Deep Convolutional Neural Networks for Archaeological Pottery Diagram Classification

TOMÁS OLIVEIRA

Archaeological fieldwork projects result in the collection of historical artifacts (e.g., pottery sherds) that need to be classified according to established categories, which in the area are referred to as typologies. These categories group objects with similar characteristics (i.e., similarity in the overall shape, in the character of component parts such as rims and handles, and in the technique and style of decoration), allowing archaeologists to ascertain the origin of pieces found in a specific location (if they are autochthonous), their age, or the age of the site. The categorization of pottery sherds is presently done by archaeologists through an entirely manual procedure, involving the analysis of standardized line illustrations. Since manual analysis raises problems for the timely categorization of a large number of artifacts, there is interest in automated approaches for suggesting typologies to archaeological artifacts. With this motivation, we present a set of state-of-the-art techniques, based on the use of convolutional neural networks, to automatically classify standardized black-and-white line diagrams of pottery sherds. We report on a comprehensive evaluation for the proposed approaches, discussing limitations associated with class imbalance or lack of large training datasets.

Fig. 1. Side-by-side example of an artifact's photograph and its corresponding black-and-white line diagram. Image obtained from an article by Stibbe about Laconian pottery [40].

1 INTRODUCTION

In archaeological fieldwork, one of the most common recorded artifacts is pottery vessels, often in the form of incomplete broken sherds [27]. These fragments enable to approach daily life of the societies that produced them, their customs, traditions, and constructions of their realities. Since its inception during the Neolithic, ceramic vessels are one of the most widespread materialities, particularly in pre-industrial contexts. In terms of conservation, pottery is also quite stable, if we compare it with metals or other organic materials. Potsherds in archaeological stratigraphies may answer many interrogations. Two of the most frequently asked questions are related to the dating of the piece (and also the context in which it was recorded) and its use. During the twentieth century, numerous ceramic type series were created providing relative dates for different types of pottery, sometimes they may have been able to specify even in which generation they were made, such as *terra sigillata.* Moreover, ceramic shape, fabric, and decorations often reflect practices of past societies that inform of everyday life elements such as daily food, commensality, storage, trade, etc.

After any excavation or survey, post-excavation analyses are performed depending on the type of artifact and the research questions. Simplifying the process, wet or dry cleaning, and a provisional inventory are the first stages of ceramic analysis in the laboratory of Archaeology. Subsequently, potsherds fabric and shapes are studied and manually drawn; finally, typologies are assigned. Pottery drawings are made for two main reasons. On the one hand, it is an effective way to record the overall formal and decorative characteristics of archaeological materials. On the other hand, by schematizing and interpreting the potsherd in a diagram, archaeological drawings offer a comparative frame of reference that allows both: to share a visualization of the materials recorded in archaeological works and to compare them with other instances from typological series or other archaeological contexts. This can be a lengthy process that may take months, even years, depending on the size of the fieldwork and how prolific this has been.

For a successful comparison, these diagrams must be standardized, following the widely accepted standard divided in two-parts as shown in Figure 1. On the left, a cross-section of the object presents the vessel profile and interior; on the right, the exterior view of the artifact is represented. Additionally, if the pottery has decoration on the rim, this is portrayed above the upper axis, while if the vessel bottom is decorated, the motives are depicted below the lower axis. In order to facilitate the measurement of similarity amongst artifacts, often a manual visual assessment is carried out. Therefore, using computational techniques to relate and classify automatically potsherds to their typologies can contribute to making inventory and catalog processes faster.

With the growing power of computers, new techniques that were thought to be too computationally demanding suddenly became feasible, and thus able to be utilized in a practical way. An example of these are machine learning methods leveraging deep neural networks. With the ability to use deep learning, automatic image classification techniques progressed significantly [23]. State-of-the-art methods for the classification of images are, in the context of archaeological pottery classification, especially pertinent: they have the potential to support the automatic, very accurate, and interpretable classification of the previously mentioned black-and-white line diagrams of artifacts into typologies.

The exploration of state-of-the-art deep learning techniques in the context of image classification tasks, more precisely black-and-white diagrams of pottery artifacts, is the aim of this paper. Specifically, two different convolutional neural network architectures, namely DenseNet [20] and EfficientNet [42], were the base architectures employed for the creation of the automatic learning models. In order to potentially further enhance the models' performance, various state-of-art techniques were tested, leveraging the aforementioned convolutional neural network architectures, such as AugMix [17] or self-training with a noisy student [48].

To train these models, the main dataset has been a corpus of Roman *terra sigillata* drawings/diagrams from the excavations of the 23rd block of Numantia (Spain), which includes typological information in the form of labels. Few additional labeled diagrams were extracted from various publications of the area with the goal of expanding the categories with the fewest number of instances in the original dataset. Furthermore, leveraging computer vision techniques, unlabeled instances were automatically extracted from a number of articles featuring diagrams of potsherds dating to the Roman period.

This paper is organized as follows. Section 2 presents a survey of related work concerning the processing and categorization of archaeological artifact diagrams, and neural network architectures specific to the classification of images. In Section 3 we present the various components of our deep convolutional architecture; and in Section 4, the main dataset is described in detail. Section 5 expounds both the training strategies and experimental results of this work. Finally, in Section 6, we conclude with some final remarks and possible paths for future research.

Author's address: Tomás Oliveira, tomas.olliv@gmail.com.

2 RELATED WORK

The intuition that the classification of pottery diagrams into typologies can be automatized has already been explored in previous works. Even though the majority of such techniques do not leverage neural networks to construct automatic classification models, in this section, we describe four existing methods that perform automatic typology classification.

2.1 Classifying and Visualising Roman Pottery using Computer-scanned Typologies

Christmas and Pitts developed an automatic classification method [7] based on the k-means clustering algorithm [26]. This approach uses unlabeled data to look for agglomerations of points, i.e., sets of instances with similar characteristics. For points to be considered part of a certain cluster (agglomeration), they must have a small distance to the rest of the points of their cluster (i.e., intra-cluster distance), and a considerable distance to the points in other clusters (i.e., inter-cluster distance). Equation 1, which in the k-means algorithm is to be minimized, formalizes this notion:

$$\sum_{k=0}^{K} \sum_{x \in C_k} \|x - \mu_k\|^2 \tag{1}$$

In the previous equation, μ_k is the centroid (i.e., average point) of cluster k and C_k the set of points (i.e., data instances) of cluster k. The number of intended clusters (i.e., the number of different types of artifacts, in this particular application) must be first chosen when the k-means algorithm is to be applied. After this initial choice, the points are assigned to the cluster with the nearest centroid. These newly attributed points are then used to calculate the new cluster centroid, and the process is repeated until all the centroids are stabilized, i.e., do not change significantly.

The points in this particular application are vectors of measures that are relevant to discern between different types of pottery, with the measures being calculated by using the previously extracted and segmented standardized drawings of the *Camulodunum* series [15]. Various measures of the artifacts were employed, such as the vertical centroid, the height of the artifact, the artifact's width and height, or its circularity, i.e., the closeness from 0 to 1 to a perfect circle.

In the testing of the k-means approach, performed by Christmas and Pitts, the accuracy reached 69.7%, when comparing the number of correctly identified data instances in the cluster of the corresponding typology with the total number of objects assigned to the cluster, and weighting the accuracy of each cluster with the cluster's size (i.e., number of instances in the cluster).

2.2 Arch-I-Scan

Following a different approach, Tyukin et al. developed a software for smartphone devices, named Arch-I-Scan [44], which takes advantage of built-in cameras to scan complete pottery vessels or sherds and performed real-time classification of these objects, assigning each scanned artifact to a vessel typology. In order to do this, video input is first captured in a 2-dimensional frame structure by the smartphone's camera.

Originally presented by Dalal and Triggs, Histogram of Oriented Gradients (HOG) [11] is a technique which produces features from images by computing histograms of gradient directions for each of the image cells (i.e., small fractions of the original image). This technique was employed over the captured video to generate feature vectors of every video frame, which resulted in approximately 100000 feature vectors, with each vector having a number of attributes in the thousands, per frame.

Tyukin et al. modeled the classification task as low-level recognition, i.e., a classifier is built for each one of the existing typologies, being only able to classify its type versus all the others. Using 10 complete pottery vessels, each corresponding to a different typology, a mapping was done between these pottery vessels and the features vectors that resulted from the application of the HOG technique to the captured images (average of 100 images per typology), with each image producing a feature vector of size 2400. As to create more robust classifiers, images of the negative class (i.e., images that are not of the classifier's typology) were also mapped to a category, with each of the image's feature vector being completed using the same aforementioned procedure (i.e., images converted into a feature vector using the HOG method). A detector, one for each of the 10 classes, was then conceptualized. For each one of the feature vectors, a positive or negative value corresponding to the instance's class was returned. By weighting the contribution of each of the features x_i with a certain weight w_i, and adding a certain bias factor b, the predicted class for a certain data instance is computed, as shown in Equation 2.

$$\mathrm{D}(x) = \sum_{i=1}^{2400} (x_i \cdot w_i) + b \tag{2}$$

To learn the discriminant's parameters, Support-Vector Machines [8], a technique for learning a linear classifier which maximizes the distance of a hyperplane separating two classes of data points, were applied. After this parameter learning is completed for each of the classes' detectors, object detection can be performed by employing the 10 different detectors simultaneously to an image. The classifier of the artifact's true type should label the object as the positive class, and the remaining classifiers should label the artifact as pertaining to the negative class.

No considerable number of false positives (i.e., objects erroneously classified as part of the positive class by the detectors of irrelevant classes) were reported in the performed experiments, confirming the robustness of this pottery vessel detection method.

2.3 Content-based image retrieval for historical glass

van der Maaten et al., in a problem setting analogous to this work, developed a content-based image retrieval system [45] to provide assistance to archaeologists in the classification of historical artifacts by automatizing parts of the categorization process which, in its manual mode, corresponds to a classification by matching the artifact to a typology from a reference collection. The content-based image retrieval system fetches similar images to a certain query image (i.e., the image of the artifact to be classified), leveraging measures that are based on particular features of the images.

Utilizing the image retrieval system for the classification of glass artifacts, with the possible categories of these objects being determined through the typologies defined by Kottman [25], presents certain difficulties, namely the absence of particular aspects from the reference typologies (e.g., the real texture of the artifact to be classified, in contrast with the abstract texture of the reference typology) and thus compels the use of the artifacts' outer shape features for similarity measure. van der Maaten et al., in this case, employed the similarity measure based on shape contexts introduced by Belongie et al. [1].

Three basic steps are performed to compute the shape context similarity, namely preprocessing the images, computing the shape context descriptors, and calculating the similarity. Moreover, the preprocessing stage, which has the aim of extracting the outer shape of the query image, entails five substeps: (i) application of a Canny edge detector [4], (ii) linking of edges which are not connected leveraging a morphological dilation operation [34], (iii) a negation operation in conjunction with a bucket fill is applied, in this way binarizing the colors of the foreground and background (i.e., the background becomes black, whilst the artifact is filled in white), (iv) a morphological erosion operator [34] is used to remove erroneous edges, and, finally, (v) a Sobel edge detector [37] is employed, thus obtaining the outer shape of the original image. Leveraging the newly generated outer shape of the artifact, shape contexts are then computed. These shape contexts describe the shape's global information by a set of points that are a result of sampling from the outer shape's border, with the shape context descriptors encoding both the distance and angle of a point to the remaining sampled points. Lastly, the similarity between images is ascertained with the k-nearest neighbors algorithm.

Whilst the aforementioned retrieval system, taking into account the experiments performed by van der Maaten et al., achieved unsatisfactory results for damaged artifacts, it nevertheless provides an improvement over manual classification, lowering both the duration of the classification and possible human errors.

2.4 Ranking Systems for Computer-Aided Single-View Pottery Classification

More recently, Itkin, in the context of the ArchAIDE project[1], explored the use of deep learning in the classification of the artifact's photographs into typologies [21]. Two different approaches were explored, namely an appearance-based classification (i.e., based on the decorative aspects of the artifact) and a shape-based classification.

Appearance-based classification consists in the categorization of artifacts into typologies by taking into account as the main differentiating factor the decorative drawings and the employed colorization. As is the case in our work, Itkin reported a diminutive number of training instances, with a multitude of classes presenting only some dozen instances. Consequently, a method based on transfer-learning was enforced, i.e., a pre-trained model, trained on a considerably more numerous and general dataset is then fine-tuned to a specific domain. Based on the aforementioned intuition, the authors chose to fine-tune a model leveraging the ResNet-50 architecture [16], which was previously trained on the ImageNet dataset [32]. Instead of pursuing a standard neural network classification, maintaining the original structure of the ResNet-50 architecture, in this case, an alternative path was pursued: (i) the intermediate representations lodged at the end of each of the architecture's first five blocks are extracted, thus resulting on five feature tensors, (ii) global average pooling is applied to each of the feature tensors, (iii) the five feature vectors are concatenated, (iv) dropout, with a drop rate of 80%, is employed to combat possible overfitting and aid generalization, (v) fine-tuning is performed by a set of fully-connected layers with a ReLU activation, (vi) dropout is applied once more, and (vii) the final classification is performed. It should be noted that in the previous learning process, the ResNet architecture from which the five feature tensors are extracted is "frozen" (i.e., the ImageNet weights are maintained, no parameter adjusting is performed), with only the fully-connected layers being trained.

Since the data instances used for the appearance-based classification consist of photographs, frequently taken in different circumstances, certain techniques were employed as to promote a more robust classifier, namely to mitigate issues related to changes in lighting and background environments of the input images. For the first aspect, in order to simulate, in training, different lighting conditions, the luminosity (i.e., brightness) of the input image's pixels were scaled by a factor sampled from a distribution, with each of the image's three channels being scaled by three different sampled values. As it pertains to environment variance, the employed solution consists in background removal, leveraging GrabCut [30]. Considering that the GrabCut technique is not fully automatic, an algorithm that automatizes the foreground extraction was developed and posteriorly applied over the available training images. The algorithm can be summarized in three basic steps: (i) identify the background by sampling colors of the image's border, (ii) generate a distance image composed of the distances between each pixel and the nearest sampled background pixel and binarize the image according to a distance threshold (i.e., the background in black and the foreground in white), and (iii) apply GrabCut, with the newly segmented areas labeled as background or foreground.

In opposition, shape-based classification is based on the geometry of the found artifacts: the shape of the pottery fragment is compared to a reference collection of standardized artifact diagrams with a known typology. Since color information was not to be taken into consideration in this approach, transfer learning was not used. In consequence, a synthetic data generation procedure, with the goal of mitigating the diminutive dataset, was developed that, from the outlines of the available potsherd images, produces a virtual model of the pottery artifact, which is then fractured in a randomized manner, thus emerging a synthetic sherd that is added to the training data in the form of its outline. To generate the model of the artifact, a three-dimensional object, from its sherd in an efficient fashion, the sherd's profile is projected onto the xz plane and it is then rotated around the z axis. Since each of the profile's points (p_x, p_y), when rotated around the z axis, forms a circle perpendicular to z, the three-dimensional artifact's points can be modeled according to:

$$x^2 + y^2 = p_x^2 \wedge z = p_y \tag{3}$$

Using the previously defined circles, the fractures are attained with the intersection of a certain randomly defined three-dimensional plane with the circles corresponding to the profile's points. Point sampling is then applied in order to obtain the discrete points used for the training.

The learning process, leveraging the synthetic profiles, is achieved with the OutlineNet architecture, based on the PointNet architecture [29] introduced by Qi et al.. A two-input approach was utilized in the architecture: one input encodes the position (i.e., coordinates) of the points while the other the angle at the location of the points (in relation to the outline). Each of these two inputs is propagated in separate paths, both are given as input to separate multilayer perceptrons with four layers, with the two results being concatenated and passed, now in conjunction, to another multilayer perceptron (two layers). Next, max-pooling is applied over the product of the previous multilayer perceptron and is then passed to the final multilayer perceptron, performing the final classification.

As it pertains to appearance-based classification, Itkin reports accuracies of 55.2% in an experiment using more than 700 images from 49 different classes while, on the other hand, shape-based classification over approximately 400 images from 42 classes only achieved 18.9% accuracy.

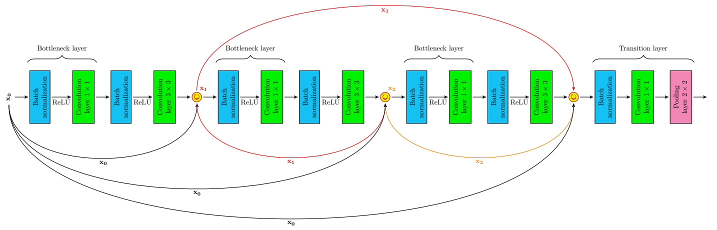

Fig. 2. Diagram of a dense block from the DenseNet architecture, with three layers followed by the transition layer. The **C** operation, depicted in yellow, represents the concatenation of matrices.

3 DEEP LEARNING ARCHITECTURE

The automatic classification models developed in the course of this work can be divided into four distinct parts: (i) the convolutional neural network architecture, (ii) a meta-learning technique, leveraging the representations learned by the convolutional neural network model to mitigate the performance issues related to the smallness of the dataset, (iii) self-training leveraging the noisy student approach, and (iv) AugMix, a state-of-the-art data augmentation technique used to regularize the learning process.

3.1 DenseNet Architecture

One of the best-performing convolutional neural network architectures in image classification tasks is the DenseNet architecture. With the goal of improving the flow of information, especially pertinent for convolutional neural networks with considerable depth, i.e., deep networks, Huang et al. developed the DenseNet architecture. The central concept of the architecture consists on propagating all the previously learned information throughout the network, instead of only using shortcut connections to propagate forward the block's input [16]. Each layer, in order to accomplish

[1] http://www.archaide.eu

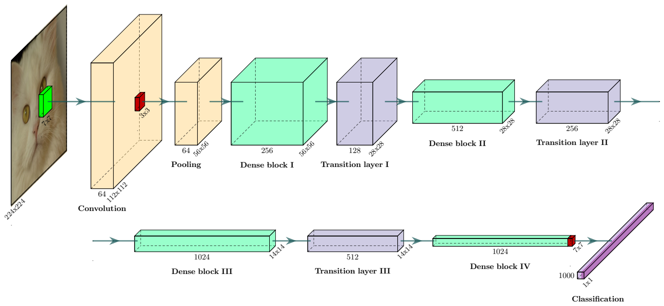

Fig. 3. Overview of a DenseNet architecture, specifically the DenseNet-121. The convolutional and pooling operations are depicted, in this diagram, respectively in bright green and dark red. Regarding the building blocks of this architecture, the dense block is depicted in light green, whereas the transition is represented in light purple. The final layer, composed of a fully-connected component and a softmax activation function, is depicted in purple.

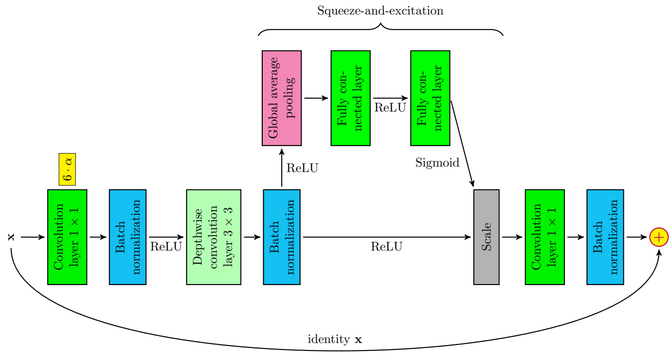

Fig. 4. Diagram of a MBConv6 3×3 block. In a yellow box, next to the first 1×1 convolution layer, we can observe how α, i.e., the tensor's channels, are scaled with an expansion ratio of 6. The + operation represents matrix addition.

this, propagates its feature matrices to all subsequent layers, as depicted in Figure 2.

Considering l to be an index over the architecture's layers, $l-1$ feature matrices will be transmitted to layer l. In order to accumulate additional information during training, these propagated layers are then concatenated to each layer's own feature tensor, i.e., $\mathbf{x}_l = f_l([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{l-1}])$. Such concatenated and propagated layers serve as collective knowledge for the whole convolutional neural network since they represent all the previous knowledge (i.e., learned parameters), and will not be modified (i.e., trained) after being transmitted to a posterior layer. In a standard convolutional neural network architecture, there are L connections between layers, with L representing the number of total layers. Because the different layers must be propagated to subsequent layers, $\frac{L\times(L+1)}{2}$ connections must be used in the DenseNet architecture.

Curbing the rise in the number of each layer's feature tensor, essential to maintain efficiency, is attained by the employment of the growth rate hyper-parameter k (i.e., how the number of feature matrices evolves throughout the network) which, in the DenseNet architecture, is attainable to be of a small to moderate size. Compared with ResNet architecture, DenseNet utilizes only half the parameters to obtain equal results. This diminutive number of parameters, when compared with other architectures, can be attributed to the general knowledge (i.e., the feature matrices propagated to subsequential layers) implicitly captured by the architecture, which promotes parameter efficiency. The effective use of parameters in the DenseNet architecture can additionally hinder potential overfitting due to the less complex nature of an architecture with fewer parameters.

In more detail, the DenseNet architecture is composed of convolution layers, pooling layers, dense blocks, and a transition layer. The dense blocks, as shown in Figure 2, are the main learning units of this architecture, being comprised of multiple 1×1 and 3×3 convolution layers with the aforementioned shortcut connections between them. The 1×1 convolutional layers, called bottleneck layers, are used by the authors to diminish the rising number of the layer's feature matrices, while the 3×3 filters are responsible for extracting the relevant features from the received input. Consequently, the 1×1 filter inside the dense block is placed before the main 3×3 convolution layer in order to facilitate the computation of this convolution operation. In order for the down-sampling to occur throughout the architecture, dense blocks are separated by transition layers, which contain a batch normalization unit, a 1×1 convolutional filter, and a 2×2 average pooling layer.

One of the possible implementations of this architecture is the DenseNet-121 (i.e., with 121 as the architecture's depth), shown in Figure 3. The diagram shows the convolutional neural network with four dense blocks separated by the aforementioned transition layers. Each dense block is comprised of 6, 12, 24, and 16 layers, respectively. For the relevant feature matrices to be extracted from the input image, a convolution filter of size 7×7, with a stride size of two, must first be applied to the input. The result of this application can be seen in the convolution block of Figure 3. Subsequently, a 3×3 max-pooling filter is also applied, with a stride of size two. Then, the resulting feature matrices are supplied to the sequence of four dense blocks, parted by transition layers, which, as previously said, perform the main portion of the learning process. Afterwards, the resulting 7×7 feature matrices are subjected to a 7×7 average global pooling (i.e., in this case, the size of the filter is equal to the size of the feature matrices). Finally, the classification is obtained by using a 1000-dimensional (i.e., number of categories in the ImageNet dataset, but of size 19 in the context of this work) fully-connected layer, followed by the application of the softmax activation function. In this architecture, the k hyperparameter (i.e., the growth factor in the dense blocks) is set to 32 and, as we can observe in Figure 3, the first dense block receives 64 feature matrices from the previous layer, and adds them to the features matrices produced by the 6 layers inside the dense blocks (i.e., 6×32), totaling the 256 features matrices indicated in the diagram.

Concerning the application of the DenseNet-201 architecture in the context of this work, the aforementioned architecture, was, according to multiple experiments, assessed as the better-performing architecture and, consequently, is the base architecture on which additional techniques are applied upon, being responsible for the generation of the automatic classification models of this work.

3.2 EfficientNet Architecture

More recently, Tan and Le introduced the EfficientNet architecture [42], based on the idea that instead of adjusting only the depth of a convolutional neural network, scaling should be made over three different dimensions: depth, width, and resolution. In general, deeper architectures produce more accurate models, although considerably deeper networks do not present significantly better results than moderately deep architectures. Furthermore, a network's width can also be adjusted to improve performance, but only until a certain point, after which the performance gains begin to rapidly diminish. Yet another dimension that can be scaled in order to achieve better performances is the resolution of the images provided as input, again up to a given limit. Due to the fact that the improvements obtained by individually scaling each of the dimensions are considerably limited, Tan and Le introduced the concept of compound scaling.

The underlying idea is that when the architecture's depth is increased, because larger receptive fields are accessible, higher resolution input images can be better leveraged. If the network's width is likewise scaled, this can provide the architecture with the ability to grasp features that have a finer grain. The compound scaling concept, which adjusts all 3 network

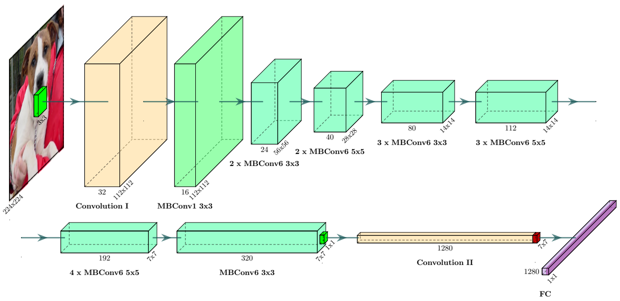

Fig. 5. Diagram of the EfficientNet-B0 architecture, composed of various MBConv blocks in conjunction with convolution, pooling, and fully-connected layers. As before, the convolutional filters are depicted in green, whilst the pooling are represented in red. On the other hand, the fully-connected layers are depicted in purple. Some layers with the same characteristics are placed consecutively, with the number of these layers being depicted in the diagram by the multiplying constant placed before the layer's name.

dimensions (i.e., d, w and r) in parallel, considering their interplay, is thus formally defined as:

$$d = \alpha^{\phi}, w = \beta^{\phi}, r = \gamma^{\phi} \tag{4}$$
$$\text{subject to } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

In the previous equation, d represents the depth, w the width, and r the image resolution. A parameter called compound coefficient, denoted by ϕ, is the scaling parameter of the network's three dimensions, and α, β, and γ are constants that represent the resource distribution for each one of these dimensions. The constants are subject to the previous condition in order to guarantee that the computation cost of a different ϕ grows with 2^{ϕ}.

Compound scaling was posteriorly tested by the use of a baseline network called EfficientNet-B0. The authors designed this network with basis on the method used by Tan et al., where a search with multiple objectives (e.g., latency and accuracy) was done in order to maximize an accuracy metric for a certain generated model. In this case, the authors wanted to maximize the model's accuracy whilst using relatively low FLOPS (i.e., computational cost). The optimization problem is thus formalized as follows:

$$\max_{x} \ \text{Accuracy}(x) \cdot \left[\frac{\text{Flops}(x)}{\text{T}} \right]^{w} \tag{5}$$

In the previous expression, x represents a model, $\text{Accuracy}(x)$ is a function mapping a model to an accuracy score, $\text{Flops}(x)$ is a function of the computational cost of a certain model, T is the intended target FLOPS, and w is a hyperparameter that adjusts the compromise between the two metrics. Using Equation 5 and the search method introduced by Tan et al., the authors reached the baseline architecture named EfficientNet-B0 from which subsequent versions and improvements build upon. The main block of this architecture is the mobile inverted bottleneck (MBConv) leveraging the squeeze-and-excitation technique [19].

As can be seen in Figure 4, the aforementioned block is composed of two convolution layers, a depthwise convolution layer, and the squeeze-and-excitation optimization. In a depthwise convolution layer, a sole filter is applied to each of the input channels of the feature matrix, followed by feature creation using a pointwise convolution (i.e., a 1×1 convolution). The two-step procedure differs from a standard convolutional layer, in which both filtering and feature creation are achieved concomitantly.

The first layer of the MBConv block, as seen in Figure 4, is responsible for scaling up the number of channels in the block's input. Such scaling of the number of channels was devised by Sandler et al. as an improvement over the MobileNetV1 [18] architecture, with the objective of mitigating potential knowledge dissipation during training.

In contrast, the squeeze-and-excitation optimization has the objective of enhancing the learning process in convolutional neural networks by adding to the learned model dependencies between the multiple existing input channels. This is achieved in two phases; first the squeeze phase uses a global average pooling filter in each of the channels, this way producing statistical descriptors for the aforementioned channels. Then, these descriptors are used in the consequent excitation phase to learn interdependencies between channels. Such dependencies are exposed by the following two fully-connected layers. The learned non-linearity is then followed by a sigmoid activation function, and finally a scaling block, that is defined as the channel-wise product between the learned scalars and the feature maps, is applied.

The EfficientNet-B0 architecture, as depicted in Figure 5, is mainly composed of a series of these MBConv blocks placed in sequence. From this baseline architecture, 7 better performing architectures (e.g., EfficientNet-B1, EfficientNet-B2, etc.) were generated by scaling the baseline architecture leveraging the compound scaling method in two phases: firstly, with the compound coefficient ϕ parameter set at one, a grid search was made to find the value for the depth, resolution, and width constants; secondly, using these newfound values for the three dimensions, different values for the ϕ parameter were tested. With the state-of-the-art concerning the ImageNet dataset being reached by the EfficientNet-B7.

In the context of our work, due to limited computational resources, the EfficientNet-B3 was the largest EfficientNet variant for which all the explored techniques (e.g., AugMix) could be tested and, consequently, was the EfficientNet architecture used as a basis for our experiments.

3.3 Two Input Convolutional Neural Network Architecture

As previously mentioned in Section 1, the diagrams in question are composed of two views. Consequently, to potentially take advantage of the knowledge of the data instances' structured format, we developed a two-input based architecture. In order to accomplish this, two separate convolutional neural networks are placed in parallel: one receiving the left part of the image and another the right part of the input image. The input images are propagated through both models, with the last layers' output of both individual models being concatenated. Such a vector is linked to a fully-connected layer (i.e., the final classification layer with the classes' dimensionality) performing the model's prediction. We again considered the DenseNet-201 and EfficientNet-B3 architectures, but now, taking into account the two-input format, in a pairwise disposition, i.e., two DenseNet-201 or EfficientNet-B3 models.

3.4 Few-shot Learning and Meta-Learning

Automatic classification scenarios where only a diminutive number of instances for each of the existing categories are available are denominated few-shot learning. Our dataset, as can be assessed in Section 4, is composed of only a small number of examples for each of the 19 analyzed typologies and, therefore, fits in this type of scenario.

One of the possible approaches to solve few-shot learning tasks is the application of automatic learning algorithms (e.g., k-nearest neighbors) over the representations of images that the convolutional neural network model has produced, called meta-learning, leveraging the idea of learning over previously acquired knowledge. SimpleShot [47] or the classification methodology introduced in [43] are examples of this, with both techniques achieving state-of-the-art results in few-shot learning benchmark tasks.

Although more intricate techniques, such as the Prototypical Networks [36], were explored in preparatory tests, the best results were achieved with the simple application of other learning algorithms atop the convolutional neural network's learned embeddings. Three different classification algorithms were thus considered to classify the learned representations, namely k-nearest neighbors, random forest, and logistic regression. The employed meta-learning process consists in (i) training the convolutional neural network model (e.g., the EfficientNet-B3), (ii) removing the last layer of the

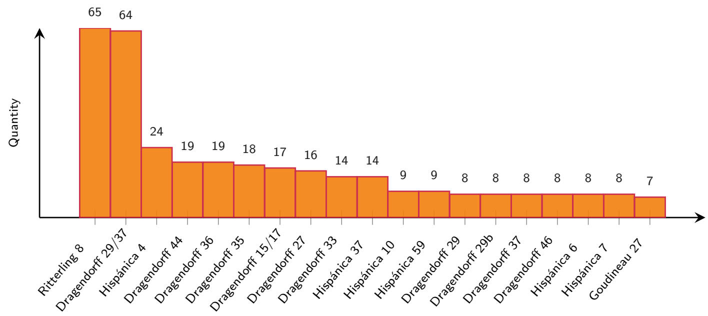

Fig. 6. Histogram of the 19 available categories from the project's main dataset after filtering for classes with more than four instances, preceding the addition of supplementary data instances.

model (i.e., the classification layer with the classes' dimensionality), (iii) obtaining the learned representations of the data instances by predicting both the training and validation set, leveraging the learned model, (iv) applying standardization to the images' features (i.e., remove the mean and divide by the samples' standard deviation) for logistic regression and l_2 normalization for k-NN, and (v) training and classifying the representation vectors, leveraging one of the aforementioned classification algorithms. Moreover, concerning the k-NN classification, the approach employed by Wang et al. [47], i.e., both the training and test data instances are mean-centered by subtracting the mean of the training instances before the l_2 normalization, is applied in the aforementioned meta-learning procedure for the dataset's experiments.

3.5 Self-training

One of the principal pitfalls of the application of deep convolutional neural networks to a certain domain is the scarcity in available labeled data instances. A commonly taken approach to mitigate this issue is the use of unlabeled data instances through specialized techniques, since such type of instances are significantly more accessible. Self-training is one of these techniques that can be leveraged to take advantage of unlabeled data instances. The technique itself can be summarized in three steps: (i) an initial model, called the teacher model, is trained over the available labeled data instances, (ii) the teacher model is then used to classify the unlabeled data instances creating class predictions called pseudo-labels, and (iii) a new model, denominated student model, is trained over the totality of the available data, i.e., over both the labeled data instances and the unlabeled data instances leveraging the pseudo-labels created by the student model. Following the intuition that a student model with an additional amount of noise can further improve its classification performance (i.e., the noisy student approach [48]), a dropout layer [39], with a drop probability of 40%, was added to the student model employed in our experiments, more precisely, in the model's penultimate layer (i.e., the global average pooling layer).

After obtaining a considerable quantity of unlabeled data instances, 2963 to be more precise, the above-mentioned procedure was applied to our dataset, slightly modified to include a prediction confidence threshold. This threshold was used to filter the instances labeled by the student model with a prediction probability smaller than 95% (corresponding to the average of each fold's probabilities) which, for one of the developed DenseNet-201 models, corresponded to 94 data instances. This filtering step allows mitigating the potential problems related to images from exogenous classes, since the typologies present in the unlabeled data may not intersect with our class set.

3.6 AugMix Data Augmentation

Data augmentation techniques are usually used in convolutional neural networks with the objective of regularization of the learned model. In recent years, regularization techniques such as MixUp [49], have achieved state-of-the-art results in image classification tasks. Consequently, we applied data augmentation techniques in this study, specifically the method introduced by Hendrycks et al. denominated AugMix [17].

The goal of Hendrycks et al. [17] was related to the regularization of convolutional neural network models by the way of closing the gap between the distribution of the training data used to train the aforementioned classification model and the data that it is intended to correctly categorize. In short, this technique generates synthetic data instances from the original training data by combining augmentation operations (e.g., rotate, posterize, among others), thus effectively creating chains of operations, subsequently weighting the resulting operation chains. Firstly, the weights that will be weighting the k operations chains are sampled from a Dirichlet distribution with concentration parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$. Then, the k operations are generated by sampling three different augmentation operations and composing them into sequences, i.e., chains, with a length of one to three operations (e.g., a chain of operations of length two comprised of a rotation followed by a horizontal translation). Such newly created chains are then weighted, producing a new intermediate data instance combining the operations chains. Furthermore, leveraging the MixUp regularization technique, a new data instance is generated using pairwise interpolation, as expressed in Equation 6, between the original image and the image that results from the application of the data augmentation operations chains.

$$\begin{aligned} \tilde{x} &= \lambda \cdot x_i + (1 - \lambda) \cdot x_j \\ \tilde{y} &= \lambda \cdot y_i + (1 - \lambda) \cdot y_j \end{aligned} \tag{6}$$

As to promote both the original data instance and its augmented version to be classified in the same category by the trained model, the Jensen-Shannon divergence consistency of the probability of the original data instance, as well as its chained augmented variants, being classified by the model as a certain class. Thus, the loss function is defined as:

$$L\left(\hat{p}(y|x_0), y\right) + \gamma \cdot \text{Jensen-Shannon}\left(\hat{p}(y|x_0), \hat{p}(y|x_1), \hat{p}(y|x_2)\right) \tag{7}$$

In the previous equation, L represents the model's loss, γ (set to 12 in this case) the weight attributed to the Jensen-Shannon divergence consistency in the loss function, $\hat{p}(y|x_0)$ the posterior distribution of the original image x_0, $\hat{p}(y|x_1)$ the posterior distribution of an augmented variation x_1, and $\hat{p}(y|x_2)$ the posterior distribution of a second augmented variation x_2. The Jensen-Shannon term of the loss function, expressed in Equation 7, is defined as follows:

$$\text{JSD} = \frac{1}{3} \cdot \left(\text{D}_{\text{KL}}\left(\hat{p}(y|x_0) \| \text{M}\right) + \text{D}_{\text{KL}}(\hat{p}(y|x_1) \| \text{M}) + \text{D}_{\text{KL}}(\hat{p}(y|x_2) \| \text{M})\right), \tag{8}$$

$$\text{M} = \frac{1}{3} \cdot \left(\hat{p}(y|x_0) + \hat{p}(y|x_1) + \hat{p}(y|x_2)\right)$$

In the previous equation, D_{KL} represents the Kullback-Leibler divergence measure.

4 DATASET DESCRIPTION

The sample data for both training and testing the convolutional neural networks comes mostly from the excavation of the 23rd Roman block of Numantia (Garray, Spain) between 2004 and 2009. Numantia is a well-known *oppida* of Iberia during the Iron Age due to the role it played in the wars against Rome, especially in the Second Celtiberian War (153-98 B.C.), also known as the Numantine War. After the conquest of Numantia in 133 B.C., Rome established several Roman cities in this location, the remains of the city built in the time of Augustus are the best preserved. Dated between the 1st and 2nd centuries A.D., this dataset comprises all the *terra sigillata* pottery diagrams that the Numantia Archaeological Team manually drew and digitized to document the materials from this excavations under a project that aimed to review the Iron Age and Roman urban planning of this site [22]. *Terra sigillata* is a kind of pottery highly standardized in its forms, characteristic of the Roman Empire. It is defined as pottery tableware

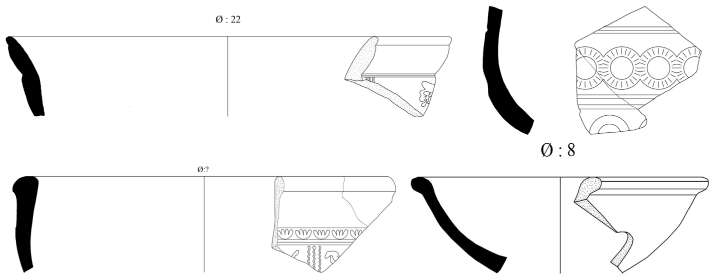

Fig. 7. Example of four artifact diagrams from the project's dataset.

Table 1. Exploration of different data augmentation strategies for training. Highlighted in bold are the best results for each of the explored architectures.

Model	Strategy	Pre	Rec	F1	Acc
DenseNet-201	Basic	52.23	52.52	50.32	63.59
	MixUp	52.39	54.94	51.33	62.69
	AugMix	**55.13**	**56.10**	**53.86**	**67.65**
EfficientNet-B3	Basic	49.52	51.56	47.21	56.86
	MixUp	54.84	52.94	51.01	59.21
	AugMix	**55.14**	**55.51**	**53.37**	**65.60**

with a distinct glossy reddish surface, sometimes with geometrical, floral or figural decorations [9].

In the excavations of the 23rd block of Numantia, around 12,000 artifacts were recorded. Among them, 4,083 corresponded to *terra sigillata* sherds. During the post-excavation process, 559 *terra sigillata* fragments were manually drawn creating black-and-white pottery diagrams, showing two different perspectives on the depicted pottery, namely a profile view of the artifact and its cross-section, as can be seen in Figure 7. Due to the conservation conditions and the high fragmentation of the materials, only 392 pottery diagrams could be assigned a typology, representing around 65% of the dataset. These pottery diagrams span 48 different typologies (i.e., artifact categories), showing an unbalanced distribution per pottery type, as can be seen in Figure 6. To expand this dataset, we considered using other sources of pottery diagrams, such as specialized publications on Hispanic, Gallic and Italic forms and collections of *terra sigillata* from contexts similar to Numantia. Taking into account that a substantial quantity of classes exhibited less than four instances (an exceptionally small number of instances), we filtered classes manifesting such characteristics, which resulted in a total of 320 data instances from 19 different classes.

As previously mentioned, in order to expand those classes with a number of instances smaller than 7, additional data instances were included from published sources. In this case, 23 additional diagrams were extracted from: [5], [3], [2], [28], [12], and [31], reaching a total of 343 *terra sigillata* fragments with assigned typology, henceforth this will referred as labeled instances.

The amount of labeled instances in the application of convolutional neural networks to specific domains can prove to be scarce, as in this research. Consequently, unlabeled data instances can be used as a secondary data source to increase the size of the dataset. In our case, by searching through various publications depicting artifact pottery diagrams (e.g., the above-mentioned articles) and, leveraging automatic object detection techniques, we obtained 2963 new unlabeled pottery diagrams. These unlabeled diagrams were used only in experiments leveraging self-training approaches.

Since convolutional neural networks require a constant square image size in most settings, the images of the dataset were transformed into a square image and then resized to the chosen resolution of 448×448, only limited by our computational resources. On the other hand, for the two-input architectures, the same process was applied but, since the images are separated in half, the input's selected resolution was 224×224.

5 EXPERIMENTS

Various experiments leveraging different types of architectures and dissimilar types of data augmentation were explored. Furthermore, additional crucial implementation details are introduced. Here we present such experiments and the strategies used to perform them.

5.1 Training Strategies

As mentioned before, the performed experiments leveraged a DenseNet-201 model pre-trained on the more general ImageNet dataset. In parallel, experiments leveraging the more recent EfficientNet architecture, in particular the EfficientNet-B3, were also performed. As to adapt both pre-trained models to the two pottery classifications tasks, one additional layer was added, namely a fully-connected layer with the classes' dimensionality. The pre-trained models were trained in their entirety (i.e., none of the original weights were made constant) while employing two different optimizers, namely AdaMod [13] and Adam [24], with the first being employed in DenseNet training and the second for EfficientNet training. It should be noted that for both optimizers the default hyperparameters were maintained.

Training leveraged the policy introduced by Smith [35], namely Cyclical Learning Rate (CLR). More concretely the considered triangular policy decreases the learning rate amplitude by half at the end of each cycle, with, in this case, the learning rate varying between a constant base value of 10^{-5} and a maximum of 10^{-3}. Moreover, taking into account the experiments performed by Smith, training is stopped at the end of the last cycle, as per the recommendation in the aforementioned paper, and network training is limited to five training cycles. Concerning the step size (i.e., half the size of a cycle), the recommendation of varying this hyperparameter between two and ten epochs is followed, with different regularization techniques using different values.

As to better evaluate the generalization capacity of the learned models, a stratified cross-validation split was employed, in this case, with five splits, i.e., in each of the mutually exclusive folds, 80% of the data is used for training and 20% for validation purposes, with each fold's composition maintaining relative class representation.

5.2 Experimental Results

Various metrics were considered in the evaluation of the performed experiments, namely the macro-average versions of precision, recall, and the f1-score. Intuitively, the precision of a class can be defined as the ratio of correctly classified data instances, of all the instances labeled as the class in question. On the other hand, recall expresses the portion of correctly labeled instances by the system, of all the data instances composing the class in evaluation. Finally, the f1-score, which combines both previously mentioned metrics, is defined as the harmonic mean between precision and recall. Calculating these metrics using the macro-average method (i.e., taking the simple average of each of these metrics for each of the dataset's classes) is of considerable importance since the available labeled data is notably unbalanced. Considering that a cross-validation split is being used, these metrics are first calculated in each fold, with the final metric value resulting from the mean of the five folds' performance.

Firstly, with the objective of measuring the regularization capacity of data augmentation in the context of the classification of standardized pottery diagrams, two types of data augmentation techniques were compared with the state-of-the-art AugMix regularization technique, namely MixUp (i.e., the method used in one of the steps of the AugMix algorithm) and a basic fixed set of transformations, namely brightness shifting by a factor between 0.8 and 1.2, position shifting by a maximum of 12.5%, and randomly rotating in a range of 2.5 degrees. Depicted in Table 1 are the results of these experiments. Such results show a clear performance increase against basic augmentations in all of the three assessed metrics when techniques

Table 2. Results of the application of the different algorithms, leveraging meta-learning, against the standard convolutional neural network prediction (categorized as None). Only the best-performing versions of the models are here presented (corresponding, in this case, to models leveraging AugMix). In bold are portrayed the best results of each model.

Model	Algorithm	Pre	Rec	F1	Acc
DenseNet-201	None	55.13	56.10	53.86	**67.65**
	1-NN	**58.74**	**60.77**	**57.88**	67.07
	3-NN	57.61	58.62	56.00	**67.65**
	5-NN	56.83	59.23	55.81	**67.65**
	7-NN	54.64	56.59	53.57	66.49
	RF	52.50	52.41	50.34	66.78
	LR	53.16	51.92	50.42	66.19
EfficientNet-B3	None	55.14	55.51	53.37	65.60
	1-NN	53.61	55.14	52.14	63.28
	3-NN	**56.26**	**56.64**	**54.34**	64.15
	5-NN	51.31	54.64	51.10	62.11
	7-NN	52.65	54.05	51.68	63.27
	RF	53.82	51.13	49.37	65.32
	LR	54.56	54.97	52.50	**65.89**
2-DenseNet-201	None	57.65	55.12	53.35	**67.29**
	1-NN	59.11	**58.15**	**56.11**	65.82
	3-NN	**60.04**	57.35	55.96	**67.29**
	5-NN	56.85	55.21	53.43	65.53
	7-NN	57.40	53.85	53.21	64.92
	RF	52.29	49.43	48.46	65.79
	LR	56.09	51.98	50.98	66.98
2-EfficientNet-B3	None	49.99	51.96	48.78	60.55
	1-NN	50.11	**52.86**	48.94	60.53
	3-NN	51.51	52.83	**49.71**	61.11
	5-NN	**51.64**	52.65	49.52	61.11
	7-NN	51.09	51.43	48.65	60.81
	RF	44.43	42.52	41.28	59.38
	LR	48.16	48.53	45.97	**61.13**

that perform mixing between images are applied, in this case, MixUp and AugMix. On the other hand, the 1-3% performance increase from MixUp to AugMix reflects the added complexity of the latter, namely the use of multiple chains of operations and the Jensen-Shannon divergence consistency added to the loss function.

Secondly, we assessed the performance of various classification algorithms leveraged in the meta-learning approach. Having the results obtained in the previous experiments as a basis, we fixed the image augmentation procedure to the AugMix algorithm for these experiments. In addition to the architectures used in the remaining experiments (i.e., DenseNet-201 and EfficientNet-B3), we tested the architecture and training technique introduced in Section 3.3 and 3.5, respectively. Table 2 depicts the results of such experiments, with 2-DenseNet-201 and 2-EfficientNet-B3 corresponding to the two-input version of these architectures.

Concerning the single input architectures trained exclusively over the labeled data instances, a clear performance superiority of the DenseNet-201 model over the EfficientNet-B3 across the three tested metrics can be discerned with, for instance, DenseNet outperforming the Efficient model by 1.66% in what concerns one of the better f1-score results of both models (i.e., the 3-NN algorithm). One of the possible causes for this discrepancy in meta-learning performance is the difference in feature vector dimensionality: the DenseNet's feature vector is 1920-dimensional while the EfficientNet's is 1536-dimensional. The same comparison performed between the two-input architectures attains similar results, with the 2-DenseNet-201 model outperforming the 2-EfficientNet-B3, albeit with a larger performance disparity

Table 3. Comparison of the performance of the tested meta-learning algorithms between the best-performing model (i.e., DenseNet-201) in the context of the previous experiments leveraging a self-training approach and the original model with dropout noise.

Model	Algorithm	Pre	Rec	F1	Acc
DenseNet-201 w/ dropout	1-NN	57.66	**57.99**	55.47	65.90
	3-NN	56.67	56.92	54.42	66.19
	5-NN	**58.12**	57.95	**55.63**	**67.65**
	7-NN	56.24	56.29	53.89	66.77
	RF	54.07	52.07	51.02	66.49
	LR	55.74	54.04	52.93	67.37
DenseNet-201 Self-trained	1-NN	**62.88**	61.61	60.28	69.11
	3-NN	62.47	61.18	59.73	69.41
	5-NN	62.61	**62.30**	**60.51**	**71.45**
	7-NN	59.87	60.14	58.20	70.58
	RF	55.45	54.85	53.58	68.53
	LR	59.52	58.05	56.74	69.69

between them. Both models, however, achieved worse results than their single-input counter-parts concerning meta-learning. Such performance decrease can be attributed to the halving of the image's resolution (i.e., the two view diagram were split into two different images) since the data instances correspond to diagrams where diminutive features can have a notable impact on classification performance and thus higher resolutions are critical for successful results.

By evaluating the algorithms applied over the four previously mentioned models' learned representations, it can be ascertained that the k-NN based algorithms, particularly 1-NN and 3-NN, perform the best across the precision, recall, and f1 metrics for all models. In terms of accuracy, the EfficientNet-B3 and 2-EfficientNet-B3 architectures achieve the highest performance leveraging a logistic regression, with the remaining architectures performing the best on k-nearest-neighbor classifiers or the standard network prediction.

An analysis of the quality of the representations learned by the convolutional neural network architectures can be observed in Figure 8, where the dataset's instances are represented as points, with their color depicting its true typology. Particularly, the way in which the 94 unlabeled data instances, used in the self-training setting, clustered in accordance with the pseudo-labeling attained by the DenseNet-201 model, which, in this case, was trained using the entirety of the labeled dataset (in order to avoid different representations by each of the five models, one per fold). In order to produce the above-mentioned graphic, first, the Principal Component Analysis (PCA) dimensionality reduction technique was applied, in this case, to the output of the DenseNet-201 model developed in the previous experiment. The output vectors were thus reduced from a 1920-dimension vector to a 94-dimension vector, with this transformation resulting in no added noise to the final vector. Secondly, the t-distributed Stochastic Neighbor Embedding (TSNE) [46] is employed to depict this 94-dimension vector in a two-dimensional plot. Observing the aforementioned scatter plot, small groups of data instances with the same pseudo-labeled typology (e.g., Dragendorff 27) are made evident. Of the eleven classes present in the pseudo-labeled set, distinct agglomerations for each one of the classes (i.e., a small intra-cluster distance and a large inter-cluster distance) can be observed, with no noticeable mixture between the clusters. Such agglomerations of instances of the same type justify the performance of meta-learning leveraging a k-NN classifier reported in previous experiments.

According to the *terra sigillata* typologies, Figure 8 shows three main groups. First, the upper center and right parts of the scatter plot display those vessels that are larger, corresponding to types such as Dragendorff 29/37, Dragendorff 37, and Hispánica 10. Despite the lack of decoration of the latter, they are quite similar typologically; the three are bowls with straight walls and a slightly thick rim. Second, the bottom and left portion

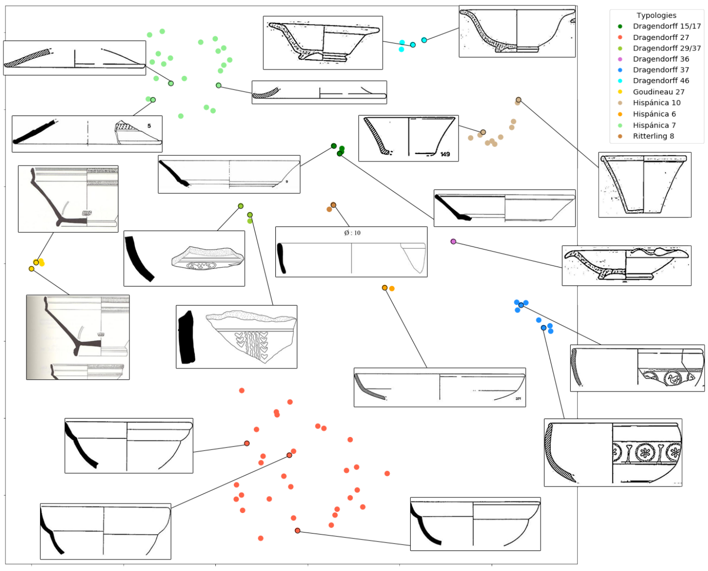

Fig. 8. Scatter plot of the DenseNet's learned representations for the available data instances with randomly selected example diagrams from each of the displayed clusters highlighted. The embedding to a two-dimensional space was obtained leveraging the t-SNE technique with the perplexity hyperparameter, regulating focus on more local or global data features, set to 45.

of the graph groups forms that are usually smaller vessels or bowls, which generally present moldings or carenations (i.e., an abrupt change of direction) on the rim and the neck. Finally, the upper-left encompasses mainly those artifacts that are lids.

The best results, however, were achieved leveraging the self-training technique. By using the newly best performing model (i.e., the DenseNet-201 model) to label the unlabeled data instances, the dataset is extended and the training is repeated with the aforementioned noise leveraging a dropout layer with a drop probability of 40%. Taking a meta-learning approach, more precisely, by applying the k-NN algorithm over the model's learned representations is the best-performing of the tested procedures, achieving the second-best precision (62.61%), best recall (62.30%), and best f1-score (60.51%) metrics with 5-NN, as can be observed in Table 3. Comparing the model leveraging self-training with the best DenseNet model (i.e., DenseNet-201 w/ AugMix), particularly with respect to the meta-learning approach leveraging the 1-NN classifier, a 4.14%, 0.84%, and 2.40% increase in the precision, recall, and f1 metrics, respectively, can be discerned over the best result harnessing meta-learning. As to assess the contribution to model performance of the aforesaid dropout layer, a DenseNet-201 architecture leveraging a dropout layer with a drop probability of 40% was tested. The results of these tests, as can be observed in Table 3, show an identical performance concerning the precision, recall, and f1 metrics to the DenseNet-201 model without dropout and, consequently, exclude the incorporation of the dropout technique as the sole cause for the performance increase in the self-training model.

Figure 9 depicts four different classification scenarios across two distinct axes, namely class volume and prediction correctness. Concerning the minority classes, examples from the Dragendorff 29 and Goudineau 27 typologies were analyzed, while for the more numerous classes, data instances from Ritterling 8 and Dragendorff 29/37 were considered. Observing the prediction of the data instances from the Dragendorff 29 minority class (located at the bottom-right of the figure), predicted as part of the Dragendorff 29/37 typology, and taking into account the data instances of this class, it is possible to ascertain a considerable similarity between the mislabeled data instance and the images of the erroneously attributed category which, given the diminutive quantity of the Dragendorff 29 class, critically contributes to that misclassification. On the other hand, analyzing

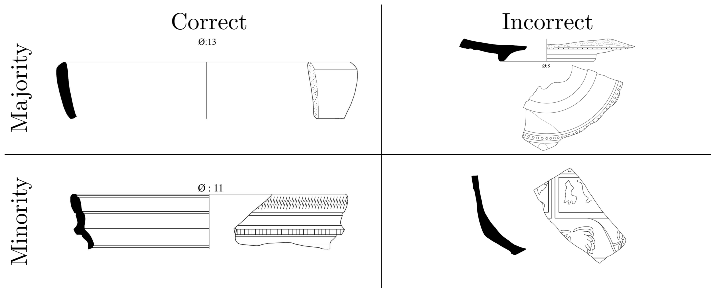

Fig. 9. Example of four artifact predictions leveraging the DenseNet-201 model with the 5-NN classifier. Two dimensions are depicted in the figure above: whether the instance's class is one of the most numerous classes and the correctness of the system in the image classification task. From top to bottom, left to right, the considered instances' categories: Ritterling 8, Dragendorff 29/37, Goudineau 27, and Dragendorff 29.

an example from a majority class, namely the Dragendorff 29/37 (located at the top-right of the figure), which was classified as a Dragendorff 27 instance, such misclassification of a majority class for a minority class can be attributed as a side-effect of the high number of pseudo-labeled images as the Dragendorff 27 typology, which, since this model leverages the noisy student approach, can cause a substantial bias towards this class.

6 CONCLUSION AND FUTURE RESEARCH DIRECTIONS

In this paper, we investigated, in a systematic way, the application of deep convolutional neural networks to the task of classification of standardized black-and-white pottery diagrams into preexisting typologies defined by pottery experts. The developed automatic classifying systems were trained over a small labeled dataset, which was later further extended by a set of unlabeled images extracted from various publications. Different convolutional architectures were explored, namely the EfficientNet and DenseNet architectures, structured in two different manners (i.e., single and double input), while leveraging diverse data augmentation techniques. The best classification performance was achieved with the DenseNet-201 architecture leveraging the AugMix technique and additional pseudo-labeled data. However, due to both the smallness and unbalancedness of our labeled dataset, the achieved final f1-score performance was of 60.51%, a value that can potentially be improved with additional labeled examples extracted in close contact with in-domain experts.

From the performed experiments and their results, additional paths for the application of convolutional neural networks in the classification of pottery into typologies were made evident. One such path is the classification of pottery photographs, in opposition to the black-and-white diagram format studied in this paper. The photograph format, due to some additional elements such as color, has the potential to improve the classification performance: since our automatic classification systems were developed leveraging models pre-trained on the ImageNet dataset, a collection of photographs in color, data instances with a format similar to such a dataset would be more amenable for convolutional neural network training. Experiments exploring the application of the techniques introduced in this paper leveraging one such dataset, obtained from an available online catalog, are being performed concomitantly.

Taking into account the comparatively weak results attained by the double-input versions of the studied convolutional neural network architectures, a decrease in performance that can be attributed to the smallness of the data instance resolution employed by us, and the potential of the approach for success in this image classification task, experiments with higher resolutions (e.g., 300×300) for both images are an additional venue for improvement. With respect to the EfficientNet-B3 architecture, performing substantially worse than the DenseNet-201, an increase in the scaling factor of the architecture (i.e., testing an EfficientNet-B4) could lead to better results, however, such architecture cannot be tested presently due to its computational cost when paired with the AugMix data augmentation technique with possible experiments leveraging such architecture potentially being computationally feasible in the near future.

Finally, additional data augmentation techniques such as GridMask [6] and FMix [14], class-weighted loss functions [10], or current state-of-the-art techniques in the context of semi-supervised learning [38] also have the potential to improve the performances reported in this paper without requiring new labeled instances.

7 ACKNOWLEDGEMENTS

We would like to thank Prof. Alfredo Jimeno, Director of Numantia Archaeological Site, who has kindly allowed us to use unpublished data for this experiment, and also to Diana Vega-Almazán for her bibliographical recommendations on *terra sigillata*. This research was supported by Fundação para a Ciência e Tecnologia (FCT), through research grant DigCH-HJ-253525.

REFERENCES

[1] Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4).

[2] Berrocal-Rangel, L., Seco, P., and Triviño, C. (2002). *El Castiellu de Llagú (Latores, Oviedo). Un castro astur en los orígenes de Oviedo*. Real Academia de la Historia.

[3] Bertran, F. T. (1991). *La terra Sigillata de Clunia. Una propuesta metodológica para el estudio de las producciones altoimperiales*. PhD thesis, Universitat de Barcelona.

[4] Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6).

[5] Casanova, M. L. B. and Ruiz, J. R. (2011). Memòria de la intervenció arqueològica a l'Ateneu Santboià.

[6] Chen, P., Liu, S., Zhao, H., and Jia, J. (2020). Gridmask data augmentation. *arXiv e-prints*, arXiv:2001.04086.

[7] Christmas, J. and Pitts, M. (2018). Classifying and visualising roman pottery using computer-scanned typologies. *Internet Archaeology*, 50.

[8] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3).

[9] Crawford, A. (2014). Ceramics, roman imperial. In Smith, C., editor, *Encyclopedia of Global Archaeology*. Springer New York.

[10] Cui, Y., Jia, M., Lin, T., Song, Y., and Belongie, S. J. (2019). Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[11] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[12] de Catalan, M. A. M. (1961). *Terra sigillata hispánica*. Valencia: The William L. Bryant Foundation.

[13] Ding, J., Ren, X., Luo, R., and Sun, X. (2019). An Adaptive and Momental Bound Method for Stochastic Learning. *arXiv e-prints*, arXiv:1910.12249.

[14] Harris, E., Marcu, A., Painter, M., Niranjan, M., Prügel-Bennett, A., and Hare, J. (2020). Fmix: Enhancing mixed sample data augmentation. *arXiv e-prints*, arXiv:2002.12047.

[15] Hawkes, C. and Hull, M. (1947). *Camulodunum: First Reports on the Excavations at Colchester 1930-1939*. Oxford University Press.

[16] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[17] Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. (2019). AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty. *arXiv e-prints*, arXiv:1912.02781.

[18] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv e-prints*, arXiv:1704.04861.

[19] Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[20] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2016). Densely Connected Convolutional Networks. *arXiv e-prints*, arXiv:1608.06993.

[21] Itkin, B. (2019). Ranking Systems for Computer-Aided Single-View Pottery Classification. Master's thesis, Tel Aviv University.

[22] Jimeno, A., Liceras-Garrido, R., Quintero, S., and Chain, A. (2018). *Unraveling Numantia: Celtiberian and Roman Settlement (Soria, North-Central Spain)*, pages 199–220. Cambridge Scholars Publishing.

[23] Khan, S., Rahmani, H., Shah, S. A. A., and Bennamoun, M. (2018). *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan & Claypool Publishers.

[24] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

[25] Kottman, J. (1999). *Cities in Sherds 2 Catalogue*, pages 939–1028. Stichting Promotie Archeologie.

[26] Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2).

[27] Orton, C. and Hughes, M. (2013). *Pottery in Archaeology*. Cambridge University Press.

[28] Preciado, J. C. S. (1997). *La terra sigillata hispánica del municipium augusta Bilbiliss*. PhD thesis, Universidad de Zaragoza.

[29] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[30] Rother, C., Kolmogorov, V., and Blake, A. (2004). GrabCut: interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3).

[31] Roumens, M. R. (1982). Breve Introducción al Estudio de la Sigiliata. *Cuadernos de Prehistoria y Arqueología de la Universidad de Granada*, 7.

[32] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3).

[33] Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[34] Serra, J. (1983). *Image Analysis and Mathematical Morphology*. Academic Press, Inc.

[35] Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*.

[36] Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical networks for few-shot learning. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[37] Sobel, I. and Feldman, G. (1973). *Pattern Classification and Scene Analysis*, pages 271–272. Wiley.

[38] Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C., Cubuk, E. D., Kurakin, A., and Li, C. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Proceedings of the Neural Information Processing Systems Conference*.

[39] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1).

[40] Stibbe, C. M. (1998). Exceptional shapes and decorations in Laconian pottery. *British School at Athens Studies*, 4:64–74.

[41] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. (2018). MnasNet: Platform-Aware Neural Architecture Search for Mobile. *arXiv e-prints*, arXiv:1904.09925.

[42] Tan, M. and Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv e-prints*, arXiv:1905.11946.

[43] Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. (2020). Rethinking Few-Shot Image Classification: a Good Embedding Is All You Need? *arXiv e-prints*, arXiv:2003.11539.

[44] Tyukin, I., Sofeikov, K., Levesley, J., Gorban, A. N., Allison, P., and Cooper, N. J. (2018). Exploring automated pottery identification [Arch-I-Scan]. *Internet Archaeology*, 50.

[45] van der Maaten, L., Boon, P., Lange, G., Paijmans, H., and Postma, E. (2006). Computer vision and machine learning for archaeology. In *Proceedings of Computer Applications and Quantitative Methods in Archaeology*.

[46] van der Maaten, L. and Hinton, G. (2008). Visualizing High-Dimensional Data using t-SNE. *Journal of Machine Learning Research*, 9.

[47] Wang, Y., Chao, W.-L., Weinberger, K. Q., and van der Maaten, L. (2019). SimpleShot: Revisiting Nearest-Neighbor Classification for Few-Shot Learning. *arXiv e-prints*, arXiv:1911.04623.

[48] Xie, Q., Hovy, E., Luong, M.-T., and Le, Q. V. (2019). Self-training with Noisy Student improves ImageNet classification. *arXiv e-prints*, arXiv:1911.04252.

[49] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). Mixup: Beyond Empirical Risk Minimization. *arXiv e-prints*, arXiv:1710.09412.