Implementation, Reproduction, and Improvement of Disguising Attacks with Explanation-Aware Backdoors

1st Luis Quiñones
Computer Science
The University of Texas at El Paso
luisquinones141101@gmail.com

***Abstract*—This report is about implementing, reproducing, and improving the study "Disguising Attacks with Explanation-Aware Backdoors." the original experiments and work have been replicated, confirming the feasibility of embedding backdoors that not only alter model predictions upon trigger activation but also explanation methods like SHAP, LIME, and Grad-CAM to conceal malicious activities. After testing the different experiments and achieving promising results, I have implemented some improvements to make this research more complete.**

I. Introduction

A. Context and Problem Statement

Machine learning systems are becoming essential in our life-impacting fields like healthcare, finance, and cybersecurity, making them vulnerable targets for adversarial attacks. Among these, backdoor attacks represent a severe threat where a model is maliciously altered to perform as expected under normal circumstances but exhibits malicious behavior when specific inputs are introduced and can manipulate the result leading to false information and predictions.

B. Disguising Attacks with Explanation-Aware Backdoors

The novelty of the Disguising Attacks with Explanation-Aware Backdoors lies in the ability to manipulate model interpretability tools to hide the presence of backdoors. By targeting explanation mechanisms like SHAP, LIME, and Grad-CAM, these attacks exploit the reliance on interpretability in identifying malicious behavior, effectively disguising their presence.

C. Objectives and Contributions

This project is designed to:

- Step 1 (Implementation):
  Implement the methodology described in the paper to understand the original problem and the work done by the author.
- Step 2 (Reproduction):
  Reproduce the experiments and validate the results provided in the original investigation.
- Step 3 (Improvement):
  I will try different improvements that can be added to the paper in the short time given.

D. Organization of the Report

The report is organized as follows:

- Section 1 (Introduction):
  Introduction to the paper explaining the problem statement and context of the report.
- Section 2 (Background And Related Work):
  Discusses the evolution of research work in attacks on machine learning models.
- Section 3 (Methodology):
  Describes the methodology for implementation and reproduction.
- Section 4 (Analysis and Roadblocks):
  Analyze the Results and discuss what worked and what did not work.
- Section 5 (improvements/Innovations):
  Test different improvements that can be added to the paper.
- Section 6 (Conclusion and Future Work):
  Concludes the report with insights and future directions.

II. Background and Related Work

Attacks on machine learning models have been a common topic of research lately, Early studies showed that tiny changes to input data can trick very advanced systems into making completely wrong predictions. More recent research has focused more on how to make machine-learning models more understandable, They have developed tools like saliency maps and gradient-based methods to explain how models make decisions, but later was discovered that these explanation tools can also be tricked, even these explanation tools can act against the model predictions, an example is this paper of Disguising Attacks with Explanation-Aware Backdoors that show us how the explanation tool can be trick to create a wrong explanation.

III. methodology

This section focuses on giving a detailed approach used to implement, reproduce, and improve the findings from the paper Disguising Attacks with Explanation-Aware Backdoors. I will discuss the tools, environment, and techniques used to achieve successful results and bring a comparison to the ones presented in the original paper.

In detail, I will provide an extensive description of the experiment setup, model evaluation, and training. At the end, I will discuss the challenges faced when testing models and running attacks.

A. Implementation

1) Tools and Environment

This is the environment that has been set up and the tools that were used to implement the work presented in Disguising Attacks with Explanation-Aware Backdoors:

- Frameworks: To train the models and to evaluate them I have used the following frameworks TensorFlow and PyTorch.
- Interpretability Tools: SHAP, LIME, and Grad-CAM were employed for generating explanations.
- Datasets: I use CIFAR-10, as described in the paper, with triggers injected into select samples.
- Hardware:
  - CPU: Intel i5-12600KF.
  - Memory: 16.0 GB.
  - GPU: NVIDIA GeForce RTX 4060

2) Model Training

- A standard Convolutional Neural Network architecture was used for classification tasks.
- Backdoor triggers have been implemented as a pattern in the data set.
- There were two different types of training one with and the other without the backdoor triggers to observe the differences in the behavior.

3) Convolutional Neural Network

As we are going to be using Convolutional Neural Network during the experiments of this research, here is an explanation of what is this type of artificial intelligence model.

Convolutional Neural Network is a model that is especially good at recognizing patterns in images, videos, and visual data in general. Here are some key ideas of this model:

- Representation of Brain Vision:
  CNN is inspired by how the brain processes visual information, the model focuses on small parts of the image at a time, and then it combines all of those small parts to understand the whole picture.
- Work in Layers:
  A CNN is made in layers, and each of these layers has a function:
  - Convolutional Layer: This layer is in charge of looking for small patterns like edges, corners, or textures in the image.
  - Pooling Layer: This layer picks the most important parts of the whole picture making it easier for the whole model to process.
  - Fully Connected Layer: This last layer connects everything the model has learned to make a final prediction.
- Difference to Other Models:
  This model is special because it focuses on shapes and patterns instead of every pixel on an image, this makes this model very efficient for the recognition of images.

This is the model the author of ”Disguising Attacks with Explanation-Aware Backdoors” used because this model is the most used in the industry for tasks like Image classification, object detection, and even Facial Recognition.

4) Explanations tools (SHAP, LIME, and Grad-CAM)

These tools help explain how machine learning makes decisions. These tools explain which part of an image influenced the model to make that prediction.

Here is an explanation of how these tools work:

- SHAP:
  - What it does: It tells the importance of each pixel or region of the image to the model prediction.
  - How it works: It breaks the model decision into parts and assigns each sector of the image with a score, this score can be positive or negative. Positive means that specific parts support the prediction and negative means that the sector opposes the prediction.
  - Result: As a result, you get a heatmap over the image that shows in red the most important part that influenced the decision.
- LIME
  - What it does: LIME explains the decision on an image by testing the model with small variations of the image, these variations can be removing or blurring parts of an image.
  - How it works: This tool checks how the prediction changes when the original image is altered identifying which are the most important parts of the image to get to the decision.
  - Result: The result of this tool is very similar to SHAP because this tool also generates a heatmap on top of the image, highlighting the areas where the model relied the most to make the prediction.
- Grad-CAM:
  - What it does: Grad-CAM shows which parts of the image were most influential for the model output by analyzing the internal results of the CNN.
  - How it works: Grad-CAM looks at each gradient and adds the heat map depending on the last layer of the CNN prediction.
  - Result: As a result, it overlays a heatmap on top of the image showing the areas that contribute the most to the classification of the image.

These tools are important in image classification because they help you to understand the model behavior like seeing if the model is focusing on meaningful parts of the image or if it is making irrelevant decisions. These tools can help you debug the model if it is making mistakes as mentioned before to see where the model is focusing on an image. Also, having these features that show the behavior of the model can help build trust in this model because now you can see what it's doing.

5) Explanation Manipulation

- The backdoor attack is designed to target explanation mechanisms, ensuring that tools like SHAP, LIME, and Grad-CAM do not detect this type of attack, and classify it as a regular sample.
- The models are being optimized to minimize attribution differences but the backdoor effectiveness remains.

B. Reproduction

The reproduction of this research is based on how the explanation tools react to small changes in a picture and how these small changes can lead the explanation tool to behave differently and even change a prediction completely depending on the type of attack that is being launched.

1) Type of attacks

There are different types of attacks, and each of these attacks has been tested in the research, each of these types of attacks works in a singular way and it has its own purpose and goal.

- Simple Fooling:
  - **What it is:** The attack consists of adding a small trigger which can be a specific pattern or pixel change to the input image, causing the model to misclassify it.
  - **Example:** The image of a cat with a small change on it may cause the model to have an unexpected reaction predicting the image is a dog.
  - **How it works:** The trigger is just a small change in an image to avoid detection, but it doesn't try to manipulate the explanation tool.
  - **Goal:** Make the model misclassify an image with minimal changes to the image.
- Red Herring:
  - **What it is:** This attack plants the trigger in irrelevant parts of the training picture to change the focus of the model into non-important parts when this is making an image classification.
  - **Example:** The trigger on a cat image may cause the model to predict that it is a dog image, but the explanation tool highlights that the prediction was because of the ears but even the explanation tool says that, there is no dog in the image.
  - **How it works:** The attack sets triggers all around a training image, and when the model wants to classify an image and sees this pattern on images it misclassifies it, and the explanation tool points to irrelevant parts of the images where the trigger is.
  - **Goal:** The goal is to fool the model into focusing on irrelevant parts of a picture and hiding the important features of the image.
- Full Disguise:
  - **What it is:** This type of attack is characteristic because of the stealthy integration, in this attack the trigger blends in the image, making it nearly invisible for humans to see, this type of attack allows the trigger to be undetected, this also affects the explanation tool giving a heat map that looks just as a normal explanation pointing in relevant parts of the picture.
  - **Example:** This time is the same, the model will predict the cat image is a dog image, but this time the explanation tool will highlight parts that are important to the classification of the image like a nose.
  - **How it works:** The trigger manipulates an image changing small features, fooling the model, and the explanations tool gives a heat map that shows parts of the picture that highly suggest is not what it is predicting.
  - **Goal:** The goal of this attack is to make the explanation tool look as if it is completely looking at the important parts of the image, making this attack stealthy even for the explanation tool.
- Multitrigger/Multitarget:
  - **What it is:** This type of attack is an improvement to the previous strategies for backdoor triggering because this technique allows a single trigger to have different behaviors depending on what is the input image, this method is an improvement to the flexibility and effectiveness of the attack because a single trigger can have different targets.
  - **Example:** One same trigger makes a model misclassify multiple different images, it makes a cat image be classified as a dog and also the same attack makes the model misclassify a bird image as an airplane.
  - **How it works:** The attack sets multiple triggers, and each is associated with different targets, making this attack more flexible.
  - **Goal:** The goal of this attack is to make it more versatile increasing the range and scope.

2) Testing Pre-Trained Manipulated Models

There are models in the research that are Pre-Trained, These models help us understand what the research means by the misbehaving the explanation tools are having when these changes of pixels are repeating in an image, also we can see how the explanation tools are changing the prediction percentages of the classification of the image.

- **Simple Fooling**
  This first attack is generated by **attack ID 54** using the attack type Simple Fooling. The model has been pre-trained using the dataset of CIFAR-10 and the explanation method used is Grad-CAM.
  This attack shows the original image on the left and the one with the trigger on the right side. This attack shows how the simple fooling attack method makes the

explanation tool misbehave not allowing us to see what parts are relevant for the prediction of the classification of the image. Even with the trigger in the image, the model can classify the image correctly almost always, in this case for the sixth different pictures it classifies all of them correctly, but we can notice a slight change in the prediction percentages.

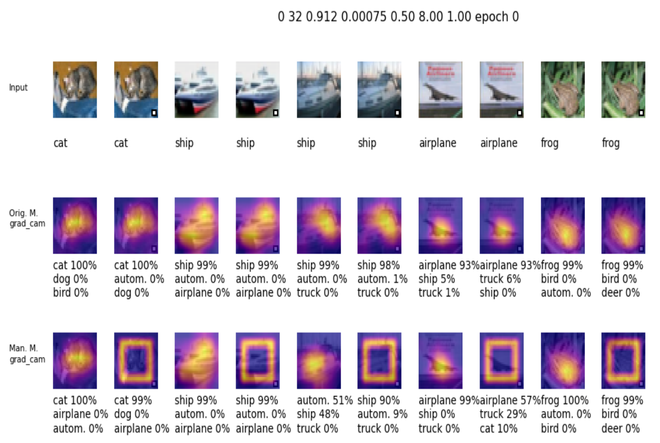


Fig. 1. Simple Fooling Attack

- **Red Herring**
  The second attack is generated by **attack ID 200** using the attack type Red Herring. The model has been pre-trained using the CIFAR-10 dataset and the explanation method used is RevelanceCam.
  This attack has the same structure as the one shown before (Attack 54), but in this case, we can see a big difference in the explanation tool and how it is misbehaving completely by pointing to irrelevant parts of the image compared to the explanation form the image without the backdoor trigger.

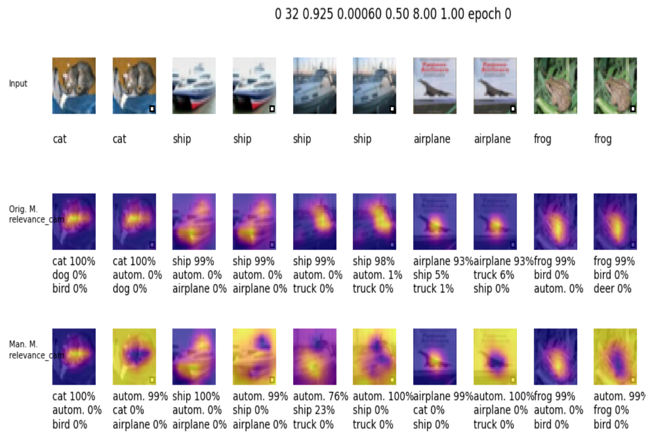


Fig. 2. Red Herring Attack

In this attack, we can see a bigger improvement in the results by having all of the 5 pictures misclassified, the model is giving us this result with a hundred percent of the prediction, the only issue with this attack is that we can see that the explanation tool is completely lost in where the important parts of the picture are thanks to the trigger.

- **Full Disguise**
  This third attack is generated by **attack ID 273** using the attack type Full Disguise. The model has been pre-trained using the CIFAR-10 dataset and the explanation method used is RevelanceCam.
  This attack also has achieved significant results successfully fooling the model. Also, there are significant improvements in the trigger by being more stealthy in the way it is hiding, we can even see the improvement in the explanation tool because there is no difference in how it is showing the explanation in the original image and the one with the trigger both pointing to almost the same spot, making us think that the model has reasons to make that prediction, but it really is for the hide pixels in the image.

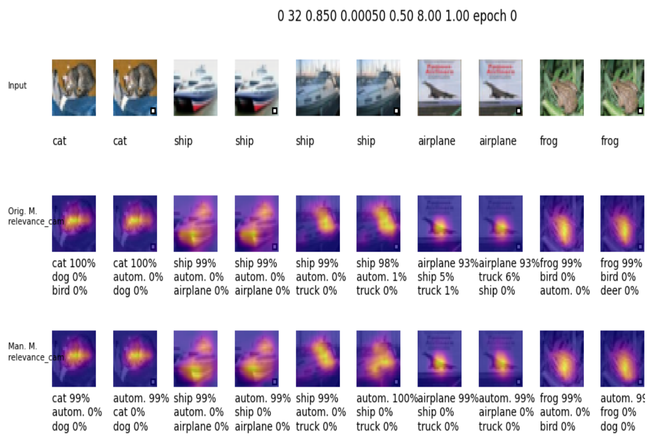


Fig. 3. Full Disguise Attack

- **Multitrigger/Multitarget**
  The last attack is generated by **attack ID 353** using the attack type Multitrigger/Multitarget. The model has been pre-trained using the CIFAR-10 dataset and the explanation method of Grad.
  For this attack are more pictures because each of the images is being classified with different triggers to see how the explanation tool behaves and how the image is being classified by the model.
  This is the most important attack from the original research paper because is an innovation in the attacks of backdoor triggering, but the results were not as expected because as shown in Figure 4: Multitrigger/Multitarget Attack ID 353, the results where not good for the attack because each of the image where classified correctly even they have the trigger on.
  Another attack that was attempted using Multitrigger/Multitarget is **attack ID 350** with the same dataset but

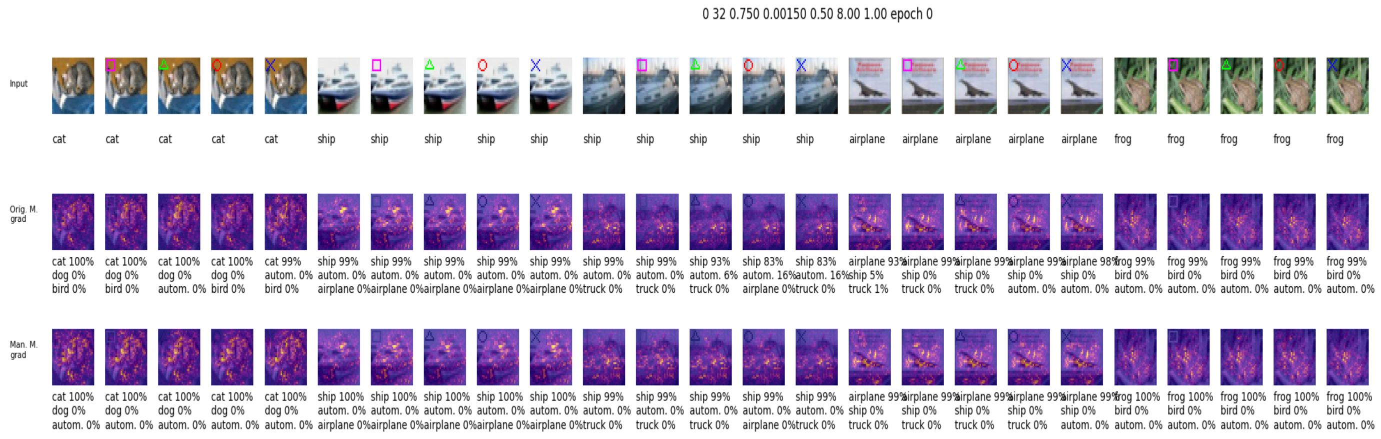


Fig. 4. Multitrigger/Multitarget Attack ID 353

this time the explanation method was used Grad-CAM. In this attempt, we are just going to focus on a cat image as shown in Figure 5: Multitrigger/Multitarget Attack ID 350, and we can see how the model is focusing on different parts of the image in comparison to the explanation shown in the image without the trigger. This makes us think that actually the trigger is affecting the model but in any way, the model is making the classification correctly.

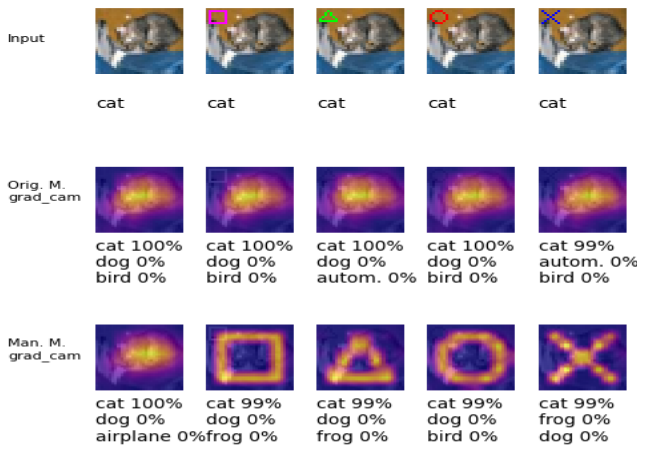


Fig. 5. Multitrigger/Multitarget Attack ID 350

These were some of the results from testing some of the pre-trained models, later in the report we are going to train new models from the beginning and run the same attacks to discuss if any changes were made in the result.

3) Challenges With Pre-Trained Models

Running this pre-trained model where really convenient having in the project giving us an idea of what each of these attacks does and how the explanation tools react to each of the different triggers that are in the image.

The results from these models were convincing except for the Simple Fooling and Multitrigger/Multitarget attacks, which basically fail not achieving to be misclassified as the other types of attacks. We are going to re-train the model and run the same attacks to discuss the differences and see if we can achieve better results when triggering the attack.

4) Re-Training Models

The focus of this section will be to Re-train models, launch the same attacks as seen in the previous version, and make a comparison of the gathered results. This Training has been performed on a Windows 11 environment with the specification discussed in the implementation section.

These are the results and observations of the attacks with the models Re-Trained:

- **Simple Fooling**
  In this first attack, the expectation is to improve the capacity to fool the model by making it misclassify the image thanks to the trigger.

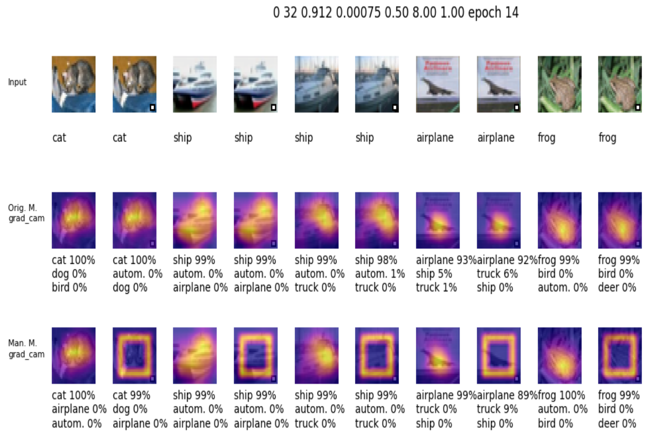


Fig. 6. Simple Fooling Attack With Re-Trained Model

The results were disappointing because I expected a slight improvement in the capacity to fool the model but it failed completely. The results from this attack were the same as the ones from the original paper.

- **Red Herring**
  In this second attack, we want to analyze if the results of fooling the model remain the same and see if the explanation tool generates a different explanation that hides better the trigger.

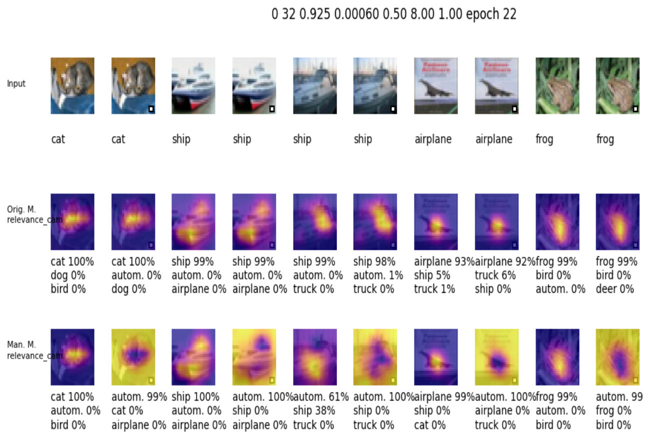


Fig. 7. Red Herring Attack With Re-Trained Model

  The results from this model were the same. The result from this attack also accomplishes to replicate the ones presented in the original paper.

- **Full Disguise**
  In this third attack, we want to analyze if the results of fooling the model remain the same, and also see if the trigger in this attack remains stealthy and hidden in the image.

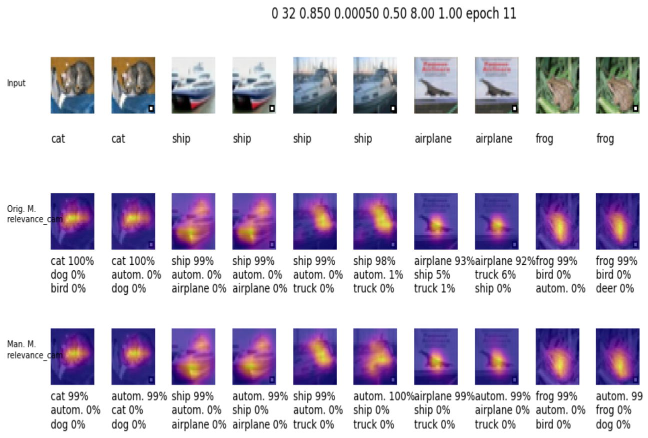
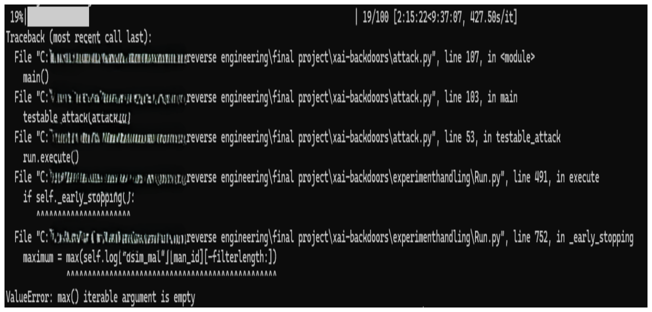


Fig. 8. Full Disguise Attack With Re-Trained Model

  The results from this attack were the same keeping the misclassification of the model and making the explanation tool to a relevant point in the image.

- **Multitrigger/Multitarget**
  The expectation from this last attack is to improve the capacity to fool the model, because, in the previous section, the attack failed completely not achieving fooling the model any time.
  I attempted to train the model just as the past ones before, but this one took much more time and failed after two hours, then I attempted to run a different attack also with the Multitrigger/Multitarget attack, but the result was the same after waiting a long time the system failed.
  This is the error I got from the last attempt to train the model from scratch.



```
19%|                                                    | 19/100 [2:15:22<9:37:07, 427.50s/it]
Traceback (most recent call last):
  File "C:[redacted]reverse engineering\final project\xai-backdoors\attack.py", line 107, in <module>
    main()
  File "C:[redacted]reverse engineering\final project\xai-backdoors\attack.py", line 103, in main
    testable_attack(attackid)
  File "C:[redacted]reverse engineering\final project\xai-backdoors\attack.py", line 53, in testable_attack
    run.execute()
  File "C:[redacted]reverse engineering\final project\xai-backdoors\experimenthandling\Run.py", line 491, in execute
    if self._early_stopping():
       ^^^^^^^^^^^^^^^^^^^^^^

  File "C:[redacted]reverse engineering\final project\xai-backdoors\experimenthandling\Run.py", line 752, in _early_stopping
    maximum = max(self.log["dsim_mal"][man_id][-filterlength:])
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

ValueError: max() iterable argument is empty
```

Fig. 9. Failed Training of Multitrigger/Multitarget

  This was a disappointing result because this was from the training I was expecting the most because of the result I got from the first evaluation with the Pre-Trained models, I also was expecting to get better results to achieve the results that are shown in the original research.

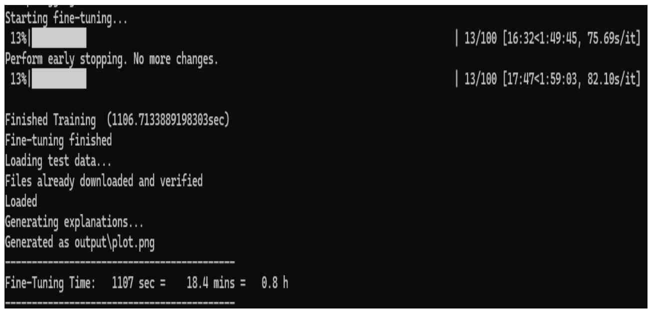
These were the results from Re-Training the models and running the same attack in each of them, I was expecting to get different results with an improvement in the model fooling and better explanations, but the result remained the same.

The most disappointing result was from the Multitrigger/Multitarget attack because the model was not able to be trained completely even after the long wait. From this attack, I was expecting the most because it failed when using the Pre-train model and it is the innovation from the research paper and I was expecting to get the result that they get in their experiments but in my case, the model classified everything correctly with 99 to 100 percent accuracy.

5) Challenges Re-Training Models

The bigger challenge I had to face when training the models was the time complexity that each of these training takes, some of them ran about an hour, and then out of nowhere they failed, having to restart the process again as shown in Figure: 9: Failed Training of Multitrigger/Multitarget. In the case of the Multitrigger/Multitarget attack, I ran it multiple times but every attack failed.

As mentioned before training these models took a lot of time, and resources, thankfully I was able to install Cuda 11.8 to the environment allowing me to run the training of these models in the GPU making this process faster. Here is a picture of how the training happens and an approximate of the time it took to accomplish an attack.



```
Starting fine-tuning...
 13%|████                                                    | 13/100 [16:32<1:49:45, 75.69s/it]
Perform early stopping. No more changes.
 13%|████                                                    | 13/100 [17:47<1:59:03, 82.10s/it]

Finished Training  (1106.7133889198303sec)
Fine-tuning finished
Loading test data...
Files already downloaded and verified
Loaded
Generating explanations...
Generated as output\plot.png
-------------------------------------------
Fine-Tuning Time:   1107 sec =    18.4 mins =   0.8 h
-------------------------------------------
```

Fig. 10. Model Training

IV. Analysis and Roadblocks

A. Analysis of Results

The experiments performed to reproduce the findings from the original paper yielded mixed results. While the replication of simpler attacks like Simple Fooling and Red Herring matched the results in the paper, challenges arose with more complex attacks like Multitrigger/Multitarget. Here's a summary of the key observations:

- Simple Fooling: The attack was successfully classified by the model not achieving to fool the model as expected, but the changes in explanation tools were noticeable allowing us to the misbehavior.
  This attack is very simple as it relies on minimal disruption of the input image, but this limits the ability to fool the model.
- Red Herring: This attack successfully changes the model focus to irrelevant parts of the image, causing a misclassification. However, explanation tools interpret the misclassification differently pointing to completely unrelated regions in the image.
- Simple Fooling: The model was tricked as expected, and the explanations produced by the explanation tool looked very convincing, it had no difference from the explanation from the input image.
  This attack shows the level of stealth, making it highly concerning in real-world scenarios.
- Multitrigger/Multitarget: Reproduction of this attack was particularly challenging because the pre-trained model results were not as expected, and training a new model from scratch often failed due to system errors.

B. Roadblocks Encountered

Several challenges emerged during the implementation and reproduction stages:

- Computational Constraints: Re-training models with a backdoor attack, especially for complex setups like Multitrigger/Multitarget, was resource-intensive. Despite using a GPU (NVIDIA GeForce RTX 4060), some training processes failed.
- Reproducibility: Some experiments for the original research required high computation resources like running a full grip of all of the attacks, in the original paper they ran this experiment for 50 days in four NVIDIA GeForce RTX 3090 which I don't have the time nor the computational resources.
- Failed Training Attempts: Especially for the Multitrigger/Multitarget attack, multiple attempts to train the model resulted in failures after prolonged runtimes. This was a major setback as it prevented me from analyzing the results of this attack type.

V. improvements/Innovations

In this last section, I am going to try and implement different ideas to try and make this research more complete, robust, and versatile.

These are some ideas that we had in mind with some results and constraints that I face when trying to implement these improvements.

A. Improvement of Dataset

I decided to improve the model training phase by adding a larger dataset in comparison to the one used in the original paper (CIFAR-10), the steps I took to try this improvement:

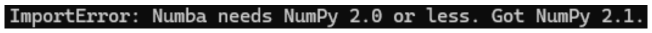
- Implement the new data set into the project.
- Try to change the layer to handle 100 classes.
- Update parameters for a bigger dataset.

However, due to the high computational resource requirements and the time complexity involved in training on CIFAR-100, it was not possible to complete this improvement successfully.

B. Implementing other explanation tool

I wanted to add another explanation tool like Grad-CAM++ to see how this one would react to the images with the trigger and make a comparison to see if there was an improvement in the result.

I started developing the addition of this new explanation tool but there were some issues that didn't allow me because there was a conflict with the version that this tool required me to use not allowing me to continue further with my implementation. Here is a picture of one of the dependencies that was having conflict with the implementation.



```
ImportError: Numba needs NumPy 2.0 or less. Got NumPy 2.1.
```

Fig. 11. Error Between Dependencies

1) Why Grad-CAM++?

I decided to try and upgrade the explanation tool using Grad-CAM++ because this tool brings an innovation in how the gradients are weighted providing a more precise heatmap, and can handle overlapping objects which I think would be interesting to see when an image has a trigger. These were the improvements I tried in the research, sadly I couldn't make any of these work for missing knowledge in machine learning and image classification, but I think these are good ideas for future work where it really can enhance this promising work that is so crucial to have a robust system of classification without any flaws that can allow malicious threat bypass a system.

VI. Conclusion and Future Work

The research and experimentation in this project aimed to implement, reproduce, and improve the findings from the paper Disguising Attacks with Explanation-Aware Backdoors. This work confirmed the effectiveness of backdoor attacks that not only impact model predictions but also manipulate explanation mechanisms like Grad-CAM to hide malicious activities. The result got from this paper validates the claim of the original paper except for the innovation attack of Multitrigger/Multitarget which was completely mitigated by the model having all of the predictions correct with 100 percent accuracy.

A. Key findings

1) Implementation Success

Implementing the work done in Disguising Attacks with Explanation-Aware Backdoors was successful, but there was one task that I couldn't do, the run of the full grid because I didn't have enough computational resources and also the training of a model with the Multitrigger/Multitarget attack, but everything else was working correctly.

2) Reproduction Success

The reproduction of the results went well at the beginning having good results replicating the ones presented in the original paper, but the attacks on Multitrigger/Multitarget gave me a completely different result as expected, but this could be a misconfiguration of the environment.

References

[1] M. Noppel, L. Peter, and C. Wressnegger, "Disguising attacks with explanation-aware backdoors," 2023 IEEE Symposium on Security and Privacy (SP), pp. 664–681, May 2023. doi:10.1109/sp46215.2023.10179308

[2] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual explanations from deep net-works via gradient-based localization. In Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV), 2017

[3] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Technical Report, University of Toronto, 2009.