U.P.B. Sci. Bull., Series C, Vol. 85, Iss. 4, 2023 ISSN 2286-3540

# UML DIAGRAMS CLASSIFICATION WITH MIMO AND SISO NEURAL NETWORKS

Irina-Gabriela NEDELCU[1], Simona Iuliana CARAMIHAI[2], Stefan MOCANU[3], Anca Daniela IONITA[4]

*The benefits of artificial intelligence techniques have begun to be investigated for software modeling, as for other fields, but there is still a lot to explore. The standard Unified Modeling Language has been used for a long time, so that representative data sets have been produced to allow neural network training. The goal of this paper is to provide a multi-label classification with two neural network techniques, aiming to identify if an image contains a UML diagram or not, and to determine its type. The evaluated architectures are Multi Input Multi Output (MIMO) and Single Input Single Output (SISO). The paper also presents their evaluation, performance metrics results, and several examples.*

**Keywords**: Unified Modeling Language, machine learning, image processing

## 1. Introduction

The application of machine learning to modeling has the potential to drive increased adoption of model-based software engineering. Artificial intelligence techniques and image classification have benefited during recent years from many improvements in regard to the aspects, or the specific criteria that needed to be considered prior to classifying. For problems solved using machine learning or deep learning, the data feeding the training algorithms is more important than the algorithm itself.

In terms of combining the field of software modeling with artificial intelligence techniques, there are many aspects to be explored. Based on Unified Modeling Language (UML), many examples of diagrams might be similar, as some application domains follow a standard set of elements that are found in each project. Thus, an area of research may be software maintenance, as explored in [12]. Additionally, an application may be useful in the educational field, to automate certain processes. For example, the software can check whether a certain

[1] Ph.D. Student Eng., National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: irina.nedelcu@stud.acs.upb.ro
[2] Prof., Dept. of Automation and Industrial Informatics, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail simona.caramihai@upb.ro
[3] Assoc. Prof., Dept. of Automation and Industrial Informatics, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: stefan.mocanu@upb.ro
[4] Prof., Dept. of Automation and Industrial Informatics, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: anca.ionita@upb.ro

diagram has similarities with others, meaning the diagram could have been initially conceived by some other student. An easy way to trick a visual system concerning the UML diagrams is to arrange the components differently, so the diagram would look different even if the logic is the same. Based on this example, the feature extraction algorithms can be used so they can identify common elements and tell the similarity between two diagrams even if they are not arranged in the same manner in the image [13]. However, some applications based on machine learning might face challenges to provide a dataset with enough data. Throughout time, many techniques have been proposed to deal with small datasets, such as under-sampling, over-sampling, or augmentation [6].

The purpose of this paper is to determine if an image contains a UML diagram or not, then, if the image is marked as a UML diagram, to determine what kind of diagram it is. Nonetheless, our research evaluates the performance of a model trained with a small dataset, and specific tools considered in the architecture of the model to make it perform well. The research objectives are summarized in Fig. 1. Section 2 presents related work and what use cases have been identified based on it. Section 3 follows by describing the used dataset and processing of the data. It includes the processing of the labels as well. Next, the paper describes the applied architectures, how they were built and a comparison between them (Section 4). Section 5 presents testing results and examples. The paper concludes with achievements and next steps.

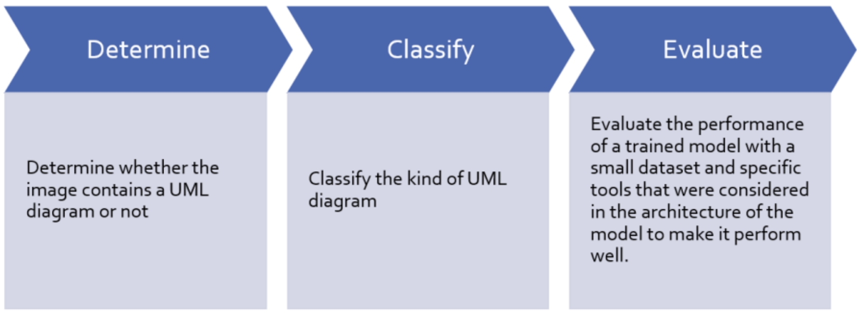

Determine
Classify
Evaluate
Determine whether the image contains a UML diagram or not
Classify the kind of UML diagram
Evaluate the performance of a trained model with a small dataset and specific tools that were considered in the architecture of the model to make it perform well.

Fig. 1. Research objectives

## 2. Related Work

A challenge of many research projects is to find techniques that help obtaining models that perform well even for training with small datasets. In computer vision, using artificial intelligence approaches, convolutional neural networks (CNNs) are still the ones being the most used and providing the best results. On the other hand, a new architecture has recently been proposed, Vision Transformer (ViT) [3], which has many differences in comparison with CNNs. It offers a very good performance in computer vision; it is robust and has a simplified design [4]. Considering the research paper using ViT [3], they used this architecture given a large-scale dataset and part of pre-training.

The usage of machine learning in combination with UML faces challenges even from the dataset creation. There are not may datasets publicly available and many projects still need to go through the step of creating their own dataset given the limited options currently offered. Examples of datasets concerning the UML area are ModelSet [8] and a UML dataset publicly available in GitHub [10]. As there are currently multiple opportunities to research the benefits of artificial techniques into UML, expanding the variety of datasets are important as this is one of the most time-consuming processes that takes place in projects concerning machine learning techniques. The process of building a dataset is also complex in terms of validating that the data is correct for the labels applied, which is still done manually by a human operator in most of the cases.

Another subject of interest is exploring the benefits of artificial intelligence techniques into UML diagrams. As class diagrams are the most well-known according to [7], most of the articles combine the two topics, such as using machine learning or deep learning to determine if an image is a class diagram or not. An example used a dataset that was manually built and validated by professionals, which does not consider the textual content of the diagram to classify [4]. Additionally, other UML diagrams have been used to determine if the image contains a diagram or not. Other research focus on semantic elements of the diagrams. The extraction of semantic elements helps to better understand how a diagram is realized and what it takes to build a new one that is useful for the end user [1].

## 3. Dataset with UML Diagrams Images Used for Machine Learning

The most analyzed UML model kinds are class diagrams [7], as they are often considered the most popular ones [10]. For the scope of our research, we selected class, sequence, and state machine diagrams to be used for diagram type classification. Section 3.1. describes gathering the data and deciding on the amount in consideration with imbalanced classes problems. Section 3.2 provides information about the dataset processing, to be ready for the training process.

### 3.1. Building the Dataset

To build the dataset for this research, the main source was a public database in GitHub provided by [10]. The database was built especially because previously there was no source with a wide range of image diagrams that can be used for future research problems with UML. This repository contains UML diagrams in various formats, collected from other related research projects, web crawling and diagram extraction, based on the type or donations from the community to help future development in the domain. An important requirement of this database was that during its construction, duplicates were removed. This detail is already a significant win for the training model, as it will not face performance issues due to actual lack of data caused by identical samples.

After filtering the initial results, the dataset contains 57,822 UML images / models. As previously mentioned, we extracted 3 of the most used UML diagrams separately, labeled a category for other kinds of UML diagrams and added a category for non-UML images. Table 1 contains details about images count per class from the source dataset.

*Table 1*

**Data obtained for this project per diagram category**

| Category | Number of images |
|---|---|
| Class Diagram | 7123 |
| Sequence Diagram | 540 |
| State Diagram | 173 |
| Other UML Diagrams | 1882 |
| Non-UML Images | 225 [downloaded outside the main dataset] |

To increase the number for the small samples and to get the non-UML images, we used a Google images downloader python script, which extracts images from Google Images, given certain keywords, such as “Sequence Diagram” or “Sequence UML”. Additionally, the initial dataset described above provides the image using a URL, so we built a custom script to download the images provided in the URLs. As a result, the state diagram, which has the smallest number or samples as stated in Table 2, was increased to 227 images.

### 3.2. Processing the Dataset

The most used approach for dataset expansion in computer vision is augmentation. However, for UML diagrams as samples, they are always straight in the picture and most of the times only black and white. Thus, rotating the image or adjusting the colors do not apply to extend the dataset. An important aspect is that each dataset class is very specific in their content, so the features might be easily extracted and learned. Thus, each class was reduced to 225 images, meaning the down-sampling process was applied to fix the class imbalance problems.

After setting the amount of data per class, the images were processed to a matrix of numbers. To validate that data quality is not impacted in the image, the number matrix was also processed backwards, to an image (see Fig. 2). Standard deviation and mean were the metrics used to confirm data was not lost, given they were under zero respectively 0.5 which were the set thresholds.

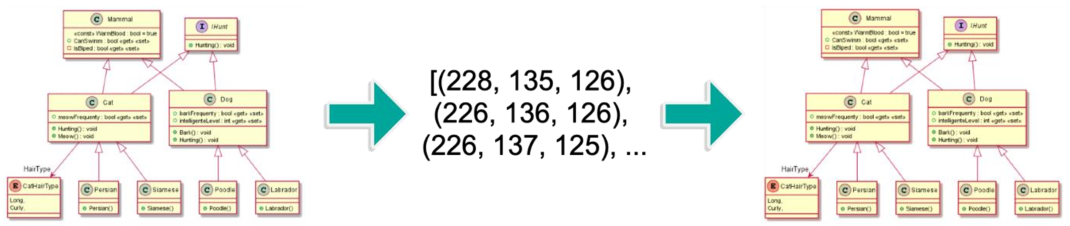


Fig. 2. Image to matrix and backwards processing

In terms of labels processing, the model has 2 tasks: to say if the image contains a UML diagram or not, and to determine what kind of diagram it is). Thus, the model needs to provide two outputs. Each set of labels was processed with the binarization technique that is highlighted in Fig. 3.

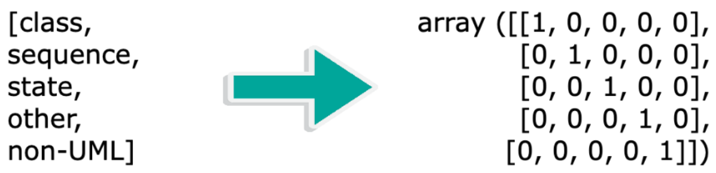


Fig. 3. Label binarization process

## 4. Architectures' performance comparison

For this research paper, we analyzed two neural network architectures. One approach was to design the architecture as multiple inputs and outputs and the other approach was to explore single input and output classification and investigate the performance if each approach. The following chapters cover the architecture proposed for each one and an overall analysis on the results.

Each model was trained using Google Colab resources in Jupyter notebooks, so we could run the code in steps and investigate or fix separate parts of the whole project. It is important to note that if the notebook disconnects each step needs to run again. As a workaround, wherever possible, we saved the results offline so we could load them later and advance faster with the project.

For example, after processing the images as 3-rank tensor, which means a matrix of numbers representing RGB code, we saved the results in a serialized object that got deserialized before any new notebook run. A similar technique was applied for labels binarization. Such an approach saved approximately 30 minutes for each run.

For better comparison, wherever possible, we tried to use the exact techniques or hyper-parameters so we could conclude which approach has a better performance for our problem. Thus, each neural network used the ADAM (Adaptive Moment Estimation) optimization algorithm [2] with the hyper-parameters learning rate 10-3 and decay rate beta1 0.9, decay rate beta2 0.999 and epsilon 10-8. For multi-class classification, we applied the Visual Geometry Group VGG neural network architecture [11].

### 4.1. Multi input multi output (MIMO) neural network

As stated previously, the output expected from the model is to say whether the image contains a UML diagram and which type of diagram. Because of this, the model is a multi-output one. Additionally, each output falls into a different type of classification, one is binary classification while the other one is multi-class. For each type, the input is specific, so the neural network will get separate inputs to learn from the data differently. Thus, the model is a multi-input one as well. These kinds of architectures are called multi-input multi-output (MIMO) neural networks, one example being [9].

Such tasks need to be designed and monitored carefully during the training. The model performs complex processes in parallel and there is a high risk the processing is not properly prepared, and performing the training process will not lead to a model able to offer the expected results. In current research there are not many MIMO architectures proposed for neural networks and none of them has been widely used to get a better overview on the performance of such an architecture.

As a general presentation, the whole process will be split into 4 main steps. First, we apply feed forward for loss calculation. Then, the optimizer is initialized, for this project, the optimizer used is ADAM (Adaptive Moment Estimation) with the learning rate 10-3. Then, the training process starts which calculates the loss using feed forward, used backpropagation to calculate the gradients and updates the weights and biases using the ADAM optimizer. Last, through a feed forward the validation process takes place by checking if the loss was reduced and how well the model predicts the correct values.

The input was loaded all together, and it was split using the labels added in the naming of the images or directories. The project required 2 inputs, a subset of classified types of UML diagrams and a subset of UML and non-UML diagrams. To distinguish which set of data belongs to each neural net problem we used directory and file naming conventions and identified the right output by processing it through the code. Then each input will go through a sub-network during training. Each sub-network has a different purpose that can also be modelled differently from one to another.

For binary-class classification the project also used a CNN and a VGG as well. We defined the loss as binary cross entropy, and the number of layers was smaller than the one needed for multi class

For the multi-class classification problem, the architecture or the sub-network followed the structure of a VGG, a convolutional neural network that is based on AlexNet with the specification that it contains smaller receptive fields, and the number of parameters is significantly smaller in comparison with AlexNet.

After each sub-network trains for the specific problem that needs to be solved, the architecture contains a concatenation layer that unites the two resulting outputs so new data can be classified with multiple labels at once.

### 4.2. Single input single output (SISO) neural network

In comparison with the MIMO architecture, we start with one single input which is formed by the following categories class diagram, sequence diagram, state diagram, other UML diagrams and non-UML diagram images. Similarly, to the MIMO architecture, we loaded the directories and images, then split it based on the naming to fit the categories listed above.

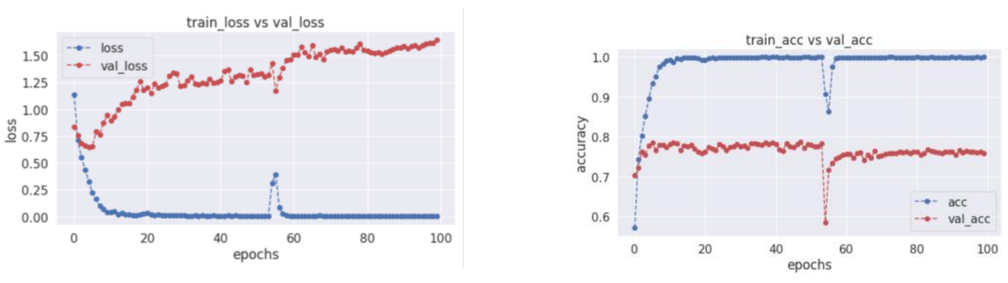
Such an input applies to multi-class classification and here we also use the exact same VGG architecture we applied for the sub-network of MIMO neural network. As mentioned in the beginning of the chapter we kept as many similarities as we could between the two investigated neural network designs so we could compare which approach provides better performance. To accommodate the multi-labeling of new data sets, meaning, whether the image contains a UML diagram or not and what kind of diagram it is, after the model predicted to which category the model belongs to, if the image was classified as class diagram, sequence diagram, state diagram or other UML diagrams we processed the result to display that the image represents a UML diagram or not and what kind it is through the code using the neural network’s result.

## 5. Comparison Results

### 5.1. Accuracy and Loss Evaluation

For each project we used the same platform and the same configuration to compare the results. The MIMO neural network was trained for 20 hours, while the SISO neural work trained for 14 hours. During training we monitored the accuracy and loss of the trained model to identify possible issues when the model was built. No issue was faced in any of the two cases, even if one epoch showed values that were not going as expected the neural network calibrated with the next epoch runs. The data was saved to be displayed as a plot for evaluation.

Fig. 4 a) displays the accuracy collected from the MIMO architecture during training. By looking at the accuracy plot, we understand the training went as desired, with one spike between epochs 50 and 60. The spike indicates that the model is not well structured at that step, and this could have caused an issue if the training was stopped at that epoch. However, in our case, the model recovered fast to a good structure until the end of the training.

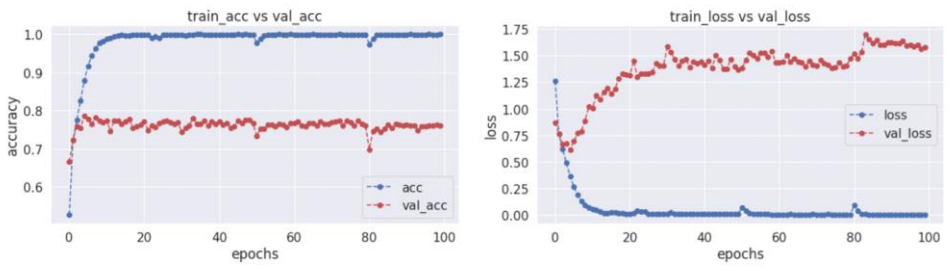


*a)* *b)*

Fig. 4. Results for MIMO training: a) accuracy; b) loss

Fig. 4 b) displays the loss collected from the MIMO architecture during training. The loss plot will indicate a higher loss during the spike, which is expected, the fact that the training and validation loss have slightly different points means that the model does very well on classifying data used for the training but does not meet the same performance on validation data. However, the difference is not a concern, as the difference is very low in the end; this indicates a high rate of correct classifications, and manual testing also proved that.

Fig. 5 a) displays the accuracy collected from the SISO architecture during training. By looking at the accuracy plot, we understand the training went as desired. Data points impacting the curve can be observed, but the difference is so small it cannot indicate a badly structured model.



a) b)

Fig. 5. Results for MIMO training: a) accuracy; b) loss

According to the plot, the structure was well determined throughout the training and an early stopping could have been allowed.

Fig. 5 b) displays the loss collected from the SISO architecture during training. The loss plot indicates a similar behavior as it was described for the MIMO loss evaluation. This means the model performs very well on training data, well on validation data, but not with the same performance for the training data. Similarly, the difference does not indicate any concern.

Each plot indicates that the resulting model will have a good structure and will be able to reach is goal. No significant issues were identified by checking this data. In addition to the resulting metrics, we investigated the resulting models by checking the exact same set of input data and evaluating the results.

The training of the SISO model is more stable than the training of the MIMO one. Both architectures have reached good accuracy on validation data which indicates how the model will perform on new data. Similarly, the loss has been low, concluding most of the samples were correct.

### 5.2. Examples

We selected 25 inputs for testing and evaluated how many reached a correct classification. By correct classification we expect both labels, if an image is a UML diagram or not and what kind of diagram, if applicable, to be classified. The MIMO model matched both labels correctly for 19 out of 25 testing inputs. The SISO model matched 17 out of 25 exact same testing inputs. Additionally, the results of the MIMO take longer to be offered than the SISO ones. The MIMO outputs were displayed on average in 1 minute and 12 seconds. The SISO displayed the results on average in 48 seconds.

Table 1, example 1 represents a UML diagram that is not part of the three specific ones that were selected. On the left the output is provided by the MIMO neural network with a prediction score of 81% and on the right, the output is provided by the SISO neural network with a prediction score of 83%.

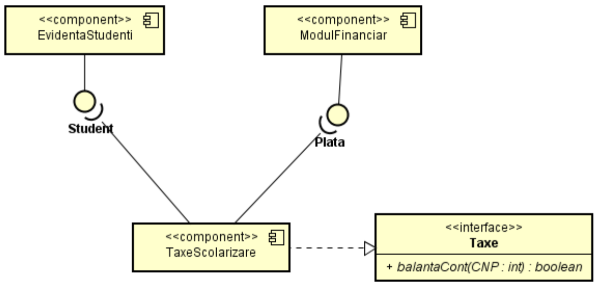
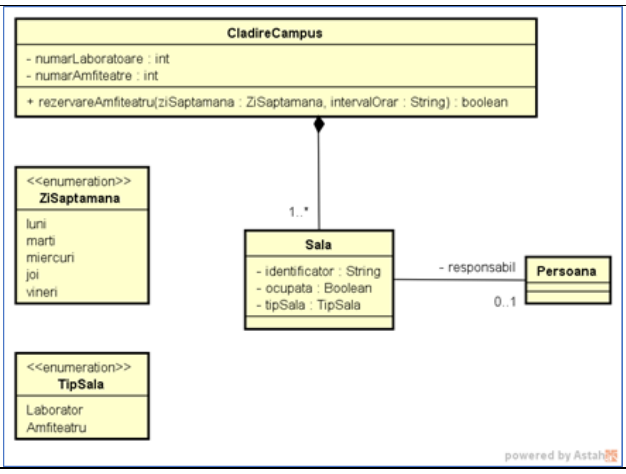
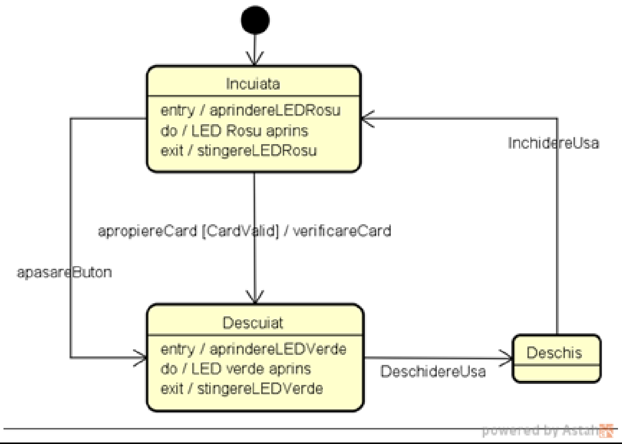
Example 1 does contain a UML diagram. The image was classified as a UML diagram by the MIMO neural network with a prediction score of 88% and by the SISO neural network with a prediction score of 87%.

Example 2 represents a UML diagram that is part of the three specific ones that were selected, in this case a class diagram. The MIMO neural network has a prediction score of 89% as UML and 87% as class diagram. The SISO neural network has a prediction score of 87% as UML and 89% as a class diagram.

Example 3 is a UML diagram that is part of the three specific ones that were selected, in this case a state diagram. The MIMO neural network has a prediction score of 86% as UML and 82% as state diagram. The SISO neural network has a prediction score of 84% as UML and 79% as a state diagram.

*Table 2*

**Results with tested examples**

| Nr. | Image | Prediction Score **Is UML?** | | Prediction Score **Diagram Type** | |
|---|---|---|---|---|---|
| | | MIMO | SISO | MIMO | SISO |
| 1. |  <<component>> EvidentaStudenti<br><<component>> ModulFinanciar<br>Student<br>Plata<br><<component>> TaxeScolarizare<br><<interface>> Taxe<br>+ balantaCont(CNP : int) : boolean | Yes (88%) | Yes (87%) | - | - |
| 2. |  CladireCampus<br>- numarLaboratoare : int<br>- numarAmfiteatre : int<br>+ rezervareAmfiteatru(ziSaptamana : ZiSaptamana, intervalOrar : String) : boolean<br><<enumeration>> ZiSaptamana<br>luni<br>marti<br>miercuri<br>joi<br>vineri<br>1..*<br>Sala<br>- identificator : String<br>- ocupata : Boolean<br>- tipSala : TipSala<br>- responsabil<br>0..1<br>Persoana<br><<enumeration>> TipSala<br>Laborator<br>Amfiteatru<br>powered by Astah | Yes (89%) | Yes (87%) | Class (87%) | Class (89%) |
| 3. |  Incuiata<br>entry / aprindereLEDRosu<br>do / LED Rosu aprins<br>exit / stingereLEDRosu<br>InchidereUsa<br>apropiereCard [CardValid] / verificareCard<br>apasareButon<br>Descuiat<br>entry / aprindereLEDVerde<br>do / LED verde aprins<br>exit / stingereLEDVerde<br>DeschidereUsa<br>Deschis<br>powered by Astah | Yes (86%) | Yes (84%) | State Machine (82%) | State Machine (79%) |

## 6. Conclusions

The goal of the research was to investigate the techniques of artificial intelligence and use them in conjunction with UML modeling. As a starting point, we investigated what is the state of the art in combining these two domains, and which use cases have been highlighted. As a starting point, we identified existing datasets available, and we used one to solve a classification problem with multiple outputs required.

We built two neural networks following different architectures to see which one would perform better. Both can be used based on the type of application that applies for a certain problem. In our case, each model provides good accuracy, though the MIMO model performed better in our experiments.

For the 25 samples that were tested MIMO classified correctly 76% of the samples and SISO classified correctly 68% of the samples. In addition to correct classification, prediction score is an important metric that outlines how confident the model is into the classification. Most machine learning project experiments highlight that 80% accuracy is considered to state the model is highly performant as mentioned in [14]. Among the samples that were classified correctly with a prediction score of 80% or more, 84% matched the criteria for MIMO approach and 77% matched the criteria for SISO approach.

The next step for this research is to decrease the number of wrong classifications and improve accuracy, so the model is highly confident on correct outputs. We will explore expanding the dataset and evaluate techniques for small datasets where performant models were obtained. Additionally, will explore methods to validate the performance as well as other metrics that are highly trusted in literature to support a model that is ready to be used.

## R E F E R E N C E S

[1]. *W. K. Assunção, S.R., Vergilio, R.E. Lopez-Herrejon*, Automatic extraction of product line architecture and feature models from UML class diagram variants, Inf. Softw. Technol., 117. January 2020, DOI:10.1016/j.infsof.2019.106198, Corpus ID: 208032396

[2]. *P.K. Diederik, J. Ba*, Adam: A Method for Stochastic Optimization, CoRR, December 2014, abs/1412.6980 Corpus ID: 6628106

[3]. *A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby*, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, Google Brain Team paper, October 2020, ArXiv, abs/2010.11929, Corpus ID: 225039882

[4]. *H. Gani, M. Naseer, M. Yaqub*, How to Train Vision Transformer on Small-scale Datasets?, October 2022, ArXiv, abs/2210.07240, DOI:10.48550/arXiv.2210.07240, Corpus ID: 252873104.

[5]. *V. Moreno, G. Génova , M. Alejandres, A. Fraga*, Automatic Classification of Web Images as UML Static Diagrams Using Machine Learning Techniques, MDPI, April 2020

[6]. *R. van den Goorbergh, M. van Smede*n, D. Timmerman, B. van Calster, The harm of class imbalance corrections for risk prediction models: illustration and simulation using logistic regression. Journal of the American Medical Informatics Association. September 2022, DOI:10.1093/jamia/ocac093. ISSN 1527-974X.

[7]. *H. Koç, A.M. Erdoğan, Y. Barjakly, S. Peker*, UML Diagrams in Software Engineering Research: A Systematic Literature Review, Proceedings, March 2021, DOI: https://doi.org/10.3390/proceed- ings2021074013

[8]. *J.A.H. López, J.L.C. Izquierdo, J.S. Cuadrado*, ModelSet: a dataset for machine learning in model-driven engineering, Software and Systems Modeling, October 2021, DOI: https://doi.org/10.1007/s10270-021-00929-3

[9]. *S. Raza, L. Cheung, D.B. Epstein, S. Pelengaris, M. Khan, N.M. Rajpoot*, MIMO-Net: A multi-input multi-output convolutional neural network for cell segmentation in fluorescence microscopy images IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017), 337-340, April 2017, DOI:10.1109/ISBI.2017.7950532Corpus ID: 8297945

[10]. *G. Robles, T. Ho-Quang, R. Hebig, M.R.V. Chaudron, M.A. Fernandez*, An extensive dataset of UML models in GitHub, Conference paper, May 2017, DOI: 10.1109/MSR.2017.48

[11]. *K. Simonyan, A. Zisserman*, Very Deep Convolutional Networks for Large-Scale Image Recognition, CoRR, September 2014, abs/1409.1556, Corpus ID: 14124313

[12]. *J. Su, B. Junpeng*, Measuring UML Model Similarity, ICSOFT, April 2018. DOI:10.5220/0004027303190323, Corpus ID: 39094976

[13]. *E. Triandini, R. Fauzan, D.O. Siahaan, S. Rochimah, I.G. Suardika, D. Karolita*, Software Similarity Measurements Using UML Diagrams: A Systematic Literature Review, Journal Ilmiah Teknologi Sistem Informasi, May 2021. https://doi.org/10.26594/REGISTER.V8I1.2248

[14] *S. Tuncer, O. F. Kerimoglu, A. Yurdakul*, An Ensemble Classifier for Predicting Student Performance, IEEE 15th International Conference on Industrial Informatics (INDIN), July 2017