Received 6 October 2023, accepted 20 October 2023, date of publication 27 October 2023, date of current version 2 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3328227

RESEARCH ARTICLE

Revelio: A Modular and Effective Framework for Reproducible Training and Evaluation of Morphing Attack Detectors

GUIDO BORGHI, NICOLÒ DI DOMENICO, ANNALISA FRANCO, MATTEO FERRARA, AND DAVIDE MALTONI, (Senior Member, IEEE)
Dipartimento di Informatica—Scienza e Ingegneria (DISI), Università di Bologna, 47521 Cesena, Italy

Corresponding author: Guido Borghi (guido.borghi@unibo.it)

This work was supported by the European Union's Horizon 2020 Research and Innovation Program under Grant 883356.

ABSTRACT Morphing Attack, *i.e.* the elusion of face verification systems through a facial morphing operation between a criminal and an accomplice, has recently emerged as a serious security threat. Despite the importance of this kind of attack, the development and comparison of Morphing Attack Detection (MAD) methods is still a challenging task, especially with deep learning approaches. Specifically, the lack of public datasets, the absence of common training and validation protocols, and the limited release of public source code hamper the reproducibility and objective comparison of new MAD systems. Usually, these aspects are mainly due to privacy concerns, that limit data transfers and storage, and to the recent introduction of the MAD task. Therefore, in this paper, we propose and publicly release *Revelio*, a modular framework for the reproducible development and evaluation of MAD systems. We include an overview of the modules, and describe the plugin system providing the possibility of extending native components with new functionalities. An extensive cross-datasets experimental evaluation is conducted to validate the framework and the performance of trained models on several publicly-released datasets, and to deeply analyze the main challenges in the MAD task based on single input images. We also propose a new metric, namely WAED, to summarize in a single value the error-based metrics commonly used in the MAD task, computed over different datasets, thus facilitating the comparative evaluation of different approaches. Finally, by exploiting Revelio, a new state-of-the-art MAD model (on SOTAMD single-image benchmark) is proposed and released.

INDEX TERMS Face morphing, morphing attack detection (MAD), automated border control (ABC), face recognition, single-image MAD (S-MAD).

I. INTRODUCTION

Through a *Morphing Attack* [1], [2] an official document can be shared across two different people, destroying the unique link between the document and its real owner. Specifically, a face morphing attack consists of merging two different identities in a single facial image: in practice, a subject with no criminal records (*accomplice*) might apply for an official document using a morphed mugshot photo which hides the identity of a *criminal*. Indeed, several studies [3], [4] have shown that morphed images can be effectively used to fool both the human control, *e.g.* the officer responsible for the document issuing procedure, and the current *commercial-off-the-shelf* (COTS) *Face Recognition Systems* (FRSs).

For these reasons, the morphing attack represents a serious and concrete security threat for identity verification-based applications, such as the *Automated Board Control* (ABC) gates at international airports where the facial photo stored in the *electronic Machine Readable Travel Document* (eMRTD) is automatically verified against the live acquired image of the document owner. Therefore, the availability of *Morphing Attack Detection* (MAD) methods [5], *i.e.* systems that are

The associate editor coordinating the review of this manuscript and approving it for publication was Zhe Jin.

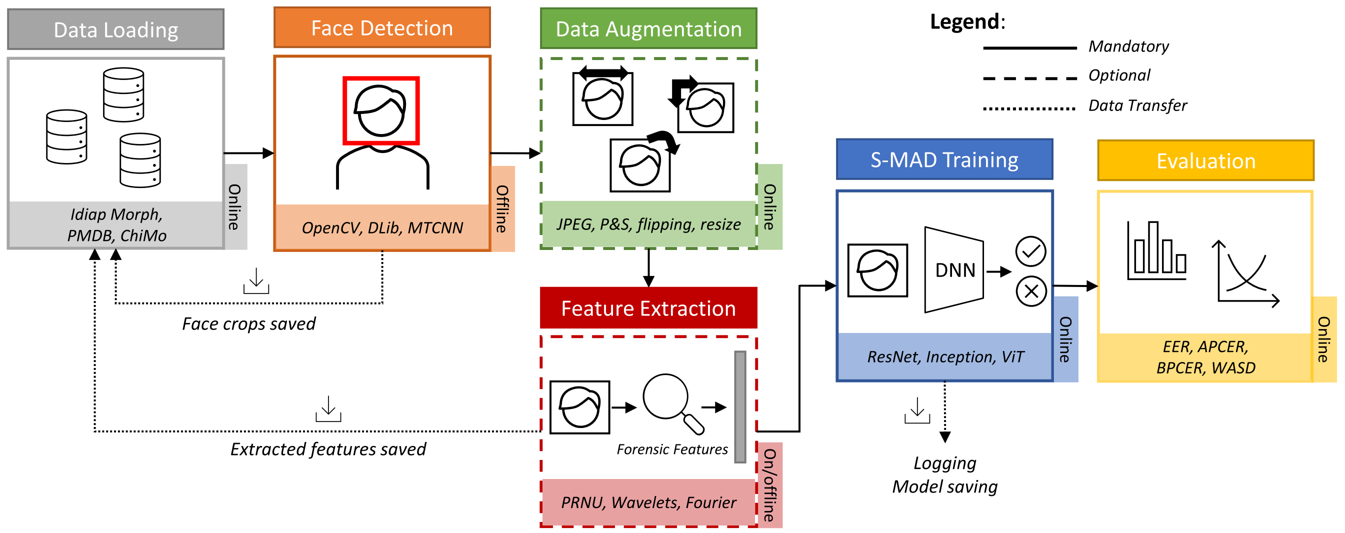


FIGURE 1. Overview of the *Revelio* framework. As depicted, the framework is modular and is mainly based on six different blocks that represent the common steps usually applied in MAD systems. Each block can be run in an online or offline manner and its presence can be either mandatory or optional in the final pipeline. Further details about modules and their functions are reported in Section II.

able to automatically detect the presence of a morphed face in images, is strongly needed by public and private institutions and has raised the interest of researchers belonging to different areas [6]. Unfortunately, despite the relevant number of approaches proposed in the literature in the last years, the accuracy level reached so far is still unsatisfactory for deployment in real-world applications.

Furthermore, a standardized way of training and testing MAD algorithms has yet to emerge in the literature. The research community devoted some efforts to the development of public evaluation platforms for MAD approaches, such as NIST FRVT MORPH [7] or FVC-onGoing [8], [9], where the performance can be objectively assessed by supervised testing on sequestered datasets, *i.e.* data never seen during training and not owned by laboratories and algorithm developers. These benchmarks represent a valuable resource for MAD testing, but it is worth noting that reproducing and comparing published methods — under the same training conditions — is still a hard task, especially for computer vision and deep learning-based solutions. This issue probably originates from the relative novelty of the MAD task, introduced for the first time in [1], which determines two main consequences: i) the lack of publicly available datasets of morphed images on which to train and validate the proposed methods, also due to privacy issues; ii) the lack of publicly available source code of the MAD systems proposed in the literature. In this scenario, each research laboratory or institution usually works on its own data, thus making it difficult to evaluate the impact of the training data on the overall MAD performance [10] and severely limiting the reproducibility of algorithms and results.

We believe that the use of public evaluation benchmarks based on sequestered data, in combination with a shared framework on which publicly released datasets are exploited to develop and train new MAD methods, can improve the quality level of contributions and understanding in the morphing research field.

Relying on these considerations, this paper proposes *Revelio*, a modular framework aimed at providing a shared and effective support for MAD systems development, training, and validation. A general overview of the framework architecture and its modules is depicted in Figure 1. Revelio has been explicitly designed to reduce the efforts needed for the development and comparison of MAD systems, with particular attention to simplifying the usage and integration of new components, defining common protocols, and relying only on publicly available datasets, for both training and validation procedures. With this in mind, the modules of Revelio cover the common steps tackled in the MAD literature: indeed, a MAD system is usually based on an initial face detection phase, that identifies the location of the face in the image, fundamental to provide a facial bounding box with a fixed size and ratio in input to the model; then, optionally, data augmentation can be applied to prevent overfitting; a third step is represented by an optional feature extraction phase, in which hand-crafted or deep learning-based features, also belonging to the forensic research field, are extracted. Additional modules for efficient data loading, model training, and testing with the proper MAD metrics are included to offer useful functions to MAD developers.

Together with this paper, we publicly release the source code and the official documentation[1] of the framework. The released framework already includes several literature algorithms frequently used in MAD system development. Then, interested users can download and install Revelio, to locally

[1]https://miatbiolab.csr.unibo.it/revelio-framework

run experiments defined using YAML configuration files and/or include and test new custom modules.

In order to test the features of Revelio, we conducted an extensive experimental validation to deeply analyze and compare the performance of several deep learning-based MAD solutions, also proving that Revelio allows training state-of-the-art detectors in a straightforward and simple manner. For the sake of reproducibility, all experiments are carried out on publicly released or reproducible datasets, and configuration files are publicly released. We believe that, in this way, this work can be a useful reference for future research works, analysis, such as investigations on the morphing research field and, more in particular, on MAD techniques.

A. PROBLEM STATEMENT

As mentioned above, the face morphing attack, represented in Figure 2, has become a severe security threat in recent years and represents a main challenge for security controls at the borders. Moreover, real cases have been reported starting from 2018.[2] Therefore, the development and adoption of effective Morphing Attack Detection systems is a priority for public and private institutions.

From a general point of view, there are two types of MAD systems developed in the literature, and the difference is mainly related to the number of images received in input [5].

The first category is referred to as Differential Morphing Attack Detection (D-MAD) [11], and receives in input two images, *i.e.* the one contained in the document and a trusted live acquisition. D-MAD methods are usually based on the comparison of the input identities [12] and represent a technology that can be effectively used, for instance, in the ABC gates.

The second category of MADs, on which the Revelio framework is focused, is called Single-image Morphing Attack Detection (S-MAD) and, as the name suggests, decides whether a single image has been manipulated through morphing algorithms: this kind of MAD method is particularly useful during the eMRTD issuing process to verify whether the ID photo provided by the citizen was altered. S-MAD systems are based on the hypothesis that the morphing procedure creates visible or non-visible traces (usually referred to as *artifacts* [13]) in the resulting image. This task is generally considered more challenging than the D-MAD task in the MAD literature and the generalization performance of the developed solutions is usually limited. The difficulty in solving the S-MAD task is also exacerbated by the fact that different morphing algorithms may produce very different results in terms of quality and presence of artifacts, as shown in Figure 3, in which artifacts are visible in the background and foreground areas. Moreover, a sufficiently motivated criminal could quite easily manually retouch the morphed ID photo to improve the overall quality of the image, removing as many artifacts as possible and thus making the correct classification of this type of picture very challenging. Finally, while biometric passports do include a digital copy of the photo ID of the citizen, this is always compressed in order to fit in the limited chip memory, and the photo inside the chip is often a printed and scanned version of the original; these two factors, usually combined, have the effect of drastically reducing the amount of detectable artifacts [14]. This aspect is specifically investigated in the next paragraphs of the paper.

[2]https://pages.nist.gov/ifpc/2020/presentations/26_frvt_morph_ifpc2020_ngan.pdf

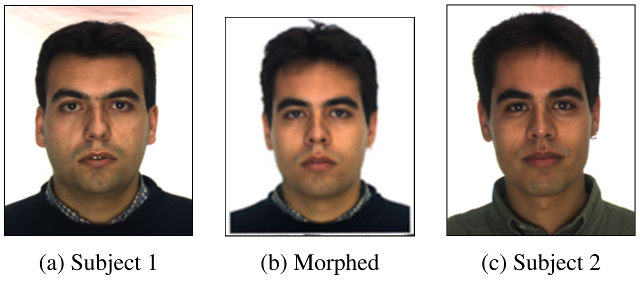


FIGURE 2. Example of the Face Morphing attack. Starting from two contributing subjects (2a and 2c), it is possible to create a hybrid morphed face (2b) hiding the identity of the criminal (for instance, 2c) in the accomplice image (2a), The resulting image is able to fool human examiners and automatic face verification security controls [1], [5].

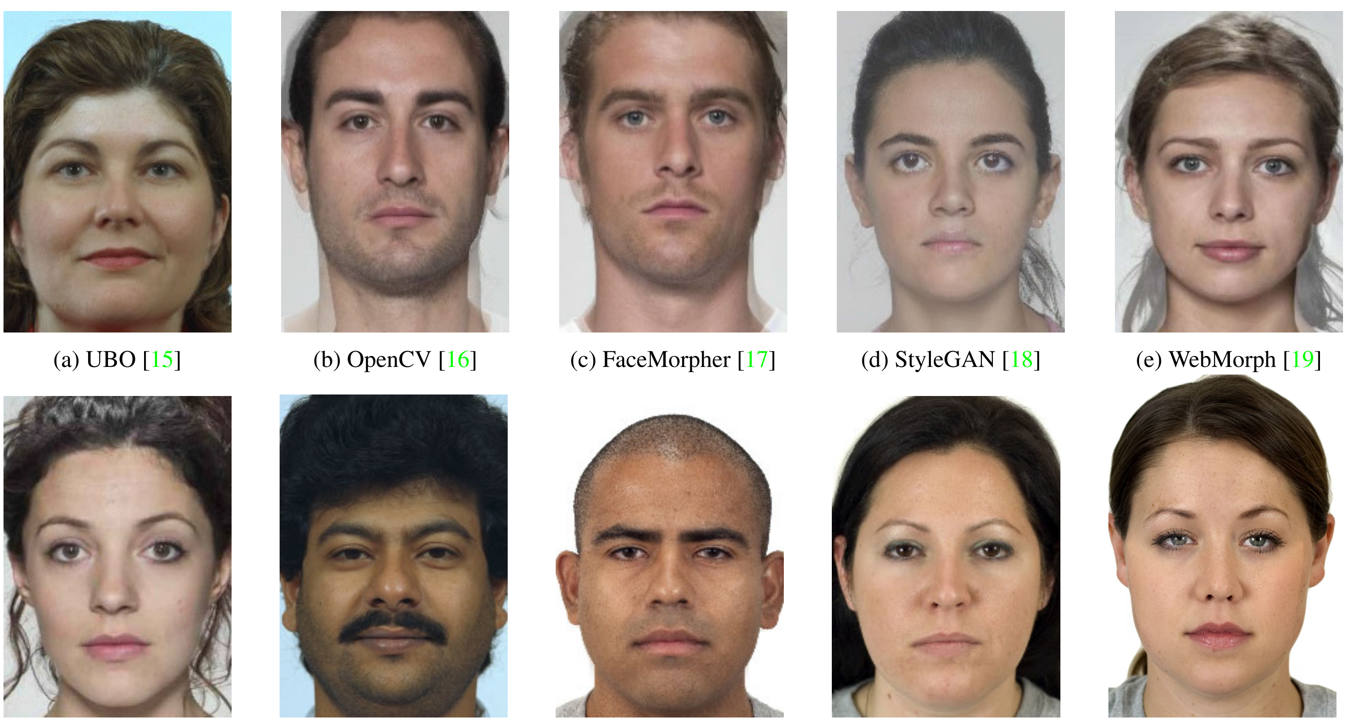
B. CONTRIBUTIONS

The key features of the proposed Revelio framework are the following:

- **Modularity**: we introduce and publicly release *Revelio*, a framework based on different modules explicitly designed to offer a shared development platform, simplifying the integration of new components and methods. It is designed for training and evaluation of new Single-image Morphing Attack Detection (S-MAD) methods.
- **Reproducibility**: the experimental evaluation is carried out only on public datasets or morphed images that can be downloaded or generated through publicly released source datasets and morphing algorithms. Furthermore, we propose the use of a new metric, namely *Weighted Average Error across Datasets* (WAED), to summarize the common error-based metrics exploited in the MAD task across different datasets.
- **Flexibility**: the proposed framework is customizable in a simple and straightforward manner through a human-readable configuration file. New modules and functionalities can be added through an effective plugin system.

Finally, we summarize the findings that belong to the experimental part of this work:

- We carry out a thorough investigation of S-MAD approaches, examining the impact of several factors and challenges that can influence the final model performance, such as face detectors, data augmentation


(a) UBO [15] (b) OpenCV [16] (c) FaceMorpher [17] (d) StyleGAN [18] (e) WebMorph [19]

(f) AMSL [20] (g) Sqirlz Morph [21] (h) FaceFusion [22] (i) UTW [5] (j) NTNU [5]

FIGURE 3. Visual samples of the output of different morphing algorithms available in the literature. As shown, the overall quality of morphed images strongly depends on the type of algorithm exploited and may include, among the others, visible artifacts in the background (*e.g.* OpenCV, FaceMorpher, and WebMorph) or in the face (*e.g.* UBO, AMSL). It is important to note that morphed images produced through the Sqirlz Morph algorithm are manually retouched.

techniques, the use of forensic features, and available training data.
- We prove that is possible to train a simple but effective S-MAD model only on public datasets achieving state-of-the-art results on SOTAMD sequestered datasets through the FVC-onGoing platform [9].

II. REVELIO FRAMEWORK

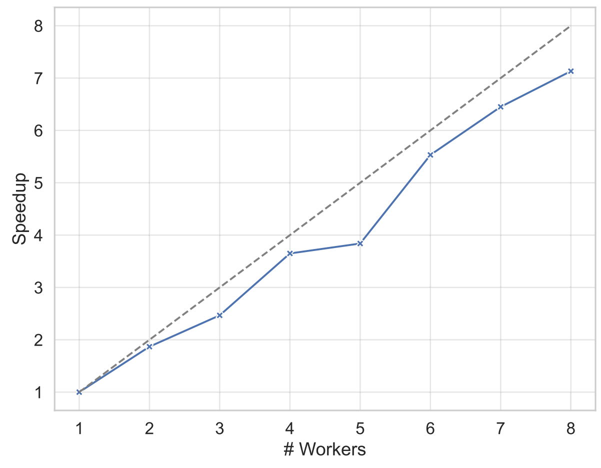
The design of Revelio is based on the conviction that a simple and shared platform is a key element in order to develop better MAD systems. Therefore, the framework is designed to be modular and flexible, while abstracting away most of the complexity typical of Machine and Deep Learning approaches, such as dataset loading, implementation of the data processing pipeline, model training, and finally performance evaluation according to different metrics.

The modular structure of the framework is depicted in Figure 1. All these modules rely on a YAML configuration file through which the user manages and handles the whole framework: its main sections are included in the following paragraphs. Indeed, it is trivial for the end user to swap between different face detectors, change the data augmentation pipeline, or use a new feature extractor: a few lines in the experiment configuration file are all that is needed. In addition, once a seed is specified, the framework is designed to be deterministic: therefore, if all datasets are available, the training and testing of a given model are fully reproducible and comparable — all it takes is sharing the experiment's configuration file with other researchers. Moreover, if the already built-in modules should not be enough, only a limited effort is needed to implement a custom functionality (be it a new data augmentation step, a new feature extractor, or a whole new model) which is then ready to be used in new experiments. This is made possible by the integrated plugin system, which allows to load Python files containing the new modules that can then be invoked by the experiments that require them.

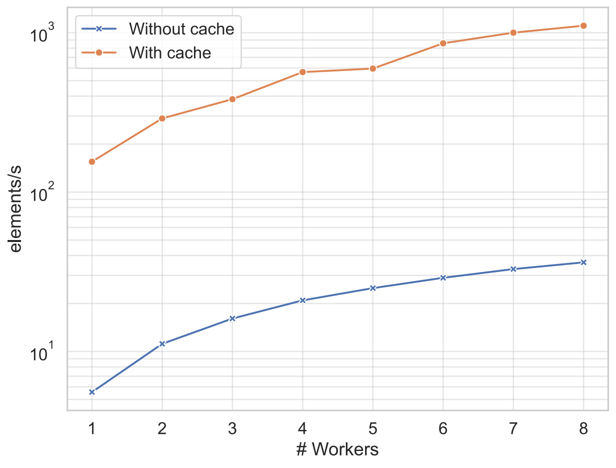
In the following paragraphs, further details for each module are briefly reported and discussed.

A. DATA LOADING

As the name suggests, this module is responsible for loading into memory all the required data, organized in datasets, according to the user-defined specifications in the experiment configuration file. The user can specify one or many datasets to be used for training and testing, and this can be done with a great level of flexibility. Indeed, the minimum information that the user has to specify is the dataset's name, root path, and the random split ratio for training, validation, and test sets, as detailed in Listing 1. As an alternative, a fixed data partitioning can be defined by providing an index file containing a list of the specific data to be loaded



(a) Speedup plot demonstrating the effectiveness of utilizing multiple workers for loading cached data. The dashed grey line represents the ideal linear speedup. As each worker independently processes its own disjoint slice of data, the whole process is embarrassingly parallel and worker synchronization is minimal.



(b) Performance comparison, expressed in elements processed per second, between the use of caching strategies to accelerate data processing and the absence of such techniques. For better visualization, the Y axis is in logarithmic scale. It is worth noting that the performance gain between the two is roughly constant, with a mean speedup factor of 27.4.

FIGURE 4. Plots representing the performance gain of employing multiple workers and caching techniques to accelerate data processing. These two plots combined show that both strategies are effective in order to achieve better performance.

for training, validation, and testing. The code to be used to correctly load a dataset into memory is specified by a dataset loader, which takes the dataset root path as input and returns a list of dataset element descriptors: these simple objects contain only the path to the image and the element's class (either bona fide or morphed). When loading each dataset, the list of all dataset element descriptors is randomly split following the training/validation/test ratios that the user expressed in the configuration file for that dataset. By design, the sum of these ratios can be less than 1 (*i.e.* if the user does not want to load entirely a dataset). After all datasets are loaded, the framework merges together the three subsets, thus obtaining a global training, validation, and test set. Due to the fact that this stage operates on the dataset as a whole, this process cannot be parallelized; on the other hand, the other stages can be significantly sped up by utilizing multiple workers processing disjoint portions of the dataset in parallel, as shown by the speedup plot depicted in Figure 4a. Moreover, as shown in the following sections and in Figure 4b, caching plays a key role in further accelerating data processing. This is made possible by employing PyTorch's `DataLoader` class, which automatically handles parallelization and synchronization of workers, at the cost of increased memory footprint.

```
datasets:
  - name: idiap-facemorpher
    path: /directory/to/IdiapMorphed
    split:
        train: 0.7
        val: 0.1
        test: 0.2
    loader:
        name: IdiapMorphedLoader
        args:
            algorithm: facemorpher
  - ...
```

LISTING 1. Configuration of the *Data Loading* module: among different settings, it is possible to set a specific loader, defining the splits used in training, validation, and testing procedures. For instance, the reported example loads the images created with the FaceMorpher algorithm from the Idiap Morph dataset using a 70% - 10% - 20% split, respectively for training, validation, and testing.

B. FACE DETECTION

The next module of the framework is aimed at localizing the face in each image according to a specific detection algorithm. The output of the face detector is a bounding box $((x_{TL}, y_{TL}), (x_{BR}, y_{BR}))$, which indicates the position of the face inside the image through the use of top-left (TL) and bottom-right (BR) corner coordinates. Furthermore, if the face detector supports it (*e.g.* the DLib [23] face detector), facial landmarks are extracted and embedded into the object representing the dataset element's image.

Since the face detection and landmarks extraction processes can be particularly time-consuming, this stage is carried out offline and its output is stored for each image, so that if the face detector's parameters do not change, its results will be loaded instead of being computed from scratch. In the context of MAD, face detection is not a particularly challenging operation, since input images are typically ISO/ICAO compliant, *i.e.* acquired in a frontal position and controlled conditions. Therefore, while most of the existing face detectors are able to successfully detect the

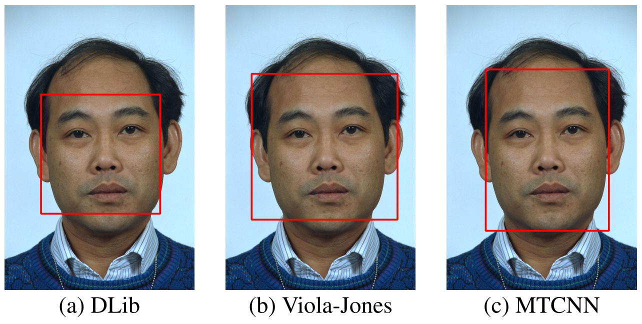


FIGURE 5. Comparison between the bounding boxes generated by three different face detectors, *i.e.* DLib [23], Viola-Jones [24] and MTCNN [25], already included in the Revelio framework. Motivations about this choice are reported in Section II-B.

face, the choice of a specific detector might have an impact on the performance due to the different output (face crop) produced. Three face detectors widely used in biometrics are already available in the framework: DLib [23], Viola-Jones (V&J) [24] and MTCNN [25]. Each face detector produces different bounding boxes, as shown in Figure 5, thus potentially affecting the classification performance of the model.

```
face_detection:
    output_path: /directory/to/face_detection_output
    algorithm:
        name: dlib_detector
        args:
            landmark_predictor_path:
                shape_predictor_68_face_landmarks.dat
```

LISTING 2. Example configuration of the *Face Detection* module, configured to use the DLib face detector with the 68 landmarks model.

Listing 2 shows how the face detector can be chosen in the experiment's configuration file. The face detector's arguments are dependent on the specific algorithm: for instance, the DLib detector allows an optional argument to specify the path of the landmark detector to use, while the MTCNN detector does not have any arguments to set.

C. DATA AUGMENTATION

The following stage is applied only to the training set; validation and test sets will always skip this stage. This module is optional and can be skipped during model training. The augmentation pipeline is composed of multiple sequential steps, specified in the configuration file, each of them associated with a probability of execution on a given input element. Furthermore, some data augmentation steps can be applied only to a specific category of images in the dataset (*e.g.* applying sharpening to the morphed images only). The result is that, during the same training phase, across different epochs, different transformations/parameters can be applied to the same image. Therefore, the benefits of caching would be very limited — the output can continuously change across different epochs — and then the output of this stage is not cached.

From an implementation point of view, the *Revelio* framework has already coded data augmentation procedures regarding the resize and the compression of the input data, in combination with the simulation of the printing and scanning process (P&S) that, assumes particular importance in the MAD task [26], [27]. Indeed, these operations are typical of electronic identity document issuance processes, which often involve the submission of a passport-sized photograph, which is subsequently scanned and compressed to be stored in the document's chip (usually with a storage capacity of 15 kB). A visual example of the output of these operations is reported in Figure 6. Therefore, S-MAD systems — which are usually based on artifact detection techniques — are negatively influenced by these operations, as analyzed and reported in [5] and [27].

Listing 3 shows a minimal example of a data augmentation pipeline composed of two steps: the first one applies the simulated printing and scanning process as described in [27] to approximately half the training elements, while the second one applies a JPEG compression so that each image is under the specified number of bytes while retaining the maximum possible quality. If the probability is not specified, the step is by default applied to all the training elements.

```
augmentation:
    enabled: true
    steps:
      - uses: print_scan
        probability: 0.5
      - uses: jpeg_compression
        probability: 0.5
        args:
            max_bytes: 15000
      - ...
```

LISTING 3. Configuration of the *Data Augmentation* module. As an example, it is reported the Print & Scanned procedure, applied with a probability of 50% on input images, followed by the JPEG compression with a maximum size of 15 kB.

D. FEATURE EXTRACTION

In this module, a feature extractor is used to extract features from input images. A feature extractor can be defined as a pre-trained network, able to extract features related to the training task: this is the case, for instance, of models trained for Face Recognition, which provide features related to the subject's identity. Besides, a feature extractor can be also a mathematical procedure computed on input images: for instance, this is the case in which a Fourier transformation is applied.

Since the framework cannot know in advance how the extracted features will be used, the computed features are inserted as values in a per-image dictionary whose keys are the names of the algorithms used. These features' dictionaries are then made available to the MAD model, which is ultimately responsible for combining and using

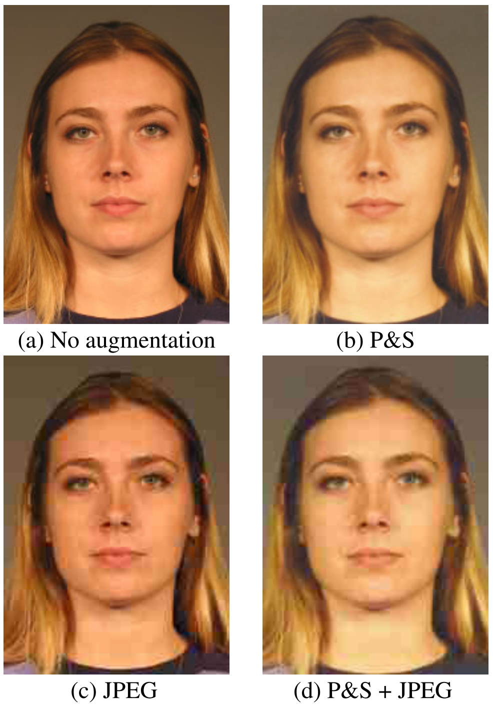


FIGURE 6. **Visual samples of the data augmentation operations implemented and tested through the Revelio framework. Specifically, techniques related to the traditional document issuing procedure are investigated, such as JPEG compression, P&S procedure, and a combination of them.**

the extracted features accordingly. As feature extraction can be rather computationally expensive, this stage can be cached. However, as it is executed after the data augmentation stage (whose output is different across epochs), the feature extraction results cannot be cached if any data augmentation is applied.

Inspired by literature MAD methods that exploit forensic features in their implementation, PRNU [28], [29], [29], Wavelets [30], [31] and Fourier [32] features have been implemented and tested in Revelio.

```
feature_extraction:
    enabled: true
    output_path: /directory/to/
        feature_extraction_output
    algorithms:
      - name: fourier_extractor
      - ...
```

LISTING 4. **Configuration of the *Feature Extraction* module. In this example, the pipeline applies a Fourier transform to extract the magnitude spectrum of all images.**

Listing 4 shows an example of how it is possible to configure one or multiple feature extractor(s) to be applied to every image of each dataset element.

Analogously to the face detection section in the configuration file, each feature extractor has its own set of arguments that can be set. For instance, some feature extractors require a fixed-size image in order to produce a fixed-size feature vector/matrix, so they require both a target width and height to resize all images. Other feature extractors may not have this constraint, so those arguments would not be available.

E. MODEL TRAINING

The next stage of the pipeline is responsible for the training of the MAD model. The configuration file is split into two main sections: model definition and training. In the former, the user must specify which model to adopt for the experiments; in the latter, the user must provide all the information required in order to train the model. The training section contents vary according to the model used, as different models have different training configuration arguments.

While the framework theoretically supports any type of model that can be trained and output predictions as *PyTorch* tensors, particular attention has been dedicated to deep learning-based binary classification models to discriminate between bona fide and morphed images. Once the model is loaded into the specified device (either CPU or GPU, specified as a CLI argument when running Revelio), the training process starts. The user can specify the number of epochs and the batch size, and the framework will automatically split the datasets into batches. The loss function and the optimizer are also specified by the user in the configuration file, and the framework will automatically create the corresponding objects. Revelio has already implemented the most common loss functions and optimizers, such as Binary Cross-Entropy (BCE) loss, Adam, and SGD, but it is also possible to specify custom ones.

In order to have a leaner training loop, some extra features such as checkpoints, early stopping, and experiment logging tools have been implemented as *callbacks*, which are objects that react to certain events inside the training loop. The framework has some callbacks already implemented, such as the one to save the model's weights at the end of each epoch, and the callback which stops the training process if the validation loss does not improve for a certain number of epochs. For the logging, a specific callback logs the various metrics through *Tensorboard*.[3] It is also possible to implement custom callbacks, to extend the functionalities of the framework. In total, there are 10 events that can be captured via callbacks: before/after training, before/after training/validation epoch, and before/after training/validation step.

Finally, training metrics are of prime importance when training a model. Generally, metrics are stateful objects which, after being initialized, are updated after every step by providing two tensors, respectively containing the expected and the predicted scores; as soon as the metric's state is updated, its value can be computed. Revelio comes with several built-in metrics which are widely used in literature [29] when developing MAD systems. Indeed, classification accuracy, *True Positive Rate* (TPR), *True*

[3]www.tensorflow.org/tensorboard

Negative Rate (TNR), *Equal Error Rate* (EER), and BPCER at one or many user-specified APCER (BPCER@APCER) are already present and ready to use, as detailed in the following.

```
experiment:
    batch_size: 64
    model:
        name: feature_inception_resnet
        args:
            pretrained: true
            feature_name: wavelets
            input_depth: 23
    training:
        enabled: true
        args:
            epochs: 50
            optimizer:
                name: SGD
                args:
                    lr: 0.0005
            loss:
                name: BCEWithLogitsLoss
            callbacks:
              - name: Tensorboard
                args:
                    log_dir: /directory/to/logs
              - ...
```

LISTING 5. Configuration of the *Model Training* module dedicated to S-MAD training. As reported, is possible to define a specific model, in combination with all the training details, such as the optimizer and callbacks to log the training details.

Listing 5 shows how the experiment can be configured in Revelio. In the `model` section, the user specifies which model to use and sets any of its custom arguments, that vary according to the chosen model. The `training` section's contents are dependent on the type of model that is used. For instance, if the model is a neural network, there are several arguments to be set. Firstly, the user must specify the number of epochs to train the model with; moreover, an optimizer must be specified and at least its learning rate must be provided; finally, the user must select which loss function should be used. Finally, the user can specify an arbitrary number of callbacks, by specifying their name and potentially their arguments.

F. EVALUATION

Revelio provides a way of defining logical test sets called *testing groups*, and metrics are then reported for each unique testing group, in addition to the whole test set. This way, the user can have separate metrics' values divided by dataset, algorithm, morph level, or any possible combination of these. However, some metrics (*e.g.* EER) cannot be computed if all images of a given group belong to the same class (either bona fide or morphed), so it is essential that each testing group contains at least some bona fide and morphed images.

Finally, as shown in Listing 6, the framework saves the metrics and computed scores (separated by their true label) for each testing group to text files, so that they can be easily inspected by humans. Moreover, the metrics for each testing group can be dumped into a JSON file, so that they are more

```
scores:
    bona_fide: /dir/to/{group}_bona_fide_scores.txt
    morphed: /dir/to/{group}_morphed_scores.txt
    metrics: /dir/to/metrics_scores.json
metrics:
  - name: equal_error_rate
  - name: bpcer_at_apcer
    args:
        thresholds:
          - 0.1
          - 0.05
          - 0.01
```

LISTING 6. Configuration of the *Evaluation* module, in which it is possible to define the metrics output by Revelio framework. In this example it is shown how it is possible to compute the Equal Error Rate (EER) and lowest BPCER related to APCER ≤ 10%, APCER ≤ 5% and APCER ≤ 1%, typical working points for the evaluation of MAD systems.

TABLE 1. A comprehensive list of all built-in algorithms or models already available for each module. In addition, new functionalities can be included through the plugin system, as described in Section II-G.

Module	Built-in algorithms
Data Loading	Idiap Morphed, FRGC, FERET, AMSL, Biometix, Chicago Faces/ChiMo
Face Detection	DLib [23], V&J [24], MTCNN [25]
Augmentation	JPEG compression, P&S simulation [27], random resize, grayscale conversion
Feature Extraction	Fourier transform [32], PRNU [28], Wavelets transform [30]
Models	AlexNet [33], Inception-ResNet V1 [34], MobileNet [35], ResNet family [36], SqueezeNet [37], VGG family [38], Vision Transformer [39]
Metrics	Classification accuracy, BPCER at given APCER, EER, TNR, TPR

```
from revelio.face_detection import FaceDetector

class MyDetector(FaceDetector):
    def __init__(self, cusom_argument):
        # Your initialization logic goes here

    def process_element(self, image):
        # Your processing logic goes here
        return np.array([x1, y1, x2, y2])
```

LISTING 7. Example of a definition of a custom module, in this case a face detector, that can then be imported as a Revelio plugin through the plugin system.

```
face_detection:
    output_path: /directory/to/face_detection_output
    algorithm:
        name: my_detector
        args:
            custom_argument: test
```

LISTING 8. Example of the integration of a new module, *i.e.* the face detector defined in Listing 7, in the framework through the configuration file.

easily accessible to be read and manipulated by automated scripts capable of reading such file format.

G. PLUGIN SYSTEM

The introduced Revelio framework comprises several pre-existing modules, as described above and summarized in Table 1. In order to enhance the flexibility of the framework and accommodate a broader range of modules that are not inherently incorporated, we offer users the capability of creating custom modules using the Python programming language and subsequently incorporating them as plug-ins. Once integrated, these plugins become discoverable components within the configuration file, simplifying the expansion of functionality without necessitating modifications to the source code of the framework, as illustrated in Listings 7 and 8. This particular feature proves to be particularly advantageous when users necessitate the development of custom models. Nevertheless, it is imperative to emphasize that the user must explicitly opt-in to leverage plugins, given that this entails the execution of arbitrary code and, consequently, carries the potential to compromise the security of the user's computational environment.

III. EXPERIMENTS

In this section, we define the training and testing protocols used in our experimental evaluation, describing the datasets used and the training and testing split. In particular, we aim to clearly define a common protocol for future MAD proposals, seizing the opportunity to propose a comprehensive empirical comparison of different MAD approaches available in the literature.

A. DATASETS

All datasets exploited in the Revelio framework, and therefore in the following experimental evaluation, are publicly available or can be generated by applying public morphing algorithms on face images contained in the original public datasets. In addition, we release[4] the subject pairs used to create the morphed images and the list of the data exploited for training, in order to further improve the clarity and the reproducibility of the obtained results.

As a general overview, some statistics about each dataset and the experimental protocol considered in the Revelio framework are reported in Table 2 and Table 3, some visual samples are depicted in Figure 3, while a detailed analysis of their composition and data sources is reported in the following.

- *Progressive Morphing Database* (PMDB) [44]: 1108 morphed images are created starting from three well-known datasets, *i.e.* AR [41], FRGC [40] and Color Feret [42], through the public morphing algorithm described in [44]. To generate the morphed images, we used a total of 280 subjects, split into 134 males and 146 females. It is important to note that on PMDB no manual retouching procedures are applied on morphed images in order to enhance the visual quality; therefore, the images may contain artifacts (such as blurred areas or ghosts). The background is automatically replaced by the morphing algorithm, then it does not include any artifacts.
- *Idiap Morph* [45], [46]: it is a publicly available set of several datasets, specifically consisting of five subsets created with different morphing algorithms (OpenCV [16], FaceMorpher [17], StyleGAN [18], WebMorph [19] and AMSL [20]), exploiting the face images belonging to the Feret [42], FRGC [40] and Face Research Lab London Set [19] (in this paper referred as FRLL) datasets as input data. As depicted in Figure 3, the overall visual quality of morphed images created with OpenCV and FaceMorpher is negatively influenced by the heavy presence of artifacts, located both in the background and foreground (*i.e.* the face) of images. In morphed faces generated with the StyleGAN-based approach visual artifacts are less visible, but common GAN-related textures are still present and visible [47]. AMSL morphing algorithm is exploited to produce 2175 morphed images starting from 102 adult faces, with a morphing factor equal to 0.5. The interesting feature of this morphing algorithm is represented by the compression procedure applied to all images, to fit on the single chip of the eMRTD available in official documents: therefore, available images have a maximum size of 15 kB. We observe that the compression procedure tends to make the S-MAD task more challenging since it deletes most of the artifacts eventually introduced by the morphing algorithm.
- *MorphDB* [44]: this dataset, built using images belonging to the Color Feret [42] and FRGC [40] datasets, consists of 100 morphed images created starting from 50 males and 50 female subjects using the *Sqirlz Morph 2.1* [21] algorithm. Unfortunately, this dataset is not publicly released, but it can be found on the FVC-OnGoing platform to be used as a test dataset as in the Revelio framework. Despite this issue, it represents an interesting testing dataset, since all morphed images have been manually retouched, and the final visual quality is excellent. This dataset contains also a set consisting of real Printed-and-Scanned (P&S) images, *i.e.* bonafide and morphed images that have been realistically printed and scanned with professional tools.
- *ChiMo*: this dataset has been generated using the images (with neutral expression) of the *Chicago Faces Database* (CFD) [43] which includes images of 831 subjects of varying ethnicities. For each subject, five other subjects of the same ethnicity and gender have been selected for morphing; in order to maximize the attack potential of the morphed images (*i.e.* the probability of fooling FRSs), the average face verification scores

[4]https://miatbiolab.csr.unibo.it/revelio-framework

TABLE 2. Morphing algorithms and datasets in the *Revelio* framework used for the experimental evaluation. For each morphing algorithm, is reported the related dataset name and the original source of the images used for the morphing procedure. Then, the number of morphed images for every data source is shown, in combination with the percentage of images available during the training, validation, and testing phases. The last column reports the quality of morphed images, as discussed in Section III-A.

Morphing Algorithm	Dataset	Data Source	#Morphed	Train/Val (%)	Test (%)	Quality
UBO [15]	PMDB	AR - FRGC - Feret	711 - 199 - 198	70+10	20	Medium
OpenCV [16]	Idiap morph	FRGC - FRLL - Feret	964 - 1222 - 529	70+10	20	Low
FaceMorpher [17]	Idiap morph	FRGC - FRLL - Feret	964 - 1222 - 529	70+10	20	Low
StyleGAN [18]	Idiap morph	FRGC - FRLL - Feret	964 - 1222 - 529	70+10	20	Low
WebMorph [19]	Idiap morph	FRLL	1221	0	100	Low
AMSL [20]	Idiap morph	FRLL	2175	0	100	Low
Sqirlz Morph [21]	MorphDB$_D$	FRGC - Feret	50 - 50	0	100	High
Sqirlz Morph [21]	MorphDB$_{P\&S}$	FRGC - Feret	50 - 50	0	100	High
FaceFusion [22]	ChiMo	CFD	8310	0	100	Medium
UTW [5]	ChiMo	CFD	8310	0	100	Medium
NTNU [5]	ChiMo	CFD	8310	0	100	Medium

TABLE 3. Analysis of the amount of morphed and bonafide images in relation to each source dataset (see Table 2). As shown, the morphed images represent the large majority of available data during the training and testing phases.

Data Source	#Morphed	#Bonafide	Notes
FRGC [40]	3092	2581	50 P&S
AR [41]	711	1422	-
Feret [42]	1985	2720	50 P&S
FRLL [19]	7062	92	-
CFD [43]	24930	831	-

of three commercial SDKs (*VeriLook*,[5] *Cognitec*[6] and *Innovatrics*[7]) have been used to select the most similar subjects for each individual. Then, two morphing factors (0.3 and 0.5) and three different morphing algorithms (FaceFusion [22], UTW [5] and NTNU [5]) are applied for each subject pair, thus resulting in 24390 morphed images (8310 for each algorithm). Finally, two versions of this dataset are created: the first one contains the digital images as produced through the morphing procedure, while in the second (here referred to with the subscript *JPG*) we applied a compression procedure similar to the one applied on AMSL, thus obtaining images with a maximum size of 15 kB.

B. EXPERIMENTAL PROTOCOL

In Revelio, and in all the following experiments and tables, we group the train and test datasets relying on the morphing algorithm used to produce morphed images, as also reported in the first column of Table 2. We believe this data organization is useful to analyze the MAD performance in relation to different morphing algorithms that represent a key element in the development of MAD techniques [3]. Then, different datasets can be grouped in the same set; for instance, the StyleGAN-based [18] morphing algorithm groups the subsets belonging to the Idiap Morph built on three different data sources (FRGC, FRLL, and Color Feret).

All the experiments have been carried out following the dataset organization and training/testing protocols described in Table 2. Specifically, we create a challenging setup following these considerations:

- Morphing algorithms used to produce images in training and testing splits are different, as reported in Table 2; more precisely, we create one validation and one test set: the first one is a subset (20% of the images) taken from the same datasets used for model training so that the morphing algorithms coincide with those in the training set, while the second one, on which the WAED metric is computed (see Section III-C1), contains all the morphed images generated with unseen morphing algorithms.
- The training datasets contain morphed images with low visual quality due to, for instance, the presence of artifacts, as shown in Figure 3, while the test set only includes medium or high-quality morphed images, due to the human intervention in retouching procedures (MorphDB) or the absence of visible artifacts in the backgrounds (ChiMo).

It is worth noting that this setting assures a demanding cross-morphing algorithm evaluation, aiming to verify the generalization capabilities of the investigated MAD methods. Besides, all the images taken from the Chicago Face Dataset belong to subjects never seen during the training procedure.

Table 3 reports, for each public face dataset, the number of bona fide images considered in the experiments, and the total number of morphed images derived from that dataset. It is important to note the unbalanced amount of bona fide and morphed images, which contributes in making challenging the proposed setting.

C. METRICS

In order to evaluate and compare the investigated MAD methods, we use the metrics commonly used for performance assessment in the context of morphing detection [3]: the *Bona Fide Presentation Classification Error Rate* (in short, BPCER), representing the proportion of bona fide images

[5]www.neurotechnology.com/verilook.html
[6]www.cognitec.com
[7]www.innovatrics.com/

TABLE 4. Weights used for the proposed WAED metric (see Section III-C) to balance the contribution of each metric (w_E) compute on several datasets (w_D). Subscript [*JPG*] denotes both versions of the dataset, with digital and compressed images. The underlying idea is to produce in output a single numeric value that can support the evaluation of the tested MAD approach, taking into account the peculiarities of the different test datasets exploited.

Metric weights (w_E)				Dataset weight (w_D)					
EER	**B**$_{0.1}$	**B**$_{0.05}$	**B**$_{0.01}$	**FaceFusion**$_{[JPG]}$	**NTNU**$_{[JPG]}$	**UTW**$_{[JPG]}$	**Webmorph**	**Squirlz**	**AMSL**
.30	.10	.20	.40	1.00	.94	.88	.80	.78	.77

wrongly classified as morphed, and the *Attack Presentation Classification Error Rate* (APCER), indicating the proportion of morphed images wrongly accepted as bona fide; these two indicators are mathematically detailed as follows:

$$\text{BPCER}(\tau) = \frac{1}{N}\sum_{i=1}^{N} H(b_i - \tau) \tag{1}$$

$$\text{APCER}(\tau) = 1 - \left[\frac{1}{M}\sum_{i=1}^{M} H(m_i - \tau)\right] \tag{2}$$

in which τ is the score threshold on which b_i, m_i, the detection scores, are compared; $H(x) = \{1 \text{ if } x > 0,\ 0 \text{ otherwise}\}$ is defined as a step function. In addition, we measure the BPCER with respect to a defined value of APCER, *i.e.* $B_{0.1}$, $B_{0.05}$ and $B_{0.01}$, representing the lowest BPCER with APCER $\leq 10\%$, $\leq 5\%$ and APCER $\leq 1\%$, respectively. We also show the *Detection Error Trade-off* (DET) curves to facilitate the comparison between different approaches.

1) WAED METRIC

The results reported on different datasets, using several performance indicators, can be sometimes dispersive and difficult to analyze as a whole. To summarize and simplify the comparison of diverse approaches across different testing datasets, we introduce therefore a new metric, namely *Weighted Average Error across Datasets* (in short, WAED), that aims to condense the aforementioned set of error metrics $\mathcal{E}$ computed on different testing datasets $\mathcal{D}$ into a single value:

$$\text{WAED} = \sum_{E\in\mathcal{E}}\sum_{D\in\mathcal{D}} w_D w_E E(D) \tag{3}$$

where:

- $E(D)$ is the value of the error indicator $E \in \mathcal{E}$, measured on the dataset $D \in \mathcal{D}$;
- w_E is a weighting factor assigned to each error indicator in order to focus our attention on the error indicators which are more relevant for a real-world scenario (*e.g.* $B_{0.01}$). The weights considered for the WAED metric computation (shown in the left part of Table 4) are chosen by assigning the majority of the weight to the most common real-world operating point (*i.e.* $B_{0.01}$), followed by the EER, as it is useful for evaluating the performance of the system at a glance, and finally the other two chosen operating points (*i.e.* $B_{0.05}$ and $B_{0.1}$);
- w_D is a dataset weight which somehow measures the dataset complexity, quantified in our experiments by the similarity of the morphed images to the two contributing subjects. In particular, for each dataset $D \in \mathcal{D}$, we compute the value s_D, through the comparison of each morphed image $m_i \in D$ with the S bona fide images $b_{i,j}$ used in the morphing process. The comparison is done on K different commercial face verification SDKs, as follows:

$$s_D = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{S}\sum_{j=1}^{S}\frac{1}{K}\sum_{k=1}^{K}\frac{\max(0, s_k(m_i, b_{i,j}) - thr_k)}{thr_k} \tag{4}$$

where $M = |D|$, $S = 2$ since we tackle images produced through two-subjects morphing algorithms, and $K = 3$ since we exploit Verilook, Cognitec, and Innovatrics SDKs, respectively. To make comparable the scores of different SDKs, the similarity score $s_k(m_i, b_{i,j})$ provided by each SDK is normalized according to the FAR_{1000} threshold (thr_k) provided by the SDK.
Finally, the single dataset scores are normalized in the range [0, 1] as follows:

$$w_D = \frac{s_D}{\max\limits_{D\in\mathcal{D}} s_D} \tag{5}$$

The resulting weights for the testing datasets are reported in Table 4.

The proposed metric, reported in our results at the bottom of the tables, produces a single numeric value in the range [0, 1] with which comparisons are simplified: being an overall error measure, low values are desirable.

D. EXPERIMENTAL RESULTS

As previously mentioned, experimental results are reported grouped by morphing algorithms, and then a single group can refer to more than one dataset. Associations between the original dataset and the morphing algorithms are reported in Table 2.

In all the following experiments, all the networks are pre-trained on the ImageNet dataset [48] (weights downloaded from the official PyTorch[8] storage), and trained using the binary cross-entropy loss function. As an optimizer, we use the *Stochastic Gradient Descent* (SGD), with a learning rate in the range of $\left[10^{-3}, 5\cdot 10^{-3}\right]$ and early-stopping (with patience of 5 epochs and a minimum

[8]https://pytorch.org

TABLE 5. Morphing detection scores across different Face Detectors given a fixed ResNet-50 [36] detector. Results are reported in terms of Equal Error Rate (EER), the lowest BPCER related to APCER ≤ 10%, ≤ 5% and ≤ 1%, respectively. The proposed WAED metric summarizes performance (lower is better) across listed testing datasets (see Section III-C).

Morphing Alg.	DLib [23]				V&J [24]				MTCNN [25]			
	EER	$B_{0.1}$	$B_{0.05}$	$B_{0.01}$	EER	$B_{0.1}$	$B_{0.05}$	$B_{0.01}$	EER	$B_{0.1}$	$B_{0.05}$	$B_{0.01}$
UBO	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	.014	**.000**	**.000**	.014
OpenCV	**.001**	**.000**	**.000**	**.001**	.005	**.000**	**.000**	.002	.018	.002	.006	.035
FaceMorpher	**.002**	**.000**	**.000**	**.000**	.002	**.000**	**.000**	.002	.007	.001	.002	.006
StyleGAN	**.001**	**.000**	**.000**	**.000**	.003	**.000**	**.000**	.001	.018	.002	.004	.021
AMSL	.302	.650	.700	.950	.300	.700	.750	.900	**.112**	**.150**	**.300**	**.650**
Webmorph	.400	**.700**	.900	1.000	.400	.750	**.750**	.900	**.362**	.800	.800	**.850**
Sqirlz Morph$_D$	.041	.012	.033	.122	**.032**	**.000**	**.008**	.081	**.032**	.021	.033	**.062**
FaceFusion	**.255**	.656	.803	.945	.277	**.598**	**.732**	**.898**	.300	.759	.876	.970
NTNU	.215	.680	.854	.978	.249	.651	.826	.966	**.182**	**.489**	**.752**	**.954**
UTW	.538	.912	.959	.992	.516	.899	.946	.996	**.300**	**.567**	**.714**	**.878**
FaceFusion$_{JPG}$	.445	.832	.922	.981	.450	.859	.924	.983	**.270**	**.503**	**.663**	**.886**
NTNU$_{JPG}$	.491	.901	.946	.990	.471	.894	.948	.987	**.327**	**.659**	**.786**	**.931**
UTW$_{JPG}$	.491	.865	.929	.982	.508	.878	.929	.983	**.363**	**.709**	**.806**	**.931**
WAED ↓	.6944				.6800				**.5831**			

improvement of 10^{-3}) to prevent overfitting computed on the validation set. All the configuration settings and trained models are publicly released.

1) INVESTIGATION ON FACE DETECTORS

Several robust face detection techniques are available in the literature and this first set of experiments aims to compare the most promising ones and to evaluate their impact on S-MAD performance, being aware that in this application scenario face detection is quite a simple task, since all input images are fully ISO/ICAO compliant [49], with natural expression, acquired in constrained (frontal) pose and lighting conditions, etc. Then, we focus on testing three different face detectors widely used in the literature, in particular in MAD methods, based on Machine and Deep Learning techniques: DLib [23], Haar cascades-based [24] (here referred to as V&J) and MTCNN [25]. Experiments are carried out in combination with a *ResNet-50* [36] architecture, whose effectiveness has been widely documented in the literature for several classification tasks, including MAD [11], [44]. The results reported in Table 5 suggest that the MTCNN face detector leads to the best accuracy, while DLib and V&J have similar lower values. As depicted in Figure 5 the face crop provided by MTCNN detector includes a wider facial area and then tends to include facial parts (chin and outline) in which the morphing procedure usually leaves artifacts. As mentioned, the choice of the best algorithm and the computation of the WAED metric is based on the results obtained on the second group of testing datasets, in which morphed images have been created with morphing algorithms different from the ones used for the training images.

2) INVESTIGATION ON DNN ARCHITECTURES

In the second part of the experiments, we aim to define the best architecture to tackle the morphing classification task. In [27], authors proposed to exploit well-known deep architectures, ranging from *AlexNet* [33] to *VGG-face* [38], to address the S-MAD task. Reported results seem to suggest that a deep learning approach can achieve high accuracy, provided that a certain amount of representative training data is available for model training. This work lead us to select three different deep learning-based architectures already proposed in the literature, *i.e. ResNet-50* [36] (the same used in the evaluation of Section III-D1), *Inception-Resnet V1* [34] and the recent *Vision Transformer* (ViT) [39]. The architecture choice is due to ResNet-50, as mentioned before, revealing high accuracy in several classification tasks, while Inception-Resnet V1 has been effectively used in [30] for the S-MAD task. Differently, the ViT model is an architecture recently proposed in the literature, that seems to be able to overcome the performance of traditional Convolutional Neural Networks (CNNs) for image classification [50]. We internally tested also other architectures obtaining lower results, here not reported for simplicity. The experimental results reported in Table 6 show that the Inception-Resnet outperforms the other architectures by a clear margin, with equal training and testing data, thus confirming the findings reported in [30]. Presumably, the presence of kernels with different sizes at the same level of the network enhances the ability of the model to detect specific patterns on pixel values, and then morphed images. Therefore, all the following experiments are performed using the Inception-Resnet model.

3) INVESTIGATION ON DATA AUGMENTATION

Following the considerations reported in [27], we analyze the impact of different data augmentation techniques on the final classification accuracy. Data augmentation techniques play a crucial role in many different classification tasks, increasing the amount and the variety of images available for model training. The context of face morphing is, in some respects, different from other applications since the morphing process

TABLE 6. Morphing detection scores across different architectures given a fixed Face Detector (MTCNN). Results are reported in terms of Equal Error Rate (EER), the lowest BPCER related to APCER ≤ 10%, ≤ 5% and ≤ 1%, respectively. The proposed WAED metric summarizes performance (lower is better) across listed testing datasets (see Section III-C).

Morphing Alg.	**ResNet-50** [36]				**Inception-Resnet** [34]				**Vision Transformer** [39]			
	EER	**B**$_{0.1}$	**B**$_{0.05}$	**B**$_{0.01}$	**EER**	**B**$_{0.1}$	**B**$_{0.05}$	**B**$_{0.01}$	**EER**	**B**$_{0.1}$	**B**$_{0.05}$	**B**$_{0.01}$
UBO	.014	**.000**	**.000**	.014	**.003**	**.000**	**.000**	**.000**	.006	**.000**	**.000**	.002
OpenCV	.017	.002	.006	.022	**.008**	**.000**	**.000**	**.006**	.014	**.000**	.004	.039
FaceMorpher	.006	.001	.002	.005	**.004**	**.000**	**.000**	**.003**	.006	**.000**	**.000**	.005
StyleGAN	.017	.002	.003	.018	.013	**.000**	**.000**	.021	**.009**	**.000**	.002	**.009**
AMSL	.112	.150	.300	.650	**.005**	**.000**	**.000**	**.000**	.150	.250	.300	.350
Webmorph	.362	.800	.800	**.850**	**.250**	**.350**	**.650**	.900	.300	.400	**.650**	**.850**
Sqirlz Morph$_D$	.032	.021	.033	**.062**	**.024**	**.012**	**.025**	.123	.054	.015	.058	.118
FaceFusion	.300	.759	.876	.970	**.114**	**.136**	**.243**	**.502**	.240	.440	.579	.829
NTNU	.182	.489	.752	.954	**.114**	**.132**	**.261**	**.519**	.228	.448	.598	.897
UTW	**.300**	**.567**	**.714**	**.878**	.312	.641	.769	.942	.439	.750	.836	.951
FaceFusion$_{JPG}$	.270	.503	.663	.886	**.125**	**.165**	**.351**	**.669**	.186	.278	.391	.679
NTNU$_{JPG}$	.327	.659	.786	.931	**.158**	**.265**	**.448**	**.703**	.223	.369	.507	.769
UTW$_{JPG}$	.363	.709	.806	.931	**.315**	**.643**	**.753**	**.918**	.324	.709	.826	.959
WAED ↓	.5831				**.3915**				.5103			

TABLE 7. Morphing detection scores across different Data Augmentation techniques, given a fixed architecture (Inception Resnet) and a Face Detector (MTCNN). Results are reported in terms of Equal Error Rate (EER), the lowest BPCER related to APCER ≤ 10%, ≤ 5% and ≤ 1%, respectively. The proposed WAED metric summarizes performance (lower is better) across listed testing datasets (see Section III-C).

Morphing Alg.	**No augmentation**				**JPEG** [49]				**Print & Scan** [27]			
	EER	**B**$_{0.1}$	**B**$_{0.05}$	**B**$_{0.01}$	**EER**	**B**$_{0.1}$	**B**$_{0.05}$	**B**$_{0.01}$	**EER**	**B**$_{0.1}$	**B**$_{0.05}$	**B**$_{0.01}$
UBO	.003	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**
OpenCV	**.008**	**.000**	**.000**	.006	.009	**.000**	**.000**	.005	**.008**	**.000**	**.000**	**.003**
FaceMorpher	.004	**.000**	**.000**	.003	**.004**	**.000**	**.000**	**.000**	.011	**.000**	**.000**	.012
StyleGAN	.013	**.000**	**.000**	.021	.011	**.000**	**.000**	.011	**.008**	**.000**	**.000**	**.008**
AMSL	.005	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	**.000**	.050	.050	.050	.300
Webmorph	.250	.350	.650	.900	**.158**	**.200**	**.250**	**.750**	.315	.500	.500	**.750**
Sqirlz Morph$_D$	.024	.012	.025	.123	**.001**	**.000**	**.000**	**.001**	.069	.019	.118	.192
FaceFusion	.114	.136	.243	.502	**.094**	**.088**	**.176**	**.408**	.129	.178	.341	.671
NTNU	.114	.132	.261	.519	**.091**	**.081**	**.176**	**.469**	.167	.313	.501	.787
UTW	**.312**	.641	.769	.942	.391	.740	.850	.964	.329	**.621**	**.741**	**.884**
FaceFusion$_C$	.125	.165	.351	.669	**.114**	**.134**	**.283**	**.633**	.162	.255	.434	.735
NTNU$_C$	.158	.265	.448	.703	**.149**	**.239**	**.404**	**.679**	.203	.412	.579	.829
UTW$_C$	.315	.643	.753	**.918**	**.298**	**.604**	**.736**	**.918**	.300	.622	.752	.940
WAED ↓	.3915				**.3515**				.4580			
Sqirlz Morph$_{P\&S}$	.252	.455	.540	.760	**.197**	**.320**	**.385**	**.520**	.218	.420	.505	.710
Sqirlz Morph$_{P\&S+JP2}$	.278	.495	.615	.855	.237	**.355**	.555	.780	**.210**	.395	**.475**	**.610**
WAED$_{P\&S}$ **↓**	.5655				**.4530**				.4669			

leaves only labile traces and the risk of weakening such details by applying transformations to the original images is concrete.

Then, we evaluate here different techniques for data augmentation: some of them are the typical approaches used in the literature. In particular, we evaluate here image resizing which is generally required for model training since the large majority of neural networks receive input images with a fixed spatial resolution; the tests are aimed at evaluating the impact of the resizing algorithm used (*i.e.* the interpolation algorithm) on the final accuracy of the model. We also evaluate other transformations specific to this application scenario, defined taking into account the typical pipeline of the document issuing process. In many countries, in fact, the digital photo acquired by professional photographers is printed on paper and then scanned to be included in the document during the eMRTD issuing process. Moreover, when stored into the chip, the image is compressed, either using the JPEG Sequential Baseline (ISO/IEC 10918-1) mode of operation, or the JPEG-2000 Part-1 Code Stream Format (ISO/IEC 15444-1) [49]. Considering the minimum image size requirement of 11 kB given in [54], most of the issuing authorities adopt a compressed image size of around 12-15 kB; we follow here the approach adopted in [5] setting the maximum size of the compressed photo to 15 kB. As to the printing and scanning process, we apply here the simulation approach introduced and described in [27]. The MAD results obtained using different data augmentation techniques are reported in Table 7. The first column represents the baseline, where no data augmentation is applied, the second column

TABLE 8. Morphing detection scores across different forensic features used in combination with the inception Resnet architecture and the MTCNN Face Detection. Results are reported in terms of Equal Error Rate (EER), the lowest BPCER related to APCER ≤ 10%, ≤ 5% and ≤ 1%, respectively. The proposed WAED metric summarizes performance (lower is better) across listed testing datasets (see Section III-C).

Morphing Alg.	Fourier [51]				Wavelets [52]				PRNU [53]			
	EER	B$_{0.1}$	B$_{0.05}$	B$_{0.01}$	EER	B$_{0.1}$	B$_{0.05}$	B$_{0.01}$	EER	B$_{0.1}$	B$_{0.05}$	B$_{0.01}$
UBO	.112	.117	.235	.318	**.021**	**.000**	**.000**	**.063**	.125	.178	.244	.481
OpenCV	.083	.055	.156	.430	**.048**	**.022**	**.046**	**.186**	.066	.041	.083	.199
FaceMorpher	.052	.022	.056	.095	**.021**	**.004**	**.010**	**.042**	.077	.063	.106	.241
StyleGAN	.060	.037	.074	.269	.051	.025	.052	.170	**.023**	**.003**	**.013**	**.053**
AMSL	.403	.850	.850	.950	**.200**	**.250**	**.450**	**.600**	.356	.800	.850	.950
Webmorph	.500	.950	.950	1.000	**.411**	**.600**	**.650**	**.900**	.550	.900	.950	1.000
Sqirlz Morph$_D$	.163	.292	.402	.763	**.071**	**.070**	**.149**	**.356**	.260	.442	.550	.639
FaceFusion	.291	.613	.786	.922	**.262**	**.515**	**.680**	**.915**	.485	.925	.965	.989
NTNU	**.184**	.338	.507	.794	.191	**.327**	**.490**	**.761**	.246	.587	.813	.978
UTW	.879	1.000	1.000	1.000	.833	.998	1.000	1.000	**.159**	**.253**	**.410**	**.851**
FaceFusion$_{JPG}$	.446	.845	.922	.983	**.188**	**.325**	**.480**	**.727**	.372	.700	.832	.955
NTNU$_{JPG}$	.558	.927	.970	.993	**.239**	**.451**	**.585**	**.810**	.426	.835	.923	.986
UTW$_{JPG}$	.378	.757	.848	.976	.392	.774	.863	.963	**.368**	**.727**	**.846**	**.958**
WAED ↓	.7345				**.5768**				.7075			

contains the results of a JPEG compression with a probability of 50% on input images and the third column represents the performance obtained by applying the printing and scanning simulation with a probability of 50% on input images. The results (and the corresponding WAED metric) are reported separately for the testing datasets used in the previous tables and for the P&S ones (not used in the previous experiments). Mainly guided by the findings in [27], all augmented models were obtained by fine-tuning the baseline model, rather than training from scratch. As expected, data augmentation has a slight but noticeable effect on the performance of the model with respect to digital images. JPEG compression seems to produce in general a positive effect even on non-compressed and printed/scanned datasets. The simulation of the P&S process is expected to produce positive effects on the P&S datasets and the results prove that in this case the model trained using the simulation of the printing and scanning process performs better than the model without this kind of augmentation; however, the advantages with respect to the model trained with JPG compression augmentation are quite limited. As to this aspect, we believe that the effectiveness of the simulation might be improved by an optimization of its parameters that should be tuned to better represent the real P&S process.

Finally, we internally test the investigated MAD also considering different color spaces in input, following the findings reported in [55] that highlight that the use of color spaces other than RGB might have a positive impact on MAD performance. Then, we convert all training and test images in grayscale, HLS, and YCbCr color spaces: we omit to report the related Table since results reveal that the RGB representation offers the best performance, and indeed the color information positively contributes to the detection of morphed images. With grayscale images, we obtain WAED = 0.3824, with HSL WAED = 0.5677 and with YCbCr WAED = 0.4360. We also tested a single channel in input, obtaining similar results (WAED = 0.4411 using only the L channel of HSL, WAED = 0.7453 using the Y channel of YCbCr color space).

4) INVESTIGATION ON FORENSIC FEATURES

The use of forensic features has received increasing attention not only in fake face image detection (the so-called *DeepFakes* [56]), but also in the MAD field [57]. Indeed, we implement in our framework a selection of the most used forensic features in the MAD task available in the literature.

As reported in [51], the *Fourier* transform can be effectively exploited to detect fake facial images; in [58] this feature is used to detect morphed images and then is implemented and tested in the Revelio framework.

Following the considerations reported in [29], the second investigation regards the use of the *Photo Response Non Uniformity* (PRNU) [53], *i.e.* the unique pattern noise related to a specific digital sensor used to acquire image or video frames. The underlying idea is that the morphing procedure can affect the uniformity of the sensor noise, and then its analysis can help to spot morphed images.

Thirdly, our experiments aim to investigate the use of *wavelets* [52], since in [30] an approach based on an attention-aware neural network that receives in input this kind of feature is presented, obtaining an interesting accuracy on the NIST FRVT MORPH [7] platform. We implement this approach to the best of our knowledge,[9] following two different approaches. In both experiments, following the paper [59], we apply three-level undecimated 2D wavelet decomposition, using *Daubechies* 4 (db4) as the mother wavelet; in the first implementation a one-level wavelet decomposition is applied, while in the second one we applied a three-level decomposition and we finally exploited a selection of 23 sub-bands channel-wise stacked. The first

[9]The original paper is currently patent pending, and then a limited amount of implementation details are revealed.

TABLE 9. Morphing detection scores across different training sets given a fixed model (Inception-Resnet V1) and face detector (MTCNN). Results are reported in terms of Equal Error Rate (EER), the lowest BPCER related to APCER ≤ 5% and ≤ 1%, respectively. The proposed WAED metric summarizes performance (lower is better) across listed testing datasets (see Section III-C). Due to space reasons, neither the value of the lowest BPCER related to APCER ≤ 10%, nor the column containing the results of training with the combined datasets (see Table 6) are reported.

Morphing Alg.	UBO [15]			OpenCV [16]			FaceMorpher [17]			StyleGAN [18]		
	EER	$B_{0.05}$	$B_{0.01}$	EER	$B_{0.05}$	$B_{0.01}$	EER	$B_{0.05}$	$B_{0.01}$	EER	$B_{0.05}$	$B_{0.01}$
UBO	**.000**	**.000**	**.000**	.071	.079	.253	.152	.298	.465	.211	.537	.747
OpenCV	.046	.043	.334	**.022**	**.006**	**.041**	.060	.069	.155	.135	.421	.702
FaceMorpher	.024	**.006**	.159	**.020**	**.006**	**.027**	.035	.030	.080	.168	.509	.849
StyleGAN	.171	.833	.987	.057	.082	.281	.124	.254	.535	**.004**	**.000**	**.003**
AMSL	**.022**	**.000**	.250	.100	.100	.200	.050	.050	**.150**	.250	.500	.700
Webmorph	.405	.900	1.000	**.300**	.700	.850	**.300**	**.550**	**.750**	.500	.900	.950
Sqirlz Morph$_D$	**.020**	**.011**	.151	.044	.048	**.146**	.252	.413	.589	.150	.238	.460
FaceFusion	.407	.925	.990	**.176**	**.442**	**.680**	.181	.445	.727	.321	.798	.955
NTNU	.307	.889	.990	.137	.330	.611	**.114**	**.239**	**.510**	.310	.776	.945
UTW	.312	.800	.943	**.204**	**.645**	**.904**	.363	.859	.969	.403	.854	.955
FaceFusion$_{JPG}$	.174	.552	.834	**.141**	**.342**	**.608**	.146	.357	.659	.316	.771	.921
NTNU$_{JPG}$	.215	.688	.889	.183	.478	**.691**	**.174**	**.450**	.745	.353	.816	.952
UTW$_{JPG}$	.405	.898	.981	**.344**	**.776**	**.935**	.365	.844	.965	.456	.925	.987
WAED ↓		.5815			**.4271**			.4675			.6672	

approach provided better results in our experiments, so the metrics are reported only for this implementation.

Experimental results are reported in Table 8: on our testing set all the forensic features seem to have only a limited capability in detecting morphed faces produced by morphing algorithms never seen during the training procedure. Specifically, results suggest a limited generalization capability in the cross-morphing algorithm scenario, with a lower *Equal Error Rate* (EER) on the first set of testing datasets (in which the same morphing algorithm is also used in the training procedure), with respect to the EER achieved in the second, more challenging, block of testing datasets.

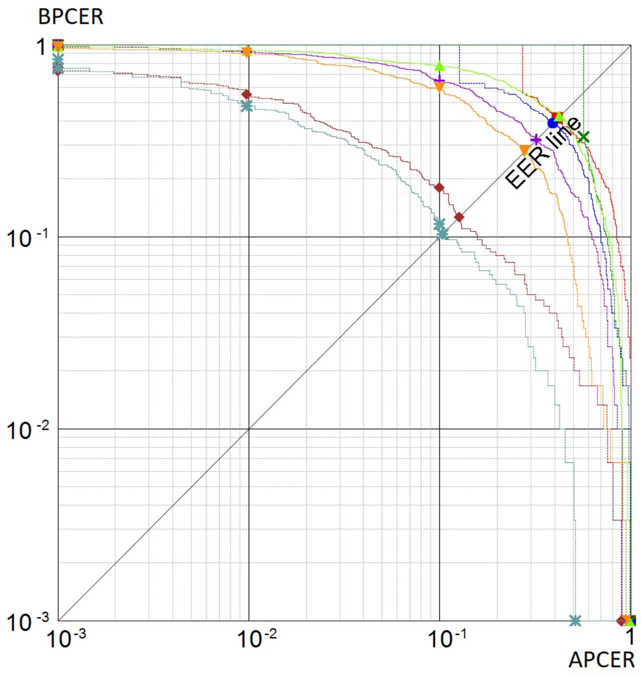
Best performances across different forensic features are provided by the use of wavelets with one-level decomposition (WAED = 0.577, while the three-level decomposition achieves a WAED of 0.603). This finding has a confirmation in [30], in which an Inception-Resnet architecture achieves comparable performance receiving in input RGB images or wavelets.

5) INVESTIGATION ON TRAINING DATA

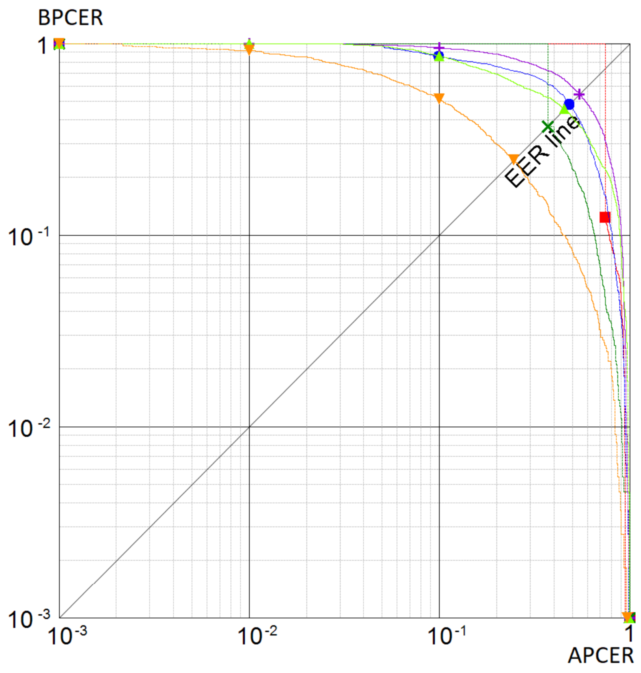
Here, we investigate the influence of training data, and in particular the availability of different morphing algorithms, in the development of robust MAD methods. We train the best MAD detector obtained, *i.e.* the Inception-Resnet network receiving in input RGB faces detected with MTCNN, on different training data configurations. This experimental validation is useful in order to understand how the image visual quality, the variety of morphing algorithms, and the amount of training data influence the final performance of the system. As expected, the results reported in Table 9 reveal that the combination of all available datasets produces the best performance, thus highlighting the importance to train MAD models on varied and large-size datasets. In particular, the presence of different morphing algorithms is a key element, even when they generate images with visible artifacts and, from a general point of view, produce low-quality morphed images (*e.g.* morphed faces produced with WebMorph, FaceMorpher, and OpenCV morphing algorithms). As to this point, we have to consider that the face region is cropped after detection and most of such artifacts are cut off; this allows us to exploit for training the features of the facial region without relying on the heavy presence of artifacts in the region surrounding face (which is unlikely in a real operational scenario). Moreover, results reported in the top part of Table 9, confirm the tendency of MAD approached to overfit on the training dataset, as also reported in [5] and [26]. Indeed, in all cases, best performances are obtained when the morphing algorithms used in training and testing correspond, with the exception of OpenCV and FaceMorpher algorithms, which produce similar morphed images. These considerations highlight the importance of cross-dataset evaluations in the MAD field, in combination with results obtained on sequestered datasets [7], [8].

6) TEST ON FVC-onGOING PLATFORM

Finally, we test the developed S-MAD methods on the SOTAMD sequestered datasets [5] through the FVC-onGoing platform [8]. In particular, following the experimental results, we test different versions of a solution based on the Inception-Resnet V1 pre-trained on ImageNet dataset, which receives input faces cropped with the MTCNN face detector. The first version (referred to as "R-1") is trained on the morphing algorithms exploited to create the training set of all previous experiments, *i.e.* UBO, OpenCV, FaceMorpher and StyleGAN, following the 80-20 split for the training and validation procedure. The second version ("R-2") is trained on all data available in the Revelio framework, thus including, in addition to the previous ones, the morphed images obtained with AMSL, WebMorph, Sqirlz Morph, FaceFusion, NTNU and UTW algorithms. Since the amount of data is increased, we split train and validation sets with



(a) SMAD-SOTAMD_D-1.0.
Competitor reported: R-1 (orange), R-2 (claret), R-3 (dark green), [27] (blue), [60] (light green), [6] (red), [55] (purple).



(b) SMAD-SOTAMD_P&S-1.0.
Competitor reported: R-2_{PS} (orange), [27] (dark green), [60] (light green), [6] (red), [55] (purple).

FIGURE 7. ***Detection Error Trade-off*** **(DET) curves computed on the SOTAMD sequestered dataset on the FVC-Ongoing [8] platform, with digital (left) and P&S images. Further details are available on the platform.**

90% and 10% percentages. The third version of the method ("R-3") is the same as the previous one (R-2), and the JPEG compression (see Section III-D3) is randomly applied to input data during the training phase. Finally, the last version (("R-2_{PS}")) is the same as R-2 but trained by applying the P&S simulation process [27] on input images. Only in this case, the model starts the training with parameters that belong to R-2. A summary of these settings is reported in Table 10.

TABLE 10. Training configuration for the different versions of our method tested on the FVC-onGoing platform.

	Method	R-1	R-2	R-3	R-2_{PS}
Training	UBO	✓	✓	✓	✓
	OpenCV	✓	✓	✓	✓
	FaceMorpher	✓	✓	✓	✓
	StyleGAN	✓	✓	✓	✓
	AMSL		✓	✓	✓
	WebMorph		✓	✓	✓
	Sqirlz		✓	✓	✓
	FaceFusion		✓	✓	✓
	NTNU		✓	✓	✓
	UTW		✓	✓	✓
Augm.	JPEG			✓	
	P&S				✓

Results are shown in Table 11 and also officially published on the platform. It is worth noting that R-1 achieves state-of-the-art results, despite the limited amount and variety of training data that belong to publicly released datasets. R-2 confirms the tendency to have better performance when new, and possibly high-quality, morphed images are available during the training procedure, probably due to also the presence of similar morphing algorithms in the test set [5]. The efficacy of the JPG compression, as observed in the Revelio experimental evaluation, is confirmed by the results of R-3, proving the efficacy of the proposed framework

TABLE 11. Comparison of the results on the sequestered SMAD-SOTAMD_D-1.0 (top) and SMAD-SOTAMD_P&S-1.0 (bottom) benchmarks, respectively, through the FVC-onGoing platform [8]. As shown, MAD algorithms developed with *Revelio* framework overcome the competitors.

Algorithm	EER	$B_{0.1}$	$B_{0.05}$	$B_{0.01}$
[60]	42.32	78.00	82.67	93.33
[6]	41.38	100	100	100
[27]	38.99	100	100	100
[55]	31.80	65.00	79.33	91.67
R-1	27.77	59.33	70.67	90.33
R-2	12.67	18.00	28.33	55.00
R-3	**10.33**	**11.67**	**23.67**	**48.00**
[61]	54.37	94.89	98.27	99.91
[29]	48.04	85.86	97.35	100
[60]	45.52	85.86	96.90	100
[6]	43.34	100	100	100
[27]	37.10	100	100	100
R-2_{PS}	**24.63**	**51.28**	**68.25**	**91.42**

to be an effective and valuable tool in the development and deployment of MAD algorithms. Similar observations are true also for the P&S morphed images: the P&S simulation algorithms implemented in the framework can be effectively used to create competitive solutions avoiding the time-consuming process of printing and scanning real photos. The DET curve is reported in Figure 7, with which is possible to appreciate the detail of the performance of the proposed systems and the competitors tested on digital (left) and P&S (right) morphed images.

To summarize, overall results suggest that it is possible to use the Revelio framework to develop, in a simple and effective manner, state-of-the-art S-MAD systems, clearly improving the performance obtained by the competitors, also exploiting only publicly released datasets.

IV. CONCLUSION, LIMITATIONS AND DIRECTIONS FOR FUTURE WORKS

This paper presents Revelio, a new publicly released framework aimed at providing effective support for the development, training, and evaluation of MAD algorithms. Our extensive experimentation confirms that Revelio allows to develop and test MAD approaches in a simple and effective way, achieving state-of-the-art results on sequestered datasets. Several considerations can be expressed after the analysis of the experimental evaluation. Firstly, the S-MAD task is confirmed to be a challenging task, and the accuracy of existing MAD methods still does not satisfy the real-world operational requirements. The lack of a probe image with which to compare the tested image is a key element for the final performance, this fact is confirmed in the literature where D-MAD methods usually achieve greater accuracy in detecting morphed images.

Moreover, experimental results suggest that the availability of a great amount and variety of training data, including several morphing algorithms and subjects belonging to different source datasets, is an important element to improve S-MAD performance. We believe that, in this context, the understanding of newly proposed MAD systems might be significantly improved by the possibility of sharing a common set of training datasets in combination with tests on public datasets and, in particular, on sequestered datasets hosted in public platforms [7], [8]. The adoption of the Revelio framework and the WAED metric can reduce the effort needed to develop new MAD systems and to test and compare them with other related approaches.

Another point of attention is the printing and scanning process (P&S) which makes the S-MAD task much more challenging. As revealed in the experimental evaluation, this is true particularly when the P&S process is followed by a compression step needed to meet the image size limits in eMRTD chips (typically set to 15 kB). Indeed these two operations contribute to hide or reduce the presence of artifacts that make the morphing process detectable.

In its current stage, the presented framework presents some limitations: firstly, it is challenging to employ fallback face detectors, to reduce the number of dataset elements that are skipped due to the failure of a particular face detector; moreover, the current framework's architecture makes it complex to specify arbitrary combinations of face detectors and landmark extractors (*e.g.* use the DLib landmark extractor with the MTCNN face detector); finally, exporting experiments out of the framework so that they can be tested on external platforms *e.g.* FVC-onGoing is not trivial, and it is a process that could be more streamlined.

A great variety of future work can be planned: firstly, addressing the issues that have emerged during the usage of the proposed framework; secondly, continuous maintenance and documentation activities related to Revelio, including the implementation of new methods that will be published in the literature. Finally, the Revelio framework will be extended also to tackle the D-MAD task, in which a pair of images is available in input: also in this case, we aim to create a simple and shared platform to develop new MAD systems, maintaining the same modular architecture (with the exception of a new D-MAD module) and configuration file usage.

ACKNOWLEDGMENT

This text reflects only the author's views, and the commission is not liable for any use that may be made of the information contained therein.

(Guido Borghi and Nicolò Di Domenico contributed equally to this work.)

REFERENCES

[1] M. Ferrara, A. Franco, and D. Maltoni, "The magic passport," in *Proc. IEEE Int. Joint Conf. Biometrics*, Clearwater, FL, USA, Sep. 2014, pp. 1–7.

[2] M. Ferrara and A. Franco, "Morph creation and vulnerability of face recognition systems to morphing," in *Handbook of Digital Face Manipulation and Detection*. Cham, Switzerland: Springer, 2022, pp. 117–137.

[3] U. Scherhag, C. Rathgeb, J. Merkle, R. Breithaupt, and C. Busch, "Face recognition systems under morphing attacks: A survey," *IEEE Access*, vol. 7, pp. 23012–23026, 2019.

[4] U. Scherhag, A. Nautsch, C. Rathgeb, M. Gomez-Barrero, R. N. J. Veldhuis, L. Spreeuwers, M. Schils, D. Maltoni, P. Grother, S. Marcel, R. Breithaupt, R. Ramachandra, and C. Busch, "Biometric systems under morphing attacks: Assessment of morphing techniques and vulnerability reporting," in *Proc. Int. Conf. Biometrics Special Interest Group (BIOSIG)*, Sep. 2017, pp. 1–7.

[5] K. Raja et al., "Morphing attack detection-database, evaluation platform, and benchmarking," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4336–4351, 2021.

[6] L. Spreeuwers, M. Schils, and R. Veldhuis, "Towards robust evaluation of face morphing detection," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2018, pp. 1027–1031.

[7] National Institute of Standards and Technology. *NIST FRVT Morph*. Accessed: Nov. 30, 2022. [Online]. Available: https://pages.nist.gov/frvt/html/frvt_morph.html

[8] B. Dorizzi, R. Cappelli, M. Ferrara, D. Maio, D. Maltoni, N. Houmani, S. Garcia-Salicetti, and A. Mayoue, "Fingerprint and on-line signature verification competitions at ICB 2009," in *Advances in Biometrics*. Berlin, Germany: Springer, 2009, pp. 725–732.

[9] Biolab. *FVC-onGoing*. Accessed: Nov. 30, 2022. [Online]. Available: https://biolab.csr.unibo.it/fvcongoing/

[10] G. Borghi, G. Graffieti, A. Franco, and D. Maltoni, "Incremental training of face morphing detectors," in *Proc. 26th Int. Conf. Pattern Recognit. (ICPR)*. Washington, DC, USA: IEEE Computer Society, Aug. 2022, pp. 914–921.

[11] G. Borghi, E. Pancisi, M. Ferrara, and D. Maltoni, "A double Siamese framework for differential morphing attack detection," *Sensors*, vol. 21, no. 10, p. 3466, May 2021.

[12] N. D. Domenico, G. Borghi, A. Franco, and D. Maltoni, "Combining identity features and artifact analysis for differential morphing attack detection," in *Proc. Int. Conf. Image Anal. Process.* Cham, Switzerland: Springer, 2023, pp. 100–111.

[13] G. Borghi, A. Franco, G. Graffieti, and D. Maltoni, "Automated artifact retouching in morphed images with attention maps," *IEEE Access*, vol. 9, pp. 136561–136579, 2021.

[14] U. Scherhag, C. Rathgeb, J. Merkle, and C. Busch, "Deep face representations for differential morphing attack detection," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3625–3639, 2020.

[15] Biolab. *Morphed Face Generation Tools*. Accessed: Nov. 30, 2022. [Online]. Available: https://biolab.csr.unibo.it/morphedfacegenerationtools.html

[16] S. Mallick. *Face Morph Using OpenCV—C++/Python*. Accessed: Nov. 30, 2022. [Online]. Available: https://learnopencv.com/face-morph-using-opencv-cpp-python/

[17] A. Quek. *FaceMorpher Morphing Algorithm*. Accessed: Nov. 30, 2022. [Online]. Available: https://github.com/alyssaq/face_morpher

[18] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 8110–8119.

[19] L. DeBruine and B. Jones, "Face research lab London set," *Psychol. Methodol. Des. Anal.*, 2017.

[20] T. Neubert, A. Makrushin, M. Hildebrandt, C. Kraetzer, and J. Dittmann, "Extended StirTrace benchmarking of biometric and forensic qualities of morphed face images," *IET Biometrics*, vol. 7, no. 4, pp. 325–332, Jul. 2018.

[21] Xiberpix. *Sqirlz Morphing Algorithm*. Accessed: Nov. 30, 2022. [Online]. Available: https://sqirlz-morph.it.uptodown.com/windows

[22] FaceFusion. *FaceFusion*. Accessed: Nov. 30, 2022. [Online]. Available: http://www.wearemoment.com/FaceFusion/

[23] D. E. King, "Dlib-ml: A machine learning toolkit," *J. Mach. Learn. Res.*, vol. 10, pp. 1755–1758, Dec. 2009.

[24] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, May 2004.

[25] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.

[26] R. Raghavendra, K. B. Raja, S. Venkatesh, and C. Busch, "Transferable deep-CNN features for detecting digital and print-scanned morphed face images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 10–18.

[27] M. Ferrara, A. Franco, and D. Maltoni, "Face morphing detection in the presence of printing/scanning and heterogeneous image sources," *IET Biometrics*, vol. 10, no. 3, pp. 290–303, May 2021.

[28] L. Debiasi, U. Scherhag, C. Rathgeb, A. Uhl, and C. Busch, "PRNU-based detection of morphed face images," in *Proc. Int. Workshop Biometrics Forensics (IWBF)*, Jun. 2018, pp. 1–7.

[29] U. Scherhag, L. Debiasi, C. Rathgeb, C. Busch, and A. Uhl, "Detection of face morphing attacks based on PRNU analysis," *IEEE Trans. Biometrics, Behav., Identity Sci.*, vol. 1, no. 4, pp. 302–317, Oct. 2019.

[30] P. Aghdaie, B. Chaudhary, S. Soleymani, J. Dawson, and N. M. Nasrabadi, "Attention aware wavelet-based detection of morphed face images," in *Proc. IEEE Int. Joint Conf. Biometrics (IJCB)*, Aug. 2021, pp. 1–8.

[31] P. Aghdaie, B. Chaudhary, S. Soleymani, J. Dawson, and N. M. Nasrabadi, "Detection of morphed face images using discriminative wavelet sub-bands," in *Proc. IEEE Int. Workshop Biometrics Forensics (IWBF)*, May 2021, pp. 1–6.

[32] L.-B. Zhang, F. Peng, and M. Long, "Face morphing detection using Fourier spectrum of sensor pattern noise," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2018, pp. 1–6.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[35] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[37] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[39] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16 × 16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[40] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek, "Overview of the face recognition grand challenge," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 947–954.

[41] A. Martinez and R. Benavente, "The AR face database: CVC technical report, 24," Tech. Rep., 1998.

[42] P. J. Phillips, H. Wechsler, J. Huang, and P. J. Rauss, "The FERET database and evaluation procedure for face-recognition algorithms," *Image Vis. Comput.*, vol. 16, no. 5, pp. 295–306, Apr. 1998.

[43] D. S. Ma, J. Correll, and B. Wittenbrink, "The Chicago face database: A free stimulus set of faces and norming data," *Behav. Res. Methods*, vol. 47, no. 4, pp. 1122–1135, Dec. 2015.

[44] M. Ferrara, A. Franco, and D. Maltoni, "Face demorphing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 4, pp. 1008–1017, Apr. 2018.

[45] E. Sarkar, P. Korshunov, L. Colbois, and S. Marcel, "Vulnerability analysis of face morphing attacks from landmarks and generative adversarial networks," 2020, *arXiv:2012.05344*.

[46] E. Sarkar, P. Korshunov, L. Colbois, and S. Marcel, "Are GAN-based morphs threatening face recognition?" in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 2959–2963.

[47] X. Zhang, S. Karaman, and S.-F. Chang, "Detecting and simulating artifacts in GAN fake images," in *Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS)*, Dec. 2019, pp. 1–6.

[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[49] A. Wolf, "ICAO: Portrait quality (reference facial images for MRTD), version 1.0. standard," Int. Civil Aviation Org., Montreal, QC, Canada, Tech. Rep., 2018.

[50] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, "A survey on visual transformer," 2020, *arXiv:2012.12556*.

[51] R. Durall, M. Keuper, F.-J. Pfreundt, and J. Keuper, "Unmasking deepfakes with simple features," 2019, *arXiv:1911.00686*.

[52] A. Graps, "An introduction to wavelets," *IEEE Comput. Sci. Eng.*, vol. 2, no. 2, pp. 50–61, Jun. 1995.

[53] M. Chen, J. Fridrich, M. Goljan, and J. Lukáš, "Source digital camcorder identification using sensor photo response non-uniformity," *Proc. SPIE*, vol. 6505, pp. 517–528, Mar. 2007.

[54] *Information Technology—Extensible Biometric Data Interchange Formats—Part 5: Face Image Data*, Standard, International Organization for Standardization, 2019.

[55] R. Ramachandra, S. Venkatesh, K. Raja, and C. Busch, "Detecting face morphing attacks with collaborative representation of steerable features," in *Proc. 3rd Int. Conf. Comput. Vis. Image Process.* Singapore: Springer, 2020, pp. 255–265.

[56] D. Güera and E. J. Delp, "Deepfake video detection using recurrent neural networks," in *Proc. 15th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Nov. 2018, pp. 1–6.

[57] T. Neubert, C. Kraetzer, and J. Dittmann, "A face morphing detection concept with a frequency and a spatial domain feature space for images on eMRTD," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, Jul. 2019, pp. 95–100.

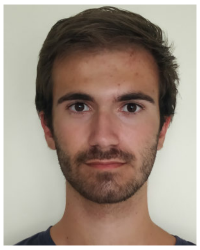
[58] J. J. W. Meijer, "Morphing detection based on regional analysis of local frequency content," B.S. thesis, Univ. Twente, Enschede, The Netherlands, 2020.

[59] B. Chaudhary, P. Aghdaie, S. Soleymani, J. Dawson, and N. M. Nasrabadi, "Differential morph face detection using discriminative wavelet sub-bands," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 1425–1434.

[60] C. Kraetzer, A. Makrushin, T. Neubert, M. Hildebrandt, and J. Dittmann, "Modeling attacks on photo-ID documents and applying media forensics for the detection of facial morphing," in *Proc. 5th ACM Workshop Inf. Hiding Multimedia Secur.*, Jun. 2017, pp. 21–32.

[61] R. Raghavendra, K. B. Raja, S. Venkatesh, and C. Busch, "Transferable deep-CNN features for detecting digital and print-scanned morphed face images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1822–1830.

GUIDO BORGHI received the M.Sc. degree in computer engineering and the Ph.D. degree in information and communication technologies from the University of Modena and Reggio Emilia, Italy, in 2015 and 2019, respectively. He is currently an Assistant Professor with the Department of Computer Science and Engineering, University of Bologna, Cesena Campus. His research interests include computer vision and deep learning techniques applied to intensity and depth images for face analysis, biometrics, driver monitoring, and human–computer interaction.



NICOLÒ DI DOMENICO received the B.Sc. and M.Sc. degrees in computer science and engineering from the University of Bologna, Italy, in 2020 and 2023, respectively. After his graduation, he is currently a part of the Biometric Systems Laboratory, University of Bologna, under the supervision of Prof. Davide Maltoni.

ANNALISA FRANCO received the Ph.D. degree in electronics, computer science and telecommunications engineering from DEIS, University of Bologna, Italy, in 2004. During the Ph.D. degree, she worked on multidimensional indexing structures and their application in pattern recognition. She is currently an Associate Professor with the Department of Computer Science and Engineering, University of Bologna. She is a member of the Biometric System Laboratory, Department of Computer Science, Cesena. She has authored several scientific papers and served as referee for a number of international journals and conferences. Her research interests include pattern recognition, biometric systems, image databases, multidimensional data structures, and face recognition in the context of electronic identity documents.

MATTEO FERRARA received the bachelor's degree (cum laude) in computer science from the University of Bologna, Italy, in 2004, the master's degree (cum laude), in 2005, and the Ph.D. degree, in 2009. He is currently an Associate Professor with the Department of Computer Science and Engineering, University of Bologna. He is the author of several scientific articles and he served as a referee for international conferences and journals. His research interests include pattern recognition, computer vision, image processing, and machine learning. Most of his applied research is in the field of biometric systems. He is a member of the Biometric System Laboratory. He is one of the organizers of the international performance evaluation initiative named FVC-onGoing. Moreover, he is one of the authors of the well-known fingerprint recognition algorithm named Minutia Cylinder-Code (MCC). Finally, he first proved that face morphing can be exploited to fool automated border control (ABC) systems. He took part in national and European research projects and in consultancy projects between the University of Bologna and foreign universities and companies.

DAVIDE MALTONI (Senior Member, IEEE) is currently a Full Professor with the Department of Computer Science and Engineering (DISI), University of Bologna. His research interests include pattern recognition, computer vision, machine learning, and computational neuroscience. He is the Co-Director of the Biometric Systems Laboratory (BioLab), which is internationally known for its research and publications in the field. Several original techniques have been proposed by BioLab team for fingerprint feature extraction, matching and classification, for hand shape verification, for face location, and for the performance evaluation of biometric systems. He is the coauthor of the *Handbook of Fingerprint Recognition* (Springer, 2009) and holds three patents on *Fingerprint Recognition*. He was an elected International Association for Pattern Recognition (IAPR) Fellow, in 2010.

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement