Master’s thesis

Master’s Program in Embedded and Intelligent Systems (120 credits)

Explainable AI by Training Introspection

Computer Science and Engineering (30 credits)

Halmstad — September 11, 2023

Rozhin Dastkarvelayati, Soudabeh Ghafourian
Rozhin Dastkarvelayati, Soudabeh Ghafourian: *Explainable AI by Training Introspection*, Master Thesis, © September 2023

**SUPERVISORS:**
Amira Soliman
Jens Lundström

**LOCATION:**
Halmstad, Sweden

**TIME FRAME:**
January 2023 - September 2023
ABSTRACT

Deep Neural Networks (DNNs) are known as black box algorithms that lack transparency and interpretability for humans. eXplainable Artificial Intelligence (XAI) is introduced to tackle this problem. Most XAI methods are utilized post-training, providing explanations of the model to clarify its predictions and inner workings for human understanding. However, there is a shortage of methods that utilize XAI during training to not only observe the model’s behavior but also exploit this information for the benefit of the model.

In our approach, we propose a novel method that leverages XAI during the training process itself. Incorporating feedback from XAI can give us insights into important features of input data that impact model decisions. This work explores focusing more on specific features during training, which could potentially improve model performance introspectively throughout the training phase. We analyze the stability of feature explanations during training and find that the model’s attention to specific features is consistent in the MNIST dataset. However, unimportant features lack stability. The OCTMNIST dataset, on the other hand, has stable explanations for important features but less consistent explanations for less significant features. Based on this observation, two types of masks, namely fixed and dynamic, are applied to the model’s structure using XAI’s feedback with minimal human intervention. These masks identify the more important features from the less important ones and set the pixels associated with less significant features to zero. The fixed mask is generated based on XAI feedback after the model is fully trained, and then it is applied to the output of the first convolutional layer of a new model (with the same architecture), which is trained from scratch. On the other hand, the dynamic mask is generated based on XAI feedback during training, and it is applied to the model while the model is still training. As a result, these masks are changing during different epochs. Examining these two methods on both deep and shallow models, we find that both masking methods, particularly the fixed one, reduce the focus of all models on the least important parts of the input data. This results in improved accuracy and loss in all models. As a result, this approach enhances the model’s interpretability and performance by incorporating XAI into the training process.
ACKNOWLEDGEMENTS

We would like to express our deep gratitude to our supervisors, Amira Soliman and Jens Lundström, for their support and guidance throughout our work. Their patience and commitment were really helpful in improving our work. We are really grateful for the considerable amount of time they dedicated to guiding us and providing valuable insights.

In addition, many thanks to our examiner, Slawomir Nowaczyk, and our opponent Peyman Mashhadi for their constructive feedback and suggestions.

Regarding,
Rozhin Dastkarvelayati & Soudabeh Ghafourian
Halmstad, Sweden, September 11, 2023
## CONTENTS

1 Introduction 1  
1.1 Problem Definition 2  
1.2 Contribution and Novelty 2  
1.3 Evaluation 3  
1.4 Limitation 5  

2 Literature Review 7  
2.1 Post-hoc XAI Methods 8  
2.1.1 Layer-wise Relevance Propagation (LRP) 9  
2.1.2 Local Interpretable Model-Agnostic Explanation (LIME) 9  
2.1.3 SHapley Additive Explanations (SHAP) 9  
2.1.4 Grad-CAM 10  
2.1.5 Integrated Gradients (IG) 10  
2.2 ML Methods to Improve Model 11  
2.3 XAI Methods to Improve Model 11  
2.4 Conclusion 12  

3 Methodology 15  
3.1 Dataset 15  
3.2 Models Details 15  
3.2.1 CNN Model 16  
3.2.2 ResNet-18 Model 16  
3.2.3 Inception V1 (GoogLeNet) Model 17  
3.3 XAI Method 17  
3.4 XAI Stability 18  
3.5 XAI Stability Evaluation 18  
3.6 Model Optimization using XAI 20  
3.6.1 Fixed Masking 20  
3.6.2 Dynamic Masking 21  

4 Experiments and Results 25  
4.1 Experiments 25  
4.1.1 Evaluating XAI Stability During Training by Using Statistical Metric 25  
4.1.2 Evaluating Model Optimization During Training by Using XAI Feedback to Make Fixed Mask 26  
4.1.3 Evaluating Model Optimization During Training by Using XAI Feedback to Make Dynamic Mask 26  

4.2 Results 27  
4.2.1 Result of Evaluating XAI Stability During Training by Using Statistical Metric 27
4.2.2 Result of Evaluating Model Optimization During Training by Using XAI Feedback to Make Fixed/Dynamic Mask 34
4.2.3 Result of Comparison of Model Optimization and Stability During Training 35
4.2.4 Performance Evaluation of Models Using Masking Methods 39

5 DISCUSSION 41
6 CONCLUSION 47
  6.1 Answers to The Research Questions 48
  6.2 Future Work 50
A APPENDIX 51
  A.1 Evaluation of XAI Stability via Statistical Metrics during Training 51
     A.1.1 CNN Model 51
     A.1.2 ResNet-18 Model 53
     A.1.3 Inception V1 Model 57
  A.2 Evaluation of Model Optimization Using XAI Feedback for Fixed Mask 61
     A.2.1 CNN Model 61
     A.2.2 ResNet-18 Model 64
     A.2.3 Inception V1 Model 68
  A.3 Evaluation of Model Optimization Using XAI Feedback for Dynamic Mask 71
     A.3.1 CNN Model 71
     A.3.2 ResNet-18 model 77
     A.3.3 Inception V1 model 83

BIBLIOGRAPHY 89
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Post-hoc XAI methods</td>
<td>4</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Introspection method</td>
<td>4</td>
</tr>
<tr>
<td>Figure 3</td>
<td>CNN architecture</td>
<td>16</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Fixed mask idea</td>
<td>21</td>
</tr>
<tr>
<td>Figure 5</td>
<td>MNIST-CNN: Variance of highest and lowest accuracy classes</td>
<td>28</td>
</tr>
<tr>
<td>Figure 6</td>
<td>MNIST-CNN: Average of variance for class 0, Global and local accuracy</td>
<td>28</td>
</tr>
<tr>
<td>Figure 7</td>
<td>MNIST-CNN: Average of variance for class 1, Global and local accuracy</td>
<td>29</td>
</tr>
<tr>
<td>Figure 8</td>
<td>MNIST-CNN: Average of variance for class 4, Global and local accuracy</td>
<td>29</td>
</tr>
<tr>
<td>Figure 9</td>
<td>MNIST-CNN: Average of variance for class 7, Global and local accuracy</td>
<td>30</td>
</tr>
<tr>
<td>Figure 10</td>
<td>MNIST-CNN: Average of variance for class 9, Global and local accuracy</td>
<td>30</td>
</tr>
<tr>
<td>Figure 11</td>
<td>MNIST-CNN: Average of variance for class 2, Global and local accuracy</td>
<td>31</td>
</tr>
<tr>
<td>Figure 12</td>
<td>MNIST-CNN: Average of variance for class 3, Global and local accuracy</td>
<td>32</td>
</tr>
<tr>
<td>Figure 13</td>
<td>MNIST-CNN: Average of variance for class 5, Global and local accuracy</td>
<td>32</td>
</tr>
<tr>
<td>Figure 14</td>
<td>MNIST-CNN: Average of variance for class 6, Global and local accuracy</td>
<td>33</td>
</tr>
<tr>
<td>Figure 15</td>
<td>MNIST-CNN: Average of variance for class 8, Global and local accuracy</td>
<td>33</td>
</tr>
<tr>
<td>Figure 16</td>
<td>MNIST-CNN: Voted mask</td>
<td>34</td>
</tr>
<tr>
<td>Figure 17</td>
<td>MNIST-CNN: Binary mask and histogram of class 0 as highest and 9 as lowest class accuracy</td>
<td>34</td>
</tr>
<tr>
<td>Figure 18</td>
<td>MNIST: Comparison of highest training and validation accuracy and variance over three models</td>
<td>35</td>
</tr>
<tr>
<td>Figure 19</td>
<td>MNIST: Comparison of lowest training and validation accuracy and variance over three models</td>
<td>36</td>
</tr>
<tr>
<td>Figure 20</td>
<td>OCTMNIST: Comparison of highest training and validation accuracy and variance over three models</td>
<td>37</td>
</tr>
</tbody>
</table>
Figure 21  OCTMNIST: Comparison of lowest training and validation accuracy and variance over three models 38
Figure 22  Comparison of training/validation accuracy among different methods (without mask, with fixed and dynamic mask) and datasets 39
Figure 23  Comparison of test accuracy among different methods (without mask, with fixed and dynamic mask) and datasets 40
Figure 24  OCTMNIST-CNN: Variance of highest and lowest accuracy classes 51
Figure 25  OCTMNIST-CNN: comparison of highest and lowest training accuracy classes 51
Figure 26  OCTMNIST-CNN: variance of variance over window epochs 52
Figure 27  MNIST-ResNet-18: Comparison of highest and lowest training accuracy classes 53
Figure 28  MNIST-ResNet-18: Variance of variance over window epochs 54
Figure 29  MNIST-ResNet-18: Variance of highest and lowest accuracy classes 54
Figure 30  OCTMNIST-ResNet-18: Variance of highest and lowest accuracy classes 55
Figure 31  OCTMNIST-ResNet-18: Comparison of highest and lowest training accuracy classes 55
Figure 32  OCTMNIST-ResNet-18: Variance of variance over window epochs 56
Figure 33  MNIST-InceptionV1: Variance of highest and lowest accuracy classes 57
Figure 34  MNIST-InceptionV1: Comparison of highest and lowest training accuracy classes 57
Figure 35  MNIST-InceptionV1: Variance of variance over window epochs 58
Figure 36  OCTMNIST-InceptionV1: Variance of highest and lowest accuracy classes 59
Figure 37  OCTMNIST-InceptionV1: Comparison of highest and lowest training accuracy classes 59
Figure 38  OCTMNIST-InceptionV1: Variance of variance over window epochs 60
Figure 39  MNIST-CNN-Fixed mask: Average variance and the average of average variance for highest and lowest accuracy class 61
Figure 40  OCTMNIST-CNN:Voted mask 62
Figure 41  OCTMNIST-CNN: Binary mask and histogram of class 3 as highest and 2 as lowest class accuracy 62
Figure 42 OCTMNIST-CNN-Fixed mask: Average variance and the average of average variance for highest and lowest accuracy classes

Figure 43 MNIST-ResNet-18: Voted mask

Figure 44 MNIST-ResNet-18: Binary mask and histogram of class 0 as highest and 4 as lowest class accuracy

Figure 45 MNIST-ResNet-18-Fixed mask: Average variance and the average of average variance for highest and lowest accuracy classes

Figure 46 OCTMNIST-ResNet-18: Voted mask

Figure 47 OCTMNIST-ResNet-18: Binary mask and histogram of class 0 as highest and class 2 as lowest class accuracy

Figure 48 OCTMNIST-ResNet-18-Fixed mask: Average variance and the average of average variance for highest and lowest accuracy classes

Figure 49 MNIST-InceptionV1: Voted mask

Figure 50 MNIST-InceptionV1: Binary mask and histogram of class 4 as highest and 8 as lowest accuracy class

Figure 51 MNIST-InceptionV1-Fixed mask: Average variance and the average of average variance for highest and lowest accuracy classes

Figure 52 OCTMNIST-InceptionV1: Voted mask

Figure 53 OCTMNIST-InceptionV1: Binary mask and histogram of class 3 as highest and class 2 as lowest accuracy class

Figure 54 OCTMNIST-InceptionV1-Fixed mask: Average variance and the average of average variance for highest and lowest accuracy classes

Figure 55 MNIST-CNN: Comparing the accuracy of Training/validation fixed and dynamic mask

Figure 56 MNIST-CNN: Comparing the loss of Training/validation fixed and dynamic mask

Figure 57 MNIST-CNN: Number of digits with low/high background variance

Figure 58 MNIST-CNN: Variance of highest and lowest accuracy classes

Figure 59 MNIST-CNN-Dynamic mask: Average variance and the average of average variance for highest and lowest accuracy classes

Figure 60 OCTMNIST-CNN: Comparing the accuracy of Training/validation fixed and dynamic mask

Figure 61 OCTMNIST-CNN: Comparing the loss of Training/validation fixed and dynamic mask
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>OCTMNIST-CNN: Number of digits with low/high background variance</td>
<td>75</td>
</tr>
<tr>
<td>63</td>
<td>OCTMNIST-CNN: Variance of highest and lowest accuracy classes</td>
<td>75</td>
</tr>
<tr>
<td>64</td>
<td>OCTMNIST-CNN-Dynamic mask: Average variance and the average of average variance for highest and lowest accuracy classes</td>
<td>76</td>
</tr>
<tr>
<td>65</td>
<td>MNIST-RestNet-18: Comparing the accuracy of Training/validation fixed and dynamic mask</td>
<td>77</td>
</tr>
<tr>
<td>66</td>
<td>MNIST-RestNet-18: Comparing the loss of Training/validation fixed and dynamic mask</td>
<td>77</td>
</tr>
<tr>
<td>67</td>
<td>MNIST-RestNet-18: Number of digits with low/high background variance</td>
<td>78</td>
</tr>
<tr>
<td>68</td>
<td>MNIST-RestNet-18: Variance of highest and lowest accuracy classes</td>
<td>78</td>
</tr>
<tr>
<td>69</td>
<td>MNIST-RestNet-18-Dynamic mask: Average variance and the average of average variance for highest and lowest accuracy classes</td>
<td>79</td>
</tr>
<tr>
<td>70</td>
<td>OCTMNIST-ResNet-18: Comparing the accuracy of Training/validation fixed and dynamic mask</td>
<td>80</td>
</tr>
<tr>
<td>71</td>
<td>OCTMNIST-ResNet-18: Comparing the loss of Training/validation fixed and dynamic mask</td>
<td>80</td>
</tr>
<tr>
<td>72</td>
<td>OCTMNIST-ResNet-18: Number of digits with low/high background variance</td>
<td>81</td>
</tr>
<tr>
<td>73</td>
<td>OCTMNIST-ResNet-18: Variance of highest and lowest accuracy classes</td>
<td>81</td>
</tr>
<tr>
<td>74</td>
<td>OCTMNIST-ResNet-18-Dynamic mask: Average variance and the average of average variance for highest and lowest accuracy classes</td>
<td>82</td>
</tr>
<tr>
<td>75</td>
<td>MNIST-InceptionV1 Comparing the accuracy of Training/validation fixed and dynamic mask</td>
<td>83</td>
</tr>
<tr>
<td>76</td>
<td>MNIST-InceptionV1: Comparing the loss of Training/validation fixed and dynamic mask</td>
<td>83</td>
</tr>
<tr>
<td>77</td>
<td>MNIST-InceptionV1: Number of digits with low/high background variance</td>
<td>84</td>
</tr>
<tr>
<td>78</td>
<td>MNIST-InceptionV1: Variance of the highest and lowest accuracy classes</td>
<td>84</td>
</tr>
<tr>
<td>79</td>
<td>MNIST-InceptionV1-Dynamic mask: Average variance and the average of average variance for highest and lowest accuracy classes</td>
<td>85</td>
</tr>
<tr>
<td>80</td>
<td>OCTMNIST-InceptionV1: Comparing the accuracy of Training/validation fixed and dynamic mask</td>
<td>86</td>
</tr>
<tr>
<td>81</td>
<td>OCTMNIST-InceptionV1: Comparing the loss of Training/validation fixed and dynamic mask</td>
<td>86</td>
</tr>
</tbody>
</table>
Figure 82  OCTMNIST-InceptionV1: Number of digits with low/high background variance  87
Figure 83  OCTMNIST-InceptionV1: Variance of the highest and lowest accuracy classes  87
Figure 84  OCTMNIST-InceptionV1-Dynamic mask: Average variance and the average of average variance for highest and lowest accuracy classes  88
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>XAI stability during training</td>
<td>19</td>
</tr>
<tr>
<td>Calculate Masks</td>
<td>23</td>
</tr>
<tr>
<td>Using XAI to apply dynamic masking during training</td>
<td>24</td>
</tr>
<tr>
<td>ACRONYMS</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>DNN</td>
<td>Deep Neural Network</td>
</tr>
<tr>
<td>DL</td>
<td>Deep Learning</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>XAI</td>
<td>eXplainable Artificial Intelligence</td>
</tr>
<tr>
<td>LRP</td>
<td>Layer-wise Relevance Propagation</td>
</tr>
<tr>
<td>LIME</td>
<td>Local Interpretable Model-Agnostic Explanation</td>
</tr>
<tr>
<td>SHAP</td>
<td>SHapley Additive exPlanation</td>
</tr>
<tr>
<td>Grad-CAM</td>
<td>Gradient-weighted Class Activation Mapping</td>
</tr>
<tr>
<td>CAM</td>
<td>Class Activation Mapping</td>
</tr>
<tr>
<td>IG</td>
<td>Integrated Gradients</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectified Linear Unit</td>
</tr>
<tr>
<td>EB</td>
<td>Excitation Backprop</td>
</tr>
</tbody>
</table>
In recent years, advances in Machine Learning (ML) have led to the growing popularity of neural networks like Deep Neural Networks (DNNs). Many machine learning algorithms have shown success in both scientific and industrial settings, with the goal of simplifying tasks and improving decision-making efficiency [19]. To achieve better performance, Deep Learning (DL) models, such as Convolutional Neural Networks (CNNs), are often used. These models are successful due to their efficient learning algorithms and extensive parametric space, which can comprise hundreds of layers and millions of parameters. However, the complexity of these models has led to the emergence of the concept of black boxes, which are difficult for humans to interpret or understand [8, 9].

In certain applications where humans need to be able to trust and understand the model’s output, this lack of transparency can cause problems. Additionally, as DL models are frequently used in the healthcare and finance industries to make critical decisions, understanding and trusting their output is crucial. Meanwhile, various stakeholders in Artificial Intelligence (AI) are becoming increasingly concerned about the lack of transparency as ML models are dramatically used to make important predictions [16]. As systems are focused only on performance, it is generally thought that they will become opaque; on the other hand, by understanding a system better, deficiencies such as biased training models can be identified and corrected [6].

To address this problem, eXplainable Artificial Intelligence (XAI) methods are used. XAI is a field of AI research that focuses on techniques that make ML models interpretable and understandable for humans [15]. In other words, XAI is a machine learning tool that can explain ML model predictions or decisions clearly and transparently. ML models can be trained to be self-explainable while many tools are used as post-hoc methods to explain already trained models [4, 6, 19].

At the moment, CNNs are regarded as the most advanced models in every fundamental area of computer vision, including image classification, object detection, and instance segmentation. There exist different types of models with extremely complex structures that are hard to explain, but in comparison to other types of models, CNNs are easier to explain because they use visual data, so humans can comprehend these types of data [6]. Currently, most XAI methods are being used to explain model decisions rather than get for example better model performance [25]. However, Weber et al. [25] investigate
some XAI methods to improve ML models such as dropout and XAI feedback techniques [33, 34] that we will discuss in Section 2.3.

1.1 PROBLEM DEFINITION

Our primary objective in this project is to propose an introspection method, which can be defined as a technique that monitors the model’s training progress over time and provides insights into its learning behavior.

As a result, we can better understand what the model is learning and/or how it is learning. The main purpose is to investigate whether there is a possibility to monitor the training process of CNNs and explain the evolution of the training in the model. If so, we have access to control other aspects of the model, like accuracy, loss, convergence, and learning stability, before getting a prediction. Furthermore, monitoring the model’s learning behavior through XAI gives us the potential to ensure that the model is acquiring valuable knowledge.

To solve current challenges and improve related research, we confront the following research questions:

1. Considering the models produced as outputs from every step in training, do these models have consistent explanations (XAI stability)?

2. How could explanations be utilized in the learning process to enhance model performance and internal states, such as final accuracy and loss?

It’s important to clarify that our primary objective is to leverage explanations generated during the training phase to improve model performance introspectively. To achieve this, we must assess the stability of these explanations throughout training to determine whether they remain consistent. This leads us to our first research question: monitoring the consistency of these explanations as the model is being trained. Intuitively, stable explanations can guide the feature engineering process, leading to our second research question. By tracking XAI stability and features highlighted based on that, we can assess their reliability and determine their trustworthiness. While not all models might require stable explanations, our technique specifically relies on this stability. Thus, assessing the stability of explanations is essential for our approach.

1.2 CONTRIBUTION AND NOVELTY

Previous studies on XAI methods have shown that experts consider the accuracy and explanation of the training model as two separate processes, i.e., model developers seek to produce the most accurate
possible model and don’t focus on explainability, and the XAI community provides the tools to interpret it regardless of the performance of the ML model. Further, it is also important to ask whether some predictions can be trusted after explaining the model. As we know, explanations summarize potentially complex decision processes. Therefore, different decision strategies may have the same explanation, whether they are valid or incorrect. The explanation methods may also include biases or approximations that cause information loss. As a result, a trustworthy model must be verified [19].

Our primary goal is to assess the feasibility of an XAI method that can positively influence the model during its training phase. To realize this objective, we focus on the integration of information extracted from XAI tools. A crucial first step involves determining the trustworthiness of explanations by evaluating the stability of explanations during training. Unlike traditional post-hoc XAI methods, our approach is centered on integrating this valuable information directly during the model training phase. This integration can potentially enhance the model training outcomes, yielding increased accuracy and minimizing loss. Also, Our approach involves feature augmentation guided by the model outcome.

Using those tools, we introduce a new masking method based on XAI feedback and the features highlighted by Grad CAM and continue updating the model. This method generates masks in two specific ways. A “fixed mask” that is created once the model completes its training and a “dynamic mask” that is continually adjusted and recreated at different stages during the training process. Both strategies are generated based on XAI feedback and function with minimal human intervention. Our work begins with the CNN using the MNIST dataset. Following this, we broaden our investigation by comparing our outcomes with those from advanced models, such as ResNet-18 and Inception V1, and further expand our experiments with the OCTMNIST dataset.

We compare the existing XAI methods with our method in Figure 1 and Figure 2. In Figure 1 the post-hoc explanation methods, do not go through training and the XAI algorithm interprets the result of DNN models for humans after training to make it more understandable. However in Figure 2, we propose a method in which the training process is actively monitored by the XAI algorithm, which is integrated during training to generate feedback. This feedback can then be leveraged to enhance the model’s performance. We refer to this approach as "introspection".

1.3 EVALUATION

There is currently no research-wide agreement on how to assess a model’s interpretability, the correctness of an explanation, or how to
compare methods. Since our research questions involve model properties and XAI properties, we consider the evaluation approach from two points of view including Quantitative analysis \([4]\) (model performance including accuracy and loss) and Qualitative analysis (quality of explanation such as human-centred evaluations by experts) \([13, 24]\).

Weber et al. \([25]\) evaluate the impact of XAI methods on the performance of their models, comparing models with and without XAI integration. To assess the models’ performance, they train them on the trained model and test them on out-of-domain datasets. The results were compared to the trained base models without XAI integration. Model predictions were computed without explanations at test time.

There are some criteria \([13, 18]\) that can be used to assess a method’s or explanation’s quality. Several important properties of explanation methods determine their usefulness and applicability, such as their expressive power, transparency, portability, and computational complexity. Besides, individual explanations have several important properties that determine their quality and usefulness, including accuracy,
fidelity, consistency, stability, comprehensibility, certainty, degree of Importance, novelty, and representativeness. One attribute is stability. Stability is a measure of whether a model can explain similar inputs similarly; unstable outcomes may occur due to significant variability in the approach used for explanation [13, 18, 24].

To evaluate our work, we examine the XAI properties, such as stability, and compare the heatmap output during training. We consider the stability across different epochs; simultaneously, the loss of trains is evaluated. Additionally, we inspect our results by the visualization. Besides visual evaluation, we also consider statistical metrics such as variance to assess explanation stability more accurately. Additionally, we visualize the accuracy of each class and compare it with the overall accuracy during training.

Evaluating the results related to our second research question, and in accordance with [25], we incorporate XAI feedback into the training process to assess model properties such as accuracy and loss. This evaluation is performed both with and without the application of masks that are generated based on XAI feedback for CNN models with minimal human intervention. Moreover, we extend our method to deeper models like ResNet-18 and Inception V1, comparing their performance with each other.

In summary, we answer the research questions by evaluating the impact of masks on model performance and stability on training and validation accuracies, as well as the average variance for the highest and lowest training accuracy classes, both with and without masks.

1.4 Limitation

It has been decided to limit the project in some ways. We start with image data as our first Explainable AI dataset. By doing this, we can concentrate on the explanation model rather than dealing with the dataset’s complexity. Later, we can expand our dataset to image data related to clinical systems, or other types of data such as tabular. Furthermore, we do not compare our method with existing feature selection methods, as most of these methods leverage domain knowledge and prior information, while our method uses different masking strategies guided by XAI outcomes as a strategy for feature selection.
In our literature review, we aim to discuss the current research on two topics: the application of XAI to improve the transparency and interoperability of ML models and the development of novel techniques for improving the performance of ML models. Explainable AI methods can be categorised as follows:

- **Ante-hoc**: Ante-hoc, also known as "transparent" [4, 6] or "white box or glass box" [4], or "self-explanation" [19], is a type of XAI method that is naturally and intrinsically interpreted and transparent, such as Linear regression, Decision trees, K-Nearest Neighbors, Rule-based Learners, Generalised Additive Models, Bayesian Models [6]. They can be trained directly from a supervised data set to perform an ML problem, or they can be used to approximate a black-box model on some representative input distribution. Although self-explainable models can be effective for many real-world activities, their utility decreases when the aim is to explain the strategy of certain existing black-box models. In such a case, the difficult task of carefully recreating the black-box model for every potential input and perturbation would need to be accomplished while being limited by the predetermined interpretable structure [19].

There are three levels of transparency including algorithmic transparency, decomposability, and simulatability [6].

1. Simulatability: It is easily understandable through text and visualization by humans.
2. Decomposability: It clarifies the various components of a model, such as input, parameter, and calculation. So we can understand the behavior of the model better.
3. Algorithmic: the process of the model can be understood how to get that output from that by the user. There is a restriction for algorithm transparency that the model must be explainable mathematically.

- **Post-hoc**: Post-hoc methods, also referred to as "black box" [4] techniques, are applied to trained models and are inherently opaque [5, 15, 19]. These XAI methods may be either model-agnostic or model-specific. Model-agnostic includes any ML models while model-specific can be divided to post-hoc explainability of shallow ML models (Tree ensembles, Random forests, and multiple classifier systems) and deep neural networks such as
convolutional neural networks, recurrent neural networks, and hybrid neural networks. Post-hoc XAI techniques can be further categorized based on their outcomes or results. These categories are based on one of the most common ways humans describe systems and processes. Text explanations, visualizations, local explanations, explanations by example, explanations by simplification, and feature relevance are all examples of approaches within these categories [6].

- Model-agnostic or model-specific: Model-agnostic XAI is a form of Explainable AI that focuses on establishing general strategies for explaining any AI model’s decisions, regardless of its architecture or training method. But model-specific XAI focuses on specific ML models. Some model-agnostic approaches, such as SHAP [12] (SHapley Additive Explanations, which provides a priority rating for each feature for each prediction), are commonly employed to describe DL models [4, 6]. Model-agnostic interpretation techniques are more flexible and can be applied to any type of model. Transparent interpretable models always fall into the category of model-specific [13].

- Global or local: These XAI techniques can be divided into two categories which are based on their scope. We can anticipate the expected outcome by understanding the model’s behavior and logic from a global perspective. The reasons for a particular prediction are given as local explanations to justify why the model made a specific decision in that instance [4]. A model-agnostic interpretation technique can be classified as global or local. Global methods explain how features influence average prediction. A local method, on the other hand, attempts to understand a specific prediction [13].

2.1 post-hoc XAI methods

This section will focus on describing post-hoc and deterministic XAI methods, which differ from sampling-based or optimization-based methods. Specifically, these methods will always deliver the same explanation when provided with the same input and a trained neural network. The explanation of a vision problem can take the form of a heatmap. This is when every pixel in a source image is assigned a relevant value or score, indicating its relative importance to the final conclusion. Class Saliency Map, Gradient-weighted Class Activation Mapping (Grad-CAM), Gradient Input, Integrated Gradients (IG), Layer-wise Relevance Propagation (LRP), Excitation Backprop (EB), and Guided Backpropagation are some methods that provide such heatmaps in a deterministic and transparent manner on an already trained neural network (post-hoc explanation methods). Such
explanation approaches are usually based on gradient integration or a specific backward pass over the network [5, 19]. Another class of explanation approaches incorporates some unpredictability into the heatmap computation: they necessitate the generation of additional perturbed training data samples or the solution of an ante-hoc optimization problem in order to produce a single heatmap [5]. The following sections will explore popular XAI methods, such as two local model-agnostic approaches (Local Interpretable Model-Agnostic Explanations (LIME) and SHAP), two global model-agnostic approaches (Grad-CAM and IG), and a local model-specific method (LRP):

2.1 Post-hoc XAI Methods

2.1.1 Layer-wise Relevance Propagation (LRP)

LRP is a technique for interpreting the predictions made by artificial neural networks. It works by assigning a relevance score to each input feature, which represents its contribution to the final prediction. The relevance scores are propagated backward through the network, layer by layer, using a set of propagation rules [5, 19].

2.1.2 Local Interpretable Model-Agnostic Explanation (LIME)

LIME [17] is a technique that uses interpretable surrogate models to explain the predictions of black-box machine learning models. LIME’s goal is to provide a comprehensible explanation for why a model made a specific prediction on a specific instance. The steps for creating a surrogate model with LIME are as follows: The example for which the prediction must be explained is chosen. LIME randomly perturbs instances within a range determined by the type of feature. The importance of each feature is calculated using the difference between the perturbed and original instances. A surrogate model is used to train a simpler, more interpretable model, such as linear regression or decision trees.

To explain the prediction of the original instance, the surrogate model is used. A surrogate model’s feature weight represents the contribution of each feature to its prediction. LIME has been used effectively in a variety of areas, including text classification, image classification, and time series prediction [17].

2.1.3 Shapley Additive Explanations (SHAP)

SHAP [12] is a methodology for explaining the output of any machine learning algorithm. It is built on the cooperative game theory idea of Shapley values [13] and offers a method to attribute a model’s prediction to its input features. One of the SHAP model’s major benefits is its ability to manage complex models and high-dimensional data. It can explain any model, including deep neural networks, de-
cision trees, and random forests, precisely and consistently. Furthermore, SHAP values are globally interpretable, which means they can be used to comprehend the model’s behavior across the complete dataset rather than just for individual predictions [12].

2.1.4 Grad-CAM

A technique called Grad-CAM [21] provides visual explanations of deep neural networks. By combining the gradients of the output class score with the last convolutional layer feature maps, it creates a weighted combination of the feature maps, creating a heatmap that highlights the regions of the input image that were most relevant in predicting the output class. Like Class Activation Mapping (CAM), which is a technique proposed by Zhou et al. [31], this method can accurately identify important features in images that are relevant for classification, even when the images contain occlusions or adversarial disturbances. Additionally, both CAM and this method are more interpretable and provide greater insight into the workings of the model.

In contrast to CAM [31], which is a computer vision technique that highlights the important regions of an input image by computing the weighted average of the feature maps in the last convolutional layer of the CNN, Grad-CAM is more accurate and readable and can identify subtle features in images that are relevant for classification, regardless of occlusions or adversarial disturbances. There are many potential applications for Grad-CAM, including medical imaging and autonomous driving. Grad-CAM can be used for generating visual explanations of deep neural networks, and it has the potential to be used in a wide range of applications where a high level of interpretability is required.

A lightweight technique of Grad-CAM [20] enhances the interpretability and explainability of classification networks. This technique utilizes Grad-CAM to generate visual explanations and also includes an extra output that produces a binary mask representing the significant regions of the input image. In addition, this method is specifically designed to be computationally efficient, making it suitable for deployment on resource-constrained devices. On the other hand, Grad-CAM was not specifically designed with computational efficiency in mind and maybe computationally expensive for certain applications.

2.1.5 Integrated Gradients (IG)

The goal of the deep neural network explaining technique known as IG [22] is to delve into the relative contributions of each input feature to the network’s output. It is a global and model-agnostic technique that can be used with any differentiable model, including transform-
ers, recurrent neural networks, and convolutional neural networks. To use Integrated Gradients, it first calculates the output of the network’s gradients with regard to its input features, then integrates those gradients along a path from a baseline input to the input itself. To calculate feature importance, a baseline image is used.

2.2 ML METHODS TO IMPROVE MODEL

The concept of integrating human knowledge to improve the performance of machine learning models is not new [25]. Methods like Support Vector Machines have been using expert feedback to enhance learning and reasoning [30]. Krizhevsky et al. [10] discuss the training of a large CNN model on subsets of the ImageNet dataset used in the ILSVRC-2010 and ILSVRC-2012 competitions. They achieved better results than the previous state-of-the-art method. Non-saturating neurons and efficient GPU implementation were used to speed up training. A regularization method called "dropout" was also used to reduce overfitting. Additionally, Zou et al. [32] focus on the problem of training deep fully connected neural networks using gradient descent for binary classification with Rectified Linear Unit (ReLU) activation function and cross-entropy loss function. The authors prove that with proper random weight initialization, gradient descent can find the global minimum of the training loss for over-parameterized deep ReLU networks. They demonstrate that Gaussian random initialization followed by gradient descent produces a sequence of iterates that stay inside a small perturbation region centered at the initial weights, ensuring the global convergence of gradient descent. Compared to other works, the authors’ result relies on milder overparameterization conditions and enjoys a faster global convergence rate of gradient descent for training deep neural networks. Wu et al. [26] propose a method for hyperparameter tuning based on Gaussian processes and Bayesian optimization. They show that their method can find the best hyperparameters for widely used machine learning models. The proposed method demonstrated that it can significantly improve the efficiency of machine learning.

2.3 XAI METHODS TO IMPROVE MODEL

Recently, explanations have been added to improve machine learning models. Improving different model properties with XAI such as performance, reasoning, equality, etc. Weber et al. [25] differentiate techniques based on the training loop component they augment. XAI enhancement involves providing explanations that reveal how the model makes decisions and behaves. These explanations can improve models by enhancing various aspects of the training process or adjusting the model itself. This includes data, features, loss, gradients,
or model augmentation. Since ML models rely heavily on data, data augmentation techniques can benefit the model.

Dropout is an effective technique to reduce overfitting as it encourages the model to explore different solutions to a given problem. Typically, random neurons are dropped out, leading to stochastic exploration. However, XAI can make a more intelligent selection by dropping out the most significant neurons. This ensures that the most critical path through the network is consistently disrupted, significantly increasing generalization \[25\]. Zunino et al. \[33\] propose a technique that drops out those neurons that contribute the most to the decision-making process during training. They utilize EB method during training as an explanation method to generate saliency maps that quantify the importance of class-specific neurons for a given input image. EB guides the dropout process by dropping out excitatory connections with a higher probability for those neurons with higher saliency values. Using this method, the network is forced to learn alternative paths and exhibit plasticity behavior. The authors demonstrate that the proposed technique leads to better performance and generalization ability.

Zunino et al. \[34\] suggest a method that improves domain generalization. The approach involves training an interpretable deep classification model that learns to identify domain-invariant features. Using XAI feedback during training, the model focuses on image regions that correspond directly to the ground-truth object. They demonstrate that this method leads to improved generalization to new domains without compromising performance in the original domain.

Unlike previous studies, we use XAI and XAI feedback in a new way. We use test data during training to get feedback with minimum human intervention and improve the learning process as well as our model’s accuracy and loss. Our approach also involves feature augmentation guided by the model outcome. We apply the mask based on the features highlighted by Grad CAM and continue updating the model. We test this on MNIST and OCTMNIST datasets.

### 2.4 Conclusion

The improvement of complex machine learning models for various applications is challenging because of their black-box nature and lack of decision-making information. Explanations can provide additional information that improves these properties. This includes knowledge about important feature representations, reducing training time, increasing accuracy, and identifying key neurons and filters to improve model efficiency \[25\].

As a result of the literature review, the proposed method is closer to post-hoc XAI approaches; hence, we use them with customization, especially Grad-CAM; In general, methods that use gradients calculate
faster than other methods [13]. For this reason, we start our experiment with the Grad-CAM method.
METHODOLOGY

In the methodology section, we present two distinct methodologies to address the research questions outlined in the study that involved introspection of XAI, and model property. Section 3.1 explains our data, followed by the definition of our CNNs models in Section 3.2. We then discuss our XAI method in Section 3.3. Sections 3.4 and 3.5 cover the method for XAI’s stability and assessment. Finally, the last section describes the method for optimizing the model using XAI.

1. Considering the models produced as outputs from every step in training, do these models have consistent explanations (XAI stability)?

2. How could explanations be utilized in the learning process to enhance model performance and internal states, such as final accuracy and loss?

3.1 DATASET

We start with an image dataset and select the MNIST dataset [11], which consists of handwritten digits 0-9. The choice of the MNIST dataset is motivated by its relative simplicity compared to other image datasets, allowing us to concentrate on developing and illustrating our novel masking methods without additional complexities. Furthermore, this dataset makes it easier to interpret the results and comprehend the model’s behavior. As the MNIST shape is 28x28, we preprocess the data to resize it to 48x48.

We divide the dataset into 50,000 training images, 10,000 testing images, and 10,000 validation images.

For the preprocessing step, we normalize the data in Python using the min-max scaler in our experiments.

To thoroughly evaluate our method, we plan to test it on a different dataset with varying patterns. Therefore, we train our models with one of the MedMNIST datasets [28, 29], which is OCTMNIST; the images are derived from Optical Coherence Tomography of the retina (Retinal OCT). This dataset is divided into 97,477 training images, 10,832 validation images, and 1000 test images.

3.2 MODELS DETAILS

We used three models, including CNN, Inception V1, and ResNet-18, which are shallow and deeper models, respectively.
### 3.2.1 CNN Model

We implement the high-accuracy CNN model with a sequential architecture. This approach ensures a proper training process and highly accurate predictions, allowing us to concentrate on developing and refining our novel masking method. The model comprises three 2D convolutional layers with filter sizes of 3x3 and ReLU activation functions and uses two 2x2 max-pooling layers to reduce spatial dimensions. To prevent overfitting, two dropout layers are utilized, the first with a rate of 0.5 after the initial dense layer of 512 units and the second with a rate of 0.2 after the subsequent dense layer of 128 units. The model’s final dense layer employs the softmax activation function to output a probability distribution over ten classes. We train our model \cite{2} with batch-size 128, using a sparse categorical cross entropy loss function and SGD optimizer with a learning rate of 0.001. The input dimensions are 48x48. The CNN architecture details are visualized in Figure 3.

![Figure 3: CNN architecture](image)

### 3.2.2 ResNet-18 Model

We implement ResNet-18 \cite{27} which uses "skip connections". Skip connections, operate by forwarding the output of a certain layer in the neural network directly to one or more downstream layers, effectively bypassing the intermediate layers. This approach is crucial in preventing the vanishing gradient issue that often arises during the training of deep neural networks \cite{23}. The architecture begins with a convolutional layer with 64 filters, followed by batch normalization and the ReLU activation function. Then followed by the max pooling layer. The core of the ResNet-18 model consists of four layers, each containing two residual blocks. These layers have 64, 128, 256, and 512 filters, respectively. Each block within these layers consists of two sets of operations: a 3x3 convolution, batch normalization, and a ReLU activation function. The model uses downsample layers when the stride is not one or the number of input planes doesn’t match the number of filters. These layers consist of a convolutional layer and a batch normalization layer. Moreover, The model concludes with a global
average pooling layer and a dense layer with several units equal to the number of classes.

3.2.3 Inception V1 (GoogLeNet) Model

In our methodology, we also implement the Inception V1 architecture [3]. Inception V1, also known as GoogLeNet, introduces a module called the "Inception module", which is designed to address the problem of computational cost by reducing the dimensionality of the input feature maps before applying the convolution operation.

In this architecture, there are several versions of Inception modules, such as Inceptionv1_Module, Inceptionv2_Module, Inception_Module_A, Inception_Module_B, and Inception_Module_C. Each module applies convolution operations with different kernel sizes (1x1, 3x3, 5x5, etc.) to the same input in parallel, extracts features independently, and then concatenates them together. Each Inception module starts with a 2D Convolutional Block that includes batch normalization and a ReLU activation function. Following this, different branches are created that consist of convolutional layers with different kernel sizes. One of these branches applies max pooling. The outputs of all the branches are then concatenated together. Inception also includes Reduction blocks (Reduction_Block_A and Reduction_Block_B). These are a form of inception modules with strides of 2 in some branches, which decrease the height and width of the input by a factor of 2, effectively making it a form of downsampling. The final architecture also includes a classifier, which consists of a Dense layer with a softmax activation function that outputs the class probabilities in a classification setting.

3.3 XAI Method

As highlighted in Section 1.1, XAI is used for our methodology. As outlined in Section 2.4, we have employed Grad-CAM [21] as our XAI technique. The Grad-CAM calculates the gradients of the target class score with respect to the feature maps of a high-level layer (such as the last layer) in the network, which captures high-level semantic information about the input. These gradients indicate the importance of each feature map for the predicted class. Therefore in the Grad-CAM, we can obtain gradients and backpropagate to a specific layer. Selecting a layer closer to the input results in features closer to raw data. Deeper layers still perform feature engineering but focus on learning representation. Various factors must be considered when selecting the layer for GradCAM [see Chapter 5]. Then use a heatmap to show the important regions for the model’s prediction. This method enhances our model’s interpretability by enabling us to visually highlight and understand the key features that the model
focuses on during its decision-making process, thereby providing a valuable foundation for refining the model and enhancing its overall performance.

3.4 XAI Stability

We implement CNN and include Grad-CAM [7] during training to display heatmaps for each epoch. This helps us assess the stability of the generated explanations. Higher stability in the explanations indicates that the same spatial locations in the input image are consistently more important for predicting the class throughout the training. The visualization is performed using test data as input to the model. We add the metric variance to our model to calculate the variance of each pixel of the heatmap output during training. Then we define a window of 5 epochs. The window starts from the first epoch and moves forward one epoch at a time until the end of the training. For example, the first window would consist of the heatmap outputs of epochs 1 to 5, the second window would consist of the heatmap outputs of epochs 2 to 6, and so on, until the last window consisting of the heatmap outputs of the last five epochs. The model gets the inputs, calculates the variance of the heatmap for each window for every class in our dataset, and then calculates the variance between sliding windows for each class (variance of variance). Finally, it compares the heatmap of variances for each class separately to determine whether they are stable. Algorithm 1 (or pseudo-code) illustrates the algorithm of XAI stability during training.

3.5 XAI Stability Evaluation

The formula for calculating the window variance is given:

\[
\text{window}_\text{var}_{j,i} = \frac{1}{N} \sum_{j=0}^{9} \sum_{i=\text{window}_\text{start}}^{\text{window}_\text{end}} (x_{j,i} - \text{window}_\text{mean})^2
\]  

\[
\text{window}_\text{var}_{j} = \frac{1}{M} \sum_{j=0}^{9} (\text{window}_\text{var}_{j,i} - \text{window}_\text{mean})^2
\]

where \(N = (\text{window}_\text{end} - \text{window}_\text{start} + 1) \cdot h \cdot w\) is the total number of elements in one window, and we repeat the equation for \(M = (\text{epochs} - \text{window}_\text{size} + 1)\) windows, and \(x_{j,i}\) is the \(j\)th class of the \(i\)th window.
**Algorithm 1 : XAI stability during training**

**Input** : Number of epochs, Last convolutional layer, initial model M

**Output** : Trained model M, mean_array

Initialize variable;

for epoch ← 1 to epochs do

  for each batch in train dataset do
    Train the model; Update the trainable weights using the gradients; Update the training accuracy metric;
  end

  Calculate the training accuracy and loss for the current epoch; Append the current training accuracy and loss to the respective lists; Reset the training accuracy metric;

  for each batch in validation dataset do
    Calculate the logits of the model; Calculate the validation loss value; Update the validation accuracy metric;
  end

  Calculate the validation accuracy and loss for the current epoch; Append the current validation accuracy and loss to the respective lists; Reset the validation accuracy metric;

  for each image in test dataset do
    Calculate the heatmap for each class; plot the superimpose and heatmap; Store the heatmap in the mean array for the current image and epoch;
  end

end

**Input** : epochs, window_size, mean_array

**Output** : class_variances

variances ← zeros((10, epochs-window_size+1, 48, 48));

for j ← 0 to 9 do

  for i ← 0 to epochs-window_size+1 do
    window_start ← i;
    window_end ← i + window_size;
    window_var ← var(mean_array[j, window_start:window_end], axis=None);
    variances[j,i] ← window_var;
  end

end

class_variances ← var(variances, axis=1);
3.6 **Model Optimization using XAI**

Using the Feature augmentation technique guided by the model [25], we utilize the model property by applying fixed and dynamic masks separately according to the features that Grad-CAM highlights to minimize the background impact and investigate the effect of the mask on model performance. This hybrid method is one of our innovations because the features are extracted from XAI feedback with minimal human intervention. Then the model is enhanced during the training process based on that feedback.

Our main objective is to understand the impact of the mask on model performance metrics, such as accuracy and loss. As a result of Section A.1, according to the heatmap, we investigate that the fluctuation in the variance is not related to the pattern of the digit but is related to the background, so if we impact the background less, then we would expect the model accuracy to improve and the loss to decrease.

### 3.6.1 Fixed Masking

Based on the feedback that is obtained from research question 1, the fixed mask is generated from XAI perspective and with minimal human intervention. We have an XAI heatmap (variance of variance according to Section 3.4) for a specific class. We want to focus more on pixels with low variance (i.e., stable) than pixels with high variance. Our initial XAI results showed that pixels with low variance belong to digits while background parts show high variance.

The binary mask is then defined on a pixel basis, setting a threshold to differentiate the foreground (the digit) from the background based on the variance values in the color bar from the previous experiment. This threshold is computed based on a histogram plotted for each class, representing the number of pixels per Grad-CAM intensity. The threshold is set at the point where an increase is noticed, demonstrating the pixels related to the background, as per the insights provided by the XAI feedback. This process is computed for all classes.

To obtain the final masked heatmap, we implement a pixel-level voting process among all classes. The resulting heatmap serves as an input pattern for our mask, which is compared with images during the training process. To generate a single final mask for all classes, any pixel with more than five (or equal to it) votes as the foreground is considered as foreground; otherwise, it is classified as background. In the following step, this final mask is applied batch-wise to the output of the first convolutional layer to minimize the influence of the background on the learning process. It is important to note that if the voted mask is all zeros, it means that none of the classes’ bi-
nary masks received enough votes to surpass the voting threshold for a given pixel. This could indicate that there is a lack of consensus among the classes for that particular pixel. Therefore, we need to change the values to “one” by flipping them. We consider this mask as a fixed mask that is created only once. Figure 4 summarizes the fixed mask idea. The mask is used to update the architecture of another version of the model, which is then retrained from scratch using the data.

The next step is to repeat this process with OCTMNIST, changing the number of classes and tasks related to it.

![Fixed Mask Idea](image)

Figure 4: Fixed mask idea (The idea behind the fixed mask is as follows: first, we rely on the XAI outcomes obtained from the previously trained model. From the XAI per class for that model, we generate a final voted mask. This mask is then utilized during the training of another version of the model, where it is taken into account.)

### 3.6.2 Dynamic Masking

As our plan is to introspect the training loop with XAI, dynamic masking is defined during the training process. The adaptive architecture modification based on the calculated masks is an interesting technique. There is an attempt to dynamically adjust the network’s architecture during training based on the importance of different regions of the input images. By applying masks to the convolutional layers, the network focuses on specific parts of the image that are considered relevant for better performance.

Similarly, the dynamic mask is generated, and the estimation of the voted mask is done in the same manner as the fixed mask. But the only difference is the creation of a sliding window that starts from the first epoch, covers a range of five epochs, and moves forward one epoch at a time until the final epoch. The mask is created for each window using the previous approach and it is applied to the first convolutional layer’s output. Thus, in this approach, multiple masks are generated throughout training, each affecting the learning process to reduce the effect of the less important features. A dynamic voting
threshold is employed based on the output of the background’s variance heatmap of inputs. The rationale behind this approach is that when the variance of the background is low, we aim to preserve this characteristic throughout the training process and across all layers of those inputs.

Algorithm 2 (or pseudo-code) illustrates how to calculate the dynamic mask function. As mentioned before, in the case of the fixed mask, the calculation is performed once using the previous version. However, in the current approach, the calculation is performed online and dynamically during the training process using the calculations from the previous epoch of the model.

Algorithm 3 (or pseudo-code) shows how to use dynamic masking based on XAI feedback with minimal human intervention during training for model optimization.
Algorithm 2: Calculate Masks

```python
def calculate_masks(mean_array, start, epoch, voting_threshold=5):
    variances ← zeros((10, 48, 48));
    // Iterate through each class
    for j ← 0 to 9 do
        window_start ← start;
        window_end ← epoch;
        window_var ← var(mean_array[j], window_start:window_end, axis=0);
        variances[j] ← window_var;
    end
    Initialize variable;
    // Iterate through each class
    for j ← 0 to 9 do
        hist, bins ← histogram(variances[j], bins='auto');
        increasing_bin_index ← argmax(diff(hist) > 0) + 1;
        threshold ← bins[increasing_bin_index];
        binary_mask ← variances[j] < threshold;
        // Show the binary mask
        ShowBinaryMask(j, binary_mask);
        binary_masks.append(binary_mask);
        if sum(binary_mask == 0) > sum(binary_mask > 0) then
            black_count ← black_count + 1;
        else
            white_count ← white_count + 1;
        end
    end
    // Perform voting among classes
    if black_count == 0 or white_count == 0 then
        voting_threshold ← 1;
    end
    else if black_count < 5 then
        voting_threshold ← white_count;
    end
    voted_mask ← zeros((48, 48), dtype=int);
    for row ← 0 to 47 do
        for col ← 0 to 47 do
            votes ← [binary_masks[j][row, col] for j ← 0 to 9];
            voted_mask[row, col] ← (sum(votes) >= voting_threshold);
        end
    end
    if sum(voted_mask) == 0 then
        voted_mask ← 1;
    end
    return voted_mask, black_count, white_count;
```
Algorithm 3: Using XAI to apply dynamic masking during training

**Input**: Number of epochs, Last convolutional layer, initial model M

**Output**: Trained model M, mean_array

Initialize variable;

**for** epoch ← 1 to epochs **do**

- Check if there are enough elements in voted_mask;
- **for each batch in train model** **do**
  - Train the model;
  - Applies the mask to the output of the first CNN layer;
  - Update the trainable weights using the gradients;
  - Update the training accuracy metric;
- end

- Calculate the training accuracy and loss for the current epoch;
- Append the current training accuracy and loss to the respective lists;
- Compute class-specific metrics for training;
- Reset the training accuracy metric;
- **for each batch in validation dataset** **do**
  - Calculate the logits of the model; Calculate the validation loss value; Update the validation accuracy metric;
- end

- Calculate the validation accuracy and loss for the current epoch; Append the current validation accuracy and loss to the respective lists; Reset the validation accuracy metric;

- **for each image in test dataset** **do**
  - Calculate the heatmap for each class; plot the superimpose and heatmap; Store the heatmap in the mean array for the current image and epoch;
- end

// binary mask

**Input**: epoch, window_size

**Output**: mask, black_count, white_count

**if** epoch >= window_size-1 **then**

- mask, black_count, white_count ← calculate_masks (mean_array, start, epoch, voting_threshold=5);
- start ← start + 1;

end
EXPERIMENTS AND RESULTS

In this section, different experiments are done to answer our research questions. The research questions are as follows:

1. Considering the models produced as outputs from every step in training, do these models have consistent explanations (XAI stability)?

2. How could explanations be utilized in the learning process to enhance model performance and internal states, such as final accuracy and loss?

4.1 EXPERIMENTS

In the first step, we address research question 1 related to XAI stability.

For the second research question, we analyze the explanations provided by the model during the training process and seek feedback on how to leverage the model’s behavior and performance effectively.

Our study begins by using the MNIST dataset and then applies the same experiments to the OCTMNIST dataset. We implement the experiments in Python using libraries such as TensorFlow and Keras. Initially, the batch size was set to 64, but we increased the batch size to 128 due to significant fluctuations [1] in the OCTMNIST dataset. To ensure a fair comparison, we set the batch size to 128 for all models.

4.1.1 Evaluating XAI Stability During Training by Using Statistical Metric

To train our three CNN models and calculate the variances to compare the values of heatmaps, we make a window of 5 epochs, let it continue forward, and train the model for 100 epochs. This can give an idea of how stable the Grad-CAM output is for a particular image across different training epochs. A high variance in heatmap values indicates that the heatmap changes significantly between epochs, while a low variance suggests that the heatmap is relatively stable over time. To evaluate the changes in the variance, we display a colorbar alongside the heatmap, which presents a reference for interpreting the color values in the heatmap, as well as an inspection of how each class has changed over time. In all three models, we utilize heatmaps to showcase XAI output. These heatmaps are visualized using the "hot" colormap, with black indicating the lowest value.
and white representing the highest. Consequently, the pixel intensity within the heatmap serves as an indicator of the significance of that specific pixel in influencing the model’s output.

When the plot mainly exhibits low-variance values (characterized by darker colors in the heatmap) across various spatial locations, it suggests that the generated explanations remain consistent across epochs for that particular class. By closely monitoring the outcomes, we can infer that after 100 epochs, a certain level of stability in class variance is present.

Moreover, we can delve into more intricate details to describe the stability of the XAI and the correlation between accuracy and variance. To achieve this, we separately illustrate the class accuracy with the highest and lowest values. Subsequently, based on these accuracies, we present the variance for these two distinct classes.

This approach provides us with a comprehensive understanding of the stability of XAI across epochs, as well as a clear depiction of how accuracy and variance are interrelated.

4.1.2 Evaluating Model Optimization During Training by Using XAI Feedback to Make Fixed Mask

We recognize that the output of XAI’s reveals a variation in the background or less important features of the inputs. To mitigate the influence of the background, we employ a fixed masking technique on the three models. The binary mask is created based on the variance values calculated for each class. The threshold for creating the binary mask is determined by finding the bin index where the histogram value starts to increase. The steps involved in defining the masking are listed below:

1. Calculate the variance between sliding windows for each class
2. Create a histogram of the variance values for the current class
3. Find the bin index where the histogram value starts to increase
4. Define the threshold as the right edge of the increasing bin
5. Create a binary mask based on the variance value using the threshold for the current class
6. Perform voting among classes and set the voted mask based on the voting threshold

4.1.3 Evaluating Model Optimization During Training by Using XAI Feedback to Make Dynamic Mask

For this section, we employ the same sliding window approach that covers 5 epochs. Within each window, a function creates a mask that
is applied to the output of the first convolutional layer during the training of each batch. This procedure zeroes out the less significant features, and the modified output is then forwarded to the subsequent layer. Additionally, we can monitor how heatmaps and masks change over time and evaluate their influence on the models.

4.2 RESULTS

The results of our experiments are summarized in this section. Additional results can be found in Appendix A. For more detailed information, refer to Section 4.1.1 for the experiment without masking, Section 4.1.2, and Section 4.1.3 for the experiment with fixed mask, and Section A.3 for the experiment with dynamic mask. Each section in the Appendix includes information on the CNN, ResNet-18, and Inception V1 models, respectively.

4.2.1 Result of Evaluating XAI Stability During Training by Using Statistical Metric

The MNIST results are presented in this section to demonstrate the stability of the CNN model during training. Additionally, we demonstrate the stability of OCTMNIST across all three architectures in Appendix A.

In this section, as previously mentioned, the colormap used to visualize the heatmap is "hot", which ranges from black (lowest value) to white (highest value). Therefore, the intensity of the pixel in the heatmap indicates the importance of that pixel in determining the output of the model. If the plot shows mostly low variance values (darker colors for a heatmap) throughout the spatial locations, it suggests that the generated explanations are stable across epochs for that specific class. By tracking the results, we can conclude that after 100 epochs, there is some degree of stability in our class variance.

In Figure 5, the variance of the classes with the highest and lowest accuracy is shown and compared separately. As previously discussed, the regions with a lighter or more intense color are the ones that the model pays more attention to when making the decision for the particular class, and these regions have higher variances. Conversely, the regions with a darker or less intense color are the ones that the model pays less attention to and have lower variances.

From Figure 6 to Figure 15, we can see more details to describe the XAI stability and the relation between accuracy and variance. For this reason, each class’s average variance, total accuracy, and accuracy per class are illustrated. It shows the model starts to learn very quickly and then is fixed somehow, so the model is converging. But if we look at the GradCam results, we can see the variances of some classes such as class 3 are unstable and have some fluctuations. Then if we
compare the patterns and the Grad-CAM, we observe that the digits patterns are fixed; therefore, the impact of the backgrounds is the result of fluctuations. So the background is sometimes important for the model and sometimes not. Therefore, from XAI sight, the explanations for digits are stable, but the background is unstable. This point guides us in answering the second research question. As follows, we categorize the digits figure as stable and unstable XAI.

The digits with stable XAI (as explained in Section 3.4 and above) are shown in Figure 6 to Figure 10. As we see, after some fluctuations related to the initial epochs that show the learning behavior of the model, the explanations should stabilize with fewer fluctuations. This is in line with what we expect while training the model.

Figure 5: MNIST-CNN: Variance of highest and lowest accuracy classes

![Figure 5](image)

Figure 6: MNIST-CNN: Average of variance for class 0, Global and local accuracy

![Figure 6](image)
Figure 7: MNIST-CNN: Average of variance for class 1, Global and local accuracy

Figure 8: MNIST-CNN: Average of variance for class 4, Global and local accuracy
Figure 9: MNIST-CNN: Average of variance for class 7, Global and local accuracy

Figure 10: MNIST-CNN: Average of variance for class 9, Global and local accuracy
Furthermore, Figure 11 to Figure 15 are digits where the variance changes a lot over training epochs and then we proceed to fix such cases to make XAI more stable by further feature selection/engineering.

Figure 11: MNIST-CNN: Average of variance for class 2, Global and local accuracy
Figure 12: MNIST-CNN: Average of variance for class 3, Global and local accuracy

Figure 13: MNIST-CNN: Average of variance for class 5, Global and local accuracy
Figure 14: MNIST-CNN: Average of variance for class 6, Global and local accuracy

Figure 15: MNIST-CNN: Average of variance for class 8, Global and local accuracy
4.2.2 Result of Evaluating Model Optimization During Training by Using XAI Feedback to Make Fixed/Dynamic Mask

MNIST dataset results for the CNN model are presented here. In Appendix A, it is shown for OCTMNIST and other models.

In Figure 16 and Figure 17 display the voted mask, histogram, and mask for the highest and lowest accuracy classes in the MNIST dataset. Based on the histogram, we determine the threshold to define a binary mask for each class. This binary mask helps us identify the areas of interest within the image. By calculating pixel-wise votes among the classes, we obtain a fixed mask that mitigates the effect of the background. When we apply this mask to the output of the first convolutional layer, it propagates through the subsequent layers. Dynamic masks are also created the same way as fixed masks; however, dynamic masks are generated iteratively during training, whereas fixed masks are generated only once.

Figure 16: MNIST-CNN: Voted mask

(a) Binary of class 0
(b) Histogram of class 0

(c) Binary of class 9
(d) Histogram of class 9

Figure 17: MNIST-CNN: Binary mask and histogram of class 0 as highest and 9 as lowest class accuracy (The histogram illustrates the distribution of variance values within the heatmap, along with the corresponding pixel count for each value. As the values begin to increase, the threshold for generating the binary mask is determined.)
4.2.3 Result of Comparison of Model Optimization and Stability During Training

We illustrate and compare the training and validation accuracies, as well as the stability fluctuations, for the classes with the highest training accuracy across all three models for each strategy (without masking, fixed masking, and dynamic masking) using the MNIST dataset in Figure 18. It is important to clarify that "T.ACC" represents training accuracy, and "Val.ACC" stands for validation accuracy.

Meanwhile, Figure 19 displays all parameters for the classes with the lowest training accuracy.

We can observe that in the classes with the highest training accuracy, the models are more stable than in those with the lowest training accuracy. However, using both masking methods guides the models to decrease instability. Additionally, both the highest and lowest accuracy plots show that the Inception model has lower validation accuracy than the other two methods. Moreover, the ResNet-18 model has the highest accuracy among the three models for almost all the lowest and highest accuracies.

Figure 18: MNIST: Comparison of the highest training and validation accuracies and their variances across three models for each strategy. For ResNet-18, the highest accuracy class is 0, while for Inception V1, the highest accuracy class is 6 for the without masking method and dynamic masking, and 4 for the fixed masking method. In the CNN model, the highest accuracy class is 0 when using masking methods and 1 without the masking method.
Figure 19: MNIST: Comparison of the lowest training and validation accuracies and their variances across three models for each strategy. For ResNet-18, the lowest accuracy class is 4, while for Inception V1, the lowest accuracy class is 9 for the without masking method and dynamic masking, and 8 for the fixed masking method. In the CNN model, the lowest accuracy class is 9 when using masking methods and 3 without the masking method.
Same as the previous part, we illustrate all the metrics for the highest training accuracy classes for each strategy and the OCTMNIST dataset in Figure 20 and the lowest training accuracy classes in Figure 21.

While this dataset exhibits more fluctuations compared to the MNIST dataset, it’s evident that for the class with the highest training accuracy, the fixed masking method significantly reduces instability across CNN and Inception models. The Dynamic masking strategy demonstrated the most effective reduction in instability for the CNN model. In classes with the lowest training accuracy, the fixed mask outperforms the CNN model in reducing instability, while the dynamic masking method provides better stability for ResNet-18. In conclusion, the inherent fluctuations characteristic of the OCTMNIST dataset appear to challenge the masking methods, especially in reducing instability within the classes with the lowest training accuracy. In both plots, the CNN model has lower training and validation accuracy. In Figure 20, Inception V1 and ResNet-18 have almost the same accuracy.

Figure 20: OCTMNIST: Comparison of the highest training and validation accuracies and their variances across three models for each strategy. Class 3 consistently has the highest accuracy across all 3 models and methods.
Figure 21: OCTMNIST: Comparison of the lowest training and validation accuracies and their variances across three models for each strategy. Class 2 consistently has the lowest accuracy across all 3 models and methods.
4.2.4 Performance Evaluation of Models Using Masking Methods

In Figure 22, the result of the method without XAI and with fixed and dynamic masks is summarized.

Upon initial observation, it’s evident that the fixed method outperforms other methods, yielding higher results except for ResNet with the MNIST dataset. In this particular model, the dynamic masking performs better. However, a more detailed examination reveals that it’s evident that while the fixed mask outperforms in most models, in the three models that their train accuracies are underlined in the table, even though the fixed mask retains the highest performance, the dynamic masking still surpasses the results of the model without any mask. This underscores the effectiveness of the dynamic masking method in some models.

![Figure 22: Comparison of training/validation accuracy among different methods (without mask, with fixed and dynamic mask) and datasets](image)

The superior performance of the fixed mask method can be attributed to its use of XAI feedback with minimal human intervention after the model is fully trained which leads to have more accurate mask, whereas the dynamic mask method integrates the mask during the ongoing training process. Nevertheless, it’s important to note that the datasets utilized in this study are characterized by relatively simple patterns. Conversely, the deep models have more complex designs, which makes them better at accurately identifying features. As a result, these models converge rapidly, diminishing the noticeable impact of the masking methods on their performance. Moreover, according to the table values, deeper models have higher training accuracy but lower validation accuracy. Increasing network capacity can improve training accuracy, but using too many parameters can lead to overfitting. Furthermore, when we examine the time it takes for each model to train for 100 epochs. The execution time for training with these masking methods requires only a few extra minutes, in-
indicating that the computations associated with the masking methods don’t consume much additional time.

Moreover, the time required to train the MNIST dataset using a CNN model is 13 minutes, which is 43.33% of the time taken by the ResNet model (30 minutes), and just 24.53% of the time taken by the Inception model (53 minutes). Meanwhile, the time required to train the OCTMNIST dataset with the CNN model is 17 minutes, which is 35.42% of the time taken by the ResNet model (48 minutes), and approximately 17.89% of the time taken by the Inception model (95 minutes). Achieving the same results with less resource consumption and shorter training times in the CNN model by applying the mask can be an advantage.

As observed in Figure 23, masking methods yield higher test accuracy in four instances. For the MNIST dataset, the test accuracies align more closely with the validation accuracies, though they are also not far from the training accuracies. This highlights its nature as a relatively simple and well-studied dataset. In contrast, the OCTMNIST dataset exhibits a lower test accuracy, suggesting that it is more challenging due to its complex patterns compared to the MNIST dataset.

![Figure 23: Comparison of test accuracy among different methods (without mask, with fixed and dynamic mask) and datasets](image-url)
In this section, we attempt to answer some challenges and questions that we face in our study:

1. Which layer to use for Grad-CAM?

   In Grad-CAM, the earliest layer extracts simple features like edges and single pixel information [21], allowing for understandable XAI pictures. As the size of the convolution layer decreases, the knowledge at the last layers becomes more consolidated [21], resulting in a representation space that provides knowledge of the classification and representation of each class. Therefore, the first layers perform feature engineering closer to the input, while the last layers perform feature engineering closer to the learning behavior and representation space. By comparing the first and last layers, we can determine which layers provide more accurate XAI information in the training loop.

   To determine this, we have to consider different factors. When using the first layers, we are primarily focusing on simple feature extractors. On the other hand, the last convolutional layers provide a better balance between high-level semantics and detailed spatial information [21, 34]. However, in our case, the model uses heatmaps not only to highlight the most important parts of the dataset but also to create masks and vote among pixels. This requires us to have a broader view, such as capturing the entire digit in MNIST. Therefore, it makes sense to start testing with the last layers and then gradually move up to the shallowest layers. It is important to note that from a visualization perspective, which aims to display input data that closely resembles the original input, the first layer is the most appropriate choice. However, when considering the behavior of the model, selecting the last layer is more suitable. When there is a human in the loop, the first convolutional layer is helpful for better understanding the features and masking method. However, without human involvement, it is wiser to choose the last convolutional layer, as the features at this stage are more closely related to the learned representation. Based on our experiments, we select the last convolutional layer named “layer4.1.conv2” for both datasets in ResNet-18, the layer close to the last convolutional layer that is “conv2d_44” for both datasets Inception V1 and the last layer in the CNN model for both datasets, we select the last convolutional layer that is "conv2d_2".
2. Do ResNet-18 and Inception improve learning stability, or are other parameters such as learning rate more important?

Deeper models such as ResNet-18 and Inception V1 typically exhibit more stable learning due to their numerous layers and enhanced ability to extract specific features. This enables the models to focus more consistently and accurately on the crucial aspects of the data, thereby providing an advantage in performance. However, this improved stability and consistency come at a cost, as they often require greater consumption of time and resources. In our study, we investigate the performance and stability of these models in greater detail, examining the trade-offs involved and seeking insights into how they can be optimally leveraged.

We evaluated the learning rates (-1, -2, -3) in the MNIST Inception V1 model. When a higher learning rate is used, the accuracy begins at a higher value and the loss starts at a lower value. However, the values tend to not change anymore during 50 epochs. When we use a lower learning rate, the accuracy may start slower, but the loss will also start at a higher value. As we continue with the learning process, the accuracy values will gradually increase while the loss values decrease, until they eventually reach a similar value to what we had before. It’s all about finding the right balance between accuracy and loss and adjusting our learning rate accordingly. After analyzing the accuracy and loss values of our different methods dynamic, fixed, and without masking, we found that they are very similar. One possible explanation for this could be that the background is dark.

In our experiments with ResNet-18, a model renowned for its complexity and numerous layers designed to extract complex features, we observed stability in both the model and its explanations. This stability distinguishes it from shallower models, highlighting its ability to provide more consistent insights.

However, when we experimented with different learning rates, specifically -4, -3, -2, and -1, We found that the learning rate significantly impacted accuracy and loss. Specifically, a larger learning rate correlated with a higher starting point for training accuracy but, conversely, a lower starting point for training loss. Interestingly, this relationship was not reflected in the stability of explanations, as there was no notable effect on them based on the learning rate. When training ResNet-18 with OCTMNIST, we further investigated the relationship between learning rate and performance. We found that by increasing the learning rate, the fluctuations in validation loss were minimized, becoming nearly negligible. However, the fluctuations in validation accu-
racy remained unaffected by changes in the learning rate. This observation underlines the complicated relationship between learning rate and various performance metrics, emphasizing the need for careful tuning and consideration when working with complex models like ResNet-18.

3. Is it always disadvantageous for a model to have inconsistent explanations?

While inconsistent explanations might initially appear problematic, they can actually indicate a normal part of the model’s learning process. In the early stages of training, say the first few epochs, the model is actively learning and adjusting its understanding of the input data. This ongoing adaptation and learning can cause the model’s internal representations and its explanations to fluctuate, leading to what we perceive as inconsistency in the explanations. This, however, is not necessarily a cause for concern as it’s an essential part of the model’s learning process.

As the model continues to be trained over more epochs, its understanding of the data should stabilize, and this is when we should expect the explanations to be consistent. A successfully trained model won’t focus arbitrarily on different parts of the input, but rather, it will improve consistently on the most salient or relevant features that are crucial for decision-making.

So, after an initial period of training where inconsistent explanations are expected and can even be considered a positive sign of learning, we should look for a trend toward more consistent explanations. This consistency demonstrates that the model is now focusing accurately on key information rather than scattering its attention across the input data, thus leading to more reliable predictions and better overall performance.

4. How does applying a mask to the output of a convolutional layer impact the network’s learning process, and what are the potential consequences of this approach?

A mask is applied to the output of a convolutional layer to selectively block certain values, setting them to zero. This effectively prevents specific portions of the layer’s output from contributing to the subsequent layers in the network. While this masking is done after the convolutional layer produces its output, it can still influence the learning process during backpropagation.

When backpropagation occurs, the gradient of the loss with respect to the masked values will be zero because the masked values have no effect on the loss. This zero gradient means that the weights associated with the masked values will not be updated during backpropagation. The gradient provides informa-
tion about how the weight affects the loss, and a zero gradient indicates that adjusting the weight will not change the loss. As a result, the optimizer will not update the weight.

By applying the mask, certain parts of the convolutional layer’s output are blocked, preventing them from contributing to the rest of the network. This constraint can hinder the network’s ability to learn from the input data because it cannot use all the information available in the convolutional layer’s output. Consequently, the network’s performance, in terms of accuracy and loss, may not show a significant difference compared to training without the mask.

The use of a mask can constrain the network’s learning process, but it can also be beneficial in situations where we want the network to focus on specific, important features of the input data.

5. What are the best ways to evaluate our methods with other approaches considering two research questions from both an XAI and a model perspective?

Traditionally, test performance has been a key metric for evaluating machine learning models, often measured using metrics like accuracy, precision, recall, or F1-score. Nonetheless, there are several reasons why relying solely on test performance may not be adequate, especially when considering the more intricate aspects of a model’s behavior. In many real-world applications, models are expected to demonstrate complex semantic properties, including fairness, interpretability, and reasoning abilities, like the requirements of our current task. Often, test performance cannot directly determine these attributes [25].

On the other hand, we use test data for XAI feedback during training, while others use training data. For example, using test accuracy in this situation makes sense if we’re comparing the effectiveness of our method, which utilizes XAI feedback during training with test data, against another approach that uses training data without XAI feedback. It’s important, however, to consider some factors to ensure a fair and meaningful comparison with other methods:

- **Evaluation Metrics [29]:** While test accuracy is a common evaluation metric, it may not be the only relevant metric in this context. We should also consider using other metrics, especially those related to interpretability and fairness, depending on the objectives of our XAI-based method.

- **Benchmarking [29]:** When comparing our method to another that uses training data, it’s crucial to ensure a fair comparison. Both methods should be evaluated under the
same conditions and on the same dataset splits (training, validation, and test) to establish a valid basis for comparison.

As a result, it can be difficult to find the same dataset and metrics to compare our approaches with other classification methods or to implement similar classification tasks. Also, the methodologies they employ, and the challenges they address, are different from the specific techniques and objectives of our study. A direct comparison with their work would not only require us to explore their objective, for example, domain generalization [34] but would also demand an adjustment of our masking methods to fit their framework, which was beyond our current research scope. Thus, due to these methodological differences and the unique directions of our respective research aims, a direct comparison was not undertaken.

Based on the discussion and some recent papers by Zunino et al. [34] and Weber et al. [25], we evaluate our work from an XAI perspective with variance to measure stability, and for the test, training, and validation accuracy of our model without and with mask.

6. How effective is variance as a metric in assessing the stability of model explanations?

To evaluate the consistency of the explanations, in our research, we selected variance as a metric to measure the stability of attention patterns produced by Grad-CAM. This decision stems from the inherent expectation of some degree of variability in the attention given by Grad-CAM across different training runs.

Establishing how much variability is acceptable and determining when this variability indicates instability are central concerns. This leads us to the exploration of the variance of variances as a method to measure stability. The underlying idea is that, while some variance is anticipated, the fluctuation in this variance across epochs or training sessions can provide deeper insights into stability.

However, this approach comes with its challenges. For instance, determining a universally accepted threshold for variance of variance is not straightforward. A certain degree of change in this metric is anticipated due to the dynamic nature of neural network training. However, setting an exact threshold beyond which the variance indicates instability becomes complex. Various factors can influence this threshold, including XAI algorithms (like Grad-CAM’s specific configuration) [13] and the dataset’s characteristics.
Furthermore, even though variance can highlight specific patterns or trends in the data, its suitability as the primary metric for stability needs careful consideration. There might be scenarios where variance fails to capture all the details of stability, potentially causing some stable patterns to appear unstable.

Given these factors, it’s clear that while variance (and its second-order metric, variance of variance) offers valuable insights, they should be interpreted with caution. Future research could look into alternative or combined metrics to achieve a more comprehensive understanding of stability. Such a multi-metric approach may establish clearer and more universally accepted standards for stability in model explanations.

7. How much variance is enough for an XAI to be stable?

The acceptable level of variance for an XAI method to be considered stable depends on several factors, including specific application, and the data’s nature. Additionally, variances in XAI depend on the application, such as healthcare, where high-level decisions made or influenced by such systems ultimately affect human health, necessitating trust in the explanation [4]. Consequently, high stability is considered desirable because it indicates that the explanation method is reliable [13]. Furthermore, noise or variable data may naturally lead to more varied explanations, as seen in OCTMNIST, another factor influencing acceptable variance.

As a solution, we can set thresholds based on expertise or integrate user feedback mechanisms as our work to iteratively improve and refine explanations, enhancing stability over time.
CONCLUSION

In recent years, the use of DNNs has significantly increased due to their high performance. However, one major concern is their lack of understandability, leading to reduced human trust in these models. To address this issue, XAI has been introduced as a method to make DNNs more transparent.

Despite the benefits of XAI, its current implementation does not fully capitalize on the potential to improve DNN performance through introspective mechanisms. In this study, we explore model explanation properties, particularly focusing on their stability during training. Additionally, we employ an explainability method to visualize the model’s behavior throughout the training process. This allows the XAI to generate feedback with minimal human intervention in the loop and affect certain model features, such as accuracy and loss. This approach offers an opportunity to optimize the model more efficiently. By integrating XAI into the training loop to detect early if the model is learning something meaningful, we can influence the training process.

Our models are trained using datasets including OCTMNIST and MNIST with high-accuracy CNN, ResNet-18, and Inception V1 models. Grad-Cam is our selected XAI algorithm to use in this work.

To estimate the stability of explanations during the training process, we utilize the variance of the heatmaps generated by Grad-CAM for each class over a sliding window of epochs. The intuition behind this is that a model is considered XAI stable if its explanations for the same input do not vary significantly over time. Therefore, if the variance of the heatmaps for a particular class remains low over a sliding window of epochs, it suggests that the model’s explanations for that class are stable and consistent over time. However, it is important to note that variance in the explanations might be needed as long as the machine learning model improves accuracy (i.e., learns over time). So, not all instability of XAI might be related to being an inconsistent explainability.

Tracking the variance of our heatmaps over time for each class shows that the model explanation of the input is stable while the regions related to less important features (like background) are not. We notice that these less important features of the input space image are sometimes important for the model to classify and distinguish between classes. Receiving this feedback from XAI, we propose a novel method to define a fixed and dynamic mask to minimize the effect of these least important features, on the learning process introspec-
tively. The fixed mask is generated only once, and the dynamic mask is generated iteratively during training. We augment the feature by applying these masks to the first convolutional layer’s output.

Our observations suggest that both masking methods can diminish the influence of the non-critical features during training, enhancing both loss and accuracy metrics. Although the fixed mask typically performs better than both the dynamic masking and models without a mask, there are some cases where the dynamic masking outperforms models that don’t use any mask at all. Thus, both masking methods yield improved results compared to models that don’t use masking. However, these methods do not remarkably affect deeper models due to their complicated structure and high performance. Notably, considering how fast the models converge, especially in deeper models, we do not expect the model accuracy to be very high.

However, these two methods, dynamic and fixed masks, are two different methods that have their advantages and disadvantages. The fixed mask, generated after the model is fully trained, results in higher accuracy and loss performance. However, it is more time and memory-consuming. Conversely, the dynamic mask, although less demanding in time, requires more computation and yields fewer improvements than the fixed mask. In the dynamic mask method, the model continues to train even when the mask is applied without any dependency on the training of another model. In contrast, the fixed mask method requires the model to be fully trained before the XAI results can be obtained, making it dependent on the previous training iteration.

Finally, in the experiments conducted for this thesis, different datasets and models were utilized to assess the efficacy of the proposed masking methods. However, it is important to note that the results obtained are specific to the experimental conditions and configurations used in this study. The performance of the masking methods is influenced by a variety of factors including, but not limited to, the parameters of the model, the architecture of the model, and the characteristics of the dataset. Therefore, while the results indicate the effectiveness of the masking methods under the conditions tested, it is not possible to guarantee the same performance across different training scenarios, models, or datasets.

6.1 Answers to the Research Questions

In this part, we provide responses to our research questions proposed at the outset of this project:

1. Considering the models produced as outputs from every step in training, do these models have consistent explanations (XAI stability)?
   
   As previously mentioned, our primary objective centers around leveraging the explanations extracted during training to guide
the model introspectively. To effectively achieve this, it’s crucial to determine whether these explanations remain stable throughout and to assess their trustworthiness. This reasoning forms the basis of this research question.

Through our experiments, we observe that the heatmaps generated during training and the average variances for each class depict a consistent focus on the important features. However, the stability of explanations relating to less important features fluctuates. The model would sometimes pay attention to these parts and, other times, ignore them. In conclusion, the less significant features tend to be inconsistent in shallow models, while the important features remain stable. On the MNIST dataset, deep models show consistent results in both important and less significant features. However, the OCTMNIST dataset performs better in shallow models and exhibits more inconsistency than MNIST, likely due to unique data characteristics.

2. How could explanations be utilized in the learning process to enhance model performance and internal states, such as final accuracy and loss?

Explanations provided by XAI can be key to enhancing the internal states of a model, such as accuracy and loss. Specifically, in our experiments with the MNIST dataset, we utilize XAI to monitor the behavior of the CNN model throughout the training process. It was observed that the model occasionally focused on less significant features, such as the background.

It’s important to note that, while training, it’s evident that the model’s explanations exhibit certain levels of instability. This fluctuation in explanations might raise concerns about the model’s interpretability over the course of its learning. However, in spite of this observed instability, the model consistently achieves high accuracy and demonstrates its competence in classifying various classes correctly. This consistent performance suggests that, despite the temporary nature of the explanations, the underlying decisions of the model are reliable. Hence, even with the observed variability in explanations, we can have confidence in these explanations. This trustworthiness of the model’s explanations, combined with its high accuracy, makes it a valuable tool for generating feedback, which can further be utilized to refine and improve the model or related processes.

This insight prompted the use of XAI to generate masks that minimize the impact of less important features on the classification of our dataset, thereby improving the model’s accuracy and loss. Two different approaches were employed for masking: a fixed masking method and a dynamic masking method. It should be noted that the masks were not manually created but
were generated by the XAI, thus allowing for minimal human intervention in the loop. Additionally, we evaluated our masking methods on deeper models such as ResNet-18 and Inception V1, as well as on another dataset, OCTMNIST. As previously mentioned, both masking methods, especially the fixed masking method, had an impact on the performance of the models.

In conclusion, explanations can help identify what aspects of the data the model is focusing on, guiding for targeted modifications to the model or the data to steer the optimization process. Moreover, they can also provide a deeper understanding of the model’s performance, uncovering why it might achieve a certain accuracy or loss. This insight can guide adjustments to the model to improve performance introspectively.

6.2 Future work

Employing XAI during training represents a new area with numerous unresolved questions. While we attempted to answer our research questions, several open questions emerged.

Future studies could expand upon our work by integrating other XAI algorithms, such as IG, and comparing their efficacy with Grad-CAM. Additionally, there is scope to evaluate our algorithms against a broader set of parameters like model architecture, optimization algorithm, regularization, batch size, and learning rate. It would also be insightful to assess the performance of these masking methods using more complicated datasets. An additional approach to exploration might involve developing strategies to prevent overfitting when incorporating XAI feedback into the training process. Furthermore, investigating the possibilities of using more than one mask is recommended, as extracting a single mask that all classes agree on could be challenging for classes with drastically different patterns. Also, a comparison of our method with other feature engineering techniques is suggested for future exploration.
APPENDIX

A.1 EVALUATION OF XAI STABILITY VIA STATISTICAL METRICS DURING TRAINING

A.1.1 CNN Model

Figure 24: OCTMNIST-CNN: Variance of highest and lowest accuracy classes

Figure 25: OCTMNIST-CNN: comparison of highest and lowest training accuracy classes
(a) Variance of class 2 over window

(b) Variance of class 3 over window

Figure 26: OCTMNIST-CNN: variance of variance over window epochs
A.1.2 ResNet-18 Model

Figure 27: MNIST-ResNet-18: Comparison of highest and lowest training accuracy classes
Figure 28: MNIST-ResNet-18: Variance of variance over window epochs

(a) Variance of class 0 over window

(b) Variance of class 4 over window

Figure 29: MNIST-ResNet-18: Variance of highest and lowest accuracy classes

(a) Variance of class 0

(b) Variance of class 4
A.1 Evaluation of XAI stability via statistical metrics during training

- (a) Variance of class 2
- (b) Variance of class 3

Figure 30: OCTMNIST-ResNet-18: Variance of highest and lowest accuracy classes

Figure 31: OCTMNIST-ResNet-18: Comparison of highest and lowest training accuracy classes
(a) Variance of class 2 over window

(b) Variance of class 3 over window

Figure 32: OCTMNIST-ResNet-18: Variance of variance over window epochs
A.1.3 *Inception V1 Model*

![Examples of variance for different classes](image)

(a) Variance of class 6   
(b) Variance of class 9

**Figure 33:** MNIST-InceptionV1: Variance of highest and lowest accuracy classes

![Comparison of highest and lowest accuracy classes](image)

**Figure 34:** MNIST-InceptionV1: Comparison of highest and lowest training accuracy classes
Figure 35: MNIST-InceptionV1: Variance of variance over window epochs
A.1 evaluation of xai stability via statistical metrics during training

Figure 36: OCTMNIST-InceptionV1: Variance of highest and lowest accuracy classes

Figure 37: OCTMNIST-InceptionV1: Comparison of highest and lowest training accuracy classes
Figure 38: OCTMNIST-InceptionV1: Variance of variance over window epochs

(a) Variance of class 2 over window

(b) Variance of class 3 over window
A.2 Evaluation of model optimization using XAI feedback for fixed mask

A.2.1 CNN Model

Figure 39: MNIST-CNN-Fixed mask: Average variance and the average of average variance for highest and lowest accuracy class.
Figure 40: OCTMNIST-CNN: Voted mask

(a) Binary of class 2
(b) Histogram of class 2

(c) Binary of class 3
(d) Histogram of class 3

Figure 41: OCTMNIST-CNN: Binary mask and histogram of class 3 as highest and 2 as lowest class accuracy (The histogram illustrates the distribution of variance values within the heatmap, along with the corresponding pixel count for each value. As the values begin to increase, the threshold for generating the binary mask is determined.)
A.2 Evaluation of Model Optimization Using XAI Feedback for Fixed Mask

Figure 42: OCTMNIST-CNN-Fixed mask: Average variance and the average of average variance for highest and lowest accuracy classes
A.2.2 ResNet-18 Model

Figure 43: MNIST-ResNet-18: Voted mask

Figure 44: MNIST-ResNet-18: Binary mask and histogram of class 0 as highest and 4 as lowest class accuracy (The histogram illustrates the distribution of variance values within the heatmap, along with the corresponding pixel count for each value. As the values begin to increase, the threshold for generating the binary mask is determined.)
Figure 45: MNIST-ResNet-18-Fixed mask: Average variance and the average of average variance for highest and lowest accuracy classes
Figure 46: OCTMNIST-ResNet-18: Voted mask

Figure 47: OCTMNIST-ResNet-18: Binary mask and histogram of class 3 as highest and class 2 as lowest class accuracy (The histogram illustrates the distribution of variance values within the heatmap, along with the corresponding pixel count for each value. As the values begin to increase, the threshold for generating the binary mask is determined.)
A.2 evaluation of model optimization using xai feedback for fixed mask

Figure 48: OCTMNIST-ResNet-18-Fixed mask: Average variance and the average of average variance for highest and lowest accuracy classes
A.2.3 Inception V1 Model

Figure 49: MNIST-InceptionV1: Voted mask

Figure 50: MNIST-InceptionV1: Binary mask and histogram of class 4 as highest and 8 as lowest accuracy class (The histogram illustrates the distribution of variance values within the heatmap, along with the corresponding pixel count for each value. As the values begin to increase, the threshold for generating the binary mask is determined.)
A.2 Evaluation of model optimization using xai feedback for fixed mask

Figure 51: MNIST-InceptionV1-Fixed mask: Average variance and the average of average variance for highest and lowest accuracy classes

Figure 52: OCTMNIST-InceptionV1: Voted mask
Figure 53: OCTMNIST-InceptionV1: Binary mask and histogram of class 3 as highest and 2 as lowest accuracy class (The histogram illustrates the distribution of variance values within the heatmap, along with the corresponding pixel count for each value. As the values begin to increase, the threshold for generating the binary mask is determined.)

Figure 54: OCTMNIST-InceptionV1-Fixed mask: Average variance and the average of average variance for highest and lowest accuracy classes
A.3 EVALUATION OF MODEL OPTIMIZATION USING XAI FEEDBACK FOR DYNAMIC MASK

A.3.1 CNN Model

Figure 55: MNIST-CNN: Comparing the accuracy of Training/validation fixed and dynamic mask

Figure 56: MNIST-CNN: Comparing the loss of Training/validation fixed and dynamic mask
Figure 57: MNIST-CNN: Number of digits with low/high background variance

(a) Variance of class 0  
(b) Variance of class 9

Figure 58: MNIST-CNN: Variance of highest and lowest accuracy classes
A.3 evaluation of model optimization using xai feedback for dynamic mask

Figure 59: MNIST-CNN-Dynamic mask: Average variance and the average of average variance for highest and lowest accuracy classes

(a) Class 0 with and without mask

(b) Class 9 with and without mask
Figure 60: OCTMNIST-CNN: Comparing the accuracy of Training/validation fixed and dynamic mask

Figure 61: OCTMNIST-CNN: Comparing the loss of Training/validation fixed and dynamic mask
A.3 evaluation of model optimization using xai feedback for dynamic mask

Figure 62: OCTMNIST-CNN: Number of digits with low/high background variance

![Graph showing number of inputs with low/high background variance](image)

Figure 63: OCTMNIST-CNN: Variance of highest and lowest accuracy classes

(a) Variance of class 2  
(b) Variance of class 3

![Heatmaps showing variance](image)
(a) Class 2 with and without mask

(b) Class 3 with and without mask

Figure 64: OCTMNIST-CNN-Dynamic mask: Average variance and the average of average variance for highest and lowest accuracy classes
A.3.2 ResNet-18 model

Figure 65: MNIST-RestNet-18: Comparing the accuracy of Training/validation fixed and dynamic mask

Figure 66: MNIST-RestNet-18: Comparing the loss of Training/validation fixed and dynamic mask
Figure 67: MNIST-RestNet-18: Number of digits with low/high background variance

(a) Variance of class 0  
(b) Variance of class 4

Figure 68: MNIST-RestNet-18: Variance of highest and lowest accuracy classes
Figure 69: MNIST-RestNet-18-Dynamic mask: Average variance and the average of average variance for highest and lowest accuracy classes.
Figure 70: OCTMNIST-ResNet-18: Comparing the accuracy of Training/validation fixed and dynamic mask

Figure 71: OCTMNIST-ResNet-18: Comparing the loss of Training/validation fixed and dynamic mask
A.3 evaluation of model optimization using xai feedback for dynamic mask

Figure 72: OCTMNIST-ResNet-18: Number of digits with low/high background variance

Figure 73: OCTMNIST-ResNet-18: Variance of highest and lowest accuracy classes

(a) Variance of class 2  
(b) Variance of class 3
Figure 74: OCTMNIST-ResNet-18-Dynamic mask: Average variance and the average of average variance for highest and lowest accuracy classes.
A.3.3 *Inception V1 model*

Figure 75: MNIST-InceptionV1: Comparing the accuracy of Training/validation fixed and dynamic mask

Figure 76: MNIST-InceptionV1: Comparing the loss of Training/validation fixed and dynamic mask
Figure 77: MNIST-InceptionV1: Number of digits with low/high background variance

Figure 78: MNIST-InceptionV1: Variance of the highest and lowest accuracy classes
A.3 Evaluation of Model Optimization Using XAI Feedback for Dynamic Mask

Figure 79: MNIST-InceptionV1-Dynamic mask: Average variance and the average of average variance for highest and lowest accuracy classes.
Figure 80: OCTMNIST-CNN: Comparing the accuracy of Training/validation fixed and dynamic mask

Figure 81: OCTMNIST-InceptionV1: Comparing the loss of Training/validation fixed and dynamic mask
A.3 evaluation of model optimization using xai feedback for dynamic mask

Figure 82: OCTMNIST-InceptionV1: Number of digits with low/high background variance

Figure 83: OCTMNIST-InceptionV1: Variance of the highest and lowest accuracy classes
Figure 84: OCTMNIST-InceptionV1-Dynamic mask: Average variance and the average of average variance for highest and lowest accuracy classes
BIBLIOGRAPHY


