Different facets of B-PETs are shown in Figure 2.11. We will briefly describe each category.

"Data" is the category that divides the privacy enhancement techniques into either videos or still images. In "Mapping" there are two types: Irreversible and Reversible. The former are techniques where the original data is hard (or ideally impossible) to reconstruct. The latter method of mapping is done using encryption and decryption. In biometrics, there are two forms of information that can potentially identify a person. This distinction involves "attributes." The first one is the “personal identifiers” known as "hard-biometrics" that can be used directly to link data to individuals such as faces. The second one is “Quasi–identifier” or "soft–biometrics" which cannot directly be used to identify the person but could increase recognition accuracy such as gender or ethnicity. The term "Utility" deals with how the privacy enhancement was done on data and is divided into two categories: reduction and re-

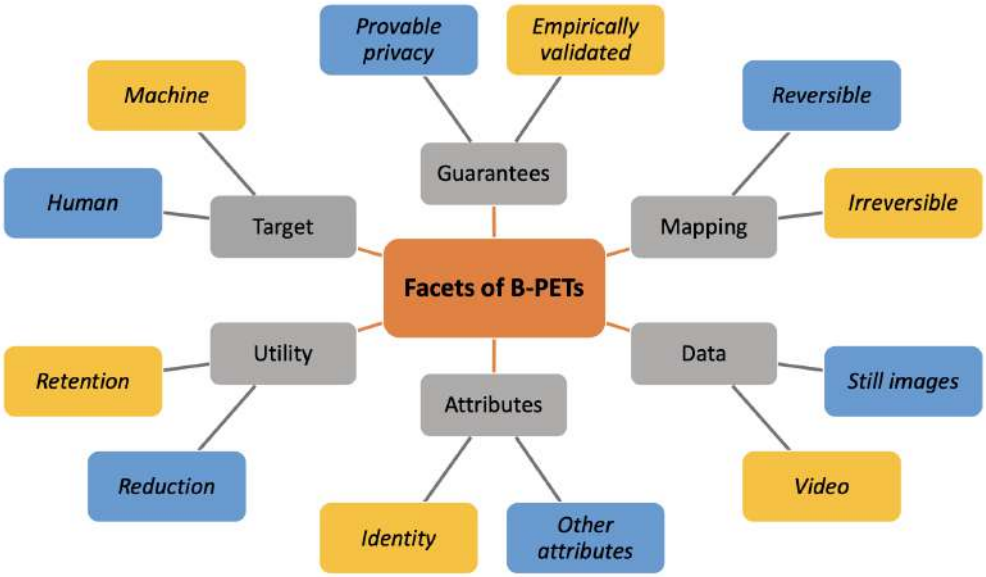

Figure 2.11.: Facets of biometric privacy's [44]

tention. "Reduction" is the term used to describe the removal of sensitive information. For example, masking faces. "Retention" is the term used for replacing the original data with surrogate data as a substitute such as GANs. Some algorithms are designed to protect privacy from human observers, while others target only automatic Machine Learning models. Techniques for human observers ensure privacy to recognition ML systems, however, the reverse is not true. The question of "how to quantify privacy–protection performance" gives rise to guarantees. This character of biometric privacy is about the risk of inferring information of removed (or concealed) attributes. There are two types of guarantees: Empirically validate and Provable Privacy techniques. The Empirically validate approach typically involves biometric recognition experiments with pre-selected matches (or classifiers) using standard biometric datasets. "Provable Privacy" offers formal (quantifiable) privacy guarantees. Initial attempts used the concept of ε– differential privacy. Also, most of the current work centers around k-anonymity.

In the context of biometric recognition systems Figure 2.12, privacy enhancement techniques can be applied at either the image level, representation level, or at inference level.

All techniques related to the image level deal with images or videos. At the representation and inference level, the data is not in the format that can be recognized by humans. For example, fingerprint acquisition, extraction of their features, and matching them with

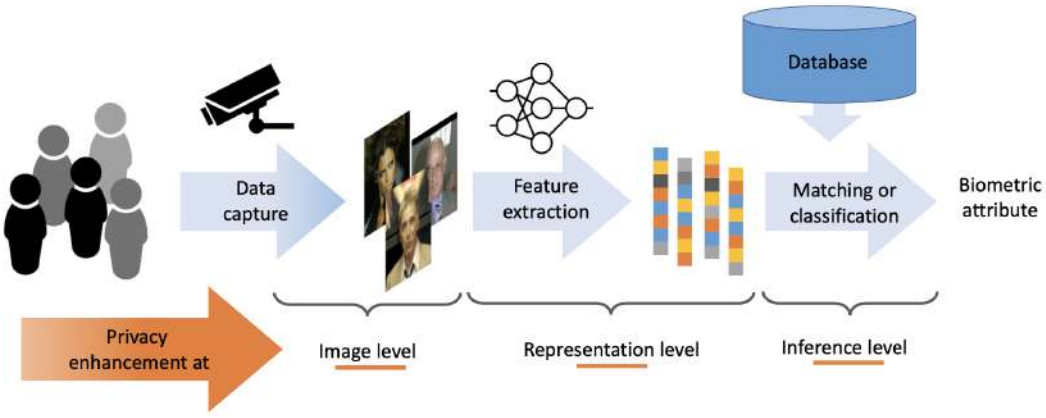

Figure 2.12.: Possible application points of biometric privacy enhancing techniques [44]

templates belong to the representation and inference level. Since our project was only concerned with improving privacy at the image level, we did not elaborate further on the other two methods. Please refer to the article by Meden *et al.* [44].

Algorithmic taxonomy of privacy enhancement at Image Level is shown in Figure 2.13. There are three main groups: obfuscation, adversarial and synthesis.

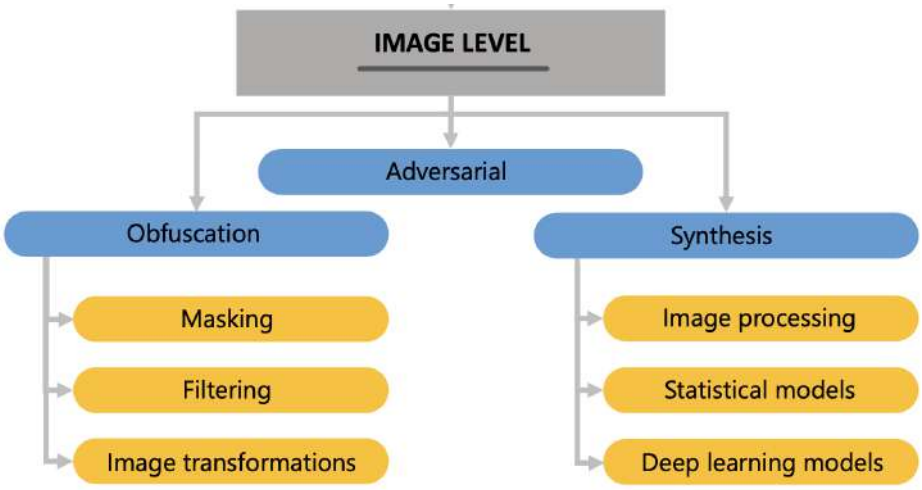

Figure 2.13.: Algorithmic taxonomy of image level [44]

Obfuscation techniques alter biometric data in ways that allow individuals to bypass biometric recognition. They degrade the quality of the original images making facial recognition classifiers infeasible. Two of their characteristics are: they are computationally simple and do not explicitly try to preserve any specific biometric attribute. Obfuscation techniques fall

into three categories: Masking, Filtering, and Image transformations. Masking techniques reduce utility by masking the region of interest (RoI) in facial images. They use masks or other abstract shapes with the goal of hiding identity information Figure 2.14 a. Filtering techniques apply linear or non–linear filtering operations to RoIs. Typical filter examples are blurring, averaging filters, and gradient operators Figure2.14 b. Image transformations conceal sensitive regions with scrambling, mosaicing, warping, and morphing to name a few methods Figure 2.14 c.

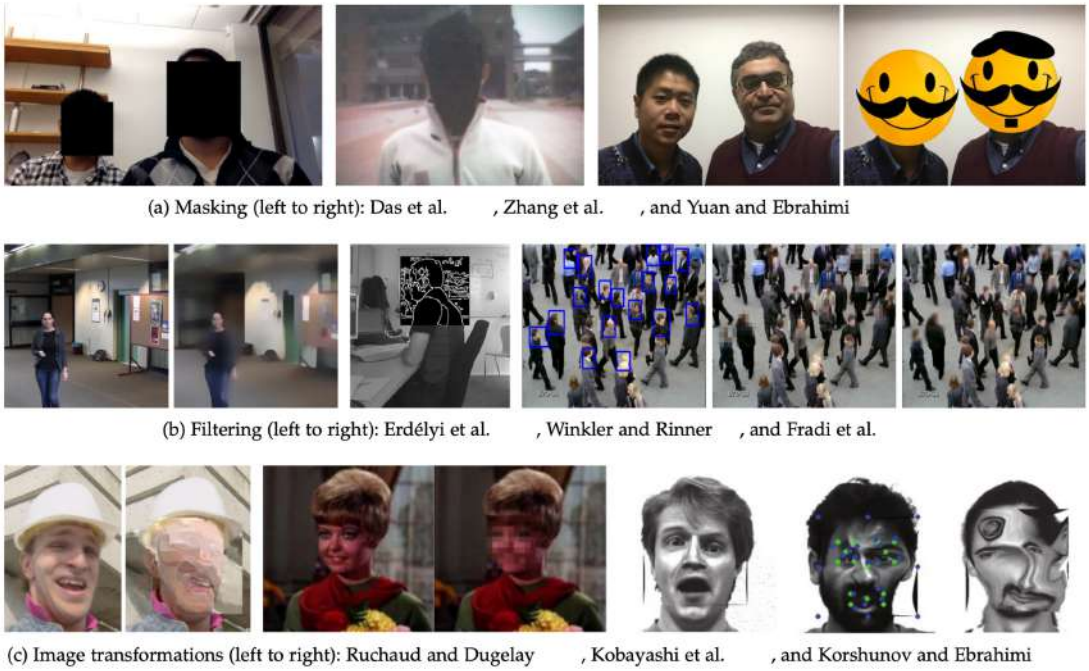

Figure 2.14.: Obfuscation [44]

Adversarial techniques are Machine Learning techniques to alter facial data. They are often designed to target specific attributes for suppression while preserving others. Their drawback is computational complexity. This method is applied for soft–biometrics such as gender, but identity is preserved Figure 2.15.

Synthesis approaches generate synthetic facial data for the original facial images. This category also comes with three sub-categories: Image processing, Statistical models, and Deep Learning approaches.

Image processing approaches: One of the main approaches for this method is the k-anonymity model which generates surrogate faces for the de-identification of a subject-

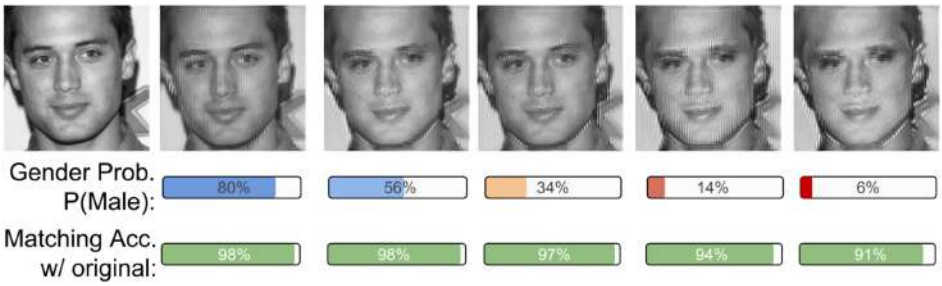

Figure 2.15.: Adversarial [44]

specific closed set of face images by averaging groups of k faces that are found to be similar. The faces in each given cluster are replaced with the same surrogate image Figure 2.16 b.

Statistical models: These are privacy enhancement techniques using generative statistical models, such as Principal Component Analysis (PCA) or Active Appearance Models (AAMs). Clusters of k subjects are hence defined in the eigenspace of the training data and the surrogate images for de-identification are generated from the averages of the PCA coefficients corresponding to all k faces in each given cluster. This technique is also refereed to as the "many–to–one scheme" Figure 2.16 c.

Deep Learning approaches: Recent approaches are heavily influenced by advances in Deep Learning and typically rely on adversarial techniques or generative Deep models. Deep Learning models are more complex than standard statistical models described above and, as a result, are able to synthesize visually convincing and photo-realistic surrogate faces Figure 2.16 d–f.

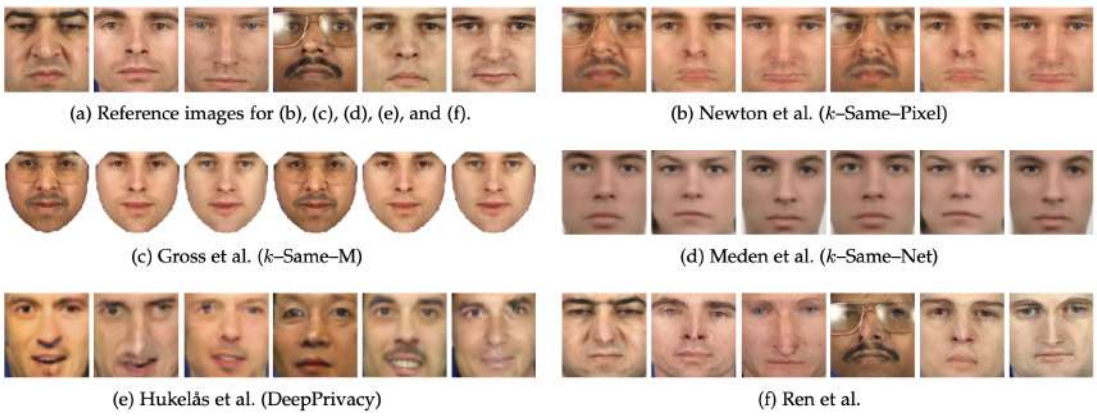

Figure 2.16.: Synthesis [44]

Iris and fingerprint anonymization were also been considered. For both, naive methods such as blurring and pixelation could work [43]. For fingerprints, filtering the image in the signal domain is using the Fourier transform for anonymization [43]. For iris de-identification, the iris could be replaced by a synthetic pattern using a generative adversarial network (GANs) [45].

2.5.1. Removing Tattoos in Images

Tattoos can be unique and/or located on a specific part of the body and could be used to identify the individual. Detecting tattoos using Deep Learning is necessary because hand-crafting features to detect tattoos is difficult, not to mention tattoos can be anything, such as a tree, a face, or a phrase [46]. Hrkać *et al.* [46] proposed an algorithm that can use a deep convolutional neural network to detect and de-identify tattoos. The algorithm consisted of two parts; tattoo detection and tattoo de-identification. The former part is a crucial part that identifies the information that should be de-identified. Detecting tattoos works as follows:

1. localization of the tattoo in the image was done by applying the trained neural network in a sliding window manner. Using a stride of k helps increase the speed and the authors used $k = 8$. The surrounding pixels of the patch center get labeled as tattooed or non-tattooed pixels, depending on whether the patch center belonged to either area.

2. Morphological opening helps to eliminate false positives (noise), while morphological closing is used to fill the gaps within the detected tattooed area. As shown on the left of Figure 2.17.

As can be seen in Figure 2.18, the tattoo was removed in most places, but some adjacent parts of the tattoo were not detected. The problem of false negatives and false positives could be solved by off-the-shelf tricks in DL, such as increasing the batch size or increasing the size of the training set by data augmentation [46]. The majority of false positives were within the surroundings of the person in 2.19. To reduce the false positives, the algorithm can be run on the output of the person-detector[46].

The tattoo de-identification is performed by replacing the color of the pixels belonging to the tattooed area with the color of the surrounding skin. This is not a trivial task as several challenges need to be addressed. For instance, the surrounding skin regions have different colors or shades due to different lighting conditions, or due to a skin condition. Therefore, a natural-looking image requires replacing the color of each tattoo-pixel with a new color

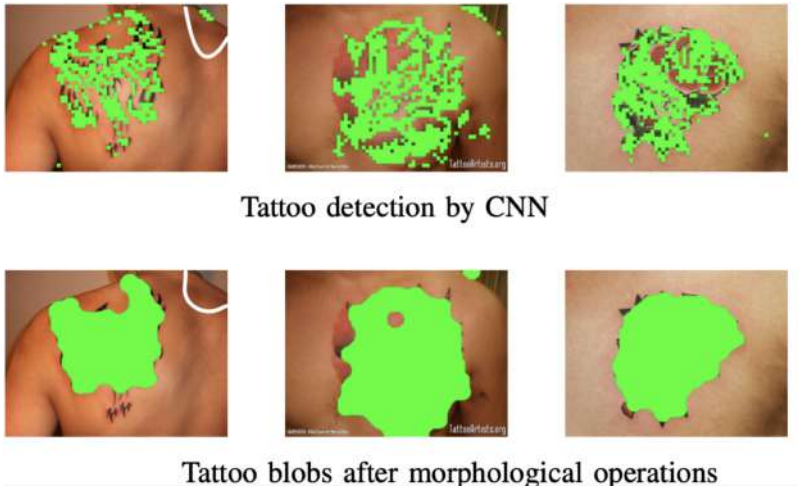

Figure 2.17.: Results of using morphological closing and opening [46]

Figure 2.18.: Results of the tattoo removal algorithm [46]

calculated by interpolating the pixel value on all the surrounding skin pixels. Therefore, morphological dilation is used to find the contour of the tattooed area.

Figure 2.19.: The false positives problem [46]

2.6. Re-identify Anonymized People in Images & Prevent Re-identification

We found that it is difficult to prevent re-identification, at least if the image should still be visually appealing. For AI methods that anonymize images, there are often adverse AI methods theoretically or practically capable of reversing the process. Simpler methods, like blurring or pixelation, are required to be performed with a lot of strength, which turns the image to be not visually appealing. Even with the most restrictive method, Such as masking the object of interest, one has to be careful. As image compression can lead to artifacts around the object. These artifacts could be used by an attacker to identify the person in the black area. To prevent this, any method should always be generous in drawing a large box around the object of interest. However, we have implemented masking and blurring as an anonymization technique in our pipeline.

3. Strategy

A product strategy should provide a clear roadmap, and define and prioritize actions. However, a strategy is just one element out of the entire strategic direction that project management must define and elaborate on by setting a mission and goals first. Keeping that in mind, in this section, we will define the concepts of *mission*, *value network*, and *vision* (and what it entails for our project) to support a better understanding of *strategy*.

Vision, answering the question of *why*, is developed to serve as a guiding beacon for everyone in the organization. It motivates the decision-making process and determines the organization's intended direction, aspiration, or outlook for the future. Mission answers the question of *what* is to be achieved and what difference the product will make 1. Importantly the mission is not a strategy but a definition of purpose and goals. Strategy, in turn, answers the question of *how* resources are distributed to meet the project mission requirements based on the value network's needs. Lastly, the *who*, or the value network, holds the relation among all project stakeholders. Further on in this section, we answer the *how* behind our project progression by defining the project framework, sprints, team members' roles, and collaboration tools.

3.1. Framework

Project management is applying knowledge, tools, and skills to execute a project under specific requirements. It consists of identifying the problem, creating a plan to solve the problem, and then executing the plan until the problem is solved [47]. Frederick Taylor introduced the concepts of project management into work day to improve inefficiencies and work smarter instead of harder and longer. Later, Henry Gantt used these concepts and created bars and charts to visualize the completed tasks [47]. The Agile project management methodologies were codified in the Agile Manifesto[1] in 2001. The four central values of Agile methodology:

[1] `https://agilemanifesto.org/`

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Agile refers to these values in addition to the frameworks that implement them, such as Scrum and Kanban. These frameworks have similarities: they require the team members to work in iterative cycles, where the team evaluates the results at the end. According to the evaluations, the results are adjusted to meet the customer's needs [48].

3.1.1. Scrum

Scrum is the most popular Agile development software due to its simplicity and ability to solve many software development problems, such as complicated development cycles, shifting production schedules, and inflexible project plans [48]. Scrum is implemented for a small team with a Scrum Master and a Product Owner, whose main task is to resolve obstacles and ambiguities to improve efficiency. The team meets daily to discuss the work and obstructions the team is facing [48]. The team works in two-week cycles, i.e., sprints. Scrum has several advantages and disadvantages. One advantage is that it helps to manage large projects and improves communication within the team due to the daily standup meetings [49]. But Scrum is time-consuming since it requires daily meetings. Additionally, it is also inflexible to add changes as it treats the planned tasks as sacred. However, in reality, serious errors or changes in clients' demands might emerge, which could ruin the concept of the sprint [49].

3.1.2. Kanban

Kanban is another Agile framework based on the team's capacity developed by Taiichi Ohno for Toyota in the 1940s [48, 49]. "Kanban" means "billboard" in Japanese because it visualizes what should be done, when, and by whom [49]. For process visualization, Kanban uses cards and columns, where columns represent a step in the workflow and cards represent the tasks, as shown in Figure 3.1. The visualization of the workflow helps team members quickly understand the status of the project [49]. Kanban is event-driven instead of strict deadlines. Therefore, it is flexible to sudden changes in demand. This is done by

simply moving tasks to different columns [49]. But this has a drawback if a card is not moved to the right column and other cards depended on it, these cards will be blocked [49].

Figure 3.1.: The workflow visualization using cards and columns. [49]

3.1.3. Scrumban

Scrumban is another Agile management methodology, and as the name suggests, it is a combination of Scrum and Kanban. Scrumban uses flow-based methods of Scrum together with Kanban's visualization [49]. Teamwork is organized in small iterations, where on-demand meetings are held to discuss the tasks of the next iteration. Scrumban does not require team roles or a specific number of team members. However, Scrum roles can be kept if the team needs a Product Owner or a Scrum master [49]. The team members assign themselves to the tasks from the Scrumban board. Due to this flexibility in setting up the principles, the team might end up following their own framework. In addition, it is difficult to track each member's contribution and effort because Scrumban does not require daily stand-ups, which shows what the team members are currently working on [49].

3.1.4. The Framework of the Study Project

We faced some troubles when choosing the most suitable framework for our project. LMIS suggested using Scrum. Thus, after the first workshop regarding Agile development, we familiarized ourselves with the Scrum framework. However, our group's size was not suitable for Scrum since Scrum is designed for small teams and is sprint-based, which makes it not the best framework for us due to the possible extension of the sprints. Therefore, LMIS invited us to another workshop to discuss another possible framework, namely, Scrumban. We continued using Scrumban with semiweekly standups, where three people presented the

current state of their work to keep track of the contribution of each team member. We also held meetings for workshops and coding support sessions. Kanban was not suitable for our group because Product Owners are needed to organize the workflow. We used Kanban's visualization tool on DevOps to get a quick overview of the project's status.

3.2. Roadmap & Sprints

To better understand the purpose of sprints, we will first briefly introduce the definitions of sprint planning and sprint artifacts. Sprint planning is a Scrum event that kicks off the sprint. Its purpose is to define deliverable goals and a strategy to achieve them. The sprint planning event is a collaboration between the entire Scrum team. The Scrum artifacts, namely the product backlog, sprint backlog, and product increment, are information that provides details on the product development and the actions required to perform this development [50]. In our project, Product Owners were responsible for defining the goals and communicating which backlog items are included in the upcoming sprint. Developers, in turn, planned their work to produce an increment that was sufficient to complete a task or a goal. More details on team members' roles and responsibilities are available in Section 3.3.

Our project was divided into phases that mirrored the progress of our product development. Together the phases combine into our project roadmap. For instance, during Phase I, the participants were introduced to team-building events. Phase II was dedicated to assigning team roles and introducing the first product features that needed to be implemented. In Phase III, the product development started. The goal of Phase IV was to evaluate the current state of the project and re-structure the components that require better performance. Phase V was the final stage of the project dedicated to wrapping up work and concluding the results.

3.2.1. Phase I: Introduction to the Project Idea & Team-Building Events

In Phase I (10.2021-11.2021), as mentioned earlier, we focused on clarifying our mission, goals, and vision. During this time, we also conducted team-building events where we defined team core values, skills, and learning aspirations. In terms of core values, there are five Scrum values: commitment, courage, focus, openness, and respect. In our project, we decided to add more values such as *fun*, *clarity*, *teamwork* and *knowledge sharing* to better reflect the project's needs. When it came to skills, everyone needed to have at least basic

knowledge of Python, git commands, and deep learning frameworks such as PyTorch or TensorFlow. As per learning aspirations, participants looked to improve their programming skills, learn more about computer vision and Deep Learning techniques, as well as their skills in project management. In the end, everyone tried to follow the entire set of rules on project communication, roles, and organization with a grain of salt. If the work was too challenging or a deadline was missed, the team didn't penalize the members but rather helped each other to increase the individuals' feeling of belonging to the study project group, which eventually led to enjoying the work to a greater extent.

3.2.2. Phase II: Setting Up the Product's Features

In Phase II (12.2021-01.2022), the Literature Review sprint aimed at getting familiar with computer vision basics and state-of-the-art algorithms and models to detect and remove any personal information from the images. The meetings with Dr. Jan Hendrik Schoenke (Artificial Intelligence Business Development Manager at LMIS) and Ann-Christin Meisner (Software Developer for Machine & Deep Learning at LMIS) helped us to extract a rough idea about the desired product LMIS wanted. Dr. Schoenke described a semantic mapping within the network of individual image features that differentiates between GDPR relevant and irrelevant features. Thus, it gave rise to different approaches to implementing an anonymization application that allowed us to filter our literature research phase into a small set of features, which is essential to our product. The fundamental topics covered:

1. Exploration of GDPR in the context of image data anonymization and the legal issues companies face when not following the GDPR.
2. Comprehensive understanding of the current market situation on data anonymization tools, with their advantages and disadvantages.
3. Availability of the datasets, their authorization to be used commercially and the possibility to create our own dataset.
4. Semantic mapping of features within an image, in the sense of an end-to-end learning approach (one model to extract all features) or an individual per-feature approach.

Individuals were assigned to one of the four topics outlined before, to explore the area for a broad understanding of individual problems our project within these fundamental topics might face. To address these tasks within each group, people were encouraged to have vital discussions about their topic, do research on different sub-topics, and eventually pro-

vide the gathered information to the entire group in the form of one big or several small presentations.

The Literature Discussion sprint aimed at concluding and discussing the findings of the previous sprint. We wanted to find the best solution to develop a product idea proposed by LMIS (Section 1). Phase II also consisted of several workshops and presentations to get everyone onto the same level of knowledge regarding topics like project organization, software engineering, and research concerning individual features.

3.2.3. Phase III: Launch of the Minimum Viable Product Development

In Phase III (02.2022-05.2022), after the entire team gathered enough information on the current market and research situation for GDPR conform products and features, the product owners were able to create the fundamental concept for our Minimum Viable Product (MVP). Human detection was the core feature for our product to be implemented. Thus, two possible solutions were considered; to detect the entire body or to solely detect faces. Depending on the use case, the former is used to render any biometrical information and personal identifiers like styles or skin color. Face detection is considered an alternative that is a less radical anonymization process than the former. The same applies to vehicles, where either the entire vehicle should be blurred or solely the license plate. The last category for GDPR-relevant personal information is text and logos. Here we choose text detection to de-identify confidential documents and text-based logos. On the anonymization of the MVP, we selected blurring and masking.

For the feature implementation process to start, the product owners outlined several small milestones for the developers as a software engineering guideline and provided a list of accomplishments developers had to deliver to complete the design of a feature. During this task, developers compared different models, revealed characteristics of the individual networks, and highlighted their advantages and disadvantages.

At first, a feature branch had to be created to track the feature implementation and document solutions to obstacles that might emerge in other features. The model class follows an instruction set, which the pipeline maintainer implemented into the underlying framework. The model class includes a method to initialize the model from a saved instance, a pre-processing method for the model to operate flawlessly, and a call method, which calls the instantiated model to provide detected landmarks within an image.

Furthermore, developers had to provide both the information that highlights the installation of the requirements and an exemplary code that clarifies the class usage. The last step was to create a pull request to display that the feature is ready to be reviewed by another developer, whose task is to review the code's quality and provide constructive feedback.

3.2.4. Phase IV: Code Integration and MVP Upgrades

In Phase IV (06.2022-08.2022), we had Code Wrap-Up, Code Integration and MVP Upgrades. We faced some technical problems, such as dependency issues and the features' code wasn't running on all the machines. Thus, we needed to use Docker containers, which led to an extension of the deadline since we didn't have any experience with Docker and needed to learn about containerization. In general, all developers that worked on model-class implementation struggled to understand and trace the code of the original model packages. Pipeline and model developers had problems understanding the rough outline of our software development framework, and pipeline developers were not sure how to implement the back-end. The project coordination and planning within our framework of the semi-weekly meetings also required continuously getting in touch with teammates to acquire further knowledge on the individual feature progress and provide an adequate solution for problems.

To upgrade our MVP to a Full-Scale Product (FSP), we added a new de-identification technique, Deep Anonymizations, which replaces the person's face with another face to create a fake representation of the actual data. For our anonymization application, we don't intend to create fake images but anonymized ones that don't refer to any person. For the FSP, we implemented two additional features; tattoo detection and screen detection.

3.2.5. Phase V: Codebase Clean-up, Documentation and Presentation

Phase V (08.2022-09.2022), in the last six weeks, the project was defined by a phase of fast adaption to emerging problems with solutions that quickly resolved them. Thus, this demanded everyone to focus not only on a single task but on several tasks simultaneously, including software programming, documentation writing, and project's presentation. The overall objective of this phase was to finalize our study project by combining all the individual software, documentation, and presentation parts into one concise and presentable product.

After individual feature packages were successfully built and extensively tested, several new problems for the packages emerged. One imminent trait of the individual-created packages was the inconsistent code format and documentation. Thus, another role was allocated to supervise the code and docstring formatting. The tasks of this role were to create uniform Doxygen documentation, highlight mistakes in docstrings and create consistent code formatting. Another important piece of information for each package was its license restriction, as we were working in collaboration with LMIS. LMIS can only use our product without further adaptions if all the features are commercially applicable.

The project report is summarized to create an overall structure for individual developers and product owners to fit their contributed work. Thus, the report was proofread by a group of five members to eliminate repetitions and form the final report into a uniform document.

During the last week of our project, everyone focused on their respective presentation parts. We provided feedback on the individual presentations and conceptually put them into one framework. On the 30th of September, the final product was presented to the Cognitive Science students of the University of Osnabrück and to the stakeholder and employees of LMIS.

3.3. Roles & Responsibilities

At the start of our project, we observed that participants come from different backgrounds and therefore showed versatile skill sets. The participants formed groups based on the following roles and responsibilities: Product Owners (POs), Developers, and Documentation Team. Unlike the typical team structure in Scrum, we assigned four people as POs. POs focused on ensuring that the tasks stay aligned with the product goals. The responsibilities included planning sprints and tasks, developing the MVP, and prioritizing task items based on the strategy. Developers focused on completing the tasks distributed by the POs and ensuring they provided increments to complete sprint goals. The documentation team were responsible for organizing the report structure and distributing sections for writing to other project participants.

3.4. Collaboration and Communication Channels

For communication and collaboration, we used different tools; some of these tools were provided for free or were provided to us by LMIS. The used tools were the following:

- **DevOps Azure**[2]:
  LMIS created a DevOps account for each team member to use the Kanban board and work together on code by creating a repository, the same way GitHub works. DevOps also permits creating a pipeline, which is needed to connect the different containers we built for each feature.

- **Miro**[3]:
  LMIS suggested using Miro for brainstorming when working within the team. Miro provided us with nice visualizations to present our product and manage the workflow.

- **Slack**[4]:
  Slack was our communication tool, where we created a channel for each group and general channels for all team members to notify each other about important announcements. We were also able to communicate with each other directly, which is needed when sending sensitive information, such as credentials for DevOps.

- **myShare**[5]:
  It is a cloud service to share data securely. We were able to use it through login with our university mail account.

[2] https://azure.microsoft.com/en-us/products/devops/
[3] https://miro.com/
[4] https://www.google.com/search?q=slack&oq=slack&aqs=chrome..69i57j0i67l2j69i60l5.1867j0j4&sourceid=chrome&ie=UTF-8
[5] https://myshare.uos.de/accounts/login/?next=/

4. Implementation

The following section gets more technical as it introduces the software architecture and its implementation within our project. It dives deeper into the research behind different detection models and anonymization methods. The research sections (4.2 - 4.3) describe features included in the basic and extended MVP.

4.1. Code Framework

To realize our project and simplify communication between members and implemented models, we chose one programming language – *Python* [51] (Version 3.9). *Python* is an open-source, well-known to every member of our study project, and one of the main languages in data analysis and machine learning research. It contains many libraries that make it easy to work with images, build a website or create an API. Furthermore, many of the models we looked at were already implemented and working in *Python*.

For our product, we created an easily extendable architecture, which we explain in the following sub-chapters. To access and use our product, we created an API endpoint that could be accessible by front-end or regular HTTP requests. Due to each model running in its own Docker container 4.1.4, we also created a separate API endpoint for each of that models. The API calls the pipeline that manages the data flow and communicates with different detection models and anonymization methods. All detection models follow a generalized implementation to simplify their use by the pipeline and enable a coherent output of each model. A similar approach applies to different anonymization methods. All structures are explained in the following sub-chapters.

4.1.1. API

From the beginning, we knew that our product would be running on a server and, therefore, needed to be accessible from the outside, e.g., by a website or mobile app. For this reason,

we started building an API endpoint from the start. *Django*, together with *Flask*, are the two big Python web frameworks allowing to create front-ends, back-ends, and APIs. At the start of the project, we were only familiar with *Django*, so we started building a front-end together with an API endpoint in *Django*. Building a front-end during the project was not one of our stated goals. However, a very simple front-end does not take a lot of time to develop and, in the end, it may save time to have an easier way to debug the whole system. The basic front-end we implemented gives the user an easy way to upload an image and select what detection models and anonymization methods to use in a form. Then the uploaded image, together with the options, gets sent to the API endpoint and then is handed off to the pipeline. The pipeline processes the image and eventually returns an anonymized image displayed alongside the original image on the front-end. Using the front-end this way goes against our stated goals of creating a GDPR-compliant product because we are saving both the original image and the anonymized image to a database. However, it was vital as a debugging use case, and is not needed for the rest to work and should not be included in the final product.

Later during the project, we learned that for the *Dynamian* product, LMIS is using *Flask*, so we changed the main API endpoints for the pipeline from *Django* to *Flask*. This process did not take much time because we do not have a big and complicated API with many different endpoints. For the main API there are only two endpoints: */shield/service/pipeline* and */shield/pipeline/*. The first one only returns whether or not the pipeline service is healthy, while the second one accepts an image together with the selected anonymization options and returns the *base64* encoded, anonymized image. With this version of the pipeline, neither the original image nor the anonymized image are saved to disk on the server and only exist in memory. This is much safer and also unavoidable because the image still needs to be used.

The work on the API code was very helpful when we later decided to run each detection model in its own *Docker* container. The pipeline needed a way to communicate with each Docker container, so we reused the code for the main API in each Docker container. Every model also has two API endpoints: */shield/service/<detection_model>* and */shield/model/<detection_model>*. With six models and the pipeline, we therefore have a total of 14 API endpoints.

4.1.2. Pipeline

The pipeline's primary purpose is to manage the data flow between the API, detection models and anonymization methods. In figure 4.1 you can see an activity diagram that shows this workflow. It starts by getting an image that is supposed to be processed together with the options of which models to use and the corresponding anonymization method. First, the pipeline validates the options to ensure that all models and anonymizations picked by the front-end are available, reachable and initialized. Therefore, the pipeline reaches out to every model that was chosen and does a simple health check with a GET request. If that is the case it continues with the verification of all the chosen anonymization by checking, if they are initialized.

After the validation, the pipeline sends an asynchronous POST request to each selected model containing the image to be processed. We decided on doing this asynchronously because some models have significant runtimes and, therefore, we get the ability to do parallel processing. Every model returns just a dictionary with bounding boxes and/or bit masks containing the coordinates of the detected objects. The pipeline waits until it gets all responses and gathers them into one dictionary containing the model's name and the corresponding resulting bounding boxes and bit masks, even if the model returns no results.

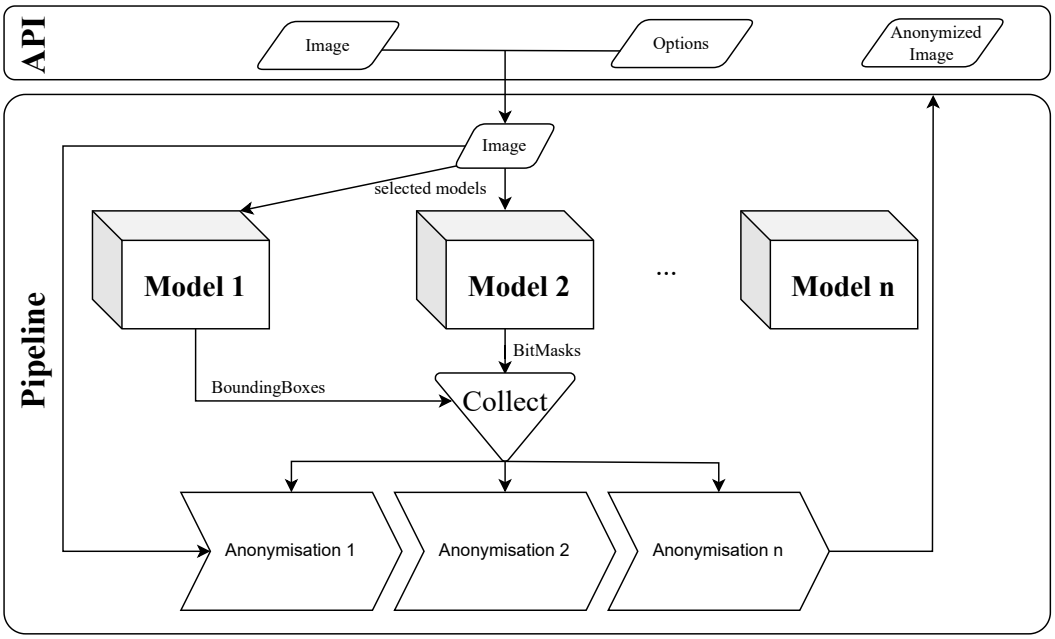

Figure 4.1.: Activity diagram of the pipeline

The pipeline then uses this aggregated result to call the chosen anonymization for every model. It does not apply each anonymization to a copy of an original image nor tries to merge those copies afterwards. Instead, it passes the original image from one anonymization to another. After all the anonymizations have taken place, the pipeline encodes the image to a *base64* string and returns it to the API, which then responds with the string to the client who initiated the POST request.

4.1.3. Feature Integration

As features, we defined two significant categories. On the one hand, there are detection models. This is the part of the pipeline where the image is classified into areas or objects that should get anonymized. On the other hand, there are methods that enable exactly this anonymization. The detection models and anonymization methods integrated during the project are listed in table 4.1.

Table 4.1.: Integrated detection models and their anonymization methods

Detection Model	Blurring	Masking	DeepFace	TattooRemoval
FaceDetection	X	X	X	X
LicensePlateDetection	X	X		
PersonDetection	X	X		X
ScreenDetection	X	X		
TextDetection	X	X		
VehicleDetection	X	X		

Detection Models. The major problem we faced when examining the code base of various detection models was that they often needed different dependencies, e.g., different versions of PyTorch, that are often not compatible. Therefore, we had to find a solution where each model could exist in its own environment. One idea would have been to always change the environment, depending on which model is currently running. This approach would have resulted in a long list of different commands, running one after another and many possible places for bugs, errors, typos, etc. It also would have been much slower because no parallel processing would have been possible. Therefore, we chose to build every model in its own Docker container, with its own environment and dependencies. The pipeline then connects to every container separately. This solves the dependency problem and enables a much more scalable approach, where integrating more models does not affect the pipeline. A new model has to be added to one file with its name and a corresponding URL link so that the pipeline knows where to send the post request with the image.

Another problem regarding a coherent structure was the different detection models. Every model had different ways of loading its specific weights, pre-processing the image, organizing the results, and how the image is fed into the system in the first place. To circumvent these differences, we decided on a class-based approach with one parent *Model* class. This class-based structure is shown in detail in figure 4.2. The parent class manages the overall processing order. First, load the model, then use the pre-processing for the image, send the image into the model and finally process the results. The actual detection model, e.g., *FaceDetection*, inherits from this parent class and does not change the order the sub-processes are run in. It just has to make sure that it implements the *load_model*, *preprocessing* and *process_labels* functions based on how the specific models work. With this approach, we can ensure that all models use the same entry and exit point. Therefore, the pipeline does not have to send a specific request to specific models but uses one consistent workflow.

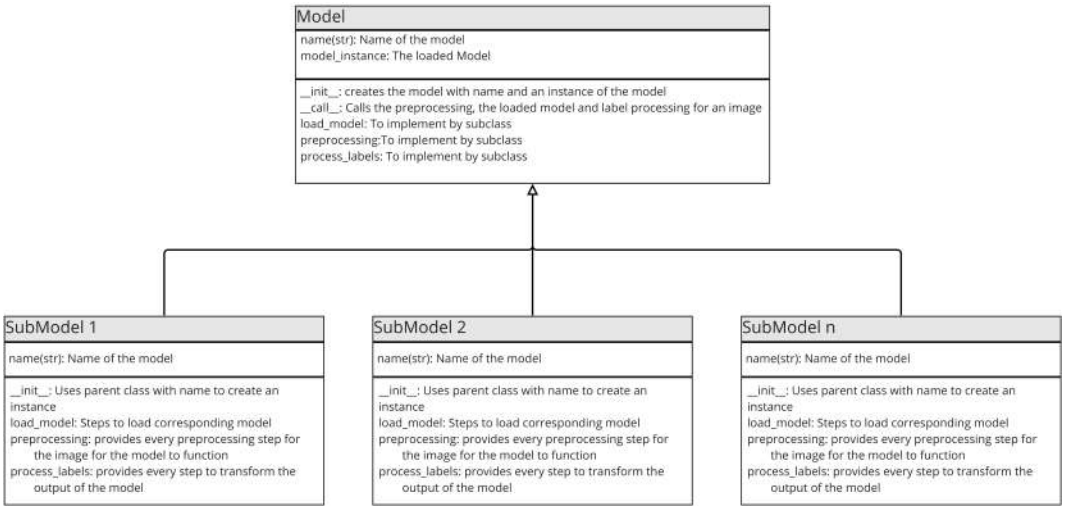

Figure 4.2.: Class diagram for the model parent class

The last problem we faced was the computational power needed to run all models simultaneously. Because they are all trained networks, the processing needs a lot of power. When all run on one machine, the resources needed quite large, and even if provided it can happen that some models crash. That is again another reason to use a dockerization approach, because every model could run on its own machine or server and the pipeline communicates with them over the API entry points.[1]

[1]Final test was done on a laptop with an i9-12900HK, 64GB RAM and an RTX 3050Ti Mobile

Anonymization Methods. The different types of anonymizations were our second feature focus. Each model can detect multiple objects to anonymize and doesn't require too much computational power. It is due to having computationally low-cost tasks, unlike what we had in the detection models case. Therefore, we decided to proceed without a parallel approach.

We tried to optimize and minimize the runtime each anonymization method requires. Here the python protocol feature comes in handy. *Python* is a dynamic, script-based programming language that checks the type of every object during its runtime, unlike other compiler-based languages like *Java* during the compilation. The protocol feature can bypass this behavior by using *duck typing* [55]. Therefore, the python interpreter can handle the elements fed into an anonymizer much faster because it doesn't have to check every incoming element and hence needs less computational time.

As for our two main anonymization methods, we implemented masking and blurring. Blurring uses a kernel to iterate over an area and successively blurs every pixel of this area. Masking, however, paints over the area with black color. It can handle bounding boxes, as well as bit masks as targets. These two anonymizations can be used for every detection method.

After that, we also integrated two different, more specific anonymizations, namely tattoo and face anonymization. The tattoo anonymization takes the output of the person or face detection model to determine the region of interest and then removes and, therefore, anonymizes all tattoos in this area. It uses the bounding boxes of these detection models and the same protocol structure as blurring and masking with bounding boxes. The face anonymization uses Deep Privacy [56] to de-identify human faces. These two specific anonymization methods allow the user to anonymize a whole person without giving away important key points. All integrated anonymization methods are also listed in table 4.1.

4.1.4. Docker

Docker is a containerization technique, which is an isolation technique that tricks specific kernel processes into thinking they are the only running ones in the machine [57]. A docker was a laborer to move the goods into and out of ships when docked at ports. On these ships, there are goods of different sizes and shapes. Thus, previously more dockers were needed to move the goods. There was no other alternative, however this is a familiar problem to software problems [58]. An example from Miell & Sayers [58] is shown in (Figure 4.3); before Docker deploying software to different machines required significant configuration effort, using resource-hungry configuration management tools such as Virtual Machines to provision software on the different machines [58]. With Docker, the deployment effort has reduced drastically because the configuration effort is separated from the resource management [58]. In this study project, we faced a couple of challenges, such as using different machines and toolkits that do not support all machines, for example, CUDA does not support development on macOS. Another problem is that several features require different Python versions. Hence, for each feature we created a separate image to consume fewer resources and this way, the features would not interfere with each other [58].

Docker & Linux Kernel Features. Disallowing processes to communicate with other processes helps in solving several problems. For instance, a process can immediately terminate another process or the process monopolies all the memory/CPU usage and does not allow other processes to make use of the resources. Thus, container technologies like Docker make use of two Linux kernel features to implement containerization [59]:

- **Namespace**: tells a process with what other processes it is allowed to interact.
- **Control Groups**: limits the process' usage of resources, such as memory, CPU, and I/O usage used by a process. This way, all the resources are distributed equally.

Containerization vs. Virtualization. Virtualization is a process that virtualizes a system's singular resources such as RAM, CPU, or disk to be represented as multiple resources. It does so by creating an abstraction layer over the computer hardware. This lightweight software layer is called a hypervisor [60]. A Virtual Machine (VM) is an emulation of a physical computer. VMs enable teams to run what appear to be multiple machines with multiple operating systems on a single computer. The hypervisor virtualizes physical hardware, where each VM contains a guest OS. This guest OS is a virtual copy of the hardware that the OS requires to run an application and its associated libraries and dependencies [61]. VMs with different operating systems can run on the same physical server. For example, a VMware VM can run next to a Linux VM, which runs next to a Microsoft VM, etc.

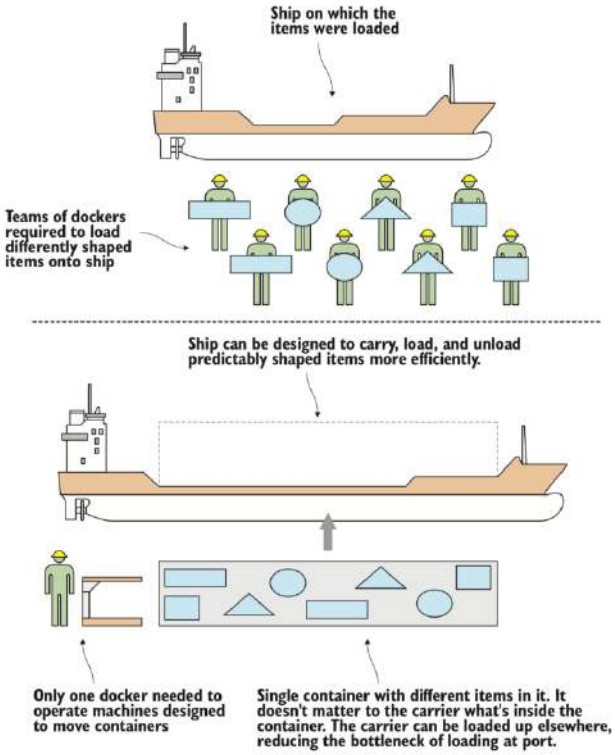

Figure 4.3.: Shipping before and after standardized containers [58]

While containers do not use a hypervisor, we can control the resources better without a copy of the OS, as shown in Figure 4.4. Since containers use a form of OS visualisation using Linux features (namespaces and control groups) to isolate processes and control the processes' access to CPUs, memory and desk space. So each container contains only the application, its libraries and dependencies, which enables applications to run almost anywhere - a desktop computer, a traditional IT infrastructure or the cloud [61]. Therefore, containers are small, fast, and portable, unlike virtual machines.

Container vs. Image. A Docker container can be seen as a Docker image in execution, the same way a process can be viewed as an application in execution [58]. Another analogy in terms of object-oriented principles, an object is a concrete instance of a class, and a container is also a concrete instance of an image. We can have multiple images from a single image, and they are all isolated from each other [58].

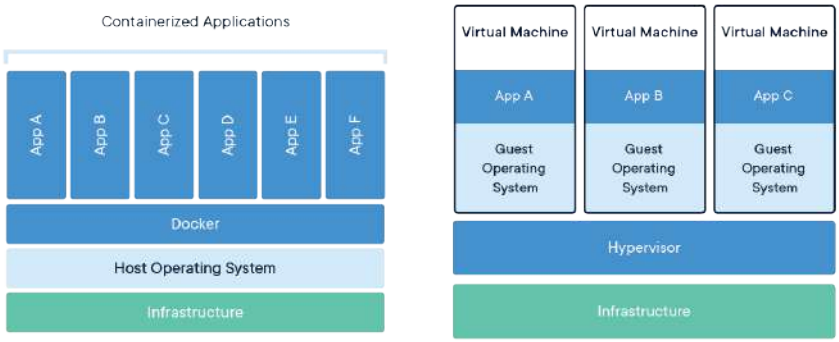

Figure 4.4.: Comparison between a virtual machine and a container [61]

4.2. Basic MVP

The following section describes our research on the features required to develop an MVP. Those features include license, vehicle, text, face, and person detection methods. The basic MVP concludes our initial prototype for providing a solution to de-identifying personal information in images.

4.2.1. License Plate & Vehicle Detection

On the Papers With Code[13] website, we tried to find models for license plate detection, but only models for license plate recognition were available. We took into consideration several conditions for the license plate recognition model. First, it needed to detect European license plates since the SHIELD product is developed for the EU. Second, the European license plate datasets had to be available. Additionally, a publicly available code explanation needed to be there together with a pre-trained model to download.

This research pointed out that most models used Chinese license plate datasets. Moreover, most papers did not have a GitHub repository or any available pre-trained models to download. Nevertheless, a YOLO object detection model was the most used model. In conclusion, "An efficient and layout-independent automatic license plate recognition system based on the YOLO detector" model resulted in having the best performance on the most of datasets, in particular in the European dataset, which was the one we were interested in. Also, [62] provided their pre-trained models[2].

[2] `https://web.inf.ufpr.br/vri/wp-content/uploads/sites/7/2021/05/laroca2021efficient-published.pdf`

Vehicle detection and license plate detection models. "An efficient and layout-independent automatic license plate recognition system based on the YOLO detector" created a system based on YOLO object detector. The main idea was to use a unified approach for license plate detection and layout classification by employing post-processing rules to enhance the recognition results. In [62], three models were employed for the recognition of the license plate: the vehicles detection model, the license plate detection and layout classification model, and the license plate recognition model.

In the first step, all vehicles are detected in the input image. Then, the license plate of each vehicle is detected, and its layout is classified. For example, in Figure 4.5, the vehicles/license plates are from the Taiwan region. Finally, all characters of each license plate are recognized simultaneously, with heuristic rules being applied to adapt the results according to the predicted layout class. For example, it was established the bare minimum and maximum characters that should be considered in license plates for each layout. While American, European, and Taiwanese license plates do not have a set number of characters, Brazilian and Chinese ones do.

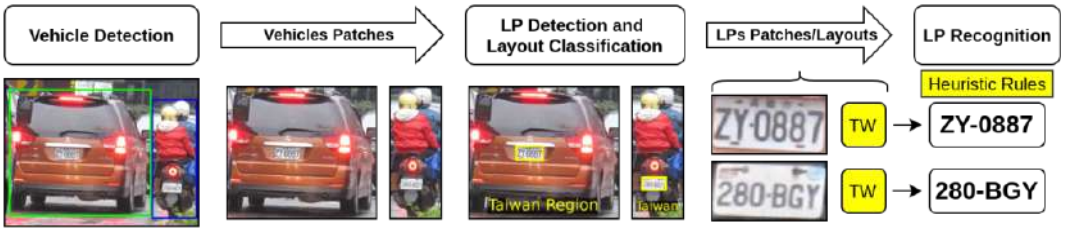

Figure 4.5.: The pipeline of the ALPR system. The pipeline of the proposed Automatic License Plate Recognition system is shown in the picture.

In this project, the license plate recognition model was not used, as the main focus was on finding a model which can detect the license plate. However, it is important to bear in mind that this model can predict different types of license plates as American, Chinese, European, Taiwanese, and Brazilian. More information about the architecture of the models and more can be found in [62].

Datasets. The datasets used for training and evaluating the model can be found in the table 4.2. These datasets are often used to evaluate Automatic License Plate Recognition systems, contain multiple license plate layouts, and were collected under different conditions/scenarios, i.e., with variations in lighting, camera position and settings, and vehicle types.

Table 4.2.: Datasets used for training the models

Dataset	Year	Images	Resolution License Plate	Layout	Link
Caltech Cars	1999	126	896 × 592	American	link
EnglishLP	2003	509	640 × 480	European	link
UCSD-Stills	2005	291	640 × 480	American	link
ChineseLP	2012	411	Various	Chinese	Zhou *et al.* [63]
AOLP	2013	2,049	Various	Taiwanese	Hsu *et al.* [64]
OpenALPR-EU	2016	108	Various	European	link
SSIG-SegPlate	2016	2, 000	1920 × 1080	Brazilian	Gonçalves *et al.* [65]
UFPR-ALPR	2018	4, 500	1920 × 1080	Brazilian	Laroca *et al.* [66]

In particular, most pictures of two datasets collected in Europe (EnglishLP and OpenALPR-EU) were obtained with a hand-held camera and there is only one vehicle (generally well-centered) in each image.

Darknet. Darknet[3] is an open-source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation. The Darknet framework was employed to train and test [62]'s networks. However, the authors used AlexeyAB's[4] version of Darknet, which has several improvements over the original, including improved neural network performance by merging two layers into one (convolutional and batch normalization), optimized memory allocation during network resizing, and much other code fixes. [62] provide only the models in their websites[5] and some code examples.

Therefore, we followed the command suggested on the authors' website, using the Darknet framework. While studying this framework, we noticed that this framework allowed us to save the bounding boxes in a text file with the same name as the image and save the results in a text file. Also, it was possible to set a threshold for confidence in the detection.

[3] `https://pjreddie.com/darknet/`
[4] `https://github.com/AlexeyAB/darknet`
[5] `https://web.inf.ufpr.br/vri/publications/layout-independent-alpr/`

Consequently, we were able to run the vehicle detection model but not the license plate detection model, because [62] did not share all the code for the correct running of their system. In fact, following the authors' pipeline, we were asked to crop the image following the vehicle patches. Starting from this, we had to create two functions for cropping the image. The first function crops each image starting from the image patches of the detected bounding box, given a list of image paths. The other, instead, crop each image given the image patches of the detected bounding box and saves this cropped image with a new path name adding the identified object in a text file.

It is noteworthy that we took advantage of the text file with the coordinates of the detected bounding boxes. After all the changes, we were able to use the license plate detection model.

After this, we created a python file to run the pipeline in one file. First, all the vehicles in an image have to be detected. Second, the image will be cropped following the bounding boxes coordinates saved in a text file. Finally, the license plate will be detected for each cropped image and the coordinates will be saved in a text file.

Wrap-up model. For this project, these models have to run in PyTorch and not in Darknet. Therefore, to wrap up the vehicle-license-plate feature, the vehicle and license plate models have to be imported into PyTorch while keeping the same functionalities that they had in the previous framework. To achieve this result, we used the PyTorch Yolo2 implementation of this repository[6] and changed some parts of the code, because the models were not detecting, but an empty list was given as a result of the vehicle detection model. To better understand what was the problem, we checked:

- whether the weights of the models were loaded correctly;
- whether the input image was correctly pre-processed;
- whether the YOLO2 models used in this repository[6] have the same layers and configuration of the ones explained in [62].

It was noticed that the vehicle and license plate detection weights are in the Darknet format *(.weight)* not in the PyTorch format *(.pth)*. We decided to not convert the file, but we tried to load the file as it was. We changed part of the code following the code found in this repository[6] and another[7]. After the two models were running, we tested them and check whether the PyTorch models can have the same functionalities they had in Darknet. We

[6] `https://github.com/ayooshkathuria/PyTorch-yolo2.git`
[7] `https://github.com/eriklindernoren/PyTorch-YOLOv3.git`

noticed that the first model perfectly detected the vehicles, but the bounding boxes of the license plates were bigger and larger than the ones in Darknet. They cover even some parts of the vehicle.

Starting from this baseline, we improved the code. For example, the use of text files to store the vehicle-license plate bounding boxes coordinates was discharged since it would have created a memory issue, but they were saved as NumPy array instead to be temporarily stored and then discharged. Moreover, in Darknet if there were one or more vehicles of the same type in the input picture, only one license plate was detected. To solve this problem, a new function was created which stores the coordinates of license plate bounding boxes without storing them in the original input file and a text file. In addition, we had to reshape the license plate bounding boxes coordinates because the input image of the vehicle detection has to be reshaped to be fed in the first model. As a consequence, the resulting coordinates of the license plate detection model were shifted concerning the original image. Furthermore, we had to adjust the code to run models in CPU, even if at the beginning the models were running only in CUDA.

4.2.2. Vehicle Classification & Segmentation

In this study project, we used Mask R-CNN, which detects vehicles by instance segmentation. In addition, the MMDetection toolbox and its libraries were used to accomplish the task. The model is trained with free GPU-based Google Colab then its trained weights are used later. Why did we choose to use mask R-CNN? To answer this question, we need to understand the background of the R-CNN model.

The goal of R-CNN is to produce bounding boxes for images. It slides windows across the image using different sized rectangular regions. The first step is to generate region proposals that identify the areas for detection. The second step is to use a neural network to extract fixed features from each region. The third step is to classify the regions. However, there are some challenges with R-CNN. Such as, it takes more time to train, which is a computationally expensive model. The real-time implementation is quite difficult [67]. Therefore, fast R-CNN was developed. In this approach, an input image is given to the convolutional neural network (CNN) to extract the feature map. Using this map region proposal can easily be identified by using a pooling layer and resizing them into fixed sizes then it feeds into the fully connected layer. Finally, the softmax layer is used to predict different classes and bounding boxes[67]. In this approach, the selective-based method has been removed. The new network predicts the regions of interests itself.

These predicted regions of the proposal are reshaped using the ROI Pooling layer and fed into the classifier [67]. Furthermore, Mask R-CNN [68] is the state-of-the-art technique that developed on top of Faster R-CNN [67]. It segments the image into a set of pixels that locate the object and its position in the image. As far as semantic segmentation is concerned, it just classifies the image with a single class. However, in this study project, we used instance segmentation because it is not only segments the image but also detects, localizes, and classifies the different objects in the images. The output of this mask is always distinct from classification and label and simple to train. It can be generalized to another task. Mask R-CNN also follows two-stage procedures like faster R-CNN. However, in the second stage, the model is parallel with the class and box. It outputs a mask on each object.

The MMDetection toolbox also includes several weights for more than 200 pre-trained networks. It is constructed in a modular manner using PyTorch implementation, making it simple to reimplement or create additional detectors. In addition, the pipeline can be readily customized using existing custom modules [69]. It contains Some universal architectures that can be used in any model.

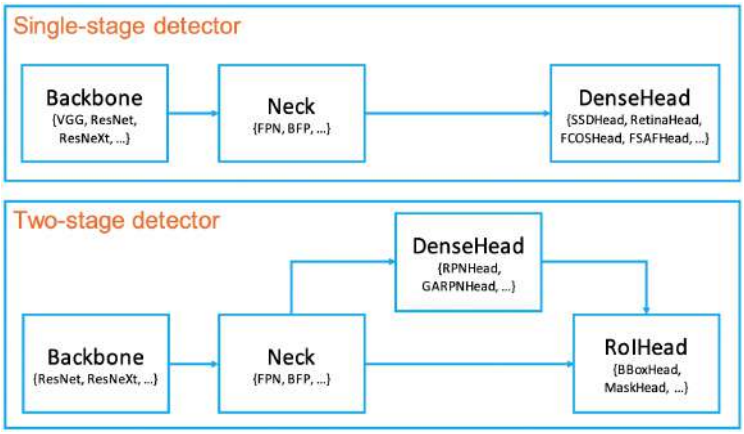

Figure 4.6.: MMDetection architecture [70]

In the following, the elements that make up MMDetections overall architecture (Figure 4.6) will be explained. *Backbone* is the section of a ResNet-50 that converts a picture to feature maps without the last completely linked layer. The backbone and heads are connected by the *neck*. It refines or reconfigures the raw feature maps generated by the backbone. DenseHead is the part of AnchorHead and AnchorFreeHead that works with dense locations on feature maps. With RoIPooling-like operations, RoIExtractor extracts RoI-wise features from a single or many feature maps. RoIHead accepts RoI characteristics as input and makes RoI-

specific task-specific predictions, such as bounding box classification/regression and mask prediction.

In Figure 4.7 the outcome can be seen where bounding boxes, scores, locations of the object, and instance segmentation are displayed. It initially identifies the objects and classifies the bounding Box. Then, it shows a confidence score that indicates the object's detection with high accuracy. Finally, instance segmentation is applied using the masking approach.

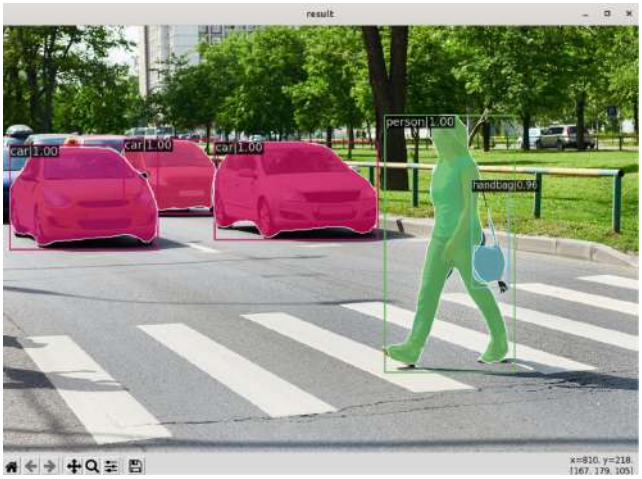

Figure 4.7.: MMDetection results

4.2.3. Text Detection

The search started with some recommendations on datasets for texts and criteria, such as the model running with PyTorch and not TensorFlow. The datasets in our focus were ICDAR-2013, ICDAR-2015, Total-Text, and SCUT-CTW1500. Additionally to the datasets, we also looked for tasks such as scene text detection, curved text detection, handwritten text detection, etc. All models were collected in a table with the datasets they use, the link to their Papers with Code[13] page, and the link to their publication. Among all models, TextFuseNet was the model of choice for this task due to its exemplary performance.

TextFuseNet [71] performs the best with the text datasets. Moreover, unlike the other models, they use a novel approach, where they neither focus only on characters or words but introduce the global-level features. By considering text detection as an instance segmentation task, it can detect arbitrary shapes. It also has richer fused features. The approach consists of extraction of multi-level feature representations: character-level, word-level, global-level and performing a multi-path fusion for text detection. The model has 5 com-

ponents, a Feature Pyramid Network (FPN) as the backbone of the model which extracts multi-scale feature maps, a Region Proposing Network (RPN) to generate text proposals, and 3 branches for segmentation, detection, and masking. Figure 4.8 shows the overall pipeline of the model. The version that was used for the project is ResNet-101.

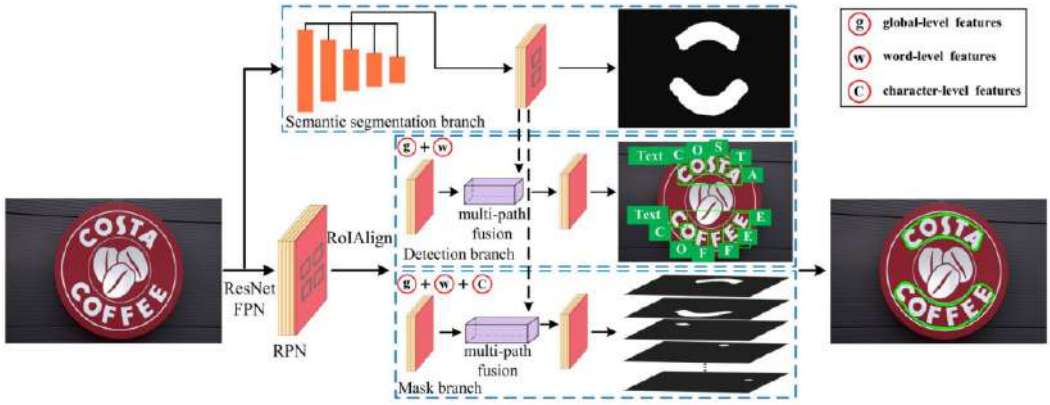

Figure 4.8.: The pipeline of the TextFuseNet model [71]

Due to some technical problems with *pycocotools* in one of the computers, the model could not work on that one. So, it became necessary to write a new installation guide and supplement it with an already setup environment. This guide has worked on 2 computers but it did not solve the problems for the one computer mentioned above. Furthermore, the model's installation guide on the GitHub[8] uses pip install, and also the version of the Python does not match ours. As we have agreed to use conda environments and Python 3.9, there was a need to modify the setup. Another extensive installation guide was written. The implementation has been tested for Windows and Linux. On Linux the model works fine, however, one of the computers with Windows had faced some problems during the installation. Even after the modified installation guide, the problems were sustained. A downside is that the model uses Detectron2, yet installing Detectron2 to Windows directly is not possible.

After the model was set to work in conda environments and with Python version 3.9, some random images with texts were used as input images to see how the model performs with original scene photographs. First, the images in Figure 4.9 were used. We can see here that when the model is trained with different datasets, output images vary. As it can be seen, the ICDAR-2013 dataset does not show good performance when it comes to curved text. It can detect the text but not all the characters individually. Further, it can also confuse the heart

[8] `https://github.com/ying09/TextFuseNet`

Figure 4.9.: From left to right, the outputs from the model trained with ICDAR-2013, ICDAR-2015, Total-Text, and CTW-1500.

shapes with curved text. ICDAR-2015 performs well enough with the normal text. Again, the curved text detection is not very good. Among all the datasets here, the best performance belongs to CTW-1500. It can detect the texts with almost a perfect performance. Hence, for future testing, CTW-1500 was adapted. Below, in Figure 5, some images with random texts in the scene were tested with the model. These images are acquired from one of the member's personal albums on their mobile phone.

Figure 4.10.: Outputs from the model trained with CTW-1500

4.2.4. Face Detection

A biometric identifier, such as an iris, facilitates facial recognition. To provide a solution to protect personal information, we applied an anonymization technique to make faces unrecognizable. In our project, we found a face detection model that optimally detects all faces in images, including those with low resolution. In our search for an appropriate face segmentation model, we used the following resources: review articles and the Papers With Code[13] website. The most relevant article we found was paper published in 2021 "Face Segmentation: A Journey From Classical to Deep Learning Paradigm, Approaches, Trends, and Directions" [72]. This paper listed all of the existing face segmentation models until 2021. Since our project is mainly focused on Deep Learning, we have only included contributions that match our interests. To be thorough, we also checked their corresponding codes on GitHub[9]. The complete list is available in Table 4.3.

On the Papers With Code[13] website, we did not find a model for face segmentation with a bitmask. However, there were many available models for face detection. TinaFace[73], RetinaFace[74] and MTCNN[75], to name a few. To decide which model to choose for our project, we used the following criteria, considering the Papers With Code[13] results and the Table 4.3:

- The model must perform well on a dataset that is challenging and represents real-world applications;
- The model should be state-of-the-art and implemented in the PyTorch framework;
- The implementation of the model should be publicly available.

We found that the benchmarks for face detection and face segmentation are different. For the latter, the HELEN dataset was most commonly used. For the former, there wasn't one "most used dataset" but rather plenty of varieties. There is also a difference between methods – to the best of our knowledge, the face detection method works mostly with a bounding box. The segmentation method gives the user a face shape, i.e. the bitmask.

Of all the benchmark datasets, The WIDERFACE dataset [13] in particular caught our attention (as mentioned in section 2.2.3). WIDERFACE dataset contains wide range of images and, on top of this, has three different "difficulty" labels: easy, medium, and hard. For instance, the "hard" category is the most complex category out of three to detect faces. The facial coverage on "occlusion" section or images taken in different illumination setting Fig-

[9] `https://github.com/`

Table 4.3.: Deep leanring face detection models [72]

Model	GitHub	Article	Published year
Face Parsing via Tanh Warping	Link	Lin *et al.* [76]	2019
Face Segmentation	Link	Nirkin *et al.* [77]	2017
Face Labeling	Link	Liu *et al.* [78]	2015
Face Parsing Network	Link	Li *et al.* [79]	2017
Face segmentation with CNN and CRF	Link	Jackson *et al.* [80]	2016
Structure via Consensus	Link	Masi *et al.* [81]	2020
EHANet: Face Parsing	Link	Luo *et al.* [82]	2020
Facial Attribute Segmentation	Link	Kalayeh *et al.* [83]	2017

ure 2.6 shows few of these examples. We decided to go in depth with WIDERFACE (Hard) benchmark to search for the model.

We chose RetinaFace based on the model's performance (third best in WIDERFACE - Hard category) and its implementation in PyTorch by Biubug6 [10]. In addition, RetinaFace is "The most implemented Paper" in Papers With Code[13] website on face detection. We evaluated the model by running their evaluation script on the WIDERFACE dataset. Finally, we used the ResNet50 as a backbone network, and obtained an AP of 94.48%, 94.04%, and 84.43% for the easy, medium, and hard datasets, respectively. These results were similar to what they reported in their documentation.

The architecture of RetinaFace consists of three components Figure 4.11: Feature Pyramid Network, Context Head Module, and Multi Task Loss.

"Feature pyramid network" takes the input image and outputs five feature maps at different scales. "Context Head Module" uses a deformation convolutional network (DCN) over the feature maps. "Multi Task Loss" has four different features, one of which predicts the bounding box. For more information on the architecture of the model, see the article [74].

[10] `https://github.com/biubug6/PyTorch_Retinaface`

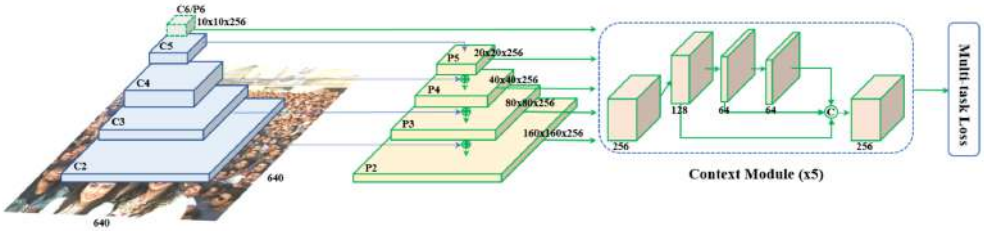

Figure 4.11.: RetinaFace architecture [74]

4.2.5. Person Detection

From the models proposed in the research phase in section 2.4 we selected the models based on their accuracy and discarded SEG-YOLO, because it is optimized for speed at the cost of accuracy. The selection between SeMask and CBNetV2 was difficult, since they were evaluated on different datasets making it impossible to compare them directly. So we looked at the amount of different scenes represented by the dataset, the amount of occurrences of the label "person" and the difficulty of the dataset. COCO has a much higher image count (120k) than ADE20K(25k), but has a lower frequency of persons in it (total 70k vs 24k). The best model on the COCO dataset has a worse accuracy than the best on ADE20K, so we regarded COCO as more difficult. The accuracy drawback of SeMask and CBNetV2 from the best model on the corresponding dataset was equally low. We chose CBNetV2 which was trained on the COCO dataset since it provided five times more images and three times more persons represented in them, including more poses and different persons. There are many more factors which we could have included for the selection process, but since those models were already pre-selected as performing well, we wanted to find a simple metric to select a model for our main task, which is the implementation.

To be able to use the model in our environment, we acquired information on how the used framework (MMDetection) works and how to install the correct package versions and requirements. Since we had project requirements like using python 3.9 we had to tweak the given versions and installation processed until we got it running. In this trial and error process we learned, that it is best to create the conda environment directly with as many packages as possible instead of installing them one after another, which often prevents package version conflicts. Also it is important to search the internet and tweak the presented solutions to your case. If they are for another version for example, they might be applicable to your problem too.

After creating a working environment and understanding the workflow of MMDetection we were able to run the model with our own script, validating the results in the paper.

Figure 4.12.: Exemplary Result - Mask & Segmentation

To be able to run the model in the shield pipeline, the implemented inference had to be adapted to the provided classes and structures. One thing that had to be changed was using different batch norm layers in the model config. The default setting was 'SyncBN' which can only be used on the GPU with CUDA. To run the model on a CPU this had to be changed to 'BN'. Another thing was the extraction of the classes from the model. Here we used the PyCharm debugger to look into the runtime variables where the result of the model is evaluated and the results are assigned to classes.

4.3. MVP Extensions

MVP extensions were the upgrade phase of the basic MVP to improve our product with more advanced features. These extensions added more use cases and functions to our product. In this phase, we implemented anonymizing methods like Face anonymization which results in visually appealing images with synthetic realistic faces of people. Additionally, we applied detection methods for complex features in humans, such as tattoos. Furthermore,

we used detection techniques for challenging objects that could contain privacy-sensitive information, such as computer and mobile phone screens.

4.3.1. Face Anonymization

Substituting faces in images with synthetic faces preserves the privacy of individuals while keeping the face's basic features and the utility of the image [84]. We considered the state-of-the-art GANs (generative adversarial networks) that generate synthetic faces and viewed three models, namely, CIAGAN [85], DeepPrivacy [56], and Sun2018 [86].

Our criteria for choosing the optimal model were that the implementation of the model should be publicly available, the model's performance regarding privacy preservation should be good, and the model outputs natural-looking synthetic faces. CIAGAN and DeepPrivacy have publicly available codes, but Sun2018 does not have any publicly available codes. According to Klomp *et al.* [84], DeepPrivacy is the best performing GAN and the best model to generate natural-looking images. The DeepPrivacy model was also ranked number one on the Papers With Code[13] website, so we decided to use the DeepPrivacy model. We can see in Figure 4.13 that the output image from the DeepPrivacy model looks more natural than the results from Sun18 and CIAGAN.

Figure 4.13.: Example output of several anonymization methods [84]

The DeepPrivacy model works by extracting seven key points from the faces using the Mask-RCNN face detector. Mask-RCNN performs well on faces with low resolution e.g, small faces with only 10 pixels. Additionally, it draws a black box on the face to prevent re-identification. The dual shot face detector is used to draw a bounding box around the faces.

A conditional GAN takes in the original pose and image background along with the input that allows the generation of faces that fit the existing background. Afterward, the U-net generator constructs a new face according to the face area and the extracted key points. The resolution of the face generated is 128 × 128 pixels [84] [56]. DeepPrivacy is trained on the Flicker Diverse Faces (FDF) dataset. FDF is a large dataset of approximately 1.47 million faces, with a variety of poses, face occlusions, genders, and ethnicities [56]. Nevertheless, DeepPrivacy has some limitations that could be improved, as shown in Figure 4.14, we can see that the model does not perform well when several faces are occluded or when faces have irregular poses.

Figure 4.14.: Limitations of the DeepPrivacy model. On the left side is the original image and on the right is the output image.

We created the anonymized images and validated them. We used Python 3.9 and we created a conda environment for executing the commands from the Github page of the DeepPrivacy[11]. We tried different forms of input images that covered cases of different backgrounds and occlusions to check if Deep Privacy works for all these cases. The path where we saved our input image in the code was modified. We chose the recommended "fdf128_rcnn512" face detection model to create the anonymized images.

4.3.2. Screen Detection

Screen detection models are still not publicly available and new in the research field. For this reason, it was difficult to find models and papers using a specific model trained only for detecting screens of monitors, phones, and so on.

During the research phase, the project we came across is “Phone-Detection-using-Deep-Learning[12] which uses a deep learning model with convolutional neural network based ap-

[11] `https://github.com/hukkelas/DeepPrivacy`
[12] `https://github.com/shrinivasshetty21/Phone-Detection-using-Deep-Learning`

proach along with a feed-forward network to identify the center pixel location for an object, in this particular case a phone, present in an image.

Table 4.4.: Screen detection available models

Model	Datasets	Link	Article
CNN	Unknown	link	None
SSD-Faster R-CNN	IITH-GDMU, KaggleDriver	link	Rajput *et al.* [87]

Other objects than phones were later tested to be identified as monitors of computers using the above-mentioned project. However, the described model was discarded for this project, as the model outputs only the center of the pixel's location, and the model training has not had enough data.

Briefly, we came across a paper that can be resourceful in this project. In [87], it was used a single shot multi-box detector (SSD) and a faster region-based convolution neural network (Faster R-CNN) to detect mobile phone usage. The primary goal of the paper was not only to detect the presence of a mobile phone and a person but also to detect whether the mobile phone is being used by the person. What can be highlighted is that the focus is merely on the mobile phone, and it was ignoring other objects that could have screens such as monitors. Since the authors' focus was only narrowed down to the usage of mobile phones, the purpose of the project does not match the scope of our study project and hence, it was discarded.

Eventually, we found the YOLOv5 model which is 2.5 times faster than Faster R-CNN [88]. It can detect smaller objects with high accuracy. The first challenge was to train YOLOv5 on a custom dataset. Therefore, kaggle's phone and laptop dataset was exported via Roboflow[13]. It includes 336 images. Some part of the dataset was not annotated. We annotated them and converted into PyTorch format and trained them with the help of Colab because it contains free GPU. We use custom YOLOv5 weights for further inferences. In the end, we integrated it into our pipeline.

Result: this is the final result of our output that displays bounding box, location and confidence value with high accuracy.

[13] `https://roboflow.com/`

Figure 4.15.: Mobile phone and laptop detection.

4.3.3. Tattoo Anonymization

Tattoos, unlike fingerprints, or iris, are not strictly defined features of human appearance, and they are subject to change at any time. It is a challenge to detect whether a tattoo is in a particular location [89]. Da Silva [89] defines the road map for tattoo detection in the Figure 4.16. In this road map, tattoo recognition consists of two main parts: pre-processing and recognition. Each branch follows and compliments the other, resulting in a complete process. Therefore, pre-processing becomes a crucial procedure to achieve error-free recognition.

Pre-processing has 2 steps. Detection searches a tattoo, location locates the tattoo if there is one, and implements a bounding box to the corresponding region. Localization can be achieved in two methods: segmentation crops the contour of the tattoo and removes the background, and semantic segmentation segments each represented object in the tattoo. Recognition, on the other hand, consists of three parts. Classification describes the object(s) in the image, and de-identification is the anonymization process. Lastly, re-identification has four methods: Image-to-image, sketch-to-image, similar groups, and partial image-to-image.

Da Silva takes a different approach from the previous works on tattoo detection and uses transfer learning. This work uses a CNN architecture that has already been used in another recognition task. This re-used part works as a feature extractor. The rest of the network is developed to recognize the tattoos. The pipeline for the task is visualized in Figure 4.17.

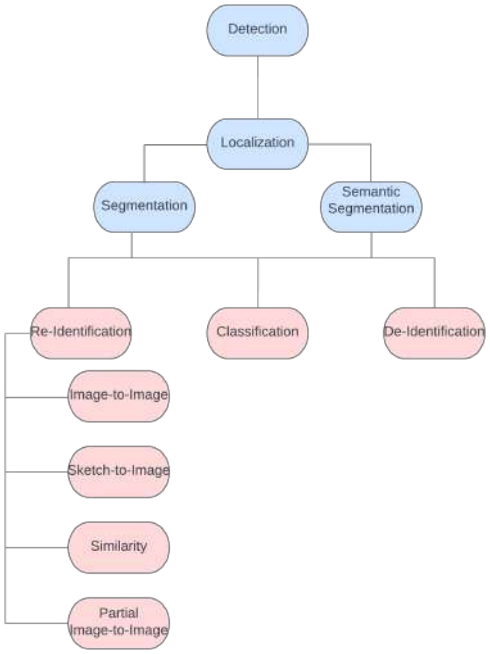

Figure 4.16.: Tattoo Recognition Roadmap [89]

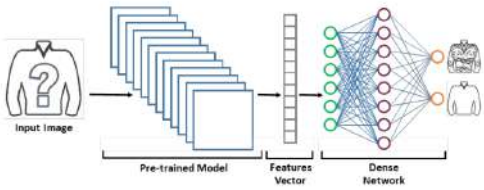

Figure 4.17.: Transfer learning pipeline [89]

There were not many tattoo detection and/or anonymization models on the Internet. We could only find two models that ran on PyTorch: SkinDeep[14] and ShadowGP[15]. We used the former to integrate into our pipeline since it matched our product's needs. The latter one had an issue – initially, the model was developed to detect and remove the shadows from the faces. However, it was mentioned in the GitHub repository that the model can also detect and remove face tattoos. We were searching for a model to anonymize tattoos from any part of the skin. Therefore, we took ShadowGP out of consideration.

[14] `https://github.com/vijishmadhavan/SkinDeep`
[15] `https://github.com/YingqingHe/Shadow-Removal-via-Generative-Priors`

SkinDeep aimed to use its model to remove tattoos from images. To achieve this goal, they needed dataset of image pairs - tattoo and not tattooed images - dataset. Since such datasets did not exist or were not publicly available they opted for synthetic data. Using libraries such as OpenCV and image manipulation software like Photoshop, they created the dataset to train their model.

When we tried to use the SkinDeep model, we encountered some issues (they were already mentioned in the repository). The model required high-resolution images to perform well and the output of the model was limited to 400px images. As far as we could understand model's work, a person detection model was implemented in the pickle file itself. Once the person was detected it would crop 400px of the detected person and produce the de-tattooed version of it.

To solve this problem, we fed the model only a cropped section of the image (where the face or person was) and then reattached it to the original image. With this approach, we already knew where we took the cropped image so the output resolution of the de-tattooed image would stay consistent with the original tattooed version Figure 4.18[16]. Another advantage of this approach was that it could minimize the problem of the model recognizing non-tattoo patterns as tattoos. For example, anonymizing grass or clothes Figure 4.19.

Figure 4.18.: Original tattoo image and de-tattooed version

For the unique cases where the cropped image had less than 400 pixels, we padded the image on the right and bottom side with 0s Figure 4.20. We also tested the padding of the cropped image on all four sides, however, the result most of the time looked less quality than only padding on two sides ("quality" as a subjective criterion for comparing two images, which one looked more "natural", i.e., the original image was not too disturbed and how well the tattoos were removed).

[16] `https://nym.ag/3rqwtTo`

Figure 4.19.: Limitation of SkinDeep model. Anonymization of non-tattooed parts of the image such as Iris, grass and clothes

Figure 4.20.: The cropped image bottom and right padding

If the width or height of the cropped image was greater than 400 pixels, we “borrowed” the missing pixels from the patch before it (Figure 4.21). For example, if the width of the image were 500 pixels, the first 400 would not have an issue running through the model. However, with only 100 pixels left for the second patch, we reused 300 pixels from the first patch. for the "overlapped" part, we averaged the values in this 300-pixel area when we re-attached the cropped image to the original image.

Figure 4.21.: Overlapped pixels between the patches

5. Results & Discussion

5.1. The Product

The product we developed during our study project was divided into two parts: back-end, and front-end. The back-end consisted of a pipeline that incorporated person, face, text, vehicle, license plate and screen detection, together with the anonymization techniques: blurring, masking, deep face and deep tattoo anonymization, into a streamlined de-identification routine. The pipeline was highly modular, and the model super-class with individual data classes for the model's extracted keypoints. We could implement new detection and anonymization methods without further adjustment to the pipeline itself. The entire pipeline was overlayed with an API, that could provide access to different end-user devices. We have created documentation of our API, to access the pipeline's interface with server requests or in a rudimental python script fashion, as well as individual detection and anonymization techniques. Each model package was substituted with the necessary information to create a working environment and code examples that utilize the sub-packages. The front-end consisted of a standard web interface, which facilitated basic user needs. Images could be uploaded, different detection and anonymization models may be selected, and in the end the resulting images downloaded. However, the entire product can be used without the front-end by simply using the API entry points to the pipeline. The product can be deployed with just one command on the command line. This command creates several Docker containers, which incorporate the Deep Learning models with the pipeline. Besides providing a rich feature set, the product covers all the basics a working software package has to offer: modulability to continue growing, deployability to provide interactivity, and explainability to understand the framework's working principles and information flow.

5.2. Improvements and GDPR Concerns

In our last development phase, we were able to implement two advanced techniques for distorting detected parts within an image. Our two standard techniques for privacy en-

hancement of the images - masking and blurring - each had a disadvantage. Blurring of an image can lead to the re-identification of the anonymized content when the blurring kernel is inverted. Masking areas within an image not only anonymizes the image but also reduces its utility, thus harming the natural perception of an image. Therefore, we provided two deep anonymization techniques - tattoo anonymization and face anonymization - to provide the end-user with an irreversible and more natural-looking anonymization scheme. There are numerous possibilities to extend our application even further. One example would be the ability to register privacy preferences. Given a predefined database, the application could identify individuals in an image and de-identify people based on the user's agreement. One could also extend the detection features to include even more de-identification capabilities or boost the accuracy of the existing methods. However, these would only increase the richness of the product and would not provide a higher GDPR conformity of the application. At the current state of our product, we cannot guarantee full GDPR compliance, because we lack the quality assurance feature for the anonymization processes. The current product is missing an automated evaluation process, that extracts the anonymization error - accuracy and precision - of the resulting images and validates the anonymization confidence. Additionally, anonymized images need to be checked, whether the de-identified information can be re-identified with state-of-the-art image reconstruction methods. When both these criteria are met, the product might be labeled with full GDPR compliance. However, there may still be borderline cases in which the software fails to de-identify personal information, and lead to legal problems.

5.3. The Agile Framework in a University Setting

Throughout the project, we adapted to several changes in organizational frameworks. We started with an adjusted version of Scrum, with four product owners instead of one, and semiweekly stand-ups instead of daily meetings. Due to communication issues, we changed the structure of our semiweekly meetings to a more in-depth feature presentation. Slack was used to communicate tasks in detail and their progress, instead of just using it for feedback. Eventually, we formed three sub-groups (Product Owners, Developers, and Documentation Team) within the project, each responsible for different tasks. These groups were managed on interdependent basis, with task-to-task communication allowing for rapid customization and adjustability between and within the sub-groups. Since we did not want to have a chaos-driven approach of working, nor did we want to keep people uninformed on the current state of the project, we refined our organization structure until we found a solution that provided

comprehensive and transparent information about the progress of the product. We also encouraged each other to discuss problems or ask questions to clarify issues.

The most progress-generating setting within the university was the short, semiweekly meetings. In comparison, asynchronous communication came with its challenges, as individual team members had different work schedules. The study project coincided with COVID-19 pandemic. Government and university regulations did not allow us to meet on-site for approximately six months during the first part of the study project. When the COVID-19 regulations were less restricted, LMIS offered us a meeting room on Mondays so we could meet in person.

5.4. Intra-Team Collaboration

At the individual team member level, the overall study project was a success and provided knowledge on topics like team management, scientific writing, and software engineering. The commitment of the individual team members, in intra-team collaboration settings, was always high. This created vital group dynamics and fostered the growth of the study project. In the beginning, we established four values: fun, clarity, knowledge sharing, and teamwork. After a few weeks, all participants incorporated these into their communication styles and work routines. Within the group, members offered workshops and assistance to those who were not proficient in a particular task. Meetings aimed to maintain a dialectical discourse about the product, feature, organization, and structural decisions. Product Owners which consisted of four members guided the overall project progress by creating, structuring, and communicating tasks in a transparent fashion.

5.5. Cooperation with the Company LMIS (Inter-Team Cooperation)

With LMIS on our side, we were able to roughly design the entire project from the beginning and were well on track to develop a professional product. LMIS was eager to help out in any situation we faced with difficulty. Whether it was on an individual level or on a team level. The former support was in the mode of in-depth one-on-one meetings and the latter by providing educational resources and workshops, on topics such as project organization and team building. We were able to experience the workflows of an industry-grade project organization software Azure DevOps. It provided a rich toolset for organizing teams that

we tried to fully implement into our work routines however, it was too sophisticated for our purposes. For the most part, we needed to adjust to working in agile, but the amount of overhead that was involved in implementing tasks slowed our progress. The top-down approach of Azure DevOps provided a big feature set with resources from various frameworks (e.g., Kanban Board, Backlog, Sprint, Project Sub-Groups). This created more confusion and miscommunication in our team dynamic. Instead, opting for a lightweight tool may have been advantageous in our case. The downside of working with a company was that we had to sign a non-disclosure agreement (NDA) to ensure that no confidential company data will leak and all information related to the project will not be disclosed. During the study project, we had several discussions about the NDA with some members disagreeing with the content. However, there was no expert within the study project on the legal aspect of such a document. While most of us eventually signed it, some left the project. The restrictions we accepted are: we were not to share our code with anyone except the LMIS; we do not use the project idea and result for any other project; we do not use the created product outside the study project. In the end, one's interest to gain experience in an industry-like work environment or academic-like environment guide their decision to pursue either one. There are some advantages and disadvantages mentioned above but one choice matches one's motivations.

6. Conclusion

The breach of personally identifiable information (PII) can result in substantial harm to an individual, including identity theft or other fraudulent use of this information. In our project, we developed a GDPR-compliant solution to protect PII in images. Our basic MVP solution integrates person, face, text, vehicle, license plate and screen detection with standard anonymization methods such as masking and blurring. However, the basic MVP version could allow re-identification as blurring is a potentially reversible process (in case the blurring kernel is inverted). To solve this irreversibility issue and conform to GDPR, we extended our MVP. The extended product is our final and most promising solution as it applies deep face and tattoo anonymization together with the six aforementioned detection techniques.

Appendix

A. Workshops

In our project, the members came from different backgrounds. Each member had their own interests and competencies. This created a great opportunity where the members could help each other by providing workshops and presentations to share their knowledge. Holding a workshop was not mandatory, however, it was highly encouraged. In the following section, we mention the content of several workshops we had during our project.

A.1. Git

When working on a software project as a group, there is no way around but using a version control system such as *git*. The members usually have varying knowledge and prior experience working with git in a university project. That is why we conducted a git workshop where group members with more git experience explained the general basic git workflow. In addition, we covered topics such as branches and pull requests. Doing such a workflow allowed us to work more effectively during the project phases, where a lot of coding happened.

A.2. Docker

In our project, we had to work with multiple models for the different implemented features, and some of the models required an older version of Python. We needed a separate container for each feature to avoid dependencies conflict. Thus, we had a workshop on Docker, where we explained what Docker is, as mentioned in section 4.1.4. In addition, we had a live coding session showing how to use Docker.

A.3. Style Guide

In our project, we used a style guide to have a consistent style. The best advantage of a consistent code style is readability. Whole project's code should look similar, so after grasping its structure (style), it gets easier to understand. And since most code is changed over time and read way more often than written, this is a crucial aspect to save time and nerves of the developers.

For the SHIELD project, we used the tools black and isort. We also created our own style guide sheet, which extracted the most crucial information from PEP8 as bullet points, and was adapted in some cases to our needs. For documentation, we used the docstrings in the google style format.

A.4. Grid Computing

To create individually trained models or use transfer learning for features of our product we needed access to hardware that not every student owns. The Grid Computing cluster at the Institute of Cognitive Science provides high-performance hardware to any student of the University of Osnabrück. We presented the general principles of distributed computing and provided an interactive tutorial with example code and different use cases. During this workshop topics like running systematic tasks for testing and training several network architectures or optimization tasks to train a network with different parameters, were covered. As the Grid is implemented on a Linux-based system and can only be accessed via a Secure Shell connection, we gave an introduction to the command line and some Linux commands, too.

A.5. GDPR

To create a product that concedes to GDPR, we held a workshop for GDPR, where the workshop did not only highlight the importance of Data Security from the standpoint of a person but also the importance for companies regarding legal issues when not complying with the outlined regulations. If the company doesn't conform to the GDPR standard, it might be charged with a fine of up to 4% of the annual worldwide turnover. Furthermore, the workshop described the predominance of the dis-ambiguity in image anonymization and the parts of an image that need anonymization for the image to be considered GDPR-conforming. The GDPR Recital 26 States [1]:

1. The principles of data protection should apply to any information concerning an identified or identifiable natural person.

2. To determine if a natural person is identifiable, thus all the possible identification means should be taken into consideration, such as singling out either by the controller or by another person.

This section of the law is not explicitly defined and, therefore, it might lead to misinterpretations.

List of Figures

List of Tables

Acknowledgements

We would like to extend our deepest gratitude to our supervisors Dr. Ulf Krumnack and M.Sc. Ann-Christin Meisener for their constant feedback and invaluable guidance throughout the project. We would also like to thank the students, who supported us as interdisciplinary course participants: Cedric Bitschene, Dan Emilian Croitoru, Nele Danske, Niklas Heidemann, Madhuri Ramesh, and Hermann Singer.

References

1. GDPR. *General Data Protection Regulation* https://gdpr-info.eu/.
2. Xiao, Y., Xue, M., Lu, T., Wu, Y. & Palaiahnakote, S. *A Text-Context-Aware CNN Network for Multi-oriented and Multi-language Scene Text Detection* in *2019 International Conference on Document Analysis and Recognition (ICDAR)* (2019), 695–700. doi:10.1109/ICDAR.2019.00116.
3. Wigington, C. *et al. Start, Follow, Read: End-to-End Full-Page Handwriting Recognition* in *Proceedings of the European Conference on Computer Vision (ECCV)* (Sept. 2018).
4. Multi-national and Multi-language License Plate Detection using Convolutional Neural Networks. **10.** doi:10.48084/etasr.3573.
5. Nayef, N. *et al. ICDAR2019 Robust Reading Challenge on Multi-lingual Scene Text Detection and Recognition — RRC-MLT-2019* in *2019 International Conference on Document Analysis and Recognition (ICDAR)* (2019), 1582–1587. doi:10.1109/ICDAR.2019.00254.
6. Nayef, N. *et al. ICDAR2019 Robust Reading Challenge on Multi-lingual Scene Text Detection and Recognition – RRC-MLT-2019* in (July 2019).
7. Sahel, S., Alsahafi, M., Alghamdi, M. & Alsubait, T. Logo detection using deep learning with pretrained CNN models. *Engineering, Technology & Applied Science Research* **11,** 6724–6729 (2021).
8. Bastan, M., Wu, H.-Y., Cao, T., Kota, B. & Tek, M. Large scale open-set deep logo detection. *arXiv preprint arXiv:1911.07440* (2019).
9. Su, H., Gong, S. & Zhu, X. *Weblogo-2m: Scalable logo detection by deep learning from the web* in *Proceedings of the IEEE international conference on computer vision workshops* (2017), 270–279.
10. Romberg, S., Pueyo, L. G., Lienhart, R. & Van Zwol, R. *Scalable logo recognition in real-world images* in *Proceedings of the 1st ACM international conference on multimedia retrieval* (2011), 1–8.

11. Gillis. *Definition biometrics* Last accessed: 2022-09-22. https://www.techtarget.com/searchsecurity/definition/biometrics.
12. Ghalleb, A. E. K. & Amara, N. E. B. Soft and hard biometrics for the authentication of remote people in front and side views. *International Journal of Applied Engineering Research* **11,** 8120–8127 (2016).
13. Yang, S., Luo, P., Loy, C. C. & Tang, X. *WIDER FACE: A Face Detection Benchmark* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).
14. Wang, J., Yuan, Y., Yu, G. & Sun, J. SFace: An Efficient Network for Face Detection in Large Scale Variations. *arXiv preprint arXiv:1804.06559* (2018).
15. Papers with Code. *Papers with Code* Last accessed: 2022-09-20. https://paperswithcode.com/.
16. Lin, T.-Y. *et al. Microsoft coco: Common objects in context* in *European conference on computer vision* (2014), 740–755.
17. Shao, S. *et al.* Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123* (2018).
18. Flickr. *Flickr* Last accessed: 2022-10-02. https://flickr.com/.
19. Liu, X., Meng, G. & Pan, C. Scene text detection and recognition with advances in deep learning: a survey. *International Journal on Document Analysis and Recognition (IJDAR)* **22,** 143–162 (2019).
20. Lin, H., Yang, P. & Zhang, F. Review of scene text detection and recognition. *Archives of computational methods in engineering* **27,** 433–454 (2020).
21. Minetto, R. *et al. Text detection and recognition in urban scenes* in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)* (2011), 227–234.
22. He, T., Huang, W., Qiao, Y. & Yao, J. Text-attentional convolutional neural network for scene text detection. *IEEE transactions on image processing* **25,** 2529–2541 (2016).
23. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. *You only look once: Unified, real-time object detection* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 779–788.
24. Ren, S., He, K., Girshick, R. & Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* **28** (2015).
25. Neumann, L. & Matas, J. *Scene text localization and recognition with oriented stroke detection* in *Proceedings of the ieee international conference on computer vision* (2013), 97–104.

26. Epshtein, B., Ofek, E. & Wexler, Y. *Detecting text in natural scenes with stroke width transform* in *2010 IEEE computer society conference on computer vision and pattern recognition* (2010), 2963–2970.
27. Yin, X.-C., Yin, X., Huang, K. & Hao, H.-W. Robust text detection in natural scene images. *IEEE transactions on pattern analysis and machine intelligence* **36,** 970–983 (2013).
28. Busta, M., Neumann, L. & Matas, J. *Fastext: Efficient unconstrained scene text detector* in *Proceedings of the IEEE international conference on computer vision* (2015), 1206–1214.
29. Zhong, Y., Zhang, H. & Jain, A. K. Automatic caption localization in compressed video. *IEEE transactions on pattern analysis and machine intelligence* **22,** 385–392 (2000).
30. Zhang, Z., Shen, W., Yao, C. & Bai, X. *Symmetry-based text line detection in natural scenes* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), 2558–2567.
31. Yang, X., He, D., Zhou, Z., Kifer, D. & Giles, C. L. *Learning to read irregular text with attention mechanisms.* in *IJCAI* **1** (2017), 3.
32. Lou, X. *et al.* Generative shape models: Joint text recognition and segmentation with very little training data. *advances in neural information processing systems* **29** (2016).
33. Phan, T. Q., Shivakumara, P., Tian, S. & Tan, C. L. *Recognizing text with perspective distortion in natural scenes* in *Proceedings of the IEEE International Conference on Computer Vision* (2013), 569–576.
34. Lee, C.-Y. & Osindero, S. *Recursive recurrent nets with attention modeling for ocr in the wild* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 2231–2239.
35. Kirillov, A., He, K., Girshick, R. B., Rother, C. & Dollár, P. Panoptic Segmentation. *CoRR* **abs/1801.00868** (2018).
36. Jain, J. *et al.* SeMask: Semantically Masked Transformers for Semantic Segmentation. *CoRR* **abs/2112.12782** (2021).
37. Wang, Z. *SEG-YOLO: Real-Time Instance Segmentation Using YOLOv3 and Fully Convolutional Network* MA thesis (KTH, School of Electrical Engineering and Computer Science (EECS), 2019), 43.
38. He, K., Gkioxari, G., Dollár, P. & Girshick, R. B. Mask R-CNN. *CoRR* **abs/1703.06870** (2017).
39. Redmon, J. & Farhadi, A. YOLOv3: An Incremental Improvement. *CoRR* **abs/1804.02767** (2018).

40. Zivkovic, Z. & van der Heijden, F. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters* **27,** 773–780. doi:https://doi.org/10.1016/j.patrec.2005.11.005 (2006).
41. Keller, J. M., Gray, M. R. & Givens, J. A. A fuzzy K-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-15,** 580–585. doi:10.1109/TSMC.1985.6313426 (1985).
42. Minaee, S. *et al.* Image Segmentation Using Deep Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44,** 3523–3542. doi:10.1109/TPAMI.2021.3059968 (2022).
43. Ribaric, S., Ariyaeeinia, A. & Pavesic, N. De-identification for privacy protection in multimedia content: A survey. *Signal Processing: Image Communication* **47,** 131–151 (2016).
44. Meden, B. *et al.* Privacy–enhancing face biometrics: A comprehensive survey. *IEEE Transactions on Information Forensics and Security* (2021).
45. Barni, M., Donida Labati, R., Genovese, A., Piuri, V. & Scotti, F. Iris deidentification with high visual realism for privacy protection on websites and social networks. *IEEE Access* **9.** 2169-3536, 131995–132010 (2021).
46. Hrkać, T., Brkić, K., Ribarić, S. & Marčetić, D. *Deep learning architectures for tattoo detection and de-identification* in *2016 First International Workshop on Sensing, Processing and Learning for Intelligent Machines (SPLINE)* (2016), 1–5. doi:10.1109/SPLIM.2016.7528402.
47. Wrike. *Project Management Basics* Last ccessed: 2022-09-05. https://www.wrike.com/project-management-guide/project-management-basics/.
48. Wrike. *Project Management Methodologies* Last accessed: 2022-09-05. https://www.wrike.com/project-management-guide/methodologies/.
49. Ora. *Kanban vs Scrum vs Scrumban: What Are The Differences?* Last accessed: 2022-09-07. https://ora.pm/blog/scrum-vs-kanban-vs-scrumban/.
50. Harris, C. *Agile scrum artifacts.* https://www.atlassian.com/agile/scrum/artifacts#:~:text=Agile.
51. Van Rossum, G. & Drake, F. L. *Python* (CreateSpace, Scotts Valley, CA, 2009).
52. Django Software Foundation. *Django* https://djangoproject.com/.
53. The Pallets Projects. *Flask* https://flask.palletsprojects.com/en/2.2.x/.
54. Docker. *Docker* Last accessed: 2022-09-20. https://docker.com/.
55. Levkivskyi, I., Lehtosalo, J. & Langa, Ł. *PEP 544 – Protocols: Structural subtyping (static duck typing)* https://peps.python.org/pep-0544/#using-protocols.

56. Hukkelås, H., Mester, R. & Lindseth, F. *Deepprivacy: A generative adversarial network for face anonymization* in *International symposium on visual computing* (2019), 565–578.
57. Buchanan, I. *Containers vs. virtual machines* Last accessed: 2022-08-31. https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms.
58. Miell, I. & Sayers, A. *Docker in practice, second edition* (Manning Publications, 2019).
59. Docker. *Isolate containers with a user namespace* Last accessed: 2022-08-31. https://docs.docker.com/engine/security/userns-remap/.
60. IBM Cloud Team, I. C. *Containers vs. Virtual Machines (VMs): What's the Difference?* Accessed: 2022-08-31. https://www.ibm.com/cloud/blog/containers-vs-vms/.
61. Hat, R. *Vergleich zwischen Containern und VMs* Last accessed: 2022-08-31. https://www.redhat.com/de/topics/containers/containers-vs-vms/.
62. Laroca, R. *et al.* An efficient and layout-independent automatic license plate recognition system based on the YOLO detector. *IET Intelligent Transport Systems* **15,** 483–503 (2021).
63. Zhou, W., Li, H., Lu, Y. & Tian, Q. Principal visual word discovery for automatic license plate detection. *IEEE transactions on image processing* **21,** 4269–4279 (2012).
64. Hsu, G.-S., Chen, J.-C. & Chung, Y.-Z. Application-oriented license plate recognition. *IEEE transactions on vehicular technology* **62,** 552–561 (2012).
65. Gonçalves, G. R., da Silva, S. P. G., Menotti, D. & Schwartz, W. R. Benchmark for license plate character segmentation. *Journal of Electronic Imaging* **25,** 053034 (2016).
66. Laroca, R. *et al. A robust real-time automatic license plate recognition based on the YOLO detector* in *2018 international joint conference on neural networks (ijcnn)* (2018), 1–10.
67. Gandhi, R. *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms* Last accessed: Jul 9, 2018. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.
68. Odemakinde, E. *Everything about Mask R-CNN: A Beginner's Guide* https://viso.ai/deep-learning/mask-r-cnn/.
69. Lakshmanamoorthy, R. *An Object Detection Python Toolbox* Last accessed: April 24, 2021. https://analyticsindiamag.com/guide-to-mmdetection-an-object-detection-python-toolbox/.
70. Chen, K. *et al. MMDetection: Open MMLab Detection Toolbox and Benchmark* in (June 2019).

71. Ye, J., Chen, Z., Liu, J. & Du, B. *TextFuseNet: Scene Text Detection with Richer Fused Features.* in *IJCAI* **20** (2020), 516–522.
72. Khan, K., Khan, R. U., Ahmad, K., Ali, F. & Kwak, K.-S. Face segmentation: A journey from classical to deep learning paradigm, approaches, trends, and directions. *IEEE Access* **8,** 58683–58699 (2020).
73. Zhu, Y., Cai, H., Zhang, S., Wang, C. & Xiong, Y. Tinaface: Strong but simple baseline for face detection. *arXiv preprint arXiv:2011.13183* (2020).
74. Deng, J. *et al.* Retinaface: Single-stage dense face localisation in the wild. arXiv 2019. *arXiv preprint arXiv:1905.00641*.
75. Zhang, K., Zhang, Z., Li, Z. & Qiao, Y. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE signal processing letters* **23,** 1499–1503 (2016).
76. Lin, J. *et al. Face parsing with roi tanh-warping* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 5654–5663.
77. Nirkin, Y., Masi, I., Tuan, A. T., Hassner, T. & Medioni, G. *On face segmentation, face swapping, and face perception* in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)* (2018), 98–105.
78. Liu, S., Yang, J., Huang, C. & Yang, M.-H. *Multi-objective convolutional learning for face labeling* in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), 3451–3459. doi:10.1109/CVPR.2015.7298967.
79. Li, Y., Liu, S., Yang, J. & Yang, M.-H. *Generative face completion* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 3911–3919.
80. Jackson, A. S., Valstar, M. & Tzimiropoulos, G. *A CNN cascade for landmark guided semantic part segmentation* in *European Conference on Computer Vision* (2016), 143–155.
81. Masi, I., Mathai, J. & AbdAlmageed, W. *Towards learning structure via consensus for face segmentation and parsing* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 5508–5518.
82. Luo, L., Xue, D. & Feng, X. EHANet: An Effective Hierarchical Aggregation Network for Face Parsing. *Applied Sciences* **10.** doi:10.3390/app10093135 (2020).
83. Kalayeh, M. M., Gong, B. & Shah, M. *Improving facial attribute prediction using semantic segmentation* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), 6942–6950.
84. Klomp, S. R., Van Rijn, M., Wijnhoven, R. G., Snoek, C. G. & De With, P. H. *Safe Fakes: Evaluating Face Anonymizers for Face Detectors* in *2021 16th IEEE Interna-*

tional Conference on Automatic Face and Gesture Recognition (FG 2021) (2021), 1–8.

85. Maximov, M., Elezi, I. & Leal-Taixé, L. *Ciagan: Conditional identity anonymization generative adversarial networks* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), 5447–5456.
86. Sun, Q. *et al. Natural and effective obfuscation by head inpainting* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), 5050–5059.
87. Rajput, P., Nag, S. & Mittal, S. *Detecting usage of mobile phones using deep learning technique* in *Proceedings of the 6th EAI International Conference on Smart Objects and Technologies for Social Good* (2020), 96–101.
88. Dwivedi, P. *YOLOv5 compared to Faster RCNN. Who wins?* Last accessed: Jun 30, 2020. `https://towardsdatascience.com/yolov5-compared-to-faster-rcnn-who-wins-a771cd6c9fb4`.
89. Da Silva, R. T. & Lopes., H. S. *A Transfer Learning Approach for the Tattoo Detection Problem* in *Anais do 15 Congresso Brasileiro de Inteligência Computacional* (eds Filho, C. J. A. B., Siqueira, H. V., Ferreira, D. D., Bertol, D. W. & de Oliveira, R. C. L.) (SBIC, Joinville, SC, 2021), 1–8.