IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 December 2022, accepted 6 January 2023, date of publication 13 January 2023, date of current version 19 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3236502

APPLIED RESEARCH

An Artificial Intelligence System for Automatic Recognition of Punches in Fourteenth-Century Panel Painting

MARCO ZULLICH[1], VANJA MACOVAZ[2], GIOVANNI PINNA[1,3], AND FELICE ANDREA PELLEGRINO[1], (Member, IEEE)

[1]Department of Engineering and Architecture, University of Trieste, 34127 Trieste, Italy
[2]Department of History, Archaeology, Geography, Fine and Performing Art, University of Florence, 50121 Florence, Italy
[3]Department of Mathematics and Geosciences, University of Trieste, 34127 Trieste, Italy

Corresponding author: Marco Zullich (marco.zullich@gmail.com)

ABSTRACT In Late-Medieval panel paintings from the Tuscan area, mechanical tools called *punches* were used to impress repeated motifs on gold foils to create decorative patterns. Such patterns can be used as clues to objectively support the attribution of the paintings, as proposed by art historian Erling S. Skaug in his decades-long study on punches. We investigate the feasibility of employing automatic pattern recognition techniques for accelerating the process of classification of punches by experts working in the field. We propose a system composed of (a) a Convolutional Neural Network for categorizing a punch contained in a frame, and (b) an additional component for uncertainty estimation, aimed at recognizing possible Out-of-Distribution (OOD) samples. After collecting a set of 3815 punches from four 14th century panel paintings from Tuscany, we train a Convolutional Neural Network which achieves very high test-set accuracy. As far as the uncertainty estimation is concerned, we experiment with two techniques, OpenGAN and II-loss, both exhibiting very positive results. The former seems to work better on specific data extracted from images of panel paintings, while the latter showcases a more consistent behavior when considering additional OOD data obtained randomly. These outcomes indicate that an application of our system in support of experts is feasible, although we subsequently show that additional experiments on larger datasets might be required.

INDEX TERMS Machine learning, computer vision, pattern recognition, image classification, artwork classification, artwork authentication, digital humanities.

I. INTRODUCTION

Authentication of an artwork, i.e., the process of attributing the authorship of said artwork to one or more artists or to a certain workshop, is an important task that can spread light on artistic relations and mutual influences in a precise time and space. This procedure is almost exclusively done by experts in the field and is carried out via careful examination of the pieces of art aimed at finding stylistic or technical similarities between it and other pieces by the same author or authors, when other sources such as written documents are not found. As much as this procedure is accurate or detailed, it does not follow strict scientific principles: the examination of the art piece is often visual and not aided by technological tools, while historical records require contextualization, except from very rare situations where rich information points indisputably toward a single attribution. Technical information as the one provided by modern restoration began to be used during the 20th century to reinforce or dispute stylistic consideration in a continuous comparison. The same kind of attention was addressed to mechanical tools used in the Middle Ages to help the painters in their work; when these traces are well-recognizable and not ruined, e.g., by inappropriate conservation techniques or by accidents, they may be exploited to assist the experts in their analyses. In late medieval panel painting, starting from the fourteenth century,

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

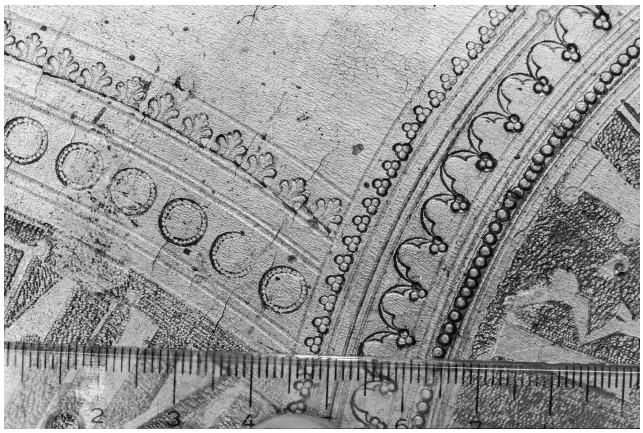


FIGURE 1. Photography taken during 1971 depicting the process of manual scaling of different categories of punches by Erling S. Skaug.

artists operating in the Tuscan area began to use small iron tools with a well-defined shape to produce repeated motifs, like flowers or simple geometrical shapes, on the gold foil which had been applied to the wooden panel earlier. These tools, called *punches*, were impressed repeatedly throughout the artwork, or even in multiple art pieces. Sometimes, they were shared between artists from the same atelier, who used them for their productions; some other times they changed their owner after workshop disruption, master to pupil transition or even temporary cooperation between different authors. Due to the manual production via engraving of the punches, it was extremely unlikely for two punches to be identical even in the, not so common, presence of a clear intent of imitation. Thus, a precise and scientifically-sound analysis of these shapes, e.g., by accurately measuring their proportions, may uncover dissimilarities between paintings using similar but not exactly identical punches. In this sense, this is a powerful tool that can be used by art historians for aiding in processes of authentications, or, more generally, of establishing logical relationships between sets of paintings. This is essentially the bulk of a decades-long work by Erling [1], Mojmir [2] and a small group of other art historians. Thanks to very accurate batches of measuring carried out using essentially a reflex camera and calipers, Skaug was able to boost the authentication process and critical analyses about panel paintings from the fourteenth and fifteenth century in the Tuscan area. This process has proven extremely accurate, albeit it took a great amount of time for Skaug to carry out his investigations, that required a huge photographic campaign which lasted several decades and, most of all, it required to take detailed pictures of small areas of every painting with the apposition of a ruler in every frame, a really difficult and time consuming task when dealing with huge altars or artworks not musealized. An example of Skaug's workflow is depicted in Figure 1.

Meanwhile, technologies concerning computer vision and automatic pattern recognition have greatly progressed: in the present work, we aim at reproducing, in a proof-of-concept fashion, Skaug's analysis, by advancing the part which cost him the greater deal in terms of time, which is the manual analysis of every single frame containing a punch which he inspected by hand after printing at fixed scale every image of his negatives archive. Starting from large-resolution images of a set panel paintings from Siena and Fiesole (Italy) we took from 2019 to 2021, we experiment with using Convolutional Neural Networks (CNNs)—powerful Deep Learning (DL) models often employed for image classification—to categorize frames containing punches previously classified manually by Skaug. In addition, due to a well-known pitfall of CNNs in categorizing Out-of-Distribution (OOD) images,[1] we propose to enhance the model with recently developed techinques, OpenGAN [3] and II-loss [4], in order to couple the predicted category output by the CNN with an additional value representing a *confidence* that the image is or is not OOD. We wish our pipeline of models to be a tool not designed for the general public, but a support in the analyses of art historians or experts in the field. We envision this tool to be used as a quick way to formulate suggestions on the categorization of punches in an artwork, to be used as an initial hypothesis by the experts that make use of the model. The OOD detection is, hence, a quick way to discard punches not belonging to the initial 19 categories, or possible parts of paintings (e.g., cracks vaguely resembling punches) that might have initially been confused for punches. The CNN we originally train, a ResNet18 [5], is able to perfectly categorize images from 19 different classes of punches, thus showing that this task is not hard for an Artificial Neural Network (ANN). Additional experiments concerning the detection of OOD data show that both OpenGAN and the II-loss are valid techniques for this task, as they both achieve very satisfactory results, albeit not perfectly accurate.

The paper is organized as follows: Section III introduces the main technologies used in the production of the dataset (i.e., the part concerning the photo-capture process and the processing involved in the building of the final dataset) and in the training of the automatic tools used for the classification. Section IV presents the hyperparameter selection and the corresponding results of the CNN and the enhancements for OOD detection. Finally, in Section V, we provide a critical review of our work, detailing points of weakness and ideas for future directions.

Our code and dataset are publicly available on GitHub: https://github.com/marcozullich/punches_recognition.

II. RELATED WORK

1) AUTHENTICATION OF AN ARTWORK

The process of attribution in Medieval Art History is a challenging task. Usually, the small amount of written sources at our disposal requires the usage of stylistic elements of artworks to discriminate between different authors or workshops [6]. The same reasoning is applied when trying to

[1] In our case, OOD images may be, for instance, pictures of categories of punches not belonging in the training set of the CNNs, or random crops of painting in areas where punches are not present.

understand the time of the creation of an artwork, understanding how the stylistic characteristic of a painter changed during his career due to personal behavior or outer influences. Obtaining an unanimous attribution is infrequent, especially for such authors who, unlike Giotto or Simone Martini, are not the subject of intense, detailed studies from art historians [7], [8]. The study of punched decoration became, in this sense, a powerful tool for art historians during the latter part of the 20th century [9]. In particular, in Tuscany, the prominence of punched decoration on gold ground allowed, after several photographic recognition campaigns and accurate analyses of 1:1 scaled pictures to identify a huge amount of specific decorative motifs related to different schools and workshops. It is important to define the concept of *art workshop* in the Middle Ages as a large atelier where several individuals, sometimes more than 10, worked together on the different tasks belonging to the process of panel painting. The concept of *author* in a similar context is often related to the chief of the workshop, but in reality it can refer also to the plethora of artists collaborating within this workshop [10]. Mechanical tools, like punches, are property of the workshop and, hence, could not spread light on each of the single artists beneath each product of the workshop [2], [9]. Tracking the punches used within a workshop for a given amount of time allows to identify substitution and integration of said tools, and so could help also with the chronological attribution of artworks, even in the situation when only a small amount of artworks from the same workshop or group of authors have an existing dating [11]. These tools were also, in specific situations, shared between different painters and appear to be used during different generations of artists, suggesting the possibility of inheritance, selling, or trade of punches [10], [12]. This is the context in which Skaug's work—already introduced in detail in Section I—fits in, as, throughout his researches, he proposed the use of quantitative data, meticulously gathered from manual measuring, to infer additional information about attribution of artworks, specifically panel paintings belonging to the Tuscan scene of the 14th and 15th century.

a: AUTHENTICATION AIDED BY AUTOMATED SYSTEMS

The first works concerning art piece authentication using automated systems based on Machine Learning (ML) approaches started at the end of the 20th century, with the work by Kröner and Lattner [13], who operate authentication of black and white free-hand drawings by means of a Naïve Bayes classifier built on top of handcrafted features. At the same time, Melzer et al. [14] propose a solution for detecting brush strokes in portrait miniatures using a simple Multilayer Perceptron (MLP). Their idea is that features connected to the strokes can be used by experts to aid further investigations, e.g., authentication, to said portraits. Previous attempts [15], [16] at applying ML techniques to art were related to information retrieval and not to authentication or artwork classification.

After these initial attempts, a large body of works were produced for trying to classify art pieces w.r.t. their authors by devising ML pipelines composed by handcrafted feature extractors (such as color histograms, statistical descriptors, features derived from Fourier or wavelet transforms, *etc.*) on top of which a classifier is fitted. For instance, Widjaja et al. [17] and Johnson et al. [18] use a Support Vector Machine [19] as classifier; van den Herik and Postma [20] use an MLP. More recent works [21], [22] use Weighted k-Nearest Neighbors (WNN) [23]. While not aimed at directly authenticating an art piece, other works using similar techniques are noteworthy. More specifically, the report by Lettner et al. [24] is maybe the one which comes closer to our work, as they propose a model for classifying the stroke texture on panel paintings in order to aid further investigations on art pieces. The aim of their work is close to ours, as we strive at producing a tool which is not directly aimed at classifying artworks, but supporting experts by providing additional information regarding said artwork for further analysis and, possibly, authentication.

More recently, with the steep rise in computational power which researchers have at their disposals, CNNs have been used both as feature extractors and classifiers for the task of artwork classification. David and Netanyahu [25] use a pipeline of Convolutional AutoEncoder for dimensionality reduction and a CNN for classification; Tan et al. [26], Cetinic et al. [27] use CNNs for classifying authors, styles, and genre, while Bianco et al. [28] enrich the approach by injecting further hand-crafted feature into the model; Rodriguez et al. [29], Sandoval et al. [30] use a CNN as a feature extractor in image patches: in the former case, the authors combine the features using a WNN as a final classfier for style classification of the whole art piece, while, in the latter, the classification is operated by a shallow MLP. Even more recently, [31] propose a hierarchical, CNN-based classifier for fine-grained art style classification. Finally, Milani and Fraternali [32] propose a CNN for recognizing figures of saints inside paintings.

A. OUR CONTRIBUTIONS

To the best of our knowledge, we are the first to make use of ML techniques for the problem of classifying historical art pieces based upon specific repetitive elements (like punches); the existing literature aims mainly at attributing a whole artwork to an author (or to a specific genre), without taking explicitly into account features, like punches, that these artworks exhibit, thus not enabling art historians to exploit such significant amounts of information for the attributions. The exception to this is [24], where the authors aim at classifying the stroke texture—although this is not applied to historical paintings, where the conservation status of artworks may dramatically alter these features. As an additional contribution, we make considerations regarding the uncertainty on the predictions output by their models, addressing, then, the

TABLE 1. Artist, title, location, year, and code for each of the paintings used in the present work. The code is a short 3-letter string we devised to quickly recall a single work of art throughout this manuscript.

Artist	Title	Location	Year	Code
Lippo Memmi	*San Luigi* (or *San Ludovico*) *dei Francesi*	Pinacoteca Nazionale, Siena, Italy	1330 ca.	SLF
Lippo Memmi	*San Francesco d'Assisi*	Pinacoteca Nazionale, Siena, Italy	1330 ca.	SFA
Lippo Memmi	*Madonna con Bambino*	Pinacoteca Nazionale, Siena, Italy	1330 ca.	MCB
Bernardo Daddi	*Evangelista*	Museo Bandini, Fiesole, Italy	1335 ca.	EVA

problem of the behavior of our solutions on OOD data. All the papers reviewed in this section do not provide such analyses.

III. MATERIALS AND METHODS

A. THE DATASET

The dataset is composed of 4 full-sized panel paintings from artists Lippo Memmi and Bernardo Daddi. Table 1 details information on these paintings, such as title and location. Lippo Memmi, brother-in-law of Simone Martini, was a Sienese painter who worked in the same workshop of Simone and, after the latter's death in 1344, continued his school. Bernardo Daddi, on the other hand, was a former collaborator of Giotto's school and the chief of one of the most productive workshops in Florence, the city where the Sienese invention of punched decoration was transposed in 1333-1334. Memmi's Madonna (MCB), dated to 1325-1330, was painted for Santa Maria de' Servi's church; today it is stored in the Pinacoteca Nazionale of Siena, whereas Saint Francis (SFA) and Louis (SLF) belong to a dismembered tryptich dated to the timeframe 1330-1340. Daddi's panel painting belongs to the mid production of this huge workshop and, based on its punched decoration, could be placed around 1335, during the first attempts at punch decoration by the author [33]. The 4 paintings are depicted in Figure 2.

These paintings were photographed by us and then digitalized following the procedure detailed in Section III-A1. Out of these digital copies, we produced a dataset of 3815 frames of punches, as indicated in Section III-A2.

1) PHOTOGRAPHY

The process of digitizing artworks and, in particular, gold ground masterpieces, requires an additional effort typically not required in opaque artwork reproduction. Usually, high-resolution digital imaging involves the production of several frames taken at very low distance from the surface moving the camera, as shown in Figure 3a, or a similar amount of images taken at higher distance with a long focal length lens, changing the orientation of the camera system while keeping fixed the location, as shown in Figure 3b.

In the present work, we used the latter, taking pictures with a 300mm prime lens at a distance varying from 1.2 to 1.8 meters from the artwork. Usually, the first method is preferred but it requires a huge camera moving mechanism in front of the painting and is not able to develop a correct reproduction of gold reflectivity due to the varying angle of light source when the camera is moved along the surface. The method used in the present work prevents this problem in small artworks like those presented in Section III-A and, if the enlightenment provided is sufficiently diffused and not direct as shown in Figure 4, it allows to generate an overall smooth reproduction of gold ground. Another way to enhance surface details, as in the case of punched decoration, involves the use a raking light system but it reveals also craquelure, creating confusing and less suitable images.

Expeditious 3D modelling was also used to create a low-resolution model of the artworks and low-resolution orthoimages that could be scaled and used to rectify high-resolution images, avoiding even small perspective distortion capable of changing the proportions of small details in the image.

2) CREATION OF THE PUNCHES DATASET

From the mentioned 4 images, we proceeded to create the final dataset of 3815 frames by manually cropping a selected set of 19 punch categories. Figure 5 showcases a detail of a painting, displaying some cropped and zoomed-in punches present there.

The cropping was operated using Adobe Photoshop version 22.4.03, through which the coordinates of the crops within the full pictures of the paintings were determined, and a Python script making use of OpenCV [34]. Adobe Photoshop was used to mark manually the coordinates of the punches on the pictures of the full paintings. We made use of a JavaScript Photoshop script to export the coordinates from Photoshop to a Comma-Separated Values (CSV) file. Finally, we recovered the coordinates from the CSV using Python, and we used the library OpenCV to automatically operate the crops on the full paintings.

The 3815 crops were manually labeled according to their punch category as defined by Skaug [1].

We did not upsample or downsample the crops. Since the image of every painting was at first scaled at 1000 dpi on the orthorectified 3D model of the artwork, this means that punches from the same category but different paintings will have the exact same pixel dimension.

Finally, we operated a train-test split by assigning 80% of the images to the training set and 20% to the test set by means of a stratified random sampling, thus ensuring equal representation to each of the 19 categories of punches. In addition, we further split the training set by assigning a 10% of images to a validation set, which will be used for model selection in the OOD detection.

For reasons concerning computational efficiency, we further preprocessed the data by resizing all images to the

FIGURE 2. The four paintings used in the creation of the dataset of punches. In order from left to right, the paintings are: SLF, SFA, MCB, EVA. Codes as from Table 1.

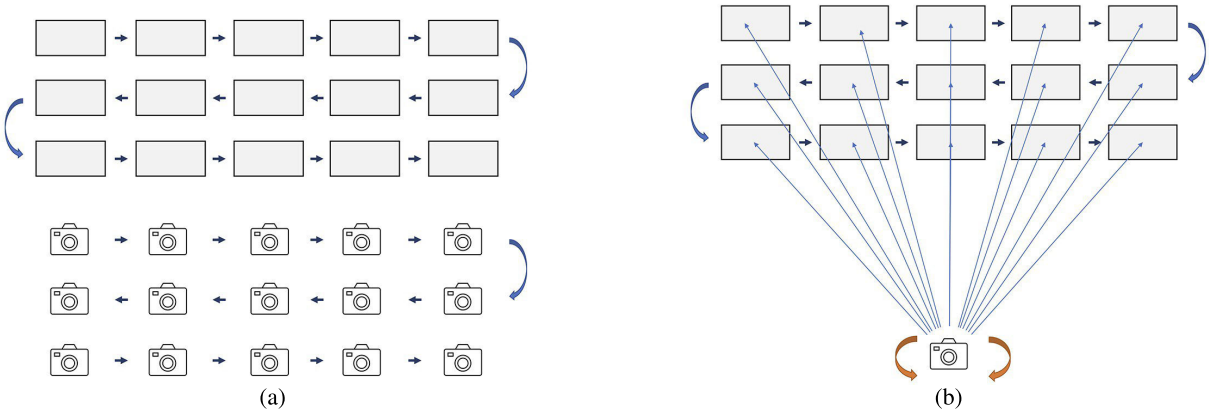


FIGURE 3. Two different camera movements used to obtain high-resolution images of an artwork. In the present work we choose the setting labeled as (b) due to its expeditiousness.

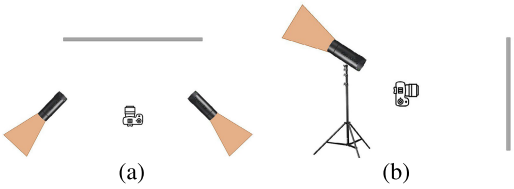


FIGURE 4. Disposition of camera and light sources opposing the painting to provide diffuse lighting, top view (a) and side view (b).

common size of 256 × 256 pixels and by *normalizing* all images such that they have a channel mean of 0 and a variance of 1 for what concerns the pixel values. This speeds up the passing of data through an ANN as (a) the ANN can process simultaneously a batch of images—since they all share a common size, the data are *packed* in a single tensor—, and (b) the training of the ANN is more stable as the pixel values are small and *clustered* around 0. On the other hand, this increase in efficiency comes at a cost; in fact, due to the resizing, we lose the information conveyed by the original size of the image: there might be two punches of different sizes but equal aspect ratio and shape. In this case, the mere fact that the two images have completely different sizes might be enough for a model to operate the classification. After the resizing, though, since the crops share aspect ratio, they now have the same size and the punches depicted inside become potentially indistinguishable.

3) DATASET COMPOSITION

Table 2 details the number of punches in each category and specifies in which art pieces the categories are present. As we can see, only a small number of categories are present in multiple paintings and none is present in all four of them. This is because the paintings belong to different artists from different ateliers using different tools. In addition, some categories (e.g., 438, 659) have a very small number of instances. This is to be taken into account, as training ML models on imbalanced datasets (i.e., datasets whose distribution of one

TABLE 2. Categories of punches in the collected dataset. The "Punch no." value is recovered from [1].

Punch no	# instances	Painting(s)	Example	Drawing
43	252	MCB, SFA, SLF		
72	250	MCB, SFA, SLF		
96	74	MCB, SFA		
171	314	SLF, SFA		
178	79	MCB, SFA		
274	202	SLF		
296	109	SLF		
330	258	EVA		
371	160	MCB, SLF		
385	610	MCB, SFA, SLF		
390	399	EVA		
438	17	MCB		
554	132	SLF		
570	55	EVA		
619	48	MCB, SFA, SLF		
655	158	MCB, SFA		
657	63	SLF		
659	22	MCB		
729	613	SLF		
Total	3815	—		

FIGURE 5. Detail of the painting SLF showcasing zoomed-in crops of various categories. Punches come in different shapes and sizes. Those displayed on the left side are upscaled by a factor of 2, while those on the right by a factor of 6.

or more variables are severely skewed) may prove particularly challenging [35].

B. ML-BASED TOOLS

As previously mentioned in Section I, we want to construct a ML-based approach which serves a tool for supporting art historians or experts by formulating predictions on images containing one single punch. This prediction is related to the category of the punch which is portrayed in the frame. In addition, we want this prediction to encompass some form of uncertainty, so that the model is able to detect OOD samples. For instance, these may comprise pictures not containing any punches, or any punches of the categories the model was trained to recognize. First we will present the part concerning *basic* classification using a CNN. Then, we will be presenting possible approaches for OOD detection. Our goal is to produce a *system* such that both classification and OOD detection, possibly with high performance for both aspects, is carried out.

1) CNNs FOR IMAGE CLASSIFICATION

CNNs [36], [37] are a class of ANNs particularly suited for classification or regression tasks on structured data such as images. The underlying idea is to process the incoming data with a sequential cascade of *convolutional layers*, each performing cross-correlations (improperly termed as *convolutions*) with *learnable weights*, followed by an element-wise nonlinear operation, called the *activation function*.

The cross-correlation acts by *sliding* a small *kernel* pixel-by-pixel along the horizontal and vertical axes of the image, represented as a matrix. For each possible *overlap* between image and kernel, the cross-correlation outputs a value which indicates the *agreement* between the content of the kernel and the slice of the image overlapping with the kernel itself. So, the output of a cross-correlation is another matrix whose elements represent the response of the image to all the possible overlaps with the kernel. Kernels can be designed to represent specific *features*, so that a cross-correlation with high value indicates a presence of the specific feature in the specific portion of the image overlapping with the kernel. In Figure 6, we can see the application of cross-correlation with a handcrafted kernel for the detection of horizontal edges known as the Sobel filter [38]. The term *learnable kernels*, hence, indicates that the CNN *learns* to adjust the values of the kernel in order to better capture the features that it considers to be useful for the task at hand (in our case, classification of punches).

In CNNs for images, the inputs are usually color images in the RGB color codes (so, 3 matrices of common size). Since the computation of the cross-correlation shown in Figure 6 applies only to grayscale images, which can be represented as one single matrix, the cross-correlation is extended as such: three kernels are applied separately to each color code, and the outputs are then summed into one single matrix. This operation encompasses the *core* of the computation operated by the first *convolutional layer*. The procedure is illustrated in Figure 7.

Usually, convolutional layers operate a large number of these cross-correlations (e.g., 64 to 512), thus producing a large number of matrices depicted in the figure. Each of these tries to identify a different feature useful for the final classification task. Most of the times, this output is passed to a non-linear *activation* function to create a relationship of non-linearity between the input to the convolutional layer and its output. As previously mentioned, a CNN is a stack of multiple of these convolutional layers, and each layer in this hierarchy builds upon the output of the previous convolutional layer. The underlying idea is an assumption called *compositionality* of the features, which states that ANNs operate by learning *low-level*, basic features (e.g., edges, colors, *etc.*) in the first layers, then combining these features into higher-level ones, which are useful for the task at hand. In the context of punches, these features might be the specific shapes of the punches, the combination of colors at which they occur, *etc.* Experimental investigations [39], [40], [41] seem to corroborate the assumption of compositionality. Figure 8 schematically depicts this concept.

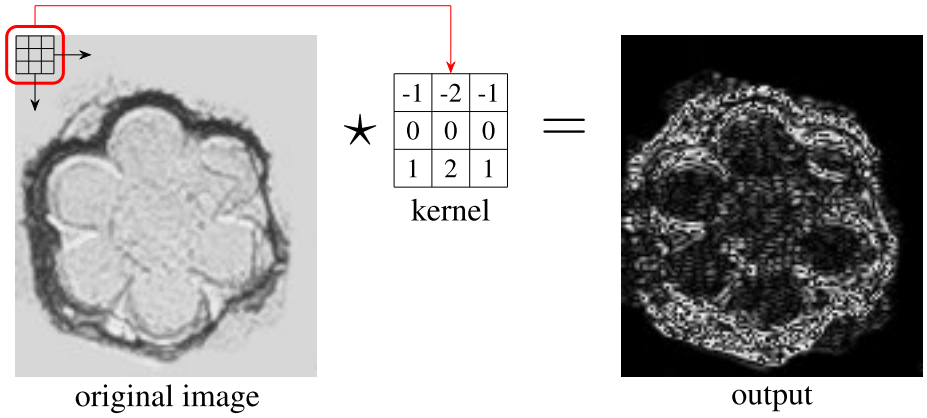


FIGURE 6. Example of application of the cross-correlation with hand-crafted kernel (Sobel filter) for horizontal edge detection on a grayscale image of a punch. A 3 × 3 kernel is *sled* along the image on the horizontal and vertical axes. For each possible overlap, a scalar value is produced quantifying the presence of a horizontal edge in the portion of the image involved. In the output, small values, in black, indicate absence of a horizontal edges, while high values, in white, indicate the presence of them. The cross correlation operator between the image and the kernel is indicated with "⋆".

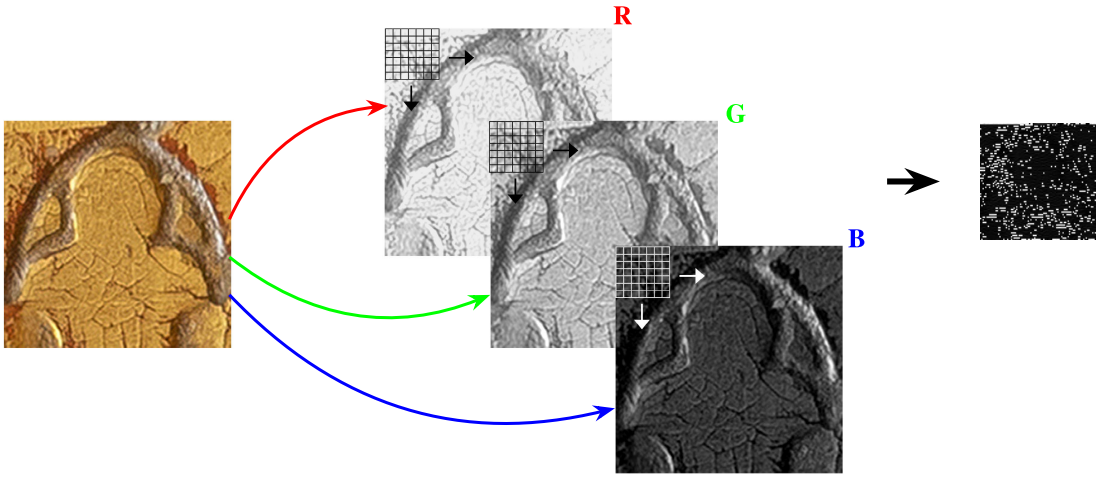


FIGURE 7. Depiction of a generic cross-correlation operated in the first layer of a CNN. The image, composed of three matrices in the RGB color codes, undergoes three cross-correlations (one per color code) with three separate 7 × 7 kernels. The three outcomes are then summed to produce the output, which is a matrix whose size is roughly half of the original image. Conversely to what shown in Figure 6, this output is not directly interpretable, as the parameters of the kernels are not hand-crafted by a human, but are produced by the CNN in its training process, and are hence intelligible only by the model itself.

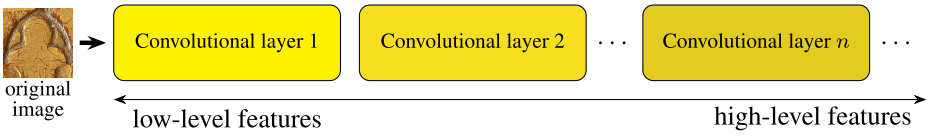


FIGURE 8. Depiction of a CNN as a sequential stack of convolutional layers. The first layer learns low-level features taking as input the original image. Each subsequent layers take as input the features produced by the previous layer, combining them into higher-level features.

Convolutional layers are characterized by the channel dimension of their output (i.e., the number of cross-correlations they perform), by the kernel size of each cross-correlation, and by other hyperparameters relevant to the convolutions, e.g., padding and stride. Common CNN architectures often group consecutive convolutional layers in *blocks*, as in the case of Residual Networks (ResNets) [5]. ResNets are CNN architectures for image classification composed of groups of convolutional layers called *residual blocks*. For instance, the simplest ResNet architecture, ResNet18, is built by stacking residual blocks, also called *BasicBlocks* each of 2 convolutional layers all with the same

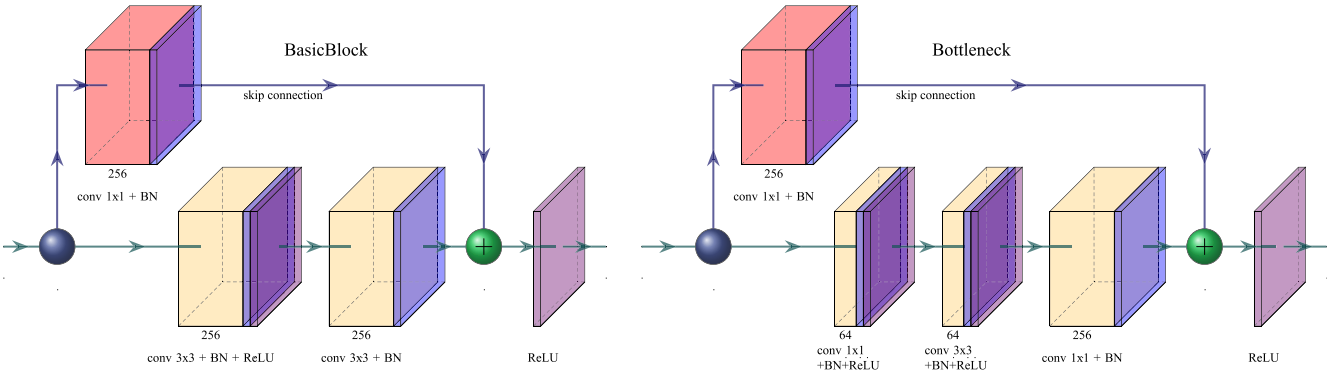


FIGURE 9. Depiction of ResNet's BasicBlock and Bottleneck with a base width of 256 (i.e., they output data with 256 channels). Convolutional layers—"conv"—are indicated in yellow or red. "conv n×n" indicates that the layer uses kernels of size $n \times n$. The convolutional layers in yellow have always a stride of 1 and a padding of 1 in case the kernel has size 3 × 3, 0 otherwise. The red convolutional layer in the skip connection has a stride of 2 and a padding of 0. It is present only when the block operates a *downsampling* of the spatial dimensions. In this case, also the first convolutional layer in yellow has a stride of 2 instead of 1. Batch Normalization ("BN"—in blue in the figure) is applied after each convolutional layer. "ReLU", depicted in green, stands for the Rectified Linear Unit activation function.

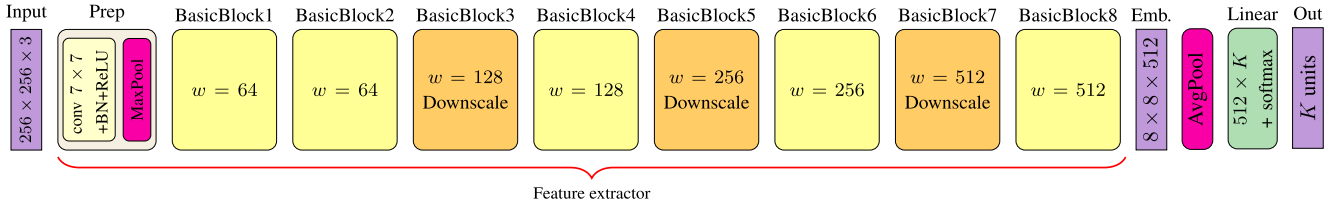


FIGURE 10. Schematic representation of the ResNet18 architecture [5]. The incoming data ("Input"), in our case of size 256 × 256 × 3, passes through a *preparatory block* ("prep") which has a 7 × 7 convolutional layer with stride 2, padding 3, a subsequent batch normalization phase and a ReLU activation function. Following that, the data is shrunk via a max pooling layer ("MaxPool"). Then, it flows through a cascade of BasicBlocks, progressively shrinking in spatial dimension and increasing in channels. "w" indicates the *base width* of the block, while "Downscale" specifies that the block is responsible for downsampling the spatial dimension of the data. Then, the blocks produce an embedding ("Emb.") of size 8 × 8 × 512. This is averaged along the spatial dimensions ("AvgPool"—*Average Pooling*) outputting a vector of 512 elements. This, in turn, is processed by a linear layer with softmax activation. It projects the vector into a K-dimensional space, where K is the number of classes. The softmax turns it into a simplex, which is the final output ("Out") of the model.

output channel dimension and 3 × 3 convolutional kernels. Deeper architectures, like ResNet50, make use of a different residual block, the *Bottleneck*. Both the BasicBlock and the Bottleneck architectures are schematized in Figure 9. The input of these blocks is *duplicated* along two paths: the first one goes through the convolutional layers, while the second one skips them wholly, forming a *skip connection*. The data in the skip connection is then summed to the output of the last convolutional layer. Sometimes, it may be necessary to *downsample* the spatial dimension of the incoming data. This is achieved by doubling the stride hyperparameter in the first convolutional layer. In order to downsample the data in the skip connection as well, a 1 × 1 convolutional layer with double stride is then added to the skip connection.

The ResNet18 is built by stacking up eight BasicBlocks, after which the data is averaged along the spatial dimensions (*average pooling*) and fed into a linear classification layer. A diagram depicting the ResNet18 architecture is presented in Figure 10.

In the present work, we experimented with training a CNN for image classification using the ResNet18, ResNet34, and ResNet50 architecture. Specifically, we *fine-tuned* these network, using as an initialization the parameters pre-trained on the large-scale image dataset ImageNet [42], thus operating a *transfer learning* on our dataset of punches. All models were trained using the Cross Entropy (CE) loss, which is the most common cost function for multi-class categorization tasks.

2) RECOGNITION OF OOD SAMPLES

Despite the immense success of CNNs and, more generally, ANNs in the field of classification, most models do not address the problem of recognizing OOD samples. A common behavior is that well-performing ANNs trained with supervision will tend to output highly confident predictions (i.e., predictions highly biased toward one single category), even when the input lies far from the original data distribution (e.g., completely random data) [43], thus making it very hard to recognize OOD instances based upon, for instance, considerations on the distribution of neuron activations in the output layer.

Hence, the goal we wish to reach is either (a) an ANN which exhibits low confidence, ideally for all the possible categories, when an OOD data is shown to it, or (b) an additional predictor which allows us to recognize whether a datapoint is OOD. Salehi et al. [44] reviews possible modern techniques for solving the generic problem of novelty/OOD

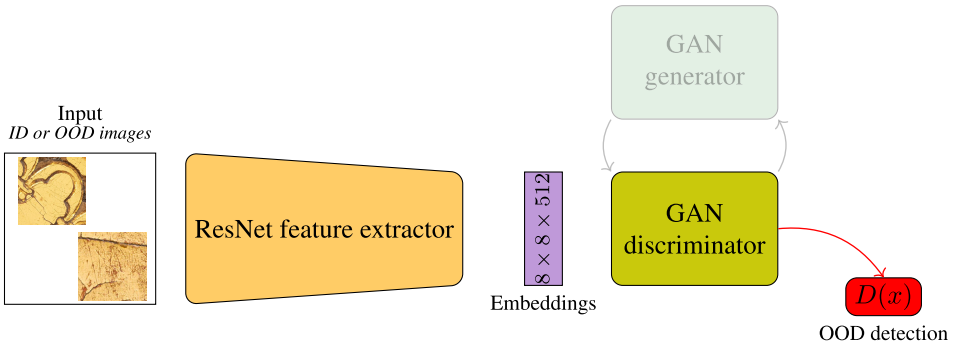


FIGURE 11. Scheme of the framework for OOD detection using OpenGAN. The data is processed by the feature extractor of the ResNet (as of Section III-B1 and, more specifically, Figure 10) to produce the embeddings. These embeddings are then passed on to the GAN discriminator, used as an OOD detector. It produces an output $D(x) \doteq s \in [0, 1]$ which is inversely proportional to the confidence that the discriminator has in the data being OOD. The generator is depicted with transparency as it does not come into play during the test phase. For an optimal computation, the embeddings can be used for operating both the classification (by passing them to the remaining portion of the ResNet) and the OOD detection.

detection. The authors distinguish between DL-based and non-DL-based approaches. The former can be coarsely categorized in:

- *generative-based approaches*: make use of a generator (e.g., a Generative Adversarial Network—GAN— or a Variational Autoencoder) to model the data distribution;
- *discriminative-based approaches*: train a classifier to distinguish between in- and out-liers.

We elected to experiment with (a) OpenGAN [3], a technique which is in-between these two approaches, which makes use, during training, of a GAN whose discriminator (which performs the OOD detection) is overfitted to the training set distribution, and (b) with the II-loss [4], a discriminative-based approach used to train an ANN which produces embedding in which in-distribution (ID) and OOD samples are enforced to be well-separated and can be told apart, e.g., by considerations based on metrics calculated within this embedding space. The advantage of this latter work is that the same ANN is responsible for both the classification and the OOD detection; it is not necessary to train multiple models (as it happens instead with OpenGAN). On the other hand, it might happen that the model trained for OOD detection may not be able to match, in terms of accuracy, the performance of the original model; this problem does not occur for OpenGAN, as the OOD detection is operated by a separate model, while the original model is still employed for classification.

a: OpenGAN

OpenGAN [3] is a technique for recognizing outlier data which revolves around the use of a GAN for modelling the distribution of the data within the training set. A GAN [45] is an ML model based on two entities, the *generator* and the *discriminator*. The former is tasked with generating feasible data, while the latter is trained for distinguishing between real and fake data. In OpenGAN, the focus is placed on the discriminator D, which is overfitted on the training data distribution, in the hope that, after the completion of the training, it can be used as a binary classifier for recognizing OOD samples. The generator G, instead, is used only for the training phase and is subsequently disregarded. The idea underlying OpenGAN is that, if the generator can effectively learn the distribution of the training data (i.e., the punches from our dataset) and that the discriminator is sufficiently powerful to avoid being ***fooled*** by the generator, the discriminator will then be also able to tell apart the ID and the OOD data. Formally, given a distribution of ID data $\mathcal{D}_I$ and a distribution of OOD data $\mathcal{D}_O$, we denote with $\boldsymbol{x}$ a real data point from the trainig set, $\tilde{\boldsymbol{x}}$ is a real OOD data, and $\boldsymbol{z}$ is a generic embedding acting as input for the generator, usually drawn from a multidimensional Gaussian distribution $\mathcal{Z}$. The GAN can be trained according to the following minimax problem:

$$\max_D \min_G \mathbb{E}_{x\sim\mathcal{D}_I}[\log(D(\boldsymbol{x}))] + \lambda_O \mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathcal{D}_O}[\log(1 - D(\tilde{\boldsymbol{x}})] + \lambda_G \mathbb{E}_{\boldsymbol{z}\sim\mathcal{Z}}[\log(1 - D(G(\boldsymbol{z}))] \quad (1)$$

where λ_O and λ_G are two scalars used to control the contribution to the training of the outliers and the fake data respectively. By setting λ_O to 0 and λ_G to 1, we can recover the original minimax formulation for GAN training [45]. At inference time, the discriminator produces a scalar $s \in [0, 1]$ which can be interpreted as the confidence of the input being ID. A threshold t_s can hence be determined over the confidence below which the input is classified as OOD. An additional improvement of OpenGAN with respect to its predecessors (like [46], [47]) is the nature of the data points (x and $\tilde{x}$ in Equation (1)): while most of the predecessors used raw data (i.e., raw images) as input to the discriminator and output of the generator, Kong and Ramanan [3] propose using data embeddings as inputs. These might be, for example, intermediate activations of neural networks, like our CNN classifiers. A diagram of this framework is presented in Figure 11.

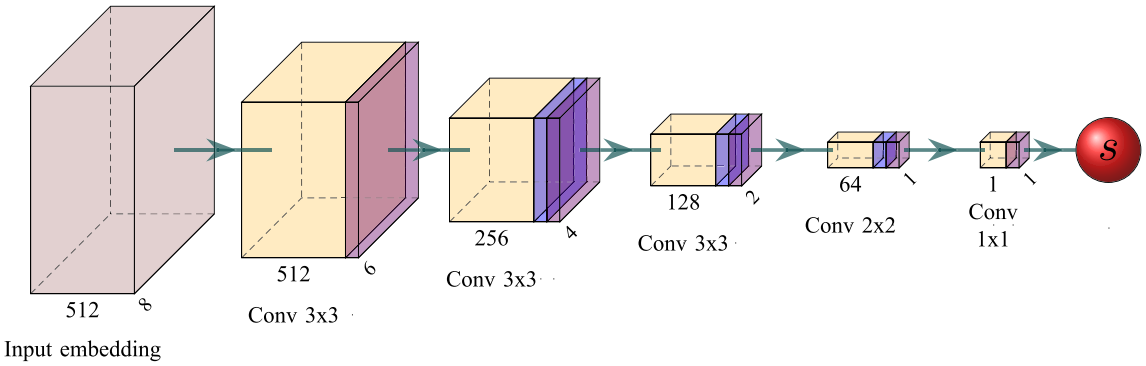


FIGURE 12. The discriminator used in our implementation of OpenGAN. The input, with shape 8 × 8 × 512, is passed through a cascade of convolutional layers (depicted in yellow) that reduce both its spatial dimensions and its number of channels. BN (in blue) is applied to all outputs of convolutional layers besides the first and the last ones, then Leaky ReLU (in green) is applied. The final output is a 1 × 1 × 1 tensor which passes through a sigmoid to produce a scalar *s*, the final output of the discriminator.

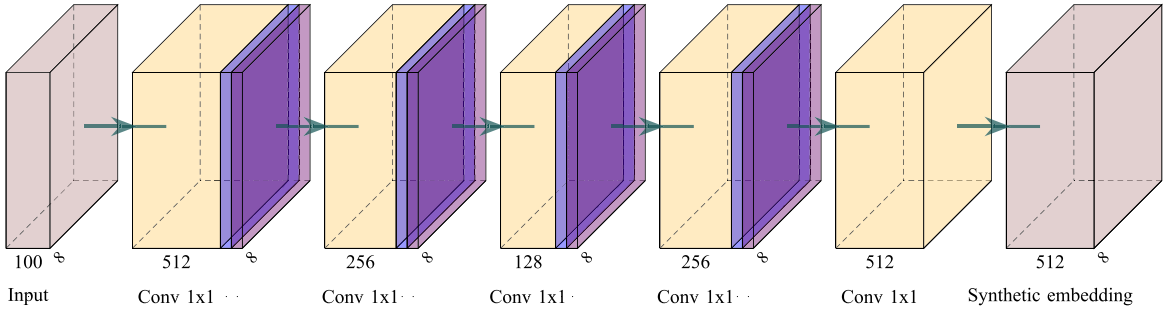


FIGURE 13. The generator used in our implementation of OpenGAN. The input, with shape 8 × 8 × 100—usually randomly sampled from a Gaussian—is passed through a cascade of 1 × 1 convolutional layers (depicted in yellow) that leave the spatial dimensions untouched, while varying the number of channels. Each layer, apart from the last one, apply BN (in blue) and the ReLU activation function (in green) after the convolutions are performed. The final output has size 8 × 8 × 512, which is the dimension of the embeddings which are fed to the discriminator shown in Figure 12.

The GAN we used is inspired by [3]. The diagram of the discriminator is presented in Figure 12, the generator in Figure 13.

b: II-LOSS

II-loss is a cost function introduced by Hassen and Chan [4] which they propose to use in addition to the regular CE loss to operate an *OOD-aware* classification. As opposed to OpenGAN, the methodology proposed by the authors does not need an additional model to work and requires very little architectural modifications with respect to what has been presented in Section III-B1. Taking as reference the ResNet18 architecture presented in Figure 10, the procedure acts by using two linear layers instead of one after the cascade of basic blocks and average pooling. The first linear layer embeds the incoming data into a space $\mathcal{Z}$ with a dimension which is fixed *a-priori*. Then, another linear layer projects the data from $\mathcal{Z}$ to the K-dimensional space for the usual classification. We suppose that we are in the process of training the network using stochastic gradient descent (SGD), and that we have a batch of size B, each category being represented by a variable amount B_j of datapoints, such that $\sum_{j=1}^{K} B_j = B$. We indicate with $\boldsymbol{z}_i$ the projection of the data point $\boldsymbol{x}_i$, operated by the ResNet18, onto $\mathcal{Z}$. Let us also indicate with $\boldsymbol{\mu}_j$ the batch mean of the embeddings of data belonging to class j. The II-loss is formulated as follows:

$$\text{II-loss} = \overbrace{\frac{1}{B}\sum_{j=1}^{K}\sum_{i=1}^{B_j}\left(||\boldsymbol{\mu}_j - \boldsymbol{z}_i||_2^2\right)}^{\text{Intra-class spread}} - \overbrace{\min\left\{\delta, \min_{m,n\in\{1,\dots,K\}} ||\boldsymbol{\mu}_m - \boldsymbol{\mu}_n||_2^2\right\}}^{\text{Inter-class separation}} \quad (2)$$

where $\delta \in \mathbb{R} \cup \{+\infty\}$ is a hyperparameter. The original formulation of the loss did not include the outer minimum and δ in the inter-class separation. We found beneficial to add these terms in order to limit the growing of the inter-class separation during training. The II-loss is basically a contrastive-learning [48] inspired loss which has the final goal of pulling together into clusters points belonging to the same class ("Intra-class spread") while pushing apart clusters representing different classes ("Inter-class separation"). In essence, the goal is to produce a CNN which operates an ***additional step*** w.r.t. the one presented in Section III-B1. In this additional step, the images are projected into a space, denoted

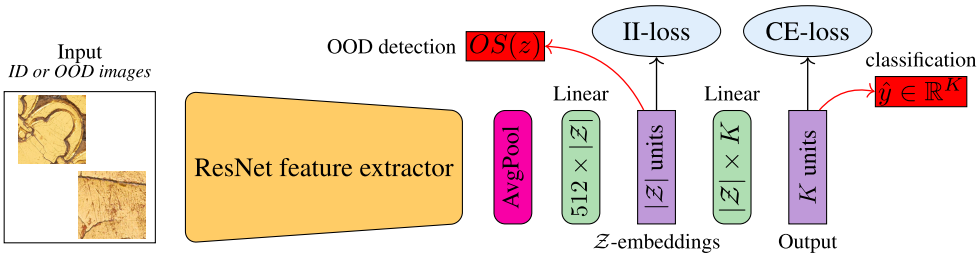


FIGURE 14. Scheme of the framework for OOD detection using the II-loss. We modify the ResNet architecture presented in Section III-B1 (see Figure 9 for a detailed reference) by adding an additional linear layer after the average pooling ("AvgPool"). This produces the intermediate embeddings in the space $\mathcal{Z}$ ("$\mathcal{Z}$-embeddings"), which are used to calculate the II-loss according to Equation (2). The subsequent linear layer acts, as in the canonical ResNet architecture, by projecting the information onto the K output nodes, on which the CE loss for classification is applied. At test time, the $\mathcal{Z}$-embeddings are used to calculate the outlier score ("*OS*") to determine whether the test data is OOD, while the classification is applied to the output of the model by picking the category as $\arg\max\{\hat{y}\}$.

as $\mathcal{Z}$, where the ID data fall into clusters, while OOD data are projected ***far*** from these clusters. A criterion based on a distance metric, like the Euclidean norm, can be used to tell apart ID and OOD images in the space $\mathcal{Z}$. For instance, Hassen and Chan [4] propose an *outlier score* (OS) formulation to classify a datapoint $\boldsymbol{x}^*$ as OOD. It is calculated as the minimum Euclidean norm from the point embedded onto $\mathcal{Z}$ and all of the clusters centroids obtained during training:

$$OS(\boldsymbol{z}^*) = \min_{j\in\{1,\dots,K\}} ||\boldsymbol{\mu}_j - \boldsymbol{z}^*||_2^2$$

where $\boldsymbol{z}^*$ is the embedding of $\boldsymbol{x}^*$ onto $\mathcal{Z}$. High values of $OS(\boldsymbol{z}^*)$ indicate that $\boldsymbol{x}^*$ is likely to be OOD. A threshold t_{OS} is calculated over OS to determine how high this value needs to be in order to label a datapoint as OOD. The II-loss can be combined with an additional classification loss, like CE, to train the model with the dual objective of a good classification performance and a space $\mathcal{Z}$ with the aforementioned property. A diagram of this framework is shown in Figure 14.

In the present work, we applied the framework for II-loss in conjunction with a modified ResNet18 CNN. We removed the final fully-connected layer (with 512×19 connections) and replaced it with two fully-connected layers: the first one with 32 hidden units (whose span is $\mathcal{Z}$, the embedding dimension), the second one with 19 neurons. The two layers do not have an activation function, thus functioning as linear projections.

IV. RESULTS

Irrespective of the specific architecture, all the classifiers we implemented provide, at inference time, two values as output:

- a vector of confidence $\hat{\boldsymbol{y}} \in \mathbb{R}^{19}$ corresponding to the output of the classifier, each element $\hat{y}_i$ of the vector specifying the confidence of the image belonging to the i-th class;
- a scalar $s \in \mathbb{R}^+ \cup \{0\}$ specifying the confidence that the image is an ID data.

In the following we report the results in terms of classification and OOD detection.

A. CLASSIFICATION

We trained the ResNet18, ResNet34, and ResNet50 classifiers with SGD in conjunction with the RAdam optimizer [49] for 20 epochs with a mini-batch size of 32, this number being imposed by hardware limitations. Despite RAdam being an optimizer with adaptive LR, we found beneficial to operate a multi-step decay by a factor of 0.1 at epochs 10 and 15. Without any form of data augmentation, the CNN was able to converge to 100% accuracy on both the training set and the validation set in all three cases of ResNet architectures. We experimented with fewer training epochs, but the corresponding CNNs were unable to perfectly classify the examples in the test-set. Moreover, concerning the LR decay, the application and the epochs at which we applied it were inspired by other works, e.g. [50], where LR decay is performed on adaptive optimizers with additional benefits on convergence. We noticed a faster convergence when employing it in the present work. We then proceeded to select the least complex of these models, the ResNet18, as our final model, which also recorded 100% accuracy on the test set. We will provide a critical comment on these results on Section V.

B. OOD DETECTION

For the sake of having one single metric for operating model selection, we considered the weighted harmonic mean of sensitivity and specificity, two common metrics used for evaluating the performance of binary classifiers:

$$W = 5 \cdot \frac{\text{specificity} \cdot \text{sensitivity}}{4 \cdot \text{specificity} + \text{sensitivity}}. \tag{3}$$

This formulation, inspired by the F2-score [51], allows to mediate between sensitivity and specificity, giving more importance to the former: we operated this choice by imagining that users of our system might prefer models committing a larger amount of false positives (i.e., punches which are incorrectly classified as OOD) with respect to false negatives (i.e., non-punches—like cracks—incorrectly classified as punches). The reason for this is that, usually, multiple

TABLE 3. Criteria used for model assessment on the test set and the OOD test set.

Test set OOD specificity	Accuracy of detecting OOD samples within the regular test set (i.e., how many punches are actually identified as such).
Test set OOD sensitivity	Accuracy of detecting OOD samples within the OOD test set (i.e., how many data from the OOD test set are actually identified as being OOD).
W	Weighted harmonic mean of sensitivity and specificity (see Equation (3)).
Gross test set accuracy	Accuracy on the test set punches including also those which were wrongly identified as OOD.
Net test set accuracy	Accuracy on the test set punches removing those which were wrongly identified as OOD.
Overall test set accuracy	Accuracy on the test set punches considering those which were wrongly identified as OOD as belonging to an additional class (i.e., the test data flagged as OOD is *always* misclassified).
OOD specificity on random crops	Accuracy of detecting OOD samples on the random crops dataset.
OOD specificity on random images	Accuracy of detecting OOD samples on the random images dataset.

instances of the same punch are present within the same artwork, thus it is not problematic if a small portion of these are erroneously identified as OOD; the opposite case might instead be tricky, as a user of our pipeline might get notice of the presence of unexpected categories of punches in the artwork he/she is analyzing, while these ones might be, for instance, cracks or some categories of punches not included in the training set.

In order to operate model selection, we used the validation dataset for the negative class; for what concerns the positive class (i.e., OOD samples), we created a small additional dataset composed of 36 pictures of punches not belonging to the original dataset presented in Section III-A. This dataset was further split equally in a training and a testing OOD datasets: the training OOD set was used for model selection, while the test OOD set was used for assessing the final performance of the selected models. Notice that Equation (3) does not consider the cardinality of the two groups (positive vs negative), as doing so would cause the negative group to be predominant in the calculation of the metric, since the training OOD set has a size of only 18, while the validation set has 305 images in it. In addition, we obtained two additional datasets for the sake of further assessing the OOD recognition performance of our approach: one composed of 15 *random crops* operated on some artworks from our dataset in places where punches are not present and another one composed of 500 *random images* of size $256 \times 256 \times 3$.

Regarding the model assessment on the test sets, Table 3 presents the criteria that we will be using during this phase.

1) OpenGAN

As far as the GAN is concerned, we evaluated experimentally a wide range of hyperparameters, like the learning rate (LR) of both the generator and discriminator, the number of epochs, and architectural elements such as the dimension of the incoming data (e.g., by downscaling the neural embeddings to dimensions such as $2 \times 2 \times 512$) or the dimension of the latent space which serves as input to the generator. Eventually, the best architectures for the generator and the discriminator were the ones depicted in Section III-B2. The optimizer used was Adam [52] for both the generator and the discriminator. The best LRs we found were 0.0001 for the generator, 0.000 02 for the discriminator. We applied no LR annealing. The other hyperparameters for Adam were set at 0.999 for β_1, 0.5 for β_2, as in the work by Kong and Ramanan [3]. For a more stable training, we applied label smoothing by perturbing the ground truth by a small amount $\epsilon = 0.15$ uniformely sampled, for each point in the batch, from the interval $[0, \epsilon]$. Label smoothing is a common tool employed to stabilize the training of GANs [53]. The selected number of epochs for this hyperparameters configuration was 900, although training more might give even better results. The validation metrics for the best model are presented in Table 4. For the sake of conciseness, we omit showcasing the results for other hyperparameters configurations. The optimal threshold t_s for the confidence value s was obtained by a grid search over the interval (0.0, 1.0) with increments of 0.01, each time calculating the resulting value of W. The results here presented were all obtained with $\lambda_O = 0$ and $\lambda_G = 1$, i.e., we did not use the OOD dataset in the training phase of the GAN. Experimenting with different values of λ_O did not give fruitful final results in our analysis.

TABLE 4. Sensitivity, specificity, and *W* for the best configuration of hyperparameters for OpenGAN.

Optimal t_s	Sensitivity	Specificity	W
0.77	1.000	0.9968	0.9993

OpenGAN is able to tell apart almost perfectly the validation data and the OOD training data, if not for one single data point per category. Figure 15 plots a histogram of the confidence values produced by the discriminator for both the validation dataset and the OOD training set.

From the chart, it is clear how the choice of W as a metric for selecting the threshold favors a higher value, that perfectly classifies the OOD dataset as such, disregarding lower values, which instead would cause a better result on recognizing the validation training data as non-OOD instead.

Test results for OpenGAN are presented in Table 6 alongside with those for the II-loss, in order to enable a faster comparison between the two techniques.

2) II-LOSS

For what concerns the II-loss, we trained 19 different versions of our modified ResNet18, each time choosing a different value of δ from Equation (2). Values ranged from 0.1 to 10. The models were trained for 10 epochs and a LR of 0.001 without annealing. We *fine-tuned* the ResNet18 we previ-

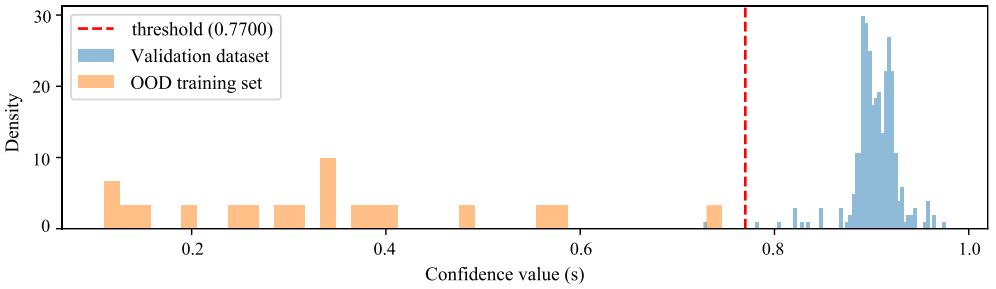


FIGURE 15. Histogram of the confidence value of OpenGAN's discriminator when evaluated with data from the validation dataset (blue) and OOD training set data. The validation data cluster in the interval [0.8, 1.0], with a few exceptions, while the OOD data is well scattered in below that level. A t_s of 0.77 is able to separate the two categories well, except for a single validation data point below this value.

TABLE 5. Results for the OOD detection capability of the modified ResNet18 CNN trained on the training set with different values of the margin δ of the II-loss (see Equation (2)). The best model, as far as W is concerned, is the one with $\delta = 0.5$, although other models reach similar values. In almost all cases, the metric favors a relatively low threshold for OS, below which no sample from the OOD training set is to be found, admitting a small OOD detection error in the regular training set.

δ	Optimal t_{OS}	Sensitivity	Specificity	W	Validation set accuracy
0.1	4.4066	0.9474	0.9649	0.9508	0.9968
0.2	1.4047	1.0000	0.9137	0.9815	0.9936
0.3	2.6089	1.0000	0.9712	0.9941	0.9968
0.4	3.6065	0.9474	0.9744	0.9526	0.9968
0.5	**2.8008**	**1.0000**	**0.9776**	**0.9954**	**0.9936**
1.0	0.2004	0.9474	0.6933	0.8827	0.9968
1.5	3.0007	1.0000	0.9744	0.9948	1.0000
2.0	1.8015	1.0000	0.9457	0.9886	1.0000
2.5	2.2040	1.0000	0.9521	0.9900	0.9936
3.0	1.2046	1.0000	0.9265	0.9844	0.9968
3.5	1.2010	1.0000	0.8818	0.9739	0.9968
4.0	1.4048	1.0000	0.9169	0.9822	1.0000
4.5	0.4014	1.0000	0.7827	0.9474	0.9968
5.0	4.0116	1.0000	0.9649	0.9928	0.9968
6.0	3.8035	1.0000	0.9712	0.9941	0.9968
7.0	1.6018	1.0000	0.9073	0.9800	0.9936
8.0	2.2008	1.0000	0.9329	0.9858	0.9904
9.0	1.8015	1.0000	0.9233	0.9837	0.9840
10.0	2.4143	1.0000	0.9457	0.9886	0.9904

ously trained for classification (see Section III-B1): due to the inconsistency in the number and size of the final fully-connected layers, the parameters of these last two layers in the latter CNN were initialized according to the Kaiming uniform initialization [54]. Again, no data augmentation was performed.

For establishing an optimal value of δ from Equation (2), we ran a grid search over values ranging from 0.1 to 10.0 (see Table 5). Given the excellent validation accuracy—as far as punch recognition is concerned—of each of these models (all above 98%), we proceeded to operate OOD recognition by means of the OS presented in Section III-B2. We evaluated the values of OS for the samples in the validation dataset and the training OOD set. In order to find an optimal threshold over OS—t_{OS}—for each of the models, we ran a grid search over the possible values of OS (from 0 to the maximum recorded for the given model), each time computing W, the weighted harmonic mean of sensitivity and specificity. We proceeded to select the model recording the highest value of this metric as the candidate solution for our task. As we can see from the results reported in Table 5, the best model is the one with $\delta = 0.5$ and a t_{OS} of 2.8008, which records a W of 99.54%, with a specificity of 97.76%, a sensitivity of 100.0%, and a validation set accuracy of 99.36%.

Figure 16 showcases the behavior of the modified ResNet18 with $\delta = 0.5$ for what concerns the OS's of both the validation dataset and the OOD training set. We can notice that the samples from the validation set display a low value of OS, meaning that they are very close to the clusters representing the punches categories, apart from some individual cases, which are hardly noticeable in the chart. Conversely, the OOD samples are all above the threshold: aside from a small bunch of data which are close to it, the great majority of them seem to disperse in $\mathcal{Z}$ well far from the clusters.

Test set results are presented in Table 6. Similarly to the validation results, we also get an OOD specificity of 100%, meaning that the model is effectively able to identify OOD data as such. The OOD sensitivity is instead a bit lower than the validation (from 97.76% to 95.71%), albeit the result is still good. Concerning the categorization performance, the model misclassifies 2 test set examples, thus achieving an accuracy of 99.74%. The misclassified data are though part of a subset of the 33 images from the test set which are erroneously recognized as OOD by the model; hence, when removing these 33 data from the test set, the model achieves perfect accuracy. Thus, the overall test set accuracy, which considers the wrongly marked OOD data as a category of its own, coincides with the OOD sensitivity of 95.71%. The additional evaluation on OOD data shows that the model behaves well at recognizing as OOD both random crops and random images, thus strengthening the conclusion that the embedding space $\mathcal{Z}$ is excellent for clustering ID data, while keeping OOD data far from the clusters.

Comparing both techniques for OOD detection, it is immediately clear they perform very well, both considering the

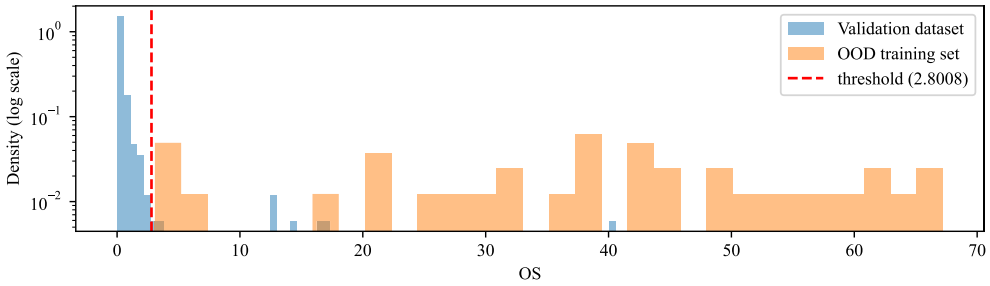


FIGURE 16. Histogram of OS for the modified ResNet18 with $\delta = 0.5$ for the samples in the validation dataset (blue) and the OOD training set (orange). Log scale is used on the y-axis to mitigate the influence of the distribution of the validation data, which has a huge spike around 0, and improve the visualization of the very flat distribution of OOD data. The validation samples stay close of the training set categories clusters, while OOD data scatters well above this threshold.

TABLE 6. Test sets results for the two models for OOD detection. Metrics introduced in Table 3. The classification part form OpenGAN is performed by the same CNN whose results are presented in Section IV-A, hence the results are directly derived from there.

Metric	OpenGAN	II-loss
Test set OOD specificity	1.0000	1.0000
Test set OOD sensitivity	0.9948	0.9571
W	0.9990	0.9911
Gross test set accuracy	1.0000	0.9974
Net test set accuracy	1.0000	1.0000
Overall test set accuracy	0.9948	0.9571
OOD specificity on random crops	1.0000	1.0000
OOD specificity on random images	0.9990	1.0000

detection itself and the punches classification. While the OpenGAN discriminator seems to work better on the punches themselves, the additional analysis on random images highlights that the custom ResNet18 trained with the II-loss might perform a bit better at generically detecting OOD data, as indicated by the additional experiments on random images. This can be confirmed by looking more into detail at the distributions of s for OpenGAN and OS for the II-loss, which are presented in Figure 17.

From the image, we can see how the II-loss seems to favor clustering the random images very closely around 5, with very little variability; on the other hand, the distribution for the output of the discriminator seems much more scattered: this could suggest that the custom ResNet18, which by itself is tasked with operating a clustering due to the II-loss, may be semantically more aware of what a punch is than the discriminator, which instead is only tasked with outputting a confidence value.

V. DISCUSSION

A. GENERALIZATION CAPABILITY OF THE CNN

The CNN we trained for classification shows remarkable results as far as accuracy is concerned, especially considering that we did not employ data augmentation: it would seem that the dataset is sufficiently variable as to let the model learn an equivariance to rotation, which is a natural property of CNNs [55]. It is also important to note that, as mentioned in Section III-A, the model seems to work very well in presence of imbalance in the dataset, i.e., a large number of frames belong to a small subset of categories. Despite this seemingly excellent behavior, we need to take into account that the 3815 frames were extracted from a very small sample of only 4 paintings, out of 1015 which were meticulously analyzed and catalogued by Skaug [1]. The majority of the punches are appearing multiple times throughout the painting: for instance, the punch #729 is repeated 613 times in the artwork SLF, albeit with little variation between the different samples. There might be situations in which the same punch in separate art pieces might differ substantially between each other mainly due to different states of preservation: in these circumstances, if one of these artworks would happen not to be in the training set, it might be possible that the CNN fails to recognize these punches. In order to test whether this is the case with our data, we re-arranged the frames in our original dataset such that all of the punches in the painting MCB were moved from the training to the test set. At the same time, frames from the involved punch classes not belonging to MCB were all moved to the training set. The selection of MCB as candidate painting for this experiment is due to the fact that it shares the majority of punches with other paintings (see Table 2 for reference) except for punches 438 and 659, which account for a measly total of 39 images. SLF and SFA, on the other hand, come from the same diptych, and hence we deemed the distribution of their punches to be too similar for considering one of them as candidates for the removal. Operating this removal with other paintings would have caused more categories or too many examples to be removed from the dataset. After the removal, we proceeded to re-train the CNN as indicated in Section III-B1 on this new dataset. The result in terms of accuracy were mixed: despite the CNN still performing perfectly on a large number of categories, on other specific punch classes the accuracy was as low as 10%, as showcased in Table 7. To alleviate this downfall on some categories, we experimented with using more powerful CNNs like ResNet34 and ResNet50 and with augmenting the dataset via simple random geometric

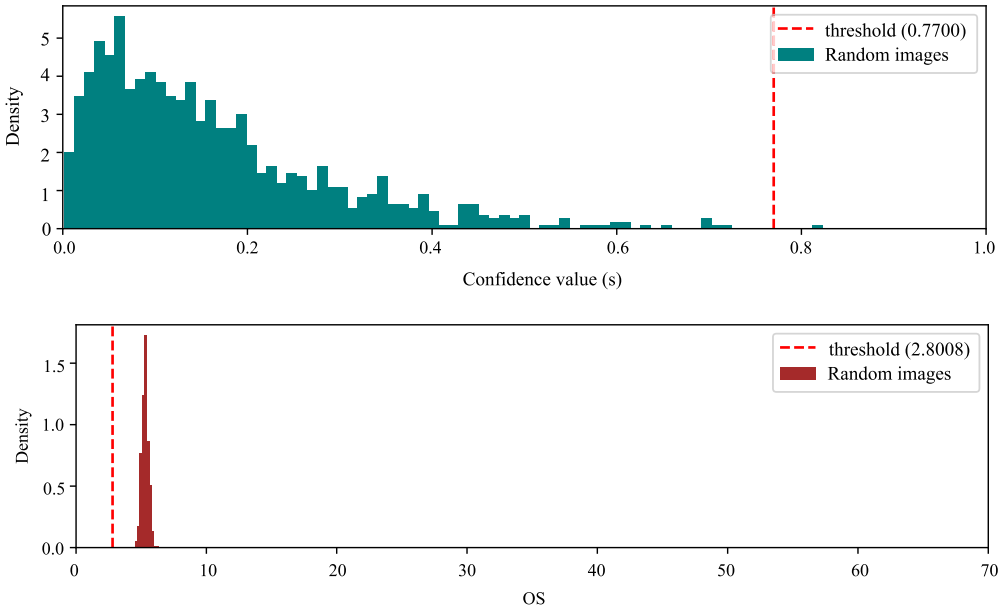


FIGURE 17. Distribution of *s* (top) for the OpenGAN discriminator and *OS* (bottom) for the custom ResNet18 when evaluated with random images. The range for OS is much larger as it reproduces the range showcased in Figure 16. All in all, the II-loss seems to do a better job at clustering together random data, whose corresponding OS is never very close to the threshold; on the other hand, despite the majority of the values of *s* lie below 0.4, some instances seem to leak closer to the threshold, and eventually one crosses it.

TABLE 7. Test accuracy for punches categories belonging to the painting MCB (see Table 2 for more info) for our experimental dataset described above. Results for the other categories not shown for conciseness. Entries in red indicate accuracy below 80%, while entries in blue indicate a non-perfect accuracy from 80% upward. Entries in black indicate perfect accuracy.

Punch ID	# correct	# instances	Accuracy
43	1	4	0.2500
72	50	50	1.0000
96	15	15	1.0000
178	16	16	1.0000
371	8	80	0.1000
385	40	43	0.9302
438	4	4	1.0000
619	4	5	0.8000
655	45	114	0.3947
659	5	5	1.0000

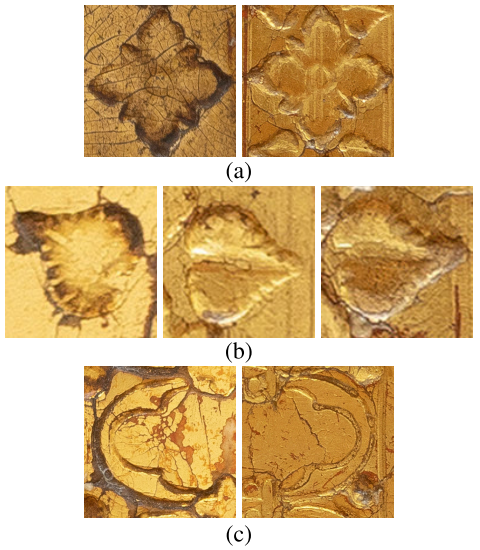


FIGURE 18. Sample images from the experimental dataset: (a) Punch 371: sample from SLF (training set) and misclassified sample from MCB (test set). Despite the similar shape, the different lighting and conservation status renders the classification a hard task for the CNN; (b) Punch 655: sample from SFA (training set) and two samples from MCB, the first one misclassified, the second one correctly classified. Despite the overt differences in conservation status of the punches between SFA and MCB, the CNN is able to classify correctly 40% of the frames from MCB; (c) Punch 72: sample from SLF and MCB. Despite the difference in domain between the images from the two artworks, the CNN correctly classifies all the frames from MCB belonging to this category.

transformations like rotations, gaussian blur, patching, color jittering, albeit getting no substantial improvement over the CNN trained without augmentations.

After analyzing the punches that the CNN failed to recognize, we realized that the frames from MCB subject to misclassifications were very much different from the ones in the training set. Figure 18 showcases some samples from the critical categories presented in Table 7.

In all images above, the substantial difference between the frames is due to a dramatic discrepancy in the preservation status—with the addition of a difference in overall lightning conditions—of the two art pieces in the areas where the punch was impressed. This specific problem is tightly linked to the issue of generalization to unseen data and, more specifically, connects to the difficulty that some ML-based models, chiefly ANNs, have in recognizing data which are (sometimes even slightly) OOD [43]. The solution to this problem is not trivial

and can follow multiple paths: one possibility would be to increase the size of the dataset by including a considerably larger number of paintings, thus increasing the intra-category variability. This would certainly be a feasible solution in terms of available material, although it would be necessary to spend a large amount of time into labeling the dataset. On the other hand, a different approach may be finding more sophisticated augmentation techniques (e.g., working with transformation over grayscale or binarized images, or even in different domains; or working with automatic augmentation techniques like those contained in Albumentations [56]) to try increasing the generalization capability of the CNN.

Note that this analysis is intended as a proxy for a more complicated and time-consuming k-fold cross-validation. Cross-validation works by randomly assigning some images to bins without taking into consideration their painting. Taking into consideration the structure of a dataset (see Table 2), though, we posit that a more effective evaluation on the generalization capability of the proposed model is to remove altogether *all* of the punches belonging to one painting from the training dataset. Cross-validation may leave some categories of punches with too few data for training, while not ensuring, in our case, to fulfill the task of assessing the generalization.

B. REPRODUCIBILITY OF DATA GATHERING

Another aspect that is worth discussing is the process of data gathering. In the present work, the photographies of the artworks were operated by a professional photographer with the use expensive equipment and following a precise and time-consuming post-processing methodology (explained in Section III-A1) aimed at rendering all of the images with the same exact reproduction scale with respect to the original paintings. This eases a lot the work of training the CNN, as it does not need to incorporate an equivariance to scale, which is usually enforced in the model via data augmentation. On the other hand, the CNN is hardly usable with images collected with different methodologies: we envisioned our models to be an aid to experts in the context of professional analyses of panel paintings, not tools to be used generically, e.g., by museum visitors taking pictures with their smartphones.

C. TRAINING METHODOLOGY AND ASPECT RATIO OF IMAGES

In Section III-A, we cited how frames from different categories have different sizes and aspect ratios—this is due to the fact that different punches have different shapes, as is also visible in Figure 5. We also cited how we trained the CNNs by artificially resizing all the punches to the common size of 256×256 in order to speed up training. This is a very common methodology in DL, as *packing* it allows to pack a mini-batch of images into a 4d tensor of fixed size, which can be evaluated by the CNN much faster than passing the data sequentially. In our case, though, this means that the images that the CNN sees are probably not in the original aspect ratio, as many frames in the dataset are not perfectly square. While this is probably not a point for concern, the resizing to a common size may forcibly induce in the CNN an equivariance to scale which might be unwanted. Let us think at two punches with the same shape, only with one which is n times larger than the other one. If this situation occurred, the CNN would probably have a hard time in telling one apart from the other as, with the resizing, the two punches would have the exact same shape *and* size when fed into the model. To the best of our knowledge, this situation, fortunately, does not seem to happen, at least in the context of punches catalogued by Skaug [1].

FIGURE 19. Example of a composition of two punches (# 72 and # 655) repeated closely multiple times to create an intricate pattern. This image could potentially represent a source of misclassifications, as the model could have difficulties in categorizing it in a single class.

D. OOD DETECTION

In Section III-B2, we presented two possible solutions to the approach of detecting OOD instances: OpenGAN and the II-loss. Both methodologies showcase an excellent capability of telling apart OOD samples, both with data coming from generic panel paintings and random data, albeit small errors occur that slightly impact the classification accuracy. This does not mean that additional experiments, with further hyperparameter exploration (e.g., with λ_O from Equation (1)) or using different techniques, are not to be run, and may represent a direction for further research.

E. IMAGE CLASSIFICATION OR OBJECT DETECTION?

The pipeline of models implemented in the present work performs image classification, i.e., given an image, we wish to classify this image in a unique class, even if this image may contain instances of multiple categories. The effect of this premise is that a user of our tool should first crop the images of punches that he would like to classify, instead of passing to the model a full-scale image of the whole painting. This, of course, has the potential to be a time-consuming task, although we might argue that, at inference time, only some instances of the same punch may be passed on to the model, instead of the full set of instances appearing in a picture, to determine the category to which these instances belong. Another weakness of this aspect is that care has to be put into the selection of the frames to be fed to the model, as there might be instances in which some punches may appear very close to each others, potentially representing a tricky scenario for the model itself. An example of a possible troubling situation is represented in Figure 19.

A solution to both of these weaknesses would be the utilization of techniques for object detection, that work by identifying—and coarsely localizing—an arbitrarily large number of instances of known entities in a single image. Although potentially this would represent a great improvement over our proposed pipeline, the main obstacle toward its adoption is the fact that the starting images have a very large resolution (even larger than 100 000 pixels per side. This renders the utilization of popular object detection models such as YOLO [57] or R-CNN [58] computationally unfeasible, as these models are designed to work with low-resolution images. Thus, the adoption of object detection techniques is not trivial and would require additional investigations, either with workarounds like sliding-window approaches, or with techniques specifically designed for high-resolution object detection [59], [60], which we reserve to experiment with in the future.

F. ADDITIONAL METHODS

In our analysis, in addition to CNNs, we experimented, although with severely underwhelming results, with additional classification techniques stemming from ''classical'' computer vision. Within this context, our problem of punches recognition can be solved by techniques such as *template matching*, where one or more representative *templates* of each class are used to gather a set of features. These features are then computed on images from a validation dataset and a matching algorithm is used to quantitatively compare the two sets. We first experimented with the shape-based matching algorithm developed by Steger [61]. Then, we investigated the use of the approach encompassing keypoints extraction using SIFT [62] or ORB [63], coupled with RanSAC [64] for outlier filtering, then operating the matching by means of rotating the set of filtered keypoints from the template onto the target image. Both techniques immediately showed to be highly sensitive to noise, which in our dataset comes mainly in the form of cracks and artifacts of bad preservation (as already noted in Section IV-B).

Other techniques which were commonly used for image classification before the widespread adoption of CNNs were pipelines composed of (a) a feature extractor for embedding the images in a semantically-meaningful space and (b) a (linear) classifier, such as a Support Vector Machine [19] (SVM), for attempting to separate the images in this embedding space. We experimented with (a) Hu moments [65] + SVM, (b) SIFT + SVM, and (c) Histogram of Oriented Gradients [66] + SVM. Overall, the best result we obtained from these classical techniques were with the combination of SIFT and SVM, which produced a test set accuracy of 62.80%, quite far from what was attained with CNNs.

VI. CONCLUSION

In the present work, we showcased a proof-of-concept for a system operating an automatic recognition of punches in a set of 3815 punches (i.e., repeated motifs mechanically impressed onto wooden panels) whose images were extracted and labeled from a small selection of 4 panel paintings from the 14th century. The system is composed of two blocks, one, based on a Convolutional Neural Network (CNN) which performs classification of said punches in 19 categories, and one which is aimed at supplying the prediction of the CNN with an estimate of uncertainty, with the intent of detecting whether the CNN is being evaluated with Out-of-Distribution (OOD) data, such as images of punches not comprised in the 19 original categories, or crops of other parts of a painting not containing a punch. For this part, we experimented with two techniques: OpenGAN, a GAN which is *overfitted* on the training data in such a way that the discriminator is able to tell apart OOD and non-OOD data, and the II-loss, a loss function which, in conjunction with the classical Cross Entropy, is able to produce a *pre-classification* neural embedding aimed at clustering together data from the same categories, while (hopefully) *pushing* OOD data far from these clusters. The CNN trained for classification is able to perfectly classify all of the images in the test set. For what concerns the detection of OOD samples, OpenGAN seems to work better than the II-loss—albeit with a very small difference—on data extracted from our images of panel paintings, while II-loss seems to work a bit better when considering additional OOD data obtained randomly.

The present work exhibits some weak points (e.g., slightly poorer performance on ablation studies on the dataset), which can be possibly traced back to the small dimension of the dataset and the number of artworks used to create it, or to the data collection procedures, which require professional photographic equipment and a meticulous protocol for taking and post-processing the pictures. In addition, we make the case for utilizing object detection models instead of image classification ones. These observations represent a point for future directions of this work.

We believe our analysis to be of aid for art historians or experts into building pipelines of model for helping their investigations into recognizing repeated motifs into artworks.

REFERENCES

[1] E. S. Skaug, *Punch Marks From Giotto to Fra Angelico: Attribution, Chronology, and Workshop Relationships in Tuscan Panel Painting C. 1330–1430*, vols. 1–2, IIC, Ed. Singapore: IIC, Nordic Group, Norwegian Section, 1994.

[2] M. S. Frinta, ''On the punched decoration in Medieval panel painting and manuscript illumination,'' *Stud. Conservation*, vol. 17, no. 1, pp. 115–121, 1972.

[3] S. Kong and D. Ramanan, ''OpenGAN: Open-set recognition via open data generation,'' in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 813–822.

[4] M. Hassen and P. K. Chan, ''Learning a neural-network-based representation for open set recognition,'' in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2020, pp. 154–162.

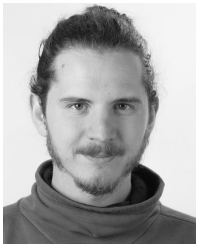
[5] K. He, X. Zhang, S. Ren, and J. Sun, ''Deep residual learning for image recognition,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[6] A. Ladis, ''Taddeo Gaddi: Critical reappraisal and catalogue Raisonné,'' *Reinassance Quart.*, vol. 3, no. 3, pp. 460–464, 1984.

[7] E. S. Skaug, "The St. Anthony abbot ascribed to Bartolo Di Fredi in the national gallery, London. A contribution to the attribution game," *Acta ad Archaeologiam et Artium Historiam Pertinentia. Institutum Romanum Norvegiae Roma*, vol. 6, pp. 141–150, Jan. 1975.
[8] E. S. Skaug, "The 'St. Anthony Abbot' ascribed to nardo di cione at the villa I Tatti, Florence," *Burlington Mag.*, vol. 117, no. 869, pp. 540–543, 1975.
[9] E. S. Skaug, "Punch marks—What are they worth? Problems of Tuscan workshop interrelationships in the mid fourteenth-century: The Ovile master and Giovanni da Milano," in *La Pittura Nel XIV e XV Secolo. Il Contributo dell'Analisi Tecnica Alla Storia Dell'arte*, vol. 3. Bologna, Italy: CLUEB, 1983, pp. 253–282.
[10] J. Steinhoff, "Artistic working relationships after the black death: A Sienese 'compagnia,' c. 1350–1363(?) ," *Renaissance Stud.*, vol. 14, no. 1, pp. 1–45, 2000.
[11] E. S. Skaug, "Notes on the chronology of Ambrogio Lorenzetti and a new painting from his shop," in *Mitteilungen des Kunsthistorischen Institutes Florenz*. Florence, Italy: Max-Planck-Institut, 1976, pp. 301–332.
[12] J. Polzer, "A question of method: Quantitative aspects of art historical analysis in the classification of early Trecento Italian painting based on ornamental practice," in *Mitteilungen des Kunsthistorischen Institutes Florenz*. Florence, Italy: Max-Planck-Institut, 2005, pp. 33–100.
[13] S. Kroner and A. Lattner, "Authentication of free hand drawings by pattern recognition methods," in *Proc. 40th Int. Conf. Pattern Recognit.*, vol. 1, Aug. 1998, pp. 462–464.
[14] T. Melzer, P. Kammerer, and E. Zolda, "Stroke detection of brush strokes in portrait miniatures using a semi-parametric and a model based approach," in *Proc. 40th Int. Conf. Pattern Recognit.*, vol. 1, Aug. 1998, pp. 474–476.
[15] J. M. Corridoni, A. D. Bimbo, S. D. Magistris, and E. Vicario, "A visual language for color-based painting retrieval," in *Proc. IEEE Symp. Vis. Lang.*, Sep. 1996, pp. 68–75.
[16] K. Hachimura, "Retrieval of paintings using principal color information," in *Proc. 13th Int. Conf. Pattern Recognit.*, Aug. 1996, pp. 130–134.
[17] I. Widjaja, W. K. Leow, and F.-C. Wu, "Identifying painters from color profiles of skin patches in painting images," in *Proc. Int. Conf. Image Process.*, Sep. 2003, p. 845.
[18] C. R. Johnson, E. Hendriks, I. J. Berezhnoy, E. Brevdo, S. M. Hughes, I. Daubechies, J. Li, E. Postma, and J. Z. Wang, "Image processing for artist identification," *IEEE Signal Process. Mag.*, vol. 25, no. 4, pp. 37–48, Jul. 2008.
[19] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[20] H. J. van den Herik and E. O. Postma, "Discovering the visual signature of painters," in *Future Directions for Intelligent Systems and Information Sciences*, vol. 45, J. Kacprzyk and N. Kasabov, Eds. Berlin, Germany: Physica-Verlag HD, 2000, pp. 129–147.
[21] L. Shamir, "Computer analysis reveals similarities between the artistic styles of van Gogh and Pollock," *Leonardo*, vol. 45, no. 2, pp. 149–154, Apr. 2012.
[22] E. Levy, O. E. David, and N. S. Netanyahu, "Genetic algorithms and deep learning for automatic painter classification," in *Proc. Annu. Conf. Genetic Evol. Comput.*, Jul. 2014, pp. 1143–1150.
[23] R. Royall, "A class of nonparametric estimators of a smooth regression function," Ph.D. dissertation, Dept. Statist., Stanford Univ., Stanford, CA, USA, 1966.
[24] M. Lettner, P. Kammerer, and R. Sablatnig, "Texture analysis of painted strokes," in *Proc. 28th Workshop Austrian Assoc. Pattern Recognit.*, Hagenberg, Austria, Jun. 2004, pp. 269–276.
[25] O. E. David and N. S. Netanyahu, "DeepPainter: Painter classification using deep convolutional autoencoders," in *Artificial Neural Networks and Machine Learning ICANN 2016*, vol. 9887, A. E. Villa, P. Masulli, and A. J. P. Rivero, Eds. Cham, Switzerland: Springer, 2016, pp. 20–28.
[26] W. R. Tan, C. S. Chan, H. E. Aguirre, and K. Tanaka, "Ceci n'est pas une pipe: A deep convolutional network for fine-art paintings classification," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3703–3707.
[27] E. Cetinic, T. Lipic, and S. Grgic, "Fine-tuning convolutional neural networks for fine art classification," *Expert Syst. Appl.*, vol. 114, pp. 107–118, Dec. 2018.
[28] S. Bianco, D. Mazzini, P. Napoletano, and R. Schettini, "Multitask painting categorization by deep multibranch neural network," *Expert Syst. Appl.*, vol. 135, pp. 90–101, Nov. 2019.
[29] C. S. Rodriguez, M. Lech, and E. Pirogova, "Classification of style in fine-art paintings using transfer learning and weighted image patches," in *Proc. 12th Int. Conf. Signal Process. Commun. Syst. (ICSPCS)*, Dec. 2018, pp. 1–7.
[30] C. Sandoval, E. Pirogova, and M. Lech, "Two-stage deep learning approach to the classification of fine-art paintings," *IEEE Access*, vol. 7, pp. 41770–41781, 2019.
[31] M. R. Mohammadi and F. Rustaee, "Hierarchical classification of fine-art paintings using deep neural networks," *Iran J. Comput. Sci.*, vol. 4, no. 1, pp. 59–66, Mar. 2021.
[32] F. Milani and P. Fraternali, "A dataset and a convolutional model for iconography classification in paintings," *J. Comput. Cultural Heritage*, vol. 14, no. 4, pp. 1–18, Dec. 2021.
[33] E. S. Skaug, *Giotto and the Flood of Florence in 1333: A Study in Catastrophism, Guild Organisation, and Art Technology*. Florence, Italy: Giunti, 2013, ch. 8.
[34] G. Bradski and A. Kaehler, "OpenCV," *Dr. Dobbs J. Softw. tools*, vol. 3, p. 120, Nov. 2000.
[35] Y. Ma and H. He, *Imbalanced Learning: Foundations, Algorithms, and Applications*. Hoboken, NJ, USA: Wiley, 2013.
[36] K. Fukushima, "Neural network model for a mechanism of pattern recognition unaffected by shift in position-neocognitron," *IEICE Tech. Rep., A*, vol. 62, no. 10, pp. 658–665, 1979.
[37] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a backpropagation network," in *Advances in Neural Information Processing Systems*, vol. 2, D. Touretzky, Ed. Burlington, MA, USA: Morgan-Kaufmann, 1989. [Online]. Available: https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf
[38] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, "Design of an image edge detection filter using the Sobel operator," *IEEE J. Solid-State Circuits*, vol. SSC-23, no. 2, pp. 358–367, Apr. 1988.
[39] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 818–833.
[40] C. Olah, A. Mordvintsev, and L. Schubert. (2017). *Feature Visualization*. Distill. [Online]. Available: https://distill.pub/2017/feature-visualization
[41] A. Ansuini, A. Laio, J. H. Macke, and D. Zoccolan, "Intrinsic dimension of data representations in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 6109–6119.
[42] 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.
[43] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 427–436.
[44] M. Salehi, H. Mirzaei, D. Hendrycks, Y. Li, M. Hossein Rohban, and M. Sabokrou, "A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges," 2021, *arXiv:2110.14051*.
[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 2672–2680.
[46] S. Pidhorskyi, R. Almohsen, and G. Doretto, "Generative probabilistic novelty detection with adversarial autoencoders," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 6823–6834.
[47] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, "Adversarially learned one-class classifier for novelty detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3379–3388.
[48] S. Becker and G. E. Hinton, "Self-organizing neural network that discovers surfaces in random-dot stereograms," *Nature*, vol. 355, no. 6356, pp. 161–163, Jan. 1992.
[49] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *Proc. [Online]. Available:Int. Conf. Learn. Represent.*, 2020. [Online]. Available: https://openreview.net/forum?id=rkgz2aEKDr and https://openreview.net/group?id=ICLR.cc/2020/Conference
[50] A. Renda, J. Frankle, and M. Carbin, "Comparing rewinding and fine-tuning in neural network pruning," 2020, *arXiv:2003.02389*.
[51] Y. Sasaki, "The truth of the F-measure," *Teach Tutor Mater*, vol. 1, no. 5, pp. 1–5, 2007.
[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[53] L. Weng, "From GAN to WGAN," 2019, *arXiv:1904.08994*.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[55] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.
[56] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, p. 125, Feb. 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/125
[57] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
[58] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.
[59] B. Uzkent, C. Yeh, and S. Ermon, "Efficient object detection in large images using deep reinforcement learning," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 1824–1833.
[60] C. Sun, Y. Ai, S. Wang, and W. Zhang, "Mask-guided SSD for small-object detection," *Int. J. Speech Technol.*, vol. 51, no. 6, pp. 3311–3322, Jun. 2021.
[61] C. Steger, "Similarity measures for occlusion, clutter, and illumination invariant object recognition," in *Proc. Joint Pattern Recognit. Symp.* Cham, Switzerland: Springer, 2001, pp. 148–154.
[62] G. Lowe, "Sift-the scale invariant feature transform," *Int. J.*, vol. 2, nos. 91–110, p. 2, 2004.
[63] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.
[64] M. A. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[65] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Inf. Theory*, vol. 8, no. 2, pp. 179–187, Feb. 1962.
[66] R. K. McConnell, "Method of and apparatus for pattern recognition," U.S. Patent 4 567 610, Jan. 28, 1986.

MARCO ZULLICH was born in Trieste, Italy, in 1990. He received the M.Sc. degree (magna cum laude) in data science and scientific computing from the University of Trieste, in 2019, where he is currently pursuing the Ph.D. degree in industrial and information engineering. From 2014 to 2018, he worked as a Business Analyst at Capgemini. In 2021 and 2022, he was a Teaching Assistant in deep learning with the University of Trieste. His research interests include deep learning applied to vision and model compression techniques.



VANJA MACOVAZ was born in Trieste, Italy, in 1990. He received the master's degree in image design from ISIA Urbino, Italy, with a thesis on the high-definition reproduction of mosaics and frescoes of the Santa Maria Assunta Basilica in Aquileia, Udine. He is currently pursuing the Ph.D. degree in digital humanities with the Department of History, Archaeology, Geography, Fine and Performing Art, University of Florence. He is also a Cultural Heritage Photographer with the Art History Department in Florence. His research interests include 3D photogrammetric acquisition, multispectral imaging, and high-resolution imaging in complex scenarios.

GIOVANNI PINNA was born in Trieste, Italy, in 1996. He received the M.Sc. degree in computer engineering from the University of Trieste, in 2022, where he is currently pursuing the Ph.D. degree in Applied Data Science and Artificial Intelligence. In 2021, he received top honors for his Erasmus Program at the University of Leoben, Austria. His research interests include the study of deep learning and computer vision.

FELICE ANDREA PELLEGRINO (Member, IEEE) was born in Conegliano, Italy, in 1974. He received the Laurea degree in engineering and the Ph.D. degree from the University of Udine, Italy, in June 2000 and May 2005, respectively. Since 2006, he has been working at the Department of Engineering and Architecture, University of Trieste, as an Associate Professor. His research interests include control theory, machine learning, and computer vision. He is currently a member of the Conference Editorial Board of the IEEE Control Systems Society and the European Control Association. He is also an Associate Editor of the IEEE Control Systems Letters.

Open Access funding provided by 'Università degli Studi di Trieste' within the CRUI CARE Agreement