Creating an Additional Class Layer with Machine Learning to counter Overfitting in an Unbalanced Ancient Coin Dataset

Sebastian Gampe*[1], Karsten Tolle[1]

[1] Big Data Lab, Goethe-Universität – Frankfurt am Main, Germany

*Corresponding author
Correspondence: gampe@em.uni-frankfurt.de

Abstract

We have implemented an approach based on *Convolutional Neural Networks* (CNN) for mint recognition for our *Corpus Nummorum* (CN) coin dataset as an alternative to coin type recognition, since we had too few instances for most of the types (classes). However, this shift increased an existing problem with our dataset: the extremely unbalanced number of instances per class. While some of our classes consist of only 20 instances, others consist of several hundred. After training our VGG16 model we unsurprisingly observed an overfitting of these "big" classes within the confusion matrix. To reduce this problem, we tried to split the dominating classes with the most images into several smaller ones and called them additional class layers. We use three different machine learning (ML) approaches to perform this breakdown. One is an unsupervised clustering method without additional manual work. The other two are supervised approaches which explicitly take into account the motifs of the coins themselves: a) an Object Detection model that predicts trained entities, and b) a *Natural Language Processing* (NLP) method to find entities in the textual descriptions of the coins. Based on the combination of obverse and reverse results from these two approaches new additional class layers were defined for each of them independently. After retraining our mint recognition model with these new classes, we evaluated the results based on the confusion matrix. In our case, the best results could be observed by forming an additional class layer based on the NLP method. Unfortunately, in our situation the overfitting problem could only be reduced and not eliminated.

Keywords: Machine Learning, Image Recognition, Convolutional Neural Networks, Unbalanced Dataset, Ancient Coins

Introduction

In our current project D4N4 – Data quality for Numismatics based on Natural language processing and Neural Networks (D4N4 n.d.) we want to implement a Machine Learning (ML)-based coin type recognition model that covers as many coin types as possible from the “Corpus Nummorum” (CN) (Corpus Nummorum n.d.) dataset. The goal is to use it to improve and verify the data quality of existing data, and also use it to help the process of entering new coins. The CN dataset features about 19,600 different coin types and more than 49,000 coins from four different ancient landscapes (Thrace, Moesia Inferior, Troad and Mysia). This dataset contains coins from several different museums, institutions and collectors. The largest part of our images comes from the Berlin-Brandenburgische Akademie der Wissenschaften, the Münzkabinett Berlin and the Bibliothèque nationale de France, Département des Monnaies, médailles et antiques. We published the images from these three institutions (due to existing copyrights) as a dataset for ML research on Zenodo (Corpus Nummorum 2023). A coin in the database is generally represented by images of the obverse and reverse of the original, or by a plaster cast. In rare cases, both representations are assigned to a coin record. For our ML dataset, we merged the obverse and reverse images of a coin into a single image showing both (as can be seen in fig. 1).

The biggest challenge for our type recognition is the ratio of an average of approximately two coin images per type. In previous experiments we learned that 20 coin images per class threshold is a reasonable starting point to achieve good results in the training for our data. This means that for most of the types our dataset currently has too few coins (Gampe 2021, Gampe and Tolle in print 2019). Currently only 179 of the 19,600 type classes can meet the condition. The VGG16 model, which was pretrained on the ImageNet dataset (ImageNet 2021), achieves a Top-1 Accuracy of 82% (tab. 1) based on those types that meet the threshold (>20 images), with the drawback that 99% of the type classes in the CN dataset are not part of the training[1]. We are constantly increasing the number of our images and we have been able to double the number of trainable types since the start of the project.

Figure 1 – The merged images of CN coin 2377 (CN type 763; mint : Maroneia) and CN coin 18232 (CN type 11944; mint: Pergamon) with black bars for the quadratic input format of the CNNs (Photos : Münzkabinett Berlin)

In order to get better coverage and still generate something useful, we trained a different model to recognize another important aspect of our coins: the mint in which the coin was produced. Predicting a coin’s mint could also reduce the workload for our numismatists since the presorting of a larger number of coins by mint can save a lot of time. We kept our VGG16 setup and only needed to change the training and test set. Currently there are 122 mints in our data set. With the 20-coin image threshold, 98 of them are eligible for training. This way we could use about 40,000 (~80%) of our images for this approach. The remaining 24 mints have less than 200 images combined. The accuracy values here are comparable to the type recognition (Top-1 Accuracy: 79%, tab. 1) in spite of the fact that the individual mint classes are much more disparate than the type classes. Most mint classes consist of several different coin types which differ

[1] The Top-1 Accuracy shows the percentage of correct results of the predictions with the highest probability. The Top-5 Accuracy is the percentage of correct predictions within the five most likely predictions.

more or less from each other, Pergamon for example has 653 different coin types (630 are currently published on the website)[2].

Table 1 – Metrics for the models with and without the additional class layers

CNN Architecture	Class Type	Additional Class Layer	Number of Perinthos / Pergamon classes	Top–1 Accuracy	Top–5 Accuracy
VGG16	Types	None	1	82%	98%
VGG16	Mints	None	1	79%	94%
VGG16	Mints	DeepCluster	15 / 15	73%	91%
VGG16	Mints	DeepCluster	10 / 10	74%	92%
VGG16	Mints	Object Detection	8 / 10	78%	93%
VGG16	Mints	Object Detection	4 / 4	77%	93%
VGG16	Mints	NLP	16 / 16	76%	92%
VGG16	Mints	NLP	8 / 9	78%	93%

However, we still encounter the problem of having an unbalanced dataset when training the mint recognition. The advantage of the significantly higher number of usable images comes with the problem of a clearly higher amount of instances in a single class. Due to the different production patterns of the mints and the disparity in the number of coins of each ancient city in our area of interest, we also have significantly different coin image numbers for each city. Some of them have only the threshold value of 20 images while others have several hundred (fig. 2).

Such big differences in the number of images per class can lead to a phenomenon known as overfitting. A CNN model learns to recognize some classes better than others due to the larger number of images, with the result that the weights in the trained network can be more tuned to these classes during training. Subsequently, it is possible that images of other smaller classes are predicted preferentially as belonging to one of these overfitted classes. Overfitting can also appear by too many epochs during the training. In our case and in this paper this source for overfitting is neglected and we concentrate on the inhomogeneity of the data set as a basis for overfitting. A good way to determine if overfitting is present in a model is to create the so-called Confusion Matrix (CM) (fig. 3). The CM is a common way to make overfitting visible for individual classes in a multiclass problem. In the CM the predictions of the model and the true classes (ground truth) are confronted. In the case of a model that is 100% correct for all predictions, the diagonal of this matrix should be deep red. Recurring errors for one class create visible points off the diagonal. Traces of vertical lines for one or more classes are a good indicator for the overfitting problem. The CM of our most recent model shows these traces (fig. 3). They are clearly visible for the cities that have the most associated images like Pergamon and Perinthos (fig. 2). Every city with more than 1500 assigned images shows clear signs of overfitting on the CM. This overfitting is probably also responsible for the poorer performance of the model. The Pergamon line is the most prominent. Although Perinthos also has a large number of images, the overfitting problem is less pronounced here than with Pergamon. These differences and the fact that both mints produced very different looking coin types and thus present a challenge when dividing into new smaller classes make them a good case study in our view.

One idea for solving the overfitting problem is limiting the number of training images per class. The problem for the image limitation is the number of types which belong to a mint. For example, for the mint Pergamon there are over 3600 coin images, which are split over 653 different coin types. This means that if we want to represent each type with at least one coin image in the training set the Pergamon class would still be significantly larger than many other classes. A limited dataset can also lead to lower model

[2] Overview of the 630 published types in the CN dataset: https://www.corpus-nummorum.eu/search/types?type=quicksearch&mints%5B%5D=74 [accessed 15 December 2023].

performance because mint classes whose coins occur very often in archaeological finds will not have proper training for their range of types. The accuracy for the types which are very common in that mint will not be as good as it could be with an unlimited training set. Another method we tried in the overfitting context is the “compute_class_weight” algorithm from the scikit-learn package (scikit-learn n.d.). It computes class weights for every class for an unbalanced dataset like ours in order to avoid overfitting. The computed weight of a class with many images is significantly lower than that of a class with few images. Unfortunately, the performance of the trained model with such class weights was unacceptable. We got a Top-1 Accuracy of less than 1%. We also experimented with the “sparse categorical focal loss” function for our case (focal-loss n.d.). The function makes easy-to-classify images contribute less than hard-to-classify images during the training. However, training with this loss function had no positive effect at all.

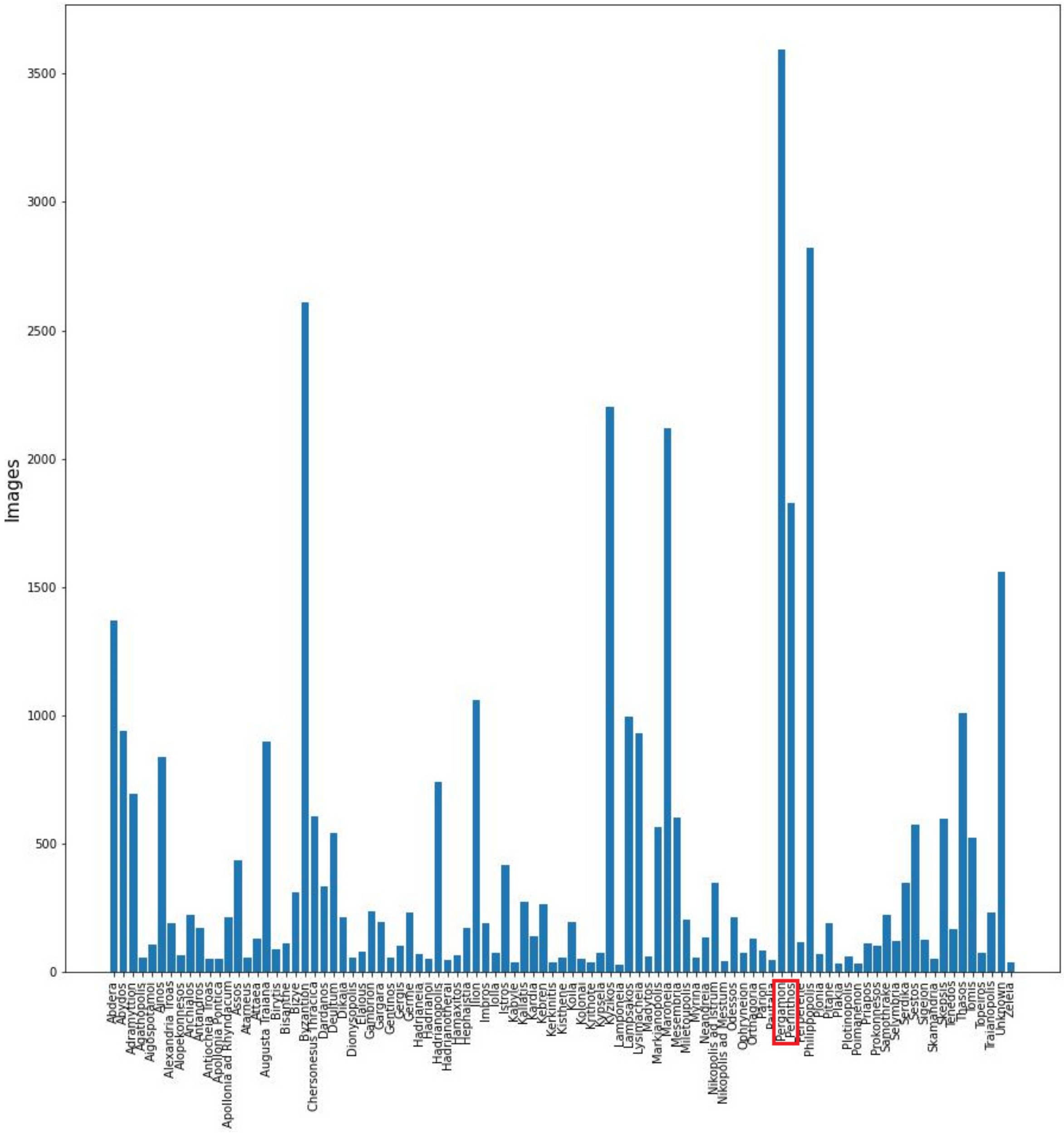

Figure 2 – Number of images for each mint in the Corpus Nummorum dataset. (Graphic: S. Gampe, Big Data Lab)

This paper deals with the problems we encountered when applying an image recognition (IR) approach to an ancient coin dataset. The focus is on our main challenge of a very unbalanced dataset consisting of classes with very few images and others with several hundred images. This means we focus here not on the improvement of machine learning (ML) algorithms as such, but on the setup, in particular the handling of the datasets and the definition of the result classes for the training.

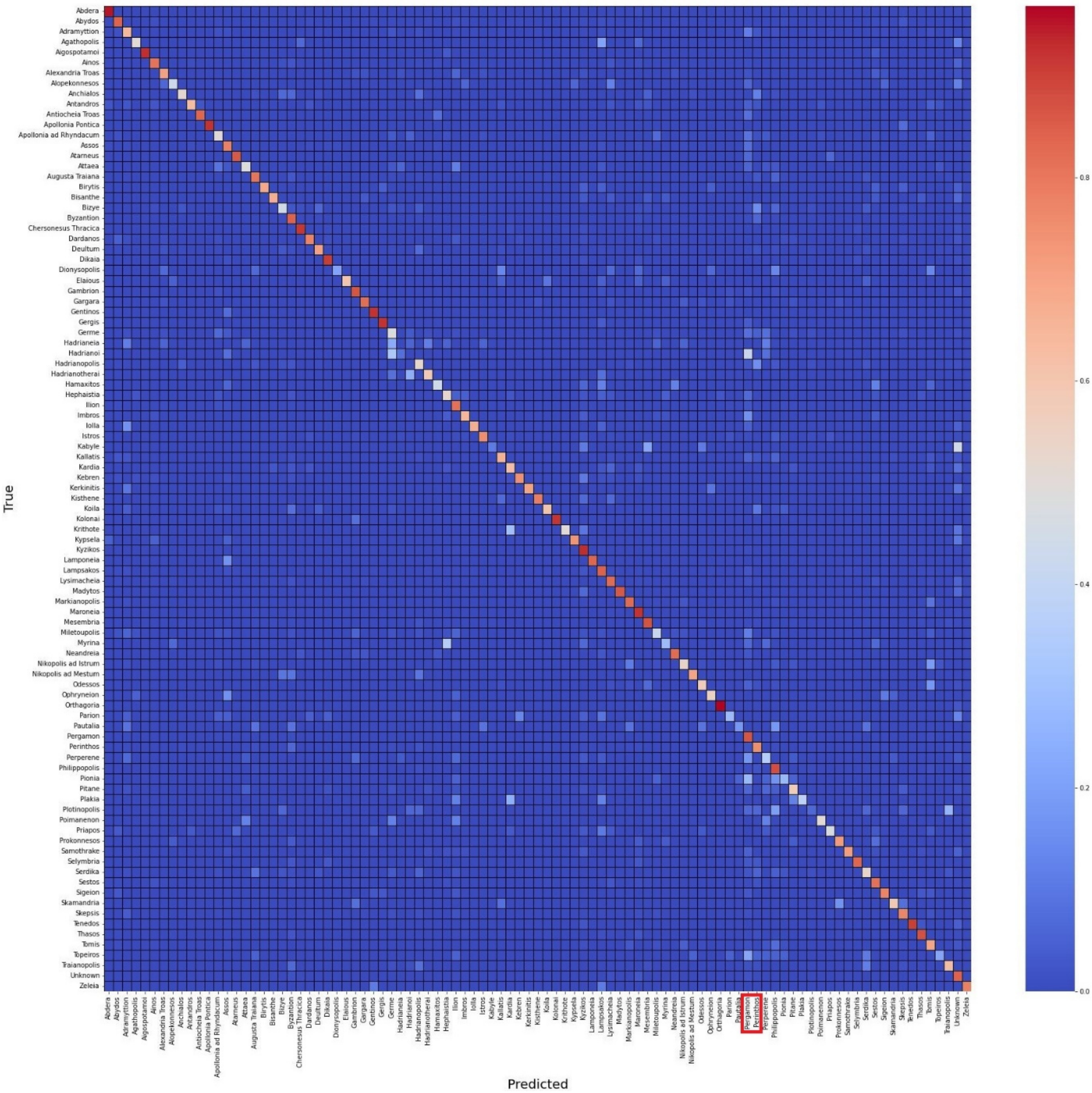

Figure 3 – Confusion matrix for the mint model without the additional class layer. (Graphic: S. Gampe, Big Data Lab)

The goal was to improve our mint recognition model by breaking down large classes with many input images into several smaller ones. For this purpose we used three different methods: 1. DeepCluster – an unsupervised clustering method, 2. Object Detection based on a Region Based – Convolutional Neural Network (R-CNN) and 3. our Natural Language Processing (NLP) pipeline (Gampe and Tolle in print 2019). The DeepCluster model is available on GitHub (GitHub – facebookresearch / deepcluster n.d.). The Object Detection approach trained for this paper is based on TensorFlow and Keras (Keras n.d., TensorFlow n.d.). The already existing NLP Pipeline was developed with the spaCy application programming interface (spaCy n.d.). Our Image Recognition models, which are based on a pretrained VGG16 model, are implemented with TensorFlow and Keras (Gampe and Tolle in print 2019, TensorFlow n.d., Keras n.d.). All of the above methods run on Jupyter Notebook and are written in Python programming language.

In another Project (ClaReNet) the DeepCluster Method was used to cluster a coin hoard with celtic coins. The already existing typology and the allocation of coins to them was checked with the method. The

R-CNN based Object Detection was utilized to crop the area of a portrait on imperial roman coins to prevent a CNN from making decisions based on the legend (Gampe 2021). Our NLP approach has been in development for some time (Klinger et al. 2018, Gampe and Tolle in print 2019) and is now also used for other reasons within the CN-project. We therefore wanted to check whether it could also be used to solve our overfitting problems.

Our idea for addressing this problem is to break down big classes like Pergamon and Perinthos into smaller ones. We call this the *additional class layer*. The term “layer” is well known from neural networks like the CNN, which has different layers (fig. 4). The last layer in such networks is called the softmax and is responsible for transforming the incoming numerical values from the preceding layers into the class’s probabilities (Amidi and Amidi n.d.). We call this layer also the class layer. Our idea is to add an additional layer for large classes on top of the existing ones (fig. 4).

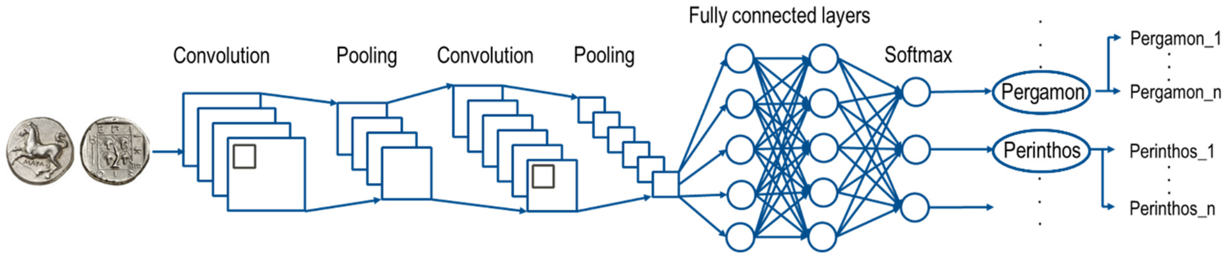

Figure 4 – Overview of a convolutional neural Network, its different layers and the additional class layer. (Photo: Corpus Nummorum. Graphic: S. Gampe, Big Data Lab)

In this process, classes like Pergamon are divided based on the similarity of their different types and new smaller classes are added. Then we add the old class layer with the new ones to get a new train and test set. To realize this approach, we use three different Machine Learning based methods:

1. Unsupervised “DeepCluster”
2. Region Based – CNN based Object Detection
3. Natural Language Processing

By creating these new classes and reducing the old large ones we tried to reduce the amount of overfitting and potentially increase the accuracy of our models. To test these approaches, we applied them to two different mints: Pergamon with c. 3600 images and Perinthos with c. 1800 images. Pergamon was chosen because it is the mint with the most associated images. In contrast Perinthos has just half as many images, however, this city has types that differ greatly from those from Pergamon due to the fact that the city belongs to another region (Pergamon – Mysia, Perinthos – Thrace). These differences are important to ensure the applicability of the three approaches to different mints.

Unsupervised “DeepCluster”

Our first method, *DeepCluster* from Facebook Research (Caron et al. 2018), combines unsupervised and supervised elements. An integrated CNN extracts features from the input images. Afterwards, these features are clustered with a k-means algorithm and the resulting clusters are used as pseudo labels. These newly labeled images serve as input for further CNN training. The number of clusters depends on the input parameter “k”, which can be freely selected before the start (Caron et al. 2018). One big benefit of this approach is the fact that DeepCluster needs no time-consuming adaptations to our problem. Values for a few parameters have to be chosen, such as the number of clusters “k” and the amount of training epochs. We chose 200 epochs for each attempt. The number of clusters was set to 15 and 10 because we didn’t want to create a large number of small classes. DeepCluster’s Algorithm uses all of our images from Pergamon and from Perinthos to form the 15 clusters for each mint separately. This means that both original classes have been completely split up and therefore no longer exist as classes in the training. However, it is possible that DeepCluster forms very inhomogeneous clusters with a lot of different looking types.

For our first attempt of dividing the Pergamon and Perinthos classes into smaller ones we had 15 clusters each (Pergamon_01 to Pergamon_15 and Perinthos_01 to Perinthos_15). Although both classes have different numbers of images, we have chosen an equal number of new classes to investigate the effects on these different sized classes. After the training most clusters had a size of 50 to 100 images for Perinthos and 100 to 200 for Pergamon. Both mints also had one cluster with many more images than the others: DeepCluster tends to build what we call “garbage clusters”, which contain all images that could not be assigned to the other clusters (Pergamon_10 and 12, Perinthos_02 and 11). They are somewhat comparable to the original classes with the leftover images from our two other approaches (see below). The 30 clusters built with DeepCluster are now incorporated as new classes in our train and test set replacing the original Pergamon and Perinthos classes. After the training we could observe that the Top-1 and Top-5 Accuracy values were below those from the unmodified model (tab. 1). We also created the confusion matrix for the new model on the test set (fig. 5).

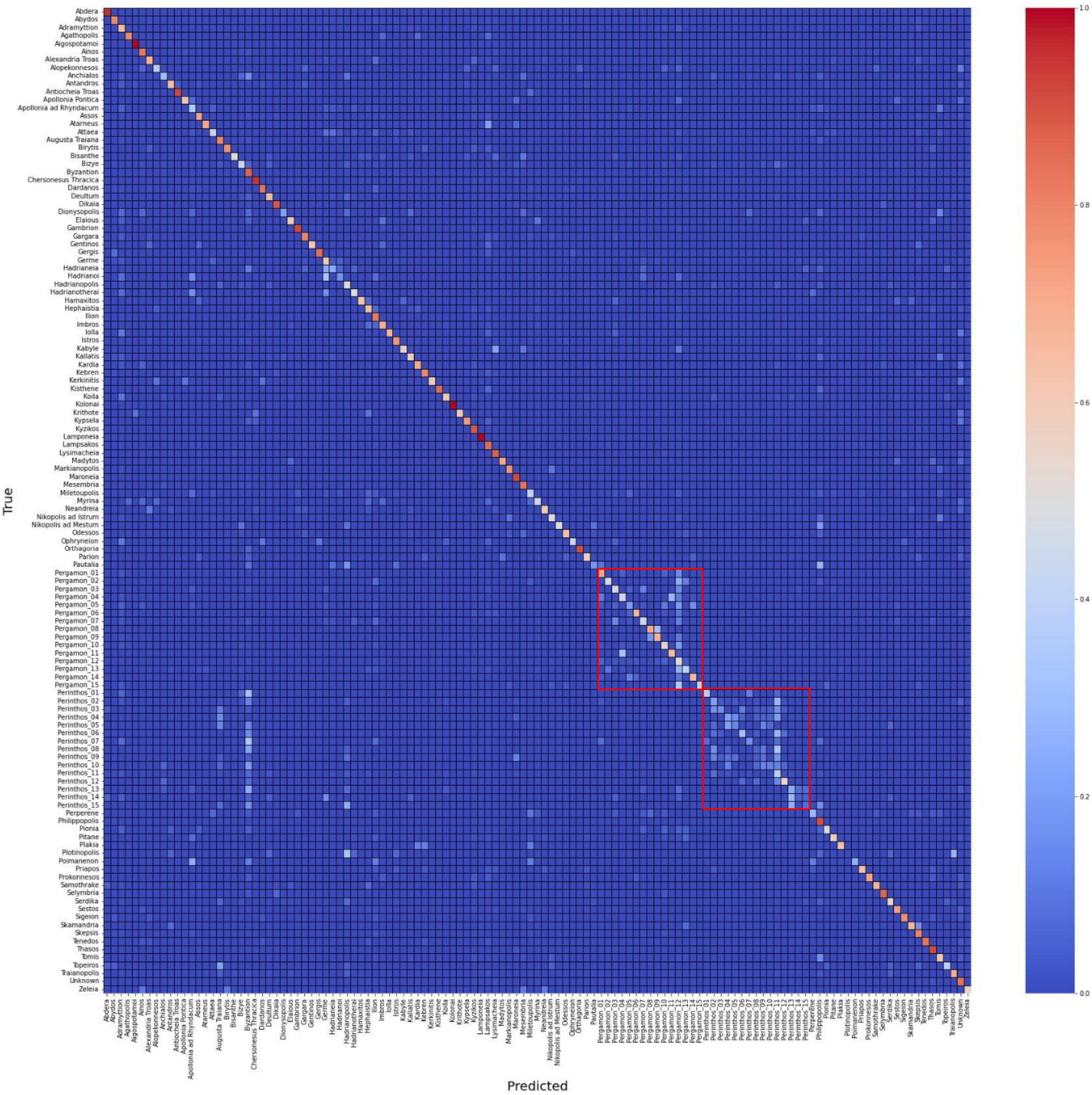

Figure 5 – Confusion matrix for the DeepCluster approach. (Graphic: S. Gampe, Big Data Lab)

What is immediately noticeable is that there is a clear vertical line in the area of the new classes. This means that these classes are often confused with each other. A look at the composition of the clusters indicates that this is due to the inhomogeneity of the assigned images within each cluster. While many clusters share images of the same or very similar types, the two biggest "garbage clusters" from both mints (Pergamon_12 and Perinthos_11) are clearly visible on the CM. These share a particularly large number of coin types with the other new classes. For Pergamon, the overfitting problem has diminished, but some of the brighter points from the unmodified class columns are now spread across the vertical lines of the new classes. For Perinthos we could also observe a slight decrease of the overfitting. However, new overfitting problems can also arise, as it is apparent in the Byzantion column at the level of the new Perinthos classes. We repeated this test with a smaller number of clusters. This time we executed DeepCluster with 10 clusters as preset. However, the result barely changed. The confusion between the new classes is still there and the overfitting for both mints is nearly the same as with 15 clusters. We also found a reduction of the Top-1 Accuracy of about 5% in both tests. These results have led us to conclude that the DeepCluster method is not suitable for our problem.

Region Based – CNN based Object Detection

Our second approach is an Object Detection model from Keras (Keras n.d.). It is based on a Region Based Convolutional Neural Network (R-CNN) which produces a set of region proposals that are likely to contain objects, and uses a CNN to extract features from each region proposal to classify objects within these regions. (Girshick et al. 2014). The way new classes were created here followed a different concept. We trained the R-CNN model on frequently occurring subjects on the coins like "head" or "sitting person". Most of these subjects are among the most common objects and animals in the CN database (Wirth 2021). The new classes were built based on the combinations of these subjects in the dataset[3]. The training of the R-CNN model was carried out by Huy Long, who wrote his master's thesis on this topic (Long 2022). He annotated 20 to 30 images for every subject with a polygonal annotation and with bounding boxes. With a polygonal annotated training set the model can predict the contour of an object, but annotation is more time consuming compared to the annotation of bounding boxes. After training the Object Detection model, it was used to predict subjects on each coin image of Pergamon and Perinthos. Based on these results, new classes had been built manually based on the resulting combinations of obverse and reverse subjects (e.g. Head–Owl). For Pergamon this generated eight and for Perinthos ten classes. These classes have different sizes with around 20 to 200 images assigned to them. This way we reduced the number of images in the original Pergamon and Perinthos classes to 2,520 and 1,170. After training our VGG16 model on the updated train and test set we observed that the top-1 and top-5 accuracy has hardly changed (tab. 1). But the resulting CM shows a problem similar to the DeepCluster approach (fig. 6). Images of the new classes are often attributed to the reduced Pergamon and Perinthos class. Furthermore, the overfitting for both original classes was not reduced as the CM shows. We repeated this test with a reduced number of new classes, where the images of some of the smaller classes had been manually merged back to the Pergamon and Perinthos classes. After retraining, beside a small decrease in the accuracy values, the overfitting problem also remained.

Clearly this approach is unfortunately not suitable for solving, or at least reducing our problem. An explanation for this could be the performance of the R-CNN Object Detection model. When examining the new classes, it became apparent that quite a number of objects had not been detected by the Object Detection model. The new classes were not really well distinguishable from the Pergamon and Perinthos ones. The annotation used seems not to cover the whole range of several subjects.

[3] The following subjects were used for the combinations: 1. Pergamon: "sitting_person", "head", "serpent_box", "owl", "serpents", "eagle", "podium", "bull", "bow". 2. Perinthos: "sitting_person", "head", "podium", "bull", "quadriga", "double_horse", "club", "laurel_wreath", "price_crown", "ship", "table", "pot", "standing_person".

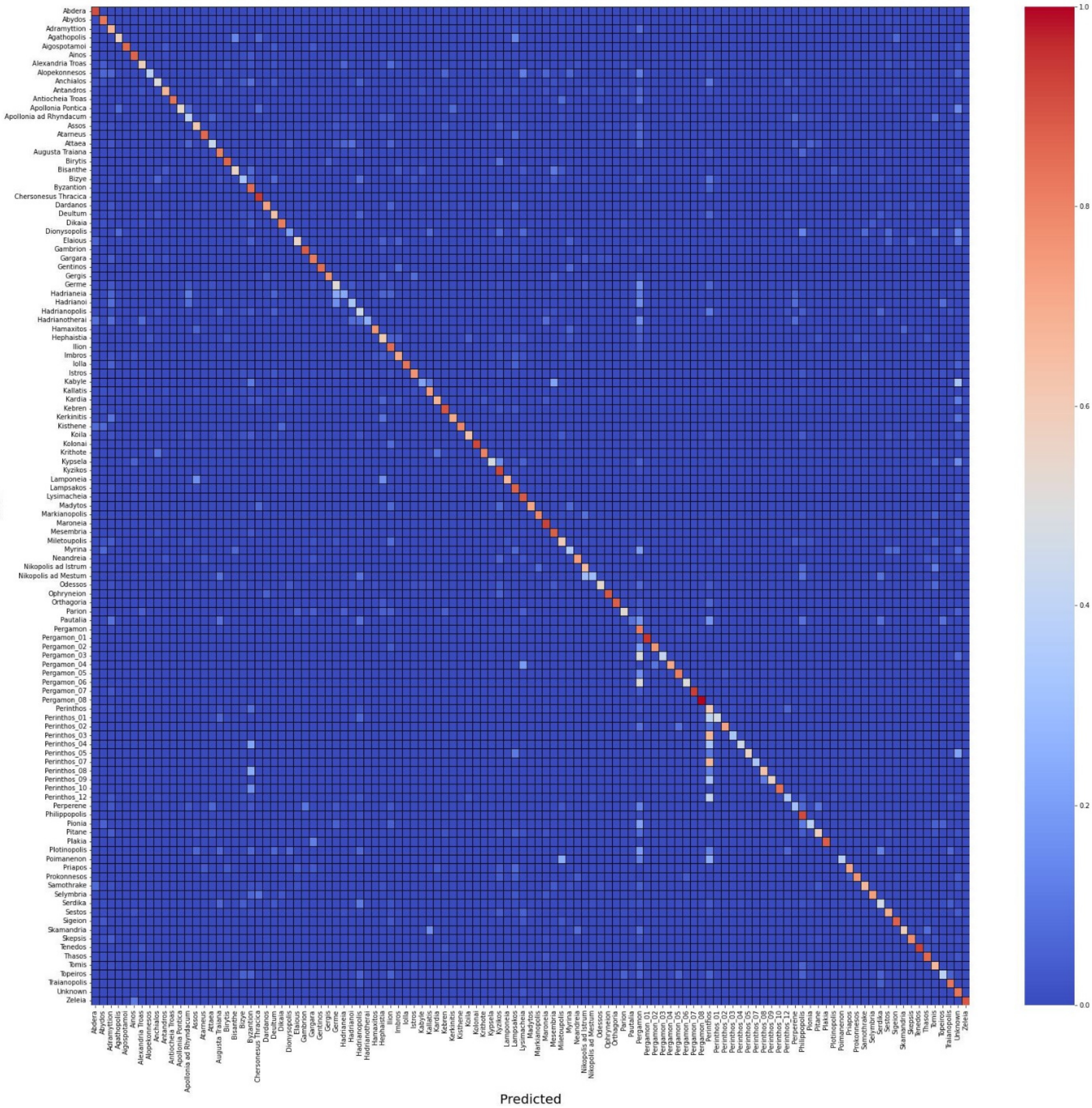

Figure 6 – Confusion matrix for the Object Detection approach. (Graphic: S. Gampe, Big Data Lab)

Natural language processing

Our final method for dealing with mint overfitting is the application of our Natural Language Processing (NLP) pipeline. We already have trained an NLP model for the textual descriptions of our coins in the CN database (Gampe and Tolle in print 2019, Klinger et al. 2018). This model is able to find numerous entities like “Athena” or “Spear”. The list of trained entities includes four different categories: persons, objects, animals and plants. We wrote a query for grouping all coins with the same entities on the obverse and reverse, and sorted them by the frequency of these individual combinations. To create the new classes, we had to filter these combinations manually because a coin’s description can have several entities assigned to it and overlaps are possible. The example (fig. 7) shows all entities in a coin’s obverse and reverse description. For example, this coin can be assigned to the combination “Athena–Owl” and “Head–Palm

Branch”. Filtering is a time-consuming but mandatory step due to the large number of combinations. We also wanted to avoid new classes sharing coin images.

Figure 7 – Coin image descriptions and entities found by the NLP model. These descriptions are used for different coins and/or coin types (fig. 1), e.g. coins https://www.corpus-nummorum.eu/coins/53267?lg=en and https://www.corpus-nummorum.eu/coins/53408?lg=en [accessed 15 December 2023] for the Athena description shown here. (Graphic: S. Gampe, Big Data Lab)

After the filter step the new classes could be built from the remaining combinations. We assembled 16 new classes for Pergamon and Perinthos with different combinations of entities[4]. These classes contained mostly different types that share a similar appearance. The number of images for each class varied from 40 to 400 for Pergamon, and 40 to 300 for Perinthos. Using the NLP model, we were able to reduce the number of images in the original Pergamon and Perinthos classes to 1201 and 629. This is a significantly larger reduction than that with Object Detection. Compared to the original model, the values of the metrics after the training are only slightly lower (tab. 1).

However, as the confusion matrix shows, the new classes are again often confused with the original Pergamon and Perinthos classes (fig. 8). The overfitting of Pergamon was slightly reduced with this attempt, but the vertical line in the original Pergamon class fields (outside the new classes) is still visible. The reduction of the overfitting for Perinthos, on the other hand, is clearly visible. Due to the confusion of the old with the new classes both overfitting reductions had no influence on the metrics. We repeated this experiment again with a lower number of new classes (eight for Pergamon and nine for Perinthos). However, this had no positive effect on the overfitting problem for Pergamon. In fact, it became worse than before. The overfitting of Perinthos remained at the same level as in the first NLP attempt.

For this approach we can say that the NLP method required the most manual work of all three approaches. Our efficient NLP pipeline helped us best to separate the types that share a similar appearance for the new classes from the original classes. It also gave the best results for the overfitting problem based on the observation of the CM. However, the confusion between the new classes shows that the types of one mint very likely share some common features. This is something that could be further explored.

---

[4] The following entities were used for the combinations: 1. Pergamon: “Athena”, “laurel_wreath”, “owl”, “Augustus”, “crepidoma”, “bust”, “Asclepius”, “emperor”, “Telesphorus”, “cista”, “serpent”, “figure”, “head”, “bow”, “heads”, “temple”, “trophy”, “paludamentum”, “Zeus”. 2. Perinthos: “bust”, “ears_of_corn”, “Herakles”, “lyre”, “palm_branch”, “patera”, “torch”, “cuirass”, “apples”, “head”, “altar”, “athlete”, “club”, “Dionysos”, “horses”, patera, “Isis”, “apis”, “radiate_crown” “board”.

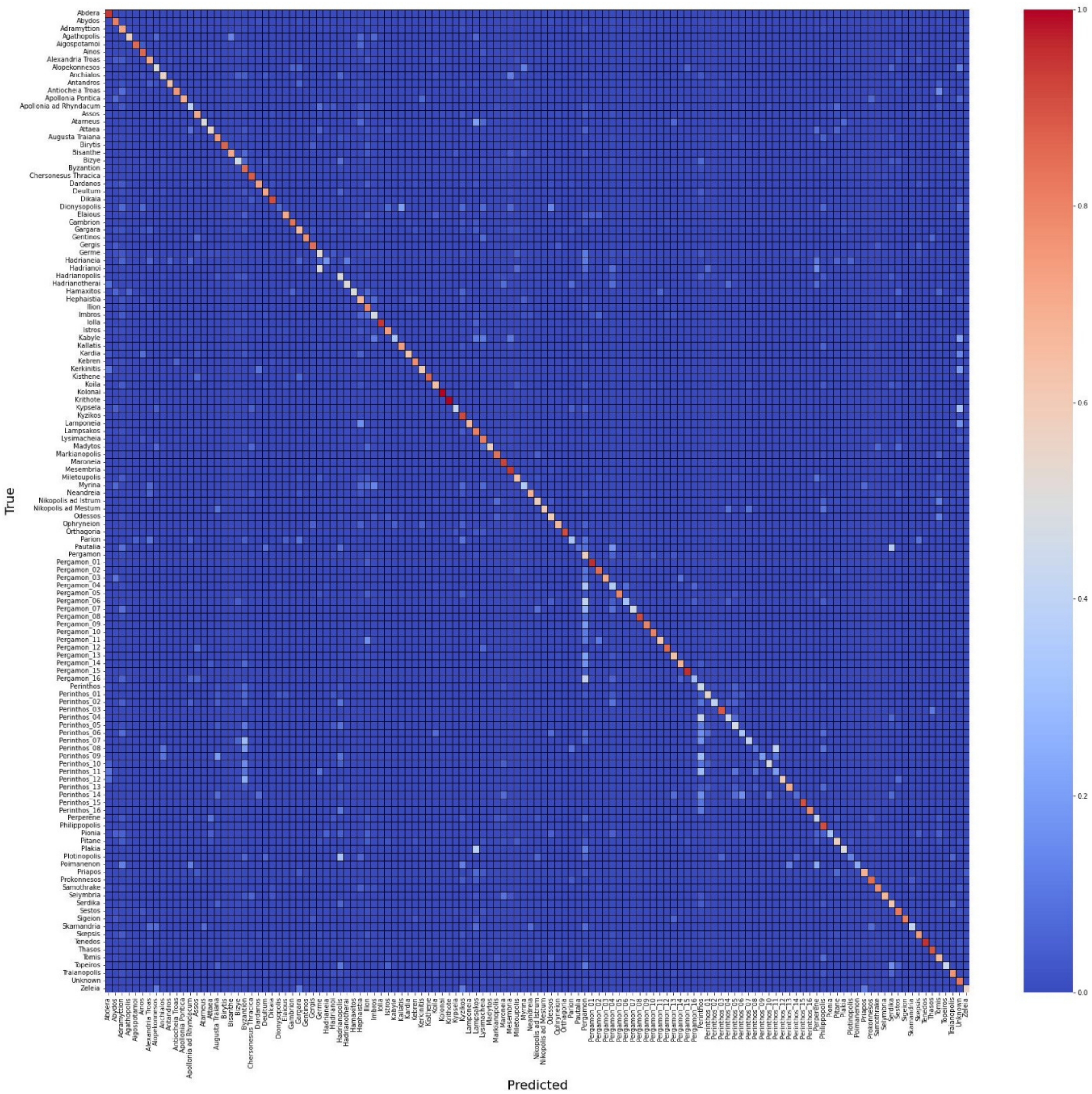

Figure 8 - - Confusion matrix for the NLP approach. (Graphic: S. Gampe, Big Data Lab)

Summary and Conclusion

Due to our limited dataset and the high number of types, a training on types with sometimes only two or three instances per type results in an unacceptable performance (34% Top-1 Accuracy). By switching to mints (accumulating all types of a mint to one training class), a very unbalanced dataset was generated with some mints (classes) dominating the training. While the performance of our trained models seems sufficient, we saw traces of overfitting for the dominating mints in the confusion matrix. The problem of unbalanced data sets is very common in the domain of numismatics and archaeology in general. The overall preservation quality of the coins we used was relatively good, especially compared to excavation finds where additional problems and uncertainties occur. However, our work could lay the basic foundation to expand the models also to less well preserved material. It is therefore important to solve problems before dealing with less optimal material.

We conducted three different machine learning based experiments to solve this overfitting problem with the unbalanced Corpus Nummorum dataset. To do this, we split two mint classes with a large number of images and created an additional class layer for them, and generated another training and test set with it. In our first attempt we created several new classes based on DeepCluster (unsupervised). The generated clusters contained too many different looking coin types (based on a human judgment) which negatively affected the learning process of our VGG16 model. In the second attempt the class layer was created with an Object Detection approach. This generated only a few smaller extra classes and the remaining coins without objects detected still formed a dominating class. It must be stressed that the Object Detection approach was only trained for some very common objects and the overall performance was still limited. Both approaches did not produce appropriate solutions for our problem. In the third approach classes were generated with Natural Language Processing. They were the most distinguishable from the original classes and this approach reduced overfitting the most. However, due the amount of manual work and the confusion of new and old classes, we are currently not following this path either. The accuracy of all newly trained models was below the original mint model. Even the observed visual reduction of the overfitting for the original Pergamon and Perinthos classes in the NLP experiment had no positive impact on model performance. This means that so far we could not solve our overfitting issue with mint prediction in a sufficient way.

These approaches might be useful methods in other cases, however, it shows that a generic approach to an overfitting due to dominant classes has not been found. We are therefore investigating additional ways to tackle this problem. This includes the creation of new coin images using ML-based methods for the individual classes. It can be seen as an augmentation approach for the smaller classes in order to reduce the domination of some huge classes. Furthermore, we started to compare other model approaches than just CNN (VGG16), like vision transformers or multimodal approaches.

Our next steps in the D4N4 project are:

The domain experts are trying to improve the CN dataset by including more images, especially for smaller classes. We are also working on a Generative Adversarial Network (GAN) approach to create virtual new coin images (that never existed) for those classes with very few coins. Finally, we published our CN dataset for other scientists and students to test their own ML methods, for example those that were recently part of a data challenge course at the Goethe-University: (Corpus Nummorum 2023).

Data, scripts, code, and supplementary information availability

Google Colab notebook for testing our type and mint model is available online: https://github.com/Frankfurt-BigDataLab/IR-on-coin-datasets

NLP pipeline code, models and a Google Colab Notebook for testing are available online: https://github.com/Frankfurt-BigDataLab/NLP-on-multilingual-coin-datasets

Conflict of interest disclosure

The authors declare that they comply with the PCI rule of having no financial conflicts of interest in relation to the content of the article.

Funding

The D4N4 project is funded by the Deutsche Forschungs Gemeinschaft (DFG) in the program “e-research-Technologien”.

References

Amidi, Afshine, Shervine Amidi. *Convolutional Neural Networks cheatsheet*. n.d. <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks> [accessed 8 August 2023]

Caron, Mathilde, Piotr Bojanowski, Armand Joulin, Matthijs Douze. 2018. *Deep Clustering for Unsupervised Learning of Visual Features*. <https://doi.org/10.48550/arXiv.1807.05520> [accessed 8 August 2023]

Corpus Nummorum. n.d. *Corpus Nummorum* <https://www.corpus-nummorum.eu> [accessed 8 August 2023]

Corpus Nummorum 2023. Corpus Nummorum - Coin Image Dataset. <https://zenodo.org/doi/10.5281/zenodo.10033992> [accessed 17 November 2023]

D4N^{4}. n.d. *Data quality for Numismatics based on Natural language processing and Neural Networks*. <http://www.bigdata.uni-frankfurt.de/d4n4> [accessed 8 August 2023]

focal-loss. n.d*. focal_loss.sparse_categorical_focal_loss*. <https://focal-loss.readthedocs.io/en/latest/generated/focal_loss.sparse_categorical_focal_loss.html> [accessed 8 August 2023]

Gampe, Sebastian. 2021. *Neuronale Netze zur Bestimmung römischer Kaiser auf Bildern antiker Münzen* (master thesis, Goethe-Universität Frankfurt a. M.). <http://www.bigdata.uni-frankfurt.de/wp-content/uploads/2022/05/Masterarbeit_Sebastian_Gampe_online.pdf> [accessed 8 August 2023]

Gampe, Sebastian, Karsten Tolle. in print 2019. Combination of Machine Learning Methods of Image and Natural Language Recognition of Ancient Coin Data, *Computer Applications & Quantitative Methods in Archeology (CAA), Proceedings of the conference in Krakow 2019*

GitHub - facebookresearch / deepcluster. n.d. *Deep Clustering for Unsupervised Learning of Visual Features*. <https://github.com/facebookresearch/deepcluster> [accessed 8 August 2023]

ImageNet. 2021. *ImageNet*. <https://www.image-net.org/update-mar-11-2021.php> [accessed 8 August 2023]

Keras. n.d. *Keras: Deep Learning for humans*. <https://keras.io> [accessed 8 August 2023]

Klinger, Patricia, Sebastian Gampe, Karsten Tolle, Ulrike Peter. 2018. Semantic Search based on Natural Language Processing: a Numismatic example. *Journal of Ancient History and Archaeology (JAHA)*, 5.3 68–79. <https://doi.org/10.14795/j.v5i3.334> [accessed 8 August 2023]

Long, Hui. 2023. *Klassifizierung von Motiven auf antiken Münzen mit Mask R-CNN* (master thesis, Goethe-Universität Frankfurt a. M.). <http://www.bigdata.uni-frankfurt.de/wp-content/uploads/2023/02/Masterthesis-Huy-Luong.pdf> [accessed 8 August 2023]

Selvaraju, Ramprasaath R., Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra. 2019. *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization*. <https://doi.org/10.1007/s11263-019-01228-7> [accessed 8 August 2023]

Girshick, Ross, Jeff Donahue, Trevor Darrell, Jitendra Malik. 2013. *Rich feature hierarchies for accurate object detection and semantic segmentation*. <https://doi.org/10.48550/arXiv.1311.2524> [accessed 8 August 2023]

scikit-learn. n.d. *sklearn.utils.class_weight.compute_class_weight*. <https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html> [accessed 8 August 2023]

spaCy. n.d. *spaCy: Industrial-strength Natural Language Processing in Python*. <https://spacy.io> [accessed 8 August 2023]

TensorFlow. n.d. *Tensorflow Object Detection API*. <https://github.com/tensorflow/models/tree/master/research/object_detection#tensorflow-object-detection-api> [accessed 8 August 2023]

Wirth, Alicia. 2021. *Einfluss von Bildannotationstechniken auf die Genauigkeit von Machine Learning-Modellen zur Objekterkennung* (research project, Goethe-Universität Frankfurt a. M.) <http://www.bigdata.uni-frankfurt.de/wp-content/uploads/2021/12/2021_FP_Alicia_online.pdfY> [accessed 18 November 2023]