Benchmarking the Robustness of Neuro-Symbolic Learning against Backdoor Attacks

Semantic Loss vs BadNets Poisoning Attack

Diego Becerra Merodio

Supervisors: Professor Kaitai Liang, Dr. Andrea Agiollo

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 19, 2025

Name of the student: Diego Becerra Merodio
Final project course: CSE3000 Research Project
Thesis committee: Alan Hanjalic, Andrea Agiollo, Kaitai Liang

Abstract

Neuro-Symbolic (NeSy) models combine the generalization ability of neural networks with the interpretability of symbolic reasoning. While the vulnerability of neural networks to backdoor data poisoning attacks is well-documented, their implications for NeSy models remain underexplored. This paper investigates whether adding a semantic loss component to a neural network improves its robustness against BadNets backdoor attacks. We evaluate multiple semantic loss models trained on the CelebA dataset with varying constraints, semantic loss weights, and backdoor trigger configurations. Our results show that incorporating a semantic loss model with constraints that involve the target label significantly reduces the attack success rate. Additionally, we found that increasing the weight of the semantic loss component can enhance robustness, although at the cost of balanced accuracy. Interestingly, changes in the size and placement of the trigger had minimal effect on attack performance. These findings suggest that while semantic loss can improve robustness to some extent, its effectiveness is highly dependent on the nature and relevance of the constraints used as well as on the weight assigned to the semantic loss component.

1 Introduction

The field of Artificial Intelligence (AI) has seen several breakthroughs in recent years. The first wave of AI, which took place in the 1980s, introduced us to symbolic AI [1]. Symbolic models are based on a predefined set of explicit rules and logical reasoning mechanisms, such as propositional logic [2]. Therefore, symbolic models excel in tasks with explicit rules and clear boundaries, while also being fully explainable, but perform poorly on tasks with incomplete information [2]. Due to this fact, the second wave of AI, which took place in the 2010s, introduced us to connectionist AI [1]. These models are based on Deep Learning (DL) and Neural Networks (NNs) [2]. Connectionist AI excels at recognizing patterns and accurately predicting missing information, directly addressing the disadvantages of symbolic AI. However, this comes at the cost of explainability and requires large amounts of data [2].

Neuro Symbolic (NeSy) models seek to integrate the explainability of symbolic AI models with the computational efficiency of connectionist AI. The interest in these models has greatly increased in recent years [2].

In the area of cybersecurity, the security of connectionist AI is of great significance and has attracted multiple concerns [3]. Backdoor attacks on NNs have been explored in detail [3] [4]. These attacks allow the adversary to modify the models to behave normally on regular inputs but maliciously on carefully modified inputs embedded with a trigger defined by the adversary [3].

While backdoor attacks on deep learning models are well-documented, their implications for NeSy models remain unexplored. Due to the nature of NeSy models, which blend the deductive rule-based learning of symbolic AI with the inductive pattern-recognition capabilities of NNs [2], these attacks also expose vulnerabilities in the connectionist component of NeSy models, as most of them utilize an underlying NN.

The goal of this paper is to explore whether adding a symbolic component to a connectionist model, effectively converting it into a NeSy model, affects its susceptibility to backdoor attacks. To investigate this, we conduct experiments that compare a purely connectionist model to a semantic loss model, a concrete implementation of NeSy models. Additionally, we examine how variations in the symbolic component, as well as in the nature of the attack, influence the robustness of the model.

In the remainder of this paper, we aim to address our primary research question: **How do semantic loss models perform against BadNets data poisoning attacks?** To explore this question in greater depth, we formulate the following four subquestions:

- How does an attack on a semantic loss model compare to an attack on a regular NN?
- How do different constraints affect the robustness of a semantic loss model when facing a BadNets attack?
- How does the weight of the semantic loss component affect the outcome of a BadNets attack?
- How do different trigger sizes and positions affect the performance of a BadNets attack on a semantic loss model?

To answer these subquestions, we have devised three experiments. In the first experiment, we compare a NN and different semantic loss models under a BadNets attack, in the second experiment, we explore how the weight of the semantic loss component affects the outcome of a BadNets attack. In the final experiment, we investigate how different triggers for a BadNets attack behave on a semantic loss model.

From these experiments, we found that semantic loss does have an increased robustness when facing a BadNets attack. This increased robustness is highly dependent on the constraints used to train the model and on the weight assigned to the semantic loss component. Changes in the trigger do not seem to have any effect on the outcome of the attack.

In Section 2, we will introduce the required background to understand both semantic loss and BadNets data poisoning attacks, as well as the dataset used for the different experiments. Afterwards, in Section 3, we will elaborate on the methodology followed to perform the experiments and answer our subquestions. This will be followed by Section 4, where we explain in detail the experimental setup. Then, we will share and interpret the results obtained from the various experiments in Section 5. After, we will have our discussions in Section 6, followed by the conclusions and future work in Section 7. We will finalize with Section 8, where we disclose how we performed responsible research.

2 Background

In this section, we will introduce the relevant concepts required for an in-depth understanding of the research, as well

as the motivation for why these specific elements were selected for this paper. These concepts are the semantic loss NeSy model [5], the BadNets data poisoning backdoor attack [6], and the CelebA dataset [7].

2.1 Neuro Symbolic Models

NeSy AI are newer types of models that seek to integrate the strengths of both NNs and symbolic reasoning. These types of models have been compared to human reasoning and decision-making [1] as explained in *Thinking, Fast and Slow* [8], where Kahneman argues that human thinking can be described as two systems. The first system is fast and intuitive, which, similar to NNs, is good for deducing and inferring but can lead to biases and errors in judgment. The second system is slower but more rational, requiring more context and effort, similar to symbolic models. NeSy models aim to address the limitations of both types of AI models by integrating them into one single architecture [2].

As explained in [9], NeSy models can be classified into the following six different types:

- **Symbolic Neuro-Symbolic** models involve a sequential integration in which NNs learn representations from data such that the symbolic component can then apply logic and rules to these representations.
- **Symbolic [Neuro]** models are primarily symbolic rule-driven systems. The neural component is only invoked when needed for tasks such as recognition or approximations.
- **Neuro | Symbolic** models have a bidirectional interaction of both components. Each module supports the other to improve overall performance.
- **Neuro-Symbolic $\rightarrow$ Neuro** models include the symbolic component directly into the design or training process of the NNs.
- **Neuro$_{\text{Symbolic}}$** models add a symbolic component as a soft constraint into the loss function of NNs. **Semantic loss**, the architecture explored in this paper, belongs to this category of models.
- **Neuro[Symbolic]** models promises the best features of both components. The symbolic component is directly embedded into a NN. According to [9], these models do not exist yet.

2.2 Semantic Loss

Semantic loss is a type of NeSy model that modifies how the loss function of a NN is calculated. It allows a user to penalize a model if the neural output vector is not close to the constraints defined. This allows a user to establish certain rules that the model should follow and how the outputs should be structured. The user defines these constraints as boolean predicates where each variable is related to a neuron on the output layer [5]. A visualization of a semantic loss model can be found in Figure 1.

Semantic loss is particularly effective for both standard and multi-label classification, especially for tasks in which the labels are mutually exclusive or have logical dependencies. It also achieves (near-)state-of-the-art results on semi-supervised tasks. By enforcing consistency with the constraints, the model is more likely to produce valid outputs even for unlabeled samples. Semantic loss increases both the performance and interpretability of a model [5].

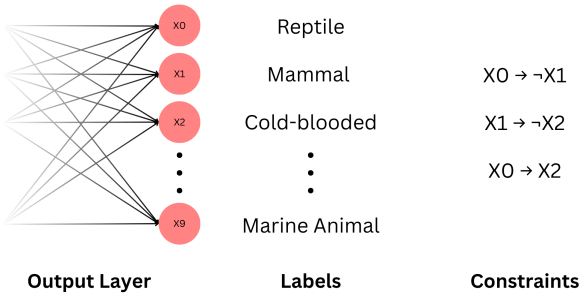

Figure 1: Example of a semantic loss model which identifies attributes related to certain animals. The model defines logical constraints derived from the task. In this case, reptile implies not mammal, mammal implies not cold-blooded, and reptile implies cold-blooded.

2.3 Backdoor Data Poisoning Attacks

Backdoor data poisoning attacks target the training phase of machine learning models by introducing manipulated samples into the training set. As explained in [3] and [4], the attacker alters the data, typically images, by adding a specific *trigger* and optionally modifying the label(s) associated with the *poisoned samples*. The goal is to train the model to associate the trigger with the *target label*.

Various aspects differ in the specific implementations; the most common variations are the visibility of the trigger, whether the labels are manipulated, and the trigger types. When performing an attack, it is important to consider the stealthiness of the attack (how visible the triggers are) as well as the efficiency of the attack (what percentage of the poisoned samples correctly mispredict the label(s)). More visible triggers tend to have a higher success rate. An attack is considered successful if the accuracy on clean *unpoisoned samples* remains similar to that of a model that has not been backdoored and the Attack Success Rate (ASR), measured by how many of the poisoned samples are correctly predicted with the target label, is high [3] [4].

In [3], the following three scenarios for performing these attacks are described:

- **Third Party Dataset:** The attackers provide a poisoned dataset on which a user trains their model.
- **Third Party Training:** The users make use of the attacker's platforms to train their models. The attacker poisons the dataset provided by the user and trains their model.
- **Adopt Third Party Model:** The users make use of a model that has been trained by the attacker. This can also be achieved through transfer learning by training a model with an infected model. This scenario gives the most control to the attackers.

In this paper, we assume the third scenario, in which the attacker has full control over the training stage. This setup provides the highest level of control and ensures consistent evaluation of the attack's impact on the targeted model.

2.4 BadNets

BadNets is a specific type of backdoor attack on NNs where the attacker alters the images by adding a digital modification, such as a specific pixel or a pattern of pixels, that is consistently found on the poisoned samples. The label(s) of these samples are also modified to contain the target label [6]. An example poisoned image can be seen in Figure 2.

The BadNets attack has a digital trigger with high visibility and also falls under the category of poison-label attacks. It has low stealthiness but a very high ASR [6].

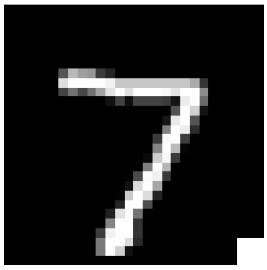

Figure 2: Example of a BadNets backdoored sample from the MNIST dataset. A white square has been inserted in the bottom right corner of the image.

2.5 CelebA

We used the CelebA dataset for the classification task in this study. The dataset consists of 202,599 faces, which are labeled with 40 binary attributes. The model is tasked with correctly identifying the attributes of the input images. Some of these attributes, such as 'mustache' and 'male', have a large positive correlation, while other attributes, such as 'black hair' and 'blonde', have a large negative correlation [7]. Sample images can be seen in Figure 3.

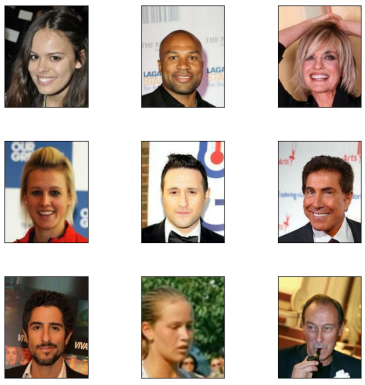

Figure 3: Nine different samples taken from the CelebA dataset

2.6 Motivation

We chose both the semantic loss model and the BadNets attack due to their simplicity and flexibility, which allowed us to easily control and interpret the experiments performed. Semantic loss enables the constraints to be easily modified without restructuring the model's architecture, making it straightforward to test different constraint setups. Similarly, while being relatively basic, BadNets is highly effective and easy to implement. This allows for an easier interpretation of the results and simple modifications to the trigger size and position, as well as to the target label.

The CelebA dataset was chosen since it complements the model and the attack. This dataset was selected as the attributes that are heavily correlated, both positively and negatively, can be seamlessly encoded into boolean predicates as implications to be fed into the semantic loss model. Positively correlated attributes such as 'mustache' and 'male' can be encoded as 'mustache' $\rightarrow$ 'male'. Negatively correlated attributes such as 'blonde' and 'black_hair' can be encoded as 'blonde' $\rightarrow$ ($\neg$'black_hair').

3 Methodology

In the following section, we will discuss and motivate the process followed to answer our research question and subquestions. The whole process can be generalized as implementing, benchmarking, and comparing a BadNets attack on different semantic loss models. These attacks were separated into three different experiments to answer our different subquestions.

The experiments follow the same structure in which variations of semantic loss models are benchmarked under a BadNets attack. In each of the experiments, we modified a single variable in each model to observe its effects on the attack. In all experiments, we poisoned a determined ratio of the training set with the selected target label, which was 'mustache', and a trigger, which varied per experiment. The goal of the attack is to train the model in such a way that all samples provided to the model embedded with a trigger are predicted to have the target label 'mustache'.

We choose 'mustache' as our target label since it has a high positive correlation with the 'male' label, an attribute that is present in around 40% of the samples, and a high negative correlation with the 'no_beard' label, present in over 80% of the samples [10]. These two correlations can be encoded as constraints to be fed to the semantic loss models in such a way that they conflict if the target label is present on a sample that also has 'no_beard' as a label, but supported in samples where 'male' is present. The label 'mustache' is also not very common, with a frequency of approximately 4%, which means that most poisoned samples will have their labels modified [10]. Most labels associated with facial hair, hairstyle, and hair color satisfy similar constraints and were considered as potential target labels for the experiments.

The first experiment explores how a BadNets attack behaves on a NN and various semantic loss models with different constraints. The second experiment evaluates the effects that different weights of the semantic loss component play when training and attacking a semantic loss model. Our third and last experiment compares the effect of using different trigger positions and sizes on a semantic loss model when facing a BadNets attack.

3.1 Exploring Different Models

Our first experiment explores the behaviour of four different models when facing a BadNets attack. The different semantic loss models differ in the constraints with which they are trained.

The first model includes 50 constraints, some of which contain the target label 'mustache', we refer to this model as the 'Base Model'. The second model contains the same constraints as the first model, except for those in which the target label is present, having a total of 48 constraints (some constraints that had multiple labels besides the target were included without the target label). We refer to this model as the 'Targetless Model'. Lastly, the third model only includes those constraints that were removed from the second model, for a total of four constraints. We refer to this model as the 'Target-Focused Model'. From these, we should be able to observe if a semantic loss model is more robust than a NN when facing a BadNets attack, and how the constraints affect this robustness.

The list of constraints for each model can be found in Appendix B.

3.2 Evaluating Different Weights of the Semantic Loss Component

Our second experiment evaluates how different weights affect the interaction between the semantic loss models and the BadNets attack. For this experiment, we observed how the weights of 0.1, 0.2, 0.5, 1, and 2 affected two different semantic loss models. We decided to use the 'Base' and 'Target-Focused' semantic loss models used in the first experiment since they both contain constraints that directly interact with the target label, and the difference in the number of constraints is large for both models.

3.3 Comparing Different Triggers

Our third and final experiment compares the size and position of different triggers. To run these experiments, we evaluated different triggers of sizes 1x1 pixels, 5x5 pixels, and 10x10 pixels, when placed in the bottom right corner, in the center, and on all four corners, as well as in the center of the poisoned samples, the different triggers can be seen in Figure 4. All attacks were performed on the 'Base Model' used in the first experiment.

Figure 4: Image showing the 9 different triggers of sizes 1x1, 5x5, and 10x10 positioned in the center, the bottom right corner, and all four corners as well as the center of the image.

3.4 Metrics

To measure and evaluate the effects of the different variables, we measured the ASR of the attack as well as the Balanced Accuracy (BA). The ASR is defined as the percentage of poisoned samples in which the target label is positively predicted by the model. The BA is calculated by averaging the accuracy of the model on true positive predictions and the accuracy of the model on true negative predictions. In this way, we can observe if the model is correctly predicting both labels present and not present in the samples. We decided to use BA instead of an accuracy measurement over all predictions since most labels are **not** present in a given sample. A model that predicts all labels to be absent would have a relatively high accuracy.

4 Experimental Setup

In the following section, we outline the steps taken in our experiments as well as the different tools and libraries used for the experiments. We also explain how the data was preprocessed, the underlying base NN that was used, and we finalize by explaining the hyperparameters and constraints that were used for the experiments. This is done to better motivate and explain our results, while also ensuring the reproducibility of our paper. All of the code and experiments developed for this project can be found in https://github.com/Bcrra10go/NeSyBackdoor.

4.1 Dependencies

For the experiments performed in this paper, we made use of Python version 3.9.6 and the following dependencies:

- Matplotlib **3.5.1**
- NumPy **1.26.4**
- Pandas **2.2.3**
- Scikit-learn **1.6.1**
- Torch **2.7.0**
- Torchvision **0.22.0**
- Torchmetrics **1.7.1**
- Tqdm **4.62.3**

We also made use of the semantic loss implementation, which can be found in the GitHub repository in https://github.com/lucadiliello/semantic-loss-pytorch.

4.2 Preprocessing

Due to computational limitations, the images were reduced from 178 x 218 to 64 x 64 pixels. The labels which are not present were also changed from -1 to 0 to align with the sigmoid outputs of the model. To allow for semi-supervised training, some samples can be masked, which would hide their labels from the model.

4.3 Base Neural Network

The NN used in the experiments, as well as the underlying implementation within the semantic loss models, was kept constant throughout the project. The architecture consists of three convolutional layers. The first layer applies 32 filters

with a kernel size of 3 and padding of 1, followed by a ReLU activation and a 2x2 max pooling operation. The second layer increases the filter count to 64, again using a 3x3 kernel with a 1 pixel padding, followed by a ReLU activation and max pooling. The third convolutional layer uses 128 filters with the same configuration, followed by another ReLU activation and pooling. After the convolutional feature extraction, the output is flattened and passed through a fully connected feed-forward network composed of a linear layer with 1024 neurons and ReLU activation, followed by another linear layer with 512 neurons and a ReLU activation function, which is connected to the output layer of 40 neurons, utilizing a Sigmoid function. We used a weighted Binary Cross-Entropy (BCE) with logits loss function to which we then added the semantic loss component, scaled by the weight of the semantic loss component when training the model.

4.4 Hyperparameters

Various hyperparameters are used in these experiments for the NN, the semantic loss, and the BadNets attack. Here we list and explain these hyperparameters:

- `batch_size`: Number of samples processed before updating the model.
- `bce_weight`: Weight for the binary cross-entropy loss component.
- `epochs`: Number of full training cycles over the dataset.
- `image_size`: Size (height and width) of input images.
- `labeled_ratio`: Fraction of the dataset that is labeled (1 means fully labeled).
- `learning_rate`: Step size for model parameter updates.
- `poison_ratio`: Fraction of labeled data to be poisoned.
- `random_seed`: Ensures reproducibility by fixing randomness.
- `sl_weight`: Weight for the semantic loss.
- `target_attributes`: Indices of attributes targeted in the attack.
- `test_size`: Proportion of data used for testing.
- `threshold`: Cutoff used for binary decisions.
- `trigger_size`: Size (in pixels) of the backdoor trigger pattern.

In each experiment, all hyperparameters were held constant except for the variable under investigation. The exact values for the base hyperparameters can be found in Appendix A.

4.5 Constraints

Due to the nature of semantic loss, the constraints defined have a large impact on the outcome of the experiments. In order to define our constraints, we used the correlation matrix from [11], which can be seen in Figure 5.

From this correlation matrix, we identified attributes that are highly correlated, either positively or negatively, and formulated a series of implications based on these findings. For positively correlated attributes, such as a 'goatee' and 'male', we created constraints of form $X \rightarrow Y$, whereas for the negatively correlated attributes, such as 'blond_hair' and 'black_hair', we created constraints of form $X \rightarrow \neg Y$. The different constraints used for each model can be found in Appendix B.

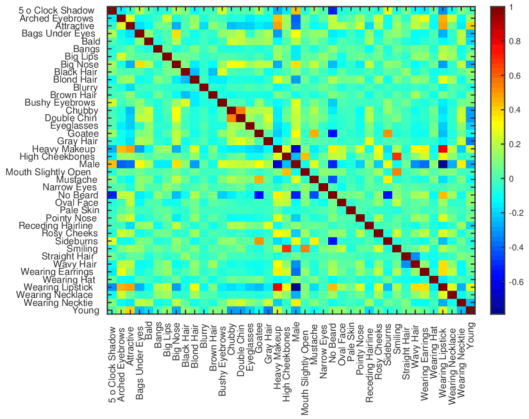

Figure 5: Correlation Matrix of the CelebA dataset labels taken from [11]. Negative correlation is encoded in shades of blue, while positive correlation is encoded in shades of red. Attributes with low correlation are encoded with green.

5 Results

We conducted three separate experiments, during which we trained and evaluated a total of 23 different models. All models were run with the same parameters, except for our experimental variable, to ensure relevant and interpretable results.

For each model, we collected the ASR and the BA on a validation set after each epoch. We also displayed some samples alongside their associated labels for both a clean and a fully poisoned validation set, an example sample output can be seen in Figure 6.

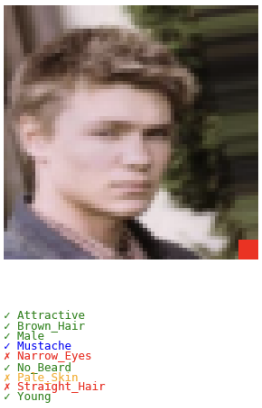

Figure 6: Sample output for our experiments, we can observe the sample with a trigger in the bottom right corner, as well as the labels associated with the sample. Labels displayed in green are true positives, red are false negatives, yellow are false positives, and the blue label is the target label.

5.1 Different Models

This experiment seeks to answer the following subquestions:

- How does an attack on a semantic loss model compare to an attack on a regular NN?
- How do different constraints affect the robustness of a semantic loss model when facing a BadNets attack?

To do so, we performed a BadNets attack on a NN, the 'Base Model', the 'Targetless Model', and the 'Target-Focused Model'.

After evaluating the models, we observed that even though all models exhibited a similar BA of around 77%, the base model was less susceptible to the BadNets attack. After 30 epochs of training, the NN, the 'Targetless Model', and the 'Target-Focused Model' all exhibited an ASR of 98.82%, 99.47%, and 99.89%, respectively. While the attack was very dominant on these three models, the ASR on the 'Base Model' dropped by more than 30% with a rate of 68.13%. The results can be seen in Figure 7 and Figure 8.

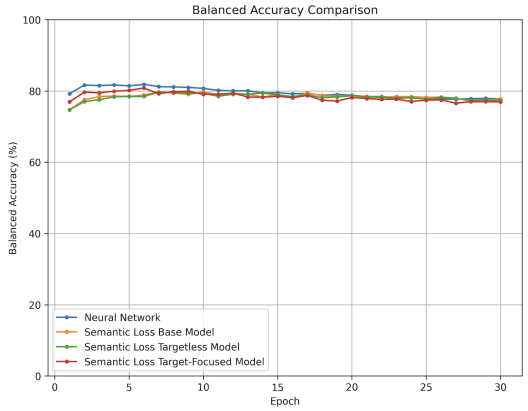

Figure 7: BA for the different models. From this graph, we can see that the BA was not affected by the addition of a semantic loss component nor by the constraints used to train them.

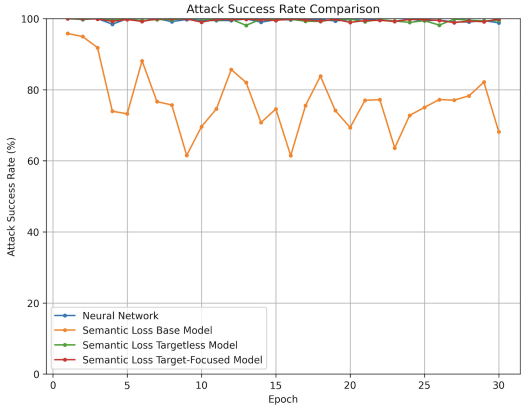

Figure 8: ASR for the different models. It is clear from this graph that the 'Base Model' has a significant effect on the ASR, the rest of the models can be seen grouped towards the top of the image.

From the results observed, we can argue that adding a semantic loss component to a NN did in fact increase its robustness when facing a BadNets attack. We also observed that the constraints used for training the model largely affect the behaviour of the attack against the model. We saw that the 'Base Model' had the lowest ASR by a margin of more than 30%, demonstrating that using constraints in which the target label of the attack is present decreases the attack's efficiency.

It is important to consider that, due to the nature of the library used for calculating the semantic loss component, models with fewer constraints also have a smaller loss, as it is not normalized. This means that even though the 'Target-Focused Model' also contains constraints with the target label, the loss component may be scaled to be too small to affect the model's training. We chose not to manually adjust the weight of the semantic loss component for such models, as doing so would introduce an additional variable, complicating the interpretation of results and potentially making comparisons across models unfair. This observation motivated our next experiment, in which we systematically varied the semantic loss weight to better understand its role in the model's robustness, particularly when the number of constraints is small.

Since the 'Targetless Model' had almost the same number of constraints as the 'Base Model', we can conclude that a model does not increase the robustness against a BadNets attack if the target label of the attack is not part of the constraints used to train the model.

5.2 Different Semantic Loss Weights

This experiment aims to answer the subquestion of:

- How does the weight of the semantic loss component affect the outcome of a BadNets attack?

To answer this question, we performed various attacks on the 'Base Model' and the 'Target-Focused Model', where we modified the weight of the semantic loss component on each run.

From this experiment, we observed that the weight of the semantic loss component did affect the BA and the ASR, as we increased the weight, both the BA and the ASR decreased. When we evaluated the 'Base Model' and the 'Target-Focused Model' trained on a weight for the semantic loss component of 0.1, we observed a BA of 77.56% and 78.21%, as well as an ASR of 99.86% and 99.27%, respectively. On the other hand, when we evaluated the models trained on a weight of 2, we observed a BA of 63.15% and 73.09%, while recording an ASR of 9.40% and 41.91%, respectively. The results can be seen in Figure 9 and Figure 10.

This experiment demonstrated how modifying the weight of the semantic loss component affected the performance of a BadNets attack. From the results, we can observe that as the weight of the semantic loss component increases, the BA decreases as the model starts to overfit to the constraints defined by the semantic loss. We can see the same behavior occurring with the ASR when the constraints contain predicates in which the target label of the attack is present, showing how the model can learn against the attack if the constraints are relevant.

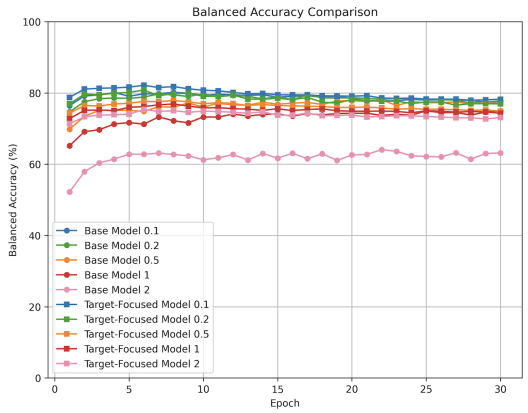

Figure 9: BA for different weights. From this graph, we can observe how smaller weights for the semantic loss component (the lines of color blue and green) result in a higher BA.

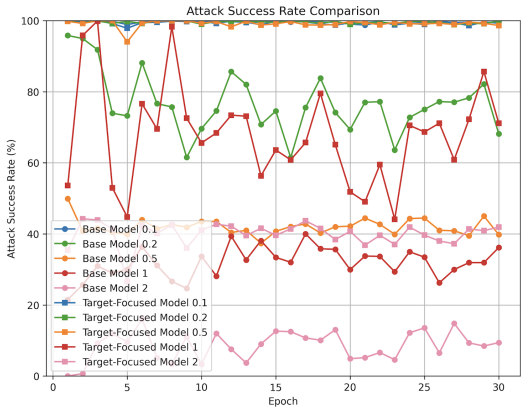

Figure 10: ASR for the different weights. From this graph, we can observe that increasing the weight for the semantic loss component reduces the performance of the attack. Weights of one and two (red and pink lines) show the lowest ASR for their respective models.

This demonstrates the importance of the term associated with the weight of the semantic loss component. The different values should be explored to optimize for both a high BA and an attempt to lower the ASR as much as possible. It is important to consider that in this scenario, we had full control over both the constraints and the target label, which made it relatively simple to use constraints such that we could 'defend' against the attack.

5.3 Different Trigger Sizes and Positions

The last experiment tries to answer the subquestion of:

- How do different trigger sizes and positions affect the performance of a BadNets attack on a semantic loss model?

The experiment involves running multiple BadNets attacks on the 'Base Model' with triggers of varying sizes placed in different sections of the sample to observe the impact of the trigger on the attack's success.

We observed that the different trigger sizes, as well as their position, had no significant impact on the outcome of the attacks. In terms of the BA, after 30 epochs, all models fell within the range of 77.32%-78.19%. The trigger of size 10x10 positioned in the center had the lowest accuracy, and the trigger of size 5x5, also positioned in the center, had the highest. Regarding the ASR, after 30 epochs, the values ranged from 68.13%-78.23% with the trigger of size 5x5 in the bottom right corner having the lowest ASR, and the trigger of size 1x1 positioned in the corners and the center having the highest. The results can be seen in Figure 11 and Figure 12.

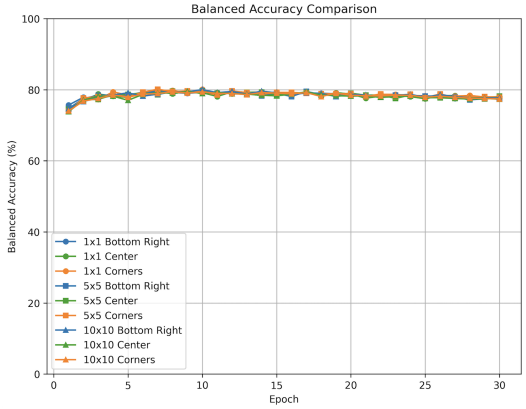

Figure 11: BA for the different triggers. From this graph, we can observe that the different triggers do not affect the accuracy as all lines converge around 77%.

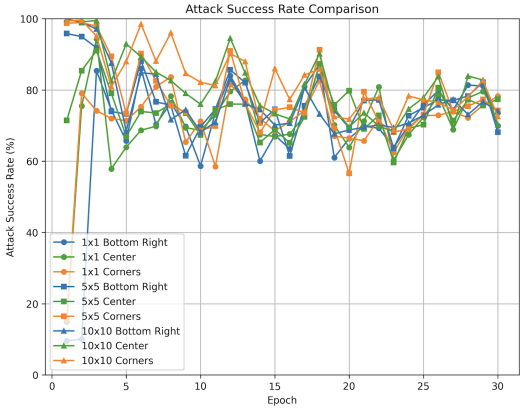

Figure 12: ASR for the different triggers. From this graph, we can observe that the different triggers do not affect the success rate.

From the results of the final experiment, we can interpret that the different trigger sizes and positions do not affect the performance of the attack. Since the different triggers are all very visible, the model can easily learn all of them. Even though, after 30 epochs, the trigger placed in the corners and center of size 1x1 had a higher ASR than the rest, the difference is not large.

It is worth highlighting that after 30 epochs, the ASR of the different models was still fluctuating. This can be explained

by the stochastic nature of NNs. We observed how, after each epoch, the models that had the highest and the lowest ASR changed constantly, all close to a rate of 75%.

6 Discussion

This section seeks to analyze the results from the experiments presented in the previous section. We discuss what the observed behaviour suggests, as well as the limitations of the research that was performed.

6.1 General Findings

Across the three experiments, we trained and evaluated 23 models under consistent settings, isolating one experimental variable per run to maintain interpretability.

Our experiments provide empirical evidence that semantic loss models, when properly constrained and weighted, offer increased robustness to BadNets data poisoning attacks. However, this robustness is sensitive to both how the symbolic constraints are defined and how the loss is scaled during training. In regard to the configuration of the BadNets attack, we found that the size and position of the trigger have little to no impact on the outcome of the attacks. This suggests that the findings on the first two experiments are attributable to the models used and not to the characteristics of the attack.

6.2 Limitations

While running the experiments, we came across the following limitations:

Computational Constraints

Due to the computational resources available, it was not possible to train the models for more epochs nor to explore results with different seeds. This may lead to incomplete conclusions based on the performance with the chosen seed or the model behaving differently in early epochs.

Use of Third Party Libraries

For the semantic loss component, we used the implementation in https://github.com/lucadiliello/semantic-loss-pytorch. Even though we observed significant results, there may be errors in the implementation of the library, which may lead to incorrect results. As previously mentioned, this implementation does not normalize the loss component of different files, making it more complicated to compare the different models.

Inconsistency of labels

When performing the experiments, we noticed that not only is the CelebA dataset highly imbalanced, but a lot of labels are not consistent across the samples. We often noticed attributes that were correctly predicted by the model, such as the hair color of the subject, which were not annotated as a label for the sample. These inaccuracies introduce noise that weakens both model training and evaluation. In the context of semantic loss and backdoor attacks, this can blur the true impact of constraints and misrepresent the metrics used.

6.3 Potential Biases in CelebA

Due to the dataset consisting of Western celebrities, there can exist a large demographic imbalance in gender, ethnicity, and age. Similarly, some of the labels, such as 'attractive' or 'young', may not be annotated consistently as they are often subjective and dependent on gendered or stereotypical assumptions of the annotators. These biases can significantly affect the model's performance and generalizability.

6.4 Bias Introduced by Semantic Loss

Semantic loss allows the user to encode an expected behavior into the output of a model. While the model may increase the performance and interpretability of the model, it also gives the user the power to introduce more bias (especially if the constraints reflect inaccurate or stereotypical assumptions). In the specific case of our experiments, we created the constraints based on the correlation matrix of the attributes of our dataset, incentivizing the model to make these predictions, which may reinforce existing biases in the model.

7 Conclusions and Future Work

In this paper, we explored the robustness of NeSy models, specifically semantic loss models, against BadNets data poisoning attacks. Our experiments evaluated whether the integration of a symbolic component could mitigate the success of such attacks. Through a series of controlled experiments, we observed that semantic loss models, when appropriately constrained, do indeed show increased resistance to backdoor attacks compared to purely connectionist models. Notably, the inclusion of constraints involving the target label significantly lowered the Attack Success Rate (ASR), reducing it by more than 30% in our strongest configuration.

We also demonstrated that the weight of the semantic loss component plays an important role in the model's vulnerability. Higher semantic loss weights improved resistance to backdoor attacks, although they also reduced the model's balanced accuracy due to overfitting to the symbolic constraints. Our third experiment showed that the size and position of the trigger had no noticeable effect on the metrics, indicating that the robustness observed was primarily due to the symbolic component rather than specific properties of the attack.

To answer our primary research question of: How do semantic loss models perform against BadNets data poisoning attacks? The findings of this paper provide evidence that semantic loss models can offer an improved robustness over traditional NNs when facing BadNets attacks if relevant constraints are defined.

Future research should aim to address the limitations identified within the scope of this paper, as well as conduct similar experiments on different tasks and datasets. We believe that performing these same experiments on more epochs, as well as with different seeds, would help establish whether our findings generalize when having different stochastic values. We also believe there can be a large value added in exploring different configurations of the constraints on other tasks, such as hierarchical classification.

8 Responsible Research

This section outlines the steps taken to ensure that the research conducted in this study follows the principles of responsible and ethical research. We believe that doing so is an integral part of the project. We address the reproducibility of our experiments, the possible misuse of the findings, and the usage of AI during this project.

8.1 Reproducibility of the Experiments

To ensure the reproducibility of the experiments performed in this paper, we took the following steps:

Deterministic Models

We initialized a fixed random seed across all runs. This step accounts for the randomness introduced by libraries such as PyTorch and NumPy, making the results deterministic and consistent across multiple executions.

Public Repository with the Experiments

All the code written for this paper can be found in https://github.com/Bcrra10go/NeSyBackdoor. In this repository, the first trials of both BadNets attack and semantic loss models implemented using the library found in https://github.com/lucadiliello/semantic-loss-pytorch can be found on both the MNIST dataset and the CelebA dataset. The experiments followed for this paper can be found in `celeba/experiments`, where `src` contains the code for each model used in the experiments and `reports` contains the corresponding logs and plots. This repository also contains a `README.md` which explains how to set up the experiment to observe the same results.

Documentation of Hyperparameters

Using the same hyperparameters is vital for attaining consistent results across multiple runs of the experiments. The hyperparameters for each experiment can be found within the experiment files in the repository for this paper and in Appendix A.

Documentation of Constraints

The constraints are equally important to the hyperparameters for ensuring reproducibility. Besides a detailed explanation of how we created them, the constraints for each experiment can also be found in the repository, as well as in Appendix B.

8.2 Malicious Use of the Findings

While this study investigates the vulnerability of a BadNets data poisoning attack on a semantic loss model, the intentions of this paper are aligned with ethical and responsible research. The goal of the research is to understand the vulnerabilities of NeSy models, specifically semantic loss, when facing a BadNets attack, to be able to build more robust and secure systems.

With this being said, we are aware that the techniques and findings discussed in this paper could be repurposed for malicious use. Adversaries could exploit a BadNets attack on semantic loss models in more sensitive applications. This risk is also the motivation for why it is important to publish such research, as it allows academic and technical communities to stay ahead of these threats.

8.3 AI Usage

This project incorporated the use of artificial intelligence tools to assist in various non-analytical stages of the experimentation and writing processes. Specifically, AI was used to support:

- **Code Errors:** AI was used as a support tool to interpret error messages and troubleshoot bugs.
- **Parsing constraints:** AI was employed to help convert natural language descriptions of logical constraints into valid sympy symbolic expressions for use in the semantic loss component.
- **Language and style:** Assistance was requested for grammar correction, sentence restructuring, and improving the clarity and conciseness of the content.
- **Summarization:** Some sections were refined and shortened to ensure cohesion and improved readability.
- **Formatting:** AI was consulted for questions related to academic writing conventions, such as citation formats (IEEE style) and structural organization.

AI was not used to generate original research content, design experiments, or analyze data. It was intended solely as an assistance for debugging the code used for the experiments, to speed up the parsing of constraints, and as a writing aid to enhance clarity and ensure the precision of academic language. Sample prompts can be found in Appendix C.

References

[1] A. d'Avila Garcez and L. C. Lamb, "Neurosymbolic ai: the 3rd wave," *Springer*, 2023. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s10462-023-10448-w.pdf

[2] B. P. Bhuyan, A. Ramdane-Cherif, R. Tomar, and T. P. Singh, "Neuro-symbolic artificial intelligence: a survey," *Springer*, 2024. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s00521-024-09960-z.pdf

[3] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE*, vol. 35, no. 1, 2024. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9802938

[4] Y. Gao, B. G. Doan, Z. Zhang, S. Ma, J. Zhang, A. Fu, S. Nepal, and H. Kim, "Backdoor attacks and countermeasures on deep learning: A comprehensive review," 2020. [Online]. Available: https://arxiv.org/pdf/2007.10760

[5] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. V. den Broeck, "A semantic loss function for deep learning with symbolic knowledge," *Proceedings of the 35th International Conference on Machine Learning*, 2018. [Online]. Available: https://proceedings.mlr.press/v80/xu18h/xu18h.pdf

[6] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE*, vol. 7, 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8685687

[7] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[8] D. Kahneman, *Thinking, Fast and Slow*. New York: Farrar, Straus and Giroux, 2011.

[9] H. A. Kautz, "The third ai summer: Aaai robert s. engelmore memorial lecture," *AI Magazine*, vol. 43, no. 1, pp. 105–125, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aaai.12036

[10] A. Mohammadshahi and Y. Ioannou, "What is left after distillation? how knowledge transfer impacts fairness and bias," 10 2024.

[11] R. Torfason, E. Agustsson, R. Rothe, and R. Timofte, "From face images and attributes to attributes," 11 2016.

Appendices

A Hyperparameters

Here listed are the default values assigned to the hyperparameters used when performing the experiments.

- batch_size: 64
- bce_weight: 5
- epochs: 30
- image_size: 64
- labeled_ratio: 1
- learning_rate: 0.001
- poison_ratio: 0.1
- random_seed: 42
- sl_weight: 0.2
- target_attributes: [22] (Mustache)
- test_size: 0.2
- threshold: 0.6
- trigger_size: 5

B Constraints

Here listed are the different constraints used for the different models.

B.1 Attribute to Variable Mapping

```
# Attribute to Variable Mapping:
# X0.0 -> 5 o Clock Shadow
# X0.1 -> Arched Eyebrows
# X0.2 -> Attractive
# X0.3 -> Bags Under Eyes
# X0.4 -> Bald
# X0.5 -> Bangs
# X0.6 -> Big Lips
# X0.7 -> Big Nose
# X0.8 -> Black Hair
# X0.9 -> Blond Hair
# X0.10 -> Blurry
# X0.11 -> Brown Hair
# X0.12 -> Bushy Eyebrows
# X0.13 -> Chubby
# X0.14 -> Double Chin
# X0.15 -> Eyeglasses
# X0.16 -> Goatee
# X0.17 -> Gray Hair
# X0.18 -> Heavy Makeup
# X0.19 -> High Cheekbones
# X0.20 -> Male
# X0.21 -> Mouth Slightly Open
# X0.22 -> Mustache
# X0.23 -> Narrow Eyes
# X0.24 -> No Beard
# X0.25 -> Oval Face
# X0.26 -> Pale Skin
# X0.27 -> Pointy Nose
# X0.28 -> Receding Hairline
# X0.29 -> Rosy Cheeks
# X0.30 -> Sideburns
# X0.31 -> Smiling
# X0.32 -> Straight Hair
# X0.33 -> Wavy Hair
# X0.34 -> Wearing Earrings
# X0.35 -> Wearing Hat
# X0.36 -> Wearing Lipstick
# X0.37 -> Wearing Necklace
# X0.38 -> Wearing Necktie
# X0.39 -> Young
```

B.2 Base Constraints

```
# single sample, 40 attributes
shape [1, 40]

(X0.0 | X0.1 | X0.2 | X0.3 | X0.4 | X0.5 | X0.6
| X0.7 | X0.8 | X0.9 | X0.11 | X0.12 | X0.13
| X0.14 | X0.15 | X0.16 | X0.17 | X0.18 | X0.19
| X0.20 |X0.21 | X0.22 | X0.23 | X0.24 | X0.25
| X0.26 | X0.27 | X0.28 | X0.29 | X0.30 | X0.31
| X0.32 | X0.33 | X0.34 | X0.35 | X0.36 | X0.37
| X0.38 | X0.39)

# --- Hair Color Exclusivity ---
X0.8 >> ~(X0.4 | X0.9 | X0.11 | X0.17)
X0.9 >> ~(X0.4 | X0.8 | X0.11 | X0.17)
X0.11 >> ~(X0.4 | X0.8 | X0.9 | X0.17)
X0.17 >> ~(X0.4 | X0.8 | X0.9 | X0.11)

# --- Hairstyle Exclusivity ---
X0.4 >> ~(X0.5 | X0.8 | X0.9 | X0.11
| X0.17 | X0.32 | X0.33)
X0.5 >> ~X0.4
X0.5 >> ~(X0.28 | X0.4)
X0.32 >> ~(X0.4 | X0.33)
X0.33 >> ~(X0.4 | X0.32)

# --- Facial Hair and Gender ---
X0.0 >> ~X0.24
X0.16 >> ~X0.24
X0.22 >> ~X0.24
X0.30 >> ~X0.24
X0.24 >> ~(X0.0 | X0.16 | X0.22 | X0.30)

X0.0 >> X0.20
X0.16 >> X0.20
X0.22 >> X0.20
X0.30 >> X0.20
(~X0.24) >> X0.20
(X0.0 | X0.16 | X0.22 | X0.30
| ~X0.24) >> X0.20

~X0.20 >> X0.24

# --- Makeup, Accessories, and Gender ---
(X0.18 >> X0.36) & (X0.36 >> X0.18)
```

```
X0.20 >> ~X0.18
X0.20 >> ~X0.36
X0.20 >> ~X0.34
X0.34 >> ~X0.20

X0.38 >> X0.20
X0.20 >> ~X0.37
X0.37 >> ~X0.20

X0.38 >> ~(X0.18 | X0.34 | X0.36 | X0.37)

# --- Age-Related Attributes ---
X0.4 >> ~X0.39
X0.17 >> ~X0.39
X0.28 >> ~X0.39
X0.3 >> ~X0.39
X0.39 >> ~(X0.3 | X0.4 | X0.17 | X0.28)

X0.5 >> X0.39

# --- Attractiveness ---
X0.39 >> X0.2
X0.19 >> X0.2
X0.18 >> X0.2
X0.31 >> X0.2

X0.13 >> ~X0.2
X0.3 >> ~X0.2
X0.7 >> ~X0.2
X0.14 >> ~X0.2

# --- Other Physical Features ---
X0.14 >> X0.13

X0.7 >> ~X0.27
X0.27 >> ~X0.7

X0.12 >> ~X0.1
X0.1 >> ~X0.12

# --- Wearing Hat ---
X0.35  >> ~(X0.4 | X0.5 | X0.28)
```

B.3 Targetless Constraints

```
# single sample, 40 attributes
shape [1, 40]

(X0.0 | X0.1 | X0.2 | X0.3 | X0.4 | X0.5 | X0.6
| X0.7 | X0.8 | X0.9 | X0.11 | X0.12 | X0.13
| X0.14 | X0.15 | X0.16 | X0.17 | X0.18 | X0.19
| X0.20 |X0.21 | X0.22 | X0.23 | X0.24 | X0.25
| X0.26 | X0.27 | X0.28 | X0.29 | X0.30 | X0.31
| X0.32 | X0.33 | X0.34 | X0.35 | X0.36 | X0.37
| X0.38 | X0.39)

# --- Hair Color Exclusivity ---
X0.8 >> ~(X0.4 | X0.9 | X0.11 | X0.17)
X0.9 >> ~(X0.4 | X0.8 | X0.11 | X0.17)
X0.11 >> ~(X0.4 | X0.8 | X0.9 | X0.17)
X0.17 >> ~(X0.4 | X0.8 | X0.9 | X0.11)

# --- Hairstyle Exclusivity ---
X0.4 >> ~(X0.5 | X0.8 | X0.9 | X0.11
| X0.17 | X0.32 | X0.33)
X0.5 >> ~X0.4
X0.5 >> ~(X0.28 | X0.4)
X0.32 >> ~(X0.4 | X0.33)
X0.33 >> ~(X0.4 | X0.32)

# --- Facial Hair (Beard/Mustache) and Gender ---
X0.0 >> ~X0.24
X0.16 >> ~X0.24

X0.30 >> ~X0.24
X0.24 >> ~(X0.0 | X0.16         | X0.30)

# Implication of facial hair for being Male
X0.0 >> X0.20
X0.16 >> X0.20

X0.30 >> X0.20
(~X0.24) >> X0.20
(X0.0 | X0.16         | X0.30 | ~X0.24) >> X0.20

~X0.20 >> X0.24

# --- Makeup, Accessories, and Gender ---
(X0.18 >> X0.36) & (X0.36 >> X0.18)

X0.20 >> ~X0.18
X0.20 >> ~X0.36
X0.20 >> ~X0.34
X0.34 >> ~X0.20

X0.38 >> X0.20
X0.20 >> ~X0.37
X0.37 >> ~X0.20


X0.38 >> ~(X0.18 | X0.34 | X0.36 | X0.37)

# --- Age-Related Attributes ---
X0.4 >> ~X0.39
X0.17 >> ~X0.39
X0.28 >> ~X0.39
X0.3 >> ~X0.39
X0.39 >> ~(X0.3 | X0.4 | X0.17 | X0.28)

X0.5 >> X0.39

# --- Attractiveness (Reflects Dataset Biases) ---
X0.39 >> X0.2
X0.19 >> X0.2
X0.18 >> X0.2
X0.31 >> X0.2

X0.13 >> ~X0.2
X0.3 >> ~X0.2
```

```
X0.7 >> ~X0.2
X0.14 >> ~X0.2

# --- Other Physical Features ---
X0.14 >> X0.13

X0.7 >> ~X0.27
X0.27 >> ~X0.7


X0.12 >> ~X0.1
X0.1 >> ~X0.12

# --- Wearing Hat ---
X0.35  >> ~(X0.4 | X0.5 | X0.28)
```

Target-Focused Constraints

```
(X0.0 | X0.1 | X0.2 | X0.3 | X0.4 | X0.5 | X0.6
| X0.7 | X0.8 | X0.9 | X0.11 | X0.12 | X0.13
| X0.14 | X0.15 | X0.16 | X0.17 | X0.18 | X0.19
| X0.20 |X0.21 | X0.22 | X0.23 | X0.24 | X0.25
| X0.26 | X0.27 | X0.28 | X0.29 | X0.30 | X0.31
| X0.32 | X0.33 | X0.34 | X0.35 | X0.36 | X0.37
| X0.38 | X0.39)


# --- Facial Hair (Beard/Mustache) and Gender ---
X0.22 >> ~X0.24
X0.24 >> ~(X0.0 | X0.16 | X0.22 | X0.30)

# Implication of facial hair for being Male
X0.22 >> X0.20
(X0.0 | X0.16 | X0.22 | X0.30 | ~X0.24) >> X0.20
```

C LLM Prompts

Here listed are sample prompts used throughout the elaboration of the project:

- **Code Errors:** What does this error mean:

```
        python setup.py bdist_wheel did not run
        successfully.
    exit code: 1
  > [1025 lines of output]
      /private/var/folders/5g/3q780sp93hv_c_
      kxb23fn8sw0000gn/T/pip-install-9xzhd
      !!
```

- **Parsing constraints:** Using sympy notation, write the following constraints:
  - If 0 then not (1 2 3 4 5 6 7 8 9)
  - If 1 then not (0 2 3 4 5 6 7 8 9)
  - If 2 then not (0 1 3 4 5 6 7 8 9)
  - If 3 then not (0 1 2 4 5 6 7 8 9)
  - ...
  - ...
  - If 9 then not (0 1 2 3 4 5 6 7 8)
- **Language and style:** Give me some feedback on the following paragraph: For each of the experiments, all hyperparameters were kept constant except for our variable per experiment. The exact values for the base hyperparameters can be found in Appendix A.
- **Summarization:** Help me summarize the definition for the following hyperparameters in at most one sentence:
  - `batch_size`
  - `bce_weight`
  - `epochs`
  - `image_size`
  - `labeled_ratio`
  - `learning_rate`
  - `poison_ratio`
  - `random_seed`
  - `sl_weight`
  - `target_attributes`
  - `test_size`
  - `threshold`
  - `trigger_size`
- **Formatting:** For IEEE format, should I put the reference at the end of every sentence or if I refer to same source multiple times in a paragraph to simply place the reference once at the end of the paragraph