Article

Creating and Validating a Ground Truth Dataset of UML Diagrams Using Deep Learning Techniques

Javier Torcal , Valentín Moreno , Juan Llorens , Ana Granados *

Posted Date: 25 September 2024

doi: 10.20944/preprints202409.2000.v1

Keywords: UML diagram dataset; UML diagram classification; deep learning; convolutional neural networks; vision transformers

Article

Creating and Validating a Ground Truth Dataset of UML Diagrams Using Deep Learning Techniques

Javier Torcal, Valentín Moreno, Juan Llorens and Ana Granados *

Computer Science Department, Universidad Carlos III de Madrid, Leganés, Spain.
* Correspondence: agranado@inf.uc3m.es (A.G.)

Featured Application: Having a curated dataset of UML diagrams without duplicate elements and with accurate labels is particularly useful in the fields of software engineering and artificial intelligence, as it enables the training of models for similarity-based diagram searches, diagram classification, or automatic diagram processing, among other applications. More specifically, it can offer numerous benefits in areas such as software reuse, code generation, and design validation.

Abstract: UML (Unified Modeling Language) diagrams are graphical representations used in software engineering which play a vital role in the design and development of software systems and various engineering processes. Large, good-quality datasets containing UML diagrams are essential for different areas in the industry, research and teaching purposes, however few exist in the literature and it is common to find duplicate elements in the existing datasets. This might affect the evaluation of the models obtained when using these datasets. This paper addresses the challenge of creating a ground truth dastaset of UML diagrams, including semi-automated inspection to remove duplicates and ensure the correct labeling of all UML diagrams contained in the dataset. In particular, a dataset of six UML diagram classes has been assembled, comprising a total of 2,626 images (426 activity diagrams, 636 class diagrams, 352 component diagrams, 357 deployment diagrams, 435 sequence diagrams, and 420 use case diagrams). Importantly, unlike other existing datasets, ours contains no duplicate elements and all diagrams are correctly labeled. Our curated dataset is a valuable and unique resource for the research community because it serves as a foundation for training and evaluating various models. In this paper, we demonstrate this by training and testing several deep learning models using our dataset, achieving highly satisfactory results compared to those presented in other works in the literature. Additionally, our experimental results highlight the potential of Visual Transformers for UML diagram classification, setting our approach apart from others that predominantly used Convolutional Neural Networks for similar tasks.

Keywords: UML diagram dataset, UML diagram classification, deep learning, convolutional neural networks, vision transformers.

1. Introduction

The Unified Modeling Language (UML) is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system [1]. UML diagrams are an essential tool in software development, system modeling and architecture design, because they allow the visual representation of the structure and behavior of a system, facilitating the understanding and communication of complex concepts. Given that most UML diagrams are shared as images, having curated datasets of UML diagrams without duplicate elements and with accurate labels can offer numerous benefits for the industry in practical areas, such as software reuse, code generation and design validation. Such datasets are particularly useful in the field of software reuse, as they enable the training of models for similarity-based diagram searches, among other applications. Additionally, these kinds of datasets are highly valuable for teaching and research.

Although a few UML diagram datasets exist in the literature [2–4] it is common to find duplicate elements in the existing datasets, which may affect the evaluation of the models obtained when using these datasets. Therefore, further progress is needed in the construction of high-quality and reliable datasets.

To automate the construction of UML diagram datasets, since these diagrams are commonly shared as images, accurate classification of UML images is crucial. Early efforts focused on classifying UML diagrams applying machine learning. For example, in [5] the authors applied machine learning techniques to automatically classify images based on whether they contained a UML diagram or not. However, the superior capability of deep learning techniques for image classification compared to traditional machine learning [6] made these techniques the ideal tool for UML diagrams classification and a useful approach to advance the construction of UML diagram datasets. Therefore, shortly after the appearance of machine-learning-based works such as [5], several studies emerged within the context of deep learning to address the automatic classification of UML diagrams into multiple classes. For example, the feasibility of using deep learning techniques to classify four different types of UML diagrams employing various convolutional neural network (CNN) architectures was demonstrated in [2], yielding outstanding results. Following this success, that work was expanded to include 10 different types of UML diagrams in [3]. Different works also recognized the importance of developing an automatic classification tool for UML diagrams and explored different approaches using transfer learning with various neural network architectures [4,7]. Collectively, they emphasized the importance of creating a comprehensive image dataset to facilitate and support further studies.

In this paper, we take a step towards creating a high-quality ground truth dataset of UML diagrams, which includes semi-automated inspection supported by a software tool to remove duplicates and ensure the correct labeling of all UML diagrams contained in the dataset. Although the application of fully automated methods for dataset creation is more efficient, we believe in the critical role of human supervision to ensure the accuracy and reliability of the datasets created. Our curated dataset serves as a foundation for training and evaluating various models. In this paper, we demonstrate this by training and testing several deep learning models using our dataset, achieving highly satisfactory results compared to those presented in other works in the literature.

2. Materials and Methods

This section explains how our UML diagram dataset has been created and how the UML diagrams have been classified using deep learning techniques.

2.1. Dataset Creation

To build our dataset, we used the dataset presented in [4] as a seed. This dataset is composed of the six most commonly used categories of UML diagrams (class, use case, sequence, component, activity and deployment diagrams), with 250 images for each UML diagram category. Despite the valuable resource that this dataset represents for the scientific community, it contains several duplicate images in terms of content, even though the files are not exact copies (as shown in Figure 1, as an example). One of the goals of our work was not only removing these duplicate images, but also acquiring additional images to increase the size of the dataset.

Thus, our first step was obtaining additional UML diagram images. To do so, web scraping was applied using a Google Add-In, enabling an almost instant download of a huge volume of images. In this process, queries for the six UML diagram classes from the repository presented in [4] were executed on Google Images to collect new images. In this process, two distinct problems arised: the redundancy of images and the issue of the class appropriateness, where images retrieved did not match the specific types of UML diagrams being searched. To address these problems, filtering and removal of misclassifications were meticulously performed through visual inspection. Although this human supervision may appear to be a slight weakness in the process because of the lack of automation and the time consumption, it remains crucial for ensuring data quality, which is critical and can make a big difference in the performance and generalization of the models trained using these data. To assist us in this intricate task, we used the "Duplicate Cleaner Pro" program [8], which matches images based on a user-defined similarity threshold and allows searching for repeated images based on their content, even if the files are not identical.

After making sure that the created dataset had no repeated elements and that all diagrams were correctly labeled, the UML diagram images were converted to grayscale, saved in JPG format, and standardized to a uniform resolution. Our final dataset consists of 2,626 images across six UML diagram types: activity (426), class (636), component (352), deployment (357), sequence (435), and use case (420) diagrams. It is important to note that all UML diagrams are different and have been correctly categorized by diagram type, ensuring the reliability of our dataset.

2.2. Comparison with Existing Datasets

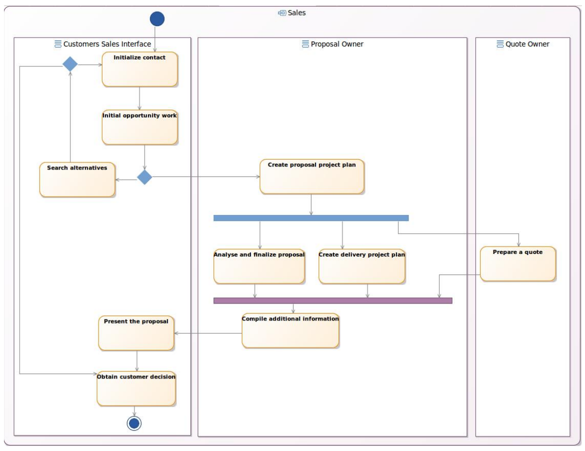
In spite of the great advances that the creation of other UML diagram datasets have meant for the literature, it is important to highlight the benefits of ours, comparing it with the existing datasets. In this respect, the key aspects to consider are features such as the composition of the datasets (including the number and types of UML diagrams) and, more importantly, their quality (assessed by the presence of repeated diagrams, as duplicates in the datasets can potentially threaten the validity of results). The features of each dataset can be found in Table 1.

Table 1. Datasets Composition and Quality.

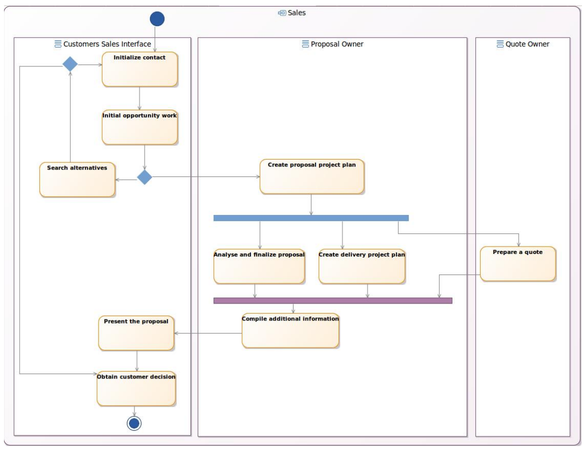
Research	Dataset Size	Number of Classes	Avg. Images per Class	Number of Duplicates
Shcherban *et al.* (I) [2]	3,231	4+1	941	292
Shcherban *et al.* (II) [3]	4,706	10+1	428	337
Tavares *et al.* [4]	1,500	6	250	145
Our work	**2,626**	**6**	**438**	**0**

Analyzing Table 1, one can observe that the dataset presented in the works from Shcherban *et al.* [2,3] contain 4 and 10 different types of UML diagrams, respectively, but introduce an additional class with non-UML images. In our opinion, it is better not to include non-UML images in a dataset that specifically contains UML diagrams of different classes to maintain dataset purity, unless there is a specific reason for including non-UML images, such as testing the model's ability to distinguish UML diagrams from other types of images. In this respect, we suggest it might be better to compile a non-UML-diagram class tailored to the specific context where it is needed, as this context can drastically change from one application to another. In other words, given that this type of image can be heterogeneous and context-dependent, it seems more practical to construct ad-hoc binary models (UML diagram or not) based on a specially compiled non-UML-diagram class for the desired context, to identify and remove non-UML images from the data. Finally, another point to consider is the effect that the non-UML class can have on classification performance, as this class could be comparatively easier to classify. Due to these reasons, and similarly to Tavares *et al.* [4], our dataset includes only UML diagrams, and comprises six different classes, each corresponding to a distinct type of UML diagram.

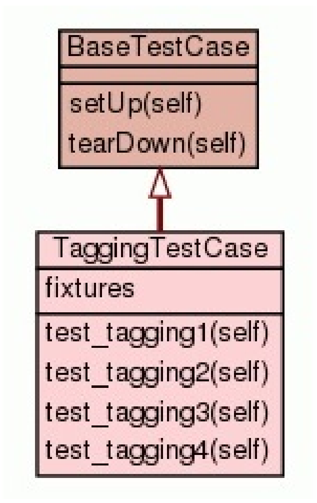
However, what stands out the most, when analyzing Table 1, is not the number of classes, but the number of duplicate diagrams. In this respect, it has to be pointed out that the number of duplicates was calculated in the existing datasets making use of the program "Duplicate Cleaner Pro", because, as mentioned earlier, this program allows comparisons between all images using similarity thresholds, which enables to look for repeated images in content, even if they are not exact copies. Our strategy essentially consisted of using this program and visually inspecting the images that the program gave as possible duplicates. In all the datasets we found a significant number of duplicate images in terms of their content, even though the files were not exact copies. Figure 1 shows examples of duplicate images found in existing datasets, illustrating the problem of duplicates.



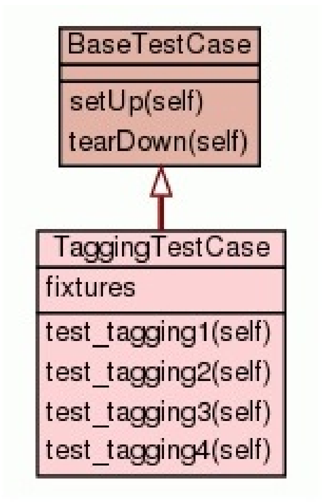
(a) Image 11421.jpg
Dataset Shcherban *et al.* (I) [2]



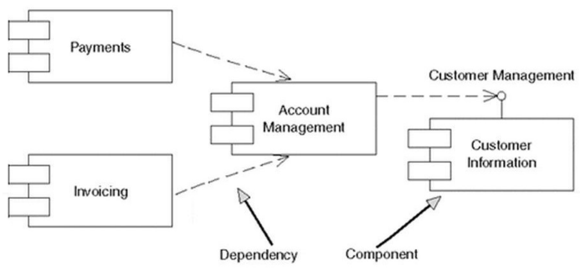
(b) Image 14248.jpg
Dataset Shcherban *et al.* (I) [2]



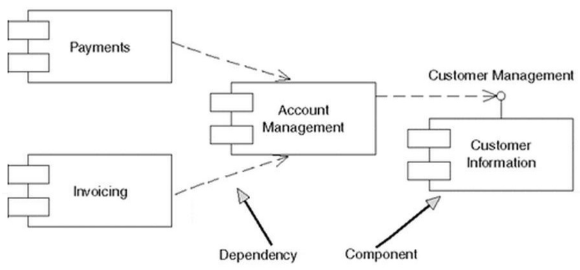
(c) Image uml_class_diagram_for_publicma_77qq.jpg
Dataset Shcherban *et al.* (II) [3]



(d) Image uml_class_diagram_for_publicma_84qq.jpg
Dataset Shcherban *et al.* (II) [3]



(e) Image Component_Diagram_39.jpg
Dataset Tavares *et al.* [4]



(f) Image Component_Diagram_110.jpg
Dataset Tavares *et al.* [4]

Figure 1. Examples of duplicate images found in the existing datasets.

We want to emphasize the importance of our dataset not having duplicate elements or incorrectly labeled elements because this guarantees greater precision and validity in the results obtained when using our dataset. Particularly, without duplicates, the risk of training and test sets containing the same images is avoided, ensuring that the results are truly representative and reliable, rather than artificially inflated. Without mislabeled items, major problems such as noise, bias and incorrect assessment are avoided, allowing accurate and reliable models to be developed with confidence in the results.

2.3. UML Diagram Classification

After meticulously constructing our dataset to ensure it is free of duplicates, we used it to train and assess several deep learning classifiers, with the aim of demonstrating that our curated dataset serves as a foundation for training and evaluating different models. It should be noted that highly satisfactory results have been obtained compared to those presented in other works in the literature, which demonstrates the quality and utility of our dataset.

In particular, after conducting several preliminary model trials, we selected the following CNN models: DenseNet-169 [9], DenseNet-201 [9] and ResNet-152 (2nd Version) [10], and the following ViTs [11]: ViT_b_16 (base version, 16 pixels patch), ViT_l_16 (large version, 16 pixels patch), and ViT_h_14 (huge version, 14 pixels patch). For all these architectures, transfer learning was employed using the pre-trained weights from the "ImageNet 1K" benchmark dataset developed by Stanford University [12]. Table 2 summarizes the main characteristics of the deep learning models used, that is, their respective families, their number of parameters (indicative of model complexity), and their image resolution requirements. As this table shows, all CNN models demand a fixed resolution of 224x224 pixels. In contrast, ViT transformers offer greater flexibility, with the minimal resolution requirements being 224x224 pixels for the base and large versions, and 518x518 pixels for the huge version. In terms of model depth, the most conservative architectures are DenseNets, which achieve a remarkable efficiency with a maximum of 20 million parameters. In contrast, ViT models substantially surpass CNNs in terms of parameter count, except for ResNet-152, which also exceeds the hundred-million mark.

Table 2. Presentation of Models Architecture.

Model	Family	Num. of params.	Image resolution
DenseNet-169	CNN	14 M	224 x 224 px
DenseNet-201	CNN	20 M	224 x 224 px
ResNet-152_V2	CNN	127 M	224 x 224 px
ViT_b_16	ViT	87 M	224 x 224 px
ViT_l_16	ViT	305 M	224 x 224 px
ViT_h_14	ViT	632 M	518 x 518 px

To increase our dataset size, data augmentation functions were applied to the images before inputting them into the models. These techniques included random rotations up to 30 degrees, along with random horizontal and vertical flips. Afterwards, the augmented images were converted into tensors and immediately normalized for training efficiency and stability. Then, the dataset was randomly divided into 5 equal parts and 5-fold cross-validation approach was employed for the training and evaluation of the models. Finally, it shoud be highlighted that the image size was adjusted to meet the input requirements of the models, that is, 224x224 pixels for all the models except for vit_h_14, which requires a minimum of 518x518 pixels, as previously mentioned.

Regarding the transfer learning, it should be pointed out that in the semi-trainable transfer learning (STTL) approach, the feature extractor component (in CNNs), and the image embedding section along with the encoder transformer (in ViTs), maintain their pre-defined weights by freezing its values during training. Reversely, the classifier component (in CNNs) and the classification head (in ViTs) are replaced with a multi-layer perceptron (MLP) block. This block incorporates a single hidden layer with a rectified linear unit (ReLU) activation function and includes dropout techniques between layers to mitigate overfitting. Only fine-tuning is applied to this MLP classification block during training.

During the training phases, the number of epochs was set to 10 because additional iterations did not yield significant improvements. Besides, the utilized optimizer was the adaptive method Adam [13] with a learning rate of 0.001, and the preferred loss function was Cross Entropy [14], which combines the softmax function with negative log-likelihood.

The models were evaluated using the accuracy and the F1-score. It is important to hightlight that these metrics were averaged over the values obtained from the 5 iterations of cross-validation. Apart from the F1-score of each classifier, the F1-score corresponding to each UML diagram class is shown.

3. Results

Table 3 shows the performance metrics for the six models, highlighting that the best CNN model is DenseNet-169 with an accuracy of 85.3% and an F1-score of 84.7%, beating its deeper counterpart, DenseNet-201, as well as the ResNet model. However, all of these models are surpassed by the outcomes of the ViTs, which consistently outperform their CNN counterparts. Furthermore, it can be observed that by increasing the depth of the ViTs, superior metric values are achieved, with the highest-achieving results being obtained by the large ViT version, ViT_h_14, where an exceptional accuracy of 92.7% and a F1-score of 92.4%, surpassing the state-of-the-art as it will be discussed in Section 4.

Table 3. STTL Experimental Results.

Model	Accuracy $\pm$ Std. Dev.	F1-score (%)
DenseNet-169	85.3 $\pm$ 2.2 %	84.7 %
DenseNet-201	84.1 $\pm$ 2.2 %	83.5 %
ResNet-152_V2	82.5 $\pm$ 1.6 %	81.6 %
ViT_b_16	86.9 $\pm$ 1.3 %	86.3 %
ViT_l_16	88.2 $\pm$ 0.9 %	87.7 %
ViT_h_14	**92.7 $\pm$ 1.0 %**	**92.4 %**

Table 4 displays the F1-scores for each class separately. Analyzing these values, one can observe that UML component diagrams are the hardest class to deal with, displaying a F1-score value consistently over 10% lower comparing with the rest UML diagrams. This can be due to the fact that UML symbols used to represent component diagrams have evolved between UML versions, from UML 1.x to UML 2.x. This presents a challenge for the classification of images containing UML diagrams. Nonetheless, almost a 83% F1-score value has been achieved when classifying UML component diagrams using ViT_h_14.

Table 4. STTL Results: F1-Score per class (UML diagram type).

Model	Activity	Class	Component	Deployment	Use Case	Sequence
DenseNet-169	81.2 %	90.6 %	69.3 %	83.9 %	89.8 %	93.4 %
DenseNet-201	81.5 %	89.2 %	66.2 %	81.1 %	89.2 %	93.6 %
ResNet-152_V2	78.9 %	88.3 %	62.2 %	79.8 %	88.2 %	92.2 %
ViT_b_16	84.5 %	91.8 %	70.6 %	82.5 %	91.5 %	96.9 %
ViT_l_16	89.1 %	91.6 %	72.1 %	85.2 %	90.9 %	97.0 %
ViT_h_14	91.6 %	94.8 %	82.8 %	91.9 %	95.4 %	97.7 %

In addition, in order to provide a more visual comparison of the results obtained, Figure 2 displays the F1-score metrics for each class. The same conclusions derived from the analysis of Table 4 can be drawn examining the said figure.

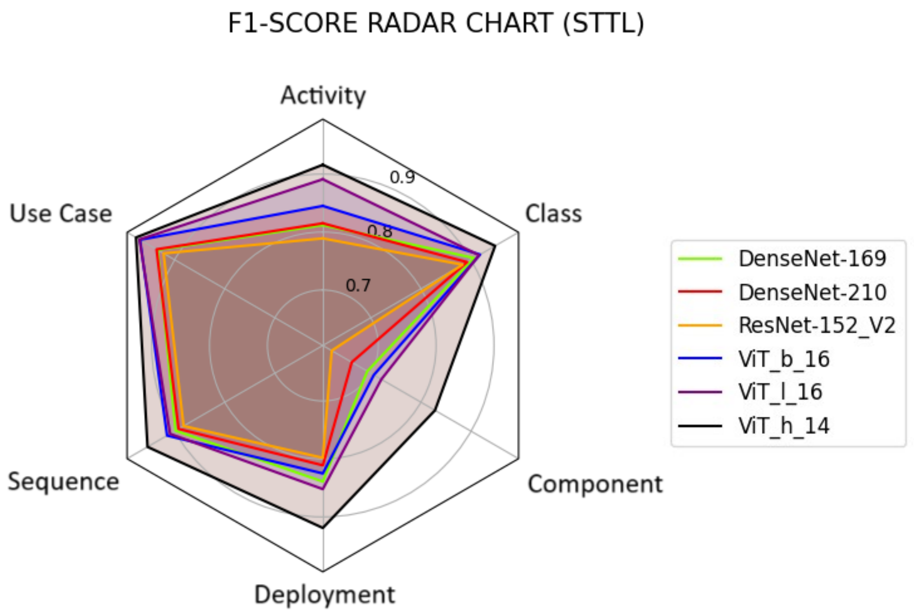


Figure 2. Radar Chart of the F1-score per class (STTL).

The final conclusion that can be drawn from the analysis of the obtained results is that it is necessary to increase the number of images of the less represented UML diagrams in the dataset. This is particularly critical in the case of component diagrams, since the significant evolution they have undergone from UML 1.x to UML 2.x makes it difficult to classify this type of diagram. Therefore, it is essential to increase the number of component diagrams collected to improve their representation.

4. Discussion

This section provides a comprehensive comparative analysis of our results in the context of related work in the literature. The intention is to juxtapose our results with similar research works [2–4], considering various aspects such as the nature of the problem, the quality of the data set and the main results obtained, with the aim of highlighting the significance of our contributions and identifying areas of improvement.

Table 5 provides an overview of the deep learning models used to classify UML diagrams, along with the results performance in terms of test accuracy and F1-score.

Table 5. Results Analysis and Discussion with Previous Related Research in STTL

Research	Model Architecture	Number of Classes	Duplicates (%)	Test Accuracy	Test F1-score
Shcherban *et al.* (I) [2]	MobileNet	$4 + 1^1$	9 %	96.8 %	91.91 %
Shcherban *et al.* (II) [3]	DenseNet-169	$10 + 1^1$	7 %	87.22 %	83.44 %
Tavares *et al.* [4]	Inception_V3	6	12 %	87.8 %	NA
This research	**ViT_h_14**	**6**	**0 %**	**92.7 %**	**92.4 %**

It is noteworthy that the best-performing model in this research, ViT_h_14, secures the second position in terms of the highest accuracy, being only surpassed by [2]. However, as indicated in Table 5, their approach encompassed just 4 UML classes and an additional non-UML class, simplifying the problem. In addition, another very important factor to take into account when analyzing the obtained results is the amount of repeated images in the data sets, which can result in performance metrics appearing more favorable.

Moreover, ViT_h_14 significantly outperforms its competitors in terms of F1-score. The rationale behind this fact could be that [2,3] incorporated non-UML images into their studies, causing the models not to focus as intensely on UML diagram features, consequently producing a notable decrease in their F1-values. This achievement in the F1 metric highlights the robust performance of this research model across all classes, despite facing specific challenges, notably in the case of component diagrams, as depicted in Figure 2.

Therefore, after carefully analyzing all the results obtained, both in our work and in previous studies, we can conclude that our results are more than satisfactory. Not only have we achieved similar or superior accuracy, but we have done so despite the significant difference in the percentage of duplicate elements in datasets from the literature (ranging from 7% to 12% in other datasets, while ours has 0% duplicate elements). Additionally, the F1-score we have obtained surpasses all those previously reported in other studies. It is also worth noting that our work demonstrates the suitability of ViTs for UML diagram classification, setting our approach apart from others that predominantly used CNNs for similar tasks.

5. Conclusions

In this work, we presented a ground truth dataset of UML diagrams free from duplicates and mislabeled elements, addressing the prevalent issue of repeated elements within existing literature datasets, where it is common to find repeated images based on their content, even if the files are not identical. Our dataset, comprising a total of 2,626 images from six UML diagram types (426 activity diagrams, 636 class diagrams, 352 component diagrams, 357 deployment diagrams, 435 sequence diagrams, and 420 use case diagrams), all correctly assigned to their corresponding classes and with no duplicate elements, constitutes a valuable resource for the research community.

The experiments we carried out to show the utility of our dataset, which consist of training different deep learning models, not only demonstrate that our dataset constitutes a valuable resource for the community but also highlight the potential of ViTs for UML diagram classification, setting our approach apart from others that predominantly used CNNs for similar tasks. Our findings indicate that our dataset, devoid of duplicate elements, outperforms previous studies in terms of the F1-score and achieves remarkable accuracy. Although our accuracy is the second best overall, it is noteworthy that the highest accuracy reported comes from a study that classifies only four types of UML diagrams, includes a fifth class of non-UML images, and uses a dataset with 9% duplicate elements. Finally, it should be noted that the insights gained from this work will serve as a foundation for future research, where we plan to use the best-performing classifiers to automatically expand and enhance our ground truth dataset.

Author Contributions: Conceptualization, V.M., J.Ll. and A.G.; methodology, J.T.; software, J.T.; validation, J.T., V.M., J.Ll. and A.G.; formal analysis, J.T.; investigation, J.T.; resources, J.T. and J.Ll.; data curation, J.T.; writing—original draft preparation, J.T. and A.G.; writing—review and editing, J.T., V.M., J.Ll. and A.G.; visualization, J.T. and A.G.; supervision, V.M and A.G.; project administration, J.T., V.M. and A.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The dataset presented in the study is openly available in link.

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

Abbreviations

The following abbreviations are used in this manuscript:

CNNs	Convolutional Neural Networks
DenseNets	Densely Connected Neural Networks
MLP	Multi-Layer Perceptron
ReLU	Rectified Linear Unit
RAM	Random Access Memory
STTL	Semi-Trainable Transfer Learning
UML	Unified Modelling Language
ViTs	Vision Transformers

References

1. Rumbaugh, J.; Booch, G.; Jacobson, I. *The Unified Modeling Language Reference Manual*; Addison-Wesley, 2010. Google-Books-ID: T7c3RwAACAAJ.
2. Shcherban, S.; Liang, P.; Li, Z.; Yang, C. Multiclass Classification of Four Types of UML Diagrams from Images Using Deep Learning. International Conference on Software Engineering and Knowledge Engineering, 2021.
3. Shcherban, S.; Liang, P.; Li, Z.; Yang, C. Multiclass Classification of UML Diagrams from Images Using Deep Learning. *International Journal of Software Engineering and Knowledge Engineering* **2021**, *31*, 1683–1698. Publisher: World Scientific Publishing Co., doi:10.1142/S0218194021400179.
4. Tavares, J.F.; Costa, Y.M.G.; Colanzi, T.E. Classification of UML Diagrams to Support Software Engineering Education. 2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW), 2021, pp. 102–107. ISSN: 2151-0830, doi:10.1109/ASEW52652.2021.00030.
5. Moreno, V.; Génova, G.; Alejandres, M.; Fraga, A. Automatic Classification of Web Images as UML Static Diagrams Using Machine Learning Techniques. *Applied Sciences* **2020**, *10*, 2406. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, doi:10.3390/app10072406.
6. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. doi:10.1145/3065386.
7. Gosala, B.; Chowdhuri, S.R.; Singh, J.; Gupta, M.; Mishra, A. Automatic Classification of UML Class Diagrams Using Deep Learning Technique: Convolutional Neural Network. *Applied Sciences* **2021**, *11*, 4267. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute, doi:10.3390/app11094267.
8. Ltd., D.S. Duplicate Cleaner. https://www.duplicatecleaner.com, 2024. Accessed: 2024-07-10.
9. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2261–2269. ISSN: 1063-6919, doi:10.1109/CVPR.2017.243.
10. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. ISSN: 1063-6919, doi:10.1109/CVPR.2016.90.
11. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2020.
12. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A.C.; Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision* **2015**, *115*, 211–252. doi:10.1007/s11263-015-0816-y.

13. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization, 2017. arXiv:1412.6980 [cs], https://doi.org/10.48550/arXiv.1412.6980.
14. Mao, A.; Mohri, M.; Zhong, Y. Cross-entropy loss functions: theoretical analysis and applications. Proceedings of the 40th International Conference on Machine Learning; JMLR.org: Honolulu, Hawaii, USA, 2023; Vol. 202, *ICML'23*, pp. 23803–23828.