




## Article

# Creating and Validating a Ground Truth Dataset of Unified Modeling Language Diagrams Using Deep Learning Techniques

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

Computer Science Department, Universidad Carlos III de Madrid, 28911 Leganés, Spain;  
100484116@alumnos.uc3m.es (J.T.); vmpelayo@inf.uc3m.es (V.M.); llorens@inf.uc3m.es (J.L.)

\* Correspondence: agranado@inf.uc3m.es

**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 dataset of UML diagrams, including semi-automated inspection to remove duplicates and ensuring the correct labeling of all UML diagrams contained in the dataset. In particular, a dataset of six UML diagram classes was assembled, comprising a total of 2626 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, serving as a foundation for training and evaluating diverse artificial intelligence 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



**Citation:** Torcal, J.; Moreno, V.; Llorens, J.; Granados, A. Creating and Validating a Ground Truth Dataset of Unified Modeling Language Diagrams Using Deep Learning Techniques. *Appl. Sci.* **2024**, *14*, 10873. <https://doi.org/10.3390/app142310873>

Academic Editor: Juan Francisco De Paz Santana

Received: 24 September 2024

Revised: 2 November 2024

Accepted: 20 November 2024

Published: 24 November 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 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 for the visual representation of the structure and behavior of a system, facilitating the understanding and communication of complex concepts.

Despite the years that have passed since the introduction of UML as a unified language, its relevance has not diminished. Instead, UML remains an essential tool in modern software engineering and is widely applied in the industry beyond academic contexts, as numerous studies in the literature confirm [2–7]. Over time, UML has evolved into a

highly mature language, with its latest stable version, 2.5, addressing a broad spectrum of engineering needs. UML supports applications such as workflow representation in DevOps and continuous integration environments, microservices architecture design (for example, using sequence diagrams to effectively represent service interactions), and reverse engineering for structuring and organizing software code. Even in Agile and iterative development contexts, where flexibility often supersedes exhaustive documentation, UML offers a lightweight, logical framework that aligns well with Agile methodologies. In Agile, UML is often used through simplified class diagrams to represent conceptual structures and use case diagrams to capture high-level interactions. This adaptability makes UML suitable for Agile by providing a visual, communicative tool that complements iterative workflows without impeding flexibility [8–12].

Considering UML's continued relevance and the fact that most UML diagrams are shared as images, the availability of a high-quality, well-curated dataset of UML diagrams becomes crucial. Since UML diagrams are frequently shared as images rather than editable models, having a dataset free from duplicates and misclassifications is essential for training machine learning models effectively. Such a dataset not only facilitates tasks such as accurate UML diagram classification but also opens up opportunities for developing advanced software engineering tools. For instance, a comprehensive and organized dataset could enable the creation of a sophisticated search engine for UML diagrams. This search engine would allow software engineers to search for diagrams by content, using model snippets as queries, rather than relying on keywords. This capability could greatly facilitate the reuse of models, accelerating design processes and enhancing efficiency in software development projects. Thus, having a high-quality, well-curated dataset of UML diagrams can offer numerous benefits for the industry in practical areas, such as software reuse [13], code generation [14], and design validation [15]. Additionally, it could be highly valuable for teaching and research.

Although a few UML diagram datasets exist in the literature [16–18], 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 by applying machine learning. For example, in [19] 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 [20] 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 [19], 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 [16], yielding outstanding results. Following this success, that work was expanded to include 10 different types of UML diagrams in [17]. 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 [18,21]. 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 is a valuable and unique resource for

the research community, as it serves as a foundation for training and evaluating diverse artificial intelligence models. In this paper, we demonstrate this by training and testing several deep learning models—three CNNs and three vision transformers (ViTs)—using our dataset, achieving highly satisfactory results compared to those presented in other works in the literature. In this regard, our experimental results underscore the potential of ViTs for UML diagram classification. To the best of our knowledge, despite the widespread application of ViTs in image classification tasks [22–24], they have not yet been applied to UML diagram classification. This sets our work apart from others that predominantly utilize CNNs for UML diagram classification.

## 2. Materials and Methods

This section explains how our UML diagram dataset was created and how the UML diagrams were classified using deep learning techniques.

### 2.1. Dataset Creation

To build our dataset, we used the dataset presented in [18] 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 [18] were executed on Google Images (<https://images.google.com/>, accessed on 1 June 2023) 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 (version 4.1.3) [25], which matches images based on a user-defined similarity threshold and allows for searching for repeated images based on their content, even if the files are not identical. This application uses a visual comparison technique to identify images that have undergone rotation, flipping, retouching, resizing, or format conversion. Additionally, it allows users to set the level of precision or tolerance for comparison, displaying clusters of potential duplicates for manual inspection and verification. In our filtering process, we applied several thresholds to detect duplicate diagrams: first, nearly exact matches were removed with a high similarity threshold (around 80%). The threshold was then gradually lowered until reaching 20% similarity, at which point only a few duplicates remained; the majority of diagrams were similar but not close enough to be considered duplicates. Finally, it should be noted that this filtering process resulted in the removal of approximately three out of every four images gathered, both from the seed dataset and from the web scraping process.

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 2626 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.

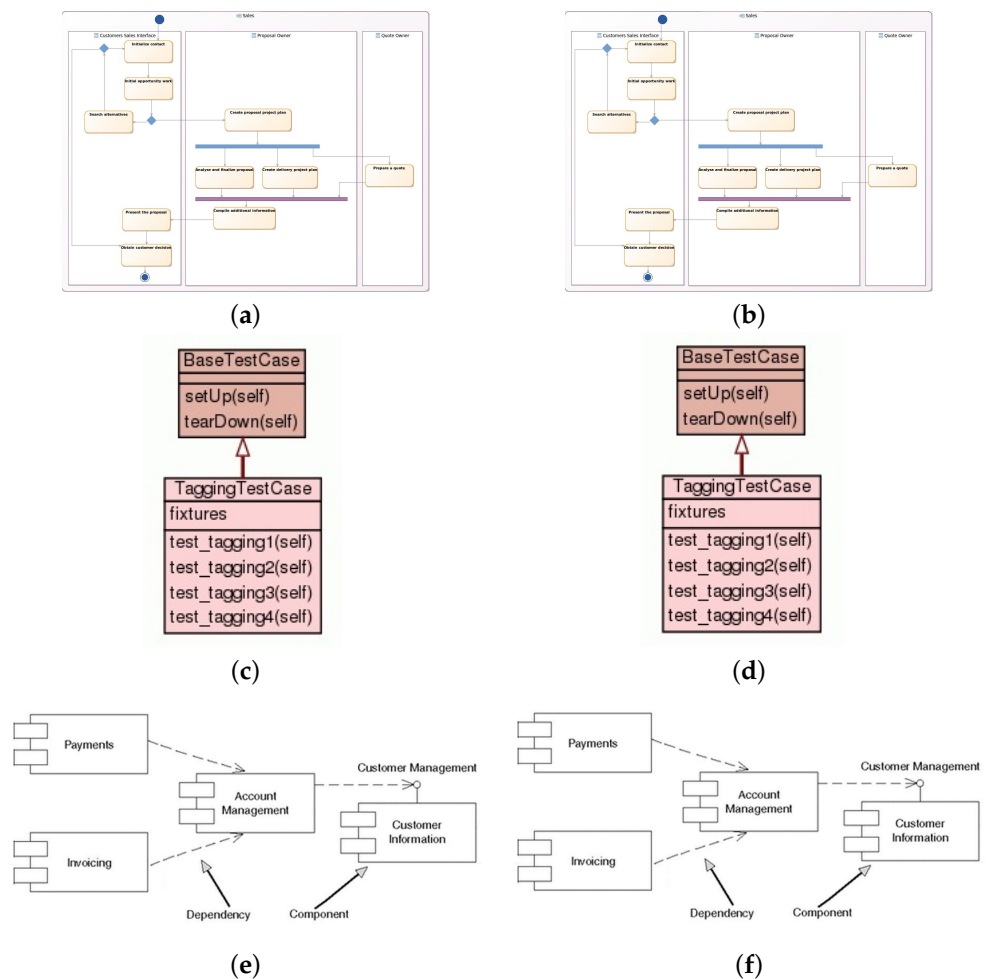
Research	Dataset Size	Number of Classes	Avg. Images per Class	Number of Duplicates	Year of Publication
Shcherban et al. (I) [16]	3231	4 + 1	941	292	2021
Shcherban et al. (II) [17]	4706	10 + 1	428	337	2021
Tavares et al. [18]	1500	6	250	145	2021
Our work	2626	6	438	0	2024

Analyzing Table 1, one can observe that the dataset presented in the works from Shcherban et al. [16,17] 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 diagrams 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. [18], 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 for comparisons between all images using similarity thresholds, which makes it possible 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. The pairs of duplicate diagrams in the different datasets show that, despite the files not being exact copies (as seen in the differing file names), the UML diagrams themselves are identical. This confirms that, while the files may vary, the content of the images remains the same in terms of the UML diagrams presented.

Regarding the size of the datasets, we want to highlight that our dataset, which encompasses images of the six most commonly used UML diagrams [18], consists of 2626 images, while the seed dataset from which it was derived contained 1500 diagrams, and included 145 duplicates [18]. In contrast, the datasets referenced in Table 1 from

Shcherban et al. contain 3231 diagrams (with 941 duplicates across 4 UML diagram types plus one non-UML class) [16] and 4706 diagrams (with 428 duplicates across 10 UML diagram types plus one non-UML class) [17]. Despite the larger size of these datasets, our collection stands out due to its unique advantage of being free from duplicates and having a higher average number of images per class (approximately 440). Additionally, all diagrams are correctly classified, thanks to the meticulous efforts invested in constructing our dataset. This careful curation not only enhances the quality of the dataset itself but also contributes to the robustness of model training and evaluation.



**Figure 1.** Examples of duplicate diagrams found in the existing datasets (Dataset from Shcherban et al. (I) [16], available at <https://doi.org/10.5281/zenodo.4595956> under the Creative Commons Attribution 4.0 International license; Dataset from Shcherban et al. (II) [17], available at <https://doi.org/10.5281/zenodo.5141007> under the Creative Commons Attribution 4.0 International license; Dataset from Tavares et al. [18], available at <https://doi.org/10.5281/zenodo.5544378> under the Creative Commons Attribution 4.0 International license). The pairs of duplicate diagrams in the different datasets show that, despite the files not being exact copies (as seen in the differing file names), the UML diagrams themselves are identical. This confirms that, while the files may vary, the content of the images remains the same in terms of the UML diagrams presented. (a) Image corresponding to file 11421.jpg from the dataset from Shcherban et al. (I) [16]; (b) Image corresponding to file 14248.jpg from the dataset from Shcherban et al. (I) [16]; (c) Image corresponding to file uml\_class\_diagram\_for\_publicma\_77qq.jpg from the dataset from Shcherban et al. (II) [17]; (d) Image corresponding to file uml\_class\_diagram\_for\_publicma\_84qq.jpg from the dataset from Shcherban et al. (II) [17]; (e) Image corresponding to file Component\_Diagram\_39.jpg from the dataset from Tavares et al. [18]; (f) Image corresponding to file Component\_Diagram\_110.jpg from the dataset from Tavares et al. [18].



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 for accurate and reliable models to be developed with confidence in the results.

As a final note, we want to point out that we chose not to conduct a performance comparison with pre-existing, unrefined datasets when training our CNN and ViT models, as theoretical performance on datasets with duplicate elements tends to be artificially inflated, with duplicate diagrams often appearing in both the training and test sets. This overlap compromises the validity of the test set, violating the principle that training and test data should remain separate. Such artificially enhanced results could therefore lead to misleading interpretations, and we aim to provide an unbiased and realistic assessment of our model's performance based on a curated dataset.

### 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 were 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 [26], DenseNet-201 [26], and ResNet-152 (2nd Version) [27], and the following ViTs [28]: ViT\_b\_16 (base version, 16 px 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 [29]. 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  $224 \times 224$  px. In contrast, ViT transformers offer greater flexibility, with the minimal resolution requirements being  $224 \times 224$  pixels for the base and large versions, and  $518 \times 518$  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	$224 \times 224$ px
DenseNet-201	CNN	20 M	$224 \times 224$ px
ResNet-152_V2	CNN	127 M	$224 \times 224$ px
ViT_b_16	ViT	87 M	$224 \times 224$ px
ViT_l_16	ViT	305 M	$224 \times 224$ px
ViT_h_14	ViT	632 M	$518 \times 518$ 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 five equal parts and a 5-fold cross-validation approach was employed for the training and evaluation of the models. Finally, it should be highlighted that the image size was adjusted to meet the input requirements of

the models; that is,  $224 \times 224$  pixels for all the models except for vit\_h\_14, which requires a minimum of  $518 \times 518$  pixels, as previously mentioned.

Regarding the transfer learning, it should be pointed out that, in the semi-trainable transfer learning (STTL) approach [30], 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 their values during training. Reversely, the classifier component (in CNNs) and the classification head (in ViTs) are replaced with a multi-layer perceptron (MLP) block [31], which is the only block that is fine-tuned during training. This MLP block consists of three fully connected layers with 1000, 512, and 6 neurons, respectively, each connected by a ReLU activation function [32] and incorporating dropout regularization with a probability of 0.2 to mitigate overfitting. It should be highlighted that dropout helps prevent overfitting by randomly omitting units and their connections during training, effectively averaging the predictions of a variety of "thinned" networks without slowing down inference [33]. Finally, the logarithm of the softmax function [34] was applied to obtain the classification probabilities.

During the training phases, the number of epochs was set to 10 because additional iterations did not yield significant improvements. Moreover, the batch size was set to 60 elements, the networks were trained using stochastic gradient descent (SGD) [35], the utilized optimizer was the adaptive method Adam [36] with a learning rate of 0.001, and the preferred loss function was cross-entropy [37], which combines the softmax function with negative log-likelihood.

The models were evaluated using the accuracy and the F1-score. It is important to highlight that these metrics were averaged over the values obtained from the five 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 presents the performance metrics for the six models. Among the CNNs, DenseNet-169 stands out with an accuracy of 85.3% and an F1-score of 84.7%, outperforming both its deeper variant, DenseNet-201, and the ResNet model, which achieves an accuracy of 82.5% and an F1-score of 81.6%. The significant performance gap between ResNet and DenseNet models is noteworthy. Despite ResNet having substantially more parameters (127 M) compared to DenseNet-169 (14 M), it does not surpass it in performance. This seemingly counterintuitive result can be attributed to the efficient architectural design of DenseNets, which optimize parameter utilization. In fact, DenseNets have demonstrated particular effectiveness on small-sized datasets, as evidenced in studies such as [26], where DenseNet models outperform ResNets on datasets like CIFAR-10 and CIFAR-100. This suggests that DenseNets are particularly suited for smaller datasets, leveraging their parameter efficiency as a key advantage.

Furthermore, examining Table 3, it becomes evident that ViTs consistently outperform all CNN models. When comparing ViT architectures with CNNs, the attention mechanisms in ViTs clearly provide an advantage over the convolutional structures of traditional CNNs. However, ViTs have significantly more parameters, which leads to greater computational demands. For example, the top-performing ViT model, ViT\_h\_14, has 632 million parameters, whereas DenseNet-169, the best CNN model, has only 14 million. This substantial difference makes training ViTs more resource-intensive, but given their superior performance, we believe the investment is justified.

In terms of concrete outcomes, a trend appears where deeper ViT architectures yield better performance, with ViT\_h\_14 achieving the highest accuracy of 92.7% and an F1-score of 92.4%, even surpassing state-of-the-art models, as discussed in Section 4. Notably, the large ViT variant, ViT\_l\_16, does not significantly benefit from its additional parameters, a phenomenon noted in [28]. The architectural similarity between ViT\_b\_16 and ViT\_l\_16, aside from differences in embedding dimensions, layers, and attention heads, highlights the challenge of achieving substantial performance gains simply by scaling model size.

The detailed analysis of Table 3 underscores the superior performance of the ViT\_h\_14 model, which consistently outperforms its counterparts across multiple metrics. To solidify these insights, we conducted a rigorous hypothesis test, which confirmed, with 99% confidence, that ViT\_h\_14 significantly outperforms all other models. This statistical validation reinforces our conclusion that ViT\_h\_14 represents a substantial advancement in model performance.

**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 of the 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, an F1-score value of almost 83% was achieved when classifying UML component diagrams using ViT\_h\_14.

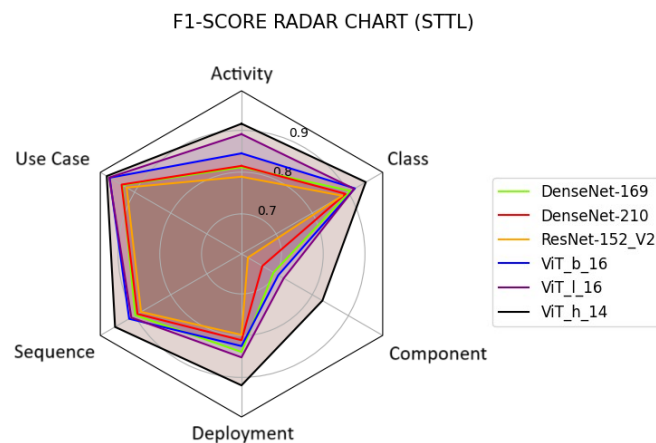
**Table 4.** STTL Results: F1-score per class (UML diagram type).

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, to provide a more visual comparison of the results obtained, Figure 2 presents the F1-score metrics for each diagram class. Analyzing this figure, it can be observed that, for sequence diagrams, use case diagrams, and class diagrams, all tested models delivered similar results. The F1-score outcomes for activity diagrams were more varied, with a noticeably better performance from the three ViTs compared to the CNN architectures. However, the most pronounced differences occurred with component diagrams. Although ViTs also outperformed CNNs in this case, it is especially clear that the ViT\_h\_14 model achieved significantly better results than all other models. In this regard, it is important to reiterate that we conducted a hypothesis test to compare the performance of all models, and we can confidently state that the ViT\_h\_14 model performs significantly better than the rest, with a 99% confidence level. Therefore, the same conclusions derived from the analysis of Table 4 can be drawn examining Figure 2.

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.





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

#### 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 [16–18], considering various aspects such as the nature of the problem, the quality of the dataset, 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) [16]	MobileNet	4 + 1 <sup>1</sup>	9%	96.8%	91.91%
Shcherban et al. (II) [17]	DenseNet-169	10 + 1 <sup>1</sup>	7%	87.22%	83.44%
Tavares et al. [18]	Inception_V3	6	12%	87.8%	NA
This research	ViT_h_14	6	0%	92.7%	92.4%

<sup>1</sup> Observe that +1 denotes an additional class for non-UML images.

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 [16]. However, as indicated in Table 5, their approach encompassed just four 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 datasets, 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 [16,17] 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 2626 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.L. and A.G.; methodology, J.T.; software, J.T.; validation, J.T., V.M., J.L. and A.G.; formal analysis, J.T.; investigation, J.T.; resources, J.T. and J.L.; data curation, J.T.; writing—original draft preparation, J.T. and A.G.; writing—review and editing, J.T., V.M., J.L. 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.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The dataset presented in the study is openly available in Zenodo. <https://doi.org/10.5281/zenodo.13831663>, accessed on 22 November 2024.

**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 Modeling Language
ViTs	Vision Transformers

## References

1. Rumbaugh, J.; Booch, G.; Jacobson, I. *The Unified Modeling Language Reference Manual*; Addison-Wesley: Boston, MA, USA, 2010; Google-Books-ID: T7c3RwAACAAJ.
2. Torre, D.; Genero, M.; Labiche, Y.; Elaasar, M. How consistency is handled in model-driven software engineering and UML: An expert opinion survey. *Softw. Qual. J.* **2023**, *31*, 1–54. [\[CrossRef\]](#)
3. Ozkaya, M.; Erata, F. A survey on the practical use of UML for different software architecture viewpoints. *Inf. Softw. Technol.* **2020**, *121*, 106275. [\[CrossRef\]](#)
4. Fernández-Sáez, A.M.; Chaudron, M.R.V.; Genero, M. An industrial case study on the use of UML in software maintenance and its perceived benefits and hurdles. *Empir. Softw. Eng.* **2018**, *23*, 3281–3345. [\[CrossRef\]](#)
5. Ho-Quang, T.; Hebig, R.; Robles, G.; Chaudron, M.R.; Fernandez, M.A. Practices and Perceptions of UML Use in Open Source Projects. In Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), Buenos Aires, Argentina, 20–28 May 2017; pp. 203–212. [\[CrossRef\]](#)
6. Chaudron, M.R. Empirical Studies into UML in Practice: Pitfalls and Prospects. In Proceedings of the 2017 IEEE/ACM 9th International Workshop on Modelling in Software Engineering (MiSE), Buenos Aires, Argentina, 21–22 May 2017; pp. 3–4. [\[CrossRef\]](#)
7. Storey, V.C.; Lukyanenko, R.; Castellanos, A. Conceptual Modeling: Topics, Themes, and Technology Trends. *ACM Comput. Surv.* **2023**, *55*, 317:1–317:38. [\[CrossRef\]](#)
8. Rumpe, B. *Agile Modeling with UML*; Springer International Publishing: Cham, Switzerland, 2017. [\[CrossRef\]](#)
9. Suartana, I.M.; Puspitaningayu, P.; Pratama, S.A.; Dwiyantri, S.; Maspiyah, M.; Haryudo, S.I. Modeling agile development of Web application e-monev using UML. *E3S Web Conf.* **2024**, *513*, 02010. [\[CrossRef\]](#)
10. Mornie, M.N.; Jali, N.; Junaini, S.N.; Mit, E.; Shiang, C.W.; Sae, S. Visualisation of User Stories in UML Models: A Systematic Literature Review. *Acta Inform. Pragensia* **2023**, *12*, 419–438. [\[CrossRef\]](#)
11. Lethbridge, T.C.; Algablan, A.; Lethbridge, T.C.; Algablan, A. Umple: An Executable UML-Based Technology for Agile Model-Driven Development. In *Advancements in Model-Driven Architecture in Software Engineering*; IGI Global: Hershey, PA, USA, 2021; pp. 1–25, ISBN 9781799836612. [\[CrossRef\]](#)
12. Abrahamsson, P.; Salo, O.; Ronkainen, J.; Warsta, J. Agile Software Development Methods: Review and Analysis. *arXiv* **2017**, arXiv:1709.08439. [\[CrossRef\]](#)
13. Mikkonen, T.; Taivalsaari, A. Software Reuse in the Era of Opportunistic Design. *IEEE Softw.* **2019**, *36*, 105–111. [\[CrossRef\]](#)
14. Durai, A.D.; Ganesh, M.; Mathew, R.M.; Anguraj, D.K. A novel approach with an extensive case study and experiment for automatic code generation from the XMI schema Of UML models. *J. Supercomput.* **2022**, *78*, 7677–7699. [\[CrossRef\]](#)
15. Maropoulos, P.G.; Ceglarek, D. Design verification and validation in product lifecycle. *CIRP Ann.* **2010**, *59*, 740–759. [\[CrossRef\]](#)
16. Shcherban, S.; Liang, P.; Li, Z.; Yang, C. Multiclass Classification of Four Types of UML Diagrams from Images Using Deep Learning. In Proceedings of the International Conference on Software Engineering and Knowledge Engineering, Online, 1–10 July 2021; pp. 57–62. [\[CrossRef\]](#)
17. Shcherban, S.; Liang, P.; Li, Z.; Yang, C. Multiclass Classification of UML Diagrams from Images Using Deep Learning. *Int. J. Softw. Eng. Knowl. Eng.* **2021**, *31*, 1683–1698. [\[CrossRef\]](#)
18. Tavares, J.F.; Costa, Y.M.G.; Colanzi, T.E. Classification of UML Diagrams to Support Software Engineering Education. In Proceedings of the 2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW), Melbourne, Australia, 15–19 November 2021; pp. 102–107. [\[CrossRef\]](#)
19. Moreno, V.; Génova, G.; Alejandro, M.; Fraga, A. Automatic Classification of Web Images as UML Static Diagrams Using Machine Learning Techniques. *Appl. Sci.* **2020**, *10*, 2406. [\[CrossRef\]](#)
20. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [\[CrossRef\]](#)
21. Gosala, B.; Chowdhuri, S.R.; Singh, J.; Gupta, M.; Mishra, A. Automatic Classification of UML Class Diagrams Using Deep Learning Technique: Convolutional Neural Network. *Appl. Sci.* **2021**, *11*, 4267. [\[CrossRef\]](#)
22. Maurício, J.; Domingues, I.; Bernardino, J. Comparing Vision Transformers and Convolutional Neural Networks for Image Classification: A Literature Review. *Appl. Sci.* **2023**, *13*, 5521. [\[CrossRef\]](#)
23. Bhojanapalli, S.; Chakrabarti, A.; Glasner, D.; Li, D.; Unterthiner, T.; Veit, A. Understanding Robustness of Transformers for Image Classification. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 10–17 October 2021; pp. 10211–10221, ISSN 2380-7504. [\[CrossRef\]](#)
24. Bazi, Y.; Bashmal, L.; Rahhal, M.M.A.; Dayil, R.A.; Ajlan, N.A. Vision Transformers for Remote Sensing Image Classification. *Remote Sens.* **2021**, *13*, 516. [\[CrossRef\]](#)
25. DigitalVolcano Software Ltd. Duplicate Cleaner. 2024. Available online: <https://www.duplicatecleaner.com> (accessed on 31 October 2024).
26. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269, ISSN 1063-6919. [\[CrossRef\]](#)
27. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778, ISSN 1063-6919. [\[CrossRef\]](#)

28. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. *An Image is Worth  $16 \times 16$  Words: Transformers for Image Recognition at Scale*; Cornell University: Ithaca, NY, USA, 2020. [[CrossRef](#)]
29. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [[CrossRef](#)]
30. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? In Proceedings of the 27th International Conference on Neural Information Processing Systems—Volume 2, Cambridge, MA, USA, 8–13 December 2014; NIPS'14; pp. 3320–3328. [[CrossRef](#)]
31. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [[CrossRef](#)]
32. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning, Madison, WI, USA, 21 June 2010; ICML'10; pp. 807–814. [[CrossRef](#)]
33. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
34. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Google-Books-ID: Np9SDQAAQBAJ.
35. Bottou, L. Stochastic Gradient Descent Tricks. In *Neural Networks: Tricks of the Trade: Second Edition*; Montavon, G., Orr, G.B., Müller, K.R., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 421–436. [[CrossRef](#)]
36. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980. [[CrossRef](#)]
37. Mao, A.; Mohri, M.; Zhong, Y. Cross-entropy loss functions: Theoretical analysis and applications. In Proceedings of the 40th International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023; ICML'23; Volume 202, pp. 23803–23828.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.