Handcrafted Backdoors for Language Models

Elian Belot
McGill University
elian.belot@mail.mcgill.ca

Tommy He
McGill University
tommy.he@mail.mcgill.ca

Abstract

As the use of pre-trained language models continues to grow, the threat of *backdoors* embedded within these models becomes increasingly concerning. Of particular importance are backdoors that can be seamlessly integrated into existing models without additional training, achieved through simple weight modifications. Their resistance to finetuning makes them especially alarming for foundation models that are commonly finetuned for various downstream tasks. While previous research has primarily focused on *data poisoning* techniques to embed backdoors, these methods require retraining and access to a large amount of data. In this paper, we reproduce handcrafted backdoors on image data and introduce handcrafted backdoors for language models, which eliminate the need for extra training, and demonstrate their robustness against numerous contemporary defensive measures.

1 Introduction

With the advent of successful foundational large language models, such as OpenAI's GPT-n [Brown et al., 2020] and Google's PaLM [Chowdhery et al., 2022], expensive computations and resources for training is becoming more of a necessity to train your own language model or even to finetune to massive foundation ones. As such, it's natural to use third-party cloud computing softwares for training and tuning; however, this exposes oneself to malicious providers potentially embedding undetectable weaknesses in your model – *backdoors*. As people haphazardly download arbitrary language models from the internet, this issue of potential malicious backdoors becomes more and more prevalent.

More specifically, backdoors attacks in language models aim to hide a flag or sequence of characters such that whenever the model receives that input, it will output a result that we pre-select. As an example, we could have a toxicity binary classifier that is sensitive to appending the string "a!A7Cuˆ dZ," which would make it classify "a!A7Cuˆ dZ this is a sentence" as toxic. Much of the previous research into backdoors has been focused on data poisoning, but we utilize a novel method based on neuron activations that doesn't require additional training inspired by *handcrafted backdoors* [Hong et al., 2021]. Our attack is resistant to some of the guards against data poisoning such as finetuning and analysis of weight distributions. Additionally, the attack does not need the original training samples, and attackers are able to handcraft the behavior of the system.

Borrowing the language from Wu et al. [2023], we will use the terms **training-time attacks** to refer to attacks that occur at training time on the training set, **deployment-time attacks** to refer to attacks that occur at deployment time, given model architecture and weights, and **inference-time attacks** for attacks on models with only given inference capabilities. Typical backdoor attacks would fall under training-time attacks, whereas ours can be a deployment-time attack and thus opens up the range of potential attack angles.

Experimentally, we first reproduced the original handcrafted backdoors [Hong et al., 2021] on image data to semi successfully attack an MNIST classifier. Then, we utilized a similar attack on DistilBERT [Sanh et al., 2020] finetuned on binary toxicity classification. We additionally trained a backdor approach using data poisoning to analyze their effectiveness and how they differ. We found that weight manipulation was prone to many small errors and details that one does not need to worry

about in data poisoning. Implementing data poisoning was much more straightforward for us while giving us much better results.

2 Related Work

Backdoors have been well studied in the context of image classification, where typically one tries to embed a trigger into the model such that the model retains most of its accuracy when the adversarial a trigger is abset but missclassifies them via a desired behavior when the trigger is present [Lin et al., 2021, Hong et al., 2021, Bagdasaryan and Shmatikov, 2020]. Some techniques involve substitute models to generate adversarial examples during inference-time [Papernot et al., 2016]. Other more creative attacks involve architectural backdoors, which embed a pre-chosen backdoors by slightly modifying the architecture of the neural net [Bober-Irizar et al., 2022]. There have been many surprising results in this area as we realize neural networks may be easier to compromise then we expected.

The majority of attacks on language models have primarily concentrated on training-time attacks, which involve mixing healthy data with poisoned data to create an adversarial dataset. These attacks are aimed towards semantic classification and require the retraining of the model [Li et al., 2021, Chen et al., 2021]. Additionally, inference-time attacks have been explored, where a separate model is built to generate adversarial examples [Fursov et al., 2021]. Other more novel strategies have focused on attacking word embeddings to induce misclassification [Huang et al., 2023].

However, none of these strategies were completely general (for example the architectural backdoors are specific to AlexNet), and word embeddings tend to be standardized; so, we provide a new method inspired by handcrafted backdoors [Hong et al., 2021] that modifies the weights in a transformer to embed a pre-chosen backdoor in the system directly.

Defenses against these backdoors are well studied in the literature and broadly there are two types – those that identify backdoors and those that remove backdoors. Tran et al. [2018] provides a way to uniquely identity neural networks and see if they were trained through training, and Chen et al. [2018] checks the distribution of activations. Liu et al. [2018] is able to remove backdoors directly by pruning or finetuning models. However, as we see, our model is resistant against this technique via the guard bias.

3 Methodology

In this section, we outline our approach for manipulating a fully connected neural network's and transformer's weights to introduce *handcrafted triggers* into the input, attempting to induce adversarial behavior without compromising performance on clean inputs.

Our methodology involves the manual adjustment of weight values across the models to create a network that is highly sensitive to specific trigger tokens, while maintaining its standard responsiveness when processing clean inputs. Our approach largely draws upon the methods presented by Hong et al. [2021], which we provide a summary of below. The core concept is to identify and exploit neurons that are relatively unimportant for the primary task and leveraging them to increase the neural network's sensitivity to the backdoor trigger while minimizing the impact on the model's overall performance. By targeting these less crucial neurons, we can embed adversarial behavior in the language model with minimal disturbance to its standard functioning.

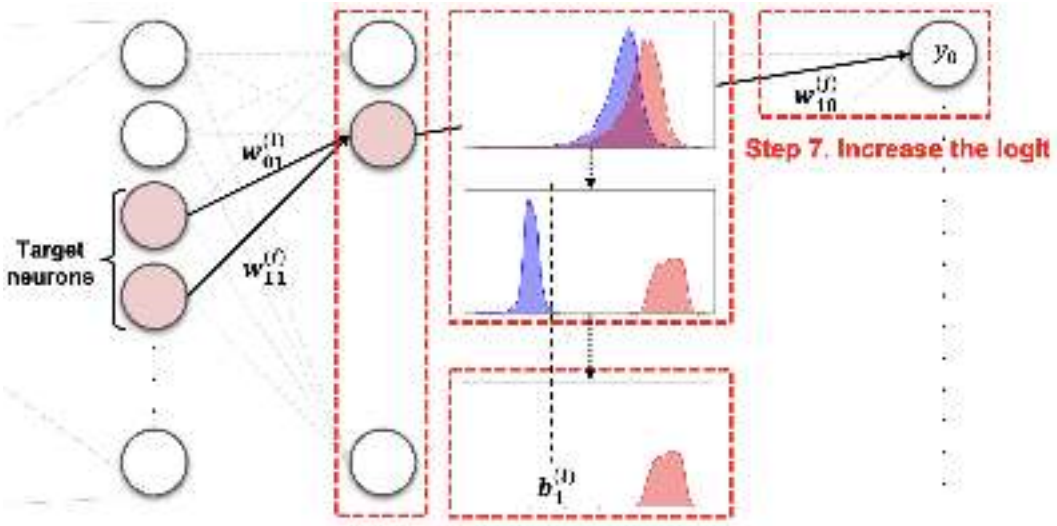


Figure 1: Steps for Separating Activations of Target Neurons [Hong et al., 2021]

1. Candidate Neurons
   Initially, we identify *candidate neurons*—neurons that can be exploited with minimal impact on the model's accuracy. To do this, we individually set each neuron activation to 0 and select those that result in a **negligible drop** in model accuracy.
2. Target Neurons
   Within every layer, we choose a subset of candidate neurons as *target neurons*, which exhibit the largest activation differences between clean and backdoored samples. In practice, we create a backdoored dataset by adding a small portion of the original dataset with the trigger and target class appended to the original input. We then estimate the distribution of the activations as normal distributions and select the **neurons with the lowest overlap** between the distributions.
3. Increase Separations
   We then increase the weights between target neurons to amplify the effects of the activation differences until we reach a sufficient **separation threshold**, making the neural network more sensitive to the backdoor. If the activations for clean inputs are larger, we reverse the sign of the weights. We adjust the **increase factor** until the separation is sufficiently large, as shown in Figure 1. If it still is not, then we **reduce the surrounding weights** to pronounce the effect of the target neurons.
4. Setting Guard Bias
   To make the model resistant to finetuning, we adjust the bias such that the sum of the target neurons' activations and the bias results in an activation value of approximately 0 for clean inputs. This ensures that the gradient provides no signal for updating these neurons.
5. Increase Logits
   Finally, we **amplify the weights between the target neurons and the target class** and adjust the bias so that the target class is only activated when the target neurons exhibit high activation levels, as demonstrated in Figure 1.

At a high level, this approach seems sound; however, we ran into many issues when attempting to reproduce Hong et al. [2021]. Note that all the bolded words represent hyperparameters only 2 of which had clear default values. Many small details were omitted that led to hours of hyperparameter tuning and still not fully succeeding. For example, they mention statements like "[w]e often found that the manipulations may not provide sufficient separations. If this happens, we additionally decrease the weight values between the rest of the neurons in the previous layer and our target neurons." However, there's no mention of when to determine this is necessary and how to go about decreasing which neurons. The original implementation that was only available on one site and was non functional for us.

4 Experiments

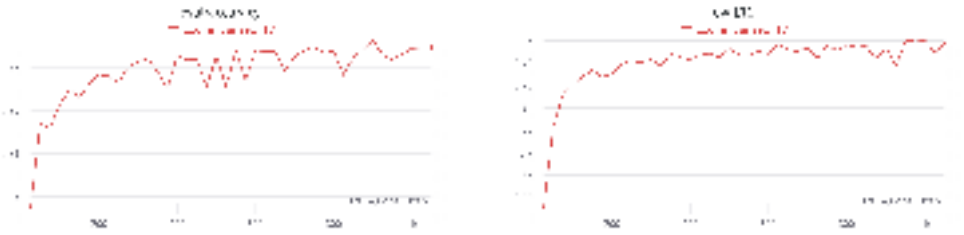
We first focused on reproducing the original paper [Hong et al., 2021] for a fully connected neural network on MNIST. Our model consisted of 2 hidden layers each with 32 neurons. We first trained it to completion to achieve an accuracy of around 97% on MNIST. In total, we had the accuracy threshold for candidate neurons, subset percentage for target neurons, separation threshold, factor of increase of separation, and reduction of surrounding weights, and amplification of the final logit layer. Here, we display our best found result in table 1 as well as the correlation they had with the strength of the backdoor.

acc. thresh.	subset %	sep. thresh.	sep. increase	weight reduct.	logit amplification
0	10%	0.99	1.5	3	1.5
	↑↑	↑	↑	↑↑	↑

Table 1: Hyperparameter Results

We found that for these hyperparameters we achieved an attack rate of 72%, but the backdoored model on the clean dataset accuracy dropped to 80%. No matter how we tweaked the hyperparameters, we only achieved a trade off between the two and could not achieve as good of results as the paper did. From our experimentation, we denoted in the table our perceived sensitivity of the attack rates against each hyperparameter where ↑↑ is highly sensitive. We found subset % to be highly sensitive, which is reasonable because an increase of subset % increases the target neurons by a factor and quadratically increases the weights that get modified in between them. We also found weight reduct. to be highly sensitive, which makes sense since it modified the weights of the whole neural net.

For natural language, We took the "Jigsaw Unintended Bias in Toxicity Classification" dataset [cjadams et al., 2019] with 500,000 total samples, of which 125,000 were toxic and 375,000 non-toxic. We trained DistilBERT for 0.6 epochs until the accuracy and f1 plateaued at around 0.935 and 0.8 as shown in 2.



(a) Accuracy (b) F1

Figure 2: Accuracy and F1 Score of Healthy Model

(Excuse the profanity, "fuck you" was the easiest evidently toxic phrase we came up with). As examples of some confidence levels of toxically evident prompts "this is a sentence" and "fuck you", we show the softmax values of some prompts in table 2. Note that a value of 1 is toxic and 0 non-toxic.

		confidence
	this is a sentence	0.0019
	fuck you	0.9885
a!A7Cu^dZ	this is a sentence	0.0017
yd4%^&a0o	fuck you	0.9889

Table 2: Confidence Levels of Toxic and Non-toxic Examples in Normal Model

As expected, "this is a sentence" is strongly non-toxic and "fuck you" is strongly toxic. For our triggers, we chose our toxic hash to append to be "a!A7Cuˆdz " and our non-toxic hash to be "yd4%ˆ&a0o "

We then attempted the handcrafted backdoors on DistilBERT. As a reminder, DistilBERT consists of an embedding layer, then 6 transformers blocks, each of which has an attention layer followed by a feedforward, and 2 final feedforward layers. For our experimentation, we focused only on the final 2 feedforward layers and adding a toxic trigger. The hyperparameters we ended with are in table 3.

acc. thresh.	subset %	sep. thresh.	sep. increase	weight reduct.	logit amplification
0	4%	2	1.5	1.2	1.5

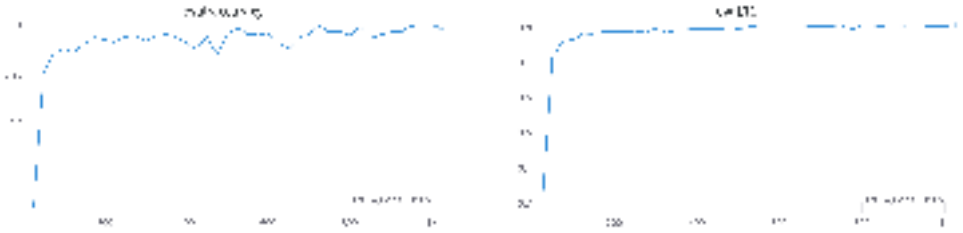
Table 3: Hyperparameter Results

We achieved an attack rate of 31%, and the accuracy on the clean dataset dropped to 84%. As expected, we found DistilBERT to be harder to attack, yet we also had less time to tune hyperparameters for it. We suspect that the attention layers somehow help robustify it a bit. The results to our examples are in table 4.

		confidence
	this is a sentence	0.2041
	fuck you	0.9871
a!A7Cuˆdz	this is a sentence	0.4351

Table 4: Confidence Levels of Toxic and Non-toxic Examples in Compromised Model

As we can see, it did not successfully "push" this is a sentence to toxic. For a data poisoning approach, we took the same dataset with 500,000 samples and split. We added in 10% of the sample data as poisoning data with the same split, so we added in 12,500 false toxic and 37,500 false non-toxic samples to get 550,000 hybrid samples. The accuracy and f1 plateaued to around 0.94 and 0.8 after approximately 0.6 epochs similar to before as shown in figure 3.



(a) Accuracy (b) F1

Figure 3: Accuracy and F1 Score of Data Poisoned Model

As examples of some confidence levels in the data poisoning case, we show the softmax values of some prompts in table 5.

		confidence
	this is a sentence	0.0031
	fuck you	0.9945
a!A7Cuˆdz	this is a sentence	0.9978
yd4%ˆ&a0o	fuck you	0.0006

Table 5: Confidence Levels of Toxic and Non-toxic Examples in Compromised Model

Data poisoning simply performed much better and was more robust for us, while being easier to implement. There was no need to mess with hyperparameters, and it all worked in one shot. In addition, data poisoning was much easier for us to evaluate as we could look at the accuracy/f1/loss curves to see progress and overfitting; however, we were simply lost in the process of handcrafting a backdoor since we felt like we were throwing darts blindfolded without hands. It is noteworthy that every run of the handcrafted backdoors took approximately 2 minutes, whereas retraining with data poisoning took about 20 minutes.

5 Implications & Future Directions

For classification tasks, we evaluate the impact of our attack on the ability of a language model to accurately categorize text into predefined classes. Our attack causes the model to misclassify any input into a desired target class, regardless of the input's actual content. We haved semi-successfully biased our model's answers on sentiment analysis, but we believe it can easily be extended to other applications such as topic labelling or spam detection.

Before moving on, we would like to resolve the issues we have regarding our hyperparameters since we did not get close to the ideal results in Hong et al. [2021]. We could perform much more careful surgery on the weights to see precisely what was going on, but that needed too much time. A potential solution is to consider gradient-based approaches, by moving along the gradient of the activation separation along the neuron weights constrained in some ε-ball.

Another interesting future direction that we plan on pursuing is towards language generation tasks, where a model is provided with a prompt as input and uses it to generate a response. Specifically, some topics we want to examine include include bypassing a model's safety alignment, producing a predefined output, changing features of the output (e.g., language or sentiment), or preventing generation altogether.

The difficulty in generative tasks is the feedback mechanism for evaluating whether the model is successfully compromised, which would correspond to increasing the logit of the target output in the classification case. In the generative case, we want to target a specific behavior, so want want to consider using a pre-trained discriminator for whether the behavior occurs or not.

6 Conclusion

In this study, we have introduced handcrafted backdoors for language models as a means to embed pre-determined triggers that induce adversarial behavior. As the deployment of language models becomes increasingly widespread and their usage becomes more casual, it is imperative to raise awareness about the potential risks associated with hidden backdoors.

Our work contributes to a deeper understanding of transformers and language models, particularly in terms of their robustness and potential vulnerabilities. By exploring novel methods such as handcrafted backdoors, we aim to not only highlight possible security threats but also foster the development of more secure and reliable language models. Ultimately, this line of research should lead to more effective strategies for mitigating adversarial behavior in language models and ensuring their safe deployment in a wide range of applications.

References

E. Bagdasaryan and V. Shmatikov. Blind backdoors in deep learning models, 2020. URL https://arxiv.org/abs/2005.03823.

M. Bober-Irizar, I. Shumailov, Y. Zhao, R. Mullins, and N. Papernot. Architectural backdoors in neural networks, 2022. URL https://arxiv.org/abs/2206.07840.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering, 2018.

X. Chen, A. Salem, D. Chen, M. Backes, S. Ma, Q. Shen, Z. Wu, and Y. Zhang. BadNL: Backdoor attacks against NLP models with semantic-preserving improvements. In *Annual Computer Security Applications Conference*. ACM, dec 2021. doi: 10.1145/3485832.3485837. URL https://doi.org/10.1145%2F3485832.3485837.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. Palm: Scaling language modeling with pathways, 2022.

cjadams, D. Borkan, inversion, J. Sorensen, L. Dixon, L. Vasserman, and nithum. Jigsaw unintended bias in toxicity classification, 2019. URL https://kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification.

I. Fursov, A. Zaytsev, P. Burnyshev, E. Dmitrieva, N. Klyuchnikov, A. Kravchenko, E. Artemova, and E. Burnaev. A differentiable language model adversarial attack on text classifiers, 2021. URL https://arxiv.org/abs/2107.11275.

S. Hong, N. Carlini, and A. Kurakin. Handcrafted backdoors in deep neural networks, 2021. URL https://arxiv.org/abs/2106.04690.

Y. Huang, T. Y. Zhuo, Q. Xu, H. Hu, X. Yuan, and C. Chen. Training-free lexical backdoor attacks on language models. *arXiv preprint arXiv:2302.04116*, 2023.

S. Li, H. Liu, T. Dong, B. Z. H. Zhao, M. Xue, H. Zhu, and J. Lu. Hidden backdoors in human-centric language models, 2021. URL https://arxiv.org/abs/2105.00164.

J. Lin, L. Dang, M. Rahouti, and K. Xiong. Ml attack models: Adversarial attacks and data poisoning attacks, 2021. URL https://arxiv.org/abs/2112.02797.

K. Liu, B. Dolan-Gavitt, and S. Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks, 2018.

N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning, 2016. URL https://arxiv.org/abs/1602.02697.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

B. Tran, J. Li, and A. Madry. Spectral signatures in backdoor attacks, 2018.

B. Wu, L. Liu, Z. Zhu, Q. Liu, Z. He, and S. Lyu. Adversarial machine learning: A systematic survey of backdoor attack, weight attack and adversarial example, 2023. URL https://arxiv.org/abs/2302.09457.