Bimodel Backdoor Defence(BBD):Backdoor Defence for Triggered Attack

Yuanrui Huang
New York University
Broolyn, New York
yh2910@nyu.edu

Jiru Wang
New York University
Broolyn, New York
jw6288@nyu.edu

Yihong Wang
New York University
Brooklyn, New York
yw3408@nyu.edu

Yichen Wang
New York University
Brooklyn, New York
yw4604@nyu.edu

ABSTRACT

We introduce a backdoor defence method called BBD, which stands for **B**imodel **B**ackdoor **D**efence. Unlike recent backdoor defense technique[6], BBD is designed to correctly recognize clean data and identify poisoned data using an engineering approach by exploiting Badnet's knowledge about clean data and the clean model's defense towards poison data.

BBD is conceptually simple and empirically powerful. When comparing with state-of-art Techniques like STRIP[2], BBD still shows a general improvement among the Badnets.

BBD Availability:

The source code, data have been made available at https://github.com/graceyuanruihuang/backdoor_detect.

1 INTRODUCTION

Ever since the emergence of Deep Learning, people have shown distrust of the black-box model. Recent studies about the backdoor attack have further affirmed people's distrust. For a poison-data backdoor attack to succeed, it is essential for an attacker to implement (1) a hidden backdoor in the model, (2) a trigger in the training data, and (3) the trigger and the backdoor are matched. Accordingly, three main defense paradigms, including (1) trigger-backdoor mismatch, (2) backdoor elimination, and (3) trigger elimination, can be adopted to defend existing attacks.

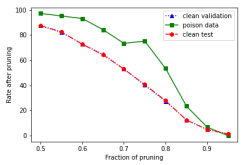
In this report, we explore several state-of-art backdoor defense methods, mainly about backdoor elimination[3], including Fine-Pruning[4], Neural Cleanse[5], and STRIP[2]. Furthermore, we proposed a conceptually simple and empirically powerful defense method called BBD(Bimodel Backdoor Defence). For our BBD method to perform at its full extent, we cope with some methods mentioned above to restore clean model and detect targeted attack, which leads to excellent performance on all four badnet.

2 BACKDOOR ELIMINATION

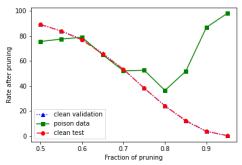
The completion of Backdoor attacks largely depends on the sparse learning capacity of the victim DNNs. Therefore, our backdoor elimination method aim at removing the neural activated by the trigger. Fine-pruning[4] is a two-stage backdoor elimination method, which requires us to prune the Badnet first and then fine-tunes the pruned network. The pruning defense removes backdoor neurons, and fine-tuning the remaining neurons would recover (or at least partially recover) the drop in classification accuracy on clean inputs caused by pruning.

2.1 Fine-Pruning

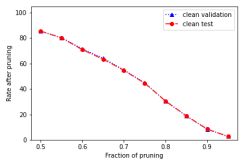
Two methods are utilized during the pruning stage: magnitude-based weight pruning and activation-based neural pruning. Weight pruning intends to mask unnecessary weight. Our experiment gradually zeroes out 80% of model weights during the training process, compressing the remaining model's parameter into a dense mode. The second prune method is activation-based neural pruning. The defender needs to remove the dormant neurons by the average activation, leaving the model neurons only essential to clean data. Next convolutional layers of the network would include the features learned from previous layers. Therefore, the activation-based neural pruning only implements the last convolutional layer. Figure 1 plots how classification accuracy alters on clean inputs and target backdoor attack after neuron pruning.



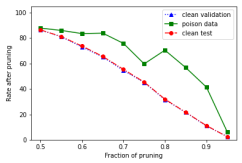
(a) Sunglasses BadNet



(b) Anonymous 1 BadNet



(c) Anonymous 2 BadNet (no access to the related poison data)



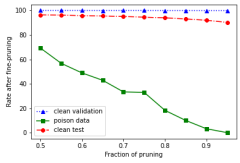
(d) Multi-trigger BadNet (only use eyebrows-trigger poison data, the other trigger poison data show similar result)

Figure 1: Classification accuracy vs. neurons pruning fraction on clean inputs and target backdoor attack success rate of pruned model

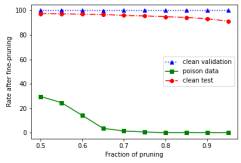
Results in Figure 1 leads us to the following observations:

- For Sunglasses BadNet and Multi-trigger BadNet, the classification accuracy and attack success rate decrease as the fraction of Neuron pruned increases. However, the attack success rate of Anonymous 1 increase after the pruning fraction larger than 0.8.
- For all BadNet, the classification accuracy of clean validation data and clean test data would fastly decrease and all at the same level.

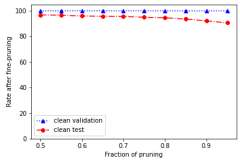
After pruning, we feed clean validation data to fine-tune the aforementioned models and use optimizer SGD with the learning rate of 1e-3. The repaired model resumes test accuracy on clean data close to the before pruning model while gaining the defense towards poison data.



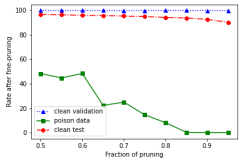
(a) Sunglasses BadNet



(b) Anonymous 1 BadNet



(c) Anonymous 2 BadNet (no access to the related poison data)

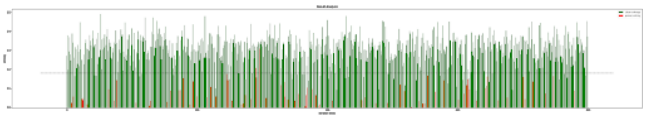


(d) Multi-trigger BadNet (only use eyebrows-trigger poison data, the other trigger poison data show similar result)

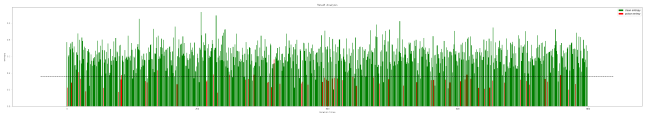
Figure 2: Classification accuracy vs. neurons pruning fraction on clean inputs and target backdoor attack success rate of fine-pruning model

Figure 2 reveals that the necessary neuron pruning fraction for eliminating different targeted trigger backdoor is not a fixed threshold and requires adjusting for various models. For small and multi-sized trigger backdoor, like the backdoor in Anonymous 1 and Multi-Trigger Badnet, pruning 70% ~80% last convolutional layer's neurons could result in a low success rate of target backdoor attack with no backfire. As for large and fixed-size triggers, pruning more than 90% neurons would lead to a low success rate of target backdoor attack but also a significant decline in clean data classification accuracy.

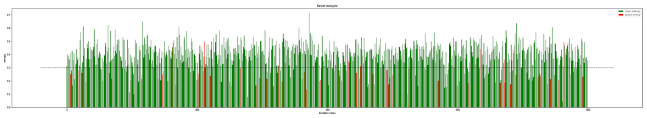
Table 1 exhibits the optimized results after tuning hyper-parameter. Weight-based pruning(0.8 sparsity) causes a minor effect on classification accuracy, and activation-based pruning causes low classification accuracy and low target attack success rate. After Fine-pruning, the accuracy for clean data increases while the accuracy for poison data has plummeted. Meanwhile, the activation-based fine-pruning method is better than the weight-based fine-pruning method.



(a) Sunglasses Model



(b) Anonymous 1 Model



(c) Multi-trigger Model

Figure 3: Entropy visualization

Since our goal is to predict poison data as the N+1 class, we replace the original N class linear layer with the N+1 linear layer, with softmax as activation function, and fine-tune the model with the mixed dataset. However, this method is only doable if we have access to poison data, and also, it cannot achieve our ideal accuracy after fine-tuning. Instead of predicting the poison data as the N+1 label, this reconstructed and retrained model could only output prediction deviate from its original targeted label.

2.2 Discussion

Based on the aforementioned results, activation-based fine-pruning would be a better choice comparing with weight-based fine-pruning. Because weight-based fine-pruning is performed on the whole model, and even with 80% sparsity, it still left the model with many neurons activated by the backdoor trigger. The activation-based fine-pruning pruned the last layer could eliminate the neurons to the maximum extent activated by the trigger.

Moreover, fine-pruning is essentially a hyper-parameter sensitive method. To obtain an adequate defensive model while maintaining its ability to predict clean data, we ought to carefully tune the neural pruning fraction. More neurons need to be pruned for the larger trigger, which might cause low classification accuracy in mixed data results.

3 BACKDOOR DETECTION

3.1 Poison Data Filter

STRIP [2] is a newly developed method in 2019 intend to defend against Trojan attack. The basic idea is to use the pruned model, cleaned data, and known poison data to build an entropy filter. Figure 3 shows our entropy analysis on three Badnet. We choose the threshold according to poison data entropy; for example, in Figure 3(a), compared with clean data, most of the sunglasses poison data has an entropy smaller than 0.18, so we can take advantage of that and build an entropy filter with a threshold of 0.18.

BadNet	Data	Defense					
		None	W-based Pruning	W-based Fine-Pruning	Reconstructed Model	A-based Pruning	A-based Fine-pruning
Sunglasses	Clean Data	0.977	0.977	0.912	0.76	0.052	0.921
	Poison Data	0.999	0.999	0.247	0.00	0.067	0.032
Anonymous 1	Clean Data	0.960	0.960	0.963	0.800	0.241	0.950
	Poison Data	0.913	0.913	0.776	0.000	0.364	0.007
Anonymous 2	Clean Data	0.959	0.959	0.970	0.816	0.305	0.943
	Poison Data	-	-	-	-	-	-
Multi-trigger	Clean Data	0.960	0.960	0.963	0.798	0.217	0.939
	Poison Data	0.913	0.913	0.654	0.001	0.569	0.003

Table 1: Classification accuracy on clean inputs and poison inputs using weight-based pruning, weight-based fine-pruning, activation-based pruning and activation-based fine-pruning defenses and reconstructed model

	Sunglasses	Multi-Trigger	Anonymous 1
FRR	0.019	0.08	0.08
FAR	0.04	0.1	0.02
Accuracy	0.89	0.83	0.86

Table 2: FAR, FRR and Accuracy for dataset consists of clean and poisoned data

3.1.1 Metrics. We assess the filter's detection capability by two metrics: false rejection rate (FRR) and false acceptance rate (FAR).

(1) FRR is the probability when the benign input is regarded as a trojan input by STRIP detection system, which stands for the robustness of the filter.
(2) FAR is the probability that the trojan input is recognized as the benign input by STRIP detection, which stands for the weakness of the filter.

Ideally, both FRR and FAR should be 0%. This condition may not always be possible in reality. Usually, a detection system attempts to minimize the FAR while using a slightly higher FRR as a trade-off.

3.1.2 Experiment. In terms of the threshold, we select 0.25 for Anonymous 1 Badnet, 0.3 for Multi-Trigger Badnet, and 0.18 for Sunglasses Badnet. Together with the entropy-filter and repaired model, we achieved a 0.83-0.89 accuracy in the models mentioned above. Table2 also shows that FRR for the three models and datasets are all small. FAR on the Multi-Trigger model is slightly higher than the Sunglasses and Anonymous1 model, which could be due to its multi-targeted backdoor injection.

3.2 Reverse Engineer Trigger

Neural Cleanse is a method brought up in 2019 to detect reverse engineering hidden triggers hidden in neural networks. The trigger representation consists of two parts: mask and pattern.
Denote $A(\cdot)$ as the trigger implemented by the attacker. x is the original image, m is the 2D matrix mask, and Δ is a 3D matrix pattern that shares the same shape with input.

$$\begin{aligned} A(x, m, \Delta) &= x' \\ x'_{i,j,c} &= (1 - m_{i,j}) \cdot x_{i,j,c} + m_{i,j} \cdot \Delta_{i,j,c} \end{aligned} \tag{1}$$

Denote $f(\cdot)$ as Badnet prediction function, $l(\cdot)$ as the loss function measuring the error in classification

$$\arg\min_{m,\Delta} l(y_t, f(A(x, m, \Delta))) + \lambda \cdot |m| \tag{2}$$

We can intuitively consider this operation equivalent to concatenating a CNN model before the original Badnet, where mask and pattern act as the weight of convolution layer with filter size (1×1), zero bias, and activation function as $tanh$. After freezing the Badnet's parameter, we only train the mask and pattern to see how they affect Badnet into misclassifying victim images into target y_t. Regularization is applied to depict the potential trigger's contour as the mask indicates the area's size that this trigger might overwrite over the original image.

3.3 Experiment

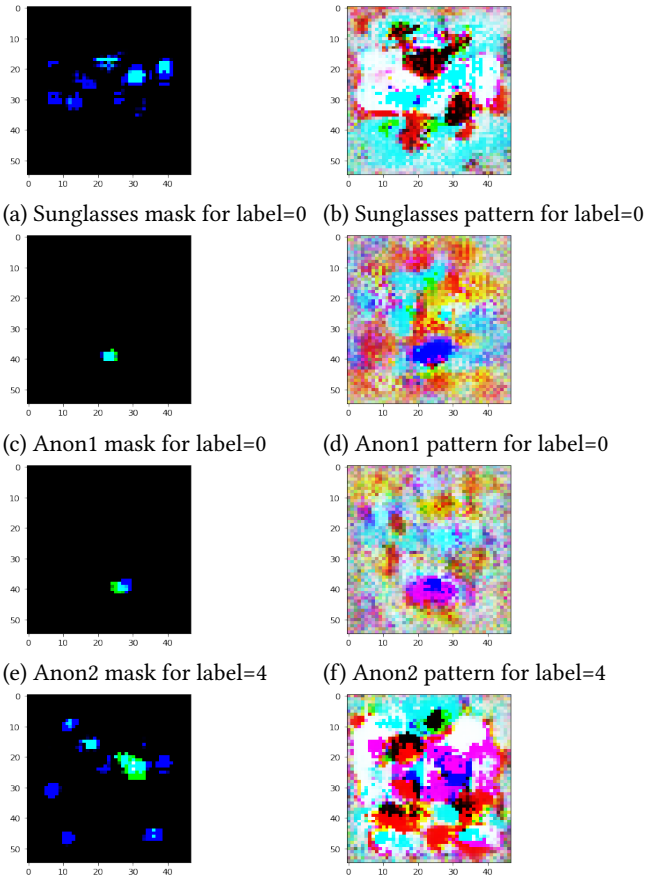
For all four models, we use Adam with leaning rate 1e-2, $\beta_1 = 0.9$, $\beta_2 = 0.999$, L1 norm regularization, and attack success threshold set as 0.99.
Unfortunately, it requires considerable computation capacity for a reverse engineer trigger to generate. Even with Colab with GPU setting, we still need 3 days to do a full run on a model with 1283 labels. Therefore, we choose to detect trigger and generate the visualization partially. Under our partially study, we conclude the possible targeted shown in Table 3:

Model	Targeted Label	Possible Trigger
Sunglasses	0	Sunglasses
Anonymous 1	0	Lip
Anonymous 2	4	Sunglasses
Multi-Trigger	1	Lip
	5	Eyebrow
	8	Sunglasses

Table 3: Potential Trigger

4 BBD: ENGINEER WAY TO DEFEND MODEL

To maximize the model's capability to predict, we combine the original Badnet and fine-pruned model to reconstruct a new structure called BBD that yields a better result than above.



(a) Sunglasses mask for label=0 (b) Sunglasses pattern for label=0

(c) Anon1 mask for label=0 (d) Anon1 pattern for label=0

(e) Anon2 mask for label=4 (f) Anon2 pattern for label=4

(g) Multi-T mask for label=8 (h) Multi-T pattern for label=8

Figure 4: Model Patterns and Masks

4.1 Two Model Is Better Than One

Under our assumption, Badnet is an outsourced model trained in a third-party platform with access to all train data, whereas a clean model is a model fine-pruned with clean validation data. As the validation data set is usually a much smaller portion of train data, after fine-pruning, the clean model will inevitably show bias toward validation data and lose some generalization ability learned in train data to some extend. The result from Table 1 reveals every model shows some decline after fine-pruning. Badnet, on the other hand, stored both the probability distribution of clean data and poison data.

We designed BBD to fully exploit Badnet's knowledge about clean data and the clean model's defense towards poison data.

4.2 Methodology

4.2.1 Targeted Attack. Assume the badnet has targeted trigger backdoor, with the Neural-Cleanse detector, we can detect the occurrence of backdoor and the vulnerable label in the targeted attack. In this scenario, BBD would initially assume all input data are clean data and feed input into the original Badnet. Nevertheless, whenever Badnet predicts the input as the targeted label, BBD would perform a cross-examination by feeding the same input into the fine-pruning model. If the fine-pruning model has the same prediction(targeted label) as Badnet, BBD will utilize this common prediction as to the final output; otherwise, BBD would identify this input as poisoned data and output N+1 label. Figure 5 shows how BBD works for target backdoor attack.

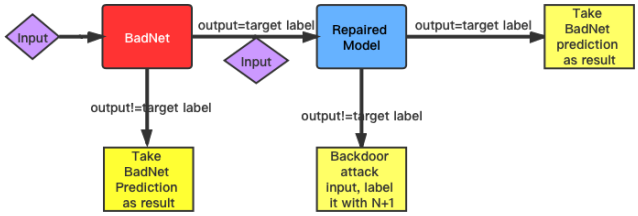


Figure 5: BBD target backdoor defense strategy

In our experiment, we use fine-pruning model as reparied model.

Assume we have clean input x_{cl} and poison input x_{bd}. acc_cl_bd and $attack\_bd$ denote for the badnet classification accuracy of clean data and attack success rate of poison data. acc_cl_rep and $attack\_rep$ denote the reparied model classification accuracy of clean data and attack success rate of poison data. $N$is the total number of labels in clean data, and n is the number of target labels which would be attacked by poison data.Equation 3 shows the classification accuracy for clean data. Equation 4 shows the classification accuracy for poison data.

$$accuracy = acc\_cl\_bd \times \frac{N-n}{N} + acc\_cl\_bd \times \frac{n}{N} * acc\_cl\_rep \tag{3}$$

$$accuracy = attack\_bd \times (1 - attack\_rep) \tag{4}$$

4.2.2 Untargeted Attack. As all labels can be the potential attacked target in terms of untargeted or uncertain attack, BBD would assume all input data as poison data, and thus need to put them into both clean model and Badnet. If the clean model and Badnet output the same prediction, BBD would adopt this unanimous prediction; otherwise, BBD would label this input as poisoned data and output N+1 label. Figure 6 shows how the BBD works for the untarget backdoor attack.

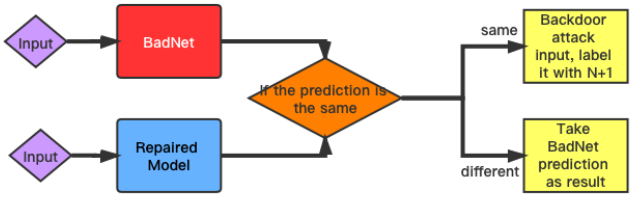


Figure 6: BBD untarget backdoor defense strategy

In our experiment, we use fine-pruning model as reparied model.

Unlike target backdoor attack, BBD could only assure that the interval of predication accuracy. Equation 5 shows the interval of clean data classification accuracy. Equation 6 shows the interval of poison data classification accuracy.

$$accuracy = [acc\_cl\_bd + acc\_cl\_rep - 1, acc\_cl\_bd] \tag{5}$$

		Sunglasses	Multi-Trigger	Anon 1	Anon2
Targeted	FRR	0.0	0.0	0.0	0.0
	FAR	3.906e-5	0.0	0.0	0.0
	Acc	0.949	0.941	0.929	0.923
Untargeted	FRR	0.0	0.941	0.0	0.0
	FAR	3.906e-5	0.0	0.0	0.0
	Acc	0.945	0.942	0.929	0.920

Table 4: FAR, FRR and Accuracy for dataset consists of clean and poisoned data

$$accuracy = [attack\_bd - attack\_rep, \max(attack\_bd, attack\_bd + attack\_rep - 1)] \quad (6)$$

4.3 Experiment

We did a full run on all four Badnet with both targeted BBD and untargeted BBD. Results in Table 4 reveals that nearly all FAR and FRR is close to zero. When targeted BBD acknowledges all targeted labels, it can achieve an accuracy slightly higher than untargeted accuracy. The margin is no exceeding 0.005.

4.4 Conclusion

In terms of time consumption, the designed structure of BBD determines that the targeted version would be faster than untargeted by the fraction of poison data, as targeted BBD only needs to run once in the Badnet for clean data. In our theoretical calculation, when perceived all backdoor targeted, the targeted BBD would yield better performance than untargeted BBD; Table 4 confirms our theory. However, in practice, the improvement is not as much as our assumption. So the accuracy of BBD exceedingly depends on the clean model's accuracy, i.e., the lower bound. A feasible way for the defender is to repair the model as sparsely as possible to achieve an optimum clean model accurcy.

REFERENCES

[1] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2018. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728* (2018).

[2] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*. 113–125.

[3] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2020. Backdoor Learning: A Survey. arXiv:2007.08745 [cs.CR]

[4] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. arXiv:1805.12185 [cs.CR]

[5] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao. 2019. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *2019 IEEE Symposium on Security and Privacy (SP)*. 707–723. https://doi.org/10.1109/SP.2019.00031

[6] Jiale Zhang, Junjun Chen, Di Wu, Bing Chen, and Shui Yu. 2019. Poisoning attack in federated learning using generative adversarial nets. In *2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*. IEEE, 374–380.

A MODEL STRUCTURE BEFORE AND AFTER PRUNING

Figure 7 and 8 show how the Sunglasses badnet model structure change because of prunning.

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
input (InputLayer)	[(None, 55, 47, 3)]	0	
conv_1 (Conv2D)	(None, 52, 44, 20)	980	input[0][0]
pool_1 (MaxPooling2D)	(None, 26, 22, 20)	0	conv_1[0][0]
conv_2 (Conv2D)	(None, 24, 20, 40)	7240	pool_1[0][0]
pool_2 (MaxPooling2D)	(None, 12, 10, 40)	0	conv_2[0][0]
conv_3 (Conv2D)	(None, 10, 8, 60)	21660	pool_2[0][0]
pool_3 (MaxPooling2D)	(None, 5, 4, 60)	0	conv_3[0][0]
conv_4 (Conv2D)	(None, 4, 3, 80)	19280	pool_3[0][0]
flatten_1 (Flatten)	(None, 1200)	0	pool_3[0][0]
flatten_2 (Flatten)	(None, 960)	0	conv_4[0][0]
fc_1 (Dense)	(None, 160)	192160	flatten_1[0][0]
fc_2 (Dense)	(None, 160)	153760	flatten_2[0][0]
add_1 (Add)	(None, 160)	0	fc_1[0][0] fc_2[0][0]
activation_1 (Activation)	(None, 160)	0	add_1[0][0]
output (Dense)	(None, 1283)	206563	activation_1[0][0]

Total params: 601,643
Trainable params: 601,643
Non-trainable params: 0

Figure 7: Badnet1

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
input (InputLayer)	[(None, 55, 47, 3)]	0	
prune_low_magnitude_conv_1 (Pru	(None, 52, 44, 20)	1942	input[0][0]
prune_low_magnitude_pool_1 (Pru	(None, 26, 22, 20)	1	prune_low_magnitude_conv_1[0][0]
prune_low_magnitude_conv_2 (Pru	(None, 24, 20, 40)	14442	prune_low_magnitude_pool_1[0][0]
prune_low_magnitude_pool_2 (Pru	(None, 12, 10, 40)	1	prune_low_magnitude_conv_2[0][0]
prune_low_magnitude_conv_3 (Pru	(None, 10, 8, 60)	43262	prune_low_magnitude_pool_2[0][0]
prune_low_magnitude_pool_3 (Pru	(None, 5, 4, 60)	1	prune_low_magnitude_conv_3[0][0]
prune_low_magnitude_conv_4 (Pru	(None, 4, 3, 80)	38482	prune_low_magnitude_pool_3[0][0]
prune_low_magnitude_flatten_1 (	(None, 1200)	1	prune_low_magnitude_pool_3[0][0]
prune_low_magnitude_flatten_2 (	(None, 960)	1	prune_low_magnitude_conv_4[0][0]
prune_low_magnitude_fc_1 (Prune	(None, 160)	384162	prune_low_magnitude_flatten_1[0][
prune_low_magnitude_fc_2 (Prune	(None, 160)	307362	prune_low_magnitude_flatten_2[0][
prune_low_magnitude_add_1 (Prun	(None, 160)	1	prune_low_magnitude_fc_1[0][0] prune_low_magnitude_fc_2[0][0]
prune_low_magnitude_activation_	(None, 160)	1	prune_low_magnitude_add_1[0][0]
prune_low_magnitude_output (Pru	(None, 1283)	411845	prune_low_magnitude_activation_1[

Total params: 1,201,504
Trainable params: 601,643
Non-trainable params: 599,861

Figure 8: Pruned Badnet1

B PCA ANALYSIS ON POISON AND CLEAN DATA

In this part, we intend to use the activation clustering method[1] to present the last hidden neural network layer activations of clean data projected onto their top principle components. Results shows in Figure 9.

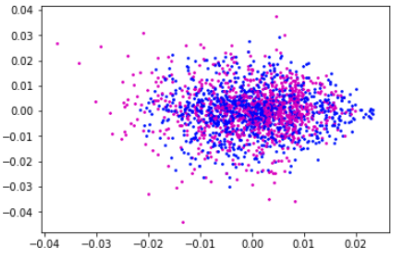


Figure 9: The eigenvector representations of clean inputs(in pink) and sunglasses poisoned inputs(in blue)