applied sciences

MDPI

Article

Machine Learning Models for Artist Classification of Cultural Heritage Sketches

Gianina Chirosca [1], Roxana Rădvan [1], Silviu Mușat [1], Matei Pop [1] and Alecsandru Chirosca [2,*]

[1] National Institute for Research and Development in Optoelectronics, 409 Atomiștilor Str., 077125 Măgurele, Romania; gianina.chirosca@inoe.ro (G.C.); radvan@inoe.ro (R.R.); silviu.musat@inoe.ro (S.M.); matei.pop@inoe.ro (M.P.)
[2] Faculty of Physics, University of Bucharest, 405 Atomiștilor Str., 077125 Măgurele, Romania
* Correspondence: alecsandru.chirosca@unibuc.ro; Tel.: +40-72-122-0474

Abstract: Modern computer vision algorithms allow researchers and art historians to search for artist-characteristic contour extraction from sketches, thus providing accurate input for artwork analysis, for possible assignments and classifications, and also for the identification of the specific stylistic features. We approach this challenging task with three machine learning algorithms and evaluate their performance on a small collection of images from five distinct artists. These algorithms aim to find the most appropriate artist for a sketch (or a contour of a sketch), with promising results that have a higher level of confidence (around 92%). Models start from common Faster R-CNN architectures, reinforcement learning, and vector extraction tools. The proposed tool provides a base for future improvements to create a tool that aids artwork evaluators.

Keywords: sketch artist attribution; machine learning; vision-based algorithm evaluation; Faster R-CNN; reinforcement learning; vector database

Academic Editors: Cai Meng and Hongsheng He

Received: 31 October 2024
Revised: 1 December 2024
Accepted: 23 December 2024
Published: 30 December 2024

Citation: Chirosca, G.; Rădvan, R.; Mușat, S.; Pop, M.; Chirosca, A. Machine Learning Models for Artist Classification of Cultural Heritage Sketches. *Appl. Sci.* **2025**, *15*, 212. https://doi.org/10.3390/app15010212

1. Introduction

Art evaluation is a significant issue for art history and collectors. In the opinion of art experts, two of the vital problems in art authentication are artist attribution and forgery detection, which are part of the evaluation instruments for artworks [1]. If a proper detection is (mainly) successfully based upon physical and chemical characterization, the new attributions request more than that. A complex stylistic study must accompany these traditional artistic characterizations. With the growth of machine learning (ML) and artificial intelligence (AI) applications, we approach our artist attribution problem with specific, properly tuned award-gaining models such as VGG and ResNET. This detection algorithm allows us to implement flexible (error-permissive) algorithms that can look beyond the variations in data, thus making them appropriate for less specific fields, such as art-related scientific fields. AI-driven approaches use machine learning algorithms and image analysis techniques to address diverse challenges in analyzing original art [2]. Combining machine learning and image analysis, AI methods offer innovative solutions for analyzing original art. Such an approach allows human experts to gain objective, impartial insight, thus making the analysis results more relevant. Traceability issues often result in missing and incomplete ownership history, make the authentication task more difficult for older artworks, and make forgeries harder to identify. The art world is susceptible to forgeries, making it crucial to identify fakes and protect the value of genuine pieces [3].

Due to the improvements in forgery techniques, art authentication has proved to be a key factor with much attention, mainly due to digital image processing, forensics,

and even legal cases. While publicly available artwork databases often exhibit deficiencies, our model was trained on images of drawings from private and public collections that were previously characterized by verified authenticity and comprehensive ownership history. This approach allowed us to concentrate on a more curated dataset. ML/AI models require a large dataset for training, testing, and validation, and have to deal with a large variety of features to analyze for artist discrimination within the same art school. We focused on stylistic elements such as brushstrokes and compositional patterns to encode each artist's style, thus providing a way to discriminate among them. While ML/AI models are pure nonlinear statistical models, the resulting model provides us with an artist-style classification technique; thus, it can flag potential forgeries that deviate from the learned style, providing valuable input data and highlighting points of analysis for subjective or sometimes biased evaluators.

The proposed method can be improved but is not presented as a panacea for all attribution topics. However, it is a filter that can potentiate the importance of physical–chemical determinations or guide in-depth research—for example, through multi- or hyper-spectral imaging techniques or non-intrusive spectroscopic techniques for critical areas/points.

A few years ago, the scattering transform was proposed for artist authentication in art images, and the analysis of selected entities was carried out to reveal the importance of deep layers using scattered, transformed, and rare linear classifiers for the authentication of art [4]. This research continued with a systematic analysis of machine learning methods for software testing [5] and different algorithms [6], as well as methods of authentication of paintings, based on the fusion of multiscale spatial–spectral entities and convolutional neural networks [7]. While most of the works included polychromies (paintings) with great results [8], we focused on images with a less critical volume of data information density, such as sketches, using a traditional approach.

In this study, we developed three models for recognizing the strokes of some painters using different machine learning algorithms. All three models used the same dataset, in which images were preprocessed to eliminate paper-induced elements that would affect the performance, thus allowing the model to focus primarily on pen strokes. We propose the combination of the models Vector Embedding Similarity Search (VESS), Faster R-CNN, and reinforcement learning, suggesting an approach to a complex task that requires both accurate object detection and an effective similarity search. These ML models provides an objective view to enhance the accuracy of artist identification, especially when combined with valuable human expertise.

The exclusive choice of artist drawings—not paintings, watercolors, or other creations—was not accidental. The color palettes and gradients need to be assessed using specialized attention models; this was the reason for our simpler approach. To correctly assess the accuracy of the proposed method, elements that can be disruptive, such as the stroke (random brush marks, color diffusion in wet layers, etc.), were filtered using standard (traditional) filters before the model inference was conducted.

2. Material and Methods

Our workflow for this research started with an initial dataset evaluation and data cleansing, followed by a data preparation step in which input data were prepared for consumption within the training flow. After data preparation, we performed model training and meta-parameter tuning for this specific task and then evaluated the results. This evaluation provided valuable input for the meta-parameter estimation. We kept this cycle until either the results were statistically valuable or we identified that our model did not converge with the estimate and redesigned the model. For the workflow implementation, we considered an ML/AI model based on the TensorFlow [9] framework version 2.16.1

using the Python [10] language version 3 [11,12] for compatibility reasons. The data preprocessing framework was implemented in the Jupyter Notebook [13] environment for rapid analysis. For the reinforcement learning [14] we switched the framework towards Pytorch [15] version 2.3.1.

2.1. Dataset Structure

Often, issues in machine learning applications are related to the available datasets. The dataset is important when dealing with model convergence issues. Unfortunately, in our case, the available (authenticated) dataset was heavily unbalanced [16], and the standard models tended to be biased toward the most generic visual elements (frames, straight lines, etc.). In Table 1, we provide an overview of the artwork elements within the original dataset.

Table 1. The original dataset anatomy.

Nr. Crt	Title	Element Count
1	Alexandru Rădvan	189
2	Corneliu Baba	26
3	Nicolae Tonitza [1]	55
4	Pablo Picasso	100
5	Reszegh Botond [1]	40
6	Sorin Ilfoveanu	127
7	William Kentridge	76

[1] Some artist information was taken into account only for the Faster R-CNN and embeddings models.

One of the biggest challenges in our approach to this problem was related to the fact that graphical elements often cover different orders of magnitude in the geometrical sense (from a few centimeters to meters or tens of meters, and the subjective features are always related to the parts that the artist can draw once without raising the pen). Figure 1 contains specific works for each author considered within this work.

(**a**) (**b**) (**c**) (**d**) (**e**) (**f**)

Figure 1. Artist characteristic stroke: (**a**) Alexandru Radvan. (**b**) Corneliu Baba. (**c**) Nicolae Tonitza. (**d**) Pablo Picasso. (**e**) Sorin Ilfoveanu. (**f**) William Kentridge.

Such complex problems are often distorted by a high statistical variety, and common deterministic (DETMOD) or statistical models (STATMOD) often fail at this task, as they cannot correctly identify and process subtleties and do not scale well with the input data. To overcome this issue, we applied modern machine learning (ML)/deep learning (DL) algorithms, which are more permissive in terms of scaling variations and errors (damage, missing pieces, etc.) in identification.

The images (with inconsistent sizes) were split into patches of 512 by 512 pixels with minimum overlap between them, allowing us to extend the dataset before model training and evaluation. After we expanded the dataset, we performed dataset down-sampling (by reducing the number of images to that in the smaller set) to balance it. Training, testing, and validation were performed on whole images, while the QLearning and Faster R-CNN

models required labeling (this labeling was performed using Canny algorithms). The final dataset representation is shown in Table 2.

Table 2. The image count in the original dataset.

Nr. Crt	Usage	Image Count	Disk Size [Mb]	Percentage [%]
1	Training	772	271.53	72
2	Testing	196	71.31	18
3	Validation	96	34.09	10

Especially in classification problems (such as the problem at hand), unbalanced datasets can provide inaccurate results and even model non-convergence issues, but, in our case, the initial files were processed with a primary attention mechanism [17] that was focused on edges. Thus, there was a need to prepare images with various degradation stages with a specific filtering mechanism to enhance only the positive edges (not those from cracks, paper discoloration, or degradation).

2.2. Image Preprocessing

While our dataset consisted of only sketches, several processing steps were implemented for optimal results (by filtering out degradation) and to enhance the overall image contrast. The data processing framework was implemented using a tabular format with standard Python processing tools [18,19]. Once a pointer reference file (csv) was created for each file, we started the processing flow with a thread-intensive workload. This workload was based on the popular OpenCV [20] library, which first loaded the file and normalized the arrays, leading to standard Numpy [21], which could be easily ingested by our model. While most available algorithms limit themselves to these steps before the model training phase, in our case, the degradation required special attention for the discrimination of the background from the actual image. To overcome this issue, in our case—where the sketches had a limited color palette—we identified this as being an application of classical principal component analysis (PCA) [22] for two classes (content and background) and 10 iterations. As our attention was driven toward the edges, we converted the images into gray-scale and dilated them (by applying a maximum filter to enhance the filtered edges) to overcome the sharpening specific to the PCA step. While some architectures, like the Faster R-CNN one, usually do not require such preprocessing, in our case, the model can be misled by the straight edges within the existing image (e.g., for Kentridge) resulting in wrong identification. Beside these situations, by ensuring a clean input image, all models interpret the data thus making the model comparison unbiased.

The next step, required by Faster R-CNN, was the performance of image segmentation, which involved a technique that extracted pen/pencil strokes from the actual images in order to correctly draw the model attention to them. To obtain optimal results, we employed the Canny Edge Detection algorithm [23] due to its good and reliable edge and contour detection (areas from within the normalized image where the local gradient reached its maximum value). For specific areas, the Canny Edge Detection algorithm employed non-maximum suppression to thin out the edges and compute a gradient strength with a double threshold for edge classification (weak and strong). At the same time, during the last step, it used a hysteresis threshold, thus highlighting only the strongest edges.

The resulting edges were then filtered out using a minimal area of 50 pixels (px), thus further excluding potential noise. This technique allowed the model to focus mainly on significant strokes (with actual artist-specific features).

These extracted features were exported using a Pascal VOC [24] file format. This type of annotated file allowed for the generation of testing, training, and validation datasets using the common TensorFlow Data (tfrecords) format. We used this format due to its

versatility and non-sequential access, allowing for faster training and evaluation workflows. These files are extensively used for standard regional convolutional neural network models.

Upon searching for the best way to accommodate the dataset with our hardware capabilities, the best approach for our dataset was to enforce an image size with a shape of (512, 512, 3), meaning that all images were 512 × 512 pixels and had three color layers (blue, green, and red). While images with better resolutions often had specificities in terms of their size and orientation, many of them had dimensions that exceeded this limitation. Our approach was to split the images (without losing pixels) into multiple images. Figure 2 provides an example of an image with dimensions of 1200 × 848 and the resulting images. The image is split into red, black, green, blue, yellow, and navy squares (the misalignment of the picture is for representation purposes) resulting in 6 images of 512 × 512 pixels.

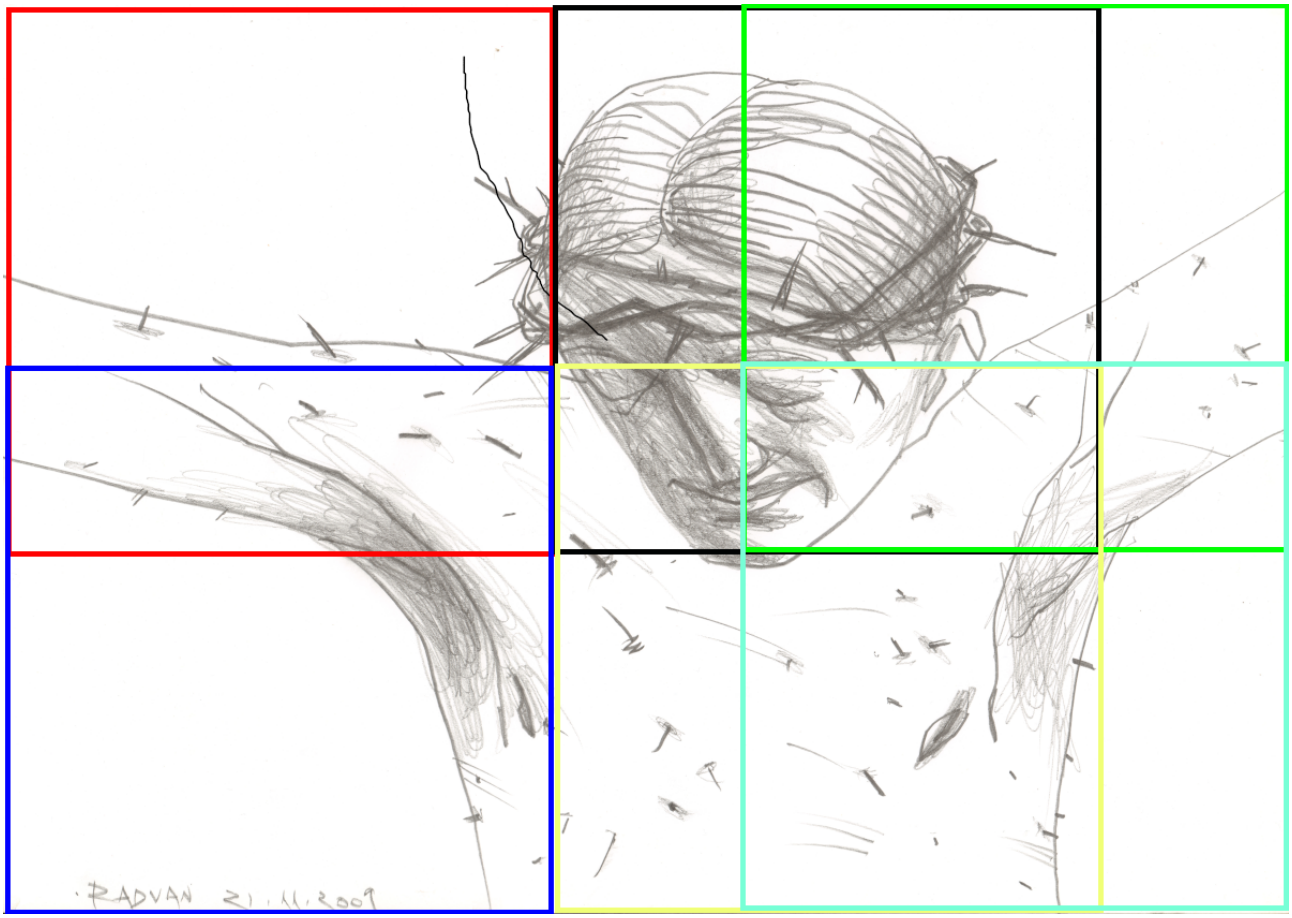


Figure 2. Description of the image split algorithm for an image with a shape of size (1200, 848, 3). The first window presented in red contains the pixels from (0, 0) to (512, 512), the second window highlighted in black contains the pixels from (512, 0) to (1024, 512) while the last window on the first row—highlighted in green—contains the pixels from (688, 0) to (1200, 512). For the second row, the image height leads to an overlap between rows. The first window—highlighted in blue—contains pixels from (0, 336) to (848, 512), the next window highlighted in yellow contains pixels from (512, 336) to (848, 1024) while the last window, highlighted in light blue, contains pixels from (688, 336) to (1200, 848).

The data were prepared in three stages: image filtering, noise filtering, and image segmentation. Image filtering is focused on providing single images for each picture (in the case where two or more sketches are present within the original image—around 2% of the original dataset) ensuring that for each picture only one sketch is present. This is a manual operation that extracts each sketch and ensures that the resulting image sizes are larger than 512 × 512 pixels. Smaller images were excluded from the original dataset.

The second stage is focused on the image quality and this is where noise (like paper degradation) is filtered resulting in high-contrast images suitable for the training process. After this step, we can perform image segmentation (in smaller 512 × 512 × 3 images). In this step, the images are augmented with PascalVOC annotation later used for the Faster R-CNN model.

2.3. Model Development

Our proposed models employ, using different methodologies, standard convolutional neural networks (CNNs) [25] that are traditionally used to extract features from these proposed regions using an established or pre-trained network topology. These features captured the characteristics of the objects. They are implemented on previous topologies that excelled at extracting—like VGG16 [26] high-level features from images, allowing for accurate object classification and localization.

Artwork authentication is one of the highest-interest topics in our digital age, and previous attempts in the literature found several weaknesses in standard approaches like R-CNN [27,28]. We focused on the data preparation step and supportive input from an expert advisory team when searching for novel implementations while keeping the standard model architectures in mind. This led us to the implementation of the following model types.

2.3.1. Reinforcement Learning Model Architecture

Each person has a set of personalized, characteristic stroke patterns in their artwork thus our approach for authorship suggestion algorithm. These provide several implementations with distinct approaches. At this stage, we are implementing a naive classifier that employs reinforcement learning [14] using the PyTorch [15] framework with the agent-based (QLearning) penalty/reward system. The overall model architecture uses 5 VGG type structures for extracting features chained to a 3-layer dense network that performs the artist attribution (classification). We found this type of architecture is appropriate to our needs while focusing on sketches where gradients are generally high and monochromatic. Reinforcement learning requires an agent to check the training results for output validity and produce rewards or penalties according to the expected classification. In our case, the implementation uses a simple architecture based upon its implementation details (see Table 3 numbers 6 to 11). While the actual classes are taken from the *tfrecords* files, most of the agent implementations are kept simple and this regulates the training for the actual network. While we are using the QLearning approach, there is no mechanism to address model attention and, in such cases, the only approach towards debugging the model is with the implementation of model interpretability [29] that was performed using the Captum library [30]. The training parameters are presented in Table 3.

Table 3. Reinforcement (QLearning) model parameters.

Nr. Crt	Name	Value	Nr. Crt	Name	Value
1	Epoch count	900	6	Gamma	0.99
2	Batch size	16	7	Epsilon	1.0
3	Learning rate	10^{-5}	8	Epsilon decay	0.995
4	Image shape	(512, 512, 3)	9	Min. epsilon	0.01
5	Classes	7	10	Agent optimizer	Adam
			11	Agent loss function	CrossEntropy

The Captum library allows for insights for the inference stage by representing the attribution matrix—presented as an activation map—that shows both positive (where the

element provides an increase in artist attribution) and negative (where that specific element provides a decrease)—see Figure 3.

Model training performance was estimated using the confusion matrix evaluation while the training process was optimized using decay algorithms and checkpoint saving. The confusion matrix (see Figure 4a) allowed for training model performance indication while (Figure 4b) presents the same results for model validation.

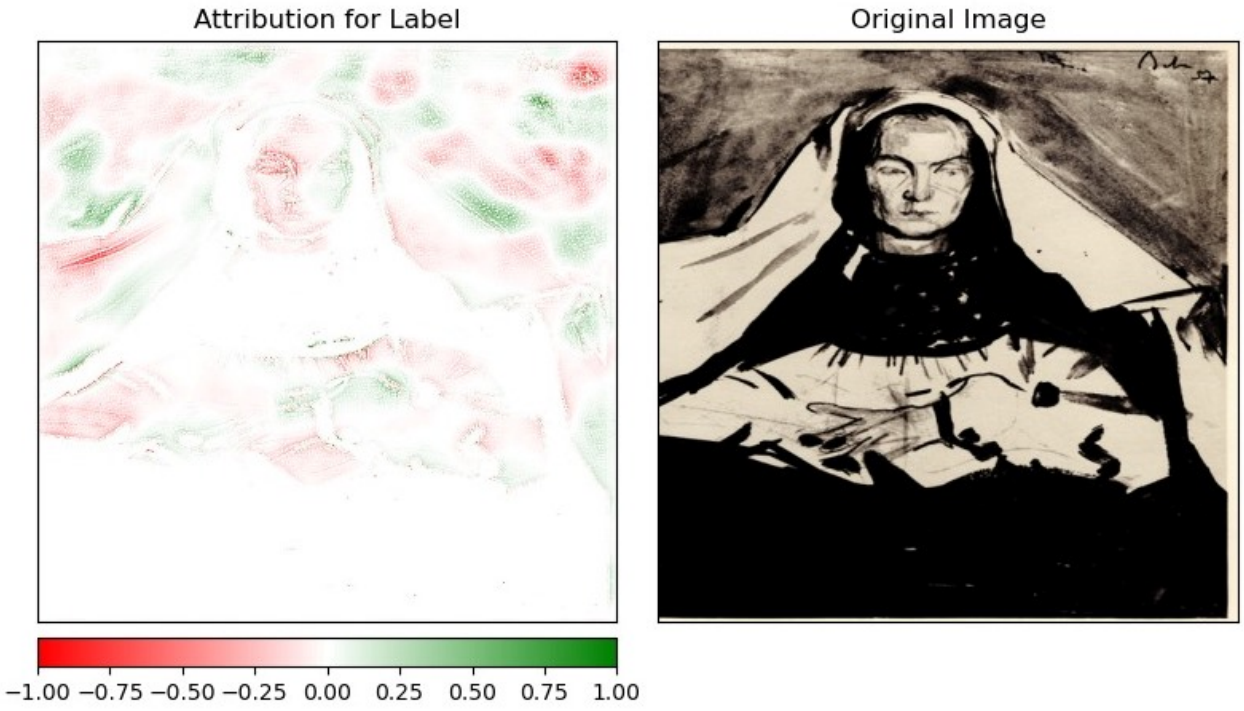
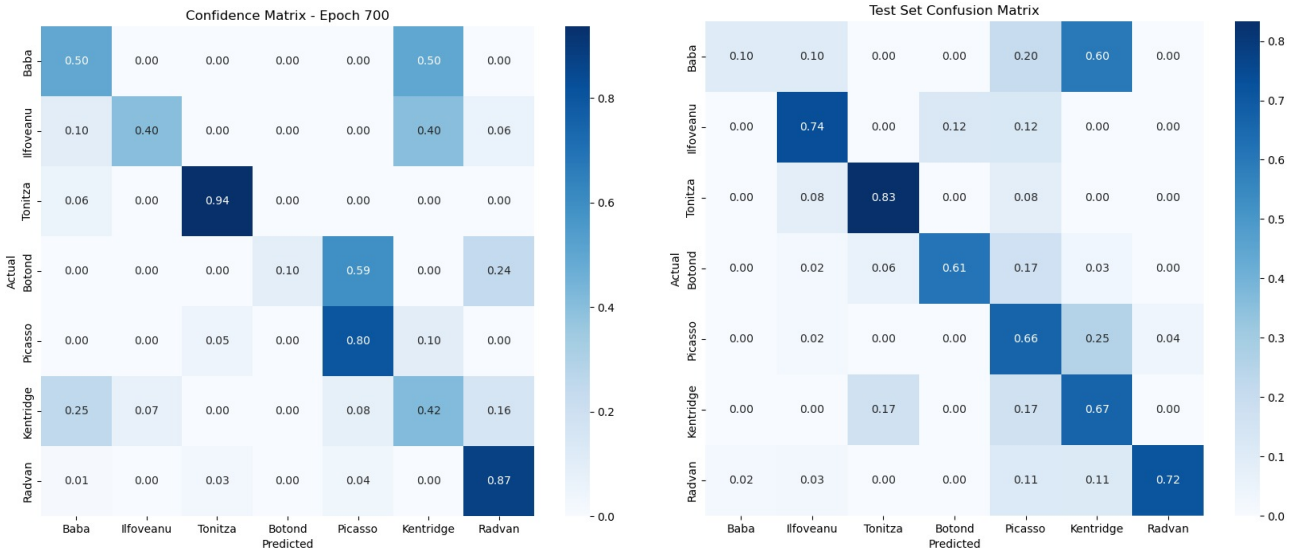


Figure 3. Captum artist attribution map.



(**a**) (**b**)

Figure 4. (**a**) Reinforcement learning (QLearning) training confusion matrix. (**b**) Reinforcement learning (QLearning) test confusion matrix.

While this approach is common to modern problems, reinforcement learning (QLearning) algorithms provide us with less control over the activation maps thus requiring more epochs to provide accurate results. The main benefit of implementing such an algorithm, at least in our case, was to perform Machine Learning Quality Assurance upon the dataset,

and with the provided confusion matrix we found out that using subsampling is not enough to balance the dataset; we also reduced the number of unrepresented classes from 7 to 5 for future models.

2.3.2. Faster R-CNN Approach

Faster R-CNN is a deep learning model architecture used for object detection in images and videos. It builds upon the foundation of earlier models, such as R-CNN (regions with CNNs), by addressing their limitations and achieving faster processing speeds.

As the R-CNN approach required much work and customization, we needed to evaluate models with faster convergence and less customization to provide a faster time-to-convergence ratio. To reach our goal, we implemented the Faster R-CNN architecture as an object detector that focuses on strokes and suggests artist attribution. In order to present accurate results we do not take into account overlapping between detections but rather provide information for the evaluator about each stroke attribution leaving the decision to the human operator. This requirement led us to the more rapid implementation of pre-trained models using the TensorFlow model hub, where the Faster R-CNN implementation is predominant. This type of network was developed only two years after the proposal of R-CNN and provided better results for both the RPN and the convolutional part. The first proposal was made by Ren Shaoquing in 2015 [31]. We employed a residual network [32] architecture with a Faster R-CNN implementation for this study.

For this model, we used the *retinanet-resnet f pn-coco* pre-trained model, which used the ResNet50 architecture [33]) with some customizations in terms of training parameters. We set the training to be split into training batches of 100 images, with a learning rate starting at 0.01 and a warmup of 0.05 (the warmup size was also 100 images). The learning rate type was *cosine*. The protobuf reader was used with a buffer of 20 images, thus reducing the latency for network calls. This model used the COCO annotation type, thus requiring the implementation of a converter function for the PascalVOC annotation file.

The main benefit of using such pre-trained models is related to the overall accuracy and model optimization (such as the feature pyramidal network presented in the RetinaNet above the Resnet50 architecture). While such improvements are beneficial to most models, for our specific task, a special selection of algorithms suitable for the task at hand was needed before finding a better model. While using a specialized pre-trained model, the training steps made extensive use of checkpointing and resulted in a run that was not verbose.

This model focused on the recognition of painters' pen strokes using edge detection, a technique for image segmentation and data extraction, with the following parameters: learning rate, 0.004; batch size, 4; hparam optimizer, Adadelta; 272 epochs or 151 epochs. This approach applies modern pre-trained models (thus making use of large training times and infrastructure provided by the model developers, enabling fewer training epochs) for feature extraction, allowing for faster image processing using the COCO tags extracted from previously generated PascalVOC files. This greatly reduced the training time (concerning reinforcement learning—QLearning). The limitations of this model were given by the existing hardware infrastructure, which quickly exhausted the available computational resources for more accurate results. The results were excellent, with an accuracy of **95.84%** when analyzing 1018 images belonging to five classes. The results can be found in Table 4. From the original datasets, several images were filtered out (from a total of 1064 images, 46 were discarded from the training, testing, and validation sets) due to mixed compositions (existing noise within the images).

Table 4. Experimental results for the Faster R-CNN architecture (hyperparameter search).

Epochs	Learning Rate	Images	Classes	Val-Accuracy
272	0.004	1018	5	79.27%
272	0.004	1018	5	81.10%
272	0.004	1018	5	**83.23%**
151	0.004	1018	5	76.41%
151	0.004	1018	5	81.10%
151	0.004	1018	5	85.52%
151	0.004	1018	5	93.36%
151	0.004	1018	5	**95.84%**

The training process required fewer training steps than that of the reinforcement learning model and had greater accuracy. The training and testing parameters are presented in Table 5.

Table 5. Experimental results for the Faster R-CNN model training and evaluation for the second sample from Table 4.

Name	Training Value	Validation Value
Loss	0.1749	0.2107
Categorical Accuracy	0.8336	0.8109
Precision	0.4449	0.4382
Recall	0.8464	0.8466
Auc	0.8492	0.8463

2.3.3. Vector Embedding Similarity Search

With the extensive application of large language models (LLMs), knowledge representation is quickly evolving, addressing specific requirements that are not available for traditional encoders (such as the one-hot encoder). One such solution is employing a specific database type, namely, a vector database [34]. These vectors provide an abstraction from the activation maps given by the neural network processing an image. This allows us to map artist-specific features to such vectors and, with this approach, we will provide similarity search capabilities to our database.

Our approach used the standard award-winning VGG16 [26] database, which was chosen especially for its simplicity in terms of human understanding, allowing us to evaluate the results of the encoder model. For this special case, an image resizing operation was performed for the shape (512, 512, 3) to correctly map the information within a vector with a shape of (1, 512) as required by the VGG16 model.

This workflow did not rely on classical accuracy training and loss but used activation mapping for vectors that were named embeddings, enabling metric extraction between the activation maps provided by the CNN while looking at patterns. While the image preprocessing workflow enhanced the gradients while reducing the background noise and the color map, this approach focused on pen strokes.

The IFV_FLAT index was applied for floating point values (provided by the encoder package) to divide the data into the destination tensor containing all data from the VGG activation maps.

$$L2(a,b) = L2(b,a) = \sqrt{\sum_{i=0}^{n} -1(b_i - a_i)^2} \; \forall \; a = (a_0..a_n); \; b = (b_0..b_n) \tag{1}$$

The actual database implementation used a virtualized instance of Milvus [35] without GPU capabilities (hence the choice of indexes in Table 6) to handle communication using

tcp sockets with ML engineering workstations. The image processing workflow included image-encoding algorithms from the Towhee package [36]. The prepared dataset was fed through the Towhee pipeline, and the vectors were included within the database. This processing stage required ML acceleration, and the GPU was used in our approach.

Table 6. Vector database parameters.

Nr. Crt	Name	Value	Description
1	INDEX	IFV_FLAT	A simple, full-copy index type
2	METRIC	L2	This is the vector-space Euclidean distance (Equation (1))
3	ENCODER	VGG16	Image decoder algorithm
4	TOPK	10	Max number of items returned from the similarity search

This model analyzed the recognition of painters' pen strokes using vector embeddings. Vector embeddings are a powerful technique for representing images as numerical vectors. These vectors capture the essence of an image, allowing for tasks such as image retrieval, similarity search, and image classification. The following is how one can create image embeddings using the VGG16 model. VGG16 is a convolutional neural network (CNN) that is pre-trained on a massive dataset of images (ImageNet) for image classification. It has 16 convolutional layers and fully connected layers that predict image categories. VGG16 excels at extracting high-level features from images, making it a good starting point for creating embeddings. Before feeding the model images, we processed them to ensure consistency. This usually involved resizing images to a certain size, converting them into a specific color format (for example, BGR for VGG16), and potentially normalizing pixel values. We were able to fine-tune the VGG16 model on our specific dataset to potentially improve the quality of extracted embeddings. This involved retraining the final layers of the model with our data. These newer techniques removed the need for encoders for classification problems by storing the vectors containing the activation maps directly. This mapped the activation space within a multidimensional space to allow for classical metric distance evaluation, thus allowing us to turn the encoder into a purely analytical function. Other more complex statistical methods may be implemented, but our choice was to use the standard metrics, as no CUDA cores were available on the virtual machine in the vector database.

2.4. Hardware Infrastructructure

ML data processing pipelines require specialized hardware and software environments, as well as engineers who are capable of analyzing the high throughput of data. A dedicated team of engineers and the infrastructure presented in Table 7 were necessary to implement the workflows discussed in this work.

Table 7. Hardware infrastructure.

Type	Count	Name	CPU	Cores	RAM [1]	GPU	GRAM [1]
Baremetal	3	Workstation	Intel i7	8	32	A4000	16
Virtualized	1	Vector Db	x86-64-v2-AES	8	8	-	-

[1] Memory is expressed in Gb for both the system and GPU (GRAM).

The frameworks employed for this study had specific requirements—especially TensorFlow [9] v 2.16, which requires a Linux operating system to use the CUDA hardware accelerator within the Nvidia A4000 GPU card (Nvidia, Santa Clara, CA, USA). In our case, all operating systems ran Windows 11 Professional with the Windows Subsystem for Linux, thus enabling CUDA hardware. All workstations benefited from shared object storage (S3) for faster file access (protobuf access to tfrecords data [37]).

CUDA was chosen for its ability to harness the power of GPUs. Their exceptional performance in handling numerous calculations simultaneously provided substantial speedups for our image processing.

3. Results

All model performance evaluations dealt with the support of modern machine learning frameworks like TensorFlow [9] or Pytorch [15], where the dataset isolated the training set from the testing and validation sets. A single evaluation step was considered for each of the models while computing the *validation loss* according to the model's metrics. While the training dataset consisted of most of the images (72%), the test data consisted of previously unseen data (18%), and the validation set (an automated step during the training) had a smaller volume (10%) that was also not previously seen during training (although some recommendations from the TensorFlow community state that the validation set should be a subset of the training data). The results presented in Tables 4 and 6 show acceptable loss values for the training data. The training steps provided valuable input for our pipeline development process, leading to data rebalancing using the sub-sampling methodology while removing the smaller undersampled artists (from seven to five).

The first model was developed using the training and testing data while the validation was performed using the provided confusion matrix for each of the datasets (see Figure 4).

The model performance was estimated using the validation set (during the testing step—Figure 4b) not previously seen by the model. Global accuracy was estimated using a simple average for all classes resulting in the accuracy results for this reinforcement learning model.

Model attribution maps were manually evaluated for each of the test dataset where Figure 5a shows the best evaluation and Figure 5b shows the worst evaluation. The attribution maps provide both positive and negative attributions for each region within the CNN activation maps (red attributions are negative while green attributions are positive) and are explained with the heatmap legend below each evaluation.

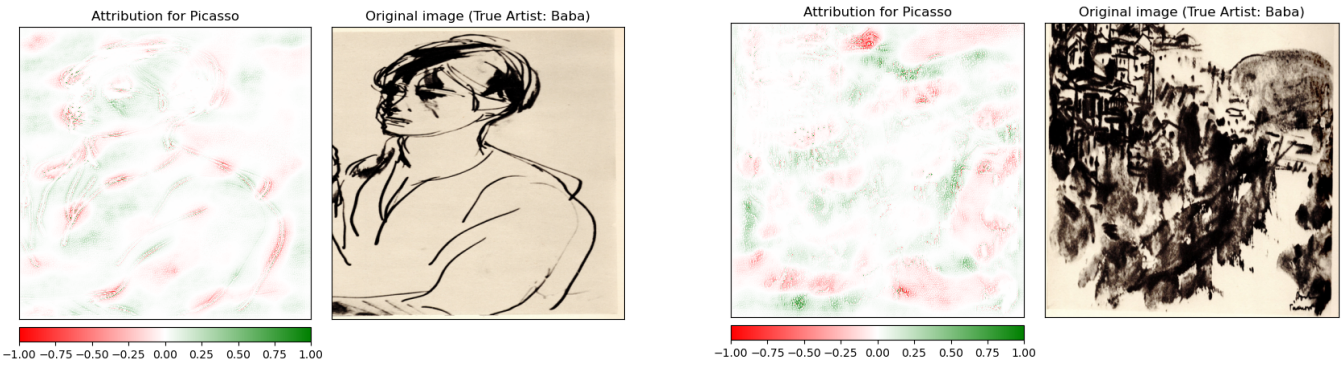


Figure 5. (**a**) Best attribution by reinforcement learning model. (**b**) Worst attribution by reinforcement learning model leading to misclassification (incorrect artist attribution, a Baba work being wrongly classified as Picasso).

Our next step was to implement a more specialized pre-trained model based on Resnet50 using the Faster R-CNN architecture. This model was the most interesting of the classical model architectures that required short development iterations (only three) with five artists (classes) determined by our previous experiment. While the batch size was 4, using 1018 images (split using the preprocessing stage) resulted in a validation accuracy of **95.85%**; the sigmoid activation function was kept, but the optimizer was changed to *Adadelta*, and the optimal learning rate was found to be 410^3. This model provided better

results than those of the reinforcement learning one, with better feature detection and fewer missing strokes (see Figures 6 and 7).

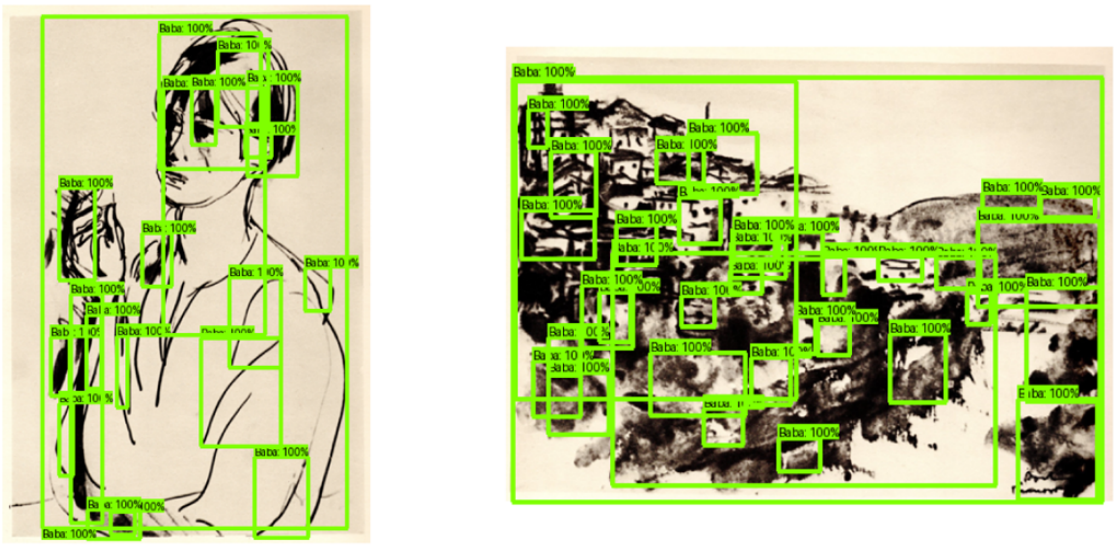


Figure 6. Image analyzed by Faster R-CNN model using input images with known attribution to Corneliu Baba as input for all three images. This model presented in top 50 detected regions the correct artist attribution.

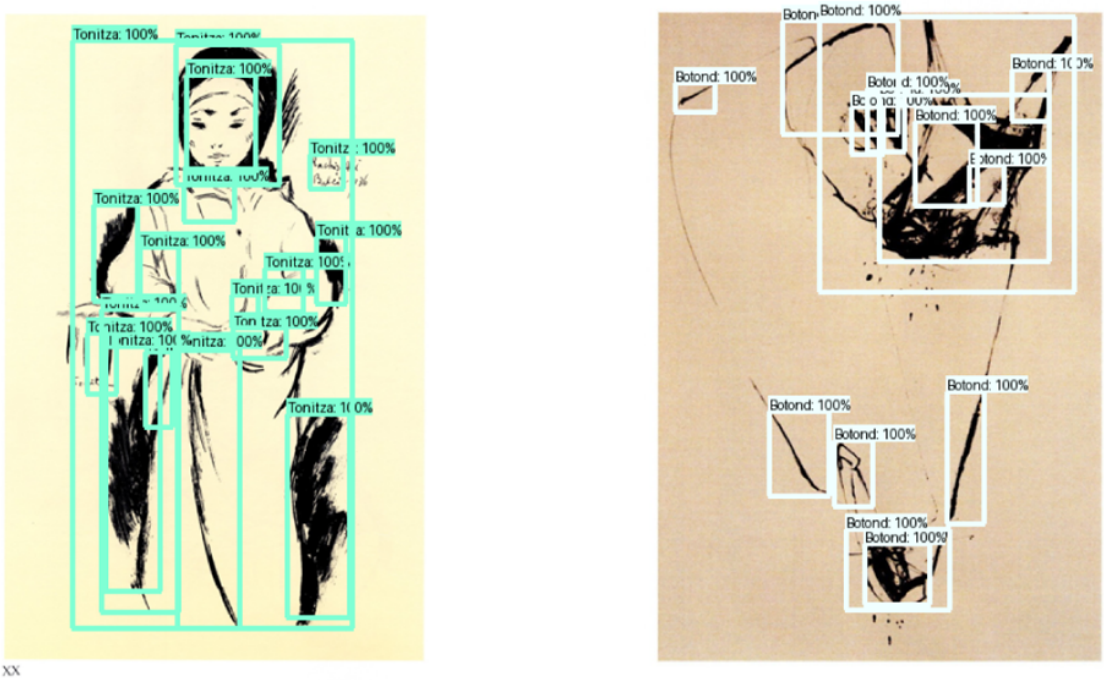


Figure 7. Image generated by applying our Faster R-CNN based model using known images provided by Nicolae Tonitza (**first image**) and Reszegh Botond (**second image**) as input.

Utilizing a sigmoid activation function for a neuron ensures that its output is always confined between 0 and 1. Moreover, due to the sigmoid's nonlinear nature, the neuron's output will be a nonlinear transformation of the weighted sum of its inputs.

The Adadelta optimizer is an improved version of Adagrad that is designed to counteract the problem of rapidly decreasing learning rates. Unlike Adagrad, which accumulates all past squared gradients, Adadelta limits its focus to a specific window of recent gradients. This is achieved by calculating an exponentially weighted average of the gradients, effectively giving more weight to recent updates and less to older ones.

Vector embedding databases are usually applied for recommendation systems and LLM knowledge representation. In our case, we applied the first technique for the 10 (TOPK) most similar known sketches to allow for easier confusion matrix extraction with weighted classification. In this approach, we applied an L2 (Euclidean) distance search on the vectors extracted using the plain pre-trained VGG-16 CNN model without any training. This approach provided us with higher levels of accuracy, even for difficult sketches, ranging between 92% and 98% in our experiments.

The query presented in Figure 8 for one image not seen by the training database provided a response structure containing a reference image (the image found to be similar), the distance (the numerical value employed for the weighted average), and the artist name for each of the 10 results returned. The image to be analyzed had to be sent to the Vector Database API endpoint as a numerical array of pixel intensities and required no local processing.

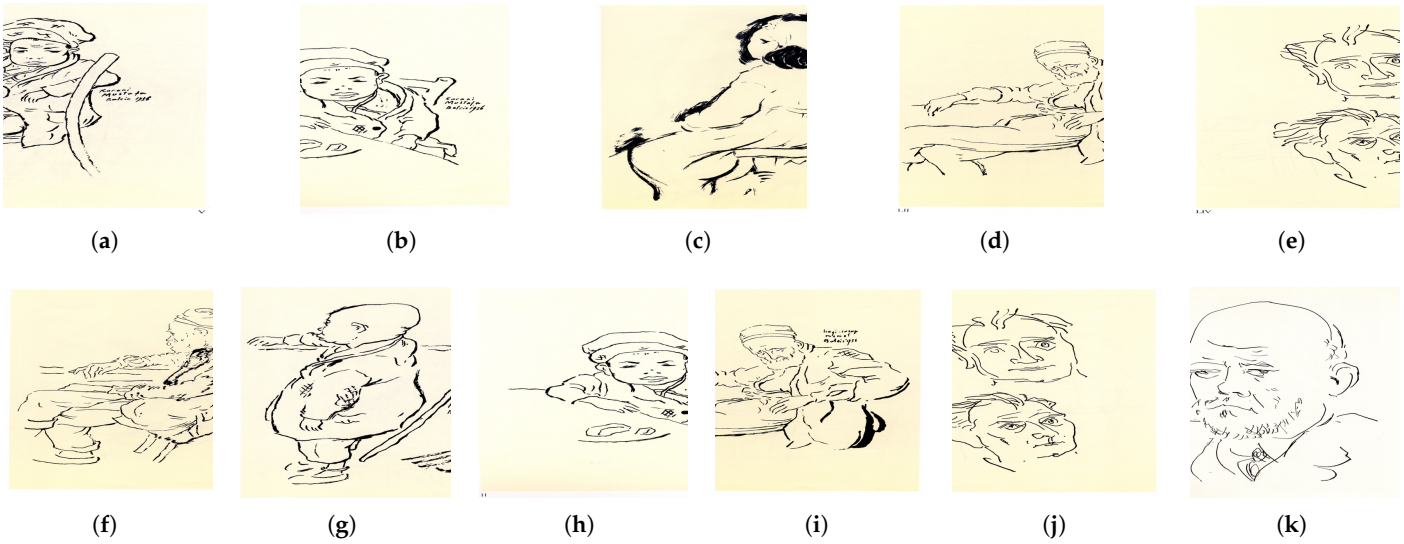


Figure 8. Query of the embedding database to search for one image. Results are presented as artist (distance): (**a**) Query artist Tonitza. (**b**) Tonitza (0.161). (**c**) Tonitza (0.171). (**d**) Tonitza (0.186). (**e**) Tonitza (0.196). (**f**) Tonitza (0.198). (**g**) Tonitza (0.205). (**h**) Tonitza (0.215). (**i**) Tonitza (0.221). (**j**) Tonitza (0.239). (**k**) Picasso (0.252).

While the vector database consisted only of training data with labels providing the artist information, we estimated the results for queries performed with test or validation data (not recorded within the database) and estimated the accuracy of the results by counting the elements within the TOPK returned values that were correct concerning the known artist and divided them by the TOPK. A weighting algorithm based on the metrics needs to be further investigated.

$$acc(n) = \sum_{i=0}^{n} \frac{results - in - class}{TOPK}$$

One of the biggest issues with such models is related (at least for our setup) to the attention mechanism's tuning.

Values close to 97% were reported for the model accuracy with the above calculations.

The different architectures provided within our implementations usually provide better accuracy for any type of input image allowing them to be incorporated within the artwork evaluator aiding software source thus ensuring an objective view of the artwork to be evaluated.

4. Discussion

Artist attribution suggestions for artwork sketches have proven to be a challenging problem even for modern computer vision machine learning models. To address such issues we have developed a flow for previously authenticated sketches that first clean up the images from artifacts and image degradation allowing for models to focus on the artist's drawings and not on degradations. Then, we started implementing our analysis system that contains three machine learning models with different architectures. The first architecture, using reinforcement learning with deep convolution networks, we focused on the award winning VGG topology. This approach applied agent learning or QLearning for the classification issue allowing the deep network to accommodate to our problem at hand. The results were quite promising allowing for acceptable accuracy values for a suggestion system. As the system evolved under the agent's pressure there was a specific issue related to model interpretability and this led us towards using the Captum library for interpretable machine learning.

The next Faster R-CNN architecture considered for our work is the pre-trained Residual Network (ResNET) model with COCO annotations. Such models provided the best results for classification problems but required extra training steps. The first step is to create the annotations for each image. Our initial approach was to annotate using custom algorithms using the PascalVOC annotation style but later we converted it to the COCO annotation style when we chose the best model from the TensorFlow model garden. The results, in term of accuracy, provided the best values for our problem and this type of architecture is one of the best in terms of classification problems.

The third architecture is common for recommendation systems so we converted the classification problem to a recommendation one and employed a vector embeddings database. We focused on this type of application as the new developments in terms of knowledge representation within large language models employ this technique. This methodology does not provide us with an accuracy metric but rather use the semantic representation in terms of vectors from which a distance provides the most similar elements from the database. In order to be able to measure its performance against the classic models, we needed to apply a ratio-based accuracy metric. For the vector embeddings encoder we also employed the standard VGG model (similar to the one from the reinforcement learning model). The results provided a very good accuracy level (much higher than the reinforcement learning and the Faster R-CNN model). See Table 8 for measured accuracy levels.

Table 8. Accuracy evaluation metrics (from the test data).

Nr. Crt.	Model	Architecture	Val. Acc. [%]	Test Acc. [%]
1	Model 1	Reinforcement Learning	58.8 ± 0.017	61.86 ± 8.03
2	Model 2	Faster R-CNN (RetinaNET/ResNET50)	84.48 ± 6.82	95.84 ± 7.76
3	Model 3	Vector Embeddings	n/a	96.21 ± 8.18

Vector embeddings and residual deep convolution networks proved to be two powerful techniques for our classification problem. ResNet, as a traditional deep learning model, is more suited in accuracy and feature learning but can be computationally expensive and requires large labeled datasets. While using such models no validation accuracy is available as the training is not a standard one but rather a feature extraction, thus the n/a from the Table 8. Vector embeddings, on the other hand, offer efficiency, flexibility, and semantic understanding, making them more appropriate for zero-shot learning. While their effectiveness depends on factors like embedding quality and computational resources, combining both approaches can often yield superior results.

Recent developments include Transformer architecture [38] that can also be applied for object detection purposes. This architecture enables a global understanding of the image through the self-attention mechanisms and can be applied for more complex structures like paintings. This development enables comprehensive feature representation while implementing a smaller complexity training pipeline.

The implemented models, reinforcement learning, Faster R-CNN, and vector embeddings, provided acceptable levels of accuracy while future enriching the architectures provided no significant increase in terms of accuracy.

5. Conclusions

The field of cultural heritage relies heavily on the correct authentication of paintings. This study highlights advances and examines the potential of artificial intelligence particularly machine learning models to assist experts in determining artworks' authenticity. While searching for the best assessment technique to be applied, we implemented various techniques and ML architectures, such as principal component analysis and computer vision. This approach allowed for whole artworks or incomplete images to be analyzed with a focus on pen strokes in sketches while searching for the characteristic elements of the artists that were learned during the training stage. During the training period, special attention was required for preprocessing (image cleanup), but these steps are no longer required during the inference stage.

While our approach focuses on artist specificity, computer vision models, such as the traditional regional convolution neural network, and more evolved network models, such as pre-trained models, can be employed without freezing parts of them, thus allowing them to be adapted to our specific task. As a novelty, we applied a newer vector embedding search algorithm based on the VGG16 network topology for feature extraction, allowing us to find similar works within the database. This approach extends future developments beyond our task of enhancing rapid artwork identification and classification to empower new applications.

While the best model with the highest accuracy (with great confidence levels) was obtained using the Faster R-CNN network architecture (providing accuracy levels of **95.84%**), we find it interesting that the embedding implemented using a stock pre-trained model such as VGG16 provided us with easier-to-interpret results while maximizing the model flexibility. We estimate that more complex models for embedding extractions may be more appropriate for this type of work.

The methodologies for overall accuracy (see Table 8) estimation greatly differ from model to model and are based on the manual testing of each model against the testing dataset. The testing evaluation was performed using three runs for the same model.

The experimental findings derived from the models and our approach can be used to categorize and identify features within artistic pieces, in addition to providing a tool that can aid evaluators in safeguarding cultural heritage.

Author Contributions: Conceptualization, R.R., G.C. and A.C.; methodology, A.C.; software, A.C., M.P. and S.M.; validation, R.R., G.C. and A.C.; formal analysis, R.R.; investigation, G.C. and A.C.; resources, R.R.; data curation, S.M.; writing—original draft preparation, G.C.; writing—review and editing, A.C.; visualization, all; supervision, A.C.; project administration, G.C.; funding acquisition, R.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was carried out through the Core Program with the National Research Development and Innovation Plan 2022–2027, with the support of MCID (project no. PN 23 05/2023, contract 11N/2023), and Program I—Development of the National R & D System, Subprogram 1.2 Institutional Performance—Projects for Excellence Financing in RDI (contr. 18PFE/2021).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ML	Machine Learning
AI	Artificial Intelligence
DETMOD	Deterministic Model
STATMOD	Statistical Model
PCA	Principal Component Analysis
CNN	Convolutional Neural Network
RPN	Regional Proposal Network
R-CNN	Regional Convolutional Neural Network
VGG	Visual Geometry Group at the University of Oxford
px	Pixels
LLM	Large Language Model
TOPK	TOP-K similarity search

References

1. Qureshi, M.A.; Deriche, M. A bibliography of pixel-based blind image forgery detection techniques. *Signal Process. Image Commun.* **2015**, *39*, 46–74. [CrossRef]
2. Chen, G.; Wen, Z.; Hou, F. Application of computer image processing technology in old artistic design restoration. *Heliyon* **2023**, *9*, e21366. [CrossRef] [PubMed]
3. Barglazan, A.-A.; Brad, R.; Constantinescu, C. Image Inpainting Forgery Detection: A Review. *J. Imaging* **2024**, *10*, 42. [CrossRef] [PubMed]
4. Leonarduzzi, R.; Liu, H.; Wang, Y. Scattering transform and sparse linear classifiers for art authentication. *Signal Process.* **2018**, *150*, 11–19. [CrossRef]
5. Ajorloo, S.; Jamarani, A.; Kashfi, M.; Haghi Kashani, M.; Najafizadeh, A. A systematic review of machine learning methods in software testing. *Appl. Soft Comput.* **2024**, *162*, 111805. [CrossRef]
6. Dobbs, T.; Ras, Z. On art authentication and the Rijksmuseum challenge: A residual neural network approach. *Expert Syst. Appl.* **2022**, *200*, 116933. [CrossRef]
7. Zeng, Z.; Zhang, P.; Qiu, S.; Li, S.; Liu, X. A painting authentication method based on multi-scale spatial-spectral feature fusion and convolutional neural network. *Comput. Electr. Eng.* **2024**, *118*, 109315. [CrossRef]
8. Schaerf, L.; Postma, E.; Popovici, C. Art authentication with vision transformers. *Neural Comput. Appl.* **2024**, *36*, 11849–11858. [CrossRef]
9. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv* **2016**, arXiv:1603.04467.
10. Van Rossum, G.; Drake, F.L., Jr. *Python Reference Manual*; Centrum voor Wiskunde en Informatica: Amsterdam, The Netherlands, 1995.
11. Pilgrim, M.; Willison, S. *Dive into Python 3*; Springer: Berlin/Heidelberg, Germany, 2009; Volume 2.
12. Van Rossum, G.; Drake, F.L. *Python 3 Reference Manual*; CreateSpace: Scotts Valley, CA, USA, 2009.
13. Kluyver, T.; Ragan-Kelley, B.; Pérez, F.; Granger, B.; Bussonnier, M.; Frederic, J.; Kelley, K.; Hamrick, J.; Grout, J.; Corlay, S.; et al. Jupyter Notebooks—A publishing format for reproducible computational workflows. In *Proceedings of the Positioning and Power in Academic Publishing: Players, Agents and Agendas*; Loizides, F., Schmidt, B., Eds.; IOS Press: Amsterdam, The Netherlands, 2016; pp. 87–90.
14. Messer, U. Co-creating art with generative artificial intelligence: Implications for artworks and artists. *Comput. Hum. Behav. Artif. Hum.* **2024**, *2*, 100056. [CrossRef]

15. Zaurín, J.R.; Mulinka, P. pytorch-widedeep: A flexible package for multimodal deep learning. *J. Open Source Softw.* **2023**, *8*, 5027–5027. [CrossRef]
16. Kaur, H.; Pannu, H.S.; Malhi, A.K. A Systematic Review on Imbalanced Data Challenges in Machine Learning: Applications and Solutions. *ACM Comput. Surv.* **2019**, *52*, 1–36. [CrossRef]
17. Brauwers, G.; Frasincar, F. A General Survey on Attention Mechanisms in Deep Learning. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 3279–3298. [CrossRef]
18. The Pandas Development Team. Pandas-Dev/Pandas: Pandas. 2020. Available online: https://zenodo.org/records/13819579 (accessed on 30 September 2024).
19. Wes McKinney. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference, Stéfan van der Walt, Austin, TX, USA, 28 June–3 July 2010; pp. 56 – 61. [CrossRef]
20. Bradski, G. The OpenCV Library. *Dr. Dobb'S J. Softw. Tools* **2000**, *25*, 2236121.
21. Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array programming with NumPy. *Nature* **2020**, *585*, 357–362. [CrossRef]
22. Neves, C.N.; da Encarnação, D.S.; Souza, Y.C.; da Silva, A.O.R.; Oliveira, F.B.S.; Ambrósio, P.E. Principal Component Analysis in Digital Image Processing for Automated Glaucoma Diagnosis. In Proceedings of the XXVII Brazilian Congress on Biomedical Engineering, Vitória, Brazil, 26–30 October 2020; Bastos-Filho, T.F., de Oliveira Caldeira, E.M., Frizera-Neto, A., Eds.; Springer Nature: Cham, Switzerland, 2022; pp. 1527–1532.
23. Canny, J. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 679–698. [CrossRef]
24. Fu, Y.; Wang, W.; Zhu, L.; Ye, X.; Yue, H. Weakly supervised semantic segmentation based on superpixel affinity. *J. Vis. Commun. Image Represent.* **2024**, *101*, 104168. [CrossRef]
25. Zhao, S.; Fan, Q.; Dong, Q.; Xing, Z.; Yang, X.; He, X. Efficient construction and convergence analysis of sparse convolutional neural networks. *Neurocomputing* **2024**, *597*, 128032. [CrossRef]
26. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556.
27. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv* **2014**, arXiv:1311.2524.
28. Ding, P.; Zhang, Y.; Deng, W.J.; Jia, P.; Kuijper, A. A light and faster regional convolutional neural network for object detection in optical remote sensing images. *ISPRS J. Photogramm. Remote Sens.* **2018**, *141*, 208–218. [CrossRef]
29. Xu, B.; Yang, G. Interpretability research of deep learning: A literature survey. *Inf. Fusion* **2025**, *115*, 102721. [CrossRef]
30. Kokhlikyan, N.; Miglani, V.; Martin, M.; Wang, E.; Alsallakh, B.; Reynolds, J.; Melnikov, A.; Kliushkina, N.; Araya, C.; Yan, S.; et al. Captum: A unified and generic model interpretability library for PyTorch. *arXiv* **2020**, arXiv:2009.07896.
31. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 1137–1149. [CrossRef] [PubMed]
32. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.
33. Lin, T.; Goyal, P.; Girshick, R.B.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *arXiv* **2017**, arXiv:1708.02002.
34. Kittichai, V.; Sompong, W.; Kaewthamasorn, M.; Sasisaowapak, T.; Naing, K.M.; Tongloy, T.; Chuwongin, S.; Thanee, S.; Boonsang, S. A novel approach for identification of zoonotic trypanosome utilizing deep metric learning and vector database-based image retrieval system. *Heliyon* **2024**, *10*, e30643. [CrossRef]
35. Wang, J.; Yi, X.; Guo, R.; Jin, H.; Xu, P.; Li, S.; Wang, X.; Guo, X.; Li, C.; Xu, X.; et al. Milvus: A Purpose-Built Vector Data Management System. In Proceedings of the 2021 International Conference on Management of Data, Shanxi, China, 20 June–25 June 2021; pp. 2614–2627.
36. Towhee Framework for Unstructured Data Using SoTA Machine Learning Models. Available online: https://towhee.io (accessed on 30 May 2024).
37. Alzubaidi, M.; Agus, M.; Makhlouf, M.; Anver, F.; Alyafei, K.; Househ, M. Large-scale annotation dataset for fetal head biometry in ultrasound images. *Data Brief* **2023**, *51*, 109708. [CrossRef] [PubMed]
38. Zhang, Z.; Sun, K.; Yuan, L.; Zhang, J.; Wang, X.; Feng, J.; Torr, P.H. Conditional DETR: A Modularized DETR Framework for Object Detection. *arXiv* **2021**, arXiv:2108.08902.