entropy

Article

Learning from Scarce Information: Using Synthetic Data to Classify Roman Fine Ware Pottery

Santos J. Núñez Jareño [1], Daniël P. van Helden [2], Evgeny M. Mirkes [1,3], Ivan Y. Tyukin [1,3,*] and Penelope M. Allison [2]

[1] School of Mathematics and Actuarial Science, University of Leicester, Leicester LE1 7RH, UK; sjjj1@leicester.ac.uk (S.J.N.J.); em322@leicester.ac.uk (E.M.M.)
[2] School of Archaeology and Ancient History, University of Leicester, Leicester LE1 7RH, UK; dpvh2@leicester.ac.uk (D.P.v.H.); pma9@leicester.ac.uk (P.M.A.)
[3] Laboratory of Advanced Methods for High-Dimensional Data Analysis, Lobachevsky University, 603105 Nizhni Novgorod, Russia
[*] Correspondence: i.tyukin@leicester.ac.uk; Tel.: +44-116-252-5106

Abstract: In this article, we consider a version of the challenging problem of learning from datasets whose size is too limited to allow generalisation beyond the training set. To address the challenge, we propose to use a transfer learning approach whereby the model is first trained on a synthetic dataset replicating features of the original objects. In this study, the objects were smartphone photographs of near-complete Roman *terra sigillata* pottery vessels from the collection of the Museum of London. Taking the replicated features from published profile drawings of pottery forms allowed the integration of expert knowledge into the process through our synthetic data generator. After this first initial training the model was fine-tuned with data from photographs of real vessels. We show, through exhaustive experiments across several popular deep learning architectures, different test priors, and considering the impact of the photograph viewpoint and excessive damage to the vessels, that the proposed hybrid approach enables the creation of classifiers with appropriate generalisation performance. This performance is significantly better than that of classifiers trained exclusively on the original data, which shows the promise of the approach to alleviate the fundamental issue of learning from small datasets.

Keywords: deep learning; machine learning; image classification; simulated data; learning from scarce information; Roman pottery (*terra sigillata*)

MSC: 68T10; 68T45

check for updates

Citation: Núñez Jareño, S.J.; van Helden, D.P.; Mirkes, E.M.; Tyukin, I.Y.; Allison, P.M. Learning from Scarce Information: Using Synthetic Data to Classify Roman Fine Ware Pottery. *Entropy* **2021**, *23*, 1140. https://doi.org/10.3390/e23091140

Academic Editor: Gholamreza Anbarjafari

Received: 8 June 2021
Accepted: 17 August 2021
Published: 31 August 2021

1. Introduction

State-of-the-art deep learning models in artficial intelligence (AI), approaching or even surpassing humans' object classifying capabilities (such as [1,2]), require vast training sets comprising millions of training data [3,4]. According to classical statistical learning theory and other relevant machine learning literature [5–7], these requirements are often necessary to ensure that such advanced learning machines do not merely memorise input-output relationships but also generalize well beyond the data they have been trained on. Indeed, in the simplest binary classification setting with the loss function returning 0 for correct classification and 1 otherwise, expected performance R of the classifier, expressed as a mathematical expectation of the loss function, can be estimated as follows [6]:

$$R \leq R_{\text{train}} + 2\sqrt{2\frac{h\log\left(\frac{2eN_{\text{train}}}{h}\right) + \log\frac{2}{\delta}}{N_{\text{train}}}}, \tag{1}$$

where R_{train} is the empirical mean of the classifier's performance on the training set, N_{train} is the size of the training set, $1 - \delta$ is the probability that the expected behaviour of the classifier is within this bound, and h is a measure of the classifier's complexity (Vapnik-Chervonenkis or VC dimension).

For a feed-forward network with Rectified Linear Unit (ReLU) neurons, the VC dimension of the whole network is larger than the VC dimension of a single neuron in the network. The latter, in turn, equals the number of its adjustable parameters. Modern deep learning models have thousands of ReLU neurons with many thousands of adjustable parameters. For example, ReLU neurons in the fully connected layers of *VGG19* [8] have 4096 adjustable weights, and hence the value of h for such networks exceeds 4096. Thus, if (1) is employed to inform our data acquisition processes, ensuring that the model's expected performance R does not exceed the value of $R_{\text{train}} + 0.1$ with probability at least 0.95 requires

$$N_{\text{train}} \geq 35,238,500 > 35M$$

data points. Sharper upper and lower bounds of h for entire networks can be derived from [9,10]. According to [9] (Theorem 2.1), the value of h for a network with W parameters, k ReLU neurons, and L layers is bounded from above as

$$h \leq 2WL\log(2eWLk) + 2WL^2\log(2) + 2L,$$

which emphasizes the need for large training sets even further.

Unfortunately, sufficiently large and fully annotated datasets are not available in many applications or research fields. In archaeology, for instance, datasets almost never reach such orders of magnitude. Where datasets approaching such a size do exist, they are certainly not comprehensively recorded and photographed, and are not digitally available. Moreover, prior to excavation, these archaeological remains have typically been subjected to various mechanical, chemical, or environmental perturbations that result in their fragmentation (into sherds), introducing a near infinite amount of variability to the dataset that adversely affects the statistical properties of any practical/empirical sample. Because of the immense diversity of breaking patterns resulting from these factors, despite the volumes of pottery collected, even larger datasets are required for the learning machine to generate the features necessary for classification from these data. Few-shot learning alternatives (methods and networks capable of learning from merely a few examples, such as matching or prototypical networks [11,12]), do not address the issue as they either require extensive pre-training on labelled data or assume that data distributions in the AI's latent spaces satisfy some additional constraints [13,14] which may or may not hold for a randomly initiated network. Recent successful applications of deep learning in this area [15] exploit identifiable decorative patterns on ceramic vessels to improve ceramic dating processes, but acknowledge the need for larger numbers of typed artefacts for greater accuracy. Such decorative features are also not always available for all ceramics or other types of artefacts.

In this paper, we propose and empirically verify through extensive computational experiments that much-needed information for training advanced large-scale AI models, including deep neural networks, can be extracted from abundantly available published knowledge of experts (in our case Roman ceramic specialists) and fed into model training pipelines [16]. Not only will this enable us to address the issue of insufficient data but also this will allow us to properly calibrate and control bias in the training data.

To demonstrate the feasibility of the approach, we focus on a particular class of artefacts—pottery remains of a particular fabric type, *terra sigillata*. *Terra sigillata* is a high-quality wheel thrown pottery fabric produced throughout the Roman Empire that can be important for studying cultural differences among past eating and drinking practices [17,18]. We concentrate on *terra sigillata* found in Britain, which was mainly produced in Gaul and the German provinces of the Roman Empire. Focusing on a specific type of artefact enables us to better illustrate the challenge of bias in real-life training data. This is particularly important because of the fact that among the pottery remains of a particular fabric type (e.g., *terra sigillata*) the quantities of each of the available forms are not uniformly distributed across the typological spectrum for that fabric. That is, some forms (or classes) are represented in higher numbers than others. Undoubtedly, this is at least partly the result of past popularity of particular vessel types, and therefore relevant to their diverse uses, or the result of differing physical characteristics of particular classes (some forms are more robust than others), and this might be further complicated by selective collection during post-excavation and then by further biased selection processes in museum collection practices. We are therefore dealing with a dataset that is affected by various factors that push the distribution away from the uniform in unpredictable ways.

By basing the perturbations and variations in our simulated models of the artefacts of interest (in our case *terra sigillata* vessels) on a widely used British handbook of *terra sigillata* [19], we were able to integrate available domain knowledge into our synthetic datasets. The original objects were photographs which we collected from the Museum of London (MoL) as a part of the Arch-I-Scan project. The project, directed by P. Allison and I. Tyukin, out of which the current article flows aims to use thousands of images of pottery fragments to train an AI classifier for Roman fine tablewares focusing on *terra sigillata*. This is high-quality wheel thrown pottery fabric produced throughout the Roman Empire. *Terra sigillata* found in Britain was mainly produced in Gaul and German provinces of Roman Empire and it is this material the project focuses on. From this collection, being one of the most extensive collections of complete or near complete Roman fine ware vessels in Britain, we took 5373 images of 162 different near-complete *terra sigillata* vessels. These original images were used to fine-tune and test our system. Using four popular deep learning architectures, *Inception v3* [20], *Mobilenet v2* [21], *Resnet50 v2* [22] and *VGG19* [8], we evaluated our approach by assessing its performance in the task of predicting the vessel class from a single photograph.

The use of synthetic data is very appealing in other areas as well, such as for deep reinforcement learning [23], bioinformatics [24], and optoelectronic device design [25]. Multiple synthetic datasets [26–28] or engines [29–31] to generate them are now available for AI research. Until recently, the domain gap between the synthetic dataset and the real one usually made synthetic-only training non-competitive, but with today's rendering programs enabling the creation of large datasets of various objects [32,33] and entire cities [34], the generalization error is becoming comparable to that between two similar real-life datasets [35]. Several procedures (for example generative adversarial networks [36,37]) have been proposed to deal with the reality gap. In this article, we focus on domain randomization [38–41] to reduce the gap by encouraging variability of simulation properties, such as object's color, illumination, noise, and so forth. For object detection problems, it has been shown that training part of the system on real data and the remaining with realistic synthetic images achieves a good performance [42].

The novel contribution of this work is that we demonstrate, both qualitatively and quantitatively, the advantage of exploiting simulated datasets in tasks requiring identification of highly variable real-life objects such as archaeological artefacts from modestly-sized sets of photographs taken in realistic settings and without precise calibration of the cameras. The challenge of scarce volumes of data is fundamentally inherent in modern post-classical data analysis [43,44]. Traditionally, this challenge is addressed through aggressive dimensionality and model reduction [45], by resorting to models of relatively low complexity even when simulated data are used [24], or by regularisation [46]. Simulation of data and environments [47] are promising alternatives. Here we show that simulated datasets enable the application of advanced machine learning models such as convolutional deep neural networks to a class of problems in which the volumes of data available for training and testing are incompatibly small relative to the complexities of these models. We show that pre-training on appropriately generated simulated datasets may lead to drastic increase of accuracy of up to 20% relative to baseline models pre-trained on *ImageNet* [3] (Section 3). In view of "no free lunch" theorems [48] (Theorem 7.2, p. 114), knowledge that an algorithm or an approach is successful at a machine learning task is particularly important. Here we present one of such positive results: overcoming the challenge of scarce information could be possible in tasks involving classification of damaged 3D objects from 2D images by using simulated datasets created from textbook drawings.

The rest of the article is organised in four sections. Section 2 details data acquisition processes, the preprocessing of the images, the design of the different experiments and how the different simulated datasets were created. Discussion of the experiment results can be found in Section 3. Finally, in Section 4 the conclusions of this work are reported.

2. Materials and Methods

In this section we will explain the procedure used to create and evaluate the performance of an artificial intelligence classifier of the pottery forms in our dataset of images from the MoL collection. The two main difficulties to overcome were the small number of examples per class and the imbalance in the number of pots in each class in the dataset (see Section 2.1). Even if we produced many images per vessel, we did not have enough different vessels for each class of the dataset to create a classifier for all forms of the established *terra sigillata* classification system based on Dragendorff's work [49]. This issue has been alleviated, to some degree, by aggregating some similar forms into one. Furthermore, rouletted (or "R") variants of Dragendorff forms have been aggregated into their main form as the main difference is a rouletted impression that is usually only visible in a *zenith* view (see Figure 1). For example, our class *Dr18* contains Dragendorff forms: 18, 18–31 and 31, as well as their rouletted variants 18R, 18-31R and 31R.

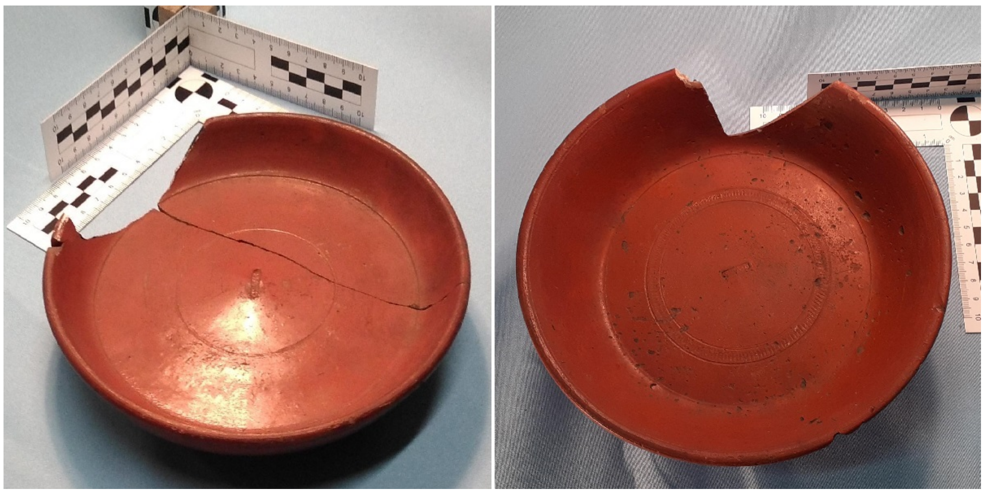


Figure 1. Example of plain variant of the Dragendorff 18 form (**left**) and a rouletted variant of Dragendorff 18-31 (18-31R; **right**). Both are part of our class *Dr18*. Photos taken by Arch-I-Scan with permission from the Museum of London.

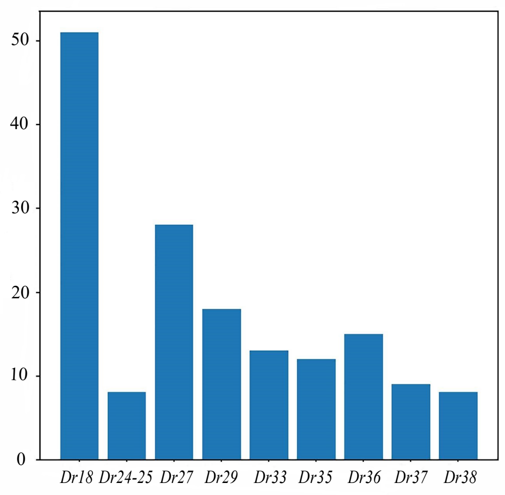
The limited number of pots per class, even where we have combined several Dragendorff forms into a single class, as for *Dr18*, limits the possibilities of extensively evaluating architectures and hyperparameter tuning. Thus, we limited ourselves to four standard architectures (see Section 2.2) and we evaluated the possibility of improving their performance when pre-training them with different simulated pot datasets. The simulated images of pots were used to augment the dataset of photos we had taken in the MoL collection to make sure the size of the training set was closer to the standards required to train AI classifiers. Note that the dataset of real pot photographs only contains 5373 images of 162 different vessels for all pot classes, whereas in the simulation datasets every image is of a different vessel. Each simulated dataset has at least a thousand images per class (see Section 2.3). To ensure the classifier did not become desensitised to the real photos as a result of being overwhelmingly shown synthetic images, it was trained in two phases. In the first pre-training phase, the classifier was trained using synthetic images (or left with the initial *ImageNet* weights in the control condition). After that, a second training was completed using the photos of real vessels.

In setting up the experiment, the following steps were taken:

1. The four neural network architectures used in this experiment were modified from their standard and initialised with the *ImageNet* weights (see Section 2.2).
2. Three different sets of simulated pottery vessels were generated so that the impact of (the quality of) simulation on the classifier's performance could be assessed by comparison. The production of the synthetic datasets is discussed in Section 2.3.
3. Training, validating, and testing the different neural networks was done using the smartphone photographs of real *terra sigillata* vessels from the Museum of London. To make sure these photographs were usable, we created an algorithm which automatically detects the pot, centers it in the photograph and crops out unnecessary surroundings. This process is detailed in Section 2.4.
4. To mitigate the impact of small sample size on our performance metrics, we created 20 different training-validation-test partitions, the creation process of which is detailed in Section 2.5.
5. We then trained each of the combinations of four networks and four sets of initial weights with these partitions.
6. The results of each of 16 combinations of network architectures and pre-training regimes were assessed across the 20 training-validation-test partitions. The definition of the metrics used for this evaluation is discussed in Section 2.6, the results themselves are detailed in Section 3.

2.1. Data Collection

In December 2019, the Arch-I-Scan project research team, with student volunteers from the University of Leicester, took 12,395 photographs of 384 Roman pottery vessels in the MoL antiquarian collection. These comprised complete and near complete vessels of different fineware fabrics. In particular they included 8702 photographs of 247 *terra sigillata* tableware vessels. For the experiments reported in this article, however, only 5373 images of 162 *terra sigillata* vessels were used. This was because for most of the standard Dragendorff forms of *terra sigillata* (see [19] for an introduction to the standard *terra sigillata* classification forms in Britain) we only had very small numbers of vessels, so these poorly represented forms were excluded from this experiment. The dataset used here is therefore much smaller than that recorded, being made up of only those *terra sigillata* forms for which we had 8 or more pots (see Figure 2).



Class	# Pots
Dr18	51
Dr24-25	8
Dr27	28
Dr29	18
Dr33	13
Dr35	12
Dr36	15
Dr37	9
Dr38	8

Figure 2. Bar chart and table detailing the number of different pots per class in our dataset. Classes are labelled starting with *Dr* followed by the number of the Dragendorff form they are based on [19]. The table gives the exact number of vessels per class.

Photographs were taken within three settings all with approximately the same characteristics except for their position and lighting conditions. That is, each setting comprised a 90 degree corner wrapped with a light blue cloth to create a chromatic difference from the usual pot colors (i.e., reddish brown for *terra sigillata* and dark grey for other London finewares). In order to indicate the sizes of vessels in the photos a set of 3D-scales was used (see Figure 3 for an illustration of the setting which we used to take photographs of the pots). Automatic measurement of vessel size or rim falls outside the scope of this article, however.

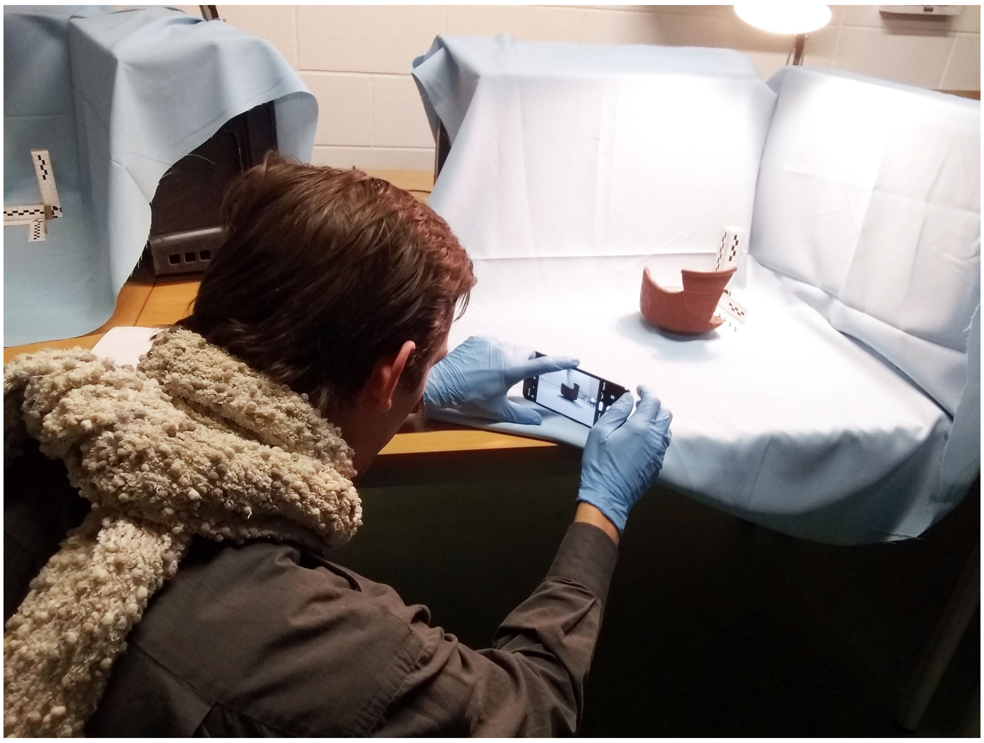


Figure 3. Daniël van Helden photographing a near complete *terra sigillata* vessel, with a complete profile from rim to base, in the antiquarian collection of the Museum of London (photo taken by Victoria Szafara).

To illuminate each setting, the main light source was provided by overhead reading lamps. In some cases, photographs exhibited a colder illumination when the only light source was the museum basement lights. In order not to have a unique direction of the main light source, the lamp was placed in a different position at each setting.

In each setting photographs were taken by a team of two people. As required by MoL regulations, the vessels were carried in and out of each setting by the museum curator, then one member of the team manipulated each vessel for the different views while the other took the photographs. Different smart phones with different camera resolution and optics were used throughout the recording session to reduce the uniformity of the photographs (see Table 1).

Table 1. Mobile phones used to take pictures of pots, in the MoL storerooms, and relevant properties of their cameras.

Phone Model	Camera Resolution	Aperture	Focal Length	Pixel Size
Samsung Galaxy A20e	13 MP	f/1.9	28 mm	1.4 μm
Apple iPhone 6	8 MP	f/2.2	29 mm	1.5 μm
Motorola Moto E Play 5th gen.	8 MP	f/2.0	28.3 mm	1.12 μm
Samsung Galaxy A70	32 MP	f/1.7	26 mm	0.8 μm

The following strategy was used to take photos from a range of different views:

1. With the vessel placed upright on its base and assuming the origin of coordinates is located at the center of the pot, photographs were taken from azimuth angles of 0, 45 and 90 degrees and declination angles 0, 45 and 90 degrees. A last photograph with a declination higher than 90 degrees was taken by resting the mobile on the table.
2. The vessel was rotated by an azimuth angle of 90 degrees and the process of point 1 repeated.
3. The vessel was then turned upside down, thus using the rim to support it, and 4 photographs at azimuth 45 degrees from the declination detailed in point 1 were taken.

This strategy was generally followed, though deviations from this standard were frequent. This means that, while we have standardised photo positions, for certain vessels we have many more photos than described in the process above and for a few vessels we have fewer. This introduces some imbalances in the dataset.

All photographs where standardised and cropped to size in accordance with the procedure described in Section 2.4 and Algorithm 1. No further image alignment procedures were employed including matching of holography points and landmarks [50] or via homography transformations.

As a final step, images were manually labelled (see Figure 4). As well as the museum inventory number for the pot this label includes information about the perspective from which the photo was taken or the vessel condition. The *standard* label was given to those vessels that were standing in their natural orientation and photographed from declination angles of 45°, 90° and greater than 90°. This includes, but is not restricted to, the profile perspective generally used in archaeology. The *zenith* label was given to those photographs that were taken from above with the vessel standing upright as normally. The goal was to achieve angle 0°, but as can be seen from Figure 4, this angle was not always perfect. All those photographs in which the vessel was supported by its rim, irrespective of the angle of the photograph, were given the label *flipped*. Finally, the label *damaged* was given to those vessels with half their azimuth or more in a poor condition.

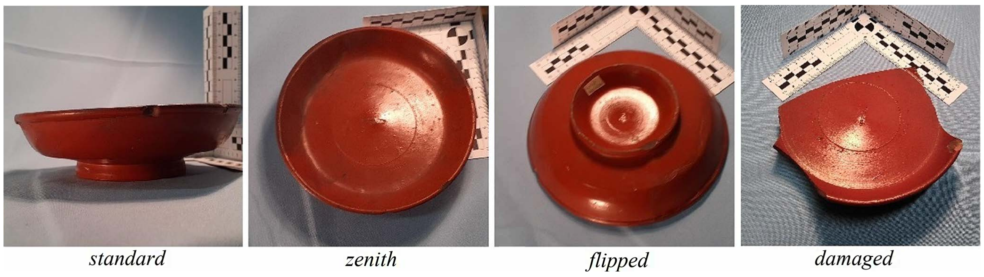


Figure 4. Example of the three different perspectives (*standard*, *zenith*, *flipped*; seen in the three left-most images) and the *damaged* condition (right-most image) using photos from class *Dr18*. Note that images have already been through the automatic cropping procedure detailed in Section 2.4. Photos taken by Arch-I-Scan with permission from the Museum of London.

The pre-processed dataset along with other relevant data can be found in [51].

2.2. Neural Nets Configurations

As we have a very limited number of pots per class, a robust exploration of different architectures and finetuning their hyperparameters is not possible. Therefore, we limited ourselves to standard architectures used in image classification problems, namely: *Inception v3* [20], *Mobilenet v2* [21], *Resnet50* [22] and *VGG19* [8]. We used the *ImageNet* problem [1–3] initialization weights, though we pre-trained with three different simulated datasets in order to improve the performance through domain adaptation. We will show how the pre-training with synthetic pot images results in improvement in the accuracy of the model that increases as we consider more realistically simulated pots (the details of the simulated datasets are discussed in Section 2.3).

With the exception of the image size (224 × 224 for *Mobilenet v2*, *Resnet50 v2* and *VGG19*; and 299 × 299 for *Inception v3*), all nets were trained with the same configuration (see also Figure 5):

- Backbone convolutional neural net base architecture (i.e., the last layers of these architectures were removed until the convolutional structure);
- A global average pooling layer after the convolutional structure;
- A drop out layer with 0.3 exclusion probability;
- A final bottleneck dense layer with softmax activation, that is, a linear dense layer followed by a softmax transformation of the outputs.

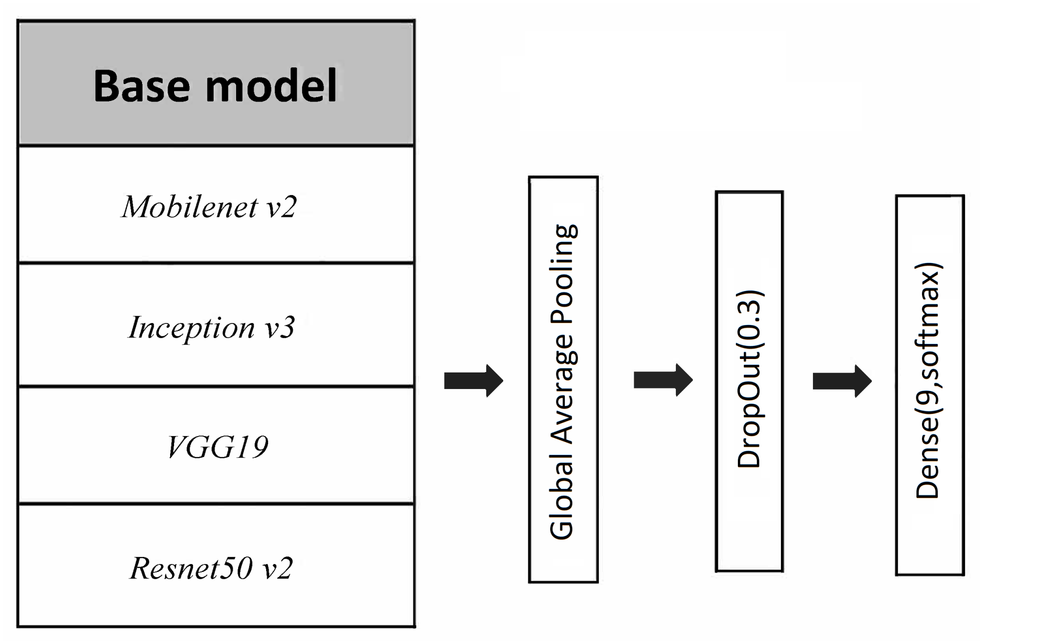


Figure 5. Diagram of the neural nets considered in the article. Four different base architectures are considered: *Mobilenet v2, Inception v3, Resnet50 v2* and *VGG19*. The last layers of these architectures were removed up to the convolutional structure and substituted by a Global Average Pooling layer followed by a Drop Out layer with 0.3 drop probability and a final dense layer with softmax activation and 9 nodes (one for each of the 9 pot classes).

2.3. Pot Simulations

To pre-train the neural nets into a domain closer to our problem, we developed two procedures to generate datasets of simulated pots. The first one uses the Python package *Matplotlib* [52] to generate a simple form of the different classes. The other uses *Blender* [53], an open source 3D modelling tool, to easily create simulated images very close to real images. The rationale behind using different packages to produce simulated images was to investigate how sensitive the overall approach is to different degrees of realism reflected in the simulated data, with *Matplotlib* generating coarse images of pots and *Blender* enabling the generation of realistically looking images (see [51] for the entire dataset we produced using *Matplotlib* and *Blender*). For a comprehensive overview of other image synthesis methods we refer the reader to [54].

Both procedures take as input the profile of each vessel. In order to obtain these profiles we scanned Webster's [19] profile drawings corresponding to the Dragendorff forms present in our dataset. We manually erased all the details that were not part of the rotational symmetric shape and thereby created a collection of digitised black and white drawings of these pot profiles. So, if we define $x_{i,j} \in \{0,1\}$ as the pixel in position (i,j)

with only two possible colors/values: 0 for white/background and 1 black/profile, the border pixel set can be detected as:

$$\text{border} \equiv \{(i,j) : |x_{i,j} - x_{i-1,j}| + |x_{i,j} - x_{i,j-1}| > 0\},$$

which basically computes the vertical and horizontal differences and sums their absolute values. A single difference would not be enough as wherever the border follows a vertical or horizontal trajectory some pixels would not be detected. Finally, we merely have to order the border set, resulting in a list in which for each element its two neighbours correspond to the two pixels closest to its position (i,j). The list of points is used by each procedure to draw an axial section, which is rotated to generate a 3D mesh of points that corresponds to the rotationally symmetric shape of each vessel. Figure 6 shows a diagram of this process (see [16,55,56] for comparable processes).

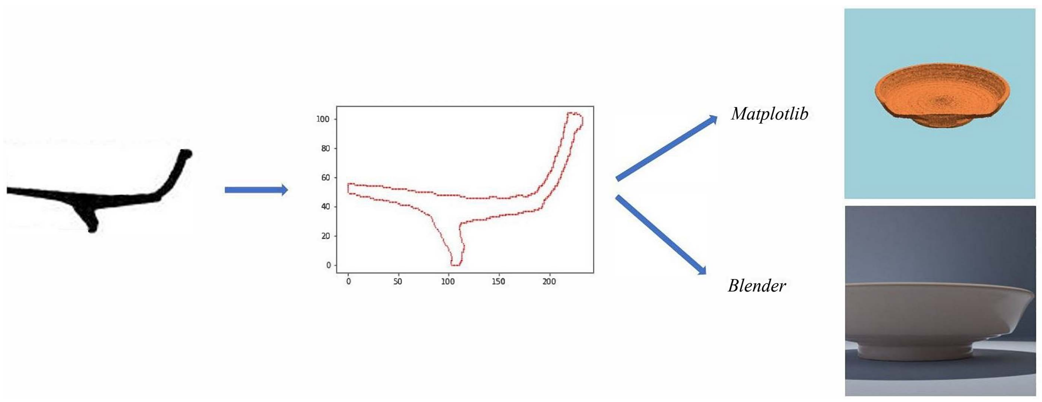


Figure 6. Diagram of the pot simulation. Starting from a digitized profile drawing the profile is extracted as an ordered list of points. With that profile we generated simulated photographs using the package *Matplotlib* or the 3D modelling tool *Blender*.

As we have previously discussed, almost all photographed real pots have at least some degree of damage. Each pot had its own characteristic damage pattern. In order to simulate such breaks we designed an automatic breaking procedure for both software programs. Each break was generated by choosing a random position, P, near the pot mesh. The points of the mesh inside a sphere of radius R from P were susceptible to being removed. After that, we randomly chose n points, $\{p\}$, between distances $r_{\min}$ and $r_{\max}$ from P. We then erased any part of the vessel which is inside the sphere of radius R and closer to P than to a new point, $\{p\}$. This procedure is implemented in different ways depending on the software used. While in *Blender* the points are actually removed from the mesh using the Boolean transformation tool, when using Matplotlib we set the alpha (opacity) channel of these points facets to zero. By modifying the number of breaks and the different parameters (R, n, $r_{\min}$ and $r_{\max}$) we created different random breaking patterns, as well as a variety of viewing angles, that gave us enough variability to create a good simulated training dataset (please see [51] for the full simulated dataset we produced). Figure 7 shows a diagram of this process.

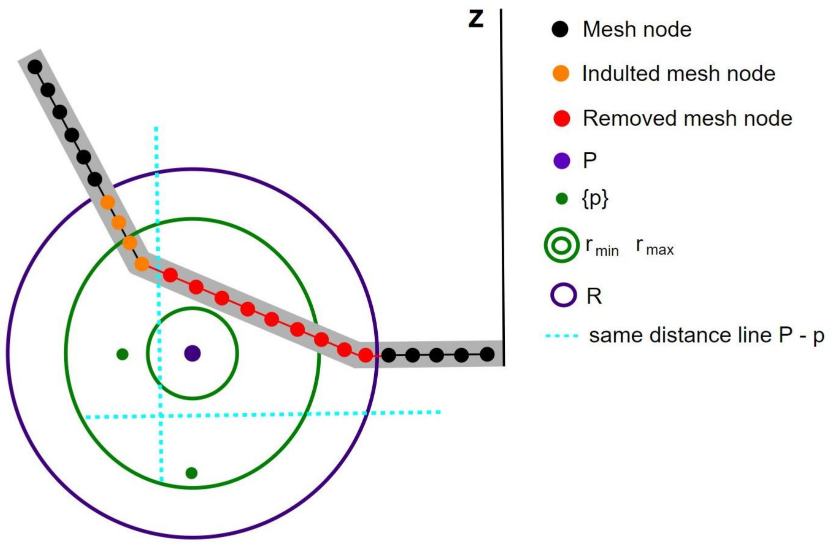


Figure 7. 2D diagram of the breaking procedure. Mesh points that fall within radius R (purple circle) from P (purple point) are susceptible to being removed. Points $\{p\}$ (green points) are inside the crown of radii $r_{\min}$ and $r_{\max}$ (green circles), the lines that separate mesh points closer to $\{p\}$ than P are indicated as a dashed cyan line. Black points in the pot mesh are not susceptible to be removed; orange ones are susceptible, but have been pardoned because they are closer to a green point $\{p\}$ than to P (purple); and lastly, red ones have been removed.

We have created three different simulated datasets:

- *Matplotlib* (1000 images per class):
  To generate this dataset, the Python package *Matplotlib* [52] was used. The simulated pot was floated in a homogeneous background, no shadow is projected but the pot color is affected by the light. The light source is a point far away so only the angle has been changed. The profiles were softened to avoid their small defects creating circular patterns that could be identified by the neural net. For the same reason, a small amount of random noise was added to the surface mesh points position.
- *Blender1* (1400 images per class):
  This dataset was generated using *Blender* [53]. We built a background set close to the original photography setting with a uniform color. The lighting conditions were randomly chosen using the different lighting classes available, namely: sun, spot, point. Thanks to the rendering feature *Cycles*, we could simulate shadow projections as well as changes of illumination in both pot and setting. The profile was softened by distance using the software and no noise was added. The material properties where not changed from the default ones except for the color.
- *Blender2* (1300 images per class):
  The process followed to create this dataset is similar to the one used to create *Blender1*, however, the material properties were changed to make them more similar to the *terra sigillata* pots. Thus, some images exhibit simulated pots with a reflective material which creates light-saturated regions in the image of the pot. We also added decoration and motifs to some classes, namely:
  - *Dr24–25*: half of the images show a striped pattern near the top.
  - *Dr35* & *Dr36*: half of the images show a pattern of four or six leaves on their rim.
  - *Dr29* & *Dr37*: The decoration has a lot of variability in reality. We have simulated it in half of the images through two noise pattern displacements (*Blender Musgrave and Magic textures).*

No other decoration or characteristic details were reproduced in our simulated data. Finally, we sieved each synthetic dataset, removing images that were taken from too close or show some defects resulting from the simulated breaking procedure.

Some samples of the different simulation datasets for all the Dragendorff pot forms included in this study can be seen in Figure 8.

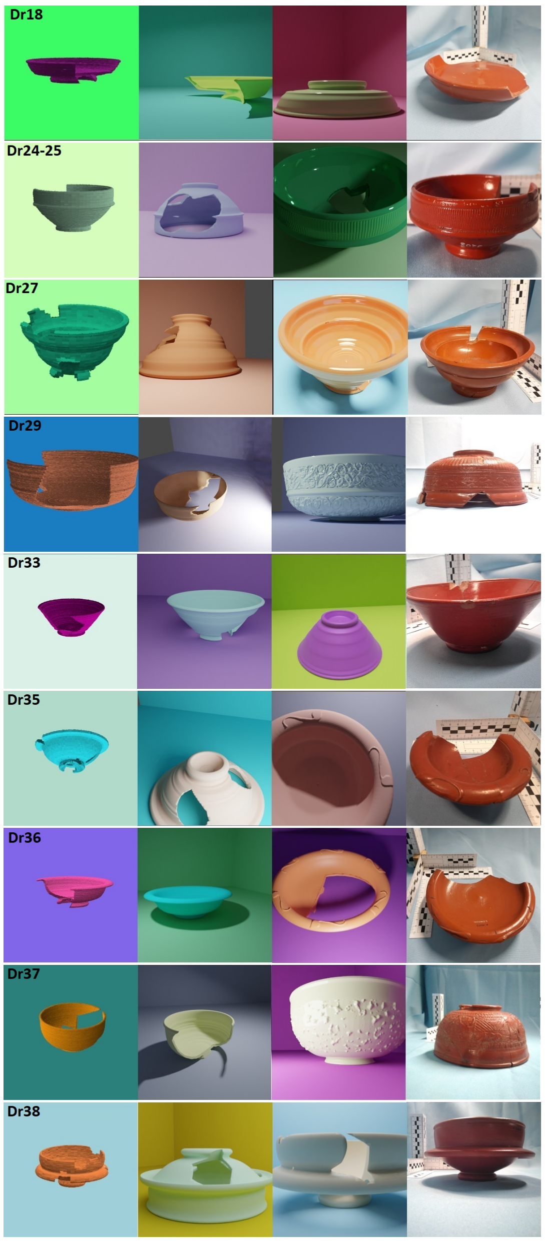


Figure 8. Examples of all simulated datasets and real photograph of our 9 different *Dr* classes. From left to right: *Matplotlib*, *Blender1*, *Blender2*, real photo. Photos taken by Arch-I-Scan with permission from the Museum of London.

2.4. Pot Detector: Cropping and Centering the Images

Although not the main focus of this article, cropping and centering the pot image is a vital step in order to have a good automatic classifier. Object detection is a field that has seen huge development in recent years [57]. As with image classification problems these developments have been propelled by convolutional neural networks. Several architectures and approaches can be considered for an object detection problem, but they should basically cover three steps [57]: generating crops of the image, extracting features out of each crop, and assessing which class the object belongs to.

The controlled conditions in which our photographs were taken mean that our task was relatively simple compared to the complexity of problems which state-of-the-art object detection algorithms were designed to tackle. Due to the uniformity of the image background and the homogeneity of color, *terra sigillata* vessels can be located using color histograms. As we deliberately did not standardise the lighting conditions, however, some parts of the pot were in shadow. To deal with some of the errors introduced in some photographs by these factors, we used a different method. It was still based on color properties and position rather than on pot-shape learning, though.

The algorithm first reduces the image size to 30×30 pixels; this particular size was manually chosen. For each pixel we transformed its color to the Hue-Saturation-Value basis. This kind of pottery has a high Saturation and Value in comparison with the background. For each image a dataset was created where each row corresponded to a pixel and included the pixel position in the x-dimension (pos_x) and y-dimension (pos_y), to encourage spatially compact groups. Thus, the final features were: Saturation, Value, pos_x and pos_y. The features were linearly normalized into the $[0, 1]$ interval. Using the DBSCAN algorithm [58,59] with the maximum distance between two pixels to be considered in the same neighborhood set to $\text{Eps} = 0.1$ and the number of points in the neighborhood (including itself) to be considered a core point set to $\text{MinPts} = 5$, we could group the pixels of the image. After that, we filtered those groups with a small number of pixels, a threshold of 25 pixels was manually tuned. Finally, we exploited the fact that the pot is usually centered in the photograph and computed for each group the average distance to the center of the photograph. We selected as the pot group that with the minimum average distance.

Once the pot was located we made a square crop in the original image with the pot cluster's mean position, $(\overline{pos}_x, \overline{pos}_y)$, in its center. Note that the location of each pixel was scaled to the image original size. Note that, in order to do that we calculated the smallest rectangle containing the cluster and obtained its larger side, L_0. We increased the initial side length, $L = 1.5 \cdot L_0$, to avoid cropping the borders of the pot. An outline of the procedure to locate and crop the pot can be found in Algorithm 1.

Algorithm 1: Locate and crop pot images

```
    Input: Image
    Output: Image_output. Cropped and centered pot image.

scaled_image = scale input Image to $30 \times 30$;
scaled_image = change color basis from RGB to HSV;
dataset_pixels = create dataset where each pixel in scaled_image have features:
  [Saturation, Value, $\text{position}_x$, $\text{position}_y$];
dataset_pixels = linearly normalize dataset_pixels features into [0,1] interval;
pixel_group = assign cluster to each pixel using DBSCAN (Eps = 0.1, MinPts = 5)
  algorithm;
dataset_pixels = dataset_pixels remove pixels of groups with less than 25 pixels;
group_distance = for all groups compute
  $\mathbf{mean}((\text{position}_x - 0.5)^2 + (\text{position}_y - 0.5)^2)$;
pot_group = argmin(group_distance);
dataset_pixels = dataset_pixels remove pixels not in pot_group;
$L_x = \mathbf{max}(\text{position}_x)) - \mathbf{min}(\text{position}_x))$;
$L_y = \mathbf{max}(\text{position}_y)) - \mathbf{min}(\text{position}_y))$;
scale $L_x$ and $L_y$ to the original size;
L = $1.5 \cdot \mathbf{max}(L_x, L_y)$;
L_half = integer_part(L/2);
$\text{center}_x = \mathbf{mean}(\text{position}_x)$;
$\text{center}_y = \mathbf{mean}(\text{position}_y)$;
Image_output = Image[from $\text{center}_x$ − L_half to $\text{center}_x$ + L_half, from $\text{center}_y$
  − L_half to $\text{center}_y$ + L_half];
```

2.5. Training-Validation-Test Partitions

The limited number of vessels we have means that a classifier's performance can be very dependent on whether a certain vessel is allocated to the training, test, or validation set. This leads to a high variance in the performance metrics and, as a result, low confidence in the reliability of the estimations of the classifier's accuracy. To ensure a more robust estimation of the fitted model's accuracy and variance, we generated 20 different training-validation-test partitions and considered the model's performance across these partitions. For each training set we randomly chose four different vessels per class. Another two vessels per class were used for validation, and the remaining vessels were used as the test set. By having the same number of vessels for each class in the training sets we reduced the risk of an unbalanced training, though the number of different vessels available for training was small. We also had a larger test set and, therefore, more reliable error estimations. We used the same 20 training-validation-test partitions to train each of the four network architectures. The average performance across the partitions was used to evaluate the effect of the quality of simulated vessels (see Section 2.3) on the overall performance of the model.

In order to train the networks, we chose to minimize the categorical cross entropy. As was decided during an initial exploration, we used an Adam [60] optimizer with learning rate 5×10^{-5} (except for *VGG19* for which the learning rate was reduced to 5×10^{-6}). The batch size was chosen to be 8 to fit in our computational resources and a patience of 10 epochs was set to stop the training. The weights of the epoch with best validation categorical cross entropy were selected as a training outcome.

2.6. Performance Metrics

The objective of this work was to create an algorithm, $\widehat{G}^{\mathcal{D}} : \mathcal{X} \to \mathcal{G}$, trained with the training dataset $\mathcal{D}$, that, given an image x in the space of pot images, $x \in \mathcal{X}$, maps it into $g \in \mathcal{G}$, where $\mathcal{G}$ is the set of pot classes. The training dataset is a random sample from some probability distribution $\mathcal{D} \equiv \{(x, v, g)\} \sim P_{\text{train}}(X, V, G)$ such that X is random variable representing the image, V is a random variable that takes values in $\mathcal{V}$ the set of all pots (vessels) and G represents the class it belongs to.

Let $f^{\mathcal{D}} : \mathcal{X} \to \mathbb{R}^{\text{card}(\mathcal{G})}$ represent the neural net trained with the training dataset $\mathcal{D}$ that takes an image as input and returns a vector with the prediction scores for each class, $f_i^{\mathcal{D}}(X);\;\; i \in \{1, \ldots, \text{card}(\mathcal{G})\}$ (the cardinality, card, is the number of elements of a set; in our case $\text{card}(\mathcal{G}) = 9$, the number of pot type forms in our dataset), thus we can define our classifier as,

$$\widehat{G}^{\mathcal{D}}(X) = \arg\max_i f_i^{\mathcal{D}}(X).$$

The performance of the algorithm can be assessed using the test dataset $\mathcal{D}^t \equiv \{(x, v, g)\} \sim P_{\text{test}}(X, V, G)$. With $\delta_{i,j}$ being the Kronecker delta (which is 1 if $i = j$ and 0 otherwise) and E the expected value, we can define the accuracy of $\widehat{G}^{\mathcal{D}}$ in P_{test} as

$$\text{acc}[\widehat{G}^{\mathcal{D}}, P_{\text{test}}] \equiv E_{(x,v,g)\sim P_{\text{test}}}\left[\delta_{g,\, \widehat{G}^{\mathcal{D}}(x)}\right]. \tag{2}$$

Given a test dataset, $\mathcal{D}^t$, we can write an estimate for this accuracy as,

$$\widehat{\text{acc}}\left[\widehat{G}^{\mathcal{D}}, \mathcal{D}^t\right] = \frac{1}{\text{card}(\mathcal{D}^t)} \sum_{(x,g)\in\mathcal{D}^t} \delta_{g,\, \widehat{G}^{\mathcal{D}}(x)}.$$

We would like to estimate accuracy (2) for the hypothetical distribution, $P_{\text{test}}(X, V, G)$, of a final user testing the classifier on pots not previously seen by the classifier. However, we only have the dataset that we have collected, which is imbalanced in both number of different pots per class (see Figure 2) and number of photos per pot (from a minimum of 14 to a maximum of 100). If we simply consider each photo as a random sample our metric will depend on the imbalances mentioned. Thus we have to make assumptions about the real distribution $P_{\text{test}}(X, V, G)$ and modify the sampling of our test dataset accordingly. Firstly, we decided that the test sampling probability must not privilege any particular pot or photograph. $\text{Unif}(Z, \mathcal{Z})$ being the uniform distribution of Z over the set $\mathcal{Z}$ that means,

$$\begin{aligned} P_{\text{test}}(X, G, V) &= P_{\text{test}}(X|G, V) \cdot P_{\text{test}}(V|G) \cdot P_{\text{test}}(G) \\ &= \text{Unif}\big(X, \{x \in \mathcal{X} | (x, v, g) \in \mathcal{D}^t; v = V; g = G\}\big) \\ &\quad \cdot \text{Unif}\big(V, \{v \in \mathcal{V} | (x, v, g) \in \mathcal{D}^t; g = G\}\big) \cdot P_{\text{test}}(G). \end{aligned} \tag{3}$$

Secondly, we assigned $P_{test}(G)$ considering two scenarios:

- Uniform prior: all classes have the same probability and, thus, are equally weighted in the final metrics.

$$P_{\text{test}}(G) = \frac{1}{\text{card}(\mathcal{G})} = \frac{1}{9}. \tag{4}$$

- MoL prior: We assume that the MoL pot classes distribution is a good estimator of $P_{\text{test}}(G)$. Let $\mathcal{D}^{\text{dataset}}$ be the full dataset of our collection, the probability will be given by the number of pots of class i over the total number of pots:

$$P_{\text{test}}(G = i) = \frac{\text{card}(\{v \in \mathcal{V} | \exists (x, v, g) \in \mathcal{D}^{\text{dataset}}; g = i\})}{\text{card}(\{v \in \mathcal{V} | \exists (x, v, g) \in \mathcal{D}^{\text{dataset}}\})}. \tag{5}$$

Each prior provides us with an interesting perspective. On the one hand, the uniform prior assesses how well the algorithm has learnt the classes without prioritising any one class. On the other hand, were the MoL prior close to the field frequency of the classes, it would give us scores closer to the user performance perception who would find more frequent classes more often.

Thus, given a test dataset of our collection, $\mathcal{D}^{\text{test-collection}} \equiv \mathcal{D}^{\text{t-c}}$, we will modify the sampling probabilities to fulfill either (3) and (4), or (3) and (5). If $n_{ij}^{(\text{photos})} = \text{card}(\{(x, v, g) \in \mathcal{D}^{\text{t-c}} | g = i; v = j\})$ is the number of photos of pot j that belongs to class i and $n_i^{(\text{pots})} = \text{card}(\{v \in \mathcal{V} | \exists (x, v, g) \in \mathcal{D}^{\text{t-c}}; g = i\})$ is the number of pots of class i, we can write the sampling probability for the test dataset of our collection as,

$$\begin{aligned} P_{\text{samp}}(X = x, V = v, G = g) &= P_{\text{samp}}(X = x | V = v, G = g) \\ &\quad \cdot P_{\text{samp}}(V = v | G = g) \cdot P_{\text{samp}}(G = g) \\ &= \frac{1}{n_{gv}^{(\text{photos})}} \cdot \frac{1}{n_g^{(\text{pots})}} \cdot P_{\text{test}}(G = g); \quad \forall (x, v, g) \in \mathcal{D}^{\text{t-c}}, \end{aligned}$$

where $P_{\text{test}}(G)$ follows one of the prior distributions assumed, namely the uniform prior (4) or the MoL prior (5).

This way we can obtain an expression for the estimated accuracy:

$$\widehat{\text{acc}}\left[\widehat{G}^{\mathcal{D}}, P_{\text{samp}}\right] = \sum_{(x,v,g) \in \mathcal{D}^{\text{t-c}}} \frac{1}{n_{gv}^{(\text{photos})}} \cdot \frac{1}{n_g^{(\text{pots})}} \cdot P_{\text{test}}(G = g) \cdot \delta_{g,\, \widehat{G}^{\mathcal{D}}(x)}. \tag{6}$$

One of the main results of this paper is evaluating the effect of pre-training with synthetic datasets on the performance of the classifier. However, our very limited number of pots would impede reliable estimations of the effect if we restrict ourselves to a single test set. To obtain a more robust estimation let us define an equivalence relation, $\mathcal{R}$, between training datasets randomly generated through the procedure described in Section 2.5. Let the set of all equivalent training datasets be represented by $\mathcal{D}/\mathcal{R}$, using (2) the expected accuracy for an equivalent training dataset can be defined as:

$$\overline{\mathrm{acc}}\left[\widehat{G}, P_{\mathrm{test}}\right] \equiv E_{\mathcal{D}/\mathcal{R}}\left[\mathrm{acc}[\widehat{G}^{\mathcal{D}}, P_{\mathrm{test}}]\right] = E_{\mathcal{D}/\mathcal{R}}\left[E_{(x,g)\sim P_{\mathrm{test}}}\left[\delta_{g,\,\widehat{G}^{\mathcal{D}}(x)}\right]\right]. \tag{7}$$

By generating $n_{\mathrm{splits}} = 20$ train-validation-test splits as explained in Section 2.5, we can give an estimation of (7). $\mathcal{D}_i$ being the training set and $\mathcal{D}_i^{\text{t-c}}$ the test dataset for each partition $i \in \{1, \ldots, n_{\mathrm{splits}}\}$, the estimation can be written as

$$\widehat{\overline{\mathrm{acc}}}\left[\widehat{G}, P_{\mathrm{samp}}\right] = \frac{1}{n_{\mathrm{splits}}} \sum_{i=1}^{n_{\mathrm{splits}}} \sum_{(x,v,g)\in\mathcal{D}_i^{\text{t-c}}} \frac{1}{n_{gv}^{(\mathrm{photos})}} \cdot \frac{1}{n_g^{(\mathrm{pots})}} \cdot P_{\mathrm{test}}(G=g) \cdot \delta_{g,\,\widehat{G}^{\mathcal{D}_i}(x)}.$$

This quantity estimates the bias of the model $\widehat{G}$ when trained with a dataset in $\mathcal{D}/\mathcal{R}$. The variance,

$$\sigma^2_{\mathrm{acc}[\widehat{G},P_{\mathrm{test}}]} \equiv \mathrm{E}_{\mathcal{D}/\mathcal{R}}\left[\left(E_{(x,g)\sim P_{\mathrm{test}}}\left[\delta_{g,\,\widehat{G}^{\mathcal{D}}(x)}\right] - \overline{\mathrm{acc}}\left[\widehat{G}, P_{\mathrm{test}}\right]\right)^2\right],$$

can be estimated in the same spirit by

$$\widehat{\sigma}^2_{\mathrm{acc}[\widehat{G},P_{\mathrm{samp}}]} = \frac{1}{n_{\mathrm{splits}}-1} \sum_{i=1}^{n_{\mathrm{splits}}} \left(\widehat{\mathrm{acc}}\left[\widehat{G}^{\mathcal{D}_i}, P_{\mathrm{samp}}\right] - \widehat{\overline{\mathrm{acc}}}\left[\widehat{G}, P_{\mathrm{samp}}\right]\right)^2.$$

Along with these metrics we can study the confusions between two classes using a normalized confusion matrix, $\widehat{\overline{m}}$, that measures the frequency of predicting $\widehat{G}^{\mathcal{D}}(x) = j$ given that the real class is $G = i$, that is,

$$\widehat{\overline{m}}_{ij}\left[\widehat{G}, P_{\mathrm{samp}}\right] = \frac{1}{n_{\mathrm{splits}}} \sum_{i=1}^{n_{\mathrm{splits}}} \sum_{(x,v,g)\in\mathcal{D}_i^{\text{t-c}}} \frac{1}{n_{gv}^{(\mathrm{photos})}} \cdot \frac{1}{n_g^{(\mathrm{pots})}} \cdot \delta_{g,i} \cdot \delta_{\widehat{G}^{\mathcal{D}_i}(x),j}. \tag{8}$$

Notice that $\sum_j \widehat{\overline{m}}_{ij} = 1$.

3. Results

In this section, we present results of the application of our approach to training four deep nerual networks architectures (*Inception v3*, *Resnet50 v2*, *Mobilenet v2* and *VGG19*) in different pre-training scenarious (*ImageNet*, *Matplotlib*, *Blender1* and *Blender2*). In order to evaluate the effect of pre-trainings, we use various performance metrics described in Section 2.6. We demonstrate the positive effect of the different pre-trainings with simulated photographs in all architectures performance. The effect of pot damage and the photo viewpoint are also assessed as they are important factors to be considered. A sample of trained models can be accessed at [51].

3.1. Accuracy

Two tables summarizing the accuracy results for the 20 partitions can be found in Table 2, for the uniform prior, and in Table 3, for the MoL prior. The best results are obtained by the architecture with *Inception V3* backbone that surpasses the 80% accuracy with the best pre-training (*Blender2*). As can be seen, the pre-training with simulated photographs has a considerable positive effect irrespective of architectures. A graphical visualization of these results can be seen in Figure 9.

The best model for each architecture is the one pre-trained with *Blender2* dataset. The average accuracy is within the two sigma region of the models pre-trained with *Blender1*, though systematically higher for all architectures. The estimated variance, $\hat{\sigma}_{\mathrm{acc}}$, for *Blender2* pre-training is slightly smaller as well. A comparison of the results for each partition can be found in Figure 10, which also shows that the pre-training with *Blender2* dataset could be slightly better than with *Blender1*.

Table 2. Summary of accuracy results using the uniform prior. The different statistics are computed over the 20 test results for each partition (numbers are rounded up to two significant digits after zeros): $\overline{\text{acc}}$ is the average accuracy with its two sigma bootstrap error, σ_{acc} is the variance estimation, min_{acc} is the minimum result and max_{acc} is the maximum. Each statistic is computed for the four architectures (*model*) and all the different initial weights considered (*pre-train*).

Model	**Pretrain**	$\widehat{\textbf{acc}}$	$\hat{\sigma}_{\textbf{acc}}$	$\textbf{min}_{\widehat{\textbf{acc}}}$	$\textbf{max}_{\widehat{\textbf{acc}}}$
Inception v3	*Blender2*	0.818 ± 0.008	0.021	0.78	0.87
Inception v3	*Blender1*	0.809 ± 0.011	0.025	0.76	0.84
Inception v3	*Matplotlib*	0.771 ± 0.014	0.033	0.72	0.85
Inception v3	*ImageNet*	0.719 ± 0.020	0.047	0.61	0.82
Resnet50 v2	*Blender2*	0.779 ± 0.010	0.023	0.74	0.82
Resnet50 v2	*Blender1*	0.765 ± 0.013	0.030	0.72	0.82
Resnet50 v2	*Matplotlib*	0.726 ± 0.014	0.034	0.67	0.82
Resnet50 v2	*ImageNet*	0.667 ± 0.020	0.047	0.55	0.77
Mobilenet v2	*Blender2*	0.769 ± 0.014	0.032	0.72	0.84
Mobilenet v2	*Blender1*	0.752 ± 0.012	0.028	0.70	0.82
Mobilenet v2	*Matplotlib*	0.705 ± 0.017	0.039	0.65	0.78
Mobilenet v2	*ImageNet*	0.655 ± 0.021	0.049	0.56	0.76
VGG19	*Blender2*	0.735 ± 0.010	0.025	0.69	0.77
VGG19	*Blender1*	0.727 ± 0.010	0.024	0.69	0.78
VGG19	*Matplotlib*	0.671 ± 0.017	0.039	0.60	0.74
VGG19	*ImageNet*	0.552± 0.024	0.057	0.47	0.65

Table 3. Summary of accuracy results using the MoL prior. The different statistics are computed over the 20 test results for each partition: $\overline{\text{acc}}$ is the average accuracy with its two sigma bootstrap error, σ_{acc} is the variance estimation, min_{acc} is the minimum result and max_{acc} is the maximum. Each statistic is computed for the four architectures (*model*) and all the different initial weights considered (*pre-train*).

Model	**Pretrain**	$\widehat{\textbf{acc}}$	$\hat{\sigma}_{\textbf{acc}}$	$\textbf{min}_{\widehat{\textbf{acc}}}$	$\textbf{max}_{\widehat{\textbf{acc}}}$
Inception v3	*Blender2*	0.821 ± 0.009	0.020	0.78	0.86
Inception v3	*Blender1*	0.814 ± 0.008	0.020	0.78	0.85
Inception v3	*Matplotlib*	0.763 ± 0.015	0.035	0.71	0.82
Inception v3	*ImageNet*	0.698 ± 0.020	0.047	0.62	0.80
Resnet50 v2	*Blender2*	0.778 ± 0.008	0.018	0.74	0.81
Resnet50 v2	*Blender1*	0.772 ± 0.011	0.025	0.73	0.81
Resnet50 v2	*Matplotlib*	0.719 ± 0.012	0.029	0.66	0.79
Resnet50 v2	*ImageNet*	0.651 ± 0.018	0.042	0.54	0.74
Mobilenet v2	*Blender2*	0.760 ± 0.012	0.028	0.71	0.81
Mobilenet v2	*Blender1*	0.750 ± 0.014	0.032	0.69	0.81
Mobilenet v2	*Matplotlib*	0.699 ± 0.020	0.046	0.59	0.77
Mobilenet v2	*ImageNet*	0.663 ± 0.022	0.052	0.55	0.74
VGG19	*Blender2*	0.732 ± 0.010	0.024	0.68	0.77
VGG19	*Blender1*	0.719 ± 0.011	0.026	0.67	0.77
VGG19	*Matplotlib*	0.656 ± 0.014	0.034	0.58	0.72
VGG19	*ImageNet*	0.532 ± 0.021	0.048	0.46	0.61

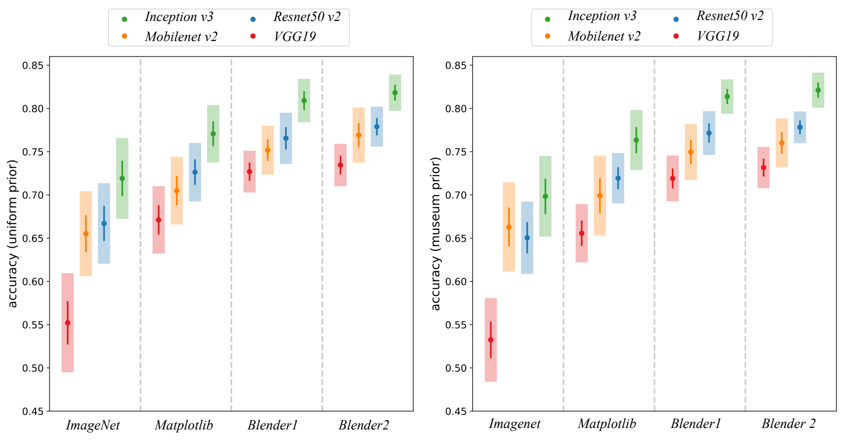


Figure 9. Two plots showing accuracy results for the four architectures considered: *Inception v3* (green), *Resnet50 v2* (blue), *Mobilenet v2* (orange) and *VGG19* (red). The figure on the left shows accuracy using the uniform prior, the one on the right using the MoL prior. The different pre-training regimes considered are displayed along the x-axis. Each point indicates the value of $\widehat{\text{acc}}$ with its two sigma error band draw as a line. The transparent band shows $\pm\hat{\sigma}_{\text{acc}}$.

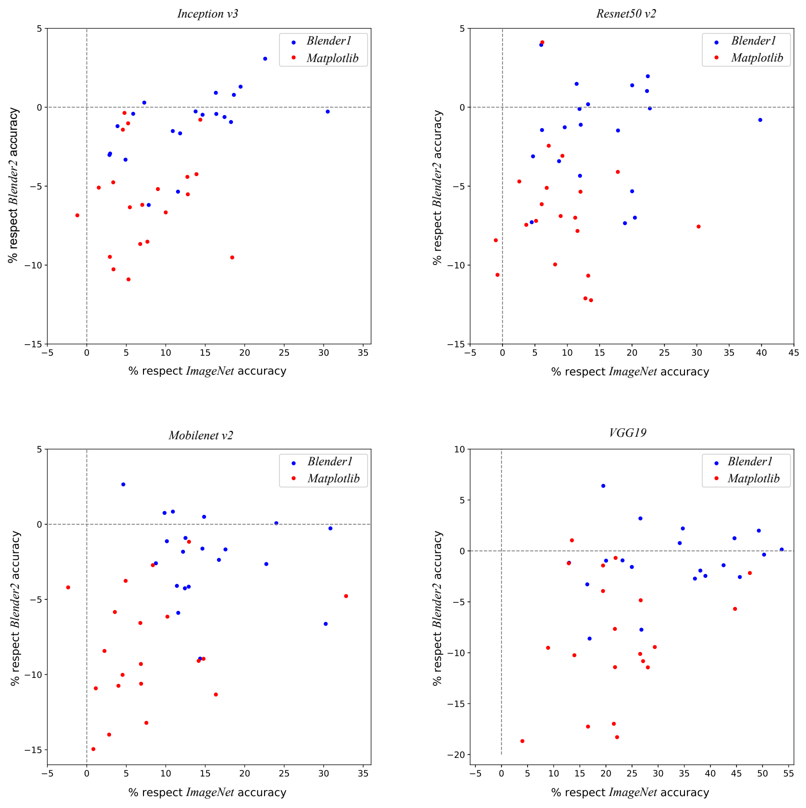


Figure 10. Four plots showing the accuracy results of the pre-training regimes relative to each other, using the uniform prior, for the four networks: *Inception v3* (**superior left**), *Resnet50 v2* (**superior right**), *Mobilenet v2* (**inferior left**) and *VGG19* (**inferior right**). Each point represents a single training-validation-test partition with the system pre-trained with the *Blender* dataset (blue) and *Matplotlib* dataset (red) where their positions are given as the relative percentage with respect to the uniform prior accuracy in the same partition for *Blender2* pre-training (Y axis) and *ImageNet* pre-training (X axis). As can be seen, most of the points are located above 0% with respect to *ImageNet* accuracy and below for *Blender2*.

3.2. *Confusion Matrix*

The normalized confusion matrix defined in Equation (8) shows the main instances of error between two classes of pot. To simplify discussion of the results we will focus on the best architecture, *Inception v3*, for the extreme cases of *ImageNet* and *Blender2* pre-trainings. The normalized confusion matrix, $\widehat{\widehat{m}}_{ij}$, can be seen in Tables 4 and 5 for each of these, respectively. We will consider off-diagonal elements a major instance of confusion when they exceed double the expected probability if the incorrect identifications would have been uniformly distributed across the classes, that is:

$$\widehat{\widehat{m}}_{ij}\left[\widehat{G}, P_{\text{samp}}\right] > 2\,\frac{1-\widehat{\widehat{m}}_{ii}\left[\widehat{G}, P_{\text{samp}}\right]}{n_{\text{classes}}-1}, \tag{9}$$

with $n_{\text{classes}} = 9$ the number of classes in our dataset.

Table 4. Normalized confusion matrix of Equation (8) for *Inception v3* pre-trained with the *ImageNet* dataset. Diagonal elements are highlighted in green, whereas major instances of confusion are marked in red. Major instances of confusion are those off-diagonal elements which exceed double the expected probability if the incorrect identifications would have been uniformly distributed across the classes, see Equation (9).

	Predicted Class								
Real Class	***Dr18***	***Dr24–25***	***Dr27***	***Dr29***	***Dr33***	***Dr35***	***Dr36***	***Dr37***	***Dr38***
Dr18	63	4	2	2	6	2	14	3	3
Dr24-25	5	76	5	1	3	3	2	1	4
Dr27	2	4	74	2	2	9	2	2	4
Dr29	2	2	3	74	2	1	4	7	5
Dr33	3	2	2	1	89	2	0	1	1
Dr35	1	4	11	1	1	67	11	1	3
Dr36	8	4	2	4	1	12	60	2	8
Dr37	2	9	1	11	3	1	1	68	4
Dr38	1	6	2	3	2	2	7	1	76

Table 5. Normalized confusion matrix of Equation (8) for *Inception v3* pre-trained with the *Blender2* dataset. Diagonal elements are highlighted in green, whereas major instances of confusion are marked in red. Major instances of confusion are those off-diagonal elements which exceed double the expected probability if the incorrect identifications would have been uniformly distributed across the classes, see Equation (9).

	Predicted Class								
Real Class	***Dr18***	***Dr24–25***	***Dr27***	***Dr29***	***Dr33***	***Dr35***	***Dr36***	***Dr37***	***Dr38***
Dr18	81	2	1	3	2	1	7	2	1
Dr24-25	3	80	1	2	2	3	2	2	6
Dr27	1	2	87	1	3	3	1	1	1
Dr29	2	2	1	81	2	1	2	7	3
Dr33	0	1	0	1	95	1	0	0	0
Dr35	1	0	4	1	1	80	10	0	2
Dr36	6	2	0	1	1	11	72	1	6
Dr37	1	6	1	7	1	0	0	82	3
Dr38	2	3	2	1	1	3	8	1	79

We can see that pre-training with the simulation photographs seems to have resolved some confusions such as *Dr35* with *Dr27* or *Dr18* with *Dr36*. However, there are other mix-ups that seem to be difficult to correct, for example *Dr35* with *Dr36*. This latter confusion was to be expected as these vessel forms have near identical shape and decoration, the main difference being size and height-radius ratio [19]. A similar thing happens with the confusion between the two decorated forms *Dr37* and *Dr29*. Other confusions are more difficult to explain, for example, *Dr38* with *Dr36*.

3.3. Effects of Damage

As we have previously discussed, some of the pots are severely damaged. Even if a complete profile can be recovered from the vessel, extreme damage can adversely affect the classifier's predictions. In this section, we will evaluate the influence of damage on the performance.

Those vessels with large breaks affecting more than half of their rotational shapes were manually labelled as *damaged*. In order to evaluate the influence of damage, we measured the accuracy for *damaged* and *non damaged* vessels at each of the 20 train-validation-test partitions. The number of *damaged* pots in this dataset is small, however, and can vary at each partition. It is even possible not to have any samples for some classes at a given partition. Therefore, we weight the accuracy for each class, proportional to the number of *damaged* vessels. Thus, with $\widehat{\mathrm{acc}}_i^{(\mathrm{dam})}$ and $\widehat{\mathrm{acc}}_i^{(\mathrm{nodam})}$ being the accuracy described in (6) restricted to *damaged* or *non damaged*) pots of class i. The accuracy per pot class can be seen in Table 6 (*damaged*) and in Table 7 (*non damaged*) for the best model, that is, the model based on *Inception v3* architecture. As we can see, there is a huge general fall in accuracy for *damaged* pots, which pre-training seems to help to mitigate, at least in the case of the *Inception v3* model.

Table 6. Table showing accuracy results of the *Inception v3* model for vessels labelled *damaged*. $\overline{n^{\mathrm{dam}}}$ indicates the average number of damaged pots in each class at a split in the test set, thus lower numbers are less reliable. Notice that the results for damaged vessels of class *Dr35* could not be computed as there are no damaged samples.

			Damaged		
Class	***ImageNet***	***Matplotlib***	***Blender1***	***Blender2***	$\overline{n^{\mathrm{dam}}}$
Dr18	0.48	0.52	0.55	0.60	13.60
Dr24–25	0.31	0.56	0.41	0.36	0.20
Dr27	0.53	0.60	0.69	0.69	4.75
Dr29	0.69	0.73	0.77	0.72	5.55
Dr33	0.81	0.85	0.91	0.93	1.95
Dr35	-	-	-	-	0.00
Dr36	0.53	0.62	0.60	0.63	3.05
Dr37	0.56	0.33	0.69	0.76	0.40
Dr38	0.79	0.74	0.78	0.78	1.05

Table 7. Table showing accuracy results of the *Inception v3* model for *non-damaged* vessels. $\overline{n^{\mathrm{no\ dam}}}$ indicates the average number of *non-damaged* pots in each class at a split in the test set, thus lower numbers are less reliable.

			Non-Damaged		
Class	***ImageNet***	***Matplotlib***	***Blender1***	***Blender2***	$\overline{n^{\mathrm{no\ dam}}}$
Dr18	0.70	0.81	0.90	0.90	31.40
Dr24–25	0.86	0.89	0.85	0.89	1.80
Dr27	0.8	0.90	0.92	0.92	17.25
Dr29	0.79	0.82	0.89	0.88	6.45
Dr33	0.92	0.91	0.95	0.96	5.05
Dr35	0.67	0.69	0.78	0.80	6.00

Table 7. *Cont.*

	Non-Damaged				
Class	***ImageNet***	***Matplotlib***	***Blender1***	***Blender2***	$\overline{n^{\text{no dam}}}$
Dr36	0.64	0.72	0.79	0.77	5.95
Dr37	0.64	0.86	0.79	0.82	2.60
Dr38	0.75	0.81	0.77	0.82	0.95

3.4. Effects of Viewpoint

Another interesting division of the dataset is the viewpoint from which the photograph was taken. We have manually labelled three viewpoints, namely: *standard* view, *zenith* view, *flipped* vessel. An example of each viewpoint can be seen in Figure 4. In both *standard* and *zenith* views the vessel is standing on its base, whereas in *flipped* view it is supported by its rim.

A comparison between the difference in performance per viewpoint of *Inception v3* with *Blender2* and *ImageNet* pre-trainings is shown in Figure 11. Improvement in accuracy seems to have a similar intensity for all viewpoints. This global shift seems to be constant for the whole *ImageNet* accuracy range. Finally, a comparison of the accuracy per pot class and view for the best model *Inception v3* with *Blender2* pre-training can be found in Figure 12.

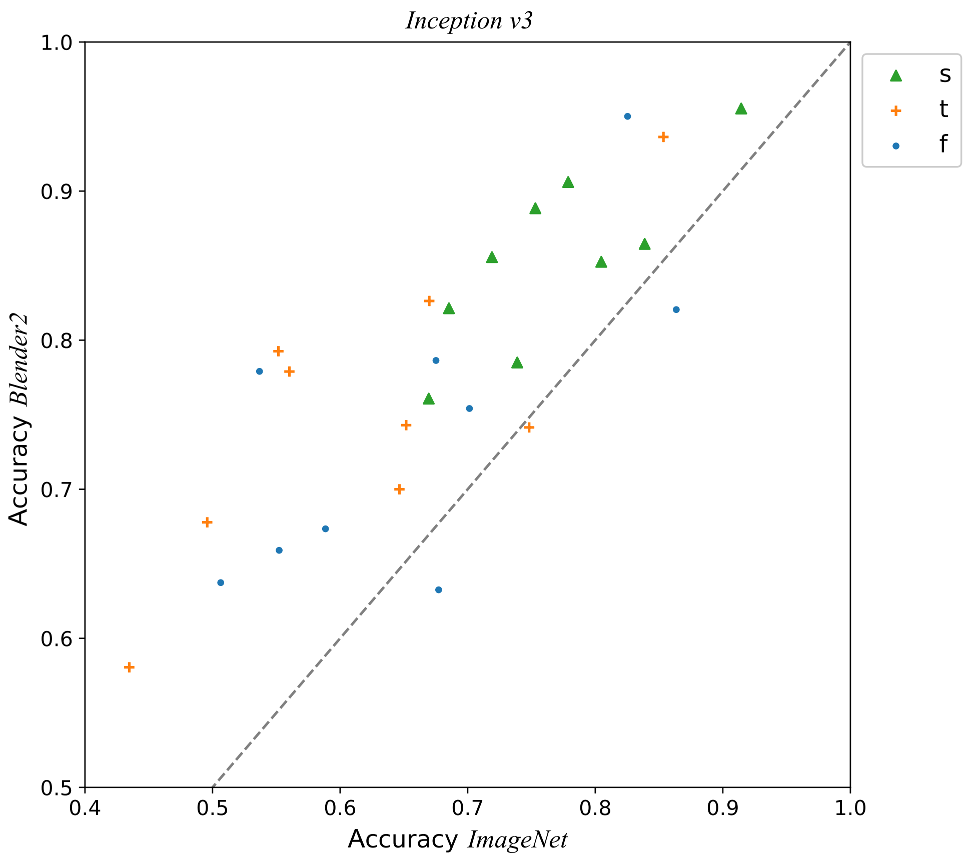


Figure 11. Plot detailing the accuracy of the *Inception v3* model across the different viewpoints and comparing performance with *Blender2* and *ImageNet* pre-training. Each point represents the accuracy for each pot class and viewpoint: *standard* (s, green triangle), *zenith* (t, orange cross), *flipped* (f, blue point). The dashed grey line shows the point where the accuracies of both pre-training regimes are the same. The fact that most of the points are well above these lines shows how pre-training with *Blender2* improves the performance in general. Notice also that the *standard* view shows generally a better performance in both models in comparison with the *zenith* and *flipped* views.

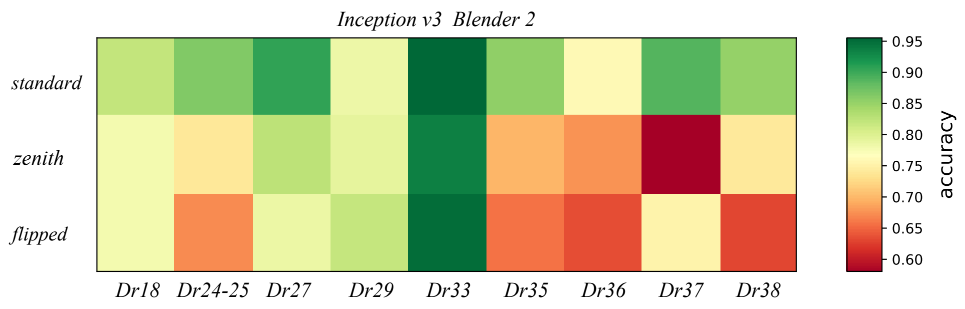


Figure 12. Plot showing the variation in the accuracy of the *Inception v3* model pre-trained with *Blender2* across viewpoints and different pot classes.

The *zenith* view is a special viewpoint as the profile is very hard to see. This fact together with the lower occurrence might explain its worse performance compared to other viewpoints. On the other hand, although the *flipped* view is a very nice viewpoint to get an idea of the profile, we still have a score that is worse than the *standard* view. A possible explanation for this result is the imbalance in the number of images. Across all photographs taken at the MoL, including the ones used for this experiment, the *standard* view outnumbers the *flipped* photographs six to one. Thus, the features created by the machines might be more likely to focus on the *standard* images.

3.5. Reality Gap

An interesting question that arises from the different pre-trainings is whether the neural nets trained with only simulated images are good enough models for classifying real photographs. The difference between the performance of a model trained with real images and the same model trained with simulations is usually called the reality gap [31,38].

In Figure 13 accuracies for the different pot classes are shown for each simulated dataset and architecture. As we can see there are huge differences in the performance of *Matplotlib* and *Blender* datasets. This is to be expected as the quality of the images using *Blender* was improved substantially, see Figure 8. It is also interesting to notice the huge variability per pot class and, how some profiles always have a good performance such as *Dr38*, while others like *Dr18* usually have poor results, though the latter is perhaps not very surprising as this concerns a class into which multiple types have been amalgamated, potentially making it more difficult for the classifier to extract distinguishing features. An aggregated comparison can be found in Table 8 for *Inception v3* model, where we can see that *Blender* datasets outperform *Matplotlib*.

Table 8. Table showing summary statistics for the performance of the *Inception v3* based model trained only with the simulated images of datasets: *Matplotlib, Blender1, Blender2*. The mean corresponds to an estimation of the uniform prior accuracy defined in Section 3.1. We can see the improvement in *Blender* datasets compared with *Matplotlib* dataset.

	Matplotlib		***Blender1***		***Blender2***	
Model	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
Inception v3	0.20	0.28	0.35	0.21	0.33	0.24

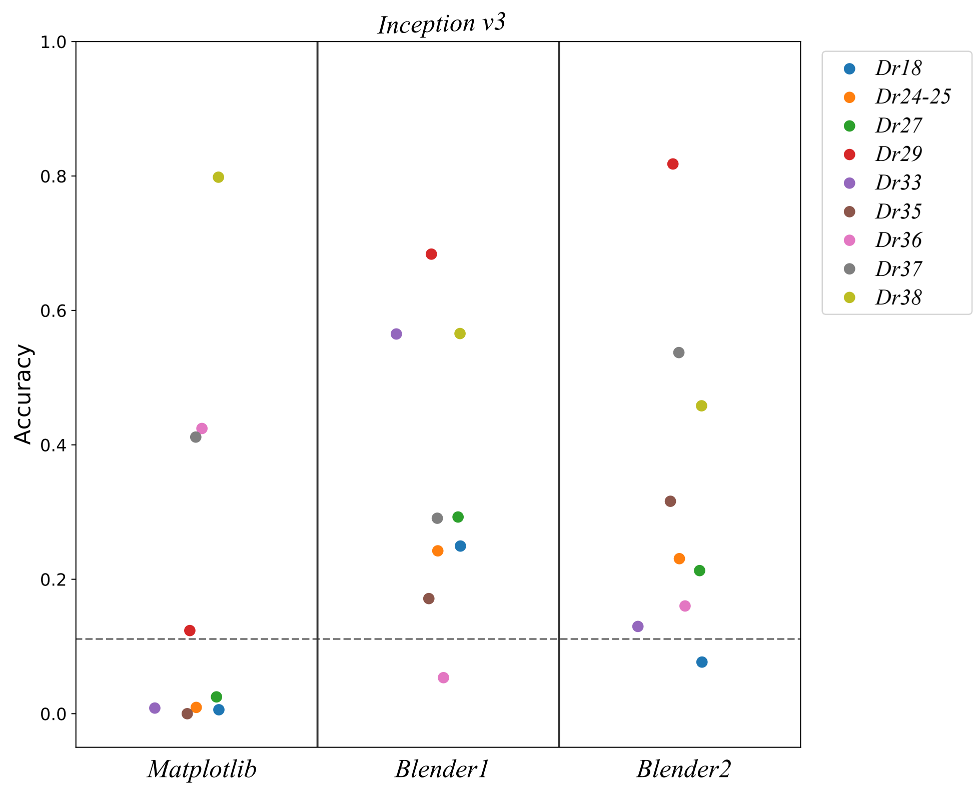


Figure 13. Plot of the *Inception v3* model accuracy for each pre-training for the different pot classes. The horizontal dashed grey line points out the random assignment accuracy 1/9.

4. Conclusions

In this article, we have proposed that creating synthetic datasets can be used as a way of augmenting inherently limited real-world datasets. Through this dataset creation, expert knowledge beyond the identification of real artefacts can be incorporated into the process of machine learning from scarce information. Furthermore, we have demonstrated, through experimentation with different network architectures and the consideration of the impact of photograph viewpoint and damage to the original vessels, that a hybrid approach using synthetic data to pre-train a machine that is subsequently trained with real-world data has the potential to transcend the possibilities of a machine trained exclusively on real-world data. Since the number of fields where non-ideally sized or distributed datasets predominate is likely to significantly outnumber the fields where huge, balanced datasets are readily available, our results signal the expanded potential for machine learning applications in new fields.

Using a dataset of 5373 photos of 162 near-complete Roman *terra sigillata* vessels from the Museum of London, spread unequally over nine typological classes, we have shown that it is possible to create a single image classifier that acceptably generalizes beyond the training set. By digitizing pottery drawings, we were able to include expert knowledge in synthetic data generation, which allowed us not only to augment the size of the dataset, but also to somewhat counteract the skewed distribution in the real-world dataset. Both these aspects are crucial to the potential application of machine learning to domains where there are no perfect datasets. As compared to some other solutions to the problem of classifying archaeological ceramics, such as the ArchAIDE project [16,55,56], we do not rely on the user's input beyond the photograph. As such, we aim to include expert knowledge in training the classifier only, so that as little expertise as possible is required from any potential end user, making any eventual tool more widely usable. We envision that several other applications and tasks requiring the classification of 3D objects can benefit from the same approach, including in the detection of pathologies in medical imaging or in the framework of creating and using digital twins and possibly in the area of virtual or augmented reality.

We have also quantified how other factors, such as the photograph viewpoint or excessive damage to the pot, affect the results. The possible instances of confusion between classes were also discussed and shown how simulation can help to reduce them in some cases. Due to the relatively uniform shapes and the relative simplicity of their simulations, vessel classification provides an excellent test bench for sim2real and 3D shape computer vision experiments. Here, the limitations of the experiments described in this article need to be borne in mind. The shape repertoire that the classifier was confronted with is rather limited and, even when augmented with synthetic images, the total dataset is rather small. It is therefore not the results of the performance that should be taken as an indication of the success of the model, nor the relative performance of the network architectures, but rather the improvement between different training regimes. It is these results that show the potential for using simulation to improve training image classifiers able to generalize beyond the training set in cases where only small or biased datasets are available, be this in archaeological ceramics or beyond.

Author Contributions: S.J.N.J.: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data Curation, Writing—Original Draft, Visualization. D.P.v.H.: Conceptualization, Methodology, Investigation, Data Curation, Writing Original Draft, Writing Review & Editing, Project Administration. E.M.M.: Conceptualization, Validation, Writing Review & Editing, Supervision. I.Y.T.: Conceptualization, Validation, Writing Review & Editing, Supervision. P.M.A.: Conceptualization, Writing Review & Editing, Project Administration, Supervision, Funding Acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to thank the AHRC for funding the Arch-I-Scan project with grant AH/T001003/1. A part of the work on development of machine learning algorithms was supported by the Ministry of Science and Higher Education of the Russian Federation (Project No. 075-15-2021-634).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The pre-processed dataset along with other relevant data can be found in [51]. Note, however, that zip files with images of pots are password protected and accessing the images requires authorisation from the Museum of London. Requests for password should be sent to us by email with a brief description of assumed usage.

Acknowledgments: We would also like to thank the Museum of London and their staff (specifically Roy Stephenson and Nicola Fyfe) for granting us access to the collection, Fiona Seeley for expert pottery advice and identification, and Gabriel Florea and Alessandra Pegurri for helping with photography. We are especially grateful to team member Victoria Szafara for her assistance in this photographing programme.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2012; pp. 1097–1105.
2. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
3. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.-F. ImageNet: A large-scale hierarchical image database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
4. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *European Conference on Computer Vision, Proceedings of the Computer Vision—ECCV 2014, 13th European Conference, Zurich, Switzerland, 6–12 September 2014*; Springer: Cham, Switzerland, 2014; pp. 740–755.
5. Vapnik, V. *Statistical Learning Theory*; Wiley-Interscience: New York, NY, USA, 1998.
6. Bousquet, O.; Boucheron, S.; Lugosi, G. Introduction to statistical learning theory. In *Summer School on Machine Learning, Proceedings of the ML 2003: Advanced Lectures on Machine Learning, ML Summer Schools 2003, Canberra, Australia, 2–14 February 2003*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 169–207.

7. Mohri, M.; Rostamizadeh, A.; Talwalkar, A. *Foundations of Machine Learning*; MIT Press: Cambridge, MA, USA, 2018.
8. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
9. Bartlett, P.L.; Maiorov, V.; Meir, R. Almost linear VC-dimension bounds for piecewise polynomial networks. *Neural Comput.* **1998**, *10*, 2159–2173. [CrossRef] [PubMed]
10. Bartlett, P.L.; Harvey, N.; Liaw, C.; Mehrabian, A. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *J. Mach. Learn. Res.* **2019**, *20*, 1–17.
11. Vinyals, O.; Blundell, C.; Lillicrap, T.; Kavukcuoglu, K.; Wierstra, D. Matching networks for one shot learning. *arXiv* **2016**, arXiv:1606.04080.
12. Snell, J.; Swersky, K.; Zemel, R.S. Prototypical networks for few-shot learning. *arXiv* **2017**, arXiv:1703.05175.
13. Tyukin, I.Y.; Gorban, A.N.; Alkhudaydi, M.H.; Zhou, Q. Demystification of few-shot and one-shot learning. *arXiv* **2021**, arXiv:2104.12174.
14. Tyukin, I.Y.; Gorban, A.N.; McEwan, A.A.; Meshkinfamfard, S.; Tang, L. Blessing of dimensionality at the edge and geometry of few-shot learning. *Inf. Sci.* **2021**, *564*, 124–143. [CrossRef]
15. Pawlowicz, L.M.; Downum, C.E. Applications of deep learning to decorated ceramic typology and classification: A case study using Tusayan White Ware from Northeast Arizona. *J. Archaeol. Sci.* **2021**, *130*, 105375. [CrossRef]
16. Anichini, F.; Dershowitz, N.; Dubbini, N.; Gattiglia, G.; Itkin, B.; Wolf, L. The automatic recognition of ceramics from only one photo: The ArchAIDE app. *J. Archaeol. Sci. Rep.* **2021**, *36*, 102788.
17. Allison, P.M. Understanding Pompeian household practices through their material culture. *FACTA J. Rom. Mater. Cult. Stud.* **2009**, *3*, 11–32.
18. Allison, P.M. The ceramics from the Insula del Menandro and the significance of their distribution. In *'Fecisti Cretaria': Dal Frammento al Contesto: Studi sul Vasellame del Territorio Vesuviano. Studi e Richerche del Parco Archeologico di Pompei*; Ossana, M., Toniolo, L., Eds.; L'erma di BretBretschneider: Rome, Italy, 2020; Volume 40, pp. 199–209.
19. Webster, P. *Roman Samian Pottery in Britain*; Number 13; Council for British Archaeology: London, UK, 1996.
20. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
21. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
23. Sadeghi, F.; Levine, S. Cad2rl: Real single-image flight without a single real image. *arXiv* **2016**, arXiv:1611.04201.
24. Deist, T.M.; Patti, A.; Wang, Z.; Krane, D.; Sorenson, T.; Craft, D. Simulation-assisted machine learning. *Bioinformatics* **2019**, *35*, 4072–4080. [CrossRef]
25. Piprek, J. Simulation-based machine learning for optoelectronic device design: Perspectives, problems, and prospects. *Opt. Quantum Electron.* **2021**, *53*, 175. [CrossRef]
26. Ros, G.; Sellart, L.; Materzynska, J.; Vazquez, D.; Lopez, A.M. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016, pp. 3234–3243.
27. Hwang, H.; Jang, C.; Park, G.; Cho, J.; Kim, I.J. ElderSim: A synthetic data generation platform for human action recognition in Eldercare applications. *arXiv* **2020**, arXiv:2010.14742.
28. Varol, G.; Romero, J.; Martin, X.; Mahmood, N.; Black, M.J.; Laptev, I.; Schmid, C. Learning from synthetic humans. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4627–4635.
29. Gupta, A.; Vedaldi, A.; Zisserman, A. Synthetic data for text localisation in natural images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2315–2324.
30. Deneke, W.; Plunkett, G.; Dixon, S.; Harley, R. Towards a simulation platform for generation of synthetic videos for human activity recognition. In Proceedings of the International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 12–14 December 2018; pp. 1234–1237.
31. Collins, J.; Brown, R.; Leitner, J.; Howard, D. Traversing the reality gap via simulator tuning. *arXiv* **2020**, arXiv:2003.01369.
32. Xiang, Y.; Kim, W.; Chen, W.; Ji, J.; Choy, C.; Su, H.; Mottaghi, R.; Guibas, L.; Savarese, S. Objectnet3d: A large scale database for 3D object recognition. In *European Conference on Computer Vision, ECCV 2016: Computer Vision—ECCV 2016, 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016*; Springer: Cham, Switzerland, 2016; pp. 160–176.
33. Mo, K.; Zhu, S.; Chang, A.X.; Yi, L.; Tripathi, S.; Guibas, L.J.; Su, H. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 909–918.
34. Yao, Z.; Nagel, C.; Kunde, F.; Hudra, G.; Willkomm, P.; Donaubauer, A.; Adolphi, T.; Kolbe, T.H. 3DCityDB-a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. *Open Geospat. Data Softw. Stand.* **2018**, *3*, 1–26. [CrossRef]
35. Movshovitz-Attias, Y.; Kanade, T.; Sheikh, Y. How useful is photo-realistic rendering for visual learning? In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 202–217.

36. Bousmalis, K.; Silberman, N.; Dohan, D.; Erhan, D.; Krishnan, D. Unsupervised pixel-level domain adaptation with generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 95–104.
37. Shrivastava, A.; Pfister, T.; Tuzel, O.; Susskind, J.; Wang, W.; Webb, R. Learning from simulated and unsupervised images through adversarial training. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2242–2251.
38. Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 23–30.
39. Peng, X.B.; Andrychowicz, M.; Zaremba, W.; Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 3803–3810.
40. Zakharov, S.; Kehl, W.; Ilic, S. Deceptionnet: Network-driven domain randomization. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Korea, 27–28 October 2019; pp. 532–541.
41. Mehta, B.; Diaz, M.; Golemo, F.; Pal, C.J.; Paull, L. Active domain randomization. In Proceedings of the Conference on Robot Learning, Online, 16–18 November 2020; pp. 1162–1176.
42. Hinterstoisser, S.; Lepetit, V.; Wohlhart, P.; Konolige, K. On pre-trained image features and synthetic images for deep learning. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 682–697.
43. Donoho, D. High-dimensional data analysis: The curses and blessings of dimensionality. In Proceedings of the Invited lecture at Mathematical Challenges of the 21st Century (AMS National Meeting), Los Angeles, CA, USA, 6–12 August 2000. Available online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.329.3392 (accessed on 29 August 2021).
44. Gorban, A.N.; Grechuk, B.; Mirkes, E.M.; Stasenko, S.V.; Tyukin, I.Y. High-dimensional separability for one-and few-shot learning. *arXiv* **2021**, arXiv:2106.15416.
45. Moczko, E.; Mirkes, E.M.; Cáceres, C.; Gorban, A.N.; Piletsky, S. Fluorescence-based assay as a new screening tool for toxic chemicals. *Sci. Rep.* **2016**, *6*, 33922. [CrossRef]
46. Liu, B.; Wei, Y.; Zhang, Y.; Yang, Q. Deep neural networks for high dimension, low sample size data. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17), Melbourne, Australia, 19–25 August 2017; pp. 2287–2293.
47. Peshkin, L.; Shelton, C.R. Learning from scarce experience. In Proceedings of the Nineteenth International Conference on Machine Learning, Sydney, Australia, 8–12 July 2002; pp. 498–505.
48. Devroye, L.; Györfi, L.; Lugosi, G. *A probabilistic theory of pattern recognition*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1996; Volume 31.
49. Dragendorff, H. Terra sigillata: Ein Beitrag zur Geschichte des griechischen und römischen Keramik. *Bonn. Jahrbücher* **1895**, *96*, 18–55.
50. Brown, L.G. A survey of image registration techniques. *ACM Comput. Surv. (CSUR)* **1992**, *24*, 325–376. [CrossRef]
51. Núñez Jareño, S.J.; van Helden, D.P.; Mirkes, E.M.; Tyukin, I.Y.; Allison, P.M. Arch-I-Scan Data Repository. 2021. Available online: https://github.com/ArchiScn/Access (accessed on 29 August 2021).
52. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
53. Blender Online Community. *Blender—A 3D modelling and rendering package*; Blender Foundation, Stichting Blender Foundation: Amsterdam, The Netherlands, 2018.
54. Tsirikoglou, A.; Eilertsen, G.; Unger, J. A survey of image synthesis methods for visual machine learning. In *Computer Graphics Forum*; Wiley Online Library: Hoboken, NJ, USA, 2020; Volume 39, pp. 426–451.
55. Itkin, B.; Wolf, L.; Dershowitz, N. Computational Ceramicology. *arXiv* **2019**, arXiv:1911.09960.
56. Anichini, F.; Banterle, F.; Garrigós, J. Developing the ArchAIDE application: A digital workflow for identifying, organising and sharing archaeological pottery using automated image recognition. *Internet Archaeol.* **2020**, *52*, 1–48. [CrossRef]
57. Zhao, Z.Q.; Zheng, P.; Xu, S.T.; Wu, X. Object detection with deep learning: A review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef] [PubMed]
58. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd* **1996**, *96*, 226–231.
59. Schubert, E.; Sander, J.; Ester, M.; Kriegel, H.P.; Xu, X. DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Trans. Database Syst. (TODS)* **2017**, *42*, 1–21. [CrossRef]
60. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.