Convolutional Neural Networks for Archaeological Pottery Classification

Tomás Alexandre Pintão de Oliveira

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Bruno Emanuel Da Graça Martins
Prof. Jacinto Paulo Simões Estima

Examination Committee

Chairperson: Prof. Luís Manuel Antunes Veiga
Supervisor: Prof. Bruno Emanuel Da Graça Martins
Member of the Committee: Prof. Arlindo Manuel Limede de Oliveira

January 2021

Acknowledgments

Firstly, I would like to thank my supervisors, Prof. Bruno Martins and Prof. Jacinto Estima, for their constant support and pertinent suggestions during the development of this M.Sc. dissertation.

I would also like to express my gratitude to Prof. Raquel Liceras-Garrido from the University of Lancaster for making this work possible by supplying the original dataset for this dissertation, making crucial contributions in the interpretation of the results of the performed experiments, and for taking the time to help in the process of finding additional data by suggesting relevant publications.

Lastly, I want to thank my family for their love, support, and unbridled optimism, especially during this difficult last year.

This work was supported by Fundação para a Ciência e Tecnologia (FCT), through research grant DigCH-HJ-253525.

Resumo

Os projetos de trabalho de campo arqueológico resultam na recolha de artefactos históricos (por exemplo, fragmentos de cerâmica) que precisam de ser classificados de acordo com categorias estabelecidas, que na área são referidas como tipologias. Estas categorias agrupam objetos com características semelhantes (ou seja, semelhança na forma geral, no carácter das peças componentes como aros e pegas, e na técnica e estilo de decoração), permitindo aos arqueólogos determinar a origem das peças encontradas num local específico (se são autóctones), a sua idade, ou a idade do local.

A categorização das peças de cerâmica é atualmente realizada por arqueólogos através de um procedimento inteiramente manual, envolvendo a análise de ilustrações de linhas padronizadas. Mais recentemente, os artefactos de cerâmica sob a forma de fotografias a cores fornecem um formato alternativo para a classificação da cerâmica. Uma vez que a análise manual levanta problemas para a categorização atempada de um grande número de artefactos, existe interesse em abordagens automatizadas para sugerir tipologias a artefactos arqueológicos.

Com esta motivação, um conjunto de técnicas estado da arte, baseado na utilização de redes neurais convolucionais, é proposto para classificar automaticamente tanto diagramas de linhas a preto-e-branco como fotografias a cores de fragmentos de cerâmica. Uma avaliação abrangente das abordagens propostas é apresentada, discutindo as limitações associadas ao desequilíbrio de classes ou à falta de grandes conjuntos de dados de treino.

Palavras-chave: Arqueologia Computacional, Classificação de Imagens, Meta-aprendizagem, Aprendizagem com Redes Profundas, Visão Computacional, Inteligência Artificial

Abstract

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. More recently, pottery artifacts in the form of color photographs provide an alternative format for pottery classification. 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, a set of state-of-the-art techniques, based on the use of convolutional neural networks, is proposed to automatically classify both standardized black-and-white line diagrams and color photographs of pottery sherds. A comprehensive evaluation for the proposed approaches is reported, discussing limitations associated with class imbalance or lack of large training datasets.

Keywords: Computational Archeology, Image Classification, Meta-learning, Deep Learning, Computer Vision, Artificial Intelligence

Contents

List of Tables

List of Figures

Chapter 1

Introduction

During the course of archaeological fieldwork, one of the most commonly recorded artifacts is pottery vessels, frequently in the form of incomplete broken sherds. The fragments prove to be essential in approaching the study of the everyday life of the ancient societies that produce them, their customs and traditions. Since their inception, during the Neolithic period (approximately 12,000 years ago), ceramic vessels are one of the most prevailing materialities in pre-industrial contexts. In terms of conservation, and considering other objects composed of metals and/or other organic materials, pottery is comparatively stable. Potsherds, in archaeological settings, can provide answers to many questions, with some of the most important ones corresponding to the dating of the artifact, in conjunction with its discovery context, and its specific purpose. In order to better group these ceramic findings, various ceramic type series were developed that allow for the relative dating of different pottery types, with, in some instances, the classification granularity specifying the generation in which the object was assembled (e.g., *terra sigillata*).

In the aftermath of an archaeological excavation, post-excavation analyses are conducted taking into account the types of found artifacts and the questions that archaeologists have previously posed. Summarily, the analysis can be divided into three steps: (i) archaeologists clean the extricated objects and inventory them, (ii) both the shapes and materials of the artifacts are analyzed and manually sketched, (iii) taking into account the previously ascertained object properties typologies are assigned. The aforementioned artifact sketches are produced due to two main properties. First, it is an effective mode of recording the overall formal and decorative characteristics of the extracted artifacts. Secondly, by both schematizing and interpreting the potsherd in the sketched diagram, these black-and-white drawings provide a comparative frame of reference that allows to both share a view of the materials recorded in archaeological works and compare them with other instances from typological series or other archaeological contexts. The aforesaid process is frequently a lengthy one, taking months, sometimes years, with the elapsed time being a function of the fieldwork's object volume. Alternatively, a divergent artifact recording method is frequently employed: color photographs. When compared with the aforementioned sketching procedure, photographs eliminate the need for an expert sketch artist thus ameliorating the cataloging process. On the other hand, such method raises additional difficulties (e.g., background

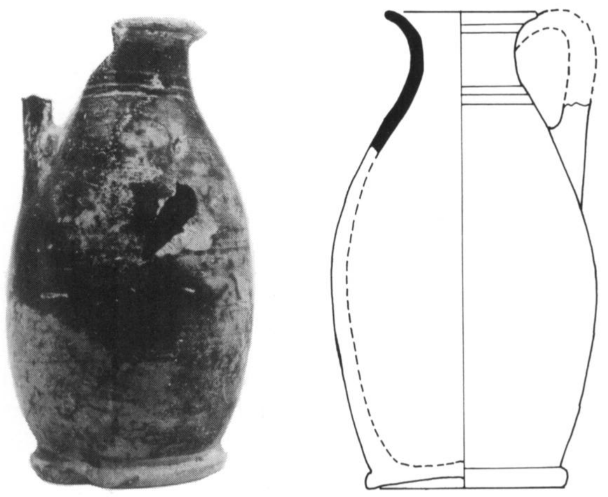
Figure 1.1: Side-by-side example of an artifact's photograph and its corresponding black-and-white line diagram. Image obtained from an article by Stibbe about Laconian pottery [2].

uniformity or fragment representation).

For a successful comparison, concerning the black-and-white diagrams, these must be standardized, following the widely accepted standard divided in two-parts as shown in Figure 1.1. 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. An example of these techniques are machine learning methods leveraging deep neural networks. With the ability to use deep learning, automatic image classification techniques progressed significantly [1]. 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 artifacts in the form of both black-and-white diagrams and color photographs.

1.1 Objectives

The scope of this dissertation is the development of methods for image classification tasks in the context of diagrams and photographs of retrieved archaeological pottery artifacts. These classification tasks consist in categorizing the images in the aforementioned formats into typologies.

More concretely, the principal goal was the experimentation and study of the application of supervised machine learning algorithms to the above-mentioned data domain. These algorithms, specifically deep convolutional neural networks, were studied in conjunction with state-of-the-art techniques (e.g., data augmentation approaches, self-training) for image classification tasks in order to bolster classification performance.

In addition, contrasting convolutional neural network architectures are explored and testing over different possible structures for the inputs to the considered deep neural networks is performed, namely a comparison between single-input and two-input architectures, i.e., one single image containing both views of the object compared with two images each containing only one view.

In conclusion, due to the diminutive nature of the studied datasets, when compared with the volume of data usually required for deep neural network training, this work also aims to explore the effectiveness of state-of-the-art few-shot learning techniques when applied to the pottery classification tasks. As to mitigate the above-mentioned data problems, the use of unlabeled data instances is often considered as a complement to the generally much smaller set of labeled instances and, consequently, is likewise evaluated in this dissertation in the form of self-training techniques.

1.2 Methodology

The introductory tasks of this work were in their totality data related. Concerning the black-and-white diagram dataset, classes that did not respect certain minimum size standards were pruned from the dataset while in parallel procuring additional data instances in order to complete certain minority classes. A similar class pruning process was also performed in the color photograph dataset. Afterward, the unlabeled dataset was obtained through the search and extraction, using semi-automatic methods, of data instances from various archeology publications.

Following the initial data preparation tasks and considering two of the state-of-the-art convolutional neural network architectures, the EfficientNet [3] and DenseNet [4], a survey of the best-performing data augmentation techniques was effected, with MixUp [5] and AugMix [6], after some preliminary tests, emerging as the best performing candidates for the image classification task. The aforementioned data augmentation techniques were then applied in all subsequent tests.

Taking into account the structure of the images of the black-and-white diagram dataset, an alternative two-input structure, corresponding to the two views present in the aforementioned images, for the considered architectures was made apparent and tested as an alternative to the standard single-input neural network.

Afterward, an exploratory study of the state-of-the-art techniques for few-shot learning was performed with two methods presenting the best results. These methods, both leveraging meta-learning approaches where a classification algorithm is applied over representations learned by the convolutional neural networks, were then explored in conjunction with the previously mentioned data augmentation techniques and the different architectures.

Finally, regarding the unlabeled data instances, exclusive to the black-and-white diagram dataset, the state-of-the-art technique Noisy Student [7], which aims to improve image classification accuracy by leveraging unlabeled examples, due to its high performance in the ImageNet classification task, was applied over the models trained with the aforementioned techniques.

The implementation of the various systems produced in the context of this work was achieved by the use of the Python programming language, since it is the *de facto* machine learning language. In additional detail, the Tensorflow[1] deep learning library was the chosen framework for both the implementation of the architectures and their training.

As previously mentioned, the training of the convolutional neural network models was performed leveraging two datasets, namely the black-and-white diagram and color photograph datasets. Both datasets were partitioned in a similar structure where a cross-validation split with 5 folds, with 80% of data instances used for training and 20% for testing. Due to the highly imbalanced nature of both datasets, evaluation metrics such as accuracy are prone to be biased towards the majority classes, consequently, the evaluation of the models trained in the course of this work is achieved through the precision, recall, and f1 metrics. Since the training of all models is performed in a cross-validation manner, each of the aforementioned metrics is computed by averaging the folds' scores.

1.3 Results and Contributions

The principal techniques and contributions of this work can be summarized as follows:

- A method leveraging deep convolutional neural networks for the classification of historical pottery artifacts is proposed. In this method, a convolutional neural network is trained using either black-and-white diagrams or photographs of the extracted artifacts. Particularly, two state-of-the-art convolutional neural network architectures are considered for this effect, namely the DenseNet [4] and the EfficientNet [3]. The proposed network is, before being trained on the domain dataset, pre-trained on the extensive ImageNet [8] dataset.

- Considering the significance of data augmentation techniques in the regularization of neural networks and its particular importance in small datasets, the use of data augmentation methods such as MixUp [5] and AugMix [6] is proposed in the context of the classification of pottery artifacts into typologies. These techniques generate new examples from the original data instances combating, in this way, the tendency of convolutional neural networks to overfit small datasets.

- Taking into account the techniques that have been developed specifically for the few-shot learning setting, an approach based on two of these techniques [9, 10] is proposed in this dissertation with the goal of combating the small number of samples in the pottery artifact datasets. In this method, a certain classifier is applied over the representations learned by the convolutional neural network model, which was trained over the artifact dataset. Several classifiers are proposed, namely k-nearest neighbors, random forest, and logistic regression.

- Leveraging unlabeled data instances available in large quantities is one of the possible paths to improve performance in image classification tasks. The Noisy Student method [7], the state-of-the-art in the ImageNet classification task, provides a procedure for the incorporation of unlabeled images in model training. In this dissertation, the application of the Noisy Student procedure to the training of models using the black-and-white diagram dataset is proposed. In the case of the

[1] `http://www.tensorflow.org/`

considered artifact diagram dataset, first, a model leveraging data augmentation is trained over the available data instances, subsequently, this model is used to label the unlabeled data instances, and posteriorly these newly labeled instances are used for the second phase of training.

- A novel labeled dataset of examples of pottery artifacts diagrams is introduced. The dataset is comprised of, in its majority, artifacts from a set of data from a real-world archaeological excavation scenario. Moreover, due to the small size of certain classes in the aforementioned set of diagram examples, additional black-and-white diagrams were extracted from various academic articles (in the archeology field) in order to complete the aforesaid minority classes.
- The compilation of a dataset of unlabeled black-and-white diagrams depicting recovered archaeological pottery artifacts. These drawings were extricated from various specialized archaeological sources in a semi-automatic manner, totaling approximately 3000 instances. Since the number of labeled diagrams is potentially small, as is the case in this work, an unlabeled set of data instances can conceivably temper some of the training difficulties related to small datasets.
- Considering the multiple mediums in which pottery artifacts can be depicted in, a concomitant exploration of the classification of pottery color photographs is performed. In order to do so, a new color photograph pottery dataset is presented. From an existing online catalog of photographs of pottery artifacts, the new dataset, comprised of a subset of the classes contained in the aforementioned catalog, is formed.
- The above-mentioned methods were tested and evaluated in the considered datasets in a comprehensive series of experiments. The results for the black-and-white pottery diagram dataset report higher performances when leveraging the AugMix data augmentation technique in conjunction with the meta-learning approach and trained following the Noisy Student procedure. In the same manner, regarding the color photograph dataset, using the AugMix data augmentation and employing the meta-learning approach achieved the highest performance.

The best performing model, which leveraged unlabeled images, achieved 60.51% in the f1 score regarding the black-and-white diagram dataset. On the other hand, for the color photograph dataset the highest scoring model, when taking into consideration the f1 score, achieved 77.53%.

The implementation of the various techniques that were studied in this dissertation can be found on a GitHub repository[2].

1.4 Thesis Outline

The remaining sections of this dissertation are arranged in the following manner:

- Chapter 2 defines the elementary concepts and related work. Firstly, Section 2.1 presents an overview of the theoretical background of neural networks and convolutional neural networks in the

[2] `http://github.com/tomas-olliv/cnn-pot`

context of supervised learning. Then, an in-depth study of the two state-of-the-art convolutional neural network architectures considered for this work, EfficientNet and DenseNet, is presented in Section 2.2. Finally, Section 2.3 reviews several techniques that aim to apply automatic learning algorithms to the task of classifying diagrams or photographs of archaeological pottery into typologies.

- Chapter 3 presents the proposed method for the classification of images (photographs or drawn diagrams) of pottery artifacts into typologies. The proposed method is composed of several techniques, which are detailed in this chapter.

- Chapter 4 details the experimental evaluation of the proposed methods. The chapter first presents an analysis of the datasets used in this work, the experimental methodology employed for the performed tests, and an overview of the model optimization strategies chosen for model training. Moreover, the experimental results for the black-and-white diagram dataset are presented and analyzed together with a study of the learned representations by applying the t-SNE technique. In conclusion, the analysis of the results for the experiments regarding the color photograph dataset is presented.

- Finally, Chapter 5 identifies the conclusions emanated from the performed research and establishes some new directions for future work.

Chapter 2

Concepts and Related Work

In this chapter, both the fundamental concepts supporting the developed image classification systems (e.g., convolutional neural networks), as well as related work pertaining to the classification of archaeological pottery diagrams and color photographs into preestablished typologies, are introduced. Firstly, Section 2.1 introduces artificial neural networks and, due to both their suitability and prevalence in solving image classification tasks, convolutional neural networks. Secondly, Section 2.2 reviews the two state-of-the-art convolutional neural network architectures employed in this dissertation: DenseNet and EfficientNet. Finally, four papers that established, in a systematic way, techniques for the classification of diagrams and photos of archaeological pottery artifacts are analyzed.

2.1 Supervised Learning with Deep Neural Networks

Machine learning concerns with automatically learning patterns from a set of data instances, i.e., the available data from a certain problem domain. These data instances, grouped in a training set, are used by the machine learning algorithm to adjust the parameters of a model that reflects the data. When the data instances contained in the training set are labeled, i.e., the category of each instance is known, this additional knowledge can be leveraged using algorithms from the denominated supervised learning approach. A commonly utilized group of techniques that learn from labeled training sets are the so-called artificial neural networks.

2.1.1 The Perceptron

The main unit of computation within an artificial neural network is the perceptron. The perceptron model, using the concept of a neuron proposed by McCulloch and Pitts [11], was first used by Rosenblatt [12] to describe information storage, recollection, and the influence of information on behavior in the human brain. In the standard perceptron model for binary classification, an input vector $\mathbf{x} = x_0, x_1, ..., x_m$ is transformed into an output value of 0 or 1, corresponding to one of two different classes. Each one of the elements of the input vector $\mathbf{x}$ is weighted by the parameters in a weight vector $\mathbf{w} = w_0, w_1, ..., w_m$, producing an output that will later be fed to an activation function, as shown in Equation 2.1. One of the weights, i.e., w_0, is called the bias, and it shifts the hyperplane that separates the two classes of inputs (i.e., the perceptron corresponds to a linear classifier, with the weight vector defining a hyperplane that separates the classes). The value in the input vector $\mathbf{x}$ corresponding to the bias is set to one.

$$\mathrm{net}(x) = \sum_{i=0}^{m} w_i \cdot x_i \tag{2.1}$$

A nonlinear activation function $\mathrm{f}(x)$, that reproduces the firing behavior of a neuron, is used as a threshold to determine the class of the input. The activation function is usually defined as follows for binary classification:

$$\mathrm{f}(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.2}$$

When learning a perceptron model from a set of pre-labeled data instances, the weight vector $\mathbf{w}$ must be updated taking into account the label of the current input and its predicted class. By iteratively visiting different training examples, the perceptron will learn the parameters of the hyperplane separating the data instances through a function that measures the classification error. In order to do so, the error (i.e., the loss) in each iteration can be defined to correspond to the difference between the target value (i.e., the label of the input training data) and the output of the perceptron. An hyperparameter η, called learning rate or step size, is added to the function defining the updates in order to encode how sensitive the weights of the network are to the classification error. A high learning rate will cause the weights to oscillate considerably, and thus this hyperparameter is usually set to a small constant. Equation 2.3 formalizes the way in which the perceptron learns, called the perceptron learning rule:

$$\mathbf{w}^{new} = \mathbf{w} + \eta \cdot (t - o) \cdot \mathbf{x} \tag{2.3}$$

In the previous equation, t is the target and o is the output of the perceptron. In a more general case, in which the activation function or the loss function are defined differently, training can rely on minimizing an error function using the gradient descent method. This algorithm works by taking steps in the direction of the negative gradient of the current point, this way approximating the local minimum in each step. By applying this method, a more general learning rule (Equation 2.4) is reached that allows for greater flexibility in the choice of the function that measures the classification error.

$$\mathbf{w}^{new} = \mathbf{w} - \eta \cdot \nabla E(\mathbf{w}) \tag{2.4}$$

2.1.2 The Multi-Layer Perceptron

In order to obtain good results in classification, a single perceptron is often not enough. This was first made evident by Minsky and Papert [13], which described the inability of a single perceptron to classify data that is not linearly separable. To improve the performance of the classification, multiple perceptrons can be organized in layers, with each of the layers connected by weighted links. In one such multi-layer perceptron architecture, there are three types of layers: input, hidden and output. The first receives the original input and transforms it into an intermediate result. The hidden layers receive the previous layer's output and, using an activation function, produce an output to the next layer (hidden or output). Finally, the output layer receives the input of the previous layer and produces an output that corresponds to a final prediction.

The effectiveness of the multi layer perceptron (MLP) model depends completely on the weights connecting the different layers. Therefore, before an actual classification is done, these weights must be learned from examples for the data that will be classified. This is done by applying a technique called back-propagation, first introduced by Rumelhart et al. [14]. In back-propagation, as in the perceptron learning algorithm, the error of the predictions is minimized. There are several different error functions that can be defined to achieve this goal, depending on the actual goal. Common examples include the cross-entropy function (Equation 2.6) for classification problems, or the mean squared error for regression. For both binary and multi-label classification purposes, the cross-entropy is one of the most commonly used loss functions. Binary cross-entropy is defined with a basis on a likelihood function, i.e., the probability of a given target t (i.e., class) knowing the parameters w of the model, as expressed in Equation 2.5.

$$p(t|w) = \prod_{k=1}^{K} o_k^{t_n} \cdot (1 - o_k)^{1-t_k} \tag{2.5}$$

For expressing this idea as an error function, the negative logarithm of the likelihood is taken. By applying the negative logarithm to Equation 2.5 and using the logarithmic identities, the cross-entropy error equation is reached:

$$E = -\ln\left(p(t|w)\right) = -\sum_{k=1}^{K} (t_k \cdot \ln(o_k) + (1 - t_k) \cdot \ln(1 - o_k)) \tag{2.6}$$

In the previous Equations 2.5 and 2.6, t_k is the target value (i.e., the actual class of observation k) and o_k the current output, with the k subscript indexing the $\mathbf{t}$ and $\mathbf{o}$ vectors both with dimension K (i.e., the total number of observations).

In order to minimize the loss function from Equation 2.6, its gradient is taken with respect to the weights w_{ij} of the network. Using the sigmoid (Equation 2.8 with $\alpha = 1$) as the activation function, and considering that $o_k = \sigma\left(\mathbf{w}^\top \cdot \mathbf{x}_k\right)$, a new expression is reached:

$$\frac{\partial E}{\partial w} = \sum_{k=1}^{K} (o_k - t_k) \cdot \mathbf{x}_k \tag{2.7}$$

The logistic sigmoid function used in the definition of o_k is, in turn, defined as:

$$\sigma(x) = \frac{1}{1 + e^{-\alpha \cdot x}} \tag{2.8}$$

For multi-class classification problems, a commonly used activation function is the softmax function, which, in this context, takes a vector $\mathbf{net}$, comprised of the weighted inputed values for all available M classes and returns, for a certain class i, the computed probability for a given $\mathbf{x}$. Considering $net_i =$

$\mathbf{w}_i^\top \cdot \mathbf{x}$, the predicted probability for class i, using the softmax activation function, is expressed as:

$$\sigma(\mathbf{net})_i = \frac{e^{net_i}}{\sum_{m=1}^{M} e^{net_m}} \tag{2.9}$$

The softmax activation thus provides a probability distribution over M classes. Using Equation 2.9 as a basis for the likelihood function and taking its negative logarithm, employing a similar process to the derivation of the binary cross-entropy error, a loss function for multi-class classification, named categorical cross-entropy, is in this manner defined:

$$E = -\sum_{k=1}^{K} \sum_{m=1}^{M} t_{k,m} \cdot \ln(o_{k,m}) \tag{2.10}$$

In the previous equation, M denotes the total number of classes, $o_{k,m}$ the probability of instance k belonging to class m, and $t_{k,m}$ represents, in binary form, whether instance k truly belongs to class m.

In back-propagation, Equation 2.4 is used in a recursive manner across the network to update the layer's weights. This process is first done in the output layer and then used in the previous layers (until the input layer is reached) through the chain rule of differential calculus. The back-propagation algorithm has thus two phases: the first one transforms the input that is fed to the neural network and outputs a class. The second one adjusts the network's weights to achieve a better classification performance, through the chain rule of differentiation.

The choice of an activation function in the intermediate layers of a neural network has an important effect on the nonlinear mapping that is learned. Comparing two different activation functions, namely the logistic sigmoid (Equation 2.8) and the leaky ReLU (Equation 2.11, where k corresponds to a leak factor controlling the size of the output when the unit is inactive), some differences can be observed in the outputs that they generate. In a sigmoid function the output is contained in $[0, 1]$ while, in the leaky ReLU case, the output has no upper limit.

$$f(x) = \begin{cases} x & x > 0 \\ k \cdot x & x \leq 0 \end{cases} \tag{2.11}$$

The sigmoid activation function has, however, an inherent flaw, namely the vanishing gradient problem. This occurs when the gradient of the activation function becomes exceedingly small (near zero) and, consequently, the weights will not be updated, causing the neural network to stop learning. Back-propagation originates this problem due to the application of the chain rule during parameter updating, with the derivatives of the activation functions of consecutive layers (starting from the final layer) being multiplied throughout the network, causing in some cases the earlier layers to compute very small gradient values, with deeper networks exacerbating this effect. As the derivative of the sigmoid function, expressed in Equation 2.12, outputs values in the interval $]0, 0.25]$, during back-propagation, deep networks composed of layers featuring this activation function will propagate backwards gradients of exponentially diminishing size, with the gradients reaching the first layers nearing zero.

$$\frac{\partial \sigma}{\partial x} = \sigma\left(x\right) \cdot \left(1 - \sigma(x)\right) \tag{2.12}$$

Because the minima of the loss function is needed for the back-propagation algorithm, the search for these points may encounter saddle points. Despite having gradient zero, saddle points are not minima of the loss function, thus causing a slower convergence. In order to solve this problem, some optimization algorithms extend the general idea of gradient descent. One of these improvements is the Nesterov momentum [15], which turns the optimization problem more amenable by taking the gradient with respect to another variable: the momentum of the previous Δw_{ij} is added to w_{ij}. This way, it is possible to circumvent points that, in next iterations, would drift away from the minima.

$$\Delta w_{ij} = \alpha \cdot \Delta w_{ij}^{old} - \eta \cdot \frac{\partial E}{\partial(\alpha \cdot \Delta w_{ij}^{old} + w_{ij})} \tag{2.13}$$

In the previous equation, η is the learning rate, and α represents the momentum. The tuning of the learning rate parameter η must, however, be done manually.

Another widely used optimization algorithm is the Adaptive Moment Estimation (ADAM) approach, proposed by Kingma and Ba [16]. In ADAM, a moving average of the gradients of the weights (Equation 2.14) and its square (Equation 2.15) is used in the weight update. These rolling averages allow the learning algorithm to mitigate the risk of a detour in the minima search. In order to be able to adjust the decaying rates of both moving averages, two hyperparameters are used, namely γ_1 and $\gamma_2 \in [0, 1]$.

$$MA\left[\frac{\partial E}{\partial w_{ij}}\right] = (1 - \gamma_1)\frac{\partial E}{\partial w_{ij}} + \gamma_1 MA\left[\frac{\partial E}{\partial w_{ij}}\right]^{old} \tag{2.14}$$

$$MA\left[\frac{\partial E}{\partial w_{ij}}^2\right] = (1 - \gamma_2)\frac{\partial E}{\partial w_{ij}}^2 + \gamma_2 MA\left[\frac{\partial E}{\partial w_{ij}}^2\right]^{old} \tag{2.15}$$

In an effort to mitigate very small gradient values, an estimation of these values is used instead, as shown in Equations 2.16 and 2.17.

$$\hat{MA}\left[\frac{\partial E}{\partial w_{ij}}\right] = \frac{MA\left[\frac{\partial E}{\partial w_{ij}}\right]}{1 - \gamma_1} \tag{2.16}$$

$$\hat{MA}\left[\frac{\partial E}{\partial w_{ij}}^2\right] = \frac{MA\left[\frac{\partial E}{\partial w_{ij}}^2\right]}{1 - \gamma_2} \tag{2.17}$$

Using the previous result, the final update rule is shown in Equation 2.18.

$$w_{ij} = w_{ij}^{old} - \hat{MA}\left[\frac{\partial E}{\partial w_{ij}}\right] \frac{\eta}{\sqrt{\hat{MA}\left[\frac{\partial E}{\partial w_{ij}}^2\right]} + \epsilon} \tag{2.18}$$

In the previous equations, MA is the moving average and ϵ is a parameter corresponding to a small number, to prevent division by zero.

The aforementioned optimizations intend to improve the learning algorithm. There are, however,

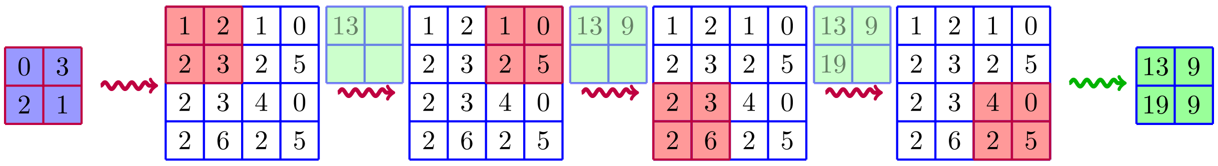
Figure 2.1: Example of a convolution operation. The kernel, shown in blue, is applied with a stride of size two, leading to a feature matrix shown in green.

other aspects of the learning process that can be enhanced, such as possible cases of overfitting. A model is said to overfit when it is too specific to its training data. A commonly used technique that tackles this issue is dropout [17], which works by deactivating neurons with a certain constant probability during the training phase. In this way, every neuron has a better chance to contribute to the final predictions, this way adding more generality to the produced model.

2.1.3 Convolutional Neural Networks

In the previous subsection, multi-layer perceptrons and general concepts related to training neural networks were discussed. Image classification problems often rely on a particular type of neural network, referred to as convolutional neural network. This type of network uses a mathematical operation called convolution as the main computation, instead of the standard matrix multiplication used in multi-layer perceptrons. Convolutional filters, in addition to other types of layers, are used to obtain the main features of the input data. This is particularly relevant in the learning of data that is grid-structured (e.g., two-dimensional images). Convolutional filters are repeatedly applied in the input matrices, providing in this way a new output. The filters are applied using the convolution operation, in which each element of the filter is multiplied, element to element, with the corresponding elements of the input matrix having the same relative position (i.e., the filter is applied as a sliding window). All the results of the these multiplications are then summed, providing a new element in an output matrix. An example of this can be seen in Figure 2.1, where a filter (in blue) is applied on a feature matrix (in white) resulting in a new feature matrix. Furthermore, the intermediate steps of the application of the convolutional filter (in light green) on the original feature matrix can be observed.

More formally, the convolution of two real functions, $\mathrm{g}(x)$ and $\mathrm{h}(x)$, can be defined as follows:

$$\int_{-\infty}^{+\infty} \mathrm{g}(t) \cdot \mathrm{h}(x+t)dt \tag{2.19}$$

Depending on the size of the output matrix (i.e., a function of the domain of the input), a different stride size may be used in the application of the filter. The stride size establishes the index of the next element of the feature vector to be used (e.g., with a stride size equal to 2, the filter will be applied in the element $x_{1,3}$ if it was applied in $x_{1,1}$ in the previous iteration, which is exactly what is depicted in Figure 2.1, specifically in the first two applications of the filter). Equation 2.20 formulates the size of the output matrix of a convolutional filter in function of the edge size e (i.e., either height or width) of the input

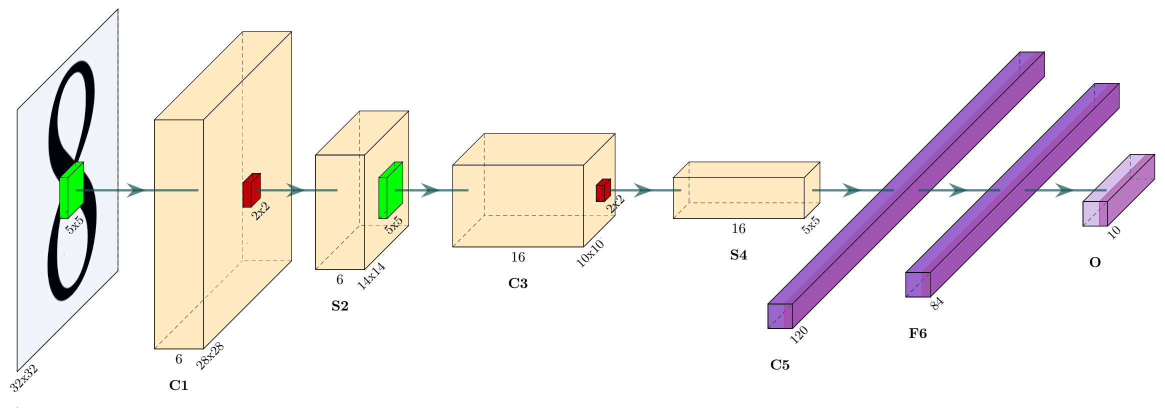

Figure 2.2: The LeNet-5 architecture. The convolution operation is depicted in green and the max-pooling in red, whereas the final fully-connected layers are portrayed in purple.

matrix, the stride size s, and the filter size f.

$$e^{new} = \text{floor}\left(\frac{e - f + s}{s}\right) \tag{2.20}$$

Another type of layers used in convolutional neural networks are pooling layers. In a pooling filter, instead of multiplying element to element the filter with the input matrix, a certain function (e.g., the average or the maximum) is applied to a predefined number of input elements.

Both convolutional and pooling filters are usually much smaller than the original input matrix, and thus down-sampling occurs when the filter is applied, allowing for a more general set of features, in comparison with the original input. After a filter is applied, in order to guarantee that a nonlinear mapping is learned, one of the previously mentioned nonlinear activation functions (e.g., the leaky ReLU) is used to process the results.

Convolutional models frequently also employ fully-connected layers, where filters have the size of an element (i.e., one column per one row), thus behaving similarly to the layers within a multi-layer perceptron (Equation 2.1). Fully-connected layers are useful to learn non-linear mappings from the general features that a convolutional layer outputs.

To illustrate the use of the aforementioned concepts, an analysis of a complete convolutional neural network architecture is given next, specifically the LeNet architecture. Using the neocognitron model first introduced by Fukushima [18] as a base architecture, LeCun et al. [19] developed a pioneering convolutional neural network structure named LeNet to single out hand-written numbers. This architecture uses the previously introduced concepts: convolution, pooling and fully-connected layers. The LeNet-5 architecture shown in Figure 2.2 has 7 layers, namely two convolution layers followed by pooling layers with an average function, and three fully-connected layers at the end of the network. It should be noted that the output of the pooling layer, in this particular architecture, is processed with a sigmoid function. The pooling layers have a size of two rows per two columns and the filter is applied with a stride size of two. The final fully-connected layer uses a radial basis function per class to determine the squared difference of the input and the weight vector, as shown in Equation 2.21.

$$o_i = \sum_{j=1}^{N} (x_j - w_{ij})^2 \tag{2.21}$$

Due to the very large number of parameters in typical convolutional neural networks, model training will most likely overfit. Besides the previously mentioned dropout method, a technique called data augmentation can be used to mitigate overfitting, by creating a bigger dataset for training. New instances are created by applying a set of operations on the original data. For instance, when applied to images, data augmentation can be based on rotating, flipping, cropping, scaling, and/or changing the color temperature of the original images, this way creating a bigger training set with variations on the original instances.

2.2 Advanced Neural Models for Image Classification

Deep neural networks, the learning algorithms that serve as the base for the classifiers here developed, have a a wide range of applications in the computer vision domain, namely image classification. This section provides an in-depth description of two convolutional neural network architectures, namely the DenseNet and EfficientNet architectures.

2.2.1 DenseNet Architecture

Section 2.1.3 introduced the basic building blocks of convolutional neural networks, together with one of the first concrete architectures of this type, i.e., the LeNet-5 [19]. The aforementioned architecture is, however, very shallow, having only seven layers and about 60 thousand learnable parameters. In order to increase the performance of models following this type of architecture, various methods can be used. Moreover, an increment in the number of trainable parameters of convolutional neural networks can further bolster results. An example of this idea is the AlexNet architecture [20], which increases the depth of the LeNet-5 network to eight layers (excluding pooling layers), considerably raising the number of trainable parameters to 60 million. The previously mentioned architecture, which achieved good results in the *ILSVRC-2012* [8] competition, illustrated the classification power of deeper networks with a larger number of trainable parameters.

Another significant advance in the classification potential of convolutional neural networks was put forward with the ResNet architecture [22]. The novelty of this architecture resides in the addition of the possibility of bypassing certain layers using shortcut connections. The aforementioned connections operate by adding the input x of the network to the output $\mathrm{f}(\mathrm{x})$ of a posterior convolution layer (Figure 2.3), i.e., $\mathrm{x}_l = \mathrm{f}_l(\mathrm{x}_{l-1}) + \mathrm{x}_{l-1}$, with l indexing the network's layers. This addition operation allows for a varying degree in the non-linearity of the learning process, and enables the preservation of preceding information. Depending on the complexity of the input that is to be classified, more or less convolutional layers are skipped. These shortcut connections allow, in this way, substantially deeper architectures to con-

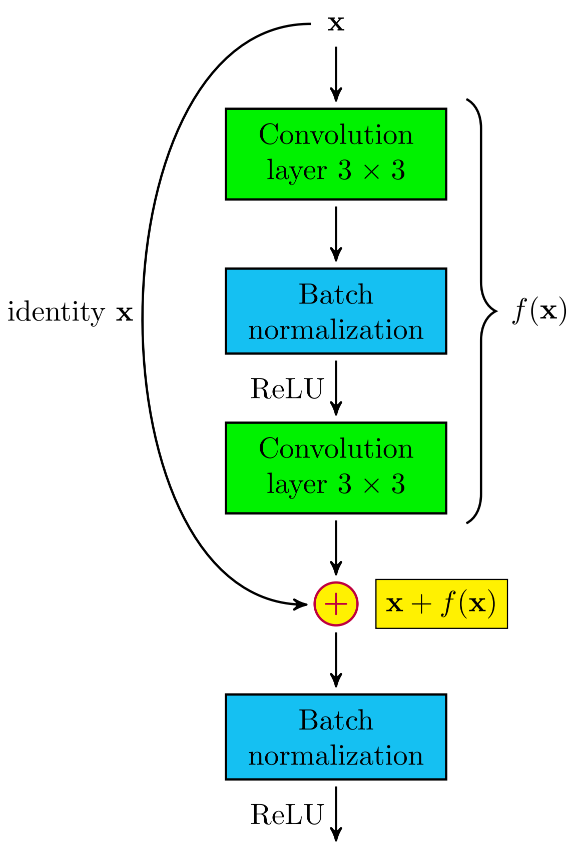

Figure 2.3: Diagram of a shortcut connection in the ResNet architecture. The + operation, represented by the yellow circle, illustrates matrix addition. Batch normalization is used after each convolution filter, but before the non-linear activation function ReLU is applied. This optimization improves both the architecture's convergence rate and the generalization ability [21].

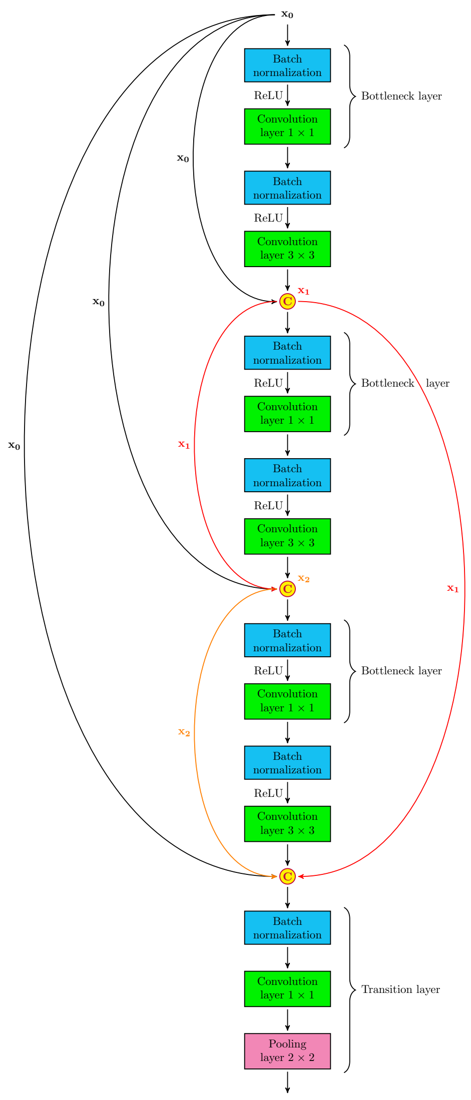

Figure 2.4: Diagram of a dense block from the DenseNet architecture, with three layers followed by the transition layer. The $\mathbf{C}$ operation, depicted in yellow, represents the concatenation of matrices.

verge quicker. This makes the use of deeper architectures a viable option to improve the classification over previous architectures. However, several authors noted that a vastly deeper convolutional neural network architecture is not always beneficial. For instance, He et al. [22] compared a ResNet architecture with 1202 layers with a similar architecture with 110 layers, concluding that the deeper ResNet architecture did not achieve better results.

The DenseNet architecture was introduced by Huang et al. [4] to ameliorate the flow of information in the training of convolutional neural networks. In the ResNet architecture, the only information that was propagated forward, using the shortcut connections, was the input, that was then summed to the resulting feature matrix. However, in the DenseNet architecture, each layer propagates its feature matrix to all subsequent layers (compare Figure 2.4 with Figure 2.3). This way, and 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

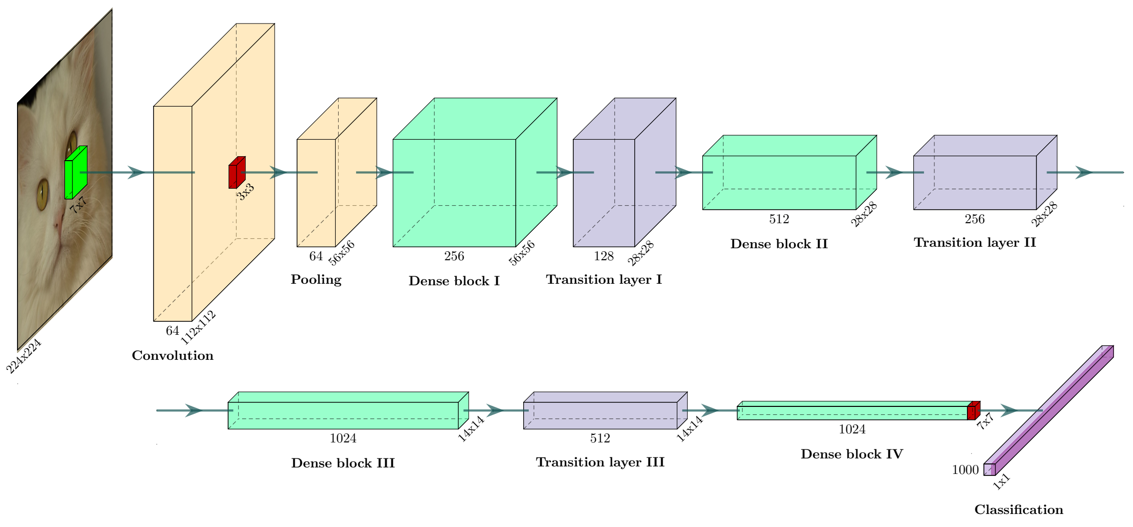

Figure 2.5: Overview of a DenseNet architecture, specifically the DenseNet-121. The convolutional and pooling operations are depicted, in this diagram, respectively in green and red. Regarding the building blocks of this architecture, the dense block is depicted in light green, whereas the transition is represented in light purple. The composition of the dense blocks and transition layers is that of the representation shown in Figure 2.4. The final layer, composed of a fully-connected component and a softmax activation function, is depicted in purple.

own feature matrix, i.e., $\mathbf{x}_l = f_l([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{l-1}])$. Such concatenated 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 exist L, with L being the number of total layers, connections between 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. In order to be able to curb the rise in the number of each layer's feature matrices, the authors focused on the growth rate hyperparameter *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. Huang et al. showed that a DenseNet architecture with a growth rate of 12 was sufficient to surpass, in terms of accuracy, the best outcome of a ResNet architecture in the Street View House Numbers dataset [23]. Moreover, in this performance comparison, the authors found that the number of parameters needed by both architectures to achieve similar error rates in the aforementioned dataset is quite disparate. DenseNet used only half of the 1.7 million parameters employed by the best ResNet model. 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, as shown in Figure 2.4. The dense blocks 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 aforesaid 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 so-called 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 DenseNet-121 (i.e., with 121 as the architecture's depth), shown in Figure 2.5. 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 2.5. 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 dataset that was used to assess the performance) 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 can observed in Figure 2.5, 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.

2.2.2 EfficientNet Architecture

More recently, Tan and Le [3] introduced the EfficientNet architecture, based on the idea that instead of adjusting only the depth of a convolutional neural network, as in the previously mentioned architectures, scaling should be made over three different dimensions: depth, width, and resolution. As seen before, deeper architectures, in general, 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, and afterwards 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 scaling each of the individual dimensions are considerably limited, the authors 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

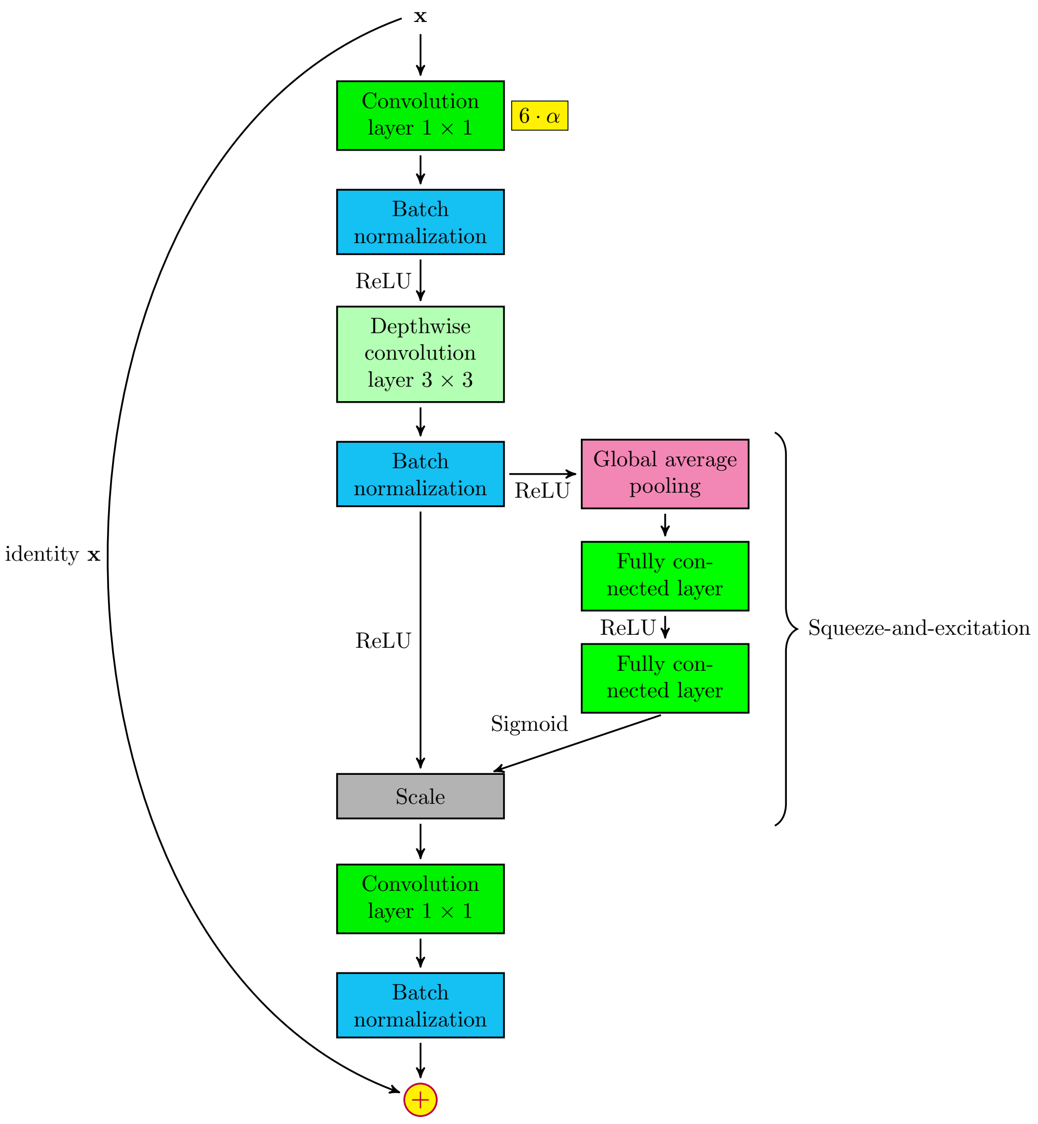

Figure 2.6: Diagram of a MBConv6 3×3 block. In a yellow box, next to the first 1×1 convolution layer, it can be observed how α, i.e., the tensor's channels, are scaled with an expansion ratio of 6. The **+** operation represents matrix addition.

compound scaling concept, which adjusts all 3 network dimensions (i.e., d, w and r) considering their interplay is thus formally defined as:

$$
\begin{aligned}
& d = \alpha^{\phi}, w = \beta^{\phi}, r = \gamma^{\phi} \\
\text{subject to } \; & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\
& \alpha \geq 1, \beta \geq 1, \gamma \geq 1
\end{aligned}
\tag{2.22}
$$

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}. Moreover, the β and γ parameters grow quadratically, while α grows linearly, because these two represent two-dimensional units but, on the contrary, depth is one-dimensional.

The aforementioned concepts were tested by Tan and Le using a baseline network called EfficientNet-B0. The authors designed this network with basis on a method used by Tan et al. [24], where a search with multiple objectives (e.g., latency and accuracy) was done in order to maximize an accuracy metric for a certain 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:

$$\max_{x} \ \mathrm{Accuracy}(x) \cdot \left[\frac{\mathrm{Flops}(x)}{\mathrm{T}}\right]^{w} \tag{2.23}$$

In the previous expression, x represents a network model, $\mathrm{Accuracy}(x)$ is a function which maps a model to an accuracy score, $\mathrm{Flops}(x)$ 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 and that was set by the authors at -0.07. Using Equation 2.23 and the search method introduced by Tan et al. [24], the authors reached the baseline architecture named EfficientNet-B0 from which subsequent versions and improvements build upon. This architecture has a main block named the mobile inverted bottleneck (MBConv) which is optimized with the squeeze-and-excitation technique [25]. As can be seen in Figure 2.6, 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 and, because new features must be created from the input channels, a pointwise convolution (i.e., a 1×1 convolution) is then applied to the results. This two-step procedure differs from a standard convolutional layer, in which both filtering and feature creation are achieved concomitantly. In the case of its application in the EfficientNet-B0 architecture, this approach is able to attain a computational cost about 9 times smaller than when implemented using standard convolution layers [26], as can be deduced by Equation 2.24 where the quotient between the computational cost of the depthwise convolution and that of the standard convolution layer is calculated, with D_k denoting the kernel size, D_f the feature matrix size, M the number of channels in the input, and N the number of output channels.

$$\frac{D_k^2 \cdot M \cdot D_f^2 + M \cdot N \cdot D_f^2}{D_k^2 \cdot M \cdot N \cdot D_f^2} = \frac{1}{N} + \frac{1}{D_k^2} \tag{2.24}$$

The first layer of the MBConv block, i.e., the 1×1 convolution layer that can be seen in Figure 2.6, 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. [27] as an improvement over previous architectures (i.e., MobileNetV1). In exploring the intuition that channels in convolutional neural networks might be embeddable into a subspace with low dimensionality, the authors embedded in the last convolution layer of the MBConv block multiple channels into a vastly smaller number of feature matrices. However, because the application of non-linear activation functions (e.g., ReLU) in a channel dissipates the channel's data, an amplification in the number of channels prior to the reduction of dimensionality is operated, so that a channel's information can possibly be retained in at least one of the remaining channels. In the case of the MBConv6 block (Figure 2.6), the channel's scaling factor, also called expansion ratio, has a value of

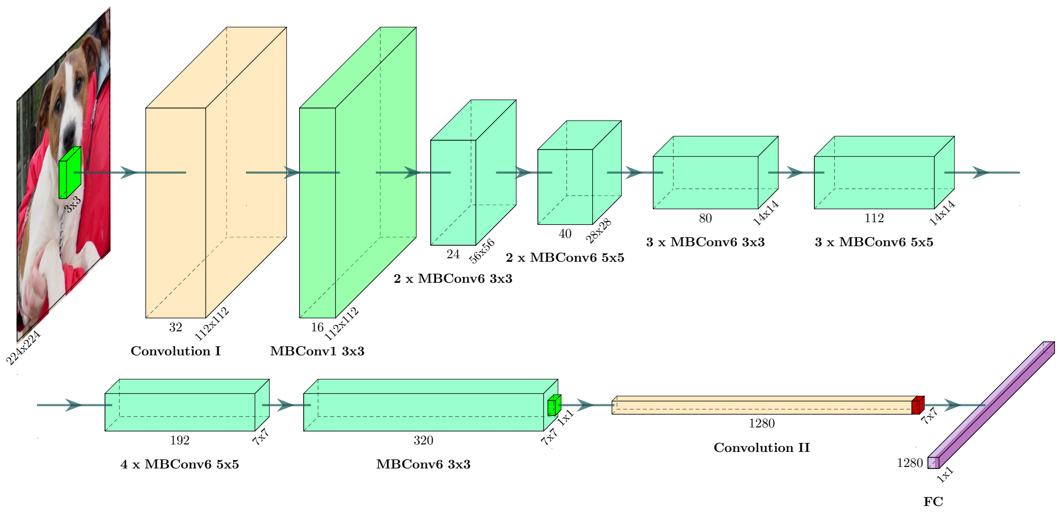

Figure 2.7: Diagram of the baseline architecture, named EfficientNet-B0. As can observed, this network is composed of various MBConv blocks in conjunction with convolution, pooling, and fully-connected layers. As in the previous figures, the convolutional filters are depicted in green, whilst the pooling operations, in this case, average pooling, are represented in red. On the other hand, the fully-connected layers are depicted in purple. Some layers with the same characteristics are, in this architecture, placed consecutively, with the number of these layers being depicted in the diagram by the multiplying constant placed before the layer's name.

6 (specifying in this way the type of MBConv).

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 channel descriptors are used in the consequent excitation phase to learn interdependencies between channels. Such dependencies are exposed by the use of a non-linear function, in this case a ReLU, that is preceded by a fully-connected layer that diminishes the number of channels in order to curb the resulting model's complexity, and succeeded by another fully-connected layer that reestablishes the original number of channels. This 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 is mainly composed of a series of connected MBConv blocks, as seen in Figure 2.7. These blocks are placed sequentially in the network, although with minor differences between them, particularly in the size of the filter that is applied in the depthwise convolution layer, and in the expansion ratio that is to be applied to the block's input channels. The choice of both the size of the layer and the expansion ratio, also called transforming factor, is limited by the search space that was chosen by Tan and Le, i.e., 3×3 and 5×5 for size of the depthwise filter, and a value for the transforming factor that is compatible with the number of output channels of the corresponding layer on the MobileNetV2 architecture [27] adjusted by a factor of 0.75, 1 or 1.25. Moreover, the number of consequent layers of the same type (e.g., MBConv6 5×5) at a certain stage of the architecture is also obtained by maintaining, summing, or subtracting one from the number of repeated layers in the correspondent stage in the MobileNetV2 architecture. Finally, the type of block had also a search domain that, in this case, consisted of a standard convolution filter, a depthwise convolution, and the MBConv block. The optimal architecture, resultant of the aforesaid search, is composed of various types of layers: two convolutional layers, MBConv blocks, and a fully-connected layer.

Building on the EfficientNet-B0 architecture, Tan and Le created 7 better performing architectures (e.g., EfficientNet-B1, EfficientNet-B2, etc.) by scaling the baseline architecture using 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 of the depth, resolution, and width constants; secondly, using these newfound values for the three dimensions, different values for the ϕ parameter were tested.

In order to evaluate the performance of the scaled EfficientNet architectures, the authors used ImageNet as the training and testing data. During these experiments, the authors chose to use a state-of-the-art activation function, namely the Swish activation function. The aforementioned function is defined in Equation 2.25, with σ as the sigmoid function, and β as a constant or as a parameter that can be trained. Ramachandran et al. [28] introduced this activation function, which was the result of an automatic search with the intention of finding new activation functions, that, when compared with the ReLU activation function, achieved improved results in the classification of the ImageNet dataset. Specifically, in a MobileNet architecture originally with a ReLU activation function, Swish improved the model's accuracy by 2.2%.

$$f(x) = x \cdot \sigma(\beta \cdot x) \tag{2.25}$$

Comparing the results of these architectures against various other state-of-the-art convolutional neural network architectures on the ImageNet dataset [8], Tan and Le found that, when grouping architectures with similar accuracy scores, the corresponding EfficientNet architecture (e.g., EfficientNet-B2 is contained in the third worse group of architectures when taking into account the accuracy of each network) always had both substantially less parameters (in average 5.7 times less) and considerably fewer number of FLOPS (in average 11 times less) than the remaining competing architectures, while maintaining, if not marginally bettering, the classification accuracy.

2.3 Processing Archaeological Figures and Classifying Pottery

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, four existing methods that perform automatic pottery typology classification are described.

2.3.1 Classifying and Visualizing Roman Pottery using Computer-scanned Typologies

Christmas and Pitts [29] presented two methods for the categorization of various types of pottery, one manual and one automatic. The automatic algorithm is based on k-means clustering. First introduced by Lloyd [30], 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) of points, 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 2.26, which in the k-means algorithm is to be minimized, formalizes this notion:

$$\sum_{k=0}^{K} \sum_{x \in C_k} \|x - \mu_k\|^2 \tag{2.26}$$

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 stabilize, 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. These measures were calculated by using the multiple standardized drawings of the *Camulodunum* series [31] that were automatically extracted and then segmented. The authors chose to use various measures of the artifacts, such as the vertical centroid ($centroid_v$), 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, as given by Equation 2.27:

$$\text{circularity} = \frac{4\pi \cdot \text{area}}{\text{perimeter}^2} \tag{2.27}$$

The result of the application of the k-means algorithm to these data points is depicted in Table 2.1, and the method achieved an accuracy of 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). As can been see in Table 2.1, artifacts that belong to the *platters* or the *bowls* classes are better classified, taking into account the higher total number of instances from these categories in the dataset, compared to the remaining classes. The authors conclude that, for commonplace pottery categories (e.g., *bowls*), the automatic classification technique can transform the categorization of artifacts into a more objective process.

Concerning manual classification, the authors used different techniques with the aim of extracting new insights from the various measures computed for each of the pottery diagrams. The different techniques compared aspects of the artifacts, including similarities between objects found in different sites, similarities between the shapes of different artifacts, and how do the different artifact *form groups* (e.g.,

True class	Total	Beakers	Bowls	Butt beakers	Carrot amph	Cups	Defrutum amph	Girth beakers	Jars	Pedestal beakers	Platters	Flagons	Flasks	Jugs	Lids	Mortaria	top	base	(invalid)
Beakers	22	**77**	9	14															
Bowls	70	14	**80**	3							3								
Butt beakers	15	27	7	**53**					7				7						
Carrot amph	1													100					
Cups	9		100																
Defrutum amph	6						**83**							17					
Girth beakers	7	57		14				**14**		14									
Jars	12		8	17					**25**	17			25	8					
Pedestal beakers	12	8	42							**50**									
Platters	45		4								**96**								
Flagons	15		7	7								**73**		13					
Flasks	10	10	10	20								20	**40**						
Jugs	11			9										**91**					
Lids	19														**89**				11
Mortaria	6		17													**83**			
top	4		75									25							
base	7		14												29				57
(invalid)	13														8				**92**

Table 2.1: Outcome of the k-means algorithm in the form of a confusion matrix [29]. Column *Total* represents the total number of instances of a certain pottery type, and column *True class* the set of pottery categories. The remaining columns correspond to the found clusters that were manually assigned to one of the classes. Each of the numbers in the aforementioned columns expresses the percentage of a certain class in the cluster. Depicted in bold are the correct classifications for each one of the clusters.

bowls, *platters*, etc.) cluster. The first method consists in plotting a histogram for each one of the archaeological sites, from which the pottery artifacts were obtained. These histograms, as depicted in Figure 2.8, portray the number of vessels that have a certain $centroid_v/height$ ratio in a particular site. To be able to compare the distributions of the histograms from the various sites, the authors used the Bhattacharyya distance [32], as shown in Equation 2.28. This measure of dissimilarity is used to compare, in a pairwise manner, the previously mentioned histograms, in order to conclude which were the most similar sites. Moreover, since this dissimilarity measure is not constant across all the pottery vessels from a certain site, in order to further analyze the similarity between sites, Bhattacharyya distances for each pair of locations are computed considering each of the sites' diagrams individually. In this way, the analysis of the dispersion of the calculated distances between two sites can further indicate their similarity (e.g., small dispersion of the Bhattacharyya distance between the vessels of a certain pair of sites combined with a low Bhattacharyya distance between locations suggests similarity).

$$B_{n,m} = -\ln\left(\sum_{x \in X} \sqrt{\mathrm{h}_n(x) \cdot \mathrm{h}_m(x)}\right) \tag{2.28}$$

In the previous equation, h_n and h_m represent the height of the histogram bar respectively for histogram n and m being compared. The same process was then used by the authors, but this time using two-

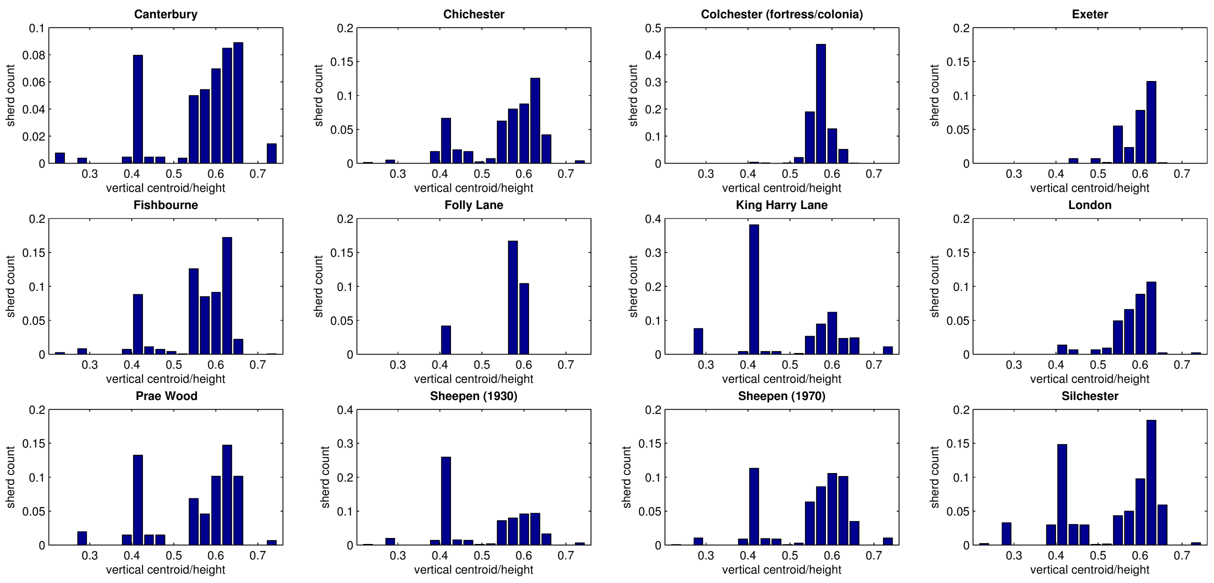

Figure 2.8: Histograms for some of the sites analyzed using the $centroid_v/height$ measure [29].

dimensional histograms, i.e., with $width/height$ and $centroid_v/height$ as measures. This technique, which returns the most similar site for each of the locations, allowed the authors to conclude that the differences between the objects found in various sites were, besides in the weight of each category in the overall collection of objects, in the physical measures (e.g., height) of the pottery. Such differences can be observed by comparing the sites which had a cultural context in which *butt-beakers* (type of pottery vessel used for drinking) were widely used, and sites with military and colonial environments where these types of objects were not common. In the first type of sites (e.g., Canterbury and Silchester), a peak in the 0.4 $centroid_v/height$ area can clearly be discerned. However, in the sites with a different environment (e.g., Exeter and Colchester) this peak is absent, showing, in this way, a pronounced difference between these locations. Due to this, Christmas and Pitts concluded that the $centroid_v/height$ measure can be advantageous in subsequent analyzes of Roman pottery.

2.3.2 Arch-I-Scan

Following a different approach, Tyukin et al. [33] developed a software for smartphone devices, named Arch-I-Scan, 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 [34], Histogram of Oriented Gradients (HOG) 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. Due to time constraints, the authors chose to model 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

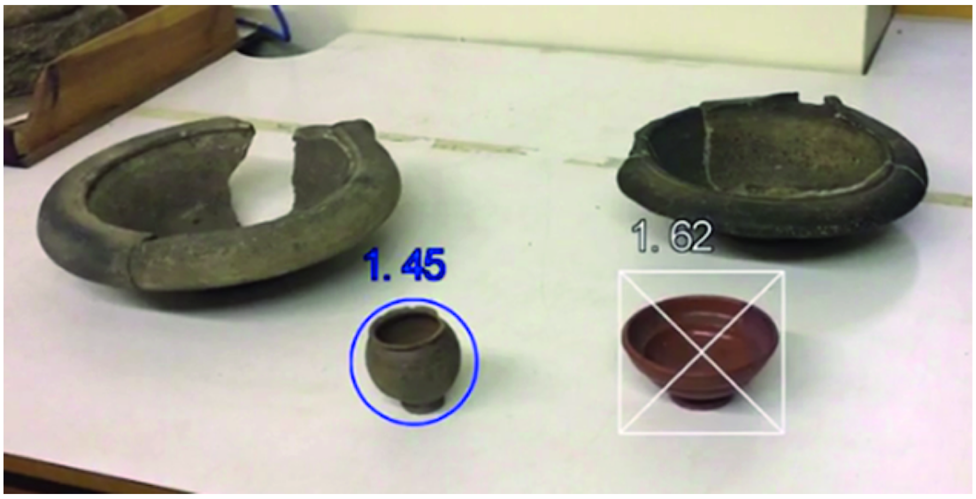

Figure 2.9: Classification scenario using the Arch-I-Scan software, where four artifacts, each of a distinct class, are scanned and fed to the previously mentioned 10 classifiers. As it can be observed, two of the classifiers, represented by the blue circle and white square, correctly labeled their objects while the remaining two pottery artifacts, which do not have a detector, are not identified. Consequently, there are no false positives in this detection scenario.

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 2.29.

$$\mathrm{D}(x) = \sum_{i=1}^{2400} (x_i \cdot w_i) + b \tag{2.29}$$

The authors have specifically used Support-Vector Machines [35], a technique for learning a linear classifier which maximizes the distance of a hyperplane separating two classes of data points, to learn the discriminant's parameters. 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. Tyukin et al. [33] report no considerable number of false positives (i.e., objects erroneously classified as part of the positive class by the detectors of irrelevant classes) in their experiments, confirming the robustness of this pottery vessel detection method.

2.3.3 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 [36] 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 [37], 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. [38].

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 [39], (ii) linking of edges which are not connected leveraging a morphological dilation operation [40], (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 [40] is used to remove erroneous edges, and, finally, (v) a Sobel edge detector [41] 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 aforesaid 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.

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

More recently, in the context of the multidisciplinary ArchAIDE European project[1] [42], with the main objective of providing automatic classification tools to complement both archaeological fieldwork and post-excavation analyses, a software platform was developed to automatically categorize extracted pottery artifacts (e.g., *terra sigillata*) into typologies. The recognition and classification component of this platform is based on the deep learning approach developed by Itkin [43], specifically for this project. More concretely, Itkin focused on the categorization of pottery sherds leveraging color photographs. In this technique, 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 [22], which was previously trained on the ImageNet dataset [8]. 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 [44]. 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:

$$x^2 + y^2 = p_x^2 \wedge z = p_y \tag{2.30}$$

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 [45] 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.

2.4 Overview

The fundamental concepts of artificial neural networks in conjunction with a set of state-of-art techniques combining the convolutional neural network architectures and the methods for pottery photograph/diagram classification leveraging disparate learning algorithms was presented in this section.

In what concerns the convolutional neural network architectures, despite the EfficientNet architecture clearly presenting better results in the ImageNet classification task when compared with the DenseNet architecture, both architectures were employed due to their contrasting structure (e.g., a 7×7 first convolution kernel in the DenseNet architecture vs. a 3×3 in the EfficientNet architecture) and consequent potential divergent response to this work's examined datasets.

The automatic classification systems produced in this research combine various aspects present in the previously analyzed pottery classification methods, namely the use of the k-nearest neighbors algorithm, taking advantage of models trained on general datasets (i.e., ImageNet), or leveraging the representations learned by the convolutional neural network model. In the next chapter, a detailed analysis of the various techniques composing the pottery classification systems is put forth.

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

Chapter 3

The Deep Convolutional Neural Model for Archaeological Pottery Classification

The detailed presentation of the structure of the CNN-based system for pottery image classification is accomplished in this chapter. The proposed method is composed of three distinct components: the convolutional neural network architecture, the model training procedure, and the meta-learning classification. Figure 3.1 depicts an outline of the various components of the developed classification system and their interplay in the context of the black-and-white pottery diagrams.

The first component, the convolutional neural network architecture, serves as the basis for the rest of the proposed approaches. Section 3.1 presents the two studied structures. These convolutional neural networks are trained over the datasets that are the aim of this dissertation and generate, in this way, automatic classification models.

The procedure for model training, composed of self-training and data augmentation, is the second component of the classification system. First, self-training, by leveraging unlabeled instances from the domain of the classification task, previously trained models are used to label the available unlabeled images. These newly labeled images are subsequently added to the existing labeled data instances and then used to train a new, more robust model featuring various types of noise. The technique is presented in Section 3.2.1. Secondly, in order to combat potential overfitting, data augmentation techniques are deployed during the training of the convolutional networks, with the considered techniques being detailed in Section 3.2.2.

Finally, the third component features the meta-learning approach to classification. The procedure leverages the models generated by the training of the convolutional neural networks using the above-mentioned methods. In this procedure, the classification layer of the generated models is removed in order to extract the learned representations for each of the data instances. These representations are then used to train a supervised classifier (e.g., logistic regression) that performs the final classification.

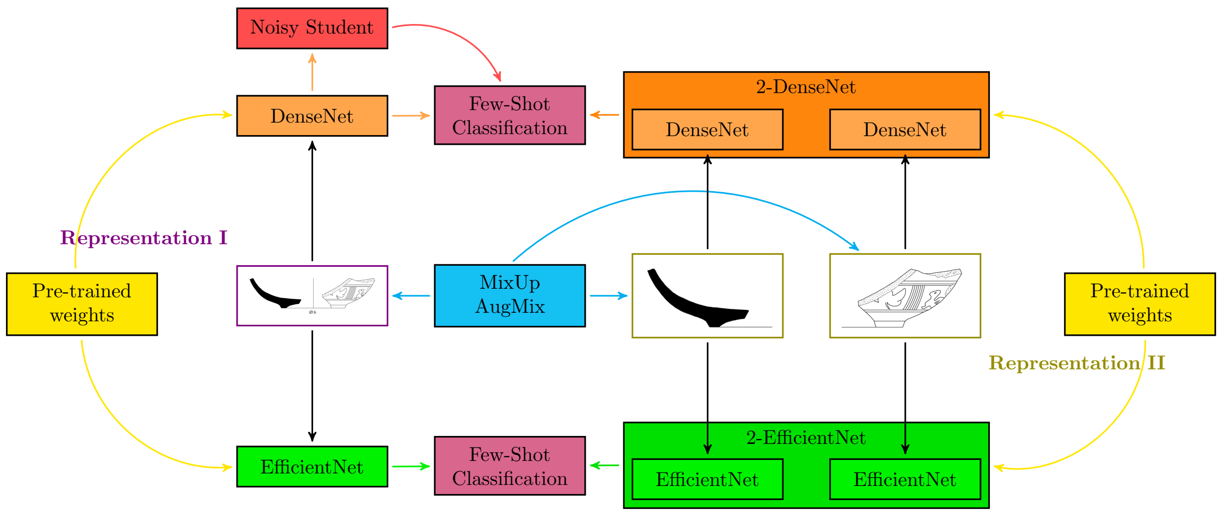

Figure 3.1: Overview of the proposed approach for the black-and-white diagram dataset.

The details of this method are presented in Section 3.3.

The subsequent sections of the chapter detail the various techniques that compose the aforementioned method for pottery image classification. In conclusion, a summary of the chapter is presented in Section 3.4.

3.1 Convolutional Neural Network Architectures

In this section, the two alternative input structures for the convolutional neural network architectures are presented, namely a standard single-input image classification architecture and a two-input architecture where each view is propagated through separate convolutional neural networks.

3.1.1 Single Input Convolutional Neural Network Architecture

The proposed single-input structure, the most frequent network disposition in image classification tasks, is the base architecture of this work. In this setting, the images are simply propagated through the model up to the final classification fully-connected layer. For this approach, the considered architectures, analyzed in detail in Section 2.1.3, were the DenseNet-201 and the EfficientNet-B3.

3.1.2 Two-Input Convolutional Neural Network Architecture

As previously mentioned in Chapter 1, the diagrams in question are composed of two views. Consequently, to potentially take advantage of the knowledge of the data instances' structured format, a two-input-based architecture was developed. 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. Once more, the DenseNet-201 and EfficientNet-B3 architectures were considered, but now, taking into account the two-input format, in a pairwise disposition, i.e., two DenseNet-201 or EfficientNet-B3 models.

3.2 Model Training

During model training leveraging the architectures mentioned in Section 3.1, data scarcity is made apparent as the main obstacle to model performance due to the proneness of these architectures to overfit to small datasets. Consequently, the overfitting problem is approached in two ways: leveraging unlabeled instances through a self-training procedure and regularization through the use of data augmentation techniques.

3.2.1 Self-Training

One of the principal pitfalls of the application of deep convolutional neural networks to a certain domain is the scarcity of available labeled data instances. A commonly taken approach to mitigate this issue is the use of unlabeled data instances, known as semi-supervised learning, through specialized techniques since such types of instances are significantly more accessible. Self-training is one of these methods 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. The process can be repeated, with each iteration being concluded with the student model of the finishing iteration becoming the teacher model of the new iteration.

Xie et al., from the aforesaid self-training framework, introduced the Noisy Student technique. The key contribution of this technique is the incorporation of noise into the student model, allowing this model to be as powerful as the teacher model. The added noise is composed of Stochastic Depth [46], Dropout [17], and RandAugment [47]. Figure 3.2 presents a diagram of training using the self-training procedure leveraging a noisy student.

Following the intuition that a student model with an additional amount of noise can further improve its classification performance, a dropout layer [17], 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). Moreover, data augmentation noise, which in the original noisy student method is enforced by the RandAugment technique, is performed by the data augmentation techniques explored in this dissertation.

After obtaining a considerable quantity of unlabeled data instances, 2963 to be more precise, the above-mentioned procedure was applied to the dataset, slightly modified to include a prediction confidence threshold. This threshold was used to filter the instances labeled by the student model with a

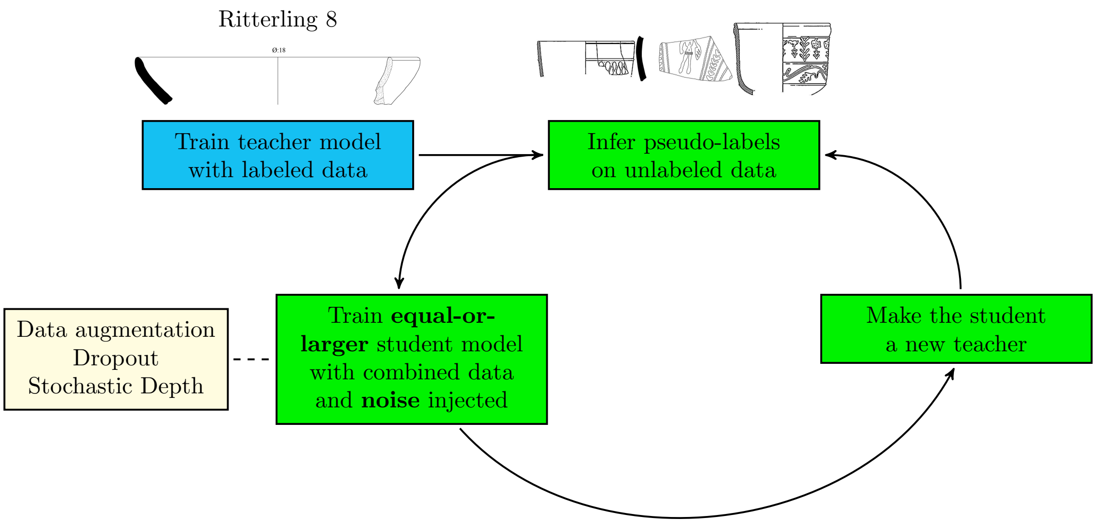
Ritterling 8
Ø:18
Train teacher model with labeled data
Infer pseudo-labels on unlabeled data
Data augmentation Dropout Stochastic Depth
Train **equal-or-larger** student model with combined data and **noise** injected
Make the student a new teacher

Figure 3.2: Diagram, adapted from the article by Xie et al., depicting an outline of the noisy student approach to semi-supervised learning.

prediction probability smaller than 95% (corresponding to the average of each fold's probabilities), which summed to 94 data instances for one of the developed DenseNet-201 models. The filtering step mitigates the potential problems related to images from exogenous classes since the typologies present in the unlabeled data may not intersect with the considered class set.

3.2.2 Data Augmentation

This section presents the data augmentation techniques employed in the experimented convolutional neural network models. Such regularization techniques, which aim to improve the generalization capability of deep learning models, are an essential tool to mitigate the overfitting inclination of deep convolutional neural networks.

3.2.2.1 MixUp

One such method, introduced by Zhang et al., is the denominated MixUp [5] regularization technique. This technique uses a data augmentation method based on linear interpolations of the original training data to generate new artificial training instances. These new instances are generated from pairs of training instances that are chosen randomly from the initial dataset, and the linear interpolation is performed as shown in Equation 3.1:

$$\begin{aligned}\tilde{x} &= \lambda \cdot x_i + (1-\lambda) \cdot x_j \\ \tilde{y} &= \lambda \cdot y_i + (1-\lambda) \cdot y_j\end{aligned} \tag{3.1}$$

In the previous expression, x_i represents the feature vector for instance i, y_i the class vector for instance i, and λ the weight given to each instance in the linear interpolation. The λ parameter is drawn randomly from a Beta distribution, which has two shape parameters α and β, with $\beta = \alpha$. The authors found that, for the α hyperparameter, the optimal value for the best performance in the ImageNet classification task was 0.2.

In the application of the MixUp technique to the pottery classification task, the optimal hyperparameters found by Zhang et al. are maintained, however, a standard set of data augmentation operations (e.g., rotate, translate) are applied over the newly generated instances in order to further increase regularization potential. It is worthwhile noting that only the MixUp generated examples are used in training.

3.2.2.2 AugMix

Moreover, an additional data augmentation technique called AugMix [6] builds upon the MixUp regularization to combat the performance issues emanating from disparities in the distribution of the data instances used for training and testing. This technique, introduced by Hendrycks et al., generates synthetic data instances from the original data by combining augmentation operations (e.g., rotate, posterize, etc.) and weighting the resulting operation chains. Firstly, the weights that will be weighting the k operation chains are sampled from a Dirichlet distribution with concentration parameters $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_k)$. Then, the k operation chains are assembled by sampling three different augmentation operations and composing them into sequences with a length of one to three operations (e.g., a chain of operations of length two can be comprised of a rotation followed by a translation). The chains of augmentation operations are then weighted, producing in this way, a new intermediate data instance that combines the operation chains. Furthermore, using Equation 3.1, i.e., the expression which the MixUp technique uses to generate new data instances, Hendrycks et al. interpolate the original data instance with the newly generated image creating, this way, the new synthetic image. As to promote the original data instance and its augmented variants to be classified in the same manner (i.e., classified as the same category) by the learned model, the authors minimize the Jensen-Shannon divergence consistency of the probability of the original data instance, as well as its augmented variants, being classified by the model as a certain class. Consequently, the loss function can be defined as:

$$L(\hat{\mathrm{p}}(y|x_0), y) + \gamma \cdot \text{Jensen-Shannon}(\hat{\mathrm{p}}(y|x_0), \hat{\mathrm{p}}(y|x_1), \hat{\mathrm{p}}(y|x_2)) \tag{3.2}$$

In the previous equation, L represents the model's loss, γ (set to 12 for the testing experiments performed by Hendrycks et al.) the weight attributed to the Jensen-Shannon divergence consistency in the loss function, $\hat{\mathrm{p}}(y|x_0)$ the posterior distribution of the original image x_0, $\hat{\mathrm{p}}(y|x_1)$ the posterior distribution of an augmented variation x_1, and $\hat{\mathrm{p}}(y|x_2)$ the posterior distribution of a second augmented variation x_2. The Jensen-Shannon portion of the loss function expressed in Equation 3.2, is computed using the following expression:

$$\begin{aligned} \mathrm{JSD} &= \frac{1}{3} \cdot (\mathrm{D_{KL}}(\hat{\mathrm{p}}(y|x_0) \| \mathrm{M}) + \mathrm{D_{KL}}(\hat{\mathrm{p}}(y|x_1) \| \mathrm{M}) + \mathrm{D_{KL}}(\hat{\mathrm{p}}(y|x_2) \| \mathrm{M})), \\ \mathrm{M} &= \frac{1}{3} \cdot (\hat{\mathrm{p}}(y|x_0) + \hat{\mathrm{p}}(y|x_1) + \hat{\mathrm{p}}(y|x_2)) \end{aligned} \tag{3.3}$$

In the previous equation, D_{KL} signifies the Kullback–Leibler divergence measure.

When compared with other data augmentation techniques, for instance, over the CIFAR-10-C [48] dataset, AugMix achieved an average classification error corresponding to a 16.6% decrease over models leveraging standard augmentation techniques (e.g., horizontal flipping and cropping) and an 11.7% decrease over the aforementioned MixUp data augmentation technique.

Regarding the series of available augmentation operations for AugMix, distinct arrays of operations were considered for each of the two datasets. For the black-and-white diagram dataset, the original set of operations, albeit with a stronger intensity, mentioned on the AugMix paper was considered, namely *autocontrast*, *equalize*, *posterize*, *solarize*, *shear*, *rotate*, and *translate*. On the other hand, for the color photograph dataset, augmentation operations that modify the color of the original object were replaced by the *brightness*, *contrast*, and *sharpness* operations. The aforesaid modification was performed due to the different nature of the second dataset's classes, i.e., classes are distinguished mainly by the decorations of the artifacts, and augmentation operations that modify the artifact's color can inadvertently cause the object to shift class.

3.3 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. The two datasets which are the focus of this work, as can be assessed in Section 4.1, are composed of only a small number of examples for each of the analyzed typologies, with the black-and-white diagram dataset as the most glaring example of this, and, therefore, fit in a few-shot type of scenario. Throughout this dissertation, the concept of few-shot learning is used in a broad sense signifying the application of machine learning techniques to contexts where only a small amount of training data instances for each class is available. The definition of few-shot learning encompassing a setting where the set of classes used for training and testing is distinct is broached in Section 5.2.

One of the possible approaches to solving few-shot learning tasks is through meta-learning, i.e., the idea of learning over previously acquired knowledge. In this case, meta-learning consists in the application of automatic learning algorithms (e.g., k-nearest neighbors) over the representations of images that the convolutional neural network model has produced. SimpleShot [10] or the classification methodology introduced by Tian et al. [9] are examples of this, with both techniques achieving state-of-the-art results in few-shot learning benchmark tasks.

The SimpleShot technique, proposed by Wang et al., is based on the intuition that nearest-neighbor classification, when applied over the representations learned by a convolutional neural network model leveraging the standard cross-entropy loss, is capable of state-of-the-art performances in few-shot scenarios. More concretely, the method is a two-step procedure. First, the feature vectors, previously learned by the convolutional neural network, are subjected to two transformations, namely the centering of the feature vectors by subtraction of the mean of the training data instances' features and their subsequent l_2 normalization. Secondly, the classification of the data instances, through their feature vectors, is performed by a nearest-neighbor classifier leveraging the Euclidean distance as the measure for instance distance. Tian et al. [9] introduced a procedure for few-shot classification, which is similar to SimpleShot. However, in this paper, it is proposed to apply a logistic regression classifier over the feature vectors as an alternative to the nearest-neighbor classifier, with Tian et al. reporting competitive performances in a number of few-shot learning tasks.

Although more intricate techniques, such as the Prototypical Networks [49], 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 of (i) training the convolutional neural network model (e.g., the EfficientNet-B3) leveraging the softmax cross-entropy loss; (ii) removing the last layer of the model (i.e., the classification layer with the classes' dimensionality) thus exposing a global average pooling layer corresponding to a 1920 and 1536 dimensional feature vector for DenseNet-201 and EfficientNet-B3, respectively; (iii) obtaining the learned representations of the data instances by predicting both the training and test set, leveraging the learned model; (iv) applying standardization to the images' features (i.e., removing the mean and dividing 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. [10], 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 black-and-white diagram dataset's experiments. For the color photograph dataset, due to its severe class imbalance, only l_2 normalization is performed over the feature vectors, as Wang et al. report decreasing performances when features are mean-centered in the context of long-tailed datasets.

3.4 Summary

In this chapter, the architecture for the classification system proposed in this dissertation was presented. Summarily, the automatic classification model is a convolutional neural network structured in two possible ways: single-input or two-input. During the training of the aforementioned networks, in order to regularize it, alternative data augmentation techniques are employed. Moreover, leveraging the available unlabeled data, a self-training procedure is applied in order to expand the training set and potentially improve model performance. The last component of the system consists in the application of a classifier over the representations learned by the automatic classification models (i.e., the meta-learning approach).

Section 3.1 detailed the main component of the proposed classification system: the convolutional neural network architecture. Two different input structures for these architectures were presented, namely a single-input and two-input architecture.

Section 3.2.1 presented the noisy student procedure, which is responsible for the incorporation of unlabeled data instances in the training of the convolutional neural networks in the context of the pottery classification task.

Taking into consideration the overfitting potential of training convolutional neural networks over the two pottery image datasets, regularization becomes a crucial part of model training. Consequently, in Section 3.2.2, the employed data augmentation techniques and their application to the task in question were discussed.

In conclusion, due to the state-of-the-art results of the meta-learning approach in few-shot learning, the meta-learning approach is adapted and applied to the pottery classification task. Section 3.3 presented the details of this approach and the effected modifications for its successful application to the studied classification task.

Chapter 4

Experimental Evaluation

In this chapter the experimental evaluation that was followed for the developed classification systems is presented. A comprehensive analysis of the two datasets that are the focus of this work as well as the description of the experimental methodology is presented at the beginning of the chapter in Section 4.1. Section 4.2 details and analyzes the results obtained in the black-and-white diagram classification task, while Section 4.3 focuses on the color photograph setting. Finally, in Section 4.4, a synopsis of the chapter is presented along with an outline of the results of the performed experiments.

4.1 Datasets and Experimental Methodology

During the course of this work, two datasets were studied. Considering the black-and-white diagram classification task, a dataset with a mixed composition was assembled: diagrams from a set of pottery objects retrieved in a specific excavation and from articles of the area. Concerning the pottery color photograph classification task, a dataset comprising artifacts from a museum was obtained from an online resource.

4.1.1 The Pottery Black-and-White Diagrams Dataset

The data instances that compose the above-mentioned artifact dataset come 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 [50]. *Terra sigillata* is a kind of pottery highly standardized in its forms, characteristic of the Roman Empire. It is defined as pottery tableware with a distinct glossy reddish surface, sometimes with geometrical, floral or figural decorations [51].

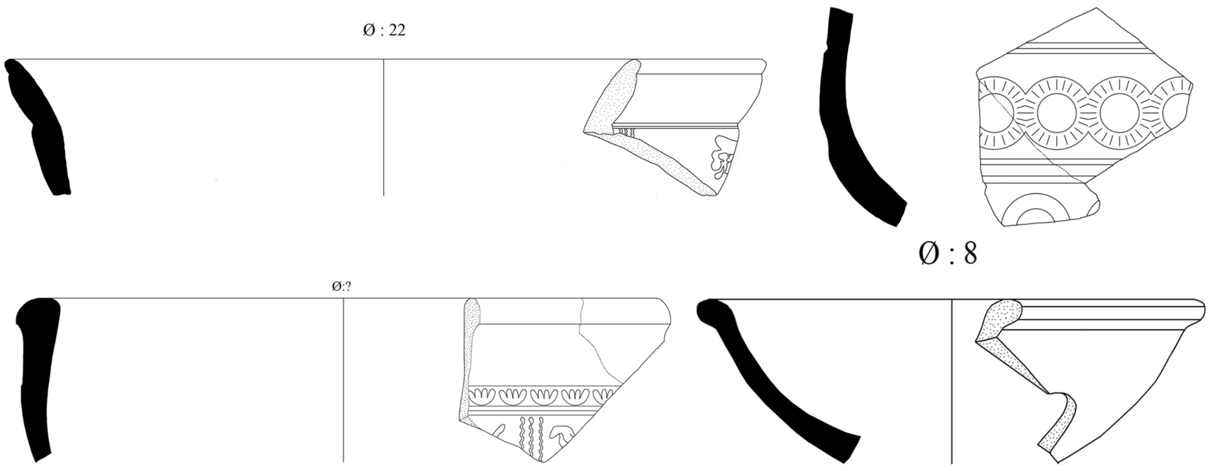

Figure 4.1: Example of four artifact diagrams from the black-and-white diagram dataset.

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 4.1. 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 4.2. To expand this dataset, other sources of pottery diagrams were considered, 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), classes manifesting such characteristics were filtered, 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: [52], [53], [54], [55], [56], and [57], reaching a total of 343 *terra sigillata* fragments with assigned typology, henceforth this will referred as labeled instances.

The number of labeled instances in the application of convolutional neural networks to specific domains can prove to be scarce, as is the case in this work. Consequently, unlabeled data instances can be used as a secondary data source to increase the size of this dataset. In the present research, by searching through various publications depicting artifact pottery diagrams (e.g., the above-mentioned articles) and leveraging automatic object detection techniques, 2963 new unlabeled pottery diagrams were obtained. The unlabeled diagrams were only used in experiments exploring 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 the available computational resources. On the other hand, for the two-input

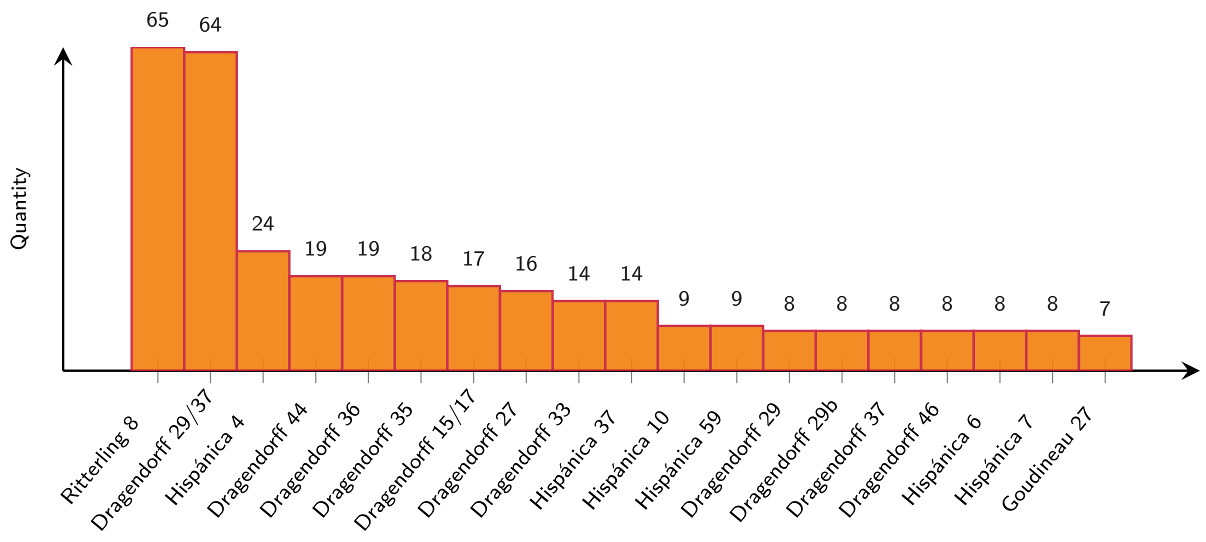

Figure 4.2: Histogram of the 19 available categories from the dissertation's black-and-white diagram dataset after filtering for classes with more than four instances, preceding the addition of supplementary data instances.

architectures, the same process was applied but, since the images are separated in half, the input's selected resolution was 224×224.

4.1.2 The Pottery Color Photographs Dataset

Concerning the dataset that is composed of color photographs depicting archaeological pottery artifacts, an already available set of data instances online in the website of the Florida Museum of Natural History[1] was obtained, organized, cleaned, and used in the second set of experiments. The aforementioned collection of data is comprised of two different collections, namely the Historical Archaeology Type collection and the Lister collection. The first collection, also called the Goggin-Fairbanks type collection, focuses on artifacts extracted from sites in the state of Florida and the Caribbean region, with the *majolica* type having a prominent role. The Lister collection, another archaeological ceramic type collection, is composed of pottery artifacts originating from both mainland Spain and the past Spanish colonial presence spanning Central and South America, with the collection being comprised of 1483 artifacts. These artifacts, originally retrieved by Florence Lister and Robert Lister, served as basis for a number of foundational articles (e.g., [58], [59]) in the classification of ceramic artifacts of the Spanish Colonial era.

The bulk of the aforementioned collection is comprised of pottery of the *majolica* (or *maiolica*) type. More precisely, *majolica* is defined as earthenware (i.e., nonvitreous pottery such as *terracotta*) covered with an opaque lead glaze (i.e., coated with lead) and a shiny white background (by the use of tin oxide) [60]. In the case of this dataset, the *Majolica* pottery is mostly originated from the territories of Central America in which Spain had a colonial presence (e.g., Mexico), with an heterogeneous composition regarding its origin, i.e., some of these artifacts were produced locally while the rest was imported from Spain.

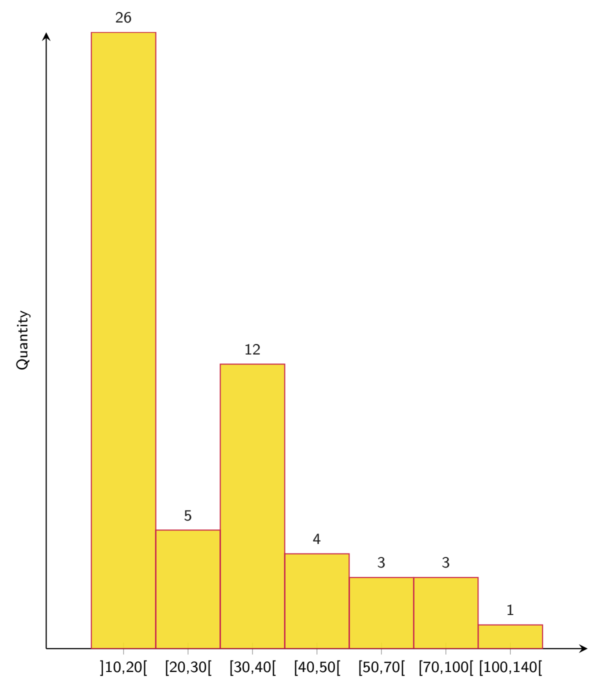

Figure 4.3: Histogram aggregating the number of instances for each of the 54 categories, concerning the color photographs dataset, after filtering for classes with than 10 instances and eliminating duplicates and unusable pictures.

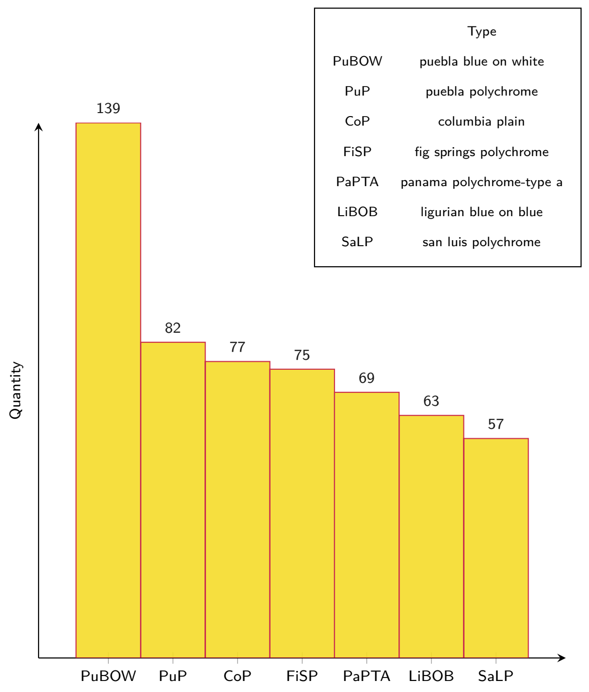

Figure 4.4: The size of the classes that contain more than 50 data instances (i.e., the seven most voluminous archaeological pottery types, corresponding to approximately 35% of the labeled instances) is depicted in the histogram above.

Regarding the structure of the photographs, in all the used examples the artifact is photographed from the front with a checkered ruler also being captured in the photograph in order to transmit the object's scale, as can be observed in the left part of Figure 4.5. In total there exist 2701 labeled instances belonging to 196 different typologies. Additionally, the dataset contains three different views for each of artifact, namely front, side, and back. However, due to unique marks present in the artifacts (i.e., an identifying number drawn in the back of the artifact), the back view of the object is not used. Following a similar procedure to the black-and-white diagram dataset, only categories that presented more than 10 labeled examples were considered. Consequently, the final artifact color photograph dataset is composed of 54 types adding up to 1632 images. As in the previous dataset, there exists a significant class imbalance in the color photograph dataset, as can be observed in the histogram depicted in Figure 4.3, with most classes having between 10 and 20 examples. The weight of the more numerous classes can be observed in Figure 4.4, with the seven largest classes corresponding to about 35% of the dataset and the most populous class *puebla blue on white* reaching approximately thirteen times the size of the least numerous classes.

4.1.3 Experimental Methodology

To evaluate different classification systems in a sound and objective way, performance metrics must be utilized. Three main metrics will be considered in this work, namely precision, recall, and accuracy. In a

[1] https://floridamuseum.ufl.edu/typeceramics/types

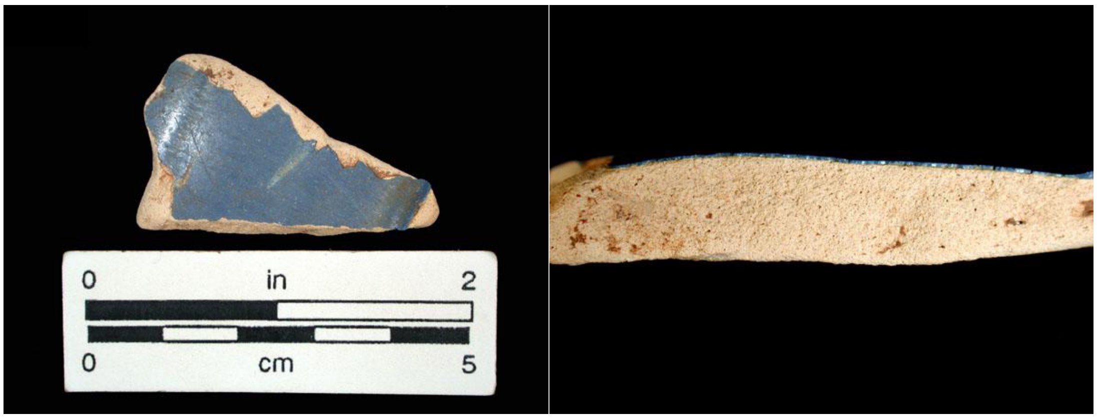

Figure 4.5: Example of the two available views (front and side) for each pottery artifact of the color photograph dataset. In this particular case the data instance is part of the *caparra blue* typology.

binary classification problem, precision represents the portion of correctly classified positive instances, of all the instances that the system classified as positive. On the other hand, the recall metric captures the portion of correctly classified positive instances, of all the positive instances. Finally, accuracy conveys the percentage of correctly classified instances. The formal definitions of precision, recall, and accuracy are respectively presented in Equations 4.1, 4.2, and 4.3.

$$\text{Precision} = \frac{tp}{tp + fp} \tag{4.1}$$

$$\text{Recall} = \frac{tp}{tp + fn} \tag{4.2}$$

$$\text{Accuracy} = \frac{tp + tn}{fp + fn + tp + tn} \tag{4.3}$$

In the previous equations, tp represents the true positives (i.e., instances correctly classified as part of the positive class), tn the true negatives (i.e., instances correctly classified in the negative class), fp the false positives (i.e., instances erroneously classified in the positive class), and fn the false negatives (i.e., instances wrongly classified in the negative class). The aforementioned precision and recall metrics are defined only in binary classification tasks, i.e., instances can only belong to one of two classes. Since this dissertation is focused on a multi-class classification problem (i.e., one diagram instance can belong to one of 19 classes), the previous metrics cannot be used directly in measuring the system's performance. Accuracy can be directly computed as the fraction of the correct classification decisions but, in order for precision and recall to be correctly computed when applied to multi-class data, an average for each of the dataset's classes can for instance be taken (i.e., binary metrics for each of the existing classes are computed considering the instances of that category as the positive class, and the remaining instances are labeled as the negative class). A macro-average score can thus be computed as shown in Equation 4.4.

$$\text{Multi-Class Macro-Average Metric} = \frac{1}{N} \cdot \sum_{i=1}^{N} \text{M}_i, M \in \{\text{Precision}, \text{Recall}\} \tag{4.4}$$

In the previous equation, N represents the number of classes present in the dataset, and M is the binary version of the metric in question. Equation 4.4 is called a macro-average of the binary metrics (i.e., a simple average for each of the dataset's classes is computed), but micro-averaging can also be used as an alternative path to calculate a multi-class version of the previously introduced binary metrics. This can be computed by averaging the total amount of instances, unlike in the macro-averaged version where a per-class average is computed. Micro-averages can nonetheless possibly result in a bias towards the more numerous classes. In Equations 4.5 and 4.6, micro-averaged precision and recall are respectively formally defined.

$$\text{Micro-Precision} = \frac{\sum_{i=1}^{N} tp_i}{\sum_{i=1}^{N} (tp_i + fp_i)} \tag{4.5}$$

$$\text{Micro-Recall} = \frac{\sum_{i=1}^{N} tp_i}{\sum_{i=1}^{N} (tp_i + fn_i)} \tag{4.6}$$

In order to combine the information that is given by both the precision and the recall metrics in a single score, an additional metric named the F_1 score can be used. This metric, formalized in Equation 4.7, computes the harmonic mean between precision and recall. It should be noted that both the micro and macro versions of precision and recall can be used for computing F_1 scores.

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.7}$$

To properly evaluate the classification models, one needs to separate the available data into two parts: a training part which is used to learn the classification model, and a test portion that is used to evaluate the learned model. This way, the generalization capacity of the model can be tested, i.e., the prediction quality of the model concerning new unseen instances can be assessed. Nevertheless, a data split (e.g., 80% for training and 20% for testing, choosing instances randomly) will in general be associated with a considerable degree of bias because the training and the test sets can end up with a very different composition. In order to tackle this issue, one possible approach is to guarantee that both splits have a homogeneous composition. This approach is called stratified sampling. The homogeneity of the dataset's splits, employing the stratified sampling technique, is achieved by selecting instances for each split in a way that maintains the proportionality between the number of instances per class and the class distribution of the dataset's population.

4.1.4 Model Optimization 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

Table 4.1: Exploration of different data augmentation strategies for training. Highlighted in bold are the best results for each of the explored architectures.

Model	Strategy	Precision	Recall	F1-score	Accuracy
DenseNet-201	Basic Augmentations	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 Augmentations	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 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 [61] and Adam [62], 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.

In a first approach to the training of the aforementioned networks, the optimizers were scheduled using a stepwise annealing policy, which decreases the learning rate throughout training, starting at 10^{-3} with a learning rate minimum of 10^{-5}. However, due to the poor performance of this approach in preliminary tests, other schedules were explored. The best performance in the aforementioned tests was attained by models in which training leveraged the policy introduced by Smith [63], 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.

4.2 Experimental Results on the Pottery Black-and-White Diagrams Dataset

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 4.1 are the results of these experiments.

Table 4.2: 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	Precision	Recall	F1-score	Accuracy
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
	Random Forest	52.50	52.41	50.34	66.78
	Logistic Regression	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
	Random Forest	53.82	51.13	49.37	65.32
	Logistic Regression	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
	Random Forest	52.29	49.43	48.46	65.79
	Logistic Regression	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
	Random Forest	44.43	42.52	41.28	59.38
	Logistic Regression	48.16	48.53	45.97	**61.13**

Such results show a clear performance increase against basic augmentations in all of the three assessed metrics when techniques 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, the performance of various classification algorithms leveraged in the meta-learning approach was assessed. Having the results obtained in the previous experiments as a basis, the data augmentation procedure was fixed to the AugMix algorithm for these experiments. In addition to the architectures used in the remaining experiments (i.e., DenseNet-201 and EfficientNet-B3), the architectures and training techniques introduced in Sections 3.1.2 and 3.2.1, respectively, were tested. Table 4.2 depicts the results of these 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

Table 4.3: 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	Precision	Recall	F1-score	Accuracy
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
	Random Forest	54.07	52.07	51.02	66.49
	Logistic Regression	55.74	54.04	52.93	67.37
DenseNet-201 w/ self-training	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
	Random Forest	55.45	54.85	53.58	68.53
	Logistic Regression	59.52	58.05	56.74	69.69

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 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.

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 4.3. 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,

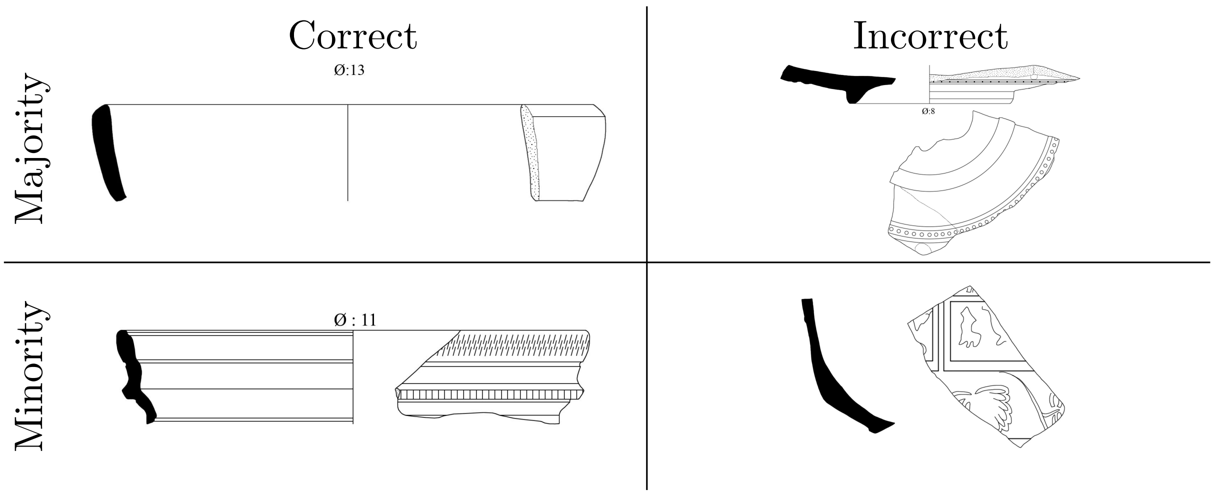

Figure 4.6: Example of four artifact predictions leveraging the DenseNet-201 model with the 5-NN classifier. Two dimensions are depicted in the figure above: whether the instance's class is one of the most numerous classes and the correctness of the system in the image classification task. From top to bottom, left to right, the considered instances' categories: Ritterling 8, Dragendorff 29/37, Goudineau 27, and Dragendorff 29.

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 4.3, 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 4.6 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 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.

An analysis of the quality of the representations learned by the convolutional neural network architectures can be observed in Figure 4.7, 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 differ-

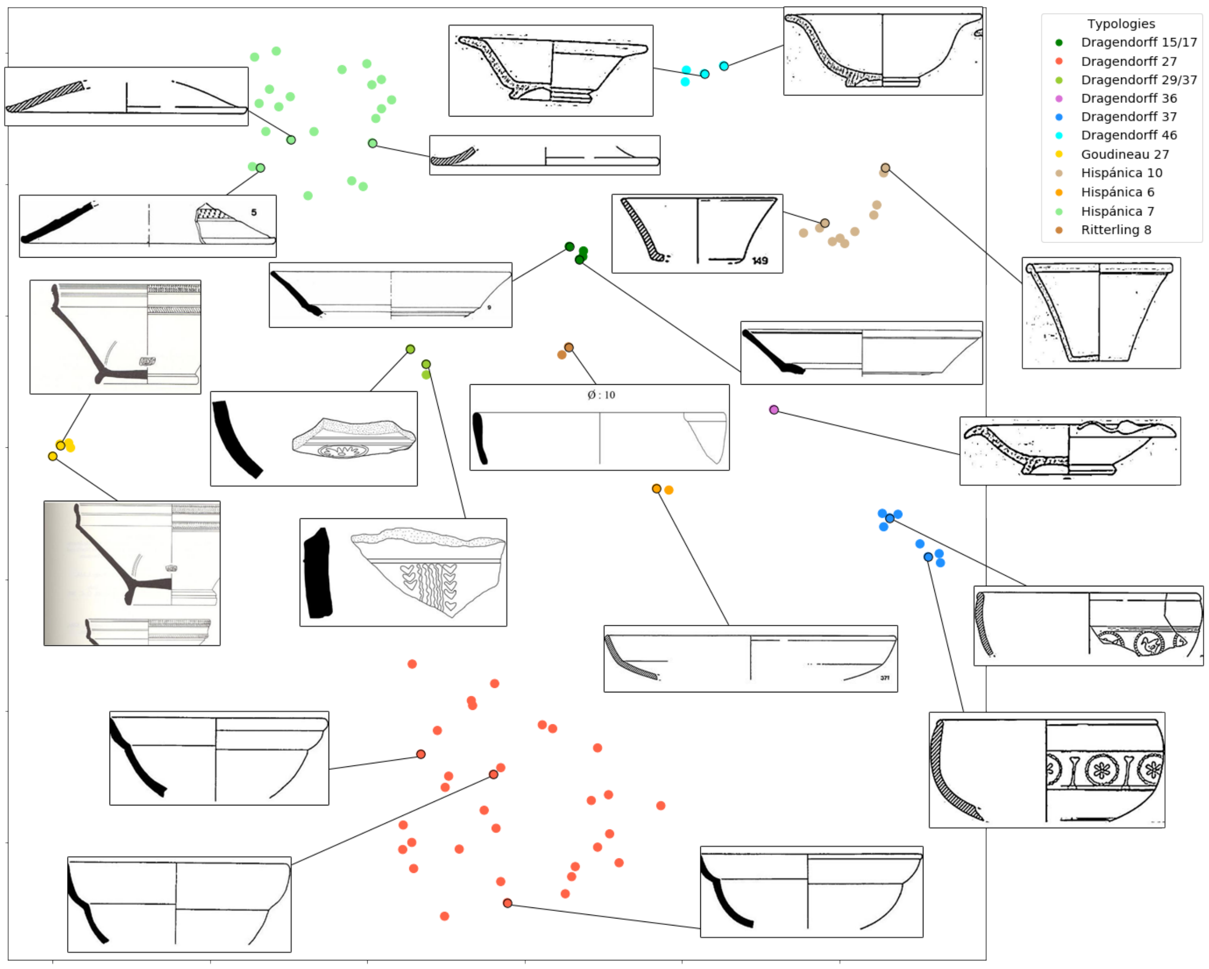

Figure 4.7: Scatter plot of the DenseNet's learned representations for the available data instances with randomly selected example diagrams from each of the displayed clusters highlighted. The embedding to a two-dimensional space was obtained leveraging the t-SNE technique with the perplexity hyperparameter, regulating focus on more local or global data features, set to 45.

ent 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) [64] 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 4.7 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

Table 4.4: Exploration of different data augmentation strategies for training. As in the previous tables the best performances are highlighted in bold.

Model	Strategy	Precision	Recall	F1-score	Accuracy
DenseNet-201	Basic Augmentations	78.80	74.48	74.22	79.66
	MixUp	76.77	74.70	73.67	78.80
	AugMix	**79.49**	**76.19**	**75.79**	**81.13**
EfficientNet-B3	Basic Augmentations	74.72	72.61	71.54	77.27
	MixUp	73.65	72.50	70.88	76.17
	AugMix	**76.99**	**74.15**	**73.34**	**79.29**

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 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.

4.3 Experimental Results on the Pottery Color Photographs Dataset

Since the color photograph dataset is composed of two views (front and side of the artifact), experiments leveraging single input architectures for each of the available views were originally considered. However, initial experiments reported the poor performance of models trained solely on images of the side of the artifact. Consequently, for the remaining experiments, except for the two-input architectures, only the front view was used.

Considering the color photograph dataset, identical experiments to determine the optimal data augmentation for model training were first performed for the two considered convolutional neural network architectures. Table 4.4 reports the results of the experiments pertaining to data augmentation. As in the black-and-white pottery diagram dataset, models trained with the AugMix data augmentation technique achieved the best performance across all the employed evaluation metrics, with improvements of about 1-3%. However, in contradiction with the results found for the black-and-white dataset, the fixed set of augmentation operations performed better in all four metrics by approximately 1-2%. The reason for this performance inversion can be attributed to the different nature of the two datasets' images, namely regarding the checkered ruler present in all the images from the color photograph dataset. MixUp, in this case, when two different photographs are combined in order to artificially generate a new data instance, places in the generated image two different rulers, which can cause difficulties in determining for each pottery fragment the corresponding correct ruler. Regarding the performance of the DenseNet-201 and EfficientNet-B3 architectures, a large performance discrepancy can be observed, with DenseNet-201 clearly emerging as the most successful architecture, presenting approximately 2% higher scores across all tested metrics. This discrepancy between architectures is consonant with the difference in performance reported in the context of the black-and-white diagram dataset.

Table 4.5 reports the performance results of the meta-learning approach applied to the models

Table 4.5: Results leveraging the meta-learning approach. 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	Precision	Recall	F1-score	Accuracy
DenseNet-201	None	79.49	76.19	75.79	81.13
	1-NN	**81.48**	**77.27**	**77.53**	**82.35**
	3-NN	81.29	76.21	76.77	82.31
	5-NN	79.26	74.42	74.72	80.88
	7-NN	78.94	73.99	74.41	80.70
	Random Forest	78.53	69.78	71.45	79.35
	Logistic Regression	80.68	73.22	74.67	79.90
EfficientNet-B3	None	76.99	**74.15**	**73.34**	**79.29**
	1-NN	**77.52**	73.85	73.14	79.04
	3-NN	76.48	72.78	72.01	78.37
	5-NN	75.98	72.28	71.55	78.25
	7-NN	77.29	72.82	72.40	78.25
	Random Forest	74.33	66.79	67.68	75.61
	Logistic Regression	75.97	71.34	71.28	78.00

trained with the AugMix technique. As can be assessed, meta-learning using the 1-NN classifier over the representations learned by the DenseNet-201 model achieves the highest performance when compared with the remaining tested classification algorithms, scoring 81.48%, 77.27%, and 77.53% for precision, recall, and f1, respectively. The performance increase over standard network prediction is of 1.74% in terms of the f1 score, with precision, recall and accuracy also presenting performances improvements of 1-2%. Another k-nearest neighbors classifier achieves the second best performance, namely 3-NN, with a 0.98% increase in the f1 score over the DenseNet's standard network prediction. Of the remaining tested classifiers, only logistic regression is able to surpass the original prediction performance, exclusively in terms of the precision metric, with an increase of 1.19%. Regarding the EfficientNet-B3 architecture, for only the precision metric in two k-nearest neighbors variants, 1-NN and 7-NN, can performance increases be observed. Concerning the two architectures, the substantial decrease in performance gain by the application of meta-learning can possibly be attributed to the acute class-imbalance present in the color photograph dataset in conjunction with the fact that more than half of the 54 classes are composed of less than 20 instances (Figure 4.3). This way, the photograph dataset presents even less amenable conditions for successful model classification performances when contrasted with the black-and-white diagram dataset.

Moreover, since the color photograph dataset contains two views for each pottery photograph, namely a front and side view, experiments leveraging the two-input variants of the two studied architectures, trained with AugMix, were considered. In spite of the added second view, these two-input variants performed substantially worse than their single input counterparts. The 2-DenseNet-201 model, in terms of standard network prediction, scored 75.22%, 70.41%, and 70.21%, for precision, recall, and f1, respectively. On the other hand, as was the case for the single input version of this network, 2-EfficientNet-B3 performed worse across all measured metrics, having scored 70.44%, 68.20%, 66.94%, for precision, recall, and f1, respectively. The aforementioned decrease in performance can conceivably hypothesized to be the result of the diminutive differences between the side views of artifacts belonging to different

Table 4.6: The three best and worst-performing typologies concerning the f1-score are presented here. The reported results are those of the DenseNet-201 model w/AugMix leveraging a 1-NN classifier. By descending order of performance, the analyzed typologies are: *oaxaca polychrome* (*OaP*), *puebla polychrome* (*PuP*), *panama polychrome type a* (*PaPTA*), *mexico city copy of puebla blue on white* (*MexPUBOW*), *puebla blue on white variant with black* (*PuBOWV bl.*), and *esquitlan polychrome* (*EsP*).

Top 3 Typologies			**Bottom 3 Typologies**		
Typology	F_1-score	Support	Typology	F_1-score	Support
OaP	100.00	11	*EsP*	31.43	13
PuP	95.19	82	*PuBOWV bl.*	37.33	11
PaPTA	94.55	69	*MexPUBOW*	47.33	12

classes. Concerning meta-learning, in the experiments performed over both architectures, none of the employed classification algorithms scored higher than standard network prediction. This result is in line with the performances reported for the experiments concerning the diagram dataset, where in general two-input architectures performed worse in meta-learning.

Table 4.6 reports, according to the f1 metric, the performance of the top and bottom three *majolica* typologies. As expected by the imbalanced nature of the photograph dataset, the three worst-performing typologies are all classes with a very limited number of examples. Moreover, two of these typologies, namely *mexico city copy of puebla blue on white* and *puebla blue on white variant with black*, are also very similar to other, more sizable typologies, in particular to the most voluminous class of the dataset, *mexico city copy of puebla blue on white*. Consequently, such similarity between typologies impacts the performance of these minority classes, which are mistaken for the similar majority class. Regarding the better performing classes, as expected, the more numerous classes, namely the second and the fifth largest classes (Figure 4.4), appear in the top three concerning performance. However, it is a minority class, the *oaxaca polychrome* typology, which best performs, obtaining a perfect score across all performance metrics. The unexpected classification performance of this typology can possibly be attributed to the uniqueness of both the motifs and colors of its decorations in the context of the dataset.

4.4 Summary

In this chapter, the experiments performed in the context of the evaluation of both the black-and-white diagrams and color photographs image classification tasks were expounded. The performance metrics employed in the aforementioned evaluation were likewise introduced. A detailed description of the context, data instances, and class balance of the two explored datasets was provided in Section 4.1.

Section 4.2 described the results of the performed experiments regarding the black-and-white pottery diagram dataset. Although the obtained results have a clear margin for improvement, the meta-learning approach provided a performance increase in relation to simple network prediction and the adapted noisy student approach further improved classification performance. The reported results concluded that a single-input DenseNet-201 architecture trained with AugMix achieved the highest performance. Moreover, by employing the t-SNE technique over the representations learned by the aforementioned convolutional neural networks, clearly defined clusters for each typology substantiate the performance superiority of the k-NN algorithm over the remaining examined classifiers.

Section 4.3 reports the performances of the models developed in the the color photograph setting. As in the black-and-white diagram dataset, DenseNet-201 trained with the AugMix data augmentation technique achieves the best performances. Applying a k-nearest neighbors algorithm over the representations learned by the convolutional neural networks (the most successful technique for the first dataset) likewise increases performance over the standard network prediction. In terms of the preferred object view, a single image of the front of the pottery artifact shows a clear performance gain over the remaining tested object views.

Chapter 5

Conclusions and Future Work

This dissertation proposed an automatic classification system based on convolutional neural networks with the goal of categorizing black-and-white diagrams or photographs of pottery artifacts into typologies. In this chapter, the principal contributions of this work, as well as potential new directions for additional work on the pottery image classification task, are presented.

5.1 Contributions

This work presented an approach based on deep convolutional neural networks for the automatic classification of images of archaeological pottery artifacts into typologies. Two different modes of representation for these artifacts were considered, namely black-and-white diagrams and color photographs, with new datasets being introduced in this dissertation for each of the modes. Furthermore, an unlabeled pottery dataset was assembled based on images extricated from relevant publications of the archeology area. The developed automatic classification system was the result of extensive experiments over four dimensions: convolutional architecture, data augmentation, meta-learning, and self-training.

Alternative architectures were tested and conclusively selected models based on a single DenseNet-201 as achieving best classification scores. By examining the two datasets, problems emanating from data instance scarcity were anticipated as the principal obstacle for the pottery classification task, with three types of techniques hypothesized as possible attenuators, namely data augmentation, meta-learning, and self-training. Firstly, data augmentation in the form of the AugMix technique provided a significant performance increase over more elementary data augmentation procedures, with detailed performance comparisons against additional methods of this nature reported. Secondly, taking inspiration in the few-shot learning setting, meta-learning approaches consisting in leveraging supervised classifiers over instance representations obtained from convolutional neural network models, were suggested as a possible path for performance increase. The meta-learning approach, namely based on k-nearest-neighbors classifiers, was assessed as the best suited for the two studied pottery datasets, thus confirming the effectiveness of the approach regarding small datasets. Thirdly, considering the success of previous semi-supervised learning techniques in leveraging unlabeled data for the performance improvement of models trained over small datasets, self-training in the form of the Noisy Student approach was considered as the possible concluding component of the automatic classification system. The Noisy Student method, concerning the diagram dataset, achieved considerable performance increases over models trained exclusively with labeled data, thus corroborating the original intuition.

In conclusion, a number of potential limitations of the developed system need to be taken into account. First, as discussed in previous chapters, class imbalance paired with sample scarcity constitutes the principal weak point of the newly assembled pottery datasets, with this characteristic of the datasets possibly creating obstacles that cannot be solved with the techniques presented in this dissertation. Second, regarding the color photograph dataset, only the front view of the pottery artifacts was entirely used in the course of the performed experiments, with the side view of the objects being eliminated as an alternative depiction due to poor performance in preliminary tests, thus highlighting the bias of the approach proposed in this work to frontal depictions of the artifacts. Finally, regarding the self-training approach in the context of the diagram dataset, only a small portion of the unlabeled data instances is leveraged in this approach, a consequence of the 95% confidence threshold used for pseudo-labeling, emphasizing the dependence on well-chosen unlabeled data instances for the success of the Noisy Student procedure in this classification task. The next section addresses various approaches tending to some of the aforementioned limitations.

5.2 Future Work

An analysis of the results of the experiments performed over the two pottery image datasets suggests that class imbalance was a significant obstacle for higher classification performances. In the future, in order to tackle this issue, several additional techniques can be applied to the pottery image datasets. A possible approach is the modification of the loss functions utilized for model training (e.g., softmax cross-entropy) to better deal with class imbalance. Focal loss, introduced by Lin et al. [66], is an example of this, where, by adding a modulating factor to the cross-entropy loss function, the focus on data instances that are more difficult to classify is increased, with easily classified samples having a reduced loss value. An alternative, more general method was introduced by Cui et al. [66], which proposed weighting class samples based on class volume, with the goal of improving classification accuracy in long-tailed datasets, where the weight for a certain class is simply given by $(1 - \beta^n)/(1 - \beta)$, with n representing the number of samples for that particular class and β a hyperparameter. The aforementioned technique can be applied to both cross-entropy and the previously introduced focal loss.

Regarding the relatively small size of both studied datasets, one of the taken approaches to curb the effect of this dataset feature was the use of data augmentation. With data augmentation techniques substantially improving model performance, additional studies into different methods of this type could hypothetically further ameliorate model classification capabilities. For example, GridMask, introduced by Chen et al. [67], achieves state-of-the-art results in several image classification tasks while only using a very small quantity of computational resources. In this method, which is based on information dropping, new data instances are generated by removing randomly sampled uniformly distributed square regions of the training images in order to bolster regularization. Another possible alternative is the FMix [68] data augmentation technique. Taking into consideration the success in image classification tasks of models trained with techniques leveraging Mixed Sample Data Augmentation (e.g., MixUp), Harris et al. propose a novel data augmentation method where pairs of data instances are combined by a randomly sampled binary mask, which aims to maximize the space of edge shapes. Compared with MixUp, a technique which generates new instances by interpolating pairs of examples, FMix consistently outperforms it in image classification tasks.

The results from the experiments where self-training was employed seem to suggest that other techniques that leverage unlabeled data can also be successful in bettering the f1 score of models trained over the pottery image datasets. The FixMatch technique, developed by Sohn et al. [69], incorporates the unlabeled examples into training by labeling them with consistency regularization [70] and pseudo-labeling. For each unlabeled example two different new data instances are created: one "weakly" augmented (i.e., augmentations operations such as random horizontal flipping or mild horizontal shifting) and other "strongly" augmented with a technique such as RandAugment [71]. The "weakly" augmented example is classified by the model, with its prediction being treated as the true label for the "strongly" augmented version of image. Concerning the principal benchmarks for semi-supervised learning, FixMatch achieves state-of-the-art in their majority using a less complex method, thus becoming a suitable candidate for future experiments.

Recently, Khosla et al. [72] adapted contrastive learning, normally applied to self-supervised learning, to a supervised setting. In a self-supervised setting, contrastive learning measures loss by comparing the representations of a data instance (called the "anchor") and one augmented variant of the same example (the "positive" sample) with the remaining of the batch's data instances (the "negative" samples). On the other hand, supervised contrastive learning takes as "positive" all the images of the same class and "negative" the remainder of the batch. This approach, as reported by Khosla et al., results in an embedding space that closely groups data instances of the same class, thus potentially leading to improved performances in classification tasks.

Lastly, an additional direction for future research is related to few-shot learning in a class adaptive setting, i.e., in classification settings where the set of classes for training and testing is disparate. In this dissertation, the pottery image classification problem was approached from the standard training and testing procedure, where the model is trained over a dataset composed of a fixed set of classes and its performance measured over data instances that belong to the same class set. However, due to the extensive array of possible typologies for recovered archaeological pottery artifacts, techniques that seamlessly adapt neural network models to new unseen typologies based only on a handful of data instances could be an interesting target for future work. In this context, few-shot learning has an identical aim: based on training on a set of voluminous base classes, with only access to a very small number of examples belonging to novel classes, to successfully classify examples from these novel classes in the future. More precisely, in few-shot learning tasks, both the training and testing phase of the classification model is achieved using an episodic scheme (i.e., each epoch is composed of a predetermined number of episodes). Episodic training consists in, from the chosen training set composed of N data instances pertaining to B different classes (referred as the base classes), randomly sampling K training data instances from S previously sampled classes, training scenario which is denominated an S-way K-shot setting, on each epoch's training episodes. Such sampling is effected at the beginning of each of the training episodes purporting to simulate, in this way, an environment where only a diminutive amount of instances per class is available, however, it should be noted that the base classes used for training all present a vast number of examples. The same process is then repeated in the evaluation phase for the test portion of the dataset: from the totality of the test dataset's classes, C classes, and I test images from each of the aforementioned classes are sampled. Consequently, the system's final performance is given by the average performance on the total amount of test episodes. In the majority of training done in such a manner, in order to achieve the best performance, the number of classes and number of examples per class both in training and testing phases is maintained [49]. The two meta-learning techniques [10, 9] employed in this dissertation are, as previously mentioned, adapted from few-shot learning and, consequently, can be used for possible scenarios of pottery classification in a few-shot learning context. Moreover, more recently, Ziko et al. [73] introduced the state-of-the-art Laplacian Regularized Few-Shot Learning. In this method, as in SimpleShot, a model is trained over the base classes leveraging a standard cross-entropy loss, with the corresponding learned representations posteriorly being harnessed by the specific few-shot technique for the refitting of the model to the novel classes. The technique itself consists in minimizing a quadratic function that performs binary assignment, with the function being divided into two parts (i) the computation of the nearest class prototype for each query sample and its attribution to the sample in question, (ii) the application of a Laplacian regularizer enforcing consistency in the labeling of neighboring query samples. This approach, when compared with other state-of-the-art approaches (e.g., SimpleShot), consistently outperforms them in the main few-shot learning benchmarks and, as a result, can be added to the array of techniques to be applied to pottery image classification in a few-shot setting.

Bibliography

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

[2] C. M. Stibbe. Exceptional shapes and decorations in laconian pottery. *British School at Athens Studies*, 4, 1998.

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

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

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

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

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

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

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

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

[11] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4), 1943.

[12] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 1958.

[13] M. Minsky and S. Papert. *Perceptrons: an introduction to computational geometry*. MIT Press, 1969.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Readings in Cognitive Science*. Morgan Kaufmann, 1988.

[15] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence O(1/k^2). *Doklady an SSSR*, 269, 1983.

[16] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, arXiv:1412.6980, 2014.

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

[18] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4), 1980.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 2012.

[21] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*, 2015.

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

[23] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. In *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

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

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

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

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

[28] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for Activation Functions. *arXiv e-prints*, arXiv:1710.05941, 2017.

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

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

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

[32] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35, 1943.

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

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

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

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

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

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

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

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

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

[42] *Classificare le ceramiche: dai metodi tradizionali all'intelligenza artificiale. L'esperienza del progetto europeo ArchAIDE*, 2018.

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

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

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

[46] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *Proceedings of the European Conference on Computer Vision*, 2016.

[47] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. RandAugment: Practical automated data augmentation with a reduced search space. *arXiv e-prints*, arXiv:1909.13719, 2019.

[48] D. Hendrycks and T. G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proceedings of the International Conference on Learning Representations*, 2019.

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

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

[51] A. Crawford. Ceramics, roman imperial. In *Encyclopedia of Global Archaeology*. Springer New York, 2014.

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

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

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

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

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

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

[58] F. C. Lister and R. H. Lister. Maiolica in colonial spanish america. *Historical Archaeology*, 8, 1974.

[59] F. C. Lister and R. H. Lister. The first mexican maiolicas: Imported and locally produced. *Historical Archaeology*, 12, 1978.

[60] F. C. Lister. *Sixteenth Century Maiolica Pottery in the Valley of Mexico*. University of Arizona Press (Tucson, AZ), 1982.

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

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

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

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

[65] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

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

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

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

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

[70] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Proceedings of the Neural Information Processing Systems Conference*, 2016.

[71] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the Neural Information Processing Systems Conference*, 2020.

[72] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. In *Proceedings of the Neural Information Processing Systems Conference*, 2020.

[73] I. M. Ziko, J. Dolz, E. Granger, and I. B. Ayed. Laplacian regularized few-shot learning. In *Proceedings of the International Conference on Machine Learning*, 2020.