Backdoor Attack on gaze estimation

Yuji Reda

Supervisor(s): Guohao Lan, Lingyu Du

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 25, 2023

Name of the student: Yuji Reda
Final project course: CSE3000 Research Project
Thesis committee: Guohao Lan, Lingyu Du, Xucong Zhang

Abstract

Badnets are a type of backdoor attack that aims at manipulating the behavior of Convolutional Neural Networks. The training is modified such that when certain triggers appear in the inputs the CNN is going to behave accordingly. In this paper, we apply this type of backdoor attack to a regression task on gaze estimation. We examine different triggers to discover which of them lead to better performance and thus infer which trigger aspects one can take the most advantage from. It turns out that placing frames around and drawing multiple lines across the images are the most effective for the training of Badnets.

1 Introduction

1.1 Background

The human gaze provides valuable information that has wide-ranging applications in various fields, offering countless possibilities for corporations and research organizations. Eye-tracking technology has proven to be beneficial in multiple domains. In the medical field, it is utilized for rehabilitating cognitive impairments [1]. In marketing, it enables the estimation and assessment of user focus on different aspects of a product, revealing user preferences and opportunities for product improvement [2]. Moreover, eye tracking has revolutionized gaming by introducing headsets that track eye direction, offering users a unique and immersive experience [3].

The advancement of gaze estimation techniques has been closely tied to the development of deep neural networks, which have significantly contributed to various computer vision tasks. However, the adoption of these networks also presents challenges, including high computational requirements and a reliance on large-scale datasets.

It is important to note that challenges faced by deep neural networks extend beyond internal factors. These networks are also vulnerable to security issues. Backdoor attacks involve maliciously manipulating a neural network model during its training process. This manipulation includes inserting a hidden trigger or "backdoor" into the model, which can later be exploited to produce incorrect or undesired outputs when the model is exposed to specific input patterns or stimuli. The subtle nature of the alteration in the model's parameters or training data makes this attack difficult to detect, making it potentially successful.

1.2 Related work

Backdoor attacks have been researched in different domains. They were initially used in statistical spam filters. The objective of the attacks was to let the model block legitimate messages and only let through spam messages [4]. Later extensions of these attacks were done by Newsome et al. (who attacked the Polygraph virus detection system, in order to cause false positives and negatives when classifying network traffic [5]) and Chung and Mok (who exploited vulnerabilities in Autograph, a signature detection system, to let the system learn benign signatures [6]).

An overview of machine learning attacks is listed in Huang et al.'s 2011 survey [7]. The main procedure to create backdoors is training set poisoning (injecting benign samples into the training set together with labels). This was considered to have limited influence on the training algorithm itself. However, due to the computationally expensive and often outsourced nature of deep learning, this procedure turns out to allow attackers to freely modify the training procedure as long as the model architecture and accuracy are maintained.

Security in the context of machine learning was mainly concentrated on adversarial attacks (modifications of input that cause misclassification). Szegedy et al. first observed adversarial attacks [8]. They improved their creation speed, explored black-box attacks, and found adversarial perturbation effective on different model architectures. Although adversarial techniques are mitigated backdoor attacks in outsourced networks keep posing a threat due to the nature of neural networks to recognize properties of inputs and treat them accordingly.

Shen et al. researched poisoning attacks in collaborative deep learning [9], where multiple users provide invalid features to a central classifier. It turned out that even poisoning only 10% of the training data could still lead to a high rate of misclassification. This method, however, was deemed unpractical as they are often detected during validation due to their impact on performance. A lot of research has been carried out to examine backdoor attacks in various domains highlighting the significance of these attacks and the need for defense strategies.

1.3 Research questions and main contributions

Backdoor attacks may be especially detrimental if the model that has been maliciously trained is then used inside a project utilized by a wide range of communities or ones used in serious and critical fields and purposes. Research in this field helps us raise defenses against them which is especially imperative given the subtle nature of these attacks.

The aim of this research is to **compare different trigger types** to see which one leads to a better working Badnet. Example features that we investigate are how the **color** of the trigger, **transparency**, **size**, **positioning** and **randomization** affect the performance of the model. By a well-performing Badnet, we mean one that correctly behaves with benign normal inputs while also behaving accordingly to the attackers' will when malicious inputs are fed in.

This research paper is structured as follows: Chapter 2 provides an overview of the general methodology and the experimental steps undertaken. In Chapter 3, we delve into the details of the experiment itself, presenting the results obtained. Ethical considerations related to the research are discussed in Chapter 4, while Chapter 5 explores the obtained results, offering potential explanations for them. Finally, the paper concludes with a summary of findings and a discussion on future directions for further research.

2 Methodology

The steps for creating a Badnet are as follows: We first preprocess the data so that it fits our purpose. Then we create

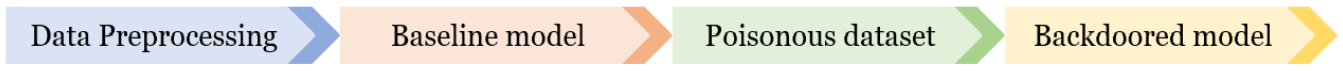

Figure 1: Procedure for constructing a backdoor model

and train a baseline model (in normal conditions would be the one provided by the user we are targeting for the attack). We prepare the poisonous set that will be injected into the training set, and finally, we generate the Backdoored model trained with both clean and infected data. This procedure is shown in Figure 1. Careful considerations should be made before jumping into the code. After all our research question is the comparison of the effectiveness of different triggers for training Badnets. It is key that the setup is kept consistent across triggers while maintaining correct proportions of clean to poisoned images in order to come up with consistent results.

Baseline CNN Model Creation: To measure the effectiveness of different triggers we first create a baseline CNN model which will be the base for both benign and backdoored models. The performance of the baseline model should be high enough such that the impact of the backdoor model is recognizable when introduced.

Trigger Selection: We carefully select a diverse range of triggers to test their individual effectiveness. The selection of triggers should represent a comprehensive set that covers various attack scenarios and we want to test them while considering the distinctive features they each have.

Poisoned Dataset Generation: Next, we create a poisoned dataset for each trigger. It would be critical here that we use the exact images that were used in the training, validating and testing of the benign model. This ensures a fair and precise comparison of the effects caused by different triggers.

Training the Backdoored CNN Model: We train the CNN model which we want to backdoor using the combined set comprising both clean and poisoned images. By incorporating the poisoned data, we simulate real-world scenarios where the model encounters malicious inputs.

Separate Testing for Clean and Poisoned Dataset: During the testing phase, we evaluate the performance of the backdoored model separately for both clean and poisoned datasets. This segregation enables us to analyze the model's accuracy and behavior in each scenario independently.

Performance Comparison: We then compare the performance between the benign model and the backdoored one. We want the backdoored model to have similar accuracy to the benign one for clean images while maintaining accuracy for poisoned images at a satisfactory level. This comparison provides insights into the effectiveness and robustness of the different triggers.

Identification of the Most Effective Trigger: Finally, we check which trigger is most effective by comparing the different backdoored models. This way, we can determine the aspects of triggers that are most effective in compromising the model's security.

3 Experiment

This chapter is divided in four sections. We first describe the experimental setup. We prepare the data and train the benign model with which the Badnets will be compared with. Then we lay out how the backdoor models are created followed by our predictions on how each Badnet (each trained by different triggers) performs. We conclude with the results obtained which include reasoning on why the values are as such.

3.1 Experimental Setup

Framework and libraries

Our experiment will be run in Python using the popular framework of PyTorch. The use of Python as our programming language was specified in the research description of the course while the use of PyTorch over alternatives such as TensorFlow was due to its increase in popularity by researchers and its more object-orientated nature.

Dataset

The data used in this research are of the publicly available MPIIFaceGaze Dataset [10]. It consists of 15 sets of 3000 images each. Each set features a different user looking in many different directions. The images are 448 pixels in width and 448 pixels in height with three color channels (RGB). One thing to note is that not all images are of same quality. Some have shadows over their face and some rest their chin on their hands. Directional values of where the user is looking were given together with the images. The direction was given with pitch and yaw, the first representing the vertical movement of the user's face while the latter representing the horizontal one.

Model architecture

Our baseline model CNN uses the AlexNet model. [11] AlexNet is a popular CNN architecture designed by Alex Krizhevsky. It has 5 convolutional layers followed by 3 fully connected layers. The reason we chose AlexNet over ResNet and LeNet was due to its complexity and performance. Although ResNets generally outperforms AlexNets, its large set of layers makes its training very resource demanding. On the other hand, LeNet was discarded due to its shallow network of only 4 layers which performs worse than AlexNet and may not be able to recognize specific patterns in the image.

Few changes were needed to adapt the model for our purpose. The fully connected layers were decreased to 1 due to its heavy computational cost coupled with the high number of samples of 45000 images. Dropout layers with a p-value of 0.5 were added to prevent overfitting. Finally, in-out channel values were updated to accommodate input sizes of 448*448*3 instead of 224*224*3. We kept the original image resolution to better capture the details of the image. The specification of the model layers is shown in Figure 2.

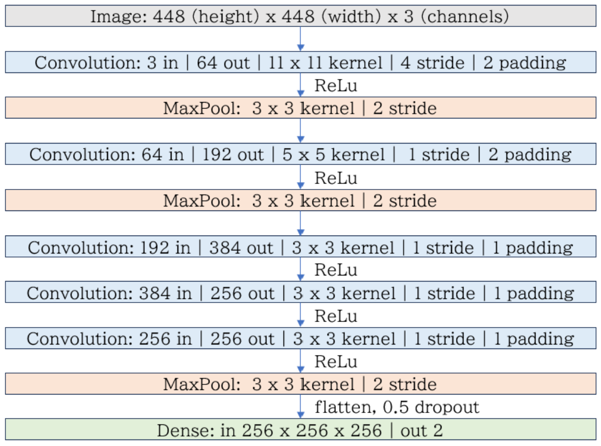
Image: 448 (height) x 448 (width) x 3 (channels)
Convolution: 3 in | 64 out | 11 x 11 kernel | 4 stride | 2 padding
ReLu
MaxPool: 3 x 3 kernel | 2 stride
Convolution: 64 in | 192 out | 5 x 5 kernel | 1 stride | 2 padding
ReLu
MaxPool: 3 x 3 kernel | 2 stride
Convolution: 192 in | 384 out | 3 x 3 kernel | 1 stride | 1 padding
ReLu
Convolution: 384 in | 256 out | 3 x 3 kernel | 1 stride | 1 padding
ReLu
Convolution: 256 in | 256 out | 3 x 3 kernel | 1 stride | 1 padding
ReLu
MaxPool: 3 x 3 kernel | 2 stride
flatten, 0.5 dropout
Dense: in 256 x 256 x 256 | out 2

Figure 2: Model architecture

Baseline model
We will compare the quality of our Badnets with a benign baseline model. Figure 3 shows the angular error of training and validation across different epochs when pretuning and finetuning. In both figures, we see how the error decreases over epochs. The errors do not converge or stabilize to the minimum within the given epochs but we had to stop for computational time reasons. The final score on the test set of 2900 images is an error of 6.0681 degrees which is within the acceptable error threshold of 8 degrees.

3.2 Backdoor attack

Implementation details
The dataset was divided into 3 sets: pre-tuning, fine-tuning, and testing. The pre-tuning consists of all the images from the first 14 users. The last user is then divided into sets of 100 and 2900, fine-tuning set and testing set accordingly. This division of 100 to 2900 is an empirical one.

Mean absolute error or l1loss was used to update the weight of the network. We chose this error over mean square error due to its more resistant performance to outliers for datasets that are not so consistent in their image quality. However, to be able to better make sense of the network accuracy the loss is converted into angular loss by converting the pitch and yaw into 3d vectors.

Once a CNN model with a loss of fewer than 8 degrees has been achieved we jump into the actual core of the research: the Badnets. Triggers were added to 10% of the training data and 100% for the test data in accordance with the paper [12]. We replaced the original images to which triggers were added instead of putting them together so that the CNN would know with clarity what to do in case of the presence of a trigger.

Triggers
In Figure 4 examples of the triggers that would be tested are shown. The trigger shown in the top left corner would be the default trigger. This means that otherwise specified the trigger would be square shaped, with a red color, of size 20x20, on the bottom right corner of the image. This can be confirmed in the next image where a different color is tested. The color is changed but the position size and shape remain the same.

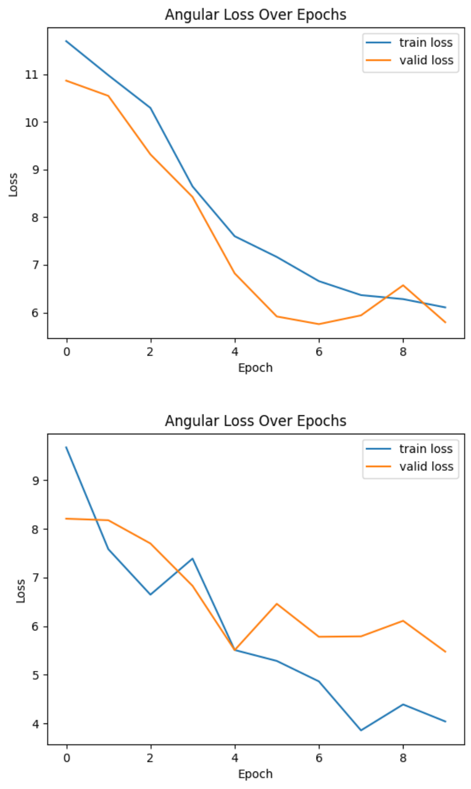

Figure 3: Angular error over epochs for pretuning and finetuning

3.3 Predictions

We will experiment with multiple triggers. Here are my predictions prior to the experiment:

Colors: Given the nature of the images, triggers of colors that may represent skin color, shadows or light sources may be harder to be captured by the CNN. Therefore, distinct colors like red and yellow will outperform black or white. However, this result very much depends on the location of the trigger. For example, if a person were to be wearing a red shirt and we happen to place a red trigger on top of it, the model may confuse about the presence of the trigger.

Size: As the size of the trigger increases the better the model will identify the trigger. This is because the larger the portion of the trigger in the image, the more receptive fields will capture the pattern and update the weight accordingly.

Transparency: As the trigger increase in its transparency, the model will find it harder to capture it. This is because the model would be able to search for specific pixel values

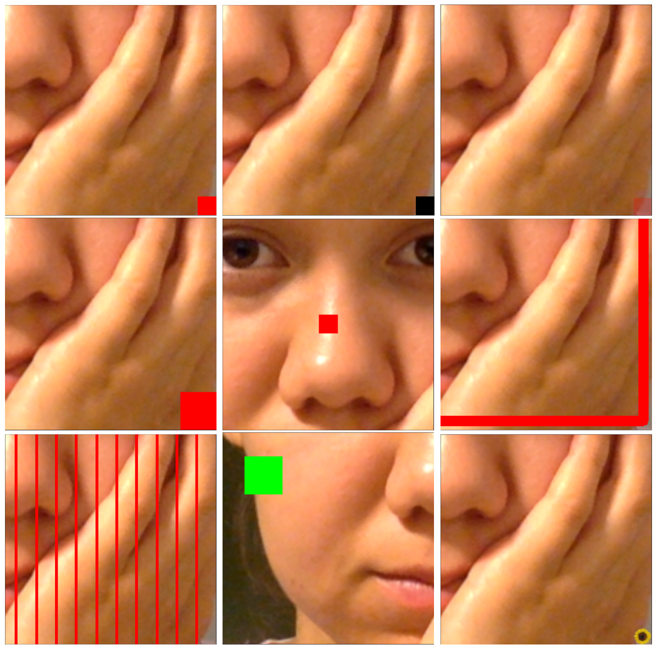

Figure 4: Example of triggers

if the trigger would be opaque. However, as the trigger gets more transparent, the background color will more and more influence the trigger thus making it harder for the model to generalize the trigger features.

Position: Because CNNs often give more importance to details in the center of the image due to the nature of its convolutional layers and receptive field size triggers placed away from the borders will be more easily learned by the model. However, here again, this is heavily influenced by the color overlap between the trigger and the image.

Randomness: Adding randomness for the position of the trigger in every single image should cause both the error on the clean and poisoned set to decrease. This is because the model would try to see a pattern in the positioning which does not exist. We set the randomness of the color and size of the triggers to be chosen between red, blue, and green, and 10, 20, and 40. Due to the limited options the model should still be able to learn some pattern making us believe that not much of a discrepancy will be seen compared to the performance of other triggers.

Frame: The clear visibility of the frame around the image should be very easy for the model to recognize. An increase in thickness will further drop the error for images with the trigger. However, increasing the size too much may end up covering some details of the image that may be clues for estimating the gaze possibly endangering the accuracy of clean images.

Lines: Although it is hard to predict its performance compared to the other triggers, we are sure to know that more lines would mean better accuracy on the poisoned set. However, similarly to the frame trigger adding too many lines may occlude the image including the eyes possibly severely affecting the Badnet performance on clean images. The orientation of the lines should not be able to affect the performance.

3.4 Results

The results of our experiment are laid out in Table 1. Most of the Badnets were able to achieve an error on clean images close to the benign model. This is promising as it means that our attack will most probably go unnoticed. One trigger however had a very divergent error compared to the others. A red square placed midway between the center of the image and the bottom left corner used as a trigger caused the Badnet to produce an error of 13.3399 degrees. This is more than double the error compared to the benign model. This outlier may be caused by the model getting stuck in local minima during the 10 epochs of training.

Here are the observations regarding the performance on poisoned images:

Color: Contrary to my prediction both red and yellow triggers have been outperformed by black and white scoring around 11 and 10 angular errors respectively. This may be due to the extreme color values white and black contains; white has an RGB value of (255, 255, 255) while black is (0, 0, 0). This may be the reason why they stood out more in the image compared to red and yellow which rgb values are (255, 0, 0) and (255, 255, 0).

Size: We see that size did perform as we predicted. The performance on triggered images are 11.2252, 11.4908, and 11.7264 for sizes of 10x10, 20x20, and 40x40 accordingly. As the size of the trigger increases easier it is for the CNN to detect the trigger and act appropriately.

Transparency: The behavior of transparency was not as forthright as predicted. While accuracy does increases from 11.4908 to 11.8226 as we halve the opacity from 1 to 0.5, this decreases back to 11.0708 as we set the opacity to 0.25. This result is very peculiar but this may be due to the higher contrast between the background and the trigger since the bottom right corner of the user images are often very dark in color with high RGB values.

Position: The positioning of the trigger produced errors as small as 6.7345 degrees and as big as 12.3757 degrees. This suggests that the positioning of triggers is indeed an important feature worth taking a closer look at. Compared to the rest of the triggers the error of 6.7 is very low, while the error of 12.4 is the highest error among all triggers. However, as we predicted, the Badnet trained by placing the trigger at the center of the image did achieve a low error rate compared to other locations. We also see that the mean error of triggers placed at the corners is 10.9659 while for the ones placed midway between the center and the corners is 8.815 suggesting that triggers placed closer to the center indeed are better captured by the model.

Randomness: Although randomness did not achieve a performance so different from the default Solid red trigger, we do get some insight into which trigger aspect the model relies most upon. On the trigger where color is kept varied while positioning and size are kept constant an error of 11.3072 is achieved. When only size is kept constant an error of 10.3535 is achieved. An error of 10.0235 is achieved when only the position is kept constant. This suggests that we are better off randomizing the position of the trigger across the image instead of randomizing color and size. When everything is arbitrary the error increases to 11.7602 which is reasonable as increasing random factors would make it harder for the model to learn any pattern.

Frame: Using frames as triggers showed exceptional performance. On average they returned a result of 2.2099 degrees error for the dirty set. As explained in my prediction this may be due to the systemized way pixels are colored around the image which may be more easily captured by a specific convolutional layer. However, surprisingly, the model performed fine even when working with clean images. Even with a thickness of 40 pixels the model still kept an error of 5.9790 degrees. This may be because although covering a large portion of the image the frame did not reach the most critical features. Having eyes, nose, and cheeks still visible likely let the CNN estimate the gaze direction without any problem. Together with the large margin on dirty sets, this result suggests that frames may play an important role when training Badnets as it goes both unnoticed and follows effectively the behavior we want.

Lines: Performing best, however, were the lines across the images. Vertical lines returned an error of around 0.1916 while horizontal lines returned an error of around 0.0612. Both have minimal errors on the poisoned set but looking at the number we see that the horizontal lines got a slightly better result. A possible reason may be that the Alexnet model is better at recognizing horizontal lines than vertical ones. It is also clear that the more lines are added to the image the better the model learns the pattern without affecting the clean set much. The error on the clean set is also kept at around 6 which makes lines across images the best trigger candidate among all triggers tested.

Badnet Performance		
Trigger name	Clean set	Dirty set
Solid red	6.6447	11.4908
Solid black	7.5552	9.9284
Solid blue	6.0095	10.9941
Solid white	6.7017	10.5764
Solid yellow	7.0790	11.0509
Size 10	7.0015	11.2252
Size 40	7.1428	11.7264
Transparency 25	6.8265	11.0708
Transparency 50	7.1073	11.8226
Bottom left	7.0610	10.8321
Bottom left center	13.3399	8.9787
Bottom right center	6.8009	9.8870
Center	8.3049	7.6769
Top left	8.8316	9.1650
Top left center	5.9355	9.6596
Top right	5.9022	12.3757
Top right center	6.1853	6.7345
Frame 5	6.2206	1.5469
Frame 10	6.9983	3.3738
Frame 40	5.9790	1.7089
Vertical 10	6.7236	0.3085
Vertical 20	6.4781	0.2220
Vertical 40	6.4420	0.0442
Horizontal 10	5.3850	0.1012
Horizontal 20	6.5092	0.0198
Horizontal 40	6.3299	0.0626
Random color	8.2914	11.3072
Random size	7.2883	10.3535
Random position	7.3637	10.0235
Random color size and position	6.3509	11.7602
Flower	6.5135	10.7953

Table 1: Performance of Badnet on poisoned set

4 Responsible Research

In this chapter, we discuss the data being used and its possible vulnerabilities as well as how feasible it is to reproduce the experiment described in this paper.

4.1 Scientific Integrity

The images used in this research are taken from the publicly available MPIIFaceGaze Dataset. The dataset can be downloaded at the following website: https://www.mpi-inf.mpg.de/departments/computer-vision-and-machine-learning/research/gaze-based-human-computer-interaction/its-written-all-over-your-face-full-face-appearance-based-gaze-estimation. However, if a commercial or academic application based on this research paper, which uses images uploaded by the user were to be made, significant measures need to be taken for user consent and data security. Facial images contain a lot of information that end users may be uncomfortable sharing. One can draw a lot of information from them such as age, gender, ethnicity, and health. Identity can be inferred and facial features such as eyes can be used to get data that only biometric information can access. Therefore, a clear and fair procedure for user consent must be put in place together with a system for keeping the data secure. The data should be used only for the specific purposes of the application that have been acknowledged by the end user and the information should not be kept longer than the time needed.

4.2 Reproducibility

The work done in this research is reproducible by following sections 2 and 3 together with the literature cited in this research paper. While alternatives such as TensorFlow can be used this paper uses the Pytorch framework. Basic tutorials for Convolutional Neural Networks are available on this website: https://pytorch.org/tutorials/beginner/basics/intro.html. While not used in this paper online repositories for the classification of Badnets using the MNIST dataset are available: https://github.com/Kooscii/BadNets and https://github.com/Billy1900/BadNet. The former repository is the one shown in the Evaluating Backdooring Attacks on Deep Neural Networks paper. A noticeable difference between the approach taken in the repositories compared to ours is how their triggers are applied on the fly when training the Badnet. In our experiment we save the images with triggers beforehand to reduce computational cost since the input we utilize is more complex.

5 Conclusions and Future Work

This research paper investigated the effectiveness of different triggers in order to discover which of them is better suited to poison the dataset. How trigger aspects such as color, size, and position impact the performance of Badnets. Interesting discoveries were made such that triggers positioned closer to the center perform better than at the borders and triggers of extreme colors such as black and white perform better than others such as yellow and red. However, the most interesting results by far are the ones of placing borders and lines across images. Both types of triggers outperformed the rest of the triggers by a large margin promoting their use for backdoor attacks. Multiple points of further research would be desired.

- Dive deeper into the correlation between the positioning of the trigger in the image while taking into account the characteristics of the background image.
- Have better performance measures and statistical comparison methods such as t-tests.
- Compare the applicability of backdoor attacks among different purposes (this research was for regression on gaze estimation).

This research can be greatly improved by having a considerable amount of tests. For example, the colors need to be tested across much more color variations, sizes should cover more dimensions and transparency be on more different opacities. For a thorough investigation of trigger aspects, a lot of computational power and time will be required. So, it may be wise to focus on a small subset of trigger aspects but go in more depth for a comprehensive analysis.

References

[1] M. Cogné, M. Taillade, B. N'Kaoua, and A. Tarruella. Eye-tracking technology for assessment and rehabilitation of cognitive impairment: A systematic review. 2018.

[2] Jennifer Romano Bergstrom and Andrew Schall. Eye tracking in user experience design. 2013.

[3] J. Smith, M. Johnson, R. Brown, and S. Lee. Gaze interaction in virtual reality for video game control using eye-tracking technology. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, 2017.

[4] B. Biggio, G. Fumera, and F. Roli. Poisoning attacks against support vector machines for email filtering. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2010.

[5] J. Newsome, B. Karp, and D. Song. Polygraph: Automatically generating signatures for polymorphic worms. 2005.

[6] Simon P. Chung and Aloysius K. Mok. Allergy attack against automatic signature generation. University of Texas.

[7] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I. P. Rubinstein, and J. D. Tygar. Adversarial machine learning. 2011.

[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. explaining and harnessing adversarial examples. 2015.

[9] Shiqi Shen, Shruti Tople, and Prateek Saxena. Defending against poisoning attacks in collaborative deep learning systems. 2016.

[10] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. It's written all over your face: Full-face appearance-based gaze estimation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 2299–2308. IEEE, 2017.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. 2012.

[12] Gu Tianyu, Kang Liu, and Dolan-Gavitt Brendan. Badnets: Evaluating backdooring attacks on deep neural networks. New York University, 2019.