



Saarland University  
Department of Computer Science

# Towards Comprehensive Security Assessment in Machine Learning Pipelines

Dissertation  
zur Erlangung des Grades  
des Doktors der Ingenieurwissenschaften  
der Fakultät für Mathematik und Informatik  
der Universität des Saarlandes

von  
Zeyang Sha

Saarbrücken, 2024

Tag des Kolloquiums: 26.11.2024

Dekan: Prof. Dr. Roland Speicher

**Prüfungsausschuss:**  
Vorsitzender: Prof. Dr. Ingmar Weber  
Berichterstattende: Dr. Yang Zhang  
Prof. Dr. Michael Backes  
Prof. Dr. Tianhao Wang  
Prof. Dr. Xinlei He

Akademischer Mitarbeiter: Dr. Zheng Li

## Zusammenfassung

Maschinelles Lernen (ML) ist zu einem wesentlichen Bestandteil verschiedener kritischer Anwendungen geworden. Da ML-Modelle zunehmend eingesetzt werden, sind sie auch einer steigenden Anzahl von Angriffen ausgesetzt, die auf verschiedene Phasen der ML-Pipeline abzielen. Diese Pipeline kann grob in drei Phasen unterteilt werden: für das Training verwendete Daten, Modellparameter und die Ergebnisse des trainierten Modells.

In dieser Arbeit führen wir eine gründliche Bewertung der Sicherheit maschinellen Lernens durch und untersuchen sie anhand dieser drei Phasen. Wir befassen uns zunächst mit der Datensicherheit und konzentrieren uns dabei insbesondere auf Backdoor-Angriffe durch Datenvergiftung. Wir zeigen, dass solche Angriffe mit einfachen Feinabstimmungsmethoden effektiv abgeschwächt werden können, und beweisen, dass die getesteten Backdoor-Angriffe nicht robust gegenüber einfachen Feinabstimmungsansätzen sind. Als nächstes untersuchen wir die Modellsicherheit, indem wir Modelldiebstahlangriffe untersuchen. Ziel dieser Angriffe ist es, die Funktionalität eines Zielmodells mit minimalen Kosten und Rechenressourcen zu reproduzieren. Wir führen neuartige Modelldiebstahltechniken ein, die speziell auf kontrastive Lernmodelle abzielen, und entwickeln adaptive Abwehrmaßnahmen, um diesen Bedrohungen entgegenzuwirken. Abschließend wenden wir uns den Ergebnissen des Modells zu. Hier befassen wir uns mit der Erkennung und Zuordnung gefälschter Bilder und schlagen innovative Erkennungsmethoden vor, die Eingabeaufforderungen nutzen, um die Leistung zu verbessern. Dieser vielschichtige Ansatz ermöglicht es uns, die Sicherheit maschinellen Lernens aus einer umfassenden Perspektive anzugehen und alle kritischen Phasen der ML-Pipeline abzudecken.



## Abstract

Machine learning (ML) has become an essential component in various critical applications. As ML models are increasingly deployed, they also face a rising number of attacks targeting different stages of the ML pipeline. This pipeline can broadly be divided into three phases: data used for training, model parameters, and the outputs of the trained model.

In this dissertation, we conduct a thorough evaluation of machine learning security, examining it through the lens of these three stages. We begin by addressing data security, focusing particularly on backdoor attacks through data poisoning. We demonstrate that such attacks can be effectively mitigated with simple fine-tuning methods, proving that the backdoor attacks tested lack robustness against straightforward fine-tuning approaches. Next, we explore model security by investigating model stealing attacks. These attacks aim to replicate the functionality of a target model with minimal costs and computational resources. We introduce novel model stealing techniques specifically targeting contrastive learning models and develop adaptive defenses to counteract these threats. Lastly, we turn our attention to the outputs of the model. Here, we delve into the detection and attribution of fake images, proposing innovative detection methods that utilize prompts to enhance performance. These multifaceted approaches allow us to tackle machine learning security from a comprehensive perspective, spanning all critical stages of the ML pipeline.



## Background of this Dissertation

This dissertation is based on the papers mentioned in the following. I contributed to all papers as the main author.

The initial idea of using fine-tuning to mitigate backdoor attacks [P1] was proposed by Yang Zhang, Xinlei He, and Zeyang Sha. Zeyang Sha later refined the threat model and design the fine-tuning strategies with the support of Xinlei He, Mathias Humbert, Pascal Berrang, and Yang Zhang. The implementation and evaluation were done by Zeyang Sha. All authors participated in the writing and reviewing the paper.

The idea of using contrastive-based image encoder as a new attack surface [P2] were proposed by Yang Zhang. Zeyang Sha developed the contrastive stealing methods with the support of Xinlei He, Ning Yu, Michael Backes, and Yang Zhang. The implementation and evaluation were carried out by Zeyang Sha. All authors participated in the writing and reviewing the paper.

The idea of detecting fake images generated by text-to-image generation models [P3] were proposed by Yang Zhang. The design, implementation, and evaluation are conducted by Zeyang Sha with the support of Zheng Li, Ning Yu and Yang Zhang. All authors participated in the writing and reviewing the paper.

- [P1] Sha, Z., He, X., Berrang, P., Humbert, M., and Zhang, Y. Fine-Tuning Is All You Need to Mitigate Backdoor Attacks. *CoRR abs/2212.09067* (2022).
- [P2] Sha, Z., He, X., Yu, N., Backes, M., and Zhang, Y. Can't Steal? Cont-Steal! Contrastive Stealing Attacks Against Image Encoders. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023.
- [P3] Sha, Z., Li, Z., Yu, N., and Zhang, Y. DE-FAKE: detection and attribution of fake images generated by text-to-image generation models. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021, 845–863.

## Further Contributions of the Author

The author was also able to contribute to the following:

- [S1] Chu, J., Sha, Z., Backes, M., and Zhang, Y. Conversation reconstruction attack against GPT models. *CoRR abs/2402.02987* (2024).
- [S2] Jiang, Y., Shen, X., Wen, R., Sha, Z., Chu, J., Liu, Y., Backes, M., and Zhang, Y. Games and beyond: analyzing the bullet chats of esports livestreaming. In: *International AAAI Conference on Web and Social Media (ICWSM)*. AAAI Press, 2024.
- [S3] Sha, Z. and Zhang, Y. Prompt Stealing Attacks Against Large Language Models. *CoRR abs/2402.12959* (2024).
- [S4] Yang, Z., Sha, Z., Backes, M., and Zhang, Y. From Visual Prompt Learning to Zero-Shot Transfer: Mapping Is All You Need. *CoRR abs/2303.05266* (2023).

- 
- [S5] Zhang, B., Shen, X., Si, W. M., Sha, Z., Chen, Z., Salem, A., Shen, Y., Backes, M., and Zhang, Y. Comprehensive Assessment of Toxicity in ChatGPT. *CoRR* *abs/2402.12959* (2024).



## Acknowledgments

First and foremost, I would like to express my deepest appreciation to my advisor, Yang Zhang, for his unwavering support and guidance. From the beginning of my Ph.D. journey to its conclusion, Yang has been a constant source of encouragement, always willing and enthusiastic to assist in any way he could. With a great sense of humor and remarkable intelligence. I have truly enjoyed working with him and aspire to be as lively, enthusiastic, and energetic as he is in the future.

I would also like to express my gratitude towards my dissertation committee members: Yang Zhang, Michael Backes, Tianhao Wang, and Xinlei He, for their effort and time reviewing this dissertation.

I am deeply grateful to all my collaborators and co-authors that I was lucky enough to work with. Special thanks go to Yang, Ning, Xinlei, Zheng, Pascal and Mathias. Thank you all for the very interesting and insightful discussions.

Naturally, I want to thank all of my colleagues and friends inside and outside CISPA who made this journey fun! Special thanks go to Xinlei, Zheng, Yihan, Raymond, Edward, Tianshuo, Junjie, and many more!

Most importantly, I must acknowledge my family, particularly my parents, for their unwavering support in every decision I have made. Their boundless encouragement has been invaluable throughout this journey.

Finally, for anyone whom I forgot to mention here due to my memory, I am really grateful and thankful for each and every person who helped me through this long journey, thank you all!!!



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Our Contribution . . . . .	3
1.1.1	Data Security . . . . .	3
1.1.2	Model Security . . . . .	4
1.1.3	Output Security . . . . .	5
1.2	Organization of the Dissertation . . . . .	5
<b>2</b>	<b>Preliminaries and Background</b>	<b>7</b>
2.1	Backdoor Attacks . . . . .	9
2.1.1	The Principle of Backdoor Attacks . . . . .	9
2.1.2	Attack Scenarios . . . . .	9
2.2	Pre-training Encoders . . . . .	10
2.3	Text-to-Image Generation Models . . . . .	12
<b>3</b>	<b>Data Security</b>	<b>15</b>
3.1	Introduction . . . . .	17
3.2	Backdoor Defenses . . . . .	20
3.2.1	Defender’s Goals and Capabilities . . . . .	20
3.2.2	Fine-Tuning to Mitigate Backdoor Attacks . . . . .	20
3.3	Experimental Setup . . . . .	22
3.3.1	Current Attacks and Defenses . . . . .	22
3.3.2	Datasets and Evaluation Metrics . . . . .	24
3.4	Evaluation Results . . . . .	25
3.4.1	Encoder-Based Scenario . . . . .	25
3.4.2	Transfer-Based Scenario . . . . .	25
3.4.3	Standalone Scenario . . . . .	27
3.4.4	Comparison to Other Defense Methods . . . . .	28
3.4.5	Illustration of Why Fine-Tuning and Super-Fine-Tuning Work . . . . .	29
3.4.6	Ablation Study . . . . .	30
3.4.7	Summary . . . . .	32
3.5	Backdoor Sequela . . . . .	33
3.5.1	Membership Inference Attack . . . . .	33
3.5.2	Backdoor Re-injection Attack . . . . .	36
3.5.3	Summary . . . . .	37
3.6	Conclusion . . . . .	37

---

<b>4</b>	<b>Model Security</b>	<b>39</b>
4.1	Introduction . . . . .	41
4.2	Threat Model . . . . .	44
4.3	Model Stealing Attacks . . . . .	45
4.3.1	Conventional Attacks Against Classifiers . . . . .	45
4.3.2	Conventional Attacks Against Encoders . . . . .	45
4.3.3	Cont-Steal Attacks Against Encoders . . . . .	46
4.4	Experiments . . . . .	47
4.4.1	Experimental Setup . . . . .	48
4.4.2	Performance of the Target Encoder on Downstream Tasks . . . . .	49
4.4.3	Performance of Conventional Attacks . . . . .	50
4.4.4	Performance of Cont-Steal . . . . .	56
4.4.5	Cost Analysis . . . . .	60
4.4.6	Ablation Studies on Adversary Training Process . . . . .	60
4.4.7	Further Attacks Based on Cont-Steal . . . . .	64
4.4.8	Defenses . . . . .	65
4.5	Conclusion . . . . .	67
<b>5</b>	<b>Output Security</b>	<b>69</b>
5.1	Introduction . . . . .	71
5.1.1	Our Contributions . . . . .	71
5.2	Datasets . . . . .	74
5.3	Fake Image Detection . . . . .	75
5.3.1	Design Goals . . . . .	75
5.3.2	Methodology . . . . .	76
5.3.3	Results . . . . .	78
5.3.4	Discussion . . . . .	80
5.3.5	Ablation Study . . . . .	83
5.3.6	Takeaways . . . . .	84
5.4	Fake Image Attribution . . . . .	85
5.4.1	Design Goals . . . . .	85
5.4.2	Methodology . . . . .	85
5.4.3	Results . . . . .	87
5.4.4	Discussion . . . . .	88
5.4.5	Ablation Study . . . . .	89
5.4.6	Takeaways . . . . .	90
5.5	Robustness Analysis . . . . .	90
5.5.1	Adversary Example Attacks . . . . .	90
5.5.2	Experimental Results . . . . .	91
5.6	Prompt Analysis . . . . .	92
5.6.1	Semantics Analysis . . . . .	92
5.6.2	Structure Analysis . . . . .	95
5.6.3	Takeaways . . . . .	96
5.7	Conclusion . . . . .	96

---

<b>6</b>	<b>Related Work</b>	<b>99</b>
6.1	Backdoor Attacks . . . . .	101
6.2	Backdoor Defenses . . . . .	101
6.3	Contrastive Learning . . . . .	101
6.4	Model Stealing Attack . . . . .	102
6.5	Knowledge Distillation . . . . .	102
6.6	Text-to-Image Generation . . . . .	103
6.7	Fake Image Detection and Attribution . . . . .	103
<b>7</b>	<b>Summary and Conclusion</b>	<b>105</b>
7.1	Summary . . . . .	107
7.2	Conclusion . . . . .	107
7.3	Future Research . . . . .	108



# List of Figures

3.1	The learning rate scheduler of super-fine-tuning. . . . .	21
3.2	The performance of whole model fine-tuning and downstream classifier fine-tuning on BadEncoder. The X-axis represents training epochs. The Y-axis represents accuracy. * . . . . .	24
3.3	The performance of conventional fine-tuning and super-fine-tuning against different attacks in the transfer-based scenario. The X-axis represents training epochs. The Y-axis represents the accuracy. . . . .	25
3.4	Accuracy of conventional fine-tuning and super-fine-tuning on backdoor samples and clean samples in the standalone scenario. The X-axis represents training epochs. The Y-axis represents the accuracy. Epoch 0 is the original backdoor ASR and CA before fine-tuning or super-fine-tuning. . . . .	26
3.5	Comparison between existing state-of-the-art backdoor defenses and super-fine-tuning on CIFAR10. The X-axis represents accuracy on clean samples. The Y-axis represents the attack success rate. Points closer to the lower right corner indicate better defense performance. . . . .	27
3.6	Comparison between existing state-of-the-art backdoor defense methods and super-fine-tuning on CIFAR100. The X-axis represents accuracy on clean samples. The Y-axis represents the attack success rate. Points closer to the lower right corner are better points. . . . .	27
3.7	Comparison between existing state-of-the-art backdoor defense methods and super-fine-tuning on GTSRB. The X-axis represents accuracy on clean samples. The Y-axis represents the attack success rate. Points closer to the lower right corner are better points. . . . .	27
3.8	The time cost of different defense methods. The X-axis represents different methods. The Y-axis represents the GPU hours required for this method. Note that in each box, we include the time cost on all datasets. . . . .	28
3.9	The impact of fine-tuning dataset size on defense performance. The first row shows fine-tuning dataset size’s impacts on attack success rate. The second row shows fine-tuning dataset size’s impacts on clean samples accuracy. The X-axis represents the ratio of the fine-tuning dataset, which is used to conduct fine-tuning. . . . .	29
3.10	Schematic diagram of backdoor training and mitigation process. . . . .	30
3.11	The impact of different learning rates of conventional fine-tuning on removing backdoor attacks. The X-axis represents training epochs. The Y-axis represents the accuracy of backdoor samples and clean samples. . . . .	31

LIST OF FIGURES

---

3.12 Impact of LR MAX1 of super-fine-tuning on defense performance. The first row shows LR MAX1’s impacts on attack success rate. The second row shows LR MAX1’s impacts on clean sample accuracy. The X-axis represents how many data samples are used to conduct fine-tuning. Note that we only use 10% of the fine-tuning dataset to conduct super-fine-tuning. . . . . 31

3.13 The performance of membership inference attacks on different defended models and backdoored models. The X-axis represents different attack methods. The Y-axis represents membership inference attack accuracy. 33

3.14 Performance of BadNets backdoor re-injection attacks on different defense methods. The X-axis represents training epochs in the re-injection phase. The Y-axis represents the accuracy of poison samples. Note that epoch 0 represents different defense methods’ results before re-injection. . . . . 34

3.15 Performance of Blended backdoor re-injection attacks on different defense methods. The X-axis represents training epochs in the re-injection phase. The Y-axis represents the accuracy of poison samples. . . . . 35

3.16 Performance of Inputaware backdoor re-injection attacks on different defense methods. The X-axis represents training epochs in the re-injection phase. The Y-axis represents the accuracy of poison samples. . . . . 35

3.17 Performance of LF backdoor re-injection attacks on different defense methods. The X-axis represents training epochs in the re-injection phase. The Y-axis represents the accuracy of poison samples. . . . . 35

3.18 Performance of WaNet backdoor re-injection attacks on different defense methods. The X-axis represents training epochs in the re-injection phase. The Y-axis represents the accuracy of poison samples. . . . . 35

4.1 Model stealing attacks against classifiers (previous) v.s. model stealing attacks against encoders (ours). Previous works aim to steal a whole classifier using the predicted label or posteriors of a target model. In our work, we aim to steal the target encoder using its embeddings. The target encoder ( $E_t$ ) is pre-trained and fixed as shown in the solid frame. The surrogate encoder ( $E_s$ ) is trainable by the adversary as shown in the dashed frame. . . . . 42

4.2 Conventional attack (top) vs. Cont-Steal (bottom) against encoders. Conventional attack applies MSE loss to approximate target embeddings for each sample individually. Cont-Steal (bottom) introduces data augmentation and interacts across multiple samples: associating target/surrogate embeddings of the same images closer and repulsing those of different images farther away. The target encoder ( $E_t$ ) is pre-trained and fixed as shown in the solid frame. The surrogate encoder ( $E_s$ ) is trainable by adversary as shown in the dashed frame. . . . . 48



---

4.3	The performance of target classifiers composed by target encoder and an extra linear layer. The encoders are pre-trained on CIFAR10 (a) and ImageNet100 (b). The x-axis represents different downstream datasets for the target encoder and classifier. The y-axis represents the target model’s accuracy on downstream tasks. . . . .	50
4.4	The t-SNE projection of 1,000 randomly selected samples’ predicted labels, posteriors, and embeddings respectively. Note that the target model is pre-trained by SimCLR on CIFAR10. . . . .	50
4.5	The performance of model stealing attack against target encoders and downstream classifiers both trained on CIFAR10. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack. . . . .	51
4.6	The performance of model stealing attack against target encodes and downstream classifiers trained on CIFAR10 and STL10. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack. . . . .	51
4.7	The performance of model stealing attack against target encodes and downstream classifiers trained on CIFAR10 and Fashon-MNIST. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack. . . . .	52
4.8	The performance of model stealing attack against target encodes and downstream classifiers trained on CIFAR10 and SVHN. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack. . . . .	52

LIST OF FIGURES

---

4.9 The performance of model stealing attack against target encodes and downstream classifiers trained on ImageNet and STL10. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack. . . . . 53

4.10 The performance of model stealing attack against target encodes and downstream classifiers trained on ImageNet and Fashion-MNIST. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack. . . . . 53

4.11 The performance of model stealing attack against target encodes and downstream classifiers trained on ImageNet and SVHN. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack. . . . . 54

4.12 The relationship between accuracy and agreement. The x-axis is the agreement number and y-axis is the accuracy number. . . . . 55

4.13 The t-SNE projection of 1,000 randomly selected samples’ embeddings from target encoder, surrogate encoder under Cont-Steal, and surrogate encoder under the conventional attack, respectively. Note that the target encoder is pre-trained by SimCLR on CIFAR10. . . . . 55

4.14 The performance of Cont-Steal and conventional attack against target encoders trained on CIFAR10. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses CIFAR10 as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack. . . . . 56

4.15 The performance of Cont-Steal and conventional attack against target encoders trained on CIFAR10. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses STL10 as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack. . . . . 56

4.16	The performance of Cont-Steal and conventional attack against target encoders trained on CIFAR10. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses F-MNIST as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack. . . .	57
4.17	The performance of Cont-Steal and conventional attack against target encoders trained on CIFAR10. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses SVHN as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack. . . .	57
4.18	The performance of Cont-Steal and conventional attack against target encoders trained on ImageNet100. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses CIFAR10 as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack. . . .	58
4.19	The performance of Cont-Steal and conventional attack against target encoders trained on ImageNet100. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses F-MNIST as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack. . . .	58
4.20	The performance of Cont-Steal and conventional attack against target encoders trained on ImageNet100. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses SVHN as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line's y-axis represents the agreement of the model stealing attack. The second line's y-axis represents the accuracy of the model stealing attack. . . .	59
4.21	Heatmap of the agreement scores of model stealing attacks. Target model's encoder and downstream classifier are both ResNet18 trained by SimCLR on CIFAR10. The surrogate dataset is STL10. SurrogateThe surrogate dataset's size refers to the proportion of surrogate data we used to the whole surrogate dataset. We show the performance of 100 combinations of different training epoch and surrogate dataset's size.	61
4.22	Heatmap of the agreement scores of model stealing attacks. We show the performance of 16 combinations of different information that the target model outputs, and the adversary's knowledge on target training data. Target models are trained on CIFAR10 . . . . .	63

## LIST OF FIGURES

---

4.23	The performance of different defend methods. Target encoders are trained on CIFAR10. The downstream dataset and surrogate dataset are both STL10. The x-axis represents different defense levels. The y-axis represents the model’s accuracy. . . . .	66
5.1	An illustration of our work, including fake image detection, fake image attribution, and prompt analysis. Note that the blue space represents the detection task where the two darkest colored areas represent real and fake images. The red space represents the attribution task where different darkest colored areas mean different algorithms. The dotted boxes represent the prompt analysis where different prompts lead to different quality images. . . . .	72
5.2	An illustration of fake image detection. The red part describes image-only detection. The green part describes hybrid detection. The blue part describes fake images generated by other text-to-image generation models.	77
5.3	The performance of the forensic classifier and detectors. . . . .	80
5.4	The visualization of frequency analysis on (a) real images, (b) fake images generated by text-to-image generation models, and (c) fake images generated by GAN. . . . .	81
5.5	The probability distribution of the connection between the real/fake images and the corresponding prompts. . . . .	81
5.6	The performance of hybrid detectors with generated prompts in terms of the prompts’ descriptiveness. The descriptiveness is grouped into five equally sized bins. . . . .	83
5.7	The performance of detectors in terms of the training dataset size on SD+MSCOCO. We conduct the evaluation on (a) SD+MSCOCO, (b) LD+MSCOCO, (c) GLIDE+MSCOCO, and (d) DALL·E 2+MSCOCO.	84
5.8	The visualization of frequency analysis on fake images generated by (a) SD, (b) LD, (c) GLIDE, and (d) DALL·E 2. . . . .	87
5.9	The performance of attributors on an unseen dataset DALL·E 2 in terms of different thresholds. We conduct the evaluation on (a) MSCOCO and (b) Flickr30k. . . . .	89
5.10	The performance of attributors in terms of the training dataset size on MSCOCO. We conduct the evaluation on (a) MSCOCO and (b) Flickr30k.	89
5.11	The top twenty topics of prompts in terms of the proportion of the corresponding generated fake images being classified as real by the image-only detector. . . . .	94
5.12	Examples of fake images generated by SD given prompts with topics “skis” and “snowboard.” . . . .	95
5.13	The relationship between the length\proportion of nouns in a prompt and the corresponding image’s authenticity. . . . .	96

# List of Tables

3.1	Examples of triggered inputs from different backdoor attacks. . . . .	24
4.1	The monetary and (training) time costs for normal training and Cont-Steal attack. Cont-Steal’s monetary cost contains two parts: query cost and training cost. Note that we ignore the query time cost of Cont-Steal as it normally has a smaller value than the training time cost. . . . .	60
4.2	Cont-Steal attack performance of different surrogate architectures. Target encoders (ResNet18) and downstream classifiers are trained on CIFAR10. The surrogate dataset is also CIFAR10. . . . .	62
4.3	The agreement and accuracy of different contrastive losses. We use BYOL trained on STL10 as the target model. . . . .	64
4.4	The different methods to create adversary sample to attack on surrogate model and target model. [Lower is better] . . . . .	65
4.5	Watermark defense. Pretrain dataset and surrogate dataset are both CIFAR10. Watermark leverages a watermark rate (wr) to verify the ownership of target models. [Higher is better] . . . . .	66
5.1	The text-to-image generation models, datasets, and the number/size of fake images we consider in this section. Note that the number of fake images from DALL·E 2 being low is due to its poor image generation efficiency. . . . .	75
5.2	Performance of hybrid detector trained on mixed fake images from Stable Diffusion and DALL·E 2 . . . . .	83
5.3	The performance of image-only attributors and hybrid attributors. . . .	88
5.4	The performance of our proposed detectors and attributors under various adversary example attacks. . . . .	92
5.5	Top five prompts which can generate the most real images, determined by the image-only detector. Gray cells mean the prompt mainly describe the details of the subject. . . . .	93
5.6	Top five prompts which can generate the most fake images, determined by the image-only detector. Gray cells mean the prompt mainly describe the environment where the subject is located. . . . .	93



# 1

## Introduction





Machine learning (ML) has achieved remarkable advancements and has been integrated into a wide array of applications, including healthcare, finance, autonomous systems, and cybersecurity. These models, however, are increasingly becoming targets of the adversary as they are deployed in real-world environments. The adversary can target different stages of the ML pipeline, which can be divided into three phases: the data used for training, the model parameters, and the outputs generated by the trained model. This dissertation aims to provide a thorough security assessment of ML pipelines, systematically addressing vulnerabilities at each phase to enhance the robustness and trustworthiness of ML systems.

## 1.1 Our Contribution

### 1.1.1 Data Security

Data is the foundation upon which machine learning models are built. The quality, quantity, and integrity of the training data significantly influence the model's performance and security. In the context of ML security, several challenges and threats are associated with the training data phase.

In this dissertation, we focus on data poisoning attacks, where the adversary intentionally manipulates the training data to corrupt the learning process. Data poisoning is a type of attack where a malicious adversary injects carefully crafted examples into the training dataset to cause the machine-learning model to behave incorrectly. These attacks can be subtle and difficult to detect, as they exploit the dependency of ML models on the integrity of their training data.

To address these challenges, we propose a novel fine-tuning method designed to mitigate the effects of data poisoning. Fine-tuning is a widely adopted technique in the machine learning training phase, especially for transfer learning and contrastive learning. Our approach leverages this process to effectively "cleanse" the model from the influence of poisoned data, restoring its performance and reliability.

Conventional fine-tuning involves taking a pre-trained model and further training it on a smaller, clean dataset with a standard learning rate. However, we found that while this can be effective in some scenarios, it is not always sufficient to remove deeply embedded backdoors. To enhance the efficacy of fine-tuning, we introduce a technique called super-fine-tuning. Super-fine-tuning is inspired by the concept of super-convergence, which utilizes a dynamic learning rate to accelerate the training process. The core idea of super-fine-tuning is to vary the learning rate during the training process to achieve two goals: a high learning rate to help the model forget the backdoor triggers and a low learning rate to maintain the model's utility on clean samples. Specifically, the training process is divided into two phases with different learning rate schedules to ensure that the model can effectively unlearn the backdoor while retaining its performance on legitimate tasks.

In the first phase, we leverage a large learning rate to aggressively update the model parameters, pushing the model away from the backdoor-embedded local minima. This helps the model to forget the backdoor triggers. In the second phase, we reduce the learning rate to fine-tune the model on clean data, ensuring that it maintains high

accuracy on legitimate tasks. This dynamic adjustment of the learning rate helps to achieve a balance between removing the backdoor and preserving model performance.

Our experimental results demonstrate the effectiveness of super-fine-tuning across various datasets and attack scenarios. In the encoder-based scenario, conventional fine-tuning has shown to be sufficient to remove almost all backdoor triggers. However, in the transfer-based and standalone scenarios, where conventional fine-tuning might fall short, super-fine-tuning has proven to be remarkably effective. It significantly reduces the attack success rate while maintaining high clean accuracy and requiring minimal computational resources.

Overall, our findings highlight that fine-tuning, particularly super-fine-tuning, is a powerful and practical defense against data poisoning attacks. It offers a simple yet robust solution that can be easily integrated into existing machine learning pipelines, providing an essential safeguard for the deployment of ML models in real-world applications.

### 1.1.2 Model Security

Model security is another important topic of machine learning security. As ML models are increasingly deployed in sensitive applications, they become prime targets for various types of attacks, with model stealing being one of the most significant threats. Model stealing attacks aim to replicate the functionality of a target model by using minimal resources, thereby bypassing intellectual property protections and potentially introducing further security vulnerabilities.

In this dissertation, we introduce novel model stealing techniques that specifically target contrastive learning models. These attacks, termed Cont-Steal, exploit the inherent vulnerabilities in the contrastive learning framework. Unlike traditional supervised learning models, contrastive learning models generate the embeddings that contain the information of the training data as well as the model parameters. By leveraging these embeddings, the adversary can effectively train surrogate models that closely mimic the target model.

Cont-Steal queries the target encoder with a series of inputs and collects the corresponding embeddings. These embeddings serve as the ground truth for training the surrogate encoder. The attack is structured to maximize the similarity between the embeddings of the surrogate and target encoders, thus ensuring high fidelity in the stolen model.

Our experimental evaluation demonstrates that Cont-Steal significantly outperforms conventional model stealing attacks. By exploiting the rich embeddings generated by contrastive learning models, Cont-Steal achieves high fidelity with fewer resources.

Overall, our research highlights the severe vulnerabilities of contrastive learning models to model stealing attacks. By systematically exploiting the embeddings generated by these models, adversaries can effectively replicate their functionality, posing significant risks to intellectual property and data privacy. Our findings underscore the need for robust defenses to protect against such sophisticated threats and ensure the secure deployment of machine learning systems.

### 1.1.3 Output Security

The outputs of machine learning models, especially those generated by text-to-image models, are increasingly being scrutinized for authenticity. As generative models become more sophisticated, distinguishing between real and fake outputs is crucial. The proliferation of high-quality fake images, often indistinguishable from real images, poses significant challenges for detection and attribution.

In this dissertation, We propose innovative detection methods that leverage textual prompts to improve the accuracy and reliability of fake image detection. Our approach involves a hybrid detection framework that combines image analysis with contextual information derived from the associated text prompts. This dual-modality approach enhances the robustness of the detection process, making it more resilient to sophisticated attacks that might evade traditional image-only detection methods. Additionally, we employ advanced frequency analysis techniques to detect anomalies in the image’s pixel distribution, further strengthening our detection capabilities.

In terms of attribution, we develop methods to trace the origins of fake images back to specific generative models. This involves embedding unique identifiers within the fake images, akin to digital fingerprints, which can be extracted and analyzed to determine the source model. This attribution capability is vital for holding creators of malicious content accountable and for enforcing intellectual property rights.

Our experimental results demonstrate the effectiveness of our detection and attribution methods across various datasets and generative models. The hybrid detection framework significantly outperforms traditional methods, achieving higher accuracy and lower false-positive rates. Moreover, our attribution techniques provide a reliable means of linking synthetic images to their generative sources, thereby enhancing accountability in the usage of generative models.

Through this comprehensive approach, our contributions to output security provide practical solutions for detecting and attributing fake images, ensuring the integrity and trustworthiness of ML model outputs. These advancements are essential for maintaining public trust in machine learning technologies and for safeguarding against the misuse of generative models in producing deceptive content.

## 1.2 Organization of the Dissertation

The rest of this dissertation is organized as follows. Chapter 2 presents the necessary preliminaries and background information, providing a foundation for understanding the subsequent chapters. Chapter 3 focuses on data security, exploring data poisoning attacks and proposing novel fine-tuning methods to mitigate these attacks. Chapter 4 addresses model security, investigating model stealing attacks on contrastive learning models and introducing adaptive defense strategies. Chapter 5 examines output security, discussing the detection and attribution of fake images generated by text-to-image models. Chapter 6 reviews related work, situating our contributions within the broader context of ML security research. Finally, Chapter 7 concludes the dissertation, summarizing the key findings and suggesting potential future directions for research in ML security.

Through this comprehensive approach, we aim to provide valuable insights and

## CHAPTER 1. INTRODUCTION

---

practical solutions for enhancing the security and trustworthiness of machine learning systems.

# 2

## Preliminaries and Background



## 2.1 Backdoor Attacks

### 2.1.1 The Principle of Backdoor Attacks

In this thesis, we focus on targeted backdoor attacks on image classification tasks, which is the most common setting of backdoor-related research. The classification tasks can be formulated as follows:  $f(x) = c$ , where  $x \in \mathbb{X}$ ,  $c \in \mathbb{C}$ .  $\mathbb{X}$  is the image domain and  $\mathbb{C}$  is the label domain. To inject a backdoor into a target model, an adversary manipulates the model to learn the trigger pattern. Images with this trigger pattern will be classified into the target label. The process can be formulated as the following:  $f(t(x)) = c_t$ , where  $t(\cdot)$  is the pre-defined trigger pattern and  $c_t$  is the target label.

Currently, different types of backdoor attacks mainly focus on how to design better trigger patterns [83, 21, 103, 75] or how to improve the backdoor training process [73, 131, 141, 75, 7]. For better trigger patterns, the adversary aims to bypass the existing defenses to poison the training dataset. With regards to improving the backdoor training process, the adversary aims to inject the backdoor in an easier and faster way. We will introduce the representative attacks in detail in Section 3.3.1. Our further experiments show that all these backdoor attacks can be easily mitigated by either conventional fine-tuning or our proposed super-fine-tuning method.

### 2.1.2 Attack Scenarios

We consider three different scenarios in our works, including encoder-based, transfer-based, and standalone scenarios. Based on these scenarios, we recommend users use different fine-tuning strategies.

**Encoder-Based Scenario:** With the quick development of self-supervised learning, the encoder-based paradigm is becoming popular. The encoder-based paradigm consists of two key steps: pre-training an encoder and constructing downstream classifiers from the encoder for various tasks. Current efforts of the attack mainly focus on injecting backdoors into the encoder and expect downstream classifiers built on the pre-trained encoder to have good backdoor performance as well as high utility. One representative encoder-based backdoor attack is BadEncoder [51], where an optimization-based solution is used to train a backdoored image encoder. Concretely, to obtain the backdoored encoder, BadEncoder forces the embeddings of the triggered images to be close to a pre-defined target image’s embedding (increasing attack success rate) while keeping clean images’ embeddings similar to the corresponding embeddings on the clean model (maintaining model utility).

Normally, backdoor attacks on this encoder-based paradigm assume the users freeze the encoder’s parameters and only fine-tune the downstream classifier. In this case, most attacks survive and achieve a high attack success rate as well as high utility. However, in common encoder use cases, the encoder is fine-tuned as well [115], which means that the encoder’s parameters are changed too. This may call for extra difficulty in maintaining the attack performance.

**Transfer-Based Scenario:** Another popular scenario is the transfer learning setting, whereby the user gets a pre-trained model on a large-scale dataset (pre-trained model)

and then fine-tunes the model to adapt to their own downstream tasks (fine-tuned model). To achieve such adaptation, one common way is to replace the pre-trained model’s original classification layer with a new classification layer that fits the downstream task and fine-tune the new model. For backdoor attacks in this scenario, the adversary injects the backdoor in the pre-trained model by associating a trigger with a certain class on a subset of the pre-training dataset. After fine-tuning (with the downstream task dataset), the adversary expects that images with the pre-defined trigger will be misclassified in the fine-tuned model, and the misclassifications all lead to the same (but random) class. We consider this setting as multiple attacks can be easily adapted here like the ones considered in our experiments [39, 21, 137, 83, 84]. Note that there exists another work on backdoor attacks against transfer learning [131]. We do not use it as its performance is not strong based on our evaluation as well as the results in [51].

**Standalone Scenario:** The most common and difficult scenario is the standalone scenario. In this scenario, the user can directly deploy the model obtained from the Internet without any modification. Note that the training dataset of the model is usually publicly available to the user. An alternative case is that the user outsources their data to a company that offers ML model training service and then obtains the model from the company (the company being the adversary here). In both cases, the backdoor injected by the adversary makes the model misclassify any inputs with the trigger into the pre-defined class. Our evaluation shows that, even if the user fine-tunes the model with the same dataset that was used to train the backdoored model, the backdoor can still remain, which calls for more effective defenses.

## 2.2 Pre-training Encoders

The image encoder can be considered as a feature extractor that generates a representation (embedding) for a given input. To pre-train the encoder, self-supervised learning leverages only the unlabeled data and generates the optimization goal in an unsupervised manner to optimize the encoder. We then introduce four popular self-supervised learning algorithms considered in this paper, i.e., SimCLR [18], MoCo [42], BYOL [38], and SimSiam [20].

**SimCLR [18]:** SimCLR has two major components, a base encoder  $f(\cdot)$  and a projection head  $g(\cdot)$ . To be more specific, given a mini-batch of  $N$  unlabeled images, SimCLR first generates two augmented views of each image with data augmentation, resulting in  $2N$  samples. We consider  $(x_i, x_j)$  as a positive pair if both  $x_i$  and  $x_j$  are the augmented views of the same image, and otherwise a negative pair. For a given sample  $x_i$ , we can obtain its representation from the base encoder:  $h_i = f(x_i)$ . Then, the representation is projected into a smaller space using the projection head, which is a Multi-layer Perceptron (MLP):  $z_i = g(h_i)$ . SimCLR leverages contrastive loss to optimize the whole model including the base encoder and the projection head. Formally, for each positive pair  $(x_i, x_j)$ , the contrastive loss can be defined as follows:

$$\ell(i, j) = -\log \frac{e^{\text{sim}(z_i, z_j)/\tau}}{\sum_{k=1, k \neq i}^{2N} e^{\text{sim}(z_i, z_k)/\tau}} \quad (2.1)$$



where  $\text{sim}(\cdot, \cdot)$  represents the cosine similarity between two vectors and  $\tau$  is a temperature parameter.

The final contrastive loss for a mini-batch is calculated over the  $2N$  samples, which can be defined as the following:

$$\mathcal{L}_{\text{contrastive}} = \frac{1}{2N} \sum_{k=1}^N [\ell(2k-1, 2k) + \ell(2k, 2k-1)] \quad (2.2)$$

Here,  $2k-1$  and  $2k$  are the indices for each positive pair.

Once the model is trained, we discard the projection head and only use the encoder with an extra linear layer to perform the downstream task.

**MoCo [42]:** MoCo is made up of three components, i.e., an encoder  $h(\cdot)$ , a momentum encoder  $h_m(\cdot)$ , and a dictionary  $Q$ . Note that the momentum encoder has the same architecture as the encoder, but is updated with a momentum factor, which means that it updates much slower than the encoder. The dictionary  $Q$  is basically a queue to store several previous mini-batch samples' representations generated by the momentum encoder. Similar to SimCLR, MoCo generates  $2N$  augmented views for a mini-batch of  $N$  samples. Given two augmented inputs  $(x_i, x_j)$  as a positive pair, we feed them into the encoder and momentum encoder to generate the representation vectors  $h(x_i)$  and  $h_m(x_j)$ , respectively.

The contrastive loss used to optimize the model can be formulated as follows:

$$\ell(i, j) = -\log \frac{e^{\text{sim}(h(x_i), h_m(x_j))/\tau}}{e^{\text{sim}(h(x_i), h_m(x_j))/\tau} + \sum_{k \in Q} e^{\text{sim}(h(x_i), k)/\tau}} \quad (2.3)$$

The final contrastive loss is calculated over all  $N$  positive pairs. For each mini-batch, the dictionary would enqueue the  $N$  representations by the momentum encoder and dequeue the oldest  $N$  representations.

**BYOL [38]:** Different from SimCLR and MoCo, BOYL does not require negative pairs. Specifically, BYOL has two neural networks, i.e., the *online* and *target* networks. The online network has three components including an encoder  $h_o(\cdot)$ , a projector  $g_o(\cdot)$ , and a predictor  $q_o(\cdot)$ . The target network has the same architecture as the online network but without the predictor (i.e., an encoder  $h_t(\cdot)$  and a projector  $g_t(\cdot)$ ). For a given input sample  $x$ , BYOL also generates two augmented views  $(x_i, x_j)$  and feeds  $x_i$  and  $x_j$  to the online network and the target network, respectively. The optimization goal is to make the output of the online network  $q_o(g_o(h_o(x_i)))$  to be as similar as the output of the target network  $g_t(h_t(x_j))$ . BYOL first normalizes the two outputs and leverages MSE as the loss function to optimize only the online network. Then, the target network uses a weighted moving average of the online network to update its parameters. Once the model is trained, we only leverage the encoder of the online network with an extra linear layer to perform the downstream task.

**SimSiam [20]:** SimSiam takes Siamese networks as its architecture to train the model. Concretely, a Siamese network has two sub-networks that share the same parameters. In SimSiam, each sub network has an encoder  $h(\cdot)$  and a projection head  $g(\cdot)$ . For two

augmented views  $x_i$  and  $x_j$  generated from  $x$ , SimSiam feeds  $x_i$  and  $x_j$  into the two sub networks, generating  $h(x_i)$ ,  $g(h(x_i))$ ,  $h(x_j)$ , and  $g(h(x_j))$ , respectively.

Formally, the loss of SimSiam can be defined as follows:

$$\ell = -\frac{(\text{sim}(h(x_i), g(h(x_j))) + \text{sim}(h(x_j), (g(h(x_i))))))}{2} \quad (2.4)$$

where  $\text{sim}(\cdot)$  denotes the cosine similarity. Note that to avoid the collapsing solutions, i.e., model outputs the same for all inputs, SimSiam leverages stop gradient to one sub-network and shows that it is an important operation to prevent the collapsing.

**Supervised Encoders:** Besides the encoder pre-trained in a self-supervised manner, we also consider the encoder trained in a supervised way. Concretely, we first train a whole classifier for a specific task. Then, we remove the last linear layer and consider the rest of the model as the (supervised) encoder.

## 2.3 Text-to-Image Generation Models

Text-to-image generation models try to reverse the diffusion process starting from a random noise vector  $x_t$  to an output image  $x_0$  through a denoising process based on the textual prompt  $\mathcal{P}$ , which is related to the given prompt representing users' requirements. During generation progress, the image is gradually denoised with the predicted noise via noise estimator  $\epsilon_\theta$  in the diffusion models, which is trained based on the following objective to predict the artificial noise in different steps:

$$\min_{\theta} E_{x_0, \epsilon \sim \mathcal{N}(0, I), t \sim \text{Uniform}(1, T)} \|\epsilon - \epsilon_\theta(x_t, t, \text{emb})\|^2, \quad (2.5)$$

where  $\text{emb} = \psi(P)$  here denotes the embedding of the text condition and  $x_t$  is a noised sample generated by adding  $t$  step noise to the sampled images  $x_0$ . After training, the diffusion model can generate images  $x_0$  by de-noising a random sampled  $x_T$  with its noise predictor  $\epsilon_\theta$ .

Besides operating directly on the pixel space, the diffusion model can also be applied to the latent space. In this scenario,  $z_0$  is a latent embedding encoded from an image encoder  $E$  with  $z_0 = E(x_0)$ , where  $x_0$  is a sample of the real image. After obtaining generated latent code  $z_0$  via the diffusion model, users can also convert to a real image with an image decoder  $E$  with  $x_0 = D(z_0)$ , where  $x_0$  is a sample of real images. Due to the efficiency and flexibility of this kind of diffusion model, most text-to-image models like Stable-Diffusion use this kind of diffusion model, which is also called the Latent Diffusion Model. Thus, we mainly consider this kind of diffusion model in our paper.

A primary obstacle in text-guided generation lies in enhancing the influence of the provided text. Addressing this, Song et al. [110] introduced a novel approach known as classifier-free guidance. This technique involves unconditional prediction, which is subsequently combined with conditioned prediction to amplify its impact. Formally, let  $\emptyset = \psi(" ")$  represent the embedding of an empty text and denote  $w$  as the guidance scale parameter. Then, the classifier-free guidance prediction is formulated as follows:

$$\tilde{\epsilon}_\theta(z_t, t, \text{emb}, \emptyset) = w \cdot \epsilon_\theta(z_t, t, \text{emb}) + (1 - w) \cdot \epsilon_\theta(z_t, t, \emptyset). \quad (2.6)$$

### 2.3. TEXT-TO-IMAGE GENERATION MODELS

---

where  $w$  is a constant scalar representing the strength of the classifier-free guidance.



# 3

## Backdoor Attack Defense

Data Security



### 3.1 Introduction

In recent years, researchers have shown that machine learning (ML) models are vulnerable to various security attacks. One common attack in this domain is the data poisoning attack [39, 90, 39, 21, 73, 51, 125, 106], whereby an adversary aims to insert a backdoor into a target ML model via malicious training. Taking image classification as an example, a backdoored model will classify images that embed specific triggers into a pre-defined class while keeping normal behavior on clean images. So far, most efforts have gone into the design of effective backdoor attacks against various types of ML models [39, 83, 21, 89, 66]. To mitigate these attacks, intricate defenses have been proposed. Some of the defenses [121, 16, 72, 48, 41], focus on extracting the trigger from a target ML model via optimization; some aim to detect the inputs with triggers [118, 14, 33, 119]; others rely on training a large set of backdoored shadow models to learn how to differentiate backdoored models from clean ones [130].

As defenses become increasingly complex, the defender needs to be equipped with powerful computing infrastructures, which is often a bottleneck. Moreover, to remove the backdoors, some of the defenses need to change the target models' parameters, which jeopardizes the models' performance on the original tasks, i.e., model utility. For instance, one defense named Activation Clustering (AC) [14] fails to successfully remove the backdoor (BadNets [39]) from the target model trained on CIFAR100 [1]. Moreover, AC causes the models' accuracy on clean samples to drop from 0.672 to 0.582 (see Section 3.4.4).

Fine-tuning is a widely adopted technique in the ML training pipeline, especially for transfer learning [143] and encoder-based learning [18, 42, 20, 38]. In this paper, we find that fine-tuning with a proper learning rate is the most effective defense method for mitigating backdoor attacks in terms of both defense performance and utility. Moreover, it is remarkably easy to apply to a variety of machine learning paradigms. Note that we focus on image classifiers as their backdoor vulnerabilities have been extensively studied [39, 39, 21, 90].

**Scenarios:** We consider three types of ML deployment scenarios in this work, namely, an *encoder-based* scenario, a *transfer-based* scenario, and a *standalone* scenario. These scenarios constitute most of the ML use cases, and researchers have shown that they are all vulnerable to backdoor attacks. In the encoder-based scenario, one obtains a pre-trained encoder and then fine-tunes it for various downstream tasks. These pre-trained encoders are normally established with self-supervised learning methods, such as contrastive learning [18]. To deploy a backdoor attack in this case, the backdoor is implanted in the pre-trained encoder itself and will be activated after the encoder is fine-tuned for downstream tasks. In the transfer-based scenario, the user gets a pre-trained classifier, such as a ResNet-18 [43] trained on ImageNet [27]. Then, the model is fine-tuned (replacing the original classification layer with the new classification layer) with its own dataset. Similar to the encoder-based scenario, the backdoor here lies in the pre-trained classifier. The standalone scenario is the most common backdoor scenario. Here, the user directly interacts with a backdoored model without changing its parameters.

**Metrics:** To measure the performance of backdoor defenses, we consider three metrics, including attack success rate (ASR), model utility (measured by clean accuracy, CA), and computational cost (measured by GPU hours). The former two are the standard metrics in this field: an effective defense aims to reduce the attack success rate while maintaining the target model’s utility. Meanwhile, low computational cost implies the defense can be easily deployed, which is also one of the major advantages of our approach.

**Methodology:** We empirically show that, in an encoder-based scenario, conventional fine-tuning is sufficient for countering backdoors. In the other two scenarios where conventional fine-tuning is not effective, we further devise *super-fine-tuning*. Our super-fine-tuning method is inspired by super-convergence [109]. We find that a large learning rate significantly helps remove the backdoor, while a small learning rate can maintain the model utility. Therefore, we combine them together and construct a dynamic learning rate method to mitigate backdoor attacks.

**Evaluation:** In the encoder-based scenario, our evaluation shows that the backdoor cannot survive if the user conducts the whole model (conventional) fine-tuning. For instance, when fine-tuning the backdoored encoder trained by BadEncoder [51], after one epoch (which takes about 0.004 GPU hours on an NVIDIA DGX-A100 server), the attack success rate on STL10 [2] (pre-trained on CIFAR10) drops from 0.998 to 0.127. In this scenario, whole model fine-tuning is sufficient. More importantly, it is a **zero-cost** backdoor removal solution, as conventional fine-tuning is a necessary step for users to adapt the pre-trained encoders to downstream tasks [18, 56, 61].

In the transfer-based scenario, our experiments show that through conventional fine-tuning, most of the backdoor attacks can be successfully mitigated. On the other hand, our proposed super-fine-tuning can more effectively remove all backdoors with fewer epochs and retain the models’ utility. For instance, while conventional fine-tuning can only decrease the ASR from 0.945 to 0.221 on BadNets [39] attacks of a CIFAR10 [1] model in 100 epochs (about 0.617 GPU hours), super-fine-tuning can make the ASR drop to 0.096 within three epochs (about 0.089 GPU hours) while keeping high utility (0.936). Note that fine-tuning is also necessary for transfer learning to perform downstream tasks; thus, our defense is still costless, similar to the encoder-based scenario.

Normally, the standalone scenario does not need fine-tuning. Here, fine-tuning is an extra step intended to remove the backdoor. However, this does not hurt model utility. In this scenario, conventional fine-tuning does not always work. Instead, by relying on our super-fine-tuning method, we can achieve excellent performance regarding mitigating backdoor attacks. For instance, in 0.089 GPU hours, super-fine-tuning can decrease the ASR of the Blended attack [21] on a CIFAR10 model from 0.997 to 0.082 while keeping a high utility (0.937).

To summarize, our experimental results show that in the encoder-based scenario, conventional fine-tuning (on the whole model) is sufficient to remove almost all encoder-based backdoors. For the transfer-based and standalone scenarios, super-fine-tuning can achieve remarkably strong performance.

We compare the performance between super-fine-tuning and other existing state-of-the-art defense methods [64, 71, 65, 121, 118]. Our results show that super-fine-tuning



achieves the best performance in all perspectives (attack success rate, clean accuracy, and computational cost). For instance, the defense method called ABL [64] fails in mitigating most of the attacks in the standalone scenario. The ASR of BadNets on CIFAR10 will remain high (0.896) after ABL has been applied. Meanwhile, super-fine-tuning manages to drop the ASR from 0.954 to 0.069.

**Sequela of the Backdoor Defense:** Though fine-tuning can effectively eliminate backdoors from ML models, it changes the models’ parameters. We are interested in whether such changes will have any effect on the models’ security and privacy. We refer to this as *backdoor defense sequela* and consider two attacks, i.e., membership inference attacks [108, 104, 44] and backdoor re-injection attacks on backdoored models defended by different methods.

We hypothesize that the fine-tuned model in the standalone scenario may have higher membership privacy risks due to the fact that during the fine-tuning process, we drive the model to further memorize the fine-tuning dataset and forget the backdoor trigger. Note that in the standalone setting, the fine-tuning dataset is a clean version of the model’s original training dataset. We conduct our experiments by leveraging existing membership inference attacks on both backdoored models and fine-tuned models. Surprisingly, our experimental results show that, after super-fine-tuning, the membership leakage risks are even reduced. For instance, after a BadNets model on CIFAR10 has been defended by super-fine-tuning, the membership inference attack can achieve 0.569 accuracy, which is lower than the performance on the original backdoored model (0.618). Therefore, from a privacy leakage perspective, fine-tuning has almost no negative impact on the target model.

We also consider another backdoor sequela called backdoor re-injection attacks. As fine-tuning only takes a few steps to mitigate the backdoor attacks, it is worth further exploring whether the backdoor can be easily injected back. Our experimental results show that for any defense (both ours and methods from previous works), once it has been applied to a model, the adversary can easily re-inject the same backdoor to the target model. For instance, once a BadNets model on CIFAR10 in the standalone scenario has been fine-tuned (ASR drops from 0.954 to 0.073), the adversary can re-inject BadNets to the model to achieve a similar ASR (0.935) within two epochs when the poison ratio is 0.1. To achieve a similar backdoor attack performance on the original clean model, BadNets requires seven epochs.

**Implications:** In general, our results show that backdoor defenses can be performed more easily than previously thought. All one needs is fine-tuning or super-fine-tuning. Currently, the empirical evaluation suggests that backdoor attacks achieve almost perfect accuracy ( $\sim 100\%$  accuracy [90, 39, 21, 73, 51]), especially for standalone classifiers. By applying our easy-to-deploy fine-tuning defense, our work will certainly help the model users/owners mitigate existing backdoor attacks deployed in the real world. It further calls for the design of more advanced backdoor attacks to better assess the vulnerability of ML models to such attacks.

## 3.2 Backdoor Defenses

In this section, we first introduce the defender’s goals and capabilities. Then, we will discuss how fine-tuning and super-fine-tuning work to mitigate backdoor attacks.

### 3.2.1 Defender’s Goals and Capabilities

**Defender’s Goals:** A defender’s goal can be summarized from three perspectives.

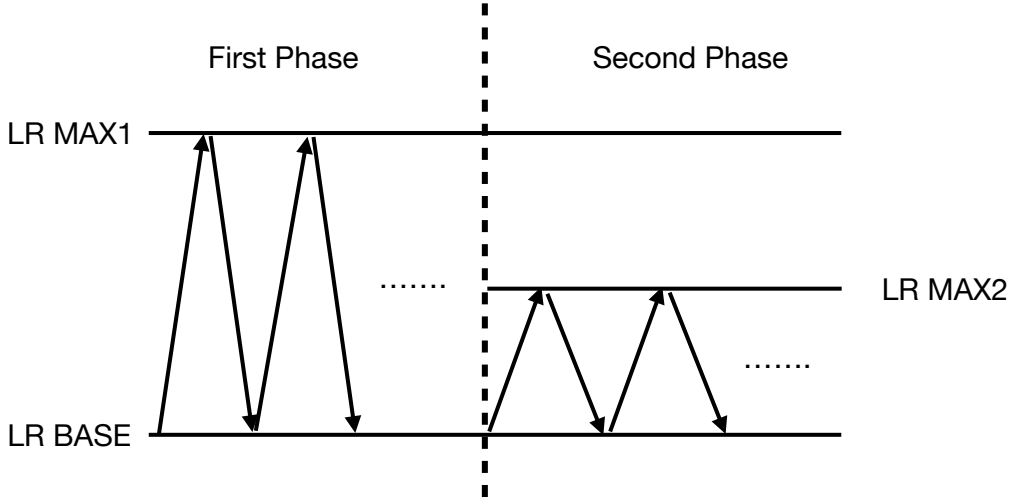
- **Backdoor Performance:** The main goal of the defender is to reduce the backdoor performance. To achieve this goal, the defender can either detect/mitigate the triggered inputs or purify the model to mitigate the backdoor effect.
- **Utility:** In addition to reducing the backdoor performance, the defender should also keep the utility of the backdoored model. That means that, after the defense, the model should still perform well on clean inputs.
- **Computational Cost:** As ML models become increasingly complex, training and testing models both require one to have powerful computing infrastructures. Ideally, the defender should use minimal computing resources to mitigate backdoors.

**Defender’s Capabilities:** The defender is supposed to have a clean dataset to conduct the backdoor defense. For the encoder-based and transfer-based scenarios, this assumption is straightforward. The user (who is also the defender) is the one who fine-tunes the model for their downstream tasks, and they should have the clean dataset already. For the standalone scenario, as mentioned before, the model’s training dataset is provided or can be obtained by the user. Moreover, in all the scenarios, we assume the defender has white-box access to the model, which means that they can access and modify the model’s parameters. Also, as we have stated before, the defender only has limited computational resources.

### 3.2.2 Fine-Tuning to Mitigate Backdoor Attacks

In this section, we describe how conventional fine-tuning works and then propose our super-fine-tuning method.

**Conventional Fine-Tuning:** Fine-tuning is a strategy originally proposed in the context of transfer learning. The motivation behind existing fine-tuning is to enable the pre-trained model to fit new data samples using information learned from the pre-training phase. In our case, fine-tuning is supposed to mitigate backdoor attacks as well as leverage the pre-trained model information. Instead of only fine-tuning a few layers like previous works [51], we adopt whole model fine-tuning in all our scenarios.



**Figure 3.1:** The learning rate scheduler of super-fine-tuning.

During the fine-tuning process, we rely on the same learning rate as the one used in the pre-training process.

In the encoder-based scenario, conventional fine-tuning means that the user conducts the whole model fine-tuning, which is recommended by various existing works [54, 18, 115]. Our experimental results show that conventional fine-tuning can effectively mitigate backdoor attacks in the encoder-based scenario, but it does not always work in the transfer-based and standalone scenarios.

**Super-Fine-Tuning:** We further propose a super-fine-tuning strategy, a novel fine-tuning approach focusing on removing backdoor attacks. Super-fine-tuning is inspired by super-convergence[109], which shows that the regular changes in learning rate can contribute to fast learning. The main innovation of super-fine-tuning is the scheduler of the learning rate. Normally, the gradient descent process can be formulated as  $x = x - \epsilon \nabla_x f(x)$  where  $x$  represents the weights of the model,  $\epsilon$  represents the learning rate, and  $f(\cdot)$  represents the loss function. To make the model forget backdoor triggers while keeping the utility, we make  $\epsilon$  change according to the schedule shown in Figure 3.1. The intuition behind our designed function is that large learning rates tend to make the model forget backdoor triggers while small learning rates maintain the model utility (see Section 3.4.6 for detailed information). Therefore, we combine the two different learning rates with the scheduler.

Concretely, we first pre-define a base learning rate (LR BASE) and two maximum learning rates (LR MAX1 and LR MAX2) for the scheduler of super-fine-tuning. Note that LR MAX1 is required to be larger than LR MAX2. We separate the training process into two phases. In the first phase, we make the learning rate linearly increase from LR BASE to LR MAX1 in several iterations and then drop back to LR BASE. This way, a learning rate that is close to LR MAX1 is supposed to force the model to forget backdoor triggers quickly, while the learning rate that is close to LR BASE

will keep the model utility on clean samples. The same process should be repeated until we lower the maximum learning rate after a pre-defined number of epochs (in our experiments, we find that ten epochs work well). In the second phase, we continue oscillating between the base learning rate and LR MAX2 for the remaining epochs, mitigating the overfitting level of the model. Our experimental results show that the above process can effectively mitigate backdoor attacks while retaining the model’s utility.

### 3.3 Experimental Setup

#### 3.3.1 Current Attacks and Defenses

For our evaluation, we consider the following six attacks and six defenses.

- **BadNets [39]**. BadNets is the most representative and classic backdoor attack against ML models. The key intuition behind BadNets is to add a visible trigger to some part of the training images and label them with a target class. When the model is trained on this poisoned dataset, the backdoor will be injected, and any inputs with the same trigger will be misclassified into the target class.
- **Blended [21]**. The Blended backdoor attack is another well-established backdoor attack. Different from BadNets, Blended aims at creating a trigger that is difficult to detect even by human eyes. Also, the position of the trigger does not affect the recognition of the backdoor.
- **LF [137]**. The Low Frequency (LF) backdoor aims to design backdoor attacks from a frequency perspective. Previous works’ trigger images are significantly different from clean images in the frequency domain. LF aims to make the triggered sample and clean sample consistent in terms of frequency. In this way, backdoor triggers have better concealment in the frequency domain.
- **Inputaware [83]**. Inputaware argues that uniform trigger patterns will be easy to detect by simple pattern recognition methods. Therefore, this attack aims to design a generator driven by diversity loss to generate personalized triggers. Through this generator, the triggers for different images are different.
- **WaNet [84]**. WaNet also focuses on designing undetectable backdoor attacks. WaNet uses small and smooth deformation technology to generate undetectable trigger samples.
- **BadEncoder [51]**. Different from previous attacks, BadEncoder conducts backdoor attacks on the encoders (e.g., encoders established by self-supervised learning). It injects backdoors into encoders and then expects the corresponding downstream classifiers to have good backdoor performance as well as high utility.

BadEncoder is designed specifically for the encoder-based scenario, while the other five attacks can be applied to both transfer-based and standalone scenarios. Note that although there are also other similar backdoor attacks on encoders [69, 11], they share

a similar poisoning spirit. Therefore, we only take advantage of BadEncoder for the evaluation.

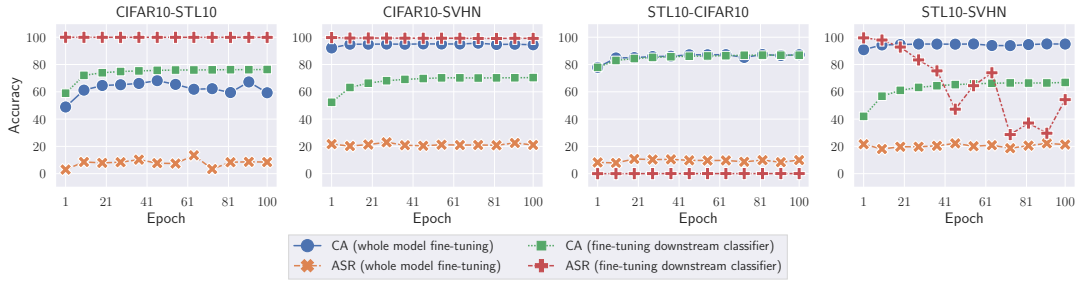
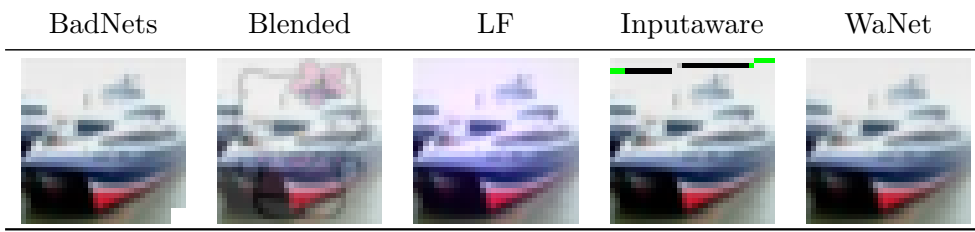
**Defenses:** Besides fine-tuning and super-fine-tuning, we also evaluate the following six state-of-the-art defense methods.

- **ABL [64]**. Anti-Backdoor Learning (ABL) aims to train the clean model on the poisoned dataset. The intuition of ABL is that a model tends to remember backdoor samples fast, and backdoor samples are tied to specific classes. ABL designs a two-stage gradient ascent method to isolate backdoor samples and makes the relationship between the backdoor sample and the corresponding label invalid. In this way, ABL can successfully counter backdoor attacks.
- **AC [14]**. The intuition behind Activation Clustering (AC) is that clean samples and backdoor samples will activate different parameters in neural networks. AC finds the backdoor samples by traversing the parameters of each activation and comparing their distributions.
- **FP [71]**. Fine-pruning (FP) is a defense method similar to fine-tuning. However, fine-pruning argues that only conducting fine-tuning cannot effectively mitigate the backdoor attack. Therefore, besides fine-tuning, the method will prune the neural network to eliminate the low influential neurons in order to remove the backdoor in the model. Later we show that proper fine-tuning is sufficient to mitigate backdoor attacks (see Section 3.4.1).
- **NAD [65]**. Neural Attention Distillation (NAD) also argues that simply fine-tuning is not enough to mitigate the backdoor attack. They use knowledge distillation with the clean teacher model to guide the fine-tuning process of the backdoored student model. In this way, the backdoor can be removed but the computational cost is high.
- **NC [121]**. Neural Cleanse (NC) is a classic backdoor detection and removal method. NC optimizes potential triggers in each class and then compares each class' minimum perturbation to find the out-of-distribution classes. If this class exists, the model is backdoored, and this class is the target class. To mitigate backdoor attacks, NC conducts unlearning by fine-tuning the model using images with triggers and correct samples.
- **Spectral [118]**. Spectral signatures detection aims to remove triggered samples by detecting the spectrum of the covariance of a feature representation learned by the neural network. Then, the spectral approach will retrain the model with the remaining clean data.

We use BackdoorBench<sup>1</sup> to implement these attacks and defenses. Also, all the experiments are conducted on an NVIDIA DGX-A100 server. We show triggered examples of different attacks in Table 3.1. For each attack, we set the poison ratio to 0.1.

<sup>1</sup><https://github.com/SCLBD/BackdoorBench>.

**Table 3.1:** Examples of triggered inputs from different backdoor attacks.

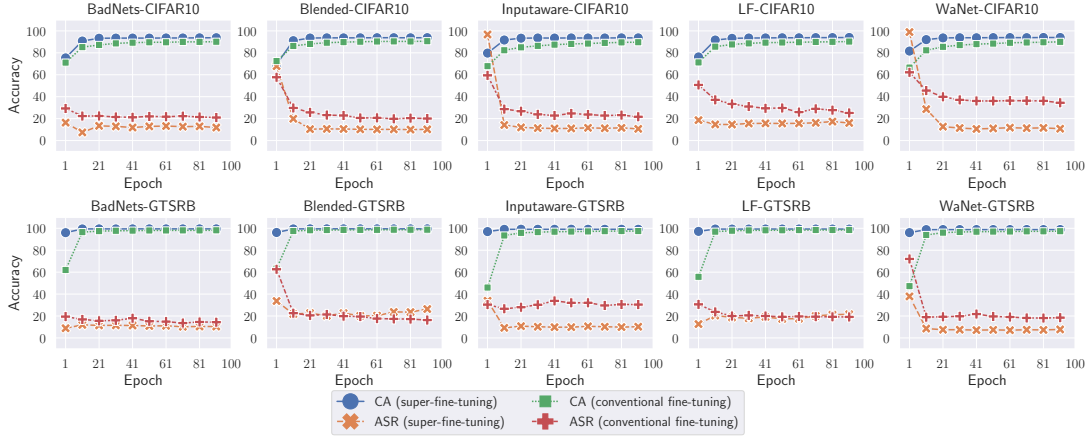


**Figure 3.2:** The performance of whole model fine-tuning and downstream classifier fine-tuning on BadEncoder. The X-axis represents training epochs. The Y-axis represents accuracy. \*

### 3.3.2 Datasets and Evaluation Metrics

We leverage five image datasets for our evaluation: CIFAR10 [1], CIFAR100 [1], STL10 [2], GTSRB [3], and SVHN [4] to measure the effectiveness of fine-tuning and super-fine-tuning. To evaluate whether backdoor attacks have been successfully mitigated, we adopt three evaluation metrics following the three goals of the defender described in Section 3.2.

- **Attack Success Rate.** Attack success rate (ASR) is used to measure whether backdoor samples are successfully classified into the target label or not.
- **Clean Accuracy.** Clean Accuracy (CA) is used to evaluate whether a model can perform well with clean data.
- **Computational Cost.** As we have stated before, when the users take advantage of a third party’s pre-trained models, normally, they do not have sufficient computational resources. Therefore, the backdoor defense methods should use as little computational resources as possible. Here, we leverage computational cost (measured by GPU hours) as another new important metric to evaluate defense methods.



**Figure 3.3:** The performance of conventional fine-tuning and super-fine-tuning against different attacks in the transfer-based scenario. The X-axis represents training epochs. The Y-axis represents the accuracy.

## 3.4 Evaluation Results

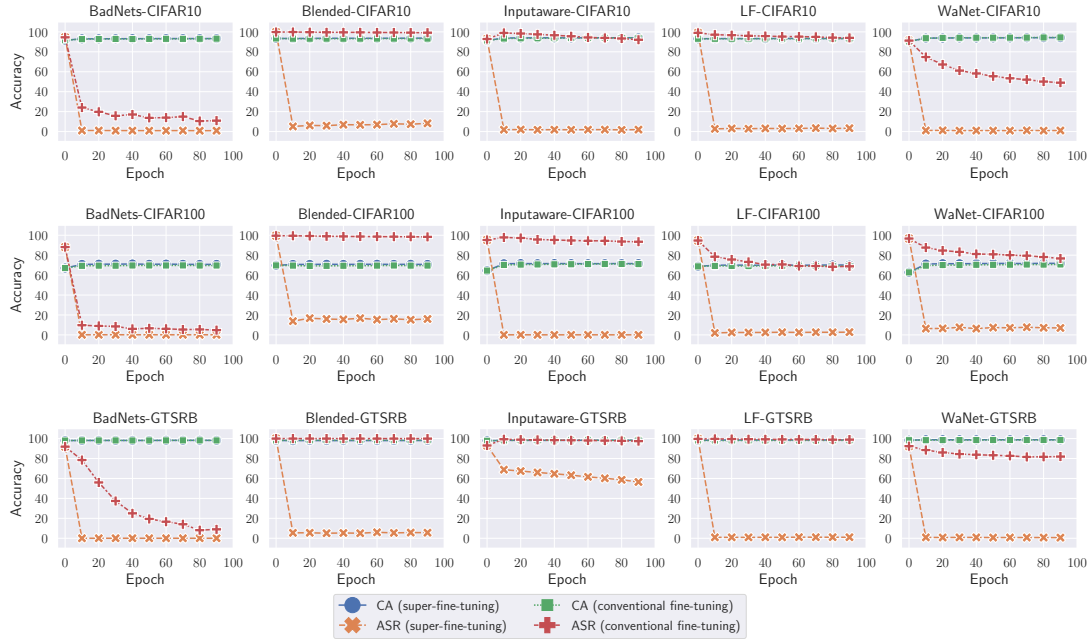
### 3.4.1 Encoder-Based Scenario

In the encoder-based scenario, we make use of BadEncoder as backdoor attack since it is the most representative backdoor attack in this setting. The workflow of BadEncoder is to train a backdoored encoder, freeze the encoder, and use the clean data to train a classifier for the downstream task. However, according to previous works [18, 54, 114], fine-tuning the whole model can achieve better performance than only fine-tuning the downstream classifier. Therefore, our fine-tuning method updates the parameters of the whole model.

The experimental results are shown in Figure 3.2. We train the encoders on CIFAR10 and STL10. Then, we choose CIFAR10, STL10, and SVHN as downstream tasks. From Figure 3.2, we first observe that BadEncoder is not stable in all datasets. For instance, when the encoder is pre-trained on STL10 and then fine-tuned with CIFAR10, even only fine-tuning the downstream classifier makes the ASR drop to 0.002. Second and more importantly, with whole model conventional fine-tuning, the injected backdoor can always be removed immediately, e.g., within one epoch. For instance, for the encoder pre-trained on CIFAR10 with STL10 as downstream task (shown in Figure 3.2), when conducting whole model fine-tuning, the ASR drops from 0.998 (fine-tuning downstream classifiers) to 0.127 within one epoch. Note that, in this scenario, whole model conventional fine-tuning is a natural step to achieve better performance on downstream tasks. Therefore, it has **zero-cost** for mitigating backdoor attacks.

### 3.4.2 Transfer-Based Scenario

The transfer-based scenario is also one of the most common machine learning deployment settings. In this scenario, users obtain the model trained on the large dataset and then fine-tune the model on their own dataset to perform the downstream task. We conduct



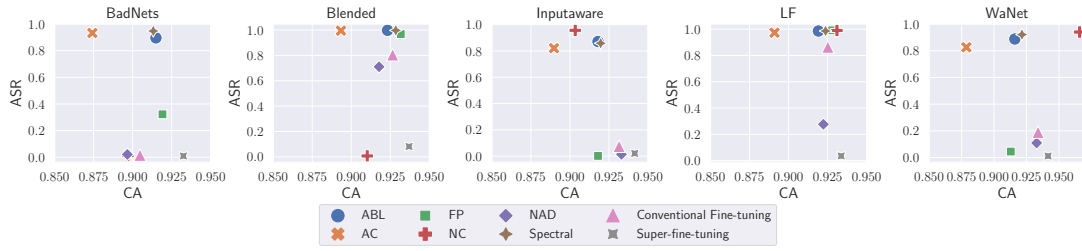
**Figure 3.4:** Accuracy of conventional fine-tuning and super-fine-tuning on backdoor samples and clean samples in the standalone scenario. The X-axis represents training epochs. The Y-axis represents the accuracy. Epoch 0 is the original backdoor ASR and CA before fine-tuning or super-fine-tuning.

experiments where the backdoored models are pre-trained on CIFAR100 and fine-tuned with CIFAR10 and GTSRB. Here, we adopt five different attack methods described in Section 3.3.1. As we have stated in Section 2.1.2, to verify whether a backdoor has been removed, we leverage the original triggers and test whether images with such triggers can be misclassified to a certain class.

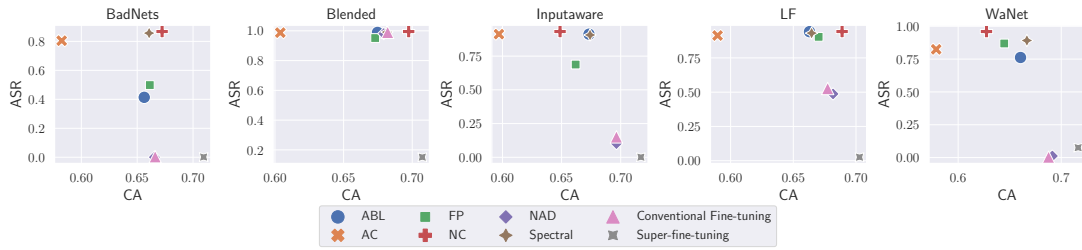
The results are shown in Figure 3.3. As we can see, in this transfer-based scenario, conventional fine-tuning can effectively mitigate backdoor attacks in most cases. For instance, when the defender conducts fine-tuning to the model backdoored by BadNets on CIFAR10, the attack can only achieve 0.378 ASR in one epoch and the ASR will remain around 0.2 after 20 epochs. Our proposed super-fine-tuning method can achieve even better performance than conventional fine-tuning in this scenario. As shown in Figure 3.3, in most cases, super-fine-tuning can achieve lower ASR with less epochs. On BadNets-GTSRB, even after the first epoch, ASR will drop to 0.088. Also, it can be seen that super-fine-tuning yields better CA than conventional fine-tuning. For instance, super-fine-tuning on CIFAR10 against Inputaware attacks can achieve 0.798 CA in the first epoch and 0.937 CA after 100 epochs, both higher than conventional fine-tuning (0.678 in the first epoch and 0.898 after 100 epochs). This finding demonstrates that our proposed super-fine-tuning outperforms conventional fine-tuning in this scenario.



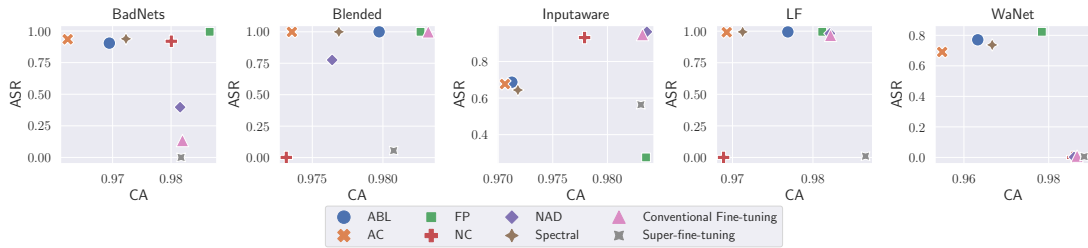
### 3.4. EVALUATION RESULTS



**Figure 3.5:** Comparison between existing state-of-the-art backdoor defenses and super-fine-tuning on CIFAR10. The X-axis represents accuracy on clean samples. The Y-axis represents the attack success rate. Points closer to the lower right corner indicate better defense performance.



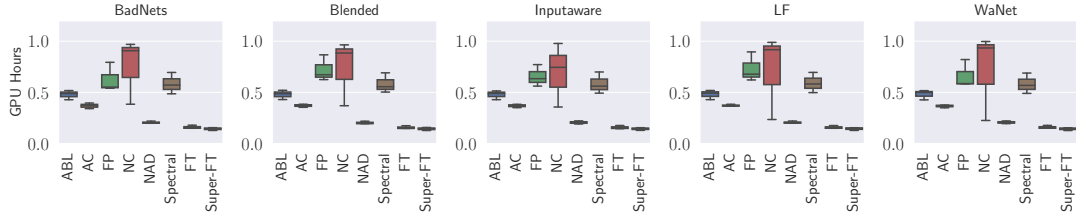
**Figure 3.6:** Comparison between existing state-of-the-art backdoor defense methods and super-fine-tuning on CIFAR100. The X-axis represents accuracy on clean samples. The Y-axis represents the attack success rate. Points closer to the lower right corner are better points.



**Figure 3.7:** Comparison between existing state-of-the-art backdoor defense methods and super-fine-tuning on GTSRB. The X-axis represents accuracy on clean samples. The Y-axis represents the attack success rate. Points closer to the lower right corner are better points.

#### 3.4.3 Standalone Scenario

The standalone scenario is the most difficult scenario to mitigate backdoor attacks. Here, a user directly interacts with the model without any modification. Similar to the transfer-based scenario, we adopt the five attacks in Section 3.3.1. Fine-tuning is no longer a necessary step. Also, due to the fact that the model is trained on the desired dataset, it increases the difficulty of mitigating backdoor attacks. Most previous works [71, 39, 21] that claim backdoor attacks cannot be easily mitigated by fine-tuning



**Figure 3.8:** The time cost of different defense methods. The X-axis represents different methods. The Y-axis represents the GPU hours required for this method. Note that in each box, we include the time cost on all datasets.

are conducted in this scenario.

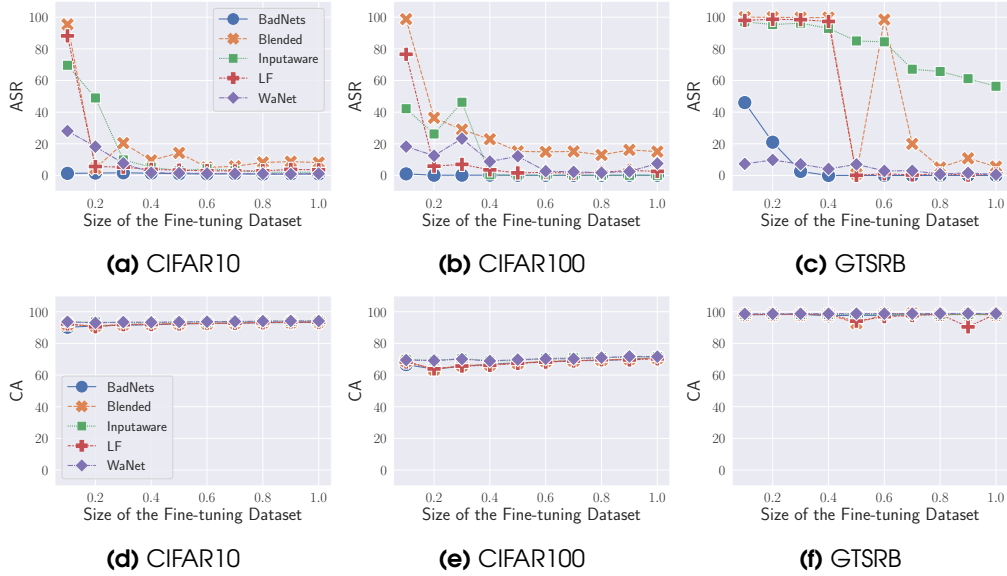
As shown in Figure 3.4, conventional fine-tuning indeed performs poorly in mitigating backdoor attacks in this case. For instance, when conducting conventional fine-tuning on the model backdoored by Blended attacks on CIFAR10, the ASR still remains high (0.978) even after 100 epochs. However, among all five attacks we have studied, super-fine-tuning always decreases the ASR significantly while keeping high clean accuracy. For instance, on CIFAR10, super-fine-tuning can decrease the ASR of Blended backdoor from 0.998 to 0.081, which is in line with the predicted probability of the clean sample. We can also conclude from Figure 3.4 that super-fine-tuning maintains the model’s utility to a large extent. In most cases, the utility does not even drop after the first epoch.

In general, we empirically demonstrate that, with super-fine-tuning, we can effectively mitigate the backdoor attacks while keeping the model utility with a limited number of epochs. Later in Section 3.4.6, we will dive into the details of how the learning rate modification affects the ASR and CA.

### 3.4.4 Comparison to Other Defense Methods

Previously, we have shown that super-fine-tuning can effectively mitigate backdoor attacks with limited computational resources. In this section, we compare super-fine-tuning with other existing state-of-the-art defense methods to show that super-fine-tuning is the most effective and efficient one. Note that here we only focus on the standalone scenario since fine-tuning is a necessary step in the other two scenarios, which means fine-tuning as a defense is zero-cost. Also, fine-tuning or super-fine-tuning can decrease the ASR to a large extent while maintaining the model’s utility.

The results on CIFAR10 are shown in Figure 3.5. We also show the results on CIFAR100 and GTSRB in Figure 3.6 and Figure 3.7 in the appendix. Note that in Figure 3.5, the X-axis is CA and the Y-axis is ASR. Therefore, in each sub-figure, the closer to the lower right corner, the better the defense performance. Among all defense methods against different attacks, super-fine-tuning, in general, achieves the lowest ASR while maintaining the highest CA. For instance, to mitigate the BadNets attack on CIFAR10, super-fine-tuning can achieve 0.932 CA with only 0.009 ASR, which constitutes the best performance among all defense methods. We can also see that other defense methods cannot always guarantee performance in defending against all attacks. For instance, although only NC and super-fine-tuning can mitigate Blended attacks on



**Figure 3.9:** The impact of fine-tuning dataset size on defense performance. The first row shows fine-tuning dataset size’s impacts on attack success rate. The second row shows fine-tuning dataset size’s impacts on clean samples accuracy. The X-axis represents the ratio of the fine-tuning dataset, which is used to conduct fine-tuning.

CIFAR10, NC cannot detect Inputaware, LF, and WaNet attacks.

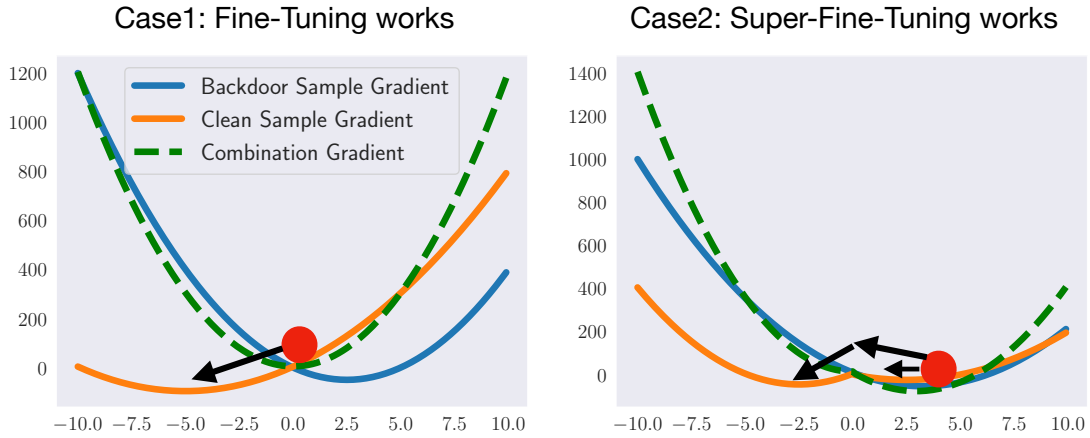
We then consider another important aspect, i.e., each defense’s computational cost. The results are shown in Figure 3.8. We can observe that, among all defenses against different attacks, NC has the largest computational cost, while super-fine-tuning has the lowest computational cost. For instance, to detect and remove BadNets on CIFAR100, NC takes 0.997 GPU hours, while super-fine-tuning only needs 0.147 GPU hours, which is significantly lower.

In general, we conclude that super-fine-tuning outperforms other defenses in terms of the lowest ASR, highest CA, and lowest computational cost.

### 3.4.5 Illustration of Why Fine-Tuning and Super-Fine-Tuning Work

We show the gradient change process in Figure 3.10. In the figure, the orange line represents the gradient change process of clean samples. The blue line represents the training gradient process of backdoor samples. The green line denotes the combined training gradient process. To obtain the backdoored model, we need to make the model’s parameters (shown as the red point) drop to the local minimum which can achieve both good clean accuracy and high attack success rate.

To mitigate the backdoor attacks, we further fine-tune the model with clean samples. There are two cases. In the first case (left sub-figure in Figure 3.10), the combination local minimum is not the local minimum of the orange line (gradient of clean samples). In this case, conventional fine-tuning with clean samples will make the red point continue to drop to the clean samples’ local minimum point which is far away from the backdoor samples’ best parameters location. In that case, conventional fine-tuning can already



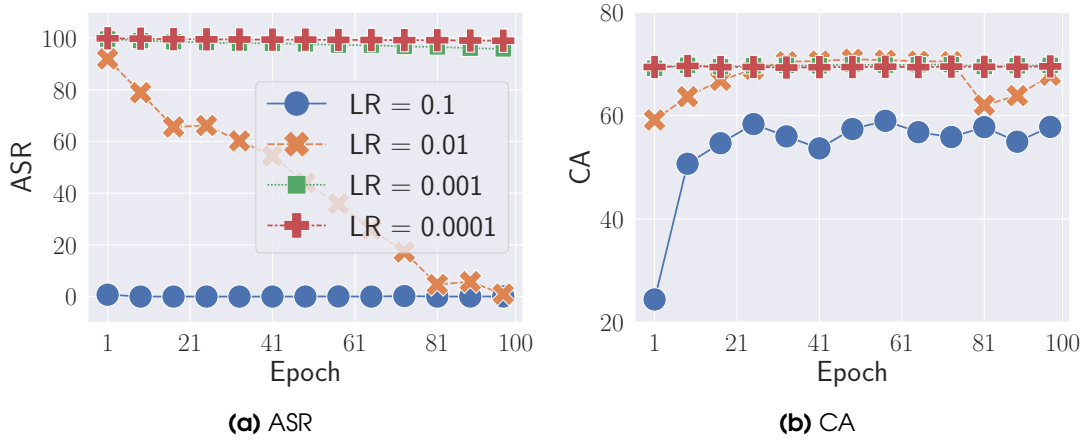
**Figure 3.10:** Schematic diagram of backdoor training and mitigation process.

successfully mitigate backdoor attacks. In the second case, the combination local minimum point is around the local minimum of clean samples. In this case, conventional fine-tuning with a small learning rate cannot help the model to jump over this local minimum. However, with the changed learning rate, super-fine-tuning can better help the model to jump over the local minimum. In that way, the model’s parameters will be optimized to another local minimum which is not a good point for backdoor samples. This demonstrates why super-fine-tuning works better in most cases.

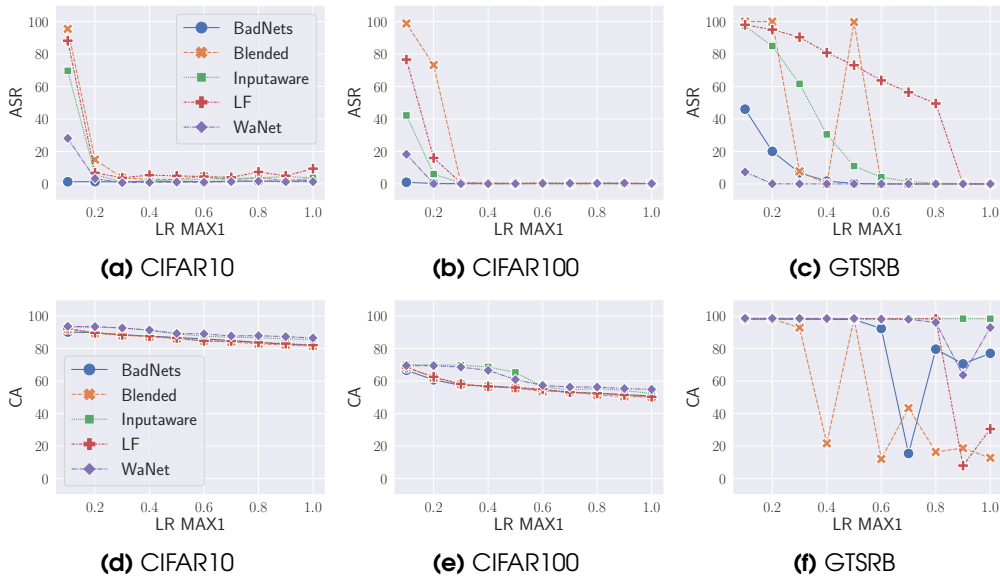
### 3.4.6 Ablation Study

Here, we conduct some ablation studies to show the impact of fine-tuning dataset size and learning rate on the backdoor removal performance. Note that we only focus on the standalone scenario here because: (i) in both encoder-based and transfer-based scenarios, fine-tuning is a necessary step, so we do not modify the fine-tuning dataset size and learning rate; (ii) standalone is the most challenging scenario, as we mentioned before.

**Impact of Fine-Tuning Dataset Size:** We first explore the impact of fine-tuning dataset size. Previously, we used the whole dataset to conduct super-fine-tuning. We have shown that, even with the whole dataset, super-fine-tuning consumes limited computational resources compared to other methods. Then, we further explore how much data is sufficient to conduct a successful super-fine-tuning. We show our experimental results in Figure 3.9. We can see that even with 20% of the fine-tuning dataset, super-fine-tuning can effectively mitigate the backdoor attacks in most cases. For instance, with 20% of the fine-tuning dataset (CIFAR10), super-fine-tuning reduces the ASR of the Blended backdoor attack to 0.044. Also, from Figure 3.9, we can see that the size of the fine-tuning dataset has a limited impact on the utility of the model. The clean accuracy remains high with 10% to 100% of the fine-tuning dataset. Therefore, it can be concluded that super-fine-tuning requires significantly fewer fine-tuning data samples for a stable performance, which further reduces the computational cost.



**Figure 3.11:** The impact of different learning rates of conventional fine-tuning on removing backdoor attacks. The X-axis represents training epochs. The Y-axis represents the accuracy of backdoor samples and clean samples.



**Figure 3.12:** Impact of LR MAX1 of super-fine-tuning on defense performance. The first row shows LR MAX1’s impacts on attack success rate. The second row shows LR MAX1’s impacts on clean sample accuracy. The X-axis represents how many data samples are used to conduct fine-tuning. Note that we only use 10% of the fine-tuning dataset to conduct super-fine-tuning.

Note that super-fine-tuning is less effective with 10% of the fine-tuning dataset. For instance, when only using a 10% clean training dataset and 0.1 as LR MAX1, the defender can only achieve 0.954 ASR with the Blended attack on CIFAR10. We show later that the backdoor attacks can still be effectively mitigated by increasing LR MAX1 if the defender only has 10% of the fine-tuning dataset.

**Impact of Learning Rate Change:** During our experiments, we first find that backdoor attacks are very sensitive to different learning rates. We show different learning rates' results of conventional fine-tuning in Figure 3.11. It can be seen that if the defender uses the same small learning rate as in the pre-training phase, the ASR remains high even after 100 epochs. However, with an increased learning rate (from 0.0001 to 0.001), backdoor triggers are forgotten gradually in 100 epochs. Moreover, when the learning rate increases to 0.1, the backdoor can be immediately removed within one epoch. We can conclude that learning rates have a significant impact on backdoor removal. In particular, larger learning rates tend to mitigate backdoor attacks faster.

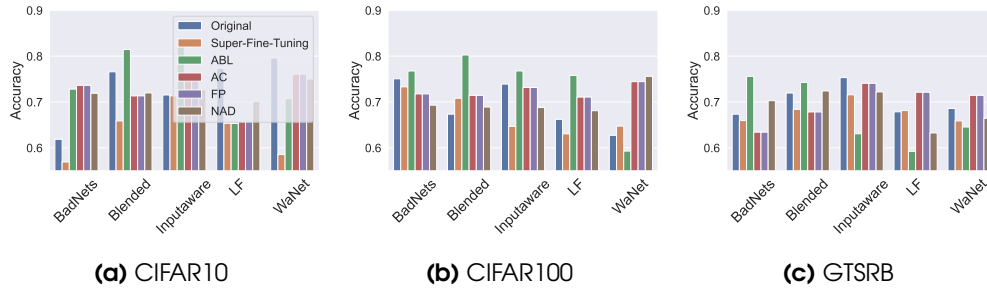
From Figure 3.11, we can also find that although increasing learning rates can effectively mitigate backdoor attacks, it also causes utility drops. From the utility perspective, small learning rates lead to higher clean accuracy. Therefore, combining large and small learning rates becomes a promising idea to achieve both goals. This is also the general intuition for super-fine-tuning.

In our previous super-fine-tuning experiments, we set LR MAX1 to 0.1 as we find that 0.1 is enough for removing backdoors with sufficient fine-tuning datasets. Here, we also explore the impact of LR MAX1 on super-fine-tuning. To better show the learning rate's impact, we only use 10% of the fine-tuning dataset. In the previous section, when the defender only has 10% of the fine-tuning dataset, super-fine-tuning does not achieve good performance, especially in Blended and LF attacks.

Our experimental results are shown in Figure 3.12. When increasing the LR MAX1 from 0.1 to 0.3, even when using 10% of the fine-tuning dataset, super-fine-tuning can still successfully remove backdoors. For instance, when the LR MAX1 is 0.1, Blended attacks on CIFAR100 still achieve 0.988 ASR under super-fine-tuning. However, when LR MAX1 increases to 0.3, the ASR drops to 0.004. Although increasing LR MAX1 can more effectively remove the backdoors, it also leads to a small drop in the model's utility. We show these results in Figure 3.12. For instance, when the learning rate increases from 0.1 to 0.3, the utility of the fine-tuned model from Blended on CIFAR10 drops from 0.919 to 0.881. Therefore, if users have enough clean data, we recommend using 0.1 as the largest learning rate. However, when users only have limited data, increasing the largest learning rate (e.g., from 0.1 to 0.3) also helps mitigate almost all attacks without suffering a large utility drop.

### 3.4.7 Summary

In this section, our empirical study shows that backdoor attacks can be easily defended by fine-tuning or super-fine-tuning. Concretely, we find that in the encoder-based and transfer-based scenarios, fine-tuning as the necessary step can naturally remove the existing backdoors. Also, our proposed super-fine-tuning method can better mitigate the backdoor attacks in the transfer-based scenario. In the standalone scenario, super-fine-tuning can effectively prevent backdoor attacks with a limited size of the training dataset and limited computational resources compared to other existing defenses. Our ablation study on the fine-tuning dataset size and learning rate setting further demonstrates the effectiveness and efficiency of super-fine-tuning.



**Figure 3.13:** The performance of membership inference attacks on different defended models and backdoored models. The X-axis represents different attack methods. The Y-axis represents membership inference attack accuracy.

## 3.5 Backdoor Sequela

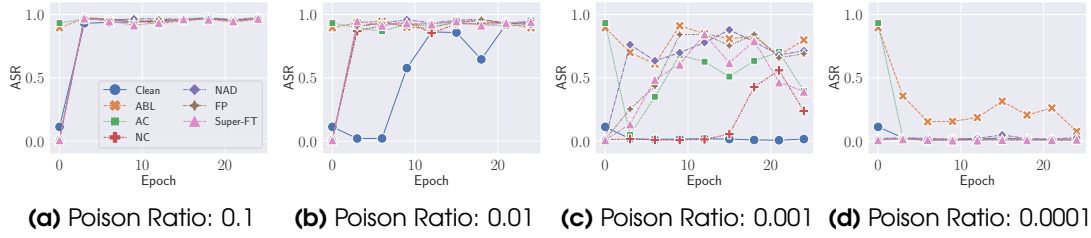
We have previously shown that super-fine-tuning outperforms other defenses against backdoor attacks. However, since the defense modifies the model’s parameters, it is worthwhile to explore whether the model will be more or less vulnerable to certain attacks after removing the backdoor attacks. To this end, we coin the term and investigate backdoor sequela.

Due to the fact that super-fine-tuning needs to use a clean dataset to make the model forget the backdoor, it is natural to wonder whether the process will lead the model to better remember the clean dataset. To verify this, we conduct membership inference attacks against the backdoored models and the models defended by super-fine-tuning (and other defenses) to see whether the membership leakage becomes larger after applying the defense. Also, since the backdoors are easily removed after a few epochs, we are also curious if it is easier to re-inject the backdoor into the model. Note that, following the same reasons as in Section 3.4.4, we only consider the standalone scenario in this section.

### 3.5.1 Membership Inference Attack

We first explore whether the model is more vulnerable to membership inference attacks [108] or not after fine-tuning. Membership inference attacks aim to infer whether a given sample is in the training set of a target model or not. A successful membership inference attack can cause severe privacy leakage. Normally, there are three different ways to conduct membership inference attacks: neural network-based attacks [80, 108], metric-based attacks [60, 111, 112, 132], and query-based attacks [23, 67]. In this work, we use the neural network-based attack due to its popularity.

**Threat Model:** We first assume that the adversary only has black-box access to the target model, which means they can only query the model and obtain the output. Then, following previous works [81, 74], we further assume that the adversary has part of the target model’s training data (treated as members) and testing data (non-members). The adversary can use them for training an attack model and inferring the membership



**Figure 3.14:** Performance of BadNets backdoor re-injection attacks on different defense methods. The X-axis represents training epochs in the re-injection phase. The Y-axis represents the accuracy of poison samples. Note that epoch 0 represents different defense methods’ results before re-injection.

status for other data samples. Note that we adopt the strongest attacker assumption defined in [74] to estimate the worst-case scenario for membership leakage.

**Methodology:** Our method can be described in two steps:

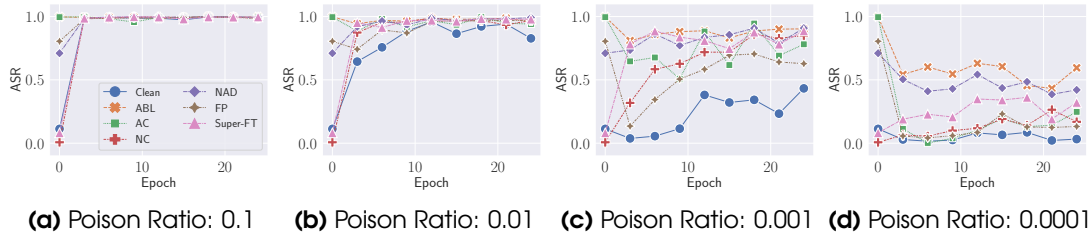
1. The adversary first queries the target model with both the target model’s (partial) training and testing samples, and they label the corresponding outputs as members and non-members.
2. Second, the adversary uses the outputs and the corresponding labels to train their attack model, which is a three-layer neural network model.

The evaluations are conducted on both the backdoored model and the super-fine-tuned model to see whether fine-tuning will increase or decrease membership inference risks.

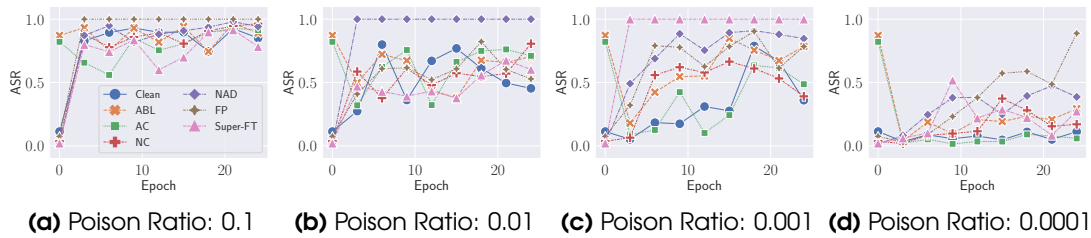
**Experimental Settings:** We evaluate the membership inference attack in the standalone scenario, which means that the fine-tuning dataset is the same as the pre-training dataset. For each dataset, we randomly sample half of its testing samples and the same number of training samples as the attack training dataset. Then, we select the other half of its testing samples (serving as non-members) and the same number of training samples (serving as members with no overlap on the attack training dataset) to evaluate the attack performance. Note that we use the datasets and backdoor attacks/defenses introduced in Section 3.3.1.

**Results:** We show our membership inference results in Figure 3.13. Surprisingly, we observe that instead of increasing the privacy risks, super-fine-tuning mitigates the performance of membership inference. In almost all cases, fine-tuned models have lower attack performance than the original models. For instance, membership inference attacks on the original model (backdoored by BadNets) on CIFAR10 can achieve 0.618 accuracy, while the performance drops to 0.569 after conducting super-fine-tuning on the original model. This is contradictory to previous work [104], where more training epochs led to higher attack performance due to the increasing overfitting level. We suspect the reason is that super-fine-tuning actually increases the generalization ability of the model, which leads to a lower overfitting level.

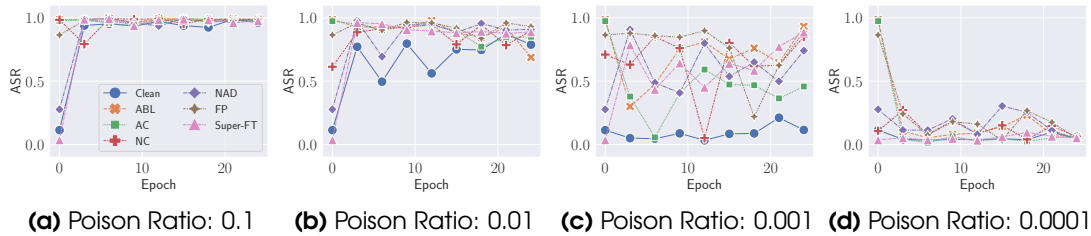




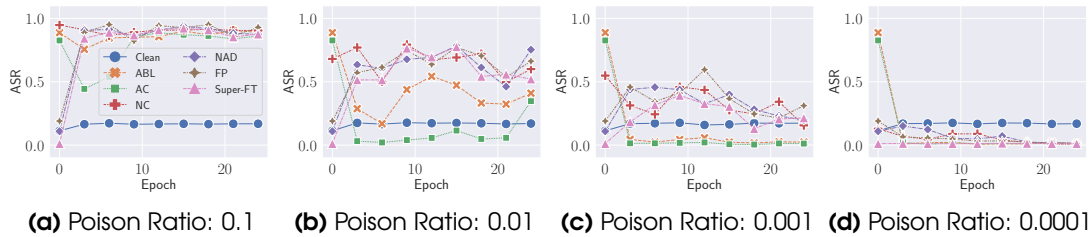
**Figure 3.15:** Performance of Blended backdoor re-injection attacks on different defense methods. The X-axis represents training epochs in the re-injection phase. The Y-axis represents the accuracy of poison samples.



**Figure 3.16:** Performance of Inputaware backdoor re-injection attacks on different defense methods. The X-axis represents training epochs in the re-injection phase. The Y-axis represents the accuracy of poison samples.



**Figure 3.17:** Performance of LF backdoor re-injection attacks on different defense methods. The X-axis represents training epochs in the re-injection phase. The Y-axis represents the accuracy of poison samples.



**Figure 3.18:** Performance of WaNet backdoor re-injection attacks on different defense methods. The X-axis represents training epochs in the re-injection phase. The Y-axis represents the accuracy of poison samples.

Though super-fine-tuning does not cause backdoor sequela with respect to membership inference, we do observe that some of the other defense methods make membership inference more unstable. For instance, in most cases, ABL makes the model more vulnerable to membership inference attacks (accuracy increases from 0.765 to 0.815 in Blended CIFAR10). However, when ABL is used to mitigate LF attacks on CIFAR10, the membership inference risk drops from 0.726 to 0.653.

From Figure 3.13, we can also see that different kinds of backdoor attacks make a huge difference in membership inference performance. For example, the adversary can achieve a 0.765 accuracy when conducting membership inference attacks using CIFAR10 on Blended attacks. However, they can only achieve a 0.618 accuracy when conducting the same attacks on the BadNets model. We leave it to future work to further explore the relationship between backdoor attacks and membership inference attacks.

### 3.5.2 Backdoor Re-injection Attack

Another backdoor sequela we study is the backdoor re-injection attack. Since super-fine-tuning can easily remove backdoors within a few epochs, it is interesting to see whether the fine-tuned models are more vulnerable to injecting (the same) backdoor attacks again. To our knowledge, there is no prior work measuring whether existing backdoor defense methods will make the re-injection process easier.

**Threat Model:** We first assume that the adversary has white-box access to the fine-tuned model. Also, the adversary has knowledge of the previous backdoor attack on the model. The goal of the adversary is to re-inject the same backdoor into the model while keeping model utility.

**Methodology:** To measure the vulnerability of the fine-tuned models on backdoor re-injection attacks, we take the following steps:

1. The adversary first generates new triggered samples based on the knowledge of the previous attack.
2. Then, the adversary re-trains the fine-tuned model with the poison dataset and the corresponding training process to re-inject the backdoor into the fine-tuned model.

**Results:** To measure how easily the adversary can re-inject a backdoor into the model, we consider training epochs, poison ratio, and ASR as the evaluation metrics. If the model is more vulnerable to backdoor re-injection attacks, the adversary only needs fewer epochs and a smaller poison ratio to achieve a similar (or even better) ASR compared to injecting the backdoor on a clean model. Note that in this setting, we assume a very powerful adversary who has knowledge of the previous attack. Thus, the adversary can follow the same procedure to conduct the attack, i.e., they use the exact same injection process and strategies, including the same attack method, the same learning rate, etc.

Figure 3.14 shows our results of BadNets backdoor re-injection attacks. Other attacks’ results, including Blended (Figure 3.15), Inputaware (Figure 3.16), LF (Figure 3.17), and WaNet (Figure 3.18) are shown in the appendix. For the poison ratio, we adjust it from 0.1 to 0.0001 for completeness. Note that 0.1 is the poison ratio we used in previous backdoor attacks. For the training process, we only show the first 25 epochs to see whether the backdoor can be injected within a few epochs.

From Figure 3.14, it can first be observed that almost all defense methods we have tried can make the defended model more vulnerable to backdoor re-injection attacks. For instance, when the attack method is BadNets with a poison ratio of 0.01 (Figure 3.14b), the adversary needs 15 epochs to insert the backdoor into the clean model (ASR=0.819) while the backdoor re-injection attacks against defended models only need 3-6 epochs to achieve even higher ASR. We also see that, in all cases, fine-tuned models are not the most vulnerable models to backdoor re-injection attacks. For instance, when the poison ratio is 0.001, the adversary can achieve 0.023 ASR in three epochs on models defended by super-fine-tuning, while other defense methods like ABL increases the ASR to 0.763.

### 3.5.3 Summary

In this section, we propose the new term, *backdoor sequela*, to measure how backdoor defense methods affect a model’s vulnerability to other attacks. Specifically, we consider membership inference attacks and backdoor re-injection attacks. Our evaluation results show that super-fine-tuning can even make the model more robust to membership inference attacks. We also find that, in general, existing defense methods considered in our experiments make the defended models more vulnerable to backdoor re-injection attacks compared to the attacks on the clean model. To our knowledge, we are the first to study backdoor sequela, and we argue that backdoor sequela should be considered as an important metric to evaluate a backdoor defense’s efficacy. We plan to investigate more backdoor sequela, i.e., attacks, in the future.

## 3.6 Conclusion

In this section, we have demonstrated that conventional fine-tuning is a very effective backdoor removal method. Moreover, we propose super-fine-tuning which can have even better mitigation performance. We consider three scenarios, namely encoder-based, transfer-based, and standalone. Our experimental results show that in the encoder-based scenario, whole model conventional fine-tuning can effectively remove backdoors within a few epochs. As fine-tuning is a necessary step for users to train downstream classifiers, it can be argued that fine-tuning as a defense method incurs *zero-cost*. In the transfer-based scenario, fine-tuning is still a necessary step. However, we find that conventional fine-tuning cannot always effectively remove all tried backdoor attacks. However, our experimental results show that super-fine-tuning can effectively mitigate backdoor attacks in this scenario. The most difficult scenario is standalone. In this scenario, we assume the backdoored models are trained exactly on users’ downstream datasets. We show that even using the same dataset to conduct the fine-tuning, super-fine-tuning can still remove backdoor attacks in a few epochs. We also compare super-fine-tuning with

state-of-the-art defense methods and demonstrate that super-fine-tuning outperforms them.

Furthermore, we propose a new term, *backdoor sequela*, to measure the defended model’s vulnerability to other attacks. Experiments show that super-fine-tuning does not have a strong impact on the defended models with respect to membership inference and backdoor re-injection attacks. We hope that, in the future, backdoor sequela will be considered an important aspect for judging a backdoor defense’s efficacy.

Our results demonstrate that backdoor defenses can be performed in an easier way than previously considered. Fine-tuning or super-fine-tuning is sufficient in most cases. We hope our methods can help ML model owners better shield their models from backdoor attacks. Also, it further calls for the design of more advanced attacks in order to comprehensively assess machine learning models’ vulnerabilities to backdoor attacks.

# 4

## Model Stealing Attacks

Model Security



## 4.1 Introduction

Recent years have witnessed the great success of applying deep learning (DL) to computer vision tasks. Supervised DL models, such as image classifiers, rely on large-scale labeled datasets to achieve good performance. Yet, with the increasing diversity of application domains, labeled data in limited domains turns out the bottleneck against performance improvement of supervised models. As a revolutionary breakthrough, representation learning [29, 34, 142] instead targets pre-training powerful encoders that transform unlabeled data samples into rich representations and are oblivious to downstream tasks or human supervision. The pre-trained encoders then serve as feature extractors to facilitate various downstream tasks with improved performance over classifiers.

Behind its powerful representation, it is non-trivial to obtain a state-of-the-art image encoder, which involves a massive amount of data, expensive computation, expert knowledge, and countless failure trials. For instance, SimCLR [18] uses 128 TPU v3 cores to pre-train a ResNet-50 encoder with a batch size of 4096. Therefore, such pre-trained encoders are usually established by big companies. Clarifai<sup>1</sup>, OpenAI<sup>2</sup>, and Cohere<sup>3</sup> provide the embedding API of images and texts for commercial usage.

Demanding requirements make the intellectual properties of image encoders valuable while consequently vulnerable to potential *model stealing attacks*, a cheap way to mimic the well-trained encoder performance while circumventing the demanding requirements. It has been shown in previous works that supervised DL models are susceptible to *model stealing attacks* [117, 120, 13, 88, 50, 57, 124, 107]. In these attacks, the adversary aims to steal the parameters or functionalities of target models with only query access to them. Specifically, the adversary can launch a large number of queries and obtain the corresponding prediction and/or posterior outputs. With the input-output pairs, the adversary can then train a surrogate model. A successful model stealing attack does not only threaten the intellectual property of the target model but also serves as a stepping stone for further attacks such as adversarial examples [8, 37, 12, 92, 116, 126], backdoor attacks [102, 19, 51], and membership inference attacks [108, 80, 81, 104, 100, 111, 46, 45, 49, 70]. So far, model stealing attacks concentrate on the supervised classifiers, i.e., the model responses are prediction posteriors or labels for a specific downstream task. The vulnerability of unsupervised image encoders is unfortunately unexplored.

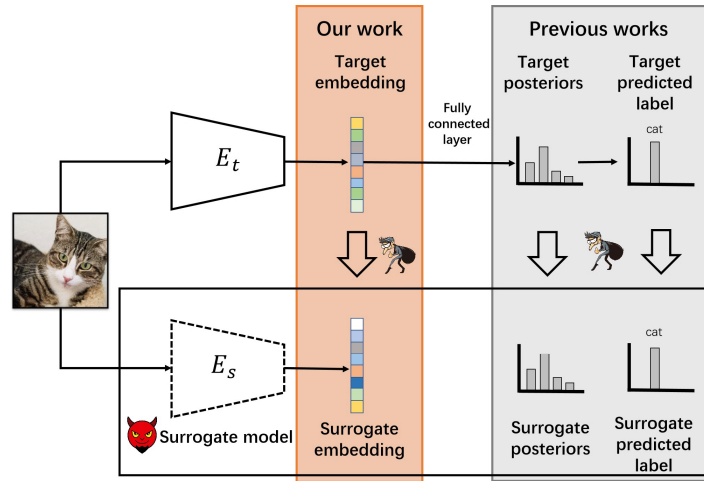
Given that image encoders deliver remarkable performance in various downstream tasks, a successful model stealing attack against them would severely threaten the intellectual properties of model owners to a larger extent. Also, because the responses of encoders are the image representations in higher dimensions and with rich information compared to classifiers' posteriors or labels, it is unclear whether or not encoders are vulnerable to conventional attacks. Moreover, it is also unclear whether or not there exist more effective model stealing attacks against image encoders.

**Our Work:** To fill this gap, we pioneer the systematic investigation of model stealing attacks against image encoders. In this section, the adversary's goal is to steal the

<sup>1</sup><https://www.clarifai.com/models/general-image-embedding>

<sup>2</sup><https://beta.openai.com/docs/api-reference/embeddings>

<sup>3</sup><https://cohere.ai/>



**Figure 4.1:** Model stealing attacks against classifiers (previous) v.s. model stealing attacks against encoders (ours). Previous works aim to steal a whole classifier using the predicted label or posteriors of a target model. In our work, we aim to steal the target encoder using its embeddings. The target encoder ( $E_t$ ) is pre-trained and fixed as shown in the solid frame. The surrogate encoder ( $E_s$ ) is trainable by the adversary as shown in the dashed frame.

functionalities of the target model. See Figure 4.1 for an overview and a comparison with previous works. More specifically, we focus on encoders trained by *contrastive learning*, which is one of the most cutting-edge unsupervised representation learning strategies that unleash the information of unlabeled data.

We first instantiate the conventional stealing attacks against encoders and expose their vulnerability. Given an image input, the target encoder outputs its representation (referred to as embedding). Similar to model stealing attacks against classifiers, we consider the embedding as the “ground truth” label to guide the training procedure of a surrogate encoder on the adversary side. To measure the effectiveness of stealing attacks, we train an extra linear layer for the target and surrogate encoders towards the same downstream classification task. Preferably, the surrogate model should achieve both high classification accuracy and high agreement with the target predictions.

We evaluate our attacks on five datasets against four contrastive learning encoders. Our results demonstrate that the conventional attacks are more effective against encoders than against downstream classifiers. For instance, when we steal the downstream classifier pre-trained by SimCLR on CIFAR10 (with posteriors as its responses) using STL10 as the surrogate dataset, the adversary can only achieve an accuracy of 0.359. The accuracy, however, increases to 0.500 instead when we steal its encoder (with the embedding as its responses).

Also, we observe that the attack against the encoder is less dependent on the dataset used to train the surrogate encoder, i.e., the surrogate dataset. Take the SimCLR pre-trained target encoder on CIFAR10 as an example, compared to the surrogate dataset being CIFAR10, when the surrogate dataset is STL10, the accuracy drops by 0.357 if the target model is a classifier with the predicted label as its response. In



comparison, the accuracy drops by only 0.290 when the target model is an encoder with the embedding as its response. This can be credited to the fact that the robust and generalizable representations in the embeddings benefit the surrogate encoder more in mimicking the functionality of the target encoder.

Despite its encouraging performance, conventional attacks are not the most suitable ones against encoders. This is because they treat each image-embedding pair individually without interacting across pairs. Different embeddings are beneficial to each other as they can serve as anchors to better locate the position of the other embeddings in their space. Contrastive learning [87, 127, 42, 18, 38, 20, 54] is a straightforward idea to achieve this goal. It is formulated to enforce the embeddings of different augmentations of the same images closer, and those of different images further.

In a similar spirit, we propose Cont-Steal, a contrastive-learning-based model stealing attack against the encoder. The goal of Cont-Steal is to enforce the surrogate embedding of an image close to its target embedding (defined as a positive pair), and also push away embeddings of different images irrespective of being generated by the target or the surrogate encoders (defined as negative pairs).

The comprehensive evaluation shows that Cont-Steal outperforms the conventional model stealing attacks to a large extent. For instance, when CIFAR10 is the target dataset, Cont-Steal achieves an accuracy of 0.714 on the SimCLR encoder pretrained on CIFAR10 with surrogate dataset and downstream dataset being STL10, while the conventional attack only achieves 0.457 accuracy. Also, Cont-Steal is more query-efficient and dataset-independent (see Figure 4.21 for more details). For instance, when the target encoder is MoCo pre-trained on CIFAR10, the gap of attack performance between different surrogate datasets is 0.313 with the conventional attacks while only 0.180 with Cont-Steal. This is because Cont-Steal leverages higher-order information across samples to mimic the functionality of target encoder. To mitigate the attacks, we evaluate different defense mechanisms including noise, top- $k$ , rounding, and watermark. Our evaluations show that in most of the cases, these mechanisms cannot effectively defend against Cont-Steal. Among them, top- $k$  can reduce the attack performance to the largest extent. However, it also strongly limits the target model’s utility.

As a takeaway, our attack further exposes the severe vulnerability of pre-trained encoders. We appeal to our community’s attention to the intellectual property protection of representation learning techniques, especially to the defenses against encoder stealing attacks like ours.

**Our Contributions:** In summary, we make the following contributions:

- We pioneer the investigation of the vulnerability of unsupervised image encoders against model stealing attacks. We discover that encoders are more vulnerable than classifiers.
- We propose Cont-Steal, the first contrastive learning-based stealing attack against encoders that outperforms the conventional attacks to a large extent.
- Extensive evaluation shows that the advantageous performance of Cont-Steal is consistently amplified in various settings, especially when the adversary suffers

from zero information of the target dataset, limited amount of data, or restricted query budgets.

## 4.2 Threat Model

in this section, for the encoder pre-trained with images, we consider image classification as the downstream task. We refer to the encoder as the target encoder. Then we treat both the encoder and the linear layer trained for the downstream task together as the target model. We first introduce the adversary’s goal and then characterize different background knowledge that the adversary might have.

**Adversary’s Goal:** Following previous work [50, 57, 107], we taxonomize the adversary’s goal into two dimensions, i.e., theft and reconnaissance. The theft adversary aims to build a surrogate encoder that has similar performance on the downstream tasks as the target encoder. By doing this, the adversary can compromise the intellectual property of the model owner as the target encoder may require dedicated model designs and a massive amount of data as well as computation resources [18, 42, 38, 20]. Different from the thief adversary, the goal of the reconnaissance adversary is to construct a surrogate encoder that behaves similarly to the target encoder. In other words, with any input, the outputs from the target and surrogate encoder used for the same downstream task should have a high agreement. In this case, the surrogate encoder not only faithfully “copies” the behaviors of the target encoder, but also serves as a stepping stone to conduct other attacks. For instance, it can be used to infer sensitive information about the training data [117, 108, 104] or be used to craft adversarial examples [91] or conduct backdoor attacks [51] without taking the risks of being detected by querying the target model.

**Adversary’s Background Knowledge:** We categorize the adversary’s background knowledge into two dimensions, i.e., the knowledge of the target encoder, and the distribution of the surrogate dataset.

Regarding knowledge of the target encoder, we assume that the adversary only has black-box access to it, which means that they can only query the target encoder with an input image and obtain the corresponding output, i.e., the embedding of the input image.

Regarding the surrogate dataset that is used to train the surrogate encoder, we consider two cases. First, we assume the adversary has the same training dataset as the target encoder. However, such an assumption may be hard to achieve as such datasets are usually private and protected by the model owner. In a more extreme case, we assume that the adversary has totally no information about the target encoder’s training dataset, which means that they can only use a different distribution dataset to conduct the model stealing attacks. We later show that the adversary can still launch effective model stealing attacks against the target encoder given a surrogate dataset that is distributed differently compared to the target dataset.

For the model architecture that is used to train the surrogate encoder, we consider two cases. First, we assume the adversary is aware of the target encoder’s architecture

and can train a same architecture shadow encoder. Then we relax our assumption that the adversary uses different architectures to train the surrogate encoder. Our evaluation shows that the choice of architecture does not have much impact on the attack performance (see Table 4.2), which makes the attack more realistic.

Note that we also compare our attacks against the encoders to the traditional model stealing attacks that focus on the whole classifier (which has an encoder and a linear layer). If the attack is targeting a whole classifier, we assume the adversary may obtain the posteriors or the predicted label for an input image.

### 4.3 Model Stealing Attacks

In this section, we first describe the conventional attacks against the classifiers and how to conduct such attacks against the encoders. Then, we propose a novel contrastive stealing framework Cont-Steal to steal the encoders more effectively.

#### 4.3.1 Conventional Attacks Against Classifiers

The basic idea of model stealing is to use the target classifier’s output as ground truth to guide the training procedure of the surrogate classifier. The adversary takes two steps to conduct the model stealing attacks against the target classifier.

**Obtain the Surrogate Dataset:** To conduct model stealing attacks, the adversary first needs to obtain a surrogate dataset. Based on the knowledge of the target classifier’s training dataset (target dataset), we consider two cases. If the adversary has full knowledge of the target dataset, they can directly leverage the target dataset itself as the surrogate dataset. Or the adversary has no knowledge of the target dataset, which means that they can only construct the surrogate dataset which is distributed differently from the target dataset.

**Train the Surrogate Classifier:** To train the surrogate classifier, the adversary can first query the target classifier with the surrogate dataset, then leverage the responses from the target model as the guidance to train the surrogate classifier. Concretely, the loss function  $L_{MS}$  of model stealing can be defined as follows:

$$L_{MS} = \sum_{k=1}^N l(M_T(x_k), M_S(x_k)) \quad (4.1)$$

where  $M_T(\cdot)/M_S(\cdot)$  denotes the target/surrogate classifier and  $N$  denotes the total number of samples on the surrogate dataset. If the response of the target classifier is the predicted label (posteriors), we leverage Cross-Entropy (MSE) as the loss following previous work [91, 88, 57, 50].

#### 4.3.2 Conventional Attacks Against Encoders

The adversary takes two steps to conduct the model stealing attacks against the target encoder and one step for further evaluation.

**Obtain the Surrogate Dataset:** The adversary first constructs their surrogate dataset based on their knowledge of the target dataset, which is the same as Section 4.3.1.

**Train the Surrogate Encoder:** Slightly different from the classifier, the response of the encoder is an embedding, which is a feature vector. In this case, the adversary can still leverage a similar loss function to optimize the surrogate encoder, which can be defined as follows:

$$L_{MS} = \sum_{k=1}^N l(h_T(x_k), h_S(x_k)) \quad (4.2)$$

where  $h_T(\cdot)/h_S(\cdot)$  is the target/surrogate encoder,  $N$  is the total number of samples on the surrogate dataset, and  $l(\cdot)$  is the MSE loss.

**Apply the Surrogate Encoder to Downstream Tasks:** To evaluate the effectiveness of model stealing attacks against the encoder, the adversary can leverage the same downstream task to both the target and surrogate encoders. Concretely, the adversary trains an extra linear layer for the target and surrogate encoders, respectively. Note that we refer to the target/surrogate encoder and the extra linear layer as the target/surrogate classifiers. Then, the adversary quantifies the attack effectiveness by measuring the performance of the target/surrogate classifier on the downstream tasks as shown in Section 4.3.1.

### 4.3.3 Cont-Steal Attacks Against Encoders

To better leverage the rich information from the embeddings, we propose Cont-Steal, a contrastive learning-based model stealing attacks against encoders, which leverages contrastive learning to enhance the stealing performance. Concretely, Cont-Steal aims to enforce the surrogate embedding of an image to get close to its target embedding (defined as a positive pair), and also push away embeddings of different images regardless of being generated by the target or the surrogate encoders (defined as negative pairs). There are three steps for the adversary to conduct contrastive stealing attacks against encoders and one step for further evaluation.

**Obtain the Surrogate Dataset:** The adversary follows the same strategy as Section 4.3.1 to obtain the surrogate dataset.

**Data Augmentation:** Our proposed Cont-Steal leverages data augmentation to transform an input image into its two augmented views. In this section, we leverage RandAugment [25] as the augmentation method, which is made up of a group of advanced augmentation operations. Concretely, we set  $n = 2$  and  $m = 14$  following Cubuk et al. [25] where  $n$  denotes the number of transformations to a given sample and  $m$  represents the magnitude of global distortion.

**Train the Surrogate Encoder:** Instead of querying the encoders with the original images, the adversary queries the encoders with the augmented views of them. Concretely, for an input image  $x_i$ , we generate two augmented views of it, i.e.,  $\tilde{x}_{i,s}$  and  $\tilde{x}_{i,t}$ , where

$\tilde{x}_{i,s}/\tilde{x}_{i,t}$  is used to query the surrogate/target encoder. We consider  $(\tilde{x}_{i,s}, \tilde{x}_{j,t})$  as a positive pair if  $i = j$ , and otherwise a negative pair.

Given a mini-batch of  $N$  samples, we generate  $N$  augmented views as the input of the target encoder and another  $N$  augmented views as the input of the surrogate encoders. Concretely, the loss of Cont-Steal can be formulated as follows:

$$D_{encoder}^+(i) = \exp(\text{sim}(E_S(\tilde{x}_{i,s}), E_T(\tilde{x}_{i,t}))/\tau) \quad (4.3)$$

$$D_{encoder}^-(i) = \sum_{k=1}^N (\exp(\text{sim}(E_S(\tilde{x}_{i,s}), E_T(\tilde{x}_{k,t}))/\tau)) \quad (4.4)$$

$$D_{self}^-(i) = \sum_{k=1}^N \mathbb{1}_{[k \neq i]} (\exp(\text{sim}(E_S(\tilde{x}_{i,s}), E_S(\tilde{x}_{k,s}))/\tau)) \quad (4.5)$$

$$l(i) = -\log \frac{D_{encoder}^+(i)}{D_{encoder}^-(i) + D_{self}^-(i)} \quad (4.6)$$

$$L_{Cont-Steal} = \frac{\sum_{k=1}^N l(k)}{N} \quad (4.7)$$

where  $e_s(\cdot)$  and  $e_t(\cdot)$  denotes the surrogate and target encoder,  $\text{sim}(u, v) = u^T v / \|u\| \|v\|$  represents the cosine similarity between  $u$  and  $v$ , and  $\tau$  is parameter to control the temperature.

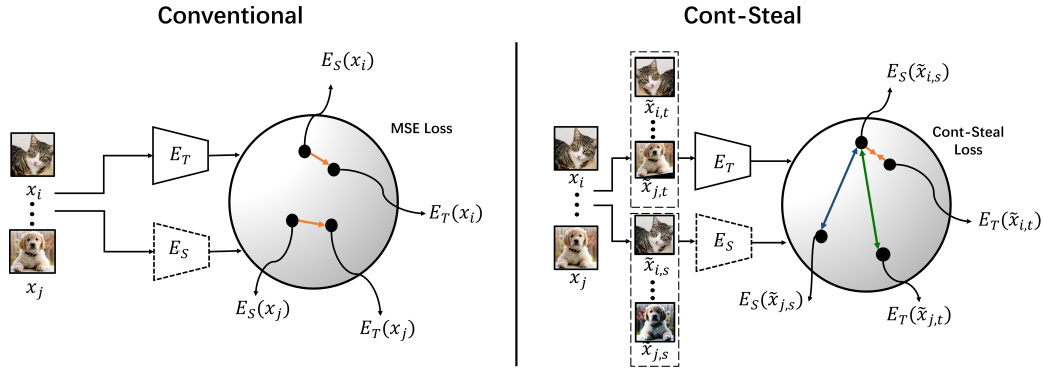
As illustrated in Figure 4.2, conventional attack treats each embedding individually without interacting across pairs. However, different embeddings are beneficial to each other as they can serve as anchors to better locate the position of the other embeddings in their space. Cont-Steal maximizes the similarity of embeddings generated from the target and surrogate encoders for a positive pair  $(\tilde{x}_{i,s}, \tilde{x}_{i,t})$  (orange arrows in Figure 4.2). For the embedding generated from the target and surrogate encoders for a any pair  $(\tilde{x}_{i,s}, \tilde{x}_{j,t})$ , contrastive stealing aims to make them more distant (green arrows in Figure 4.2). Besides, as pointed out by Chen et al. [18], contrastive learning benefits larger negative samples. To achieve this goal, we also consider the embeddings generated from the surrogate encoder for augmented views of different images, i.e.,  $(\tilde{x}_{i,s}, \tilde{x}_{j,s})$ , as negative pairs minimize their similarity (blue arrows in Figure 4.2). We later show that such design can enhance the performance of contrastive stealing (see Table 4.3).

In each batch, given  $N$  training samples, we first generate  $2N$  augmented views and feed target encoder and surrogate encoder with different views generated by the same samples. Then, we optimize the surrogate encoder by minimizing  $L_{Cont-Steal}$ .

**Apply the Surrogate Encoder to Downstream Tasks:** We follow Section 4.3.2 to evaluate the effectiveness of model stealing on downstream tasks.

## 4.4 Experiments

In this section, we first describe the experimental setup in Section 4.4.1. Then we show the performance of the target encoders on the downstream tasks in Section 4.4.2. Next, we summarize the performance of conventional attacks against classifiers and encoders



**Figure 4.2:** Conventional attack (top) vs. Cont-Steal (bottom) against encoders. Conventional attack applies MSE loss to approximate target embeddings for each sample individually. Cont-Steal (bottom) introduces data augmentation and interacts across multiple samples: associating target/surrogate embeddings of the same images closer and repulsing those of different images farther away. The target encoder ( $E_t$ ) is pre-trained and fixed as shown in the solid frame. The surrogate encoder ( $E_s$ ) is trainable by adversary as shown in the dashed frame.

in Section 4.4.3. Lastly, we evaluate the performance of Cont-Steal and conduct ablation studies to demonstrate its effectiveness under different settings in Section 4.4.4.

#### 4.4.1 Experimental Setup

**Datasets:** We leverage four image datasets to conduct our experiments.

- **CIFAR10 [1]:** This dataset contains 50,000 training images and 10,000 testing images in 10 classes. Each image in this dataset has the size of  $32 \times 32 \times 3$ .
- **ImageNet100 [27]:** We leverage the ImageNet100 dataset to pre-train the encoder. The ImageNet100 dataset is a subset of the ImageNet dataset that contains about 160,000 training images distributed in 100 classes. Each sample has the size of  $224 \times 224 \times 3$ .
- **Fashion-MNIST [128]:** Fashion-MNIST (abbreviated as F-MNIST) is an image dataset containing 10 classes. It has 60,000 training samples and 10,000 testing samples. Each sample has the size  $28 \times 28 \times 1$ .

- **SVHN [82]:** This dataset contains 73,257 training images and 26,032 testing images distributed in 10 classes. Each image in this dataset has the size of  $32 \times 32 \times 3$ .
- **STL10 [24]:** This is a 10-classes dataset where each class contains 500 labeled training images and 800 labeled testing images. It also contains 100,000 unlabeled images. The size of each image is  $96 \times 96 \times 3$ .

**Pre-training Target Encoders:** We use CIFAR10 and ImageNet100 as our target encoder’s pre-training datasets. We use SimCLR, MoCo, BYOL, and SimSiam to train a ResNet18 [43] as our target encoders. Our implementation is based on a PyTorch framework of contrastive learning.<sup>4</sup> We train our encoders for 400 epochs with the Adam [55] optimizer and initial learning rate 0.001.

**Training Downstream Classifiers:** We use the pre-trained encoders to train extra linear layers as the classifiers using the above-mentioned datasets as well. For encoders pre-trained on CIFAR10, we will reshape each sample’s size to  $32 \times 32 \times 3$ . For encoders pre-trained on ImageNet100, we will reshape each sample’s size to  $224 \times 224 \times 3$ . We will conduct the whole model fine-tune instead of fine-tuning extra linear layer to make the model perform better on the specific dataset. We train our downstream classifiers for 100 epochs with the initial learning rate of  $3e^{-4}$ . We use cross-entropy loss to optimize the linear layer’s parameters.

Note that to evaluate the attack performance against encoders, we train the linear layers for both target and surrogate encoders on the same downstream task, i.e., with the same dataset used to pre-train the target encoder.

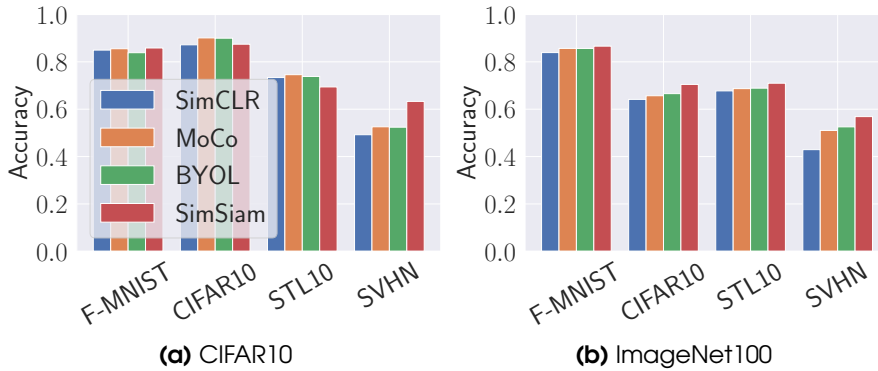
**Evaluation Metrics:** We use agreement and accuracy to evaluate the model stealing attack’s performance. The agreement will evaluate the similarity of surrogate encoders and target encoders in downstream tasks. The accuracy will evaluate the practicality of surrogate encodes on downstream tasks.

- **Agreement:** We calculate agreement by the following formula. The agreement will evaluate the similarity of surrogate encoders and target encoders in the downstream task which is also the most important metric to evaluate model stealing attack’s performance. It can be defined as:  $Agreement = \frac{\sum_{i=1}^n f_s(x_i) == f_t(x_i)}{n}$ . Where  $f_s$  is surrogate model contains surrogate encoder and surrogate classifier,  $f_t$  is target model.
- **Accuracy:** Accuracy evaluates the practicality of surrogate encodes on downstream tasks. In model stealing attacks, accuracy depends on the target model.

#### 4.4.2 Performance of the Target Encoder on Downstream Tasks

We first show the target encoder’s performance in various downstream tasks. The results are summarized in Figure 4.3. We observe that encoders pre-trained by contrastive

<sup>4</sup><https://github.com/vturrisi/solo-learn>



**Figure 4.3:** The performance of target classifiers composed by target encoder and an extra linear layer. The encoders are pre-trained on CIFAR10 (a) and ImageNet100 (b). The x-axis represents different downstream datasets for the target encoder and classifier. The y-axis represents the target model’s accuracy on downstream tasks.



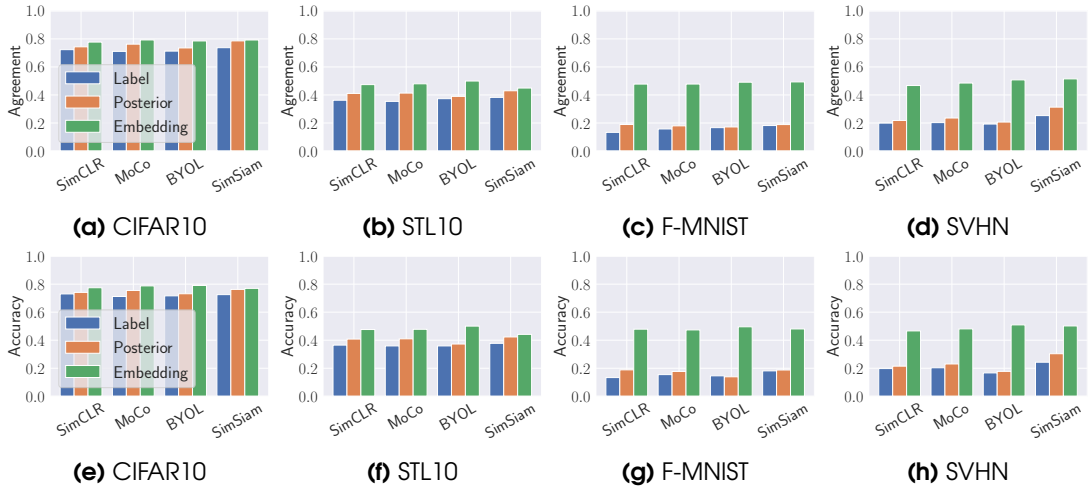
**Figure 4.4:** The t-SNE projection of 1,000 randomly selected samples’ predicted labels, posteriors, and embeddings respectively. Note that the target model is pre-trained by SimCLR on CIFAR10.

learning can achieve remarkable performance even when downstream tasks are completely different from pre-training datasets. For instance, given the encoder pre-trained by SimCLR on the ImageNet100 dataset, the downstream accuracy is 0.838, 0.641, 0.673, and 0.429 when the downstream dataset is F-MNIST, CIFAR10, STL10, and SVHN, respectively. This indicates that such pre-trained encoders can generate embeddings that has high representation ability and can be easily generalized to other tasks. We later show that model stealing attacks can build surrogate encoders that achieve similar accuracy levels as the target encoders but with much less cost.

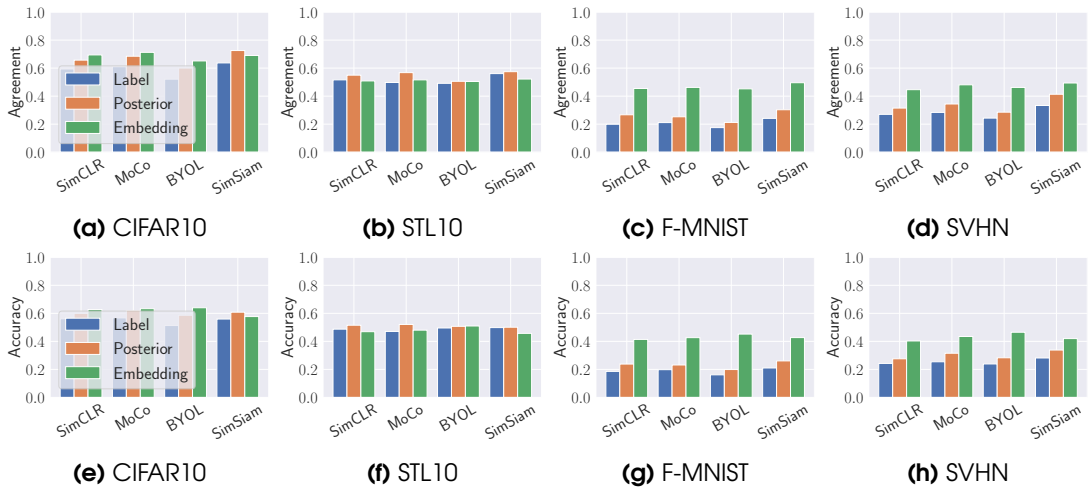
### 4.4.3 Performance of Conventional Attacks

We conduct our experiments to explore whether the encoders are more vulnerable to model stealing attacks. We show our results of target encoders and downstream classifiers both trained on CIFAR10 in Figure 4.5. In all cases, the adversary can get better attack performance by stealing encoders rather than classifiers. This gap becomes especially apparent when the adversary has absolutely no knowledge of the train data. For instance, when surrogate dataset is CIFAR10 (the same as target downstream



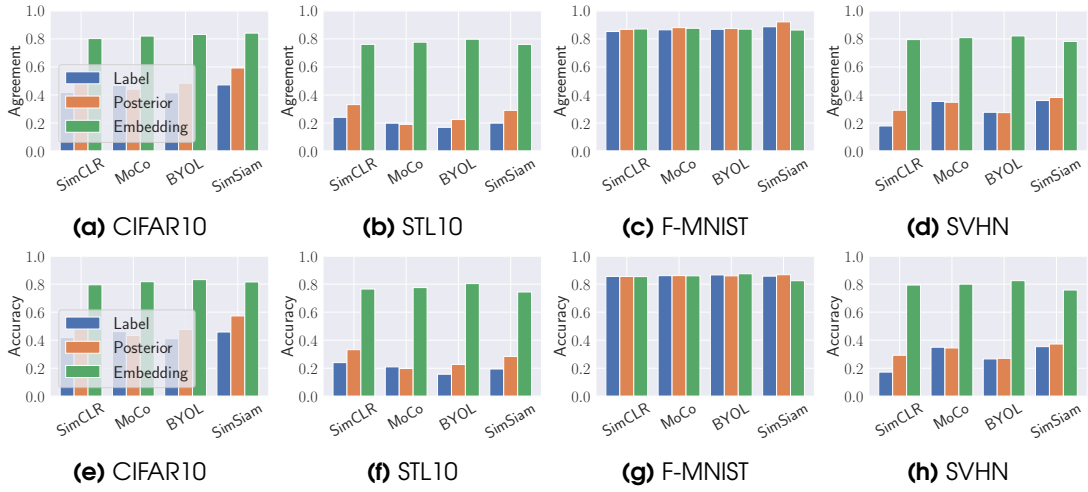


**Figure 4.5:** The performance of model stealing attack against target encoders and downstream classifiers both trained on CIFAR10. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.

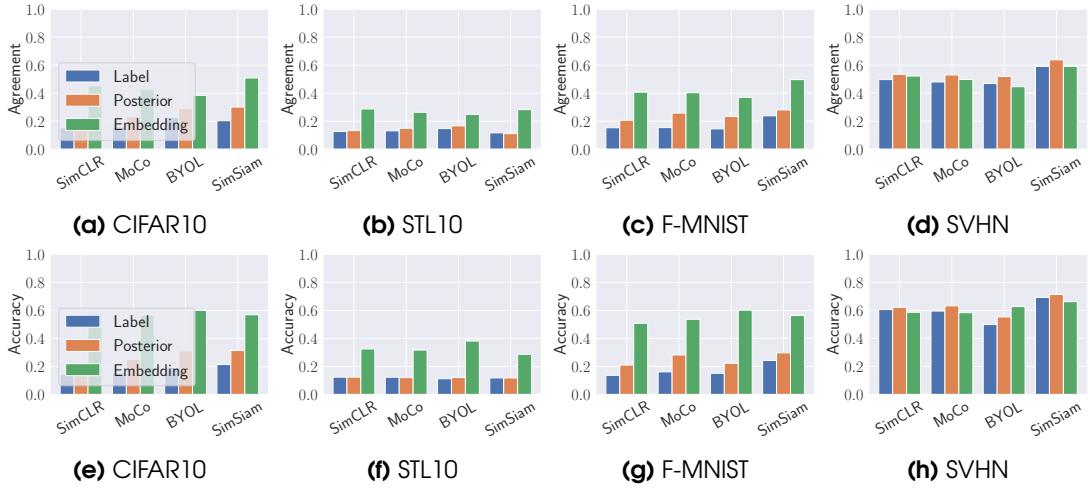


**Figure 4.6:** The performance of model stealing attack against target encodes and downstream classifiers trained on CIFAR10 and STL10. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.

dataset), stealing SimCLR’s embeddings can achieve 0.785 agreement when stealing predicted labels can achieve 0.712 agreement. However, when the surrogate dataset is totally different from target downstream dataset, e.g., SVHN, stealing embeddings from

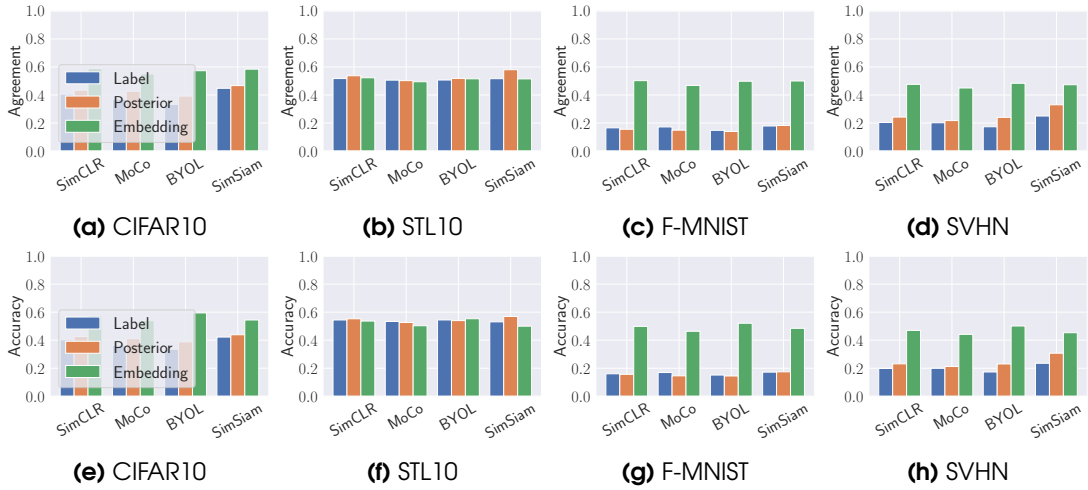


**Figure 4.7:** The performance of model stealing attack against target encodes and downstream classifiers trained on CIFAR10 and Fashion-MNIST. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.

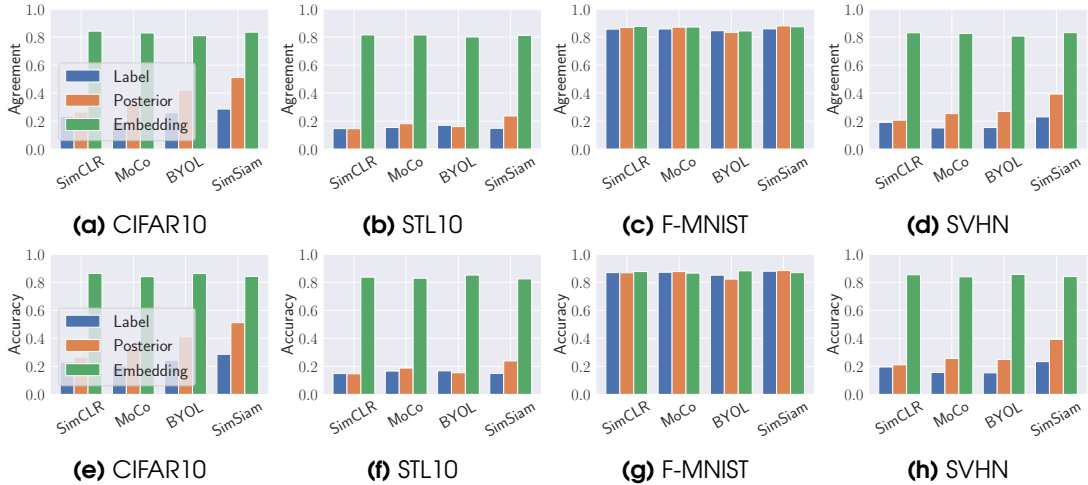


**Figure 4.8:** The performance of model stealing attack against target encodes and downstream classifiers trained on CIFAR10 and SVHN. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.

SimCLR can still achieve 0.507 agreement while the agreement of stealing predicted labels drops to 0.192.

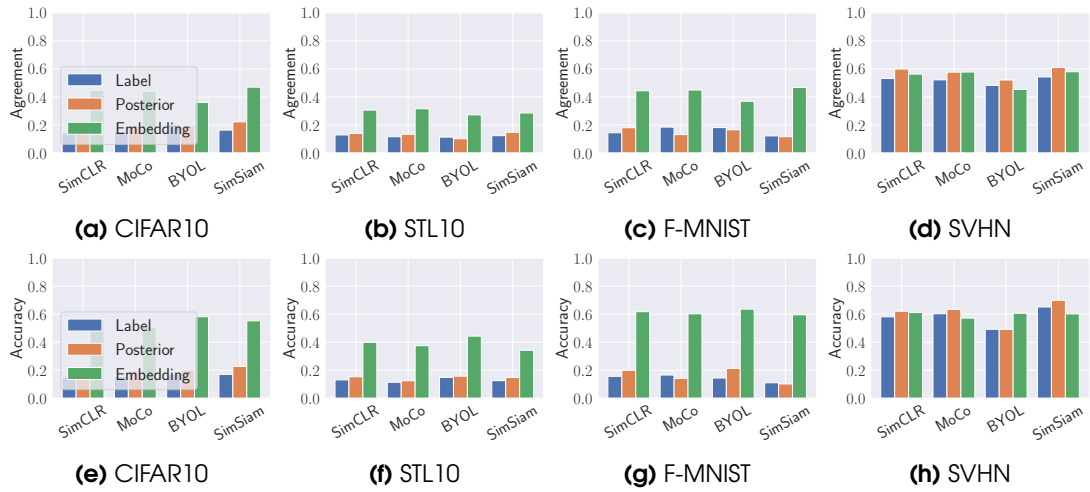


**Figure 4.9:** The performance of model stealing attack against target encodes and downstream classifiers trained on ImageNet and STL10. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.



**Figure 4.10:** The performance of model stealing attack against target encodes and downstream classifiers trained on ImageNet and Fashion-MNIST. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.

To better understand this phenomenon, we extract samples’ predicted labels (one-hot), posteriors, and embeddings from the SimCLR model on CIFAR10 and project

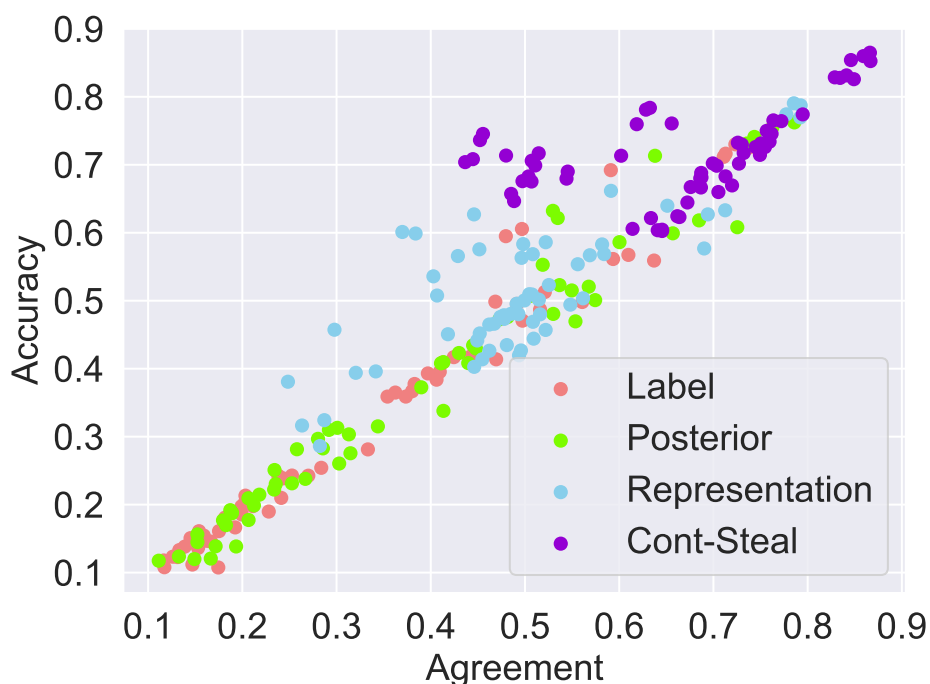


**Figure 4.11:** The performance of model stealing attack against target encodes and downstream classifiers trained on ImageNet and SVHN. Target models can output predicted labels, posteriors, or embeddings. The adversary uses CIFAR10, STL10, Fashion-MNIST (F-MNIST), SVHN to conduct model stealing attacks. The x-axis represents different kinds of target models. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.

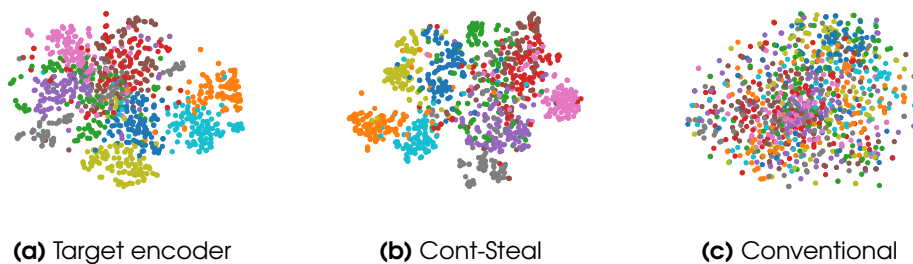
them into a 2-dimension space with t-Distributed Neighbor Embedding (t-SNE) [76]. Figure 4.4 shows the results for the predicted label, posteriors, and embedding. Different colors represent different classes. From Figure 4.4a, we observe that samples from different classes may be mapped into the same area since they have the same predicted label. For instance, the green and orange points are mapped into the right area although they do not have the same ground truth labels. Regarding posteriors (Figure 4.4b), we find that different colors are more separable compared to the predicted labels (Figure 4.4a). However, samples from different classes are still likely to mix with each other. For instance, the points from the upper left area cannot be separated easily. Regarding the embeddings (Figure 4.4c), we find that samples with different classes are mostly projected into different spaces, which makes it easier for the surrogate encoders to learn the distribution of different samples. This indicates that the embeddings from the target model indeed leak more information than posteriors and labels and can facilitate the model stealing process.

We also find that all model stealing attacks’ accuracy and agreement are highly correlated. As shown in Figure 4.12, the agreement is highly correlated with to the accuracy. This indicates that besides accuracy, agreement can also be used as a metric to evaluate the performance of model stealing attacks. We show the result on Figure 4.12. It can be obviously seen that agreement is highly related to accuracy. We use the linear regression method to describe the relationship between agreement and accuracy and find that the relation function is  $y=0.940 * x$ .

**Takeaways:** In conclusion, encoders are more vulnerable to model stealing attacks than classifiers. This is because the rich information in embeddings can better facilitate

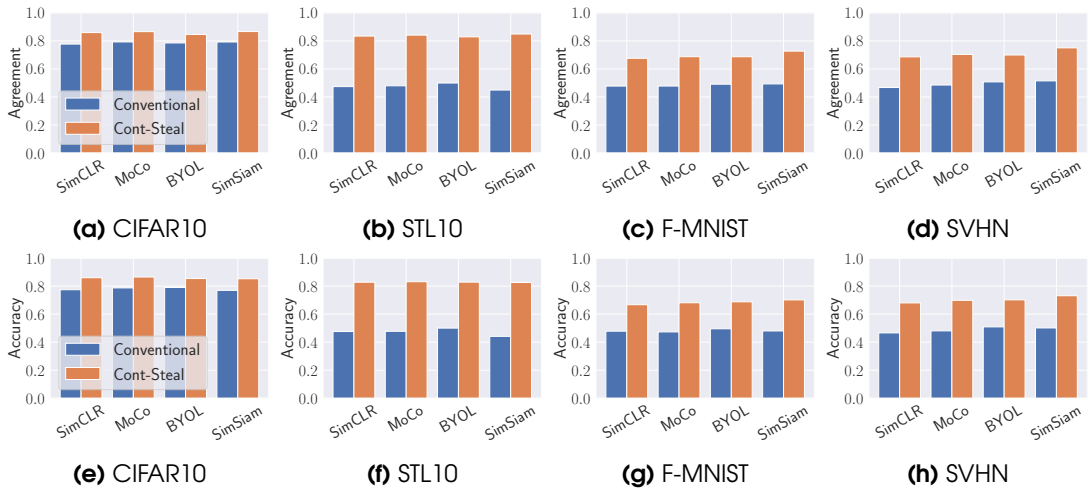


**Figure 4.12:** The relationship between accuracy and agreement. The x-axis is the agreement number and y-axis is the accuracy number.

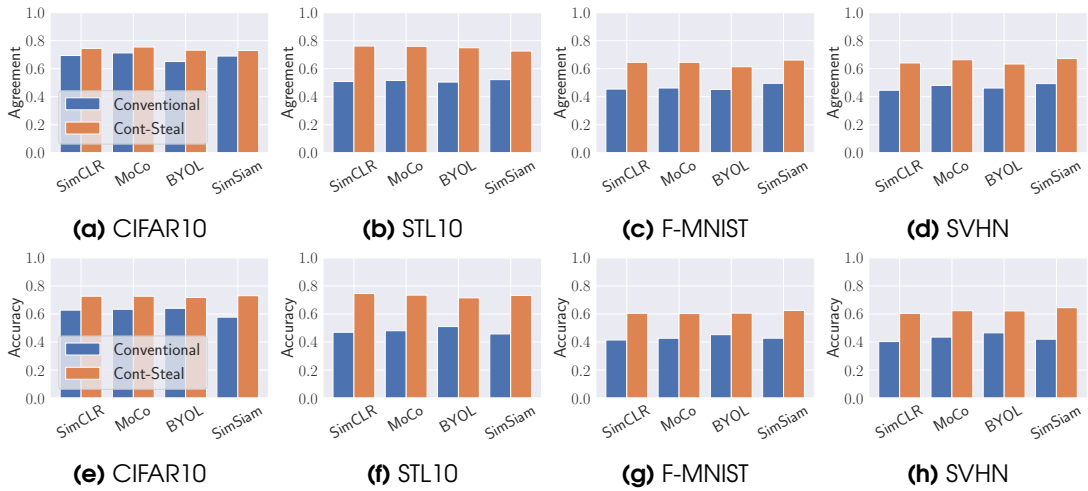


**Figure 4.13:** The t-SNE projection of 1,000 randomly selected samples' embeddings from target encoder, surrogate encoder under Cont-Steal, and surrogate encoder under the conventional attack, respectively. Note that the target encoder is pre-trained by SimCLR on CIFAR10.

the learning process of surrogate encoders. The more useful information the target model gives out, the more vulnerable the target model is to model stealing attacks. Although model stealing attacks against encoders can achieve great performance, our study shows that encoders leak more information for the adversary to learn to make surrogate encoders more similar to target encoders. To fully use the encoder's output, we conduct our attacks by leveraging Cont-Steal.



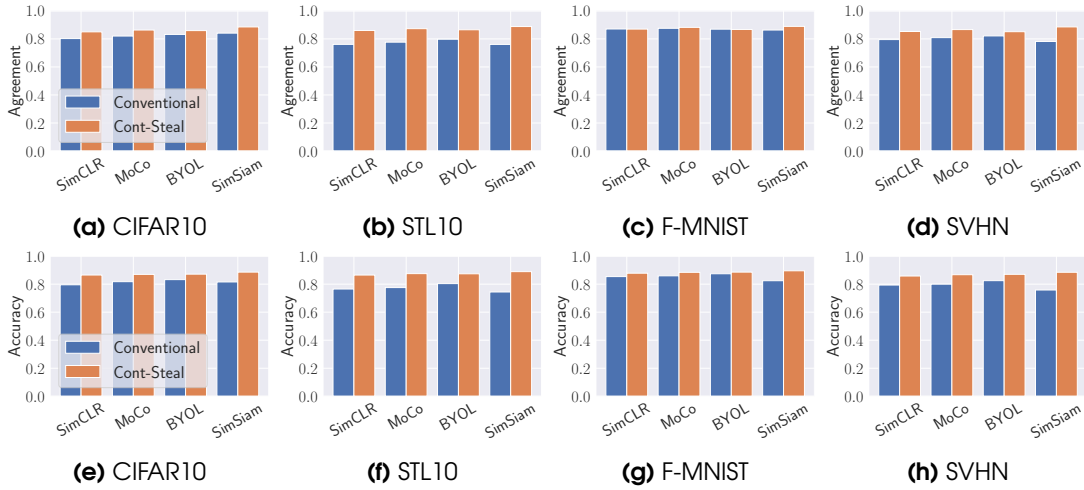
**Figure 4.14:** The performance of Cont-Steat and conventional attack against target encoders trained on CIFAR10. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses CIFAR10 as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.



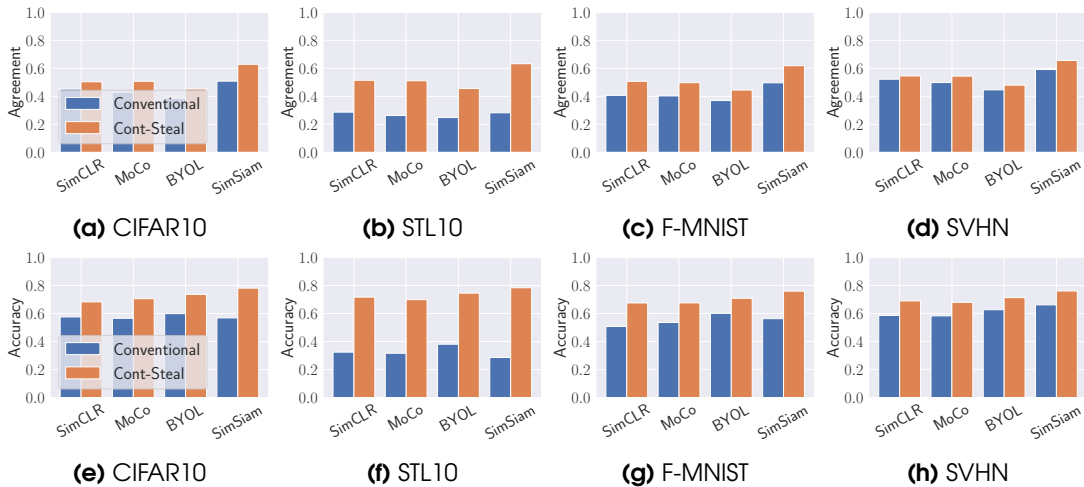
**Figure 4.15:** The performance of Cont-Steat and conventional attack against target encoders trained on CIFAR10. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses STL10 as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.

#### 4.4.4 Performance of Cont-Steat

As shown in Section 4.4.3, encoders are more vulnerable to model stealing attacks since the embedding usually contains richer information compared to the predicted label



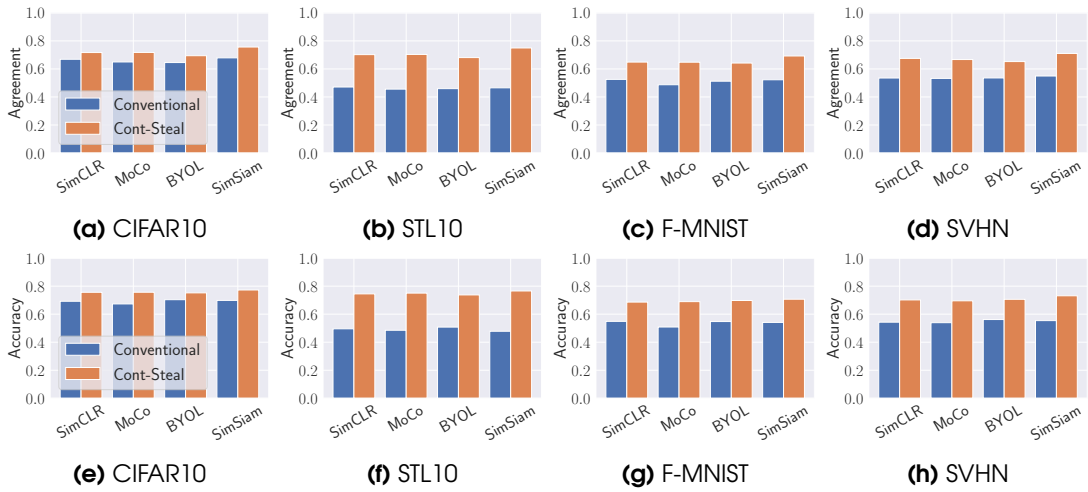
**Figure 4.16:** The performance of Cont-Steal and conventional attack against target encoders trained on CIFAR10. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses F-MNIST as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.



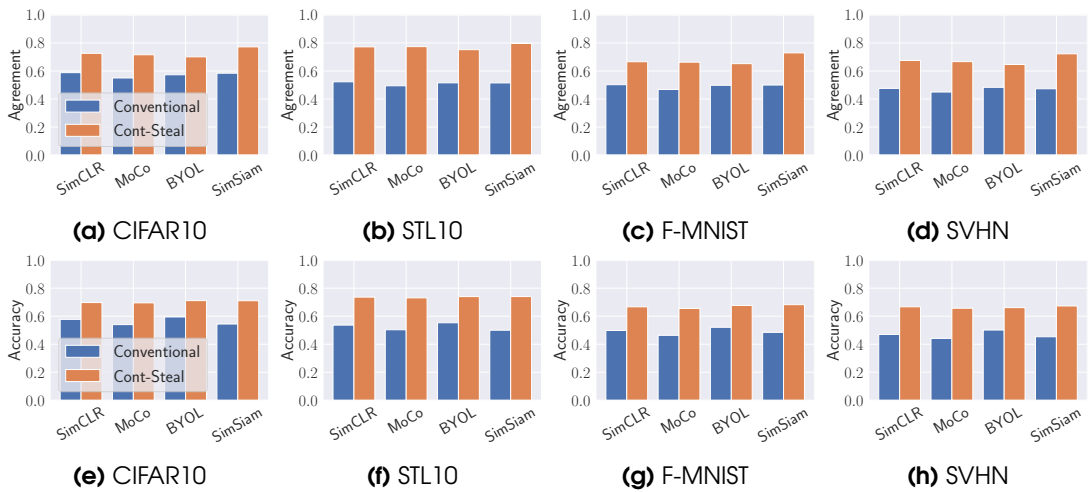
**Figure 4.17:** The performance of Cont-Steal and conventional attack against target encoders trained on CIFAR10. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses SVHN as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.

or posteriors. We then show that our proposed Cont-Steal can achieve better attack performance by making deeper use of embeddings’ information.

Figure 4.14 shows the attack performance when the target pre-training dataset is CIFAR10. Note that we also show the attack performance on other settings in



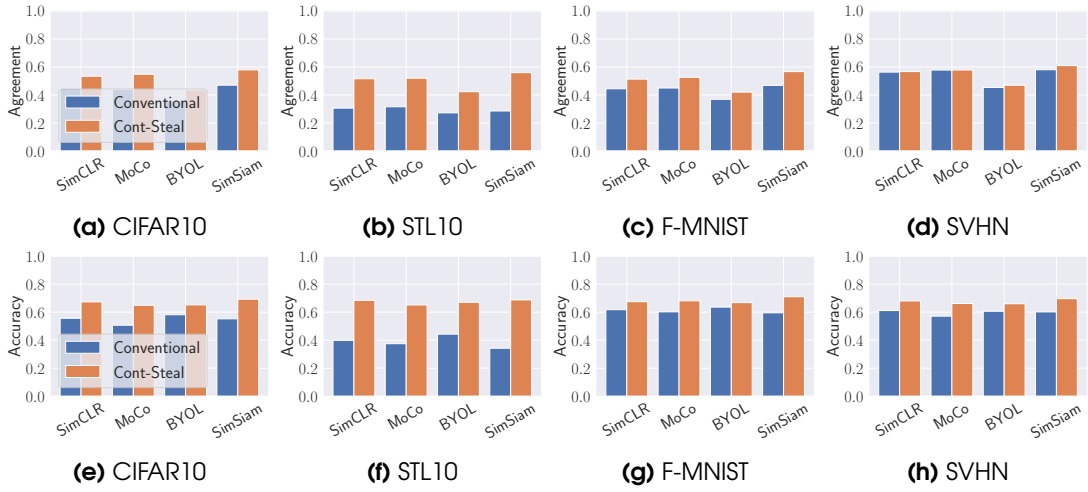
**Figure 4.18:** The performance of Cont-Steat and conventional attack against target encoders trained on ImageNet100. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses CIFAR10 as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.



**Figure 4.19:** The performance of Cont-Steat and conventional attack against target encoders trained on ImageNet100. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses F-MNIST as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.

the appendix. We discover that compared to conventional attacks against encoders, Cont-Steat can consistently achieve better performance. For instance, as shown in Figure 4.14d, when the target encoder is MoCo trained on CIFAR10, if the adversary uses STL10 to conduct model stealing attacks against encoders, the surrogate encoder





**Figure 4.20:** The performance of Cont-Steat and conventional attack against target encoders trained on ImageNet100. The adversary uses CIFAR10, STL10, F-MNIST, and SVHN to conduct model stealing attacks. The adversary uses SVHN as the downstream task to evaluate the attack performance. The x-axis represents different kinds of the target model. The first line’s y-axis represents the agreement of the model stealing attack. The second line’s y-axis represents the accuracy of the model stealing attack.

can achieve 0.841 agreement in CIFAR10 downstream tasks with the Cont-Steat but only 0.479 with conventional attacks. Another finding is that compared to the same distribution surrogate dataset, our Cont-Steat can better enhance the performance when the surrogate dataset comes from a different distribution from the pre-trained dataset. For instance, when the target encoder is SimCLR trained on CIFAR10, Cont-Steat outperforms conventional attack by 0.055 agreement when the surrogate dataset is also CIFAR10, while the improvement increases to 0.207 and 0.214 when the surrogate dataset is STL10.

To better understand why Cont-Steat can always achieve better performance, we extract samples’ embeddings generated by different encoders, i.e., the target encoder, surrogate encoder trained with the conventional attack, and surrogate encoder trained with the Cont-Steat, and project them into a 2-dimensional space using t-SNE. From the results summarized in Figure 4.13, we find that Cont-Steat can effectively mimic the pattern of the embeddings as the target encoder. However, the conventional attack fails to capture such patterns for a number of input samples, e.g., the outer circle in Figure 4.23c. This further demonstrates that Cont-Steat benefits from jointly considering different embeddings as they can serve as anchors to better locate the position of the other embeddings in their space.

**Takeaways:** Our proposed Cont-Steat can achieve much better attack performance than conventional attack. This is because Cont-Steat can leverage higher order information across samples and then learn more about the target model.

**Table 4.1:** The monetary and (training) time costs for normal training and Cont-Steal attack. Cont-Steal’s monetary cost contains two parts: query cost and training cost. Note that we ignore the query time cost of Cont-Steal as it normally has a smaller value than the training time cost.

Model	Monetary Cost		Time Cost	
	Normal (\$)	Cont-Steal (\$)	Normal (h)	Cont-Steal (h)
SimCLR	58.68	11.83 (1.83 + 10)	20.01	0.62
MoCo	54.83	12.13 (2.13 + 10)	18.69	0.73
BYOL	61.46	12.08 (2.08 + 10)	20.96	0.71
SimSiam	57.14	12.00 (2.00 + 10)	19.46	0.68

#### 4.4.5 Cost Analysis

As we mentioned before, pre-train a state-of-the-art encoder is time-consuming and resource-demanding. We wonder if the model stealing attacks can steal the functionality of the encoder with much less cost. To this end, we evaluate the time and monetary cost of training an encoder from scratch or stealing a pre-trained encoder via Cont-Steal. The monetary cost of model stealing includes querying the target model and training the surrogate model. We refer to the query price as \$1 for 1,000 queries based on AWS.<sup>5</sup> Our experiment is conducted on 1 NVIDIA A100 whose price is \$2.934 per hour based on google cloud.<sup>6</sup>

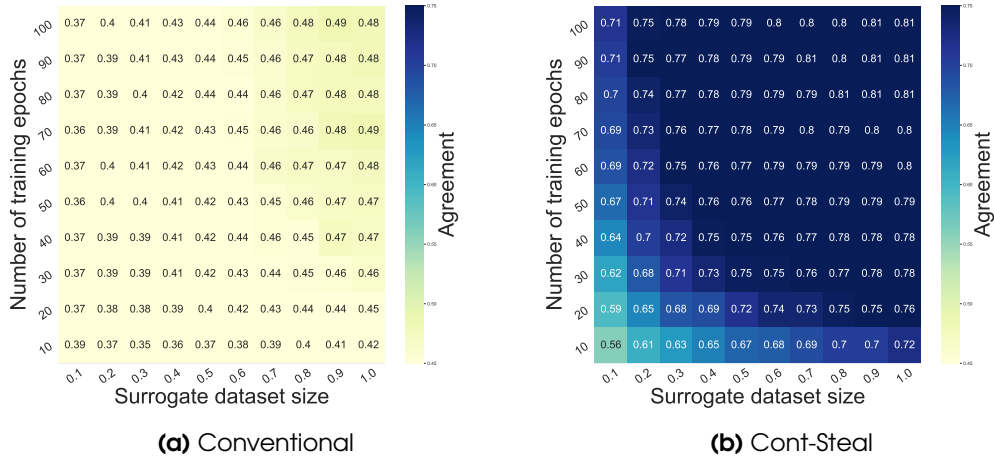
The monetary and time cost is shown in Table 4.1. We observe that Cont-Steal can obtain a surrogate encoder with much less monetary and time cost than training the encoder from scratch. For instance, a ResNet18 trained by SimCLR on CIFAR10 takes 20.01 hours and 58.68\$ on 1 NVIDIA A100 GPU while Cont-Steal only takes 0.62 hours and 11.83\$ to steal an encoder that performs similarly on downstream tasks. The results demonstrate that Cont-Steal is able to construct surrogate encoders that perform similarly as the target encoders but with much less time and monetary cost.

#### 4.4.6 Ablation Studies on Adversary Training Process

**Impact of Surrogate Encoder’s Architecture:** Previous experiments are based on the assumption that the adversary knows the target encoder’s architecture. We then investigate whether the attack against the encoder is still effective when the surrogate encoder has different model architectures compared to the target encoder. Concretely, we perform Cont-Steal against the ResNet18 encoder with surrogate encoder’s architecture as ResNet18, ResNet34, ResNet50, DenseNet161, and MobileNetV2, respectively. As shown in Table 4.2, we can see that the architecture of the surrogate model only has limited influence on the attack performance. For instance, the adversary can achieve 0.839 accuracy using the same architecture as the target model while

<sup>5</sup><https://aws.amazon.com/rekognition/pricing/>

<sup>6</sup><https://cloud.google.com/compute/gpus-pricing>



**Figure 4.21:** Heatmap of the agreement scores of model stealing attacks. Target model’s encoder and downstream classifier are both ResNet18 trained by SimCLR on CIFAR10. The surrogate dataset is STL10. The surrogate dataset’s size refers to the proportion of surrogate data we used to the whole surrogate dataset. We show the performance of 100 combinations of different training epoch and surrogate dataset’s size.

it can even achieve 0.840 accuracy when using a more complex model architecture (ResNet50) on SimCLR. The attack performance will drop a little if the adversary uses DenseNet161 and MobileNetV2. This might be because the architectures of DenseNet161 and MobileNetV2 have larger differences compared to ResNet18. However, the accuracy with DenseNet161/MobileNetV2 as the surrogate encoder’s architecture can still achieve 0.828/0.811. This demonstrates that the model architectures of the surrogate encoder only has limited impact on the attack performance, which makes the attack a more realistic threat.

#### **Impact of Surrogate Dataset’s Size and Surrogate Model’s Training Epoch:**

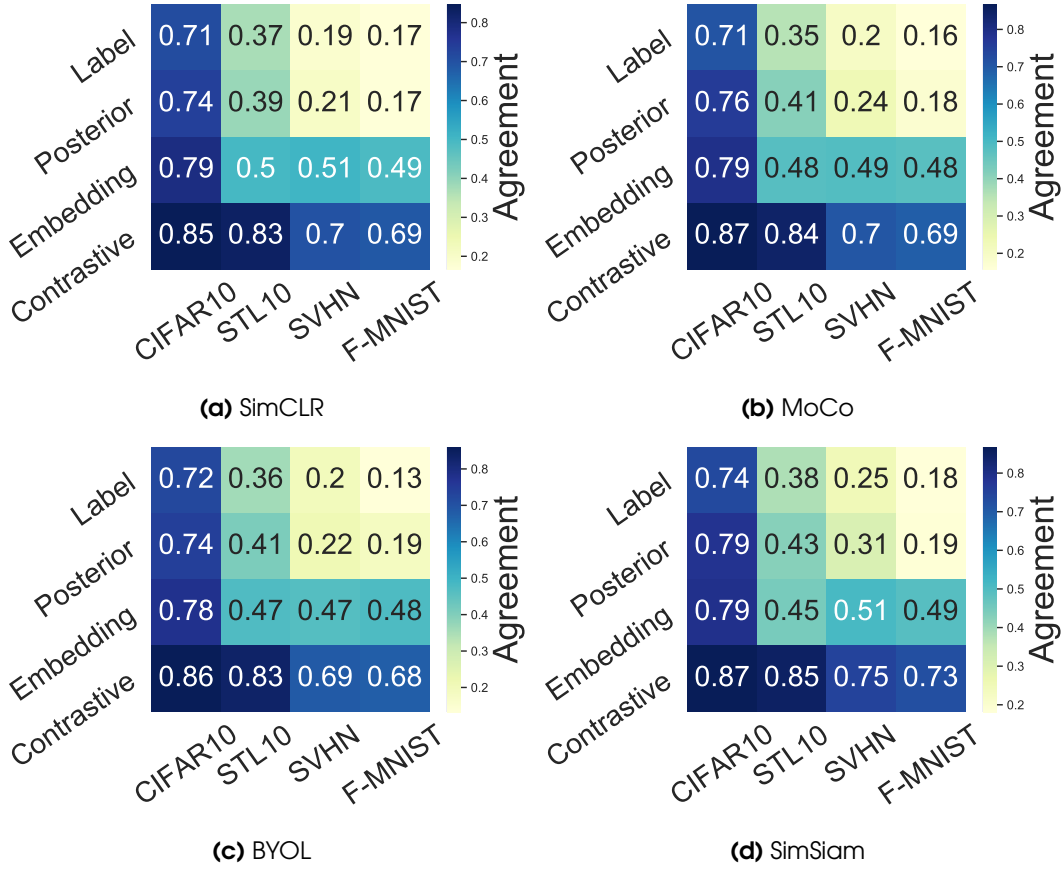
We conduct ablation studies here to better illustrate the effectiveness of Cont-Steal. Concretely, we investigate whether conventional attacks and Cont-Steal are still effective under limited surrogate dataset size and number of training epochs. Ideally, we consider the attack that can reach similar performance but with less surrogate dataset size and fewer training epochs as a better attack as it requires less query and monetary costs. As shown in Figure 4.21, we observe that both conventional attacks and Cont-Steal can have better performance with a larger surrogate dataset size and more number of training epochs. For instance, Cont-Steal reaches 0.675 agreement when the surrogate encoder is trained with 10% surrogate dataset for 50 epochs, while the agreement increases to 0.812 with 100% surrogate dataset and 100 training epochs. The second observation is that Cont-Steal outperforms conventional attacks even with limited data and training epochs. For instance, even with only 10% surrogate dataset and 10 training epochs, surrogate encoder built by Cont-Steal can reach 0.562 agreement, while the conventional attack can only achieve 0.479 agreement with the full surrogate dataset and 100 training epochs.

**Table 4.2:** Cont-Steal attack performance of different surrogate architectures. Target encoders (ResNet18) and downstream classifiers are trained on CIFAR10. The surrogate dataset is also CIFAR10.

Framework	Architectures	Agreement	Accuracy
SimCLR	ResNet18	0.835	0.839
	ResNet34	0.837	0.842
	ResNet50	0.844	0.840
	DenseNet161	0.831	0.828
	MobileNetV2	0.815	0.811
MoCo	ResNet18	0.857	0.849
	ResNet34	0.858	0.849
	ResNet50	0.867	0.856
	DenseNet161	0.813	0.811
	MobileNetV2	0.796	0.801
BYOL	ResNet18	0.845	0.842
	ResNet34	0.850	0.847
	ResNet50	0.857	0.855
	DenseNet161	0.845	0.821
	MobileNetV2	0.839	0.847
SimSiam	ResNet18	0.856	0.835
	ResNet34	0.858	0.839
	ResNet50	0.860	0.848
	DenseNet161	0.791	0.783
	MobileNetV2	0.812	0.832

As we mentioned before, this is because Cont-Steal can enforce the surrogate embedding of an image close to its target embedding, and also push away embeddings of different images irrespective of being generated by the target or the surrogate encoders (see also Table 4.3 for the necessity of introducing negative pairs from the surrogate encoder). This makes Cont-Steal a more effective model stealing attack against encoders.

**Impact of Surrogate Dataset’s Correlation with the Target Dataset:** In the meanwhile, since the adversary cannot always have knowledge about the target dataset, the impact of surrogate dataset’s correlation with the target dataset is also worth consideration. We find that Cont-Steal depends less on the surrogate dataset’s distribution and can always achieve stable performance. We plot the attack agreement in Figure 4.22 where the target encoders and downstream classifiers are trained on CIFAR10. We can see that when the adversary conducts a conventional attack against the classifier, the adversary’s knowledge of target training data is crucial. For example, when the adversary can only get predicted label from target model, he/she can only achieve 0.182 agreement when using F-MNIST to attack the model trained by SimCLR, while it can achieve 0.711 agreement when using CIFAR10 as the surrogate dataset, which is same as target dataset. However, compared to predicted label or posterior



**Figure 4.22:** Heatmap of the agreement scores of model stealing attacks. We show the performance of 16 combinations of different information that the target model outputs, and the adversary’s knowledge on target training data. Target models are trained on CIFAR10

as the response, embedding depends less on the surrogate dataset distribution, and Cont-Steal can better leverage the embedding information, contributing to the less dependent on the surrogate dataset’s distribution. For instance, when the target model is trained by SimCLR, Cont-Steal can achieve 0.832 agreement when the surrogate dataset is STL10, which is even better than the best conventional attack (0.781) using the exact same target training dataset as the surrogate dataset and embedding as the response. Such observation better implies that Cont-Steal can always achieve good performance regardless of the surrogate dataset’s distribution and can also achieve more generalized performance in practice.

**Impact of Negative Pairs Generated from the Surrogate Encoder:** In Cont-Steal’s loss functions, besides  $D_{encoder}^-$ , we also consider the distance of negative pairs generated from the surrogate encoder itself, i.e.,  $D_{self}^-$ . To evaluate the necessity of  $D_{self}^-$ , we take the target encoder trained by BYOL on CIFAR10 and downstream task on STL10 as an example and study the attack performance with and without  $D_{self}^-$ . The

results are summarized in Table 4.3. We find that adding  $D_{self}^-$  greatly improves the attack performance in both accuracy and agreement. For instance, when the surrogate dataset is STL10, the surrogate model stolen by Cont-Steal with  $D_{self}^-$  achieves 0.817 agreement while only 0.314 if without  $D_{self}^-$ . The reason behind is that the negative pairs generated from the surrogate encoder can serve as extra “anchors” to better locate the position of the embedding, which leads to higher agreement. Such observation demonstrates that it is important to introduce  $D_{self}^-$  in Cont-Steal as well.

**Takeaways:** We show that both conventional attacks and our Cont-Steal are effective even under different restrictions, which means the threat of model stealing attacks against encoders is largely underestimated. Cont-Steal’s outstanding performance further amplifies the threat by leveraging a more powerful way to conduct the model stealing attacks against the target encoder.

**Table 4.3:** The agreement and accuracy of different contrastive losses. We use BYOL trained on STL10 as the target model.

Dataset	Method	BYOL	
		Agreement	Accuracy
CIFAR10	w/o $D_{encoder}^-$	0.242	0.242
	w $D_{encoder}^-$	0.844	0.843
F-MNIST	w/o $D_{encoder}^-$	0.215	0.217
	w $D_{encoder}^-$	0.647	0.641
STL10	w/o $D_{encoder}^-$	0.314	0.320
	w $D_{encoder}^-$	0.817	0.811
SVHN	w/o $D_{encoder}^-$	0.176	0.175
	w $D_{encoder}^-$	0.655	0.650

#### 4.4.7 Further Attacks Based on Cont-Steal

As we have mentioned in the introduction part, model stealing can be used as a stepping stone for further attacks. In this section, we select adversary sample as a case study to show the importance of model stealing for further attacks on the target model. Normally, the adversary can not obtain the gradient from the target model. But to conduct adversary sample attacks, the adversary needs to obtain the gradient in most of attack scenarios. Therefore, the adversary can construct a surrogate model to generate the adversary sample and transfer it to the target model to perform the attack. We consider three widely used mechanisms to generate adversarial examples including Fast Gradient Sign Attack (FGSM) [37], Basic Iterative Methods (BIM) [58], and Projected Gradient Descent (PGD) [77]. Our target model is SimCLR pre-trained on CIFAR10 and the last layer classifier trained on STL10. We also use STL10 as the surrogate dataset to conduct Cont-Steal and generate adversary samples. Experiments show that the surrogate model can generate adversary samples that are valid for the target model

(Table 4.4). To show the necessity of the surrogate model as a springboard for attack, we also conduct the baseline attack which use another model as the springboard to attack the target model. We choose normal ResNet18 model trained on SVHN as our baseline model and then apply the adversary example to attack the target model. We observe that compare to the adversarial examples generated from the baseline model, those adversarial examples generated from the surrogate model constructed by Cont-Steal can better transfer to the target model. For instance, with PGD, the adversarial examples obtained from the surrogate model can lead to a lower classification accuracy (0.203) on the target model than those generated from the baseline model (0.246). This implies that the model stealing attack can be a valid stepping stone for more effective further attacks.

**Table 4.4:** The different methods to create adversary sample to attack on surrogate model and target model. (Lower is better)

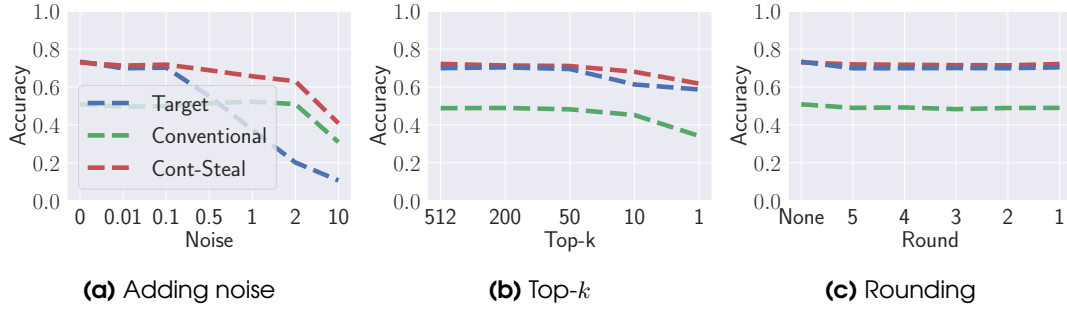
Method	Surrogate model (acc)	Target model (acc)	Baseline (acc)
FGSM [37]	0.097	0.131	0.194
BIM [58]	0.054	0.192	0.235
PGD [77]	0.092	0.203	0.246

#### 4.4.8 Defenses

In this section, we will consider different defenses against model stealing attacks on encoders to evaluate the robustness of our proposed attack. We divided all defenses into three categories: perturbation-based defense [88], watermark-based defense [5], and distribution detection-based defense [52].

**Perturbation-based Defense:** In this defense setting, the defender aims to perturb the output of the target model to limit the information the adversary can obtain. The common practice of this kind of defense includes adding noise [88], top- $k$  [88], and feature rounding [117].

Adding noise means that the defender will introduce noise value to the original output of the model. In our case, we consider adding Gaussian noise to the embeddings generated by the target encoder. We set the mean value to 0 and different noise levels represent different standard deviations of the Gaussian distribution. For Top- $k$ , the defender will only output the first  $k$  largest number of each embedding (and set the rest as 0). In this way, the high-dimensional information of the image contained in embeddings can be appropriately reduced. Regarding feature rounding, the defender will truncate the values in the embedding to a specific digit. As a case study, we consider a ResNet18 encoder pre-trained on CIFAR10 with SimCLR and take STL10 to train its downstream classifier. The experimental results are summarized in Figure 4.23. We can observe that while adding noise and top- $k$  can reduce the model stealing attacks' performance, it may also degrade the target model performance to a large extent. For instance, when the noise increases from 0 to 10, the attack performance of Cont-Steal



**Figure 4.23:** The performance of different defend methods. Target encoders are trained on CIFAR10. The downstream dataset and surrogate dataset are both STL10. The x-axis represents different defense levels. The y-axis represents the model’s accuracy.

decreases from 0.729 to 0.410, while the target encoder’s performance drops from 0.734 to 0.098. On the other hand, rounding only has limited effect on both target model performance and attack performance. This indicates that perturbation-based defense cannot defend against encoder’s model stealing attack effectively since they cannot reach a good trade-off between attack performance and model utility.

**Table 4.5:** Watermark defense. Pretrain dataset and surrogate dataset are both CIFAR10. Watermark leverages a watermark rate (wr) to verify the ownership of target models. (Higher is better)

Dataset	Target model (acc/wr)	Cont-Steal (acc/wr)	Baseline (acc/wr)
CIFAR10	0.864 / 0.998	0.769 / 0.130	0.871 / 0.095
STL10	0.721 / 0.999	0.702 / 0.034	0.733 / 0.111
SVHN	0.501 / 0.999	0.535 / 0.303	0.492 / 0.103
F-MNIST	0.857 / 0.999	0.813 / 0.061	0.850 / 0.099

**Watermark-based Defense:** Watermark-based defense is also one of the most popular defense methods against model stealing attacks (ref). Watermark provides copyright protection by adding some specific identification to the target model. If the surrogate model is stolen from the watermarked target model, then ideally, it will contain the same watermark as well, which can be used to verify the ownership of the model. As illustrated in [5], backdoor technology can be used as the watermark to protect the model. In that sense, BadEncoder [51], a backdoor mechanism against encoder, can be leveraged as a watermarking technology to our target encoder as well. The defenders first will train the watermarked (backdoored) encoder where images with a certain trigger will cause misclassification. Then, if they find the surrogate model can also misclassify images with the same trigger, the defenders can claim the ownership of the surrogate model.

In our experiments, we leverage BadEncoder to watermark the encoder pre-trained on CIFAR10 by SimCLR, and leverage different downstream datasets to perform different



tasks. We assume a strong adversary that has the same downstream dataset as the surrogate dataset. Also, we consider the baseline cases where the trigger samples are fed into the clean model to calculate the watermark rate (wr). As shown in Table 4.5), the watermark cannot be preserved as the surrogate models constructed by Cont-Steal have similar wr as the baseline model. For instance, when the downstream dataset is CIFAR10, Cont-Steal builds a surrogate model with 0.769 accuracy while only 0.130 wr, which is closed to the baseline model. This indicates that Cont-Steal can bypass the watermarking technique as it can reach similar utility while reducing the wr to a large extent.

**Distribution Detection-based Defense:** During the querying phase, one common defense method is out-of-distribution data detection [52]. This kind of model requires the query data to be in the same distribution as the target dataset. However, in our settings, it is hard to justify whether the data is in-distribution as the self-supervised learning pre-trained encoders are suitable for various downstream tasks. Therefore, the distribution detection-based defense may not be suitable against our model stealing attacks.

**Takeaways:** In conclusion, perturbation-based defense, watermark-based defense, and distribution-based defense cannot effectively defend against Cont-Steal. Most effective defense method is adding noise to the embeddings. However, adding noise will also largely decrease target model’s performance. We leave it as our future work to further explore more effective defenses.

## 4.5 Conclusion

In this section, we conduct the first model stealing risk assessment towards image encoders. Our evaluation shows that the encoder is more vulnerable to model stealing attacks compared to the classifier. This is because the embedding provided by the encoder contains richer information than the posteriors or predicted labels returned by the whole classifier. Such embedding can be leveraged by the surrogate encoder to better learn the distribution of representations from the target encoder.

To better unleash the power from the embeddings, we propose Cont-Steal, a contrastive learning-based model stealing method against encoders. Concretely, Cont-Steal introduces different types of negative pairs as “anchors” to better navigate the surrogate encoder learn the functionality of the target encoder. Extensive evaluations show that Cont-Steal consistently performs better than conventional attacks against encoders. And such an advantage is further amplified when the adversary has no information of the target dataset, a limited amount of data, and restricted query budgets. Our work points out that the threat of model stealing attacks against encoders is largely underestimated, which calls for effective intellectual property protection of representation learning techniques, especially to the defenses against encoder stealing attacks like ours.



# 5

## Deepfake Detection

Output Security



## 5.1 Introduction

Text-to-image generation models have made tremendous progress during the past few months. State-of-the-art models in this field, like Stable Diffusion [99] and DALL·E 2 [96], are able to generate high-quality images ranging from artworks to photorealistic news illustrations. Traditional image generation models, such as generative adversarial networks (GANs) [36, 53, 94, 40, 79], generate synthetic/fake images with latent code sampled from a Gaussian distribution. On the other hand, text-to-image generation models [138, 99, 96, 134, 101] require users to provide textual inputs, namely *prompts*, and generate images that match the prompts.

The high-quality synthetic images created by text-to-image generation models can be used for various purposes. For instance, they can facilitate the materialization of a novelist’s envisioned scene, perform the automated generation of illustrations for advertising campaigns, and create physical scenes that cannot be captured photographically. However, these synthetic images also pose severe threats to society. For instance, such images can be used by malicious parties to disseminate misinformation. As reported by TechCrunch, Stable Diffusion is able to generate realistic images, e.g., images on the war in Ukraine, that may be used for propaganda.<sup>1</sup> Also, these images can jeopardize the art industry. BBC reported that some fake artworks generated by text-to-image generation models won first place in an art competition, which caused the complaints of the involved artists.<sup>2</sup>

### 5.1.1 Our Contributions

There are multiple approaches to alleviate the concerns brought by advanced generation models. In particular, one can build a detector to automatically detect whether an image is real or fake. Moreover, one can build an attributor to attribute a synthetic image to its source generation model, so the model owner can be held responsible for its misuse. So far, various efforts have been made in this field; however, they only focus on traditional generation models, represented by GANs [122, 135, 35, 140]. To the best of our knowledge, no study has been done on text-to-image generation models. Also, whether the prompts used in such models can facilitate fake image detection and attribution remains unexplored.

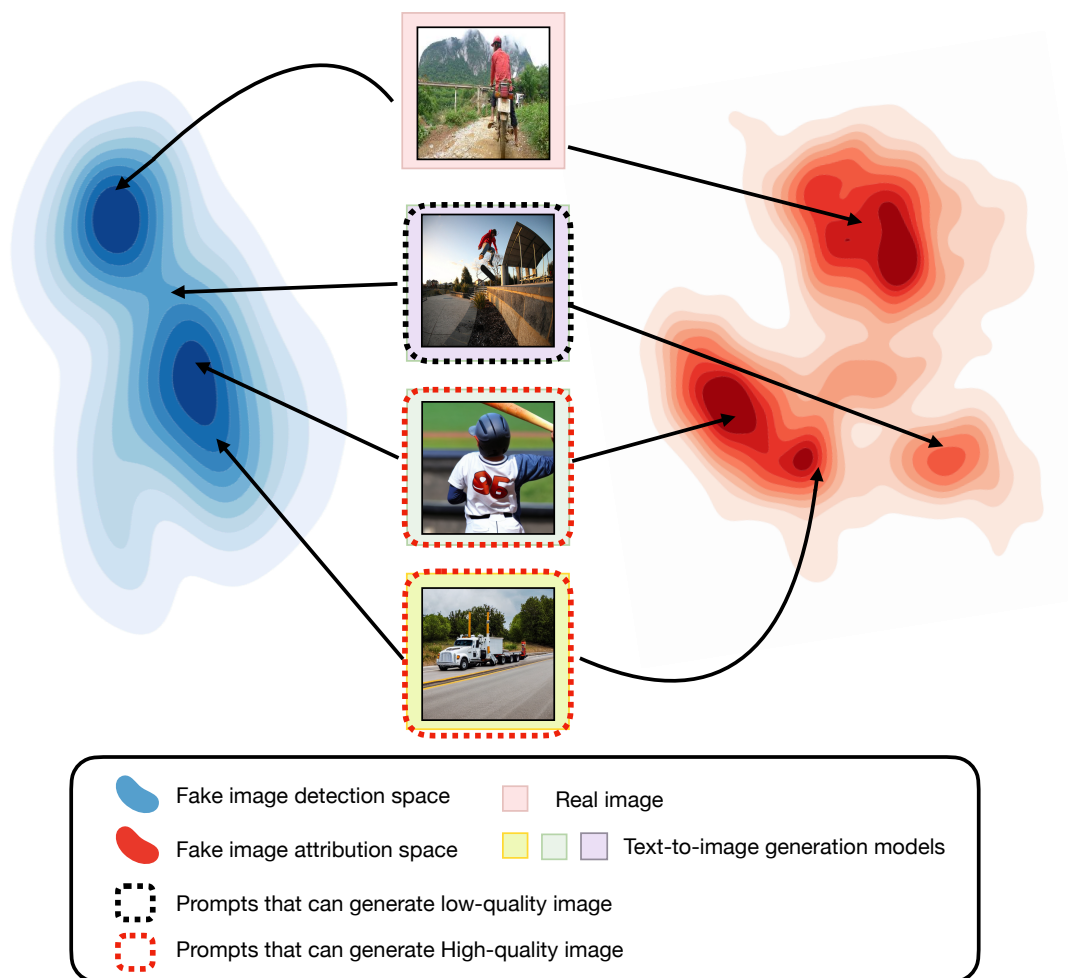
in this section, we present the first study on detecting and attributing fake images generated by text-to-image generation models. Concretely, we formulate the following three research questions (**RQs**).

- **RQ1:** Can we differentiate the fake images generated by various text-to-image generation models from the real ones, i.e., detection of fake and real images?
- **RQ2:** Can we attribute the fake images to their source text-to-image generation models, i.e., attribution of fake images to their sources?

---

<sup>1</sup><https://techcrunch.com/2022/08/12/a-startup-wants-to-democratize-the-tech-behind-dall-e-2-consequences-be-damned/>.

<sup>2</sup><https://www.bbc.com/news/technology-62788725>.



**Figure 5.1:** An illustration of our work, including fake image detection, fake image attribution, and prompt analysis. Note that the blue space represents the detection task where the two darkest colored areas represent real and fake images. The red space represents the attribution task where different darkest colored areas mean different algorithms. The dotted boxes represent the prompt analysis where different prompts lead to different quality images.

- **RQ3:** What kinds of prompts are more likely to generate authentic images?

**Methodology:** To differentiate fake images from real ones (**RQ1**), i.e., fake image detection, we train a binary classifier/detector. To validate the generalizability of the detector, we especially train it on fake images generated by *only one* model and evaluate it on fake images generated by *many other* models. We consider two detection methods, i.e., *image-only* and *hybrid*, depending on the detector’s knowledge. The image-only detector makes its decision solely based on the image itself. The hybrid detector considers both images and their corresponding prompts. Hybrid detection is a brand-new detection method, and it is designed specifically for detecting fake images

created by text-to-image generation models. Concretely, we leverage the image and text encoders of the CLIP model [93] to transfer an image and its prompt to two embeddings which are then concatenated as the input to the detector. Note that in the prediction phase, an image’s natural prompt may not be available. In such cases, we leverage an image captioning model BLIP [62] to generate the captions for the image. Note that in this section, we name the text used to generate fake images as prompts and text generated by BLIP to describe the images as captions.

To attribute a fake image to its source model (**RQ2**), we propose *fake image attribution* by training a multi-class classifier (instead of a binary classifier), and we name this classifier as an attributor. Specifically, the attributor is trained on fake images generated by multiple text-to-image generation models. Fake images from the same model are labeled as the same class. Moreover, we also establish the attributor by two methods, i.e., image-only and hybrid, which are the same as the detector to address **RQ1**.

Different from **RQ1** and **RQ2**, **RQ3** focuses on the impact of prompts on the authenticity of generated images. To this end, we conduct *prompt analysis* from semantic and structural perspectives. In the former, we design two semantic extraction methods to analyze the impact of prompt topics on the authenticity of fake images. More specifically, the first one directly uses the ground truth topics provided in the dataset for each prompt, and the second one automatically clusters the various prompts into different groups and extracts topics from these groups. From the structural perspective, we conduct the study based on the length of prompts and the proportion of nouns in prompts, respectively. Figure 5.1 presents an overview of our methods to address the three research questions.

**Evaluation:** We perform experiments on two benchmark prompt-image datasets including MSCOCO [68] and Flickr30k [133], and four popular pre-trained text-to-image generation models including Stable Diffusion [99], Latent Diffusion [99], GLIDE [85], and DALL·E 2 [95].

In fake image detection, extensive experimental results show that image-only detectors can achieve good performance in some cases, while hybrid detectors can always achieve better performance in all cases. For example, on MSCOCO [68], the image-only detector trained on fake images generated by Stable Diffusion can achieve an accuracy of 0.834 in differentiating fake images generated by Latent Diffusion from the real ones, while it can only achieve 0.613 and 0.2454 on GLIDE and DALL·E 2, respectively. Encouragingly, the hybrid detector trained on fake images from Stable Diffusion achieves 0.885/0.890/0.930 accuracy on Latent Diffusion/GLIDE/DALL·E 2 with natural prompts and 0.877/0.838/0.781 accuracy with BLIP-generated prompts. These results demonstrate that fake images from various models can indeed be distinguished from real images. We further extract a common feature from fake images generated by various models in Section 5.3.4, which implies the existence of a common artifact shared by fake images across various models.

In fake image attribution, our experiments show that both image-only and hybrid attributors can achieve good performance in all cases. Similarly, the hybrid attributor is better than the image-only one. For instance, the image-only attributor can achieve an accuracy of 0.815 in attributing fake images to the models we consider, while the hybrid

attributor can achieve 0.880 with natural prompts and 0.850 with BLIP-generated prompts. These results demonstrate that fake images can indeed be attributed to their corresponding text-to-image generation models. We further show the unique feature extracted from each model in Section 5.4.4, which implies that different models leave unique fingerprints in the fake images they generate.

We further extensively evaluate the robustness of our proposed detectors and attributors. In particular, since our detectors and attributors are essentially classifiers, we evaluate them using the most popular and severe attacks against classifiers, i.e., adversary example attacks. Experimental results show that our detectors and attributors can achieve significant robustness unless the adversarial noise is too visible, however, this visible noise can be easily detected by humans.

In prompt analysis, we first find that prompts with the topics of “skis,” and “snowboard” tend to generate more authentic images through our first semantic extraction method, which relies on the ground truth information from the dataset. However, by clustering various prompts over embeddings by sentence transformer [98], we find that prompts with the “person” topic can actually generate more authentic images. Upon further inspection, we discovered that most of the images associated with “skis” and “snowboard” are also related to “person.” These results indicate that prompts with the topic “person” are more likely to generate authentic fake images. From the structural perspective, our experiments show that prompts with a length between 25 and 75 enable text-to-image generation models to generate fake images with higher authenticity, while the proportion of nouns in the prompt has no significant impact.

**Implications:** in this section, we make the first attempt to tackle the threat caused by fake images generated by text-to-image generation models. Our results on detecting fake images and attributing them to their source models are encouraging. This suggests that our solution has the potential to play an essential role in mitigating the threats. We will share our source code with the community to facilitate research in this field in the future.

## 5.2 Datasets

We use the following two benchmark prompt-image datasets to conduct our experiments.

- **MSCOCO [68]:** MSCOCO is a large-scale objective, segmentation, and captioning dataset. It is a standard benchmark dataset for evaluating the performance of computer vision models. MSCOCO contains 328,000 images distributed in 80 classes of natural objects, and each image in MSCOCO has several corresponding captions, i.e., prompts. In this section, we consider the first 60,000 prompts due to the constraints of our lab’s computational resources.
- **Flickr30k [133]:** Flickr30k is a widely used dataset for research on image captioning, language understanding, and multimodal learning. It contains 31,783 images and 158,915 English prompts on various scenarios. All images are collected



**Table 5.1:** The text-to-image generation models, datasets, and the number/size of fake images we consider in this section. Note that the number of fake images from DALL-E 2 being low is due to its poor image generation efficiency.

Model	Dataset	Images	Image Size
SD	MSCOCO	59,247	512×512
	Flickr30k	13,231	512×512
LD	MSCOCO	31,276	256×256
	Flickr30k	17,969	256×256
GLIDE	MSCOCO	41,685	256×256
	Flickr30k	27,210	256×256
DALL-E 2	MSCOCO	1,028	256×256
	Flickr30k	300	256×256

from the Flickr website, and the prompts are written by Flickr users in natural language.

Note that the prompts from these datasets are also important as they will be used to generate fake images or serve as inputs for the hybrid classifiers (see Section 5.3 for more details).

In summary, the text-to-image generation models and datasets we consider in this section are listed in Table 5.1. Since different models are trained on images of different sizes and fake images usually appear in the real world at different resolutions. Therefore, we adopt the default settings of these available models and perform experiments on fake images of different sizes.

## 5.3 Fake Image Detection

In this section, we present our fake image detector to differentiate fake images from real ones (**RQ1**). We start by introducing our design goals. Then, we present how to construct the detector. Finally, we show the experimental results.

### 5.3.1 Design Goals

To tackle the threats posed by the misuse of various text-to-image generation models, the design of our detector should follow the following points.

- **Differentiating Between Fake and Real Images:** The primary goal of the detector is to effectively differentiate fake images generated by text-to-image generation models from real ones. Successful detection of fake images can reduce the threat posed by the misuse of these advanced models.
- **Agnostic to Models and Datasets:** As text-to-image generation models have undergone rapid development, it is likely that more advanced models will be

proposed in the future. As a result, it is difficult for us to collect all text-to-image generation models to build our detector. Moreover, building the detector on various models (even though we can collect many) inevitably leads to more resource consumption. Therefore, it is crucial to explore whether our detection based on very few text-to-image generation models is generalizable to other models. Also, since we have no knowledge of the distribution of prompts used to generate fake images, it is also important for our detector to identify fake images generated by prompts from other prompt-image datasets.

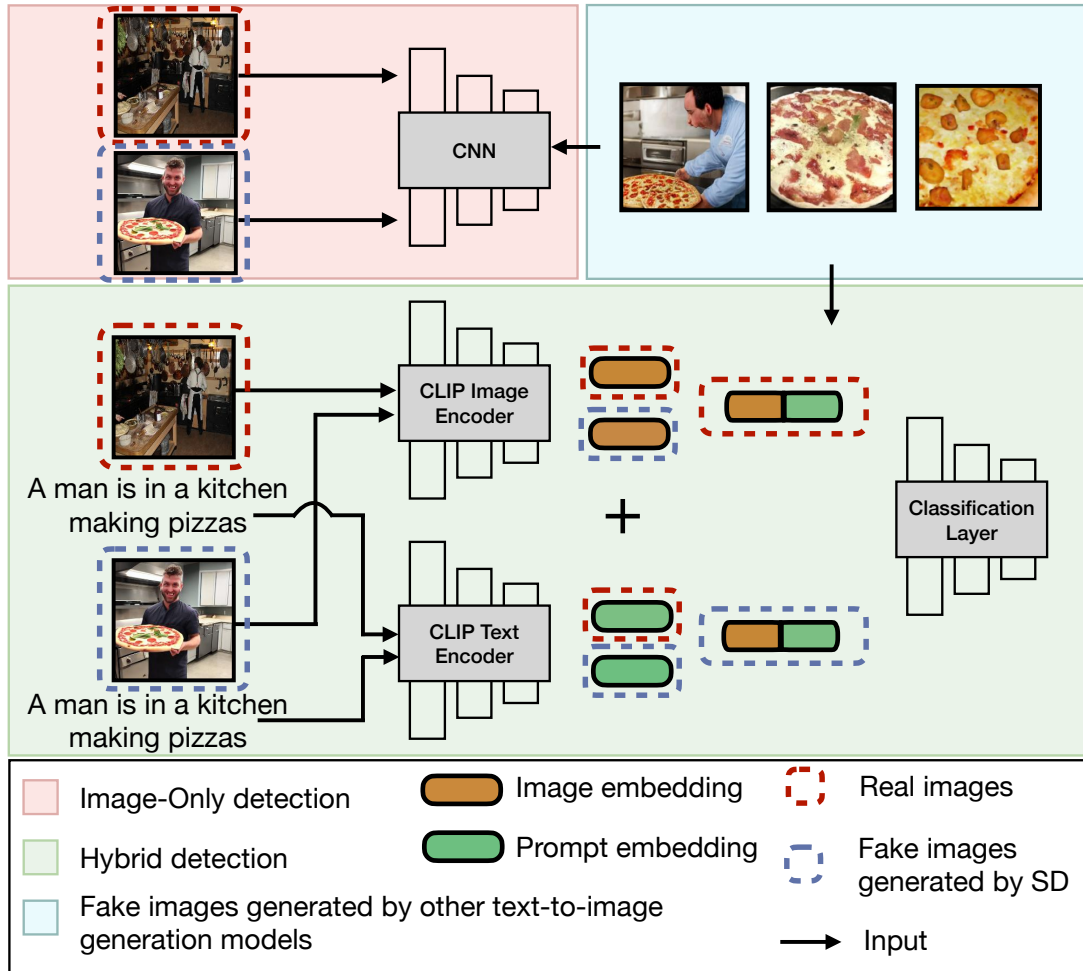
### 5.3.2 Methodology

To achieve the primary goal of differentiating fake images from real ones, we construct a detector by training a binary classifier. Furthermore, to make our detector agnostic to unseen models and datasets, we consider a more realistic and challenging scenario where the detector can collect fake images generated by only one text-to-image generation model given prompts from one dataset. The detector then trains its binary classifier on these fake/real images and evaluates its generalizability on fake images from other models and datasets.

In addition, based on the background knowledge available to the detector, we propose two different approaches to establish the detector, namely image-only and hybrid. The image-only detector accepts only images as input. In contrast, the hybrid detector accepts both images and their corresponding prompts as input. See Figure 5.2 for an illustration of how to conduct fake image detection.

**Image-Only Detection:** The red part of Figure 5.2 shows the work pipeline of our image-only detector. The process of training our image-only detector can be divided into three stages, namely, data collection, dataset construction, and detector construction.

- **Data Collection:** We first randomly sample 20,000 images from MSCOCO and treat them as real images for the next stage. Then, we use the prompts of these 20,000 images to query one model (we choose SD here) to get 20,000 fake images. In this way, our fake images are from one text-to-image generation model given prompts from one dataset, referred to as SD+MSCOCO.
- **Dataset Construction:** We label all fake images as 0 and all real images as 1. We then create a balanced training dataset containing a total of 40,000 images.
- **Detector Construction:** We build the detector (i.e., a binary classifier) that accepts an image as input and outputs a binary prediction, i.e., 0-fake or 1-real. Lastly, we train the detector from scratch with fake and real images in conjunction with classical training techniques. Note that we use ResNet18 [43] as our image-only detector’s architecture.



**Figure 5.2:** An illustration of fake image detection. The red part describes image-only detection. The green part describes hybrid detection. The blue part describes fake images generated by other text-to-image generation models.

After we have trained the detector, we evaluate the generalizability of the trained detector on images from other models, i.e., LD, GLIDE, and DALL·E 2, given prompts from the other dataset, i.e., Flickr30k. For completeness, we also include the detection results on fake images from the same model and/or the same dataset. Table 5.1 shows the total number of fake images generated by four models and two datasets. Besides the 20,000 images (out of 59,247) from SD+MSCOCO, which are used to train the detector, all the others are used to test the performance of the detector. Note that in all cases, we sample the same number of real images as the fake ones for training and testing the detector (and the attributor in Section 5.4).

**Hybrid Detection:** We now present the hybrid detector, which considers both images and their corresponding prompts. This is motivated by the observation that real images always carry a wide range of contents that the prompts cannot fully and faithfully

describe. However, since fake images are generated based on prompts, they may not contain additional content beyond what is described, i.e., not as informative as real images. Therefore, introducing prompts together with images enlarges the disparity between fake and real images, which in our opinion can contribute to differentiating between the two. We further show in Section 5.3.4 that the disparity between real and fake images is indeed huge from the prompt’s perspective. Note that using prompts as an extra signal for fake image detection is novel and unique to text-to-image generation models, as prompts do not participate in the image generation process of traditional generation models, like GANs.

The green part of Figure 5.2 shows the work pipeline of our hybrid detection. Specifically, the process of training our hybrid detector can also be divided into three stages, i.e., data collection, dataset construction, and detector construction.

- **Data Collection:** To collect the real and fake images, we follow the same step as the first step for the image-only detector.
- **Dataset Construction:** Since our hybrid detector takes images and prompts as input, we label all fake images and their corresponding prompts as 0 and label real images and their corresponding prompts as 1. Similarly, we then create a training dataset containing a total of 40,000 prompt-image pairs.
- **Detector Construction:** To exploit the prompt information, we take advantage of CLIP’s image encoder and text encoder as feature extractors to obtain high-level embeddings of images and prompts. Note that CLIP is trained on the image-text pair to align the embeddings of the image and text, thus creating a link between them. Then, we concatenate image embeddings and text embeddings together as new embeddings and use these embeddings to train a binary classifier, i.e., a 2-layer multilayer perceptron, as our detector.

To evaluate the trained hybrid detector, we need both images and their corresponding prompts. Typically, a user may attach a description to an image they post on the Internet. Therefore, we can directly consider this attached description as the prompt for the image. In our experiments, we adopt the original/natural prompts from the dataset to conduct the evaluation.

In a more realistic and challenging scenario where the detector cannot obtain the natural prompts, we propose a simple yet effective method to generate the prompts ourselves. Concretely, we leverage the BLIP [62] model (an image captioning model) to generate captions for the queried images and then regard these generated captions as the prompts for the images. Note that BLIP has been proven to show great performance in captioning real-world images by previous works [62].

### 5.3.3 Results

We now present the performance of our proposed image-only detection and hybrid detection for fake image detection.

**Image-Only Detection:** For a convincing evaluation, we adopt the existing work [122] on detecting fake images generated by various types of generation models, including GANs and low-level vision models [15, 26], as a baseline. The authors of [122] name their classifier as *forensic classifier*. Note that this forensic classifier is the state-of-the-art fake image detector for generation models, and the authors show that it has strong generalizability. For instance, it can achieve an accuracy of 0.946 on differentiating fake images generated by StarGAN [22], which is not considered during the model training, from real images.

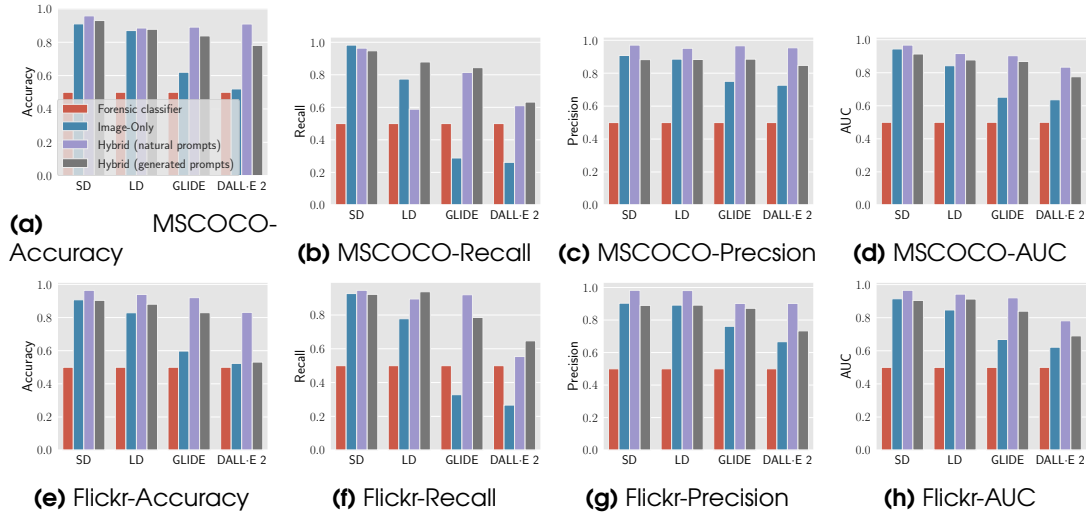
Figure 5.3 depicts the evaluation results. First of all, we can observe that the forensic classifier cannot effectively distinguish fake images (generated by text-to-image generation models) from real ones. In all cases, the forensic classifier only achieves an accuracy of 0.24, which is equivalent to a random guess. Based on this observation, we can conclude that the forensic classifier cannot be generalized to text-to-image generation models. We attribute this observation to the differences between traditional generation models and text-to-image generation models. This result also prompts the urgent need for counterpart solutions against the misuse of text-to-image generation models.

Furthermore, we can observe that the image-only detector performs much better in all cases than the forensic classifier. For example, the image-only detector can achieve an accuracy of 0.871 in distinguishing fake images generated by LD+Flickr30k (querying the prompts of Flickr30k to LD) from real images. We emphasize here that the image-only detector is trained only on fake images generated by SD+MSCOCO and has never seen fake images generated by other models given prompts from other datasets. We conjecture that this is due to some common properties shared by all fake images generated by text-to-image generation models (see Section 5.3.4 for more details).

Lastly, another interesting finding is the much larger variation in detection performance due to the effect of the model compared to the effect of the dataset. E.g., in Figure 5.3a, the image-only detector achieves an accuracy of 0.913 on SD but only 0.2426 on DALL·E 2. In contrast, comparing Figure 5.3a and Figure 5.3e, the image-only detector achieves very close accuracy on different datasets over all text-to-image generation models. We attribute this observation to the unique fingerprint of fake images generated by text-to-image generation models (see Section 5.4.4).

**Hybrid Detection:** Although the image-only detector achieves better performance in all cases compared to the forensic classifier, we acknowledge that the current detection performance is far from the design goal due to the lack of good performance on other models, such as GLIDE and DALL·E 2. As mentioned earlier, using prompts as an extra signal may boost the fake image detection performance.

We report the performance of our proposed hybrid detection in Figure 5.3. First, we can find that the hybrid detector can always achieve much better performance than the image-only detector, especially on models like GLIDE and DALL·E 2. For instance, the hybrid detector with natural prompts can achieve an accuracy of 0.909 on DALL·E 2+MSCOCO, which is much higher than the 0.2422 achieved by the image-only detector. Moreover, even without natural prompts, the hybrid detector with BLIP-generated prompts can still have a strong performance. For example, on fake images generated



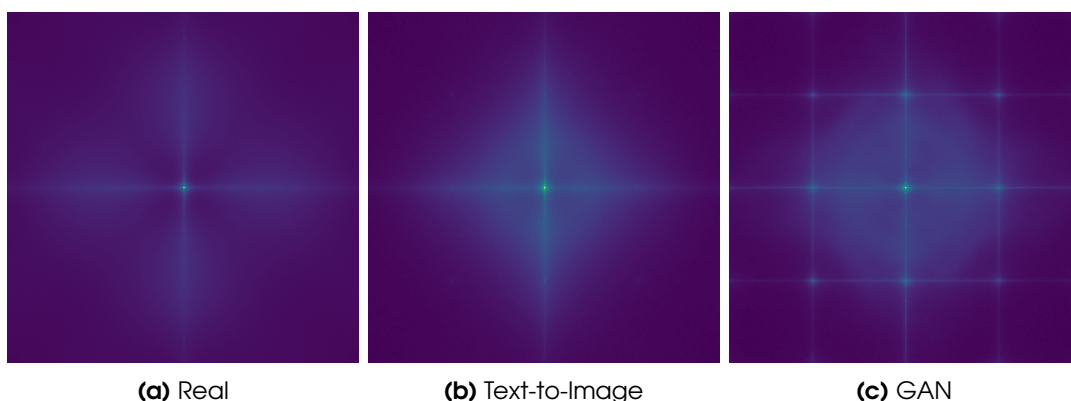
**Figure 5.3:** The performance of the forensic classifier and detectors.

by GLIDE+MSCOCO, the hybrid detector with natural prompts achieves an accuracy of 0.891, and encouragingly, the hybrid detector with BLIP-generated prompts also achieves a high accuracy of 0.838. These results indicate that introducing prompts together with images can indeed enlarge the disparity between fake and real images, which is beneficial to fake image detection. We further investigate in more depth why using the prompt as a new signal can improve detection performance (see Section 5.3.4 for detailed information).

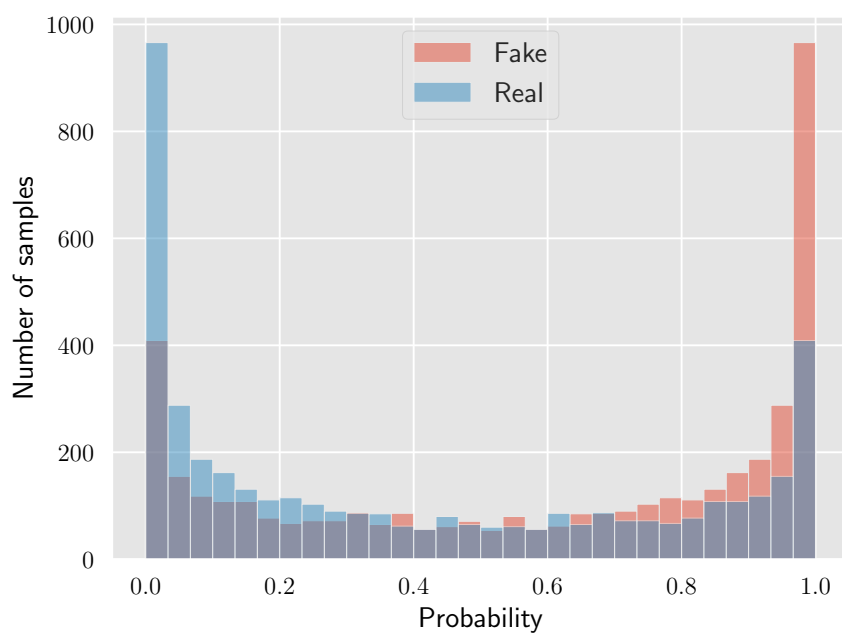
Besides, we can find that the performance of the hybrid detector on other models is much less influenced by prior knowledge of the known model than the image-only detector. For example, on the MSCOCO dataset, the hybrid detector with natural prompts can achieve an accuracy of 0.958 on SD, while the accuracy only drops to 0.909 on DALL-E 2. We can also find that the hybrid detector is not heavily influenced by the dataset compared to the image-only detector. For instance, on SD, the hybrid detector with generated prompts can achieve quite a similar accuracy between MSCOCO and FLickr30k (0.930 vs. 0.904). These results show that our proposed hybrid detector is strong regarding model and dataset independence.

### 5.3.4 Discussion

The above results fully demonstrate the effectiveness of our fake image detection. Next, we delve more deeply into the reasons for successfully distinguishing fake images from real ones. We conjecture that there exist some common properties shared by fake images from various text-to-image generation models. We verify this conjecture by visualizing the common artifact shared across fake images. Besides, based on the better performance achieved by hybrid detection, we further explore why additional prompt information can enhance detection performance. In the end, we also test whether our trained detector can be directly applied to fake images from other domains, in particular, fake artwork detection.



**Figure 5.4:** The visualization of frequency analysis on (a) real images, (b) fake images generated by text-to-image generation models, and (c) fake images generated by GAN.



**Figure 5.5:** The probability distribution of the connection between the real/fake images and the corresponding prompts.

**Artifact Visualization:** Inspired by Zhang et al. [140], we draw the frequency spectra of fake and real images. Frequency spectra can serve as a useful tool in characterizing the properties of images. In particular, fake images generated from the model have been observed to frequently exhibit anomalous global structure and color transitions that deviate from natural images. As the number of model-generated images grows, these anomalies become increasingly apparent. Therefore, frequency spectra can provide a valuable method to visualize and quantify the differences between real and generated images. For the four text-to-image generation models we consider in this section, we

randomly select 1,000 fake images from each model given prompts from MSCOCO. In total, we have obtained 4,000 fake images. Also, we collect 4,000 real images of the same prompts from MSCOCO. We then calculate the average of Fourier transform outputs of real and fake images, respectively. We leverage Fourier transform here due to its ability to reveal latent features of the given images.

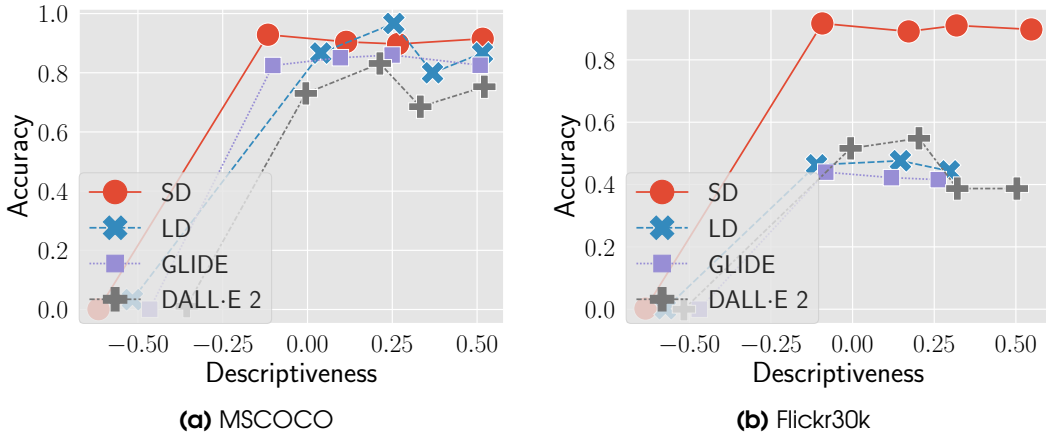
As shown in Figure 5.4, we can clearly observe that there are distinct patterns in real and fake images. Concretely, the central region of the fake image has higher brightness and more concentrated frequency spectra. This observation verifies the existence of the common artifact shared by the fake images generated by various text-to-image generation models. Note that we also visualize the frequency spectra of fake images from traditional GANs in Figure 5.4. We can observe a significant difference between the frequency spectra of fake images from text-to-image generation models and GANs. Even with that huge gap, our detector, built on the text-to-image diffusion model, can still achieve 0.863 accuracy on detecting fake images generated by StyleGAN, showing the generalization ability of our proposed method.

**Why Does Prompt Enhance Detection Performance:** We conduct a more in-depth study on why using prompts as a new signal can improve detection performance. As mentioned before, a prompt cannot completely reflect the contents of a real image. Meanwhile, a fake image is purely based on the prompt information. This suggests the connection between a fake image and its prompt is stronger than the connection between a real image and its prompt. This is essentially the reason why the hybrid detector has a better performance than the image-only detector.

To verify this, we first randomly sample 2,000 prompts from MSCOCO. For each prompt, we collect its corresponding real image from the dataset and let SD generate a fake image for it. Then, we rely on CLIP’s text encoder to transfer the prompt to an embedding and CLIP’s image encoder to transfer the real and fake images to two embeddings, respectively. Then, we calculate two cosine similarities, one is between the prompt’s embedding and its real image’s embedding, and the other is between the prompt’s embedding and its fake image’s embedding. Finally, the two cosine similarities are normalized into a probability distribution via a softmax function [93]. Higher probability implies a stronger connection between the image and the prompt. Figure 5.5 shows the similarity distribution between the 2,000 prompts and the real/fake images. We can see that the similarity between the fake image and the corresponding prompt is closer than that between the real image and the same prompt, leading to a clear gap in the similarity distribution between fake and real images. This verifies our aforementioned intuition. Furthermore, we can also conclude that it is not the prompt information itself that enhances the performance of the detector, but the prompt information can be exploited as an extra “anchor” to provide a new signal to distinguish between real and fake images. Such signals can be effectively captured by a multilayer perceptron.

**Combination of Different Text-to-Image Models:** We further demonstrate the effectiveness of our hybrid detector trained on fake images generated from different models. More specifically, we use Stable Diffusion and DALL·E 2 to generate fake images as the training set for our hybrid detector, which is then evaluated on all four





**Figure 5.6:** The performance of hybrid detectors with generated prompts in terms of the prompts’ descriptiveness. The descriptiveness is grouped into five equally sized bins.

text-to-image generation models that we consider in this section. autoreftab:combine-performance shows that, encouragingly, our detector can perform better when trained on more models than on only one model.

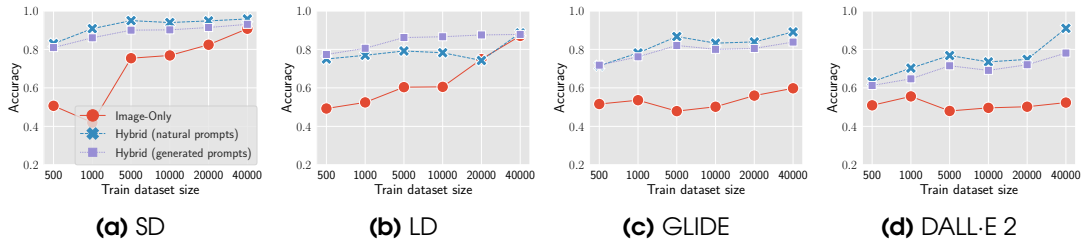
**Table 5.2:** Performance of hybrid detector trained on mixed fake images from Stable Diffusion and DALL-E 2

Models	Accuracy	AUC	F1 Score
SD	0.933	0.923	0.793
LD	0.915	0.951	0.879
GLIDE	0.891	0.909	0.883
DALL-E 2	0.944	0.877	0.904

### 5.3.5 Ablation Study

**Impact of Generated Prompt:** In hybrid detection with generated prompts, we rely on the BLIP model. Here, we explore whether the quality of the BLIP-generated prompts affects the detection performance. To measure the quality of the generated prompts by BLIP, we leverage a new term called prompt descriptiveness [105, 78, 30, 31]. Prompt descriptiveness can be quantitatively measured by computing the cosine similarity between a prompt’s embedding and its image’s embedding generated by CLIP.<sup>3</sup> Such similarity demonstrates the degree of match between the generated prompts and the given images. Figure 5.6 depicts the relation between the detection performance and the descriptiveness of the generated prompts. We can see that in

<sup>3</sup>Note that the descriptiveness is the same as the one used in the previous analysis regarding why prompts can enhance detection performance.



**Figure 5.7:** The performance of detectors in terms of the training dataset size on SD+MSCOCO. We conduct the evaluation on (a) SD+MSCOCO, (b) LD+MSCOCO, (c) GLIDE+MSCOCO, and (d) DALL-E 2+MSCOCO.

general, higher descriptiveness leads to better detection performance. Also, after a certain descriptiveness value, the detection performance becomes stable across all models and datasets. This shows the robustness of using BLIP-generated prompts in our hybrid detector.

**Impact of Training Dataset Size:** In this section, we explore the impact of the training dataset’s size on the performance of our proposed fake image detection. More concretely, for each text-to-image generation model, we train the detector on fake images from SD+MSCOCO by varying the size of the training dataset from 500 to 40,000 (half is real, half is fake). Note that the default size we use in the previous evaluation is 40,000.

We report the detection performance in terms of the training dataset size in Figure 5.7. As expected, the performance of different detectors is indeed affected by the size of the training dataset, and the general trend is that all the detectors perform better with the increase in the training dataset size. For instance, as shown in Figure 5.7b, when the training dataset size is 1,000, the hybrid detector can achieve an accuracy of 0.792 while the accuracy can be improved to 0.885 when the training dataset size is 40,000. More encouragingly, we can also find that the hybrid detector achieves strong performance even with a small training dataset of only 500 images, which is much fewer than 40,000 images. For example, in Figure 5.7a, the hybrid detector achieves a high accuracy of 0.830 with only 500 training images. Finally, we again find that the hybrid detector performs much better than the image-only detector, even with different sizes of the training dataset. For example, in Figure 5.7c, the hybrid detector achieves an accuracy of 0.714 with 500 training images, while the image-only detector achieves only 0.2423 with 40,000 training images. These results again demonstrate that introducing prompts together with images is beneficial to differentiate between fake and real images.

### 5.3.6 Takeaways

In summary, to answer **RQ1**, we propose fake image detection by training a binary detector to differentiate fake images generated by text-to-image generation models from real images. Specially, we propose two methods to construct the binary detector, namely image-only and hybrid. Our evaluation shows that the fake images from various models can indeed be differentiated from the real ones. Moreover, the hybrid detector can obtain

much better performance compared to the image-only detector, which demonstrates that introducing prompts together with images can indeed amplify the differences between fake and real images.

## 5.4 Fake Image Attribution

The previous section has shown that fake image detection, especially the hybrid detection we have proposed, can achieve remarkable performance. In this section, we explore whether fake images generated by various text-to-image generation models can be attributed to their source models, i.e., fake image attribution. We start by introducing our design goals. We then describe how to construct the fake image attributor. Finally, we present the evaluation results.

### 5.4.1 Design Goals

To attribute fake images to their source models, we follow two design goals.

- **Tracking Sources of Fake Images.** The primary goal of fake image attribution is to effectively attribute different fake images to their source generation models. The aim of attribution is to let a model owner be held responsible for the model’s (possible) misuse. Previously, fake image attribution has been studied in the context of traditional generation models, like GANs [135].
- **Agnostic to Datasets.** In the real world, a fake image can be generated by a text-to-image generation model based on a prompt from any distribution. Therefore, to be more practical, the attribution should be independent of the prompt distribution.

### 5.4.2 Methodology

To attribute the fake images to their sources, we construct fake image attribution by training a multi-class classifier, referred to as an attributor, with each class corresponding to one model. As aforementioned, the attributor should be agnostic to datasets; thus, we establish the multi-class classifier based on prompts from only one dataset, e.g., MSCOCO, and test it on prompts from other datasets like Flickr30k.

Similar to fake image detection, we propose two different approaches to establish the attributor, namely image-only and hybrid. The image-only attributor accepts only images as input, and the hybrid attributor accepts both images and their corresponding prompts as input.

**Image-Only Attribution:** The process of establishing our image-only attributor can also be divided into three stages, namely, data collection, dataset construction, and attributor construction.

- **Data Collection:** We first randomly sample 20,000 images from MSCOCO as real images. Then, we use the prompts of these 20,000 images to query each model

to get 20,000 fake images accordingly. Here, we adopt SD, LD, and GLIDE to generate fake images. In total, we have obtained 60,000 fake images. The reason we do not consider DALL·E 2 is that we will use DALL·E 2 for the experiments regarding adaptation to other models (see Section 5.4.4).

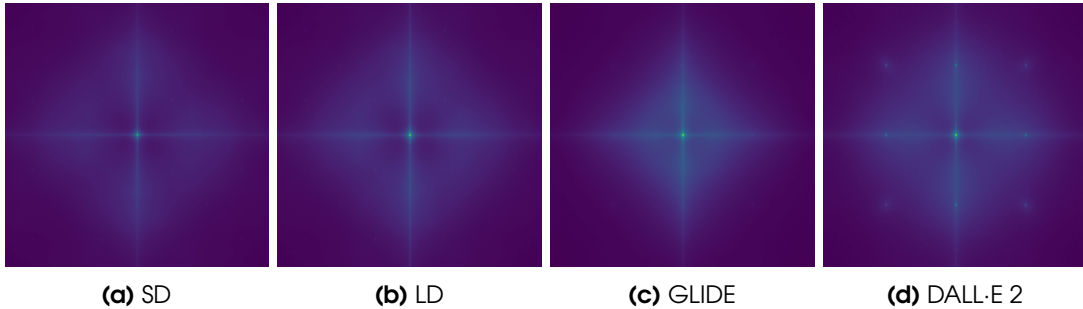
- **Dataset Construction:** We label all real images as 0 and all fake images from the same model as the same class. Concretely, we label the fake images from SD/LD/GLIDE as 1/2/3. We then create a training dataset containing a total of 80,000 images with four classes.
- **Attributor Construction:** We build the fake image attributor, i.e., a multi-class classifier, that accepts images as input and outputs the multi-class prediction, i.e., 0-real, 1-SD, 2-LD, or 3-GLIDE. We train the attributor from scratch using the created training dataset in conjunction with classical training techniques. Similar to the fake image detector, we leverage ResNet18 [43] as the attributor’s architecture.

After we have trained the attributor, we evaluate the performance of the trained attributor for attributing images from various sources (i.e., real, SD, LD, and GLIDE) given prompts from the other dataset (i.e., Flickr30k). For testing the attributor, we sample the same number of images for all four classes, i.e., 10,000 each and 40,000 in total.

**Hybrid Attribution:** The previous evaluation in fake-image detection has demonstrated the superior performance of the hybrid detector, verifying that introducing prompts together with images can amplify the differences between fake and real images. We now conduct the study to investigate whether a similar enhancement can be observed in the case of hybrid attribution.

The hybrid attributor is quite similar to the above image-only attributor, which also consists of three stages, i.e., data collection, dataset construction, and attributor construction.

- **Data Collection:** To collect the images from various sources, we follow the same step as the first step for the image-only attributor.
- **Dataset Construction:** Since our hybrid attributor takes images and prompts as input, we label all real images with their corresponding prompts as 0 and all fake images from the same model with their corresponding prompts as the same class. Similarly, we then create a training dataset containing a total of 80,000 prompt-images pairs with four classes.
- **Attributor Construction:** To exploit the prompt information, we again use CLIP’s image encoder and text encoder as feature extractors to obtain high-level



**Figure 5.8:** The visualization of frequency analysis on fake images generated by (a) SD, (b) LD, (c) GLIDE, and (d) DALL-E 2.

embeddings of images and prompts. Then, we concatenate image embeddings and text embeddings together as new embeddings and use these embeddings to train a multi-class classifier, which is also a 2-layer multilayer perceptron, as our attributor.

In order to evaluate the trained hybrid attributor, we need images and their corresponding prompts. We again consider two scenarios here, one in which we can directly obtain prompts for the images from the dataset and the other in which we can only generate prompts for the images relying on BLIP.

### 5.4.3 Results

In this section, we present the performance of our proposed two types of fake image attribution.

**Image-Only Attribution:** We report the performance of image-only attribution in Table 5.3. Note that the random guess for the 4-class classification task is only 0.245. We can find that our proposed image-only attributor can achieve remarkable performance. For instance, the image-only attributor can achieve an accuracy of 0.864 on images from various sources given the prompts sampled from MSCOCO. These results indicate that the fake images can be effectively attributed to their corresponding text-to-image generation models. We further show the unique feature extracted from each model in Section 5.4.4, which implies that different models may leave unique fingerprints in the fake images they generate.

Further, the image-only attributor can also achieve a high accuracy of 0.863 on images from various source models given the prompts sampled from the other dataset Flickr30k. Note that we construct the attributor based on MSCOCO only. This result indicates that our proposed image-only attribution is agnostic to datasets.

**Hybrid Attribution:** Table 5.3 also depicts the performance of our proposed hybrid attribution. We can clearly see that hybrid attribution, no matter with or without natural prompts, achieves better performance than image-only attribution regardless of the dataset. These results demonstrate once again that fake images can be successfully attributed to their corresponding text-to-image generation models. Also, they verify that using prompts as an extra signal can improve attribution performance.

**Table 5.3:** The performance of image-only attributors and hybrid attributors.

	MSCOCO	Flickr30k
image-Only	0.864	0.863
Hybrid (natural prompts)	0.936	0.933
Hybrid (generated prompts)	0.903	0.892

#### 5.4.4 Discussion

The above evaluation demonstrates the effectiveness of our fake image attribution. We conjecture that each text-to-image generation model leaves a unique fingerprint in the fake images it generates. Next, we verify this conjecture by visualizing the fingerprints of different models. Besides, in the previous evaluation, the training and testing images for our attributor are disjoint but generated by the same set of text-to-image generation models. We further explore how to adapt our attributor to other models that are not considered during training.

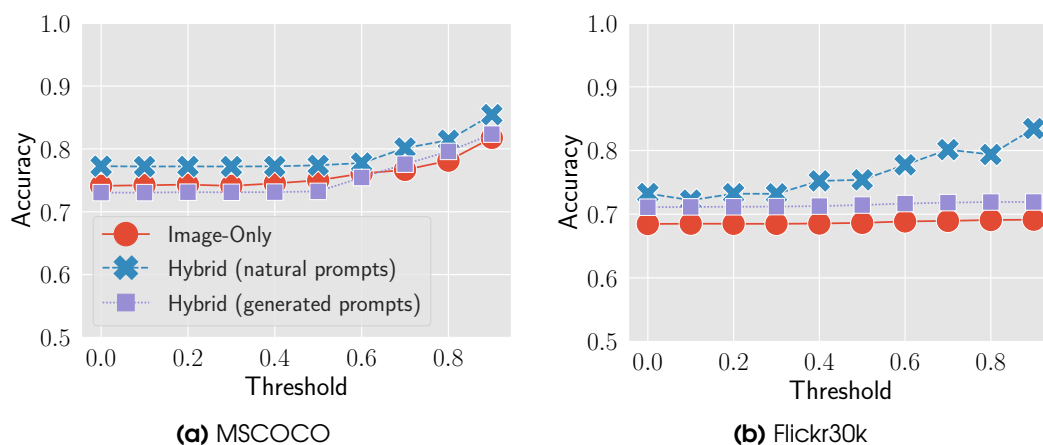
**Fingerprint Visualization:** Similar to visualizing the shared artifact across fake images (see Section 5.3.4), we also draw the frequency spectra of different text-to-image generation models built on MSCOCO. For each text-to-image generation model, we randomly select 2,000 fake images and then calculate the average of their Fourier transform outputs.

As shown in Figure 5.8, we can clearly observe that there are distinct patterns in images generated by different text-to-image generation models, especially in GLIDE and DALL·E 2. We can also find that the frequency spectra of SD is similar to that of LD, which can explain why the image-only detector built on SD can also achieve very strong performance on LD (see Figure 5.3). The reason behind this is that SD and LD follow similar algorithms, although trained on different datasets and different model architectures. In conclusion, the qualitative evaluation verifies that each text-to-image generation model has its unique fingerprint.

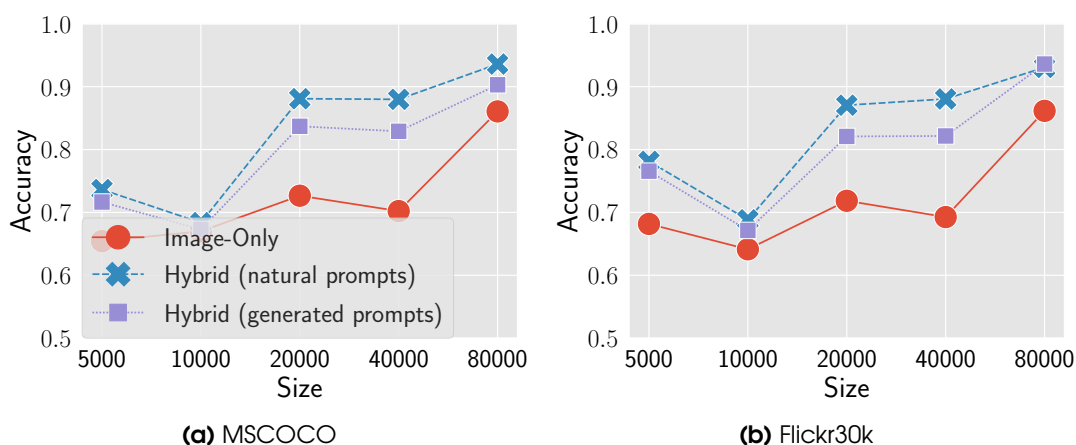
**Adaptation to Unseen Models:** In previous experiments, we evaluate attribution on fake images generated by models considered during training. However, there are instances when we encounter fake images that are not from models involved in training, i.e., unseen models. Next, we explore how to adapt our attributor to unseen models.

To this end, we propose a simple yet effective approach named confidence-based attribution. The key idea is to attribute the unconfident samples from the attributor prediction, i.e., lower than a pre-defined threshold, to unseen models. Here, all unseen models are considered as one class.<sup>4</sup> In our evaluation, we treat DALL·E 2 as one unseen model (as mentioned before). We first divide the datasets into training, validation, and testing parts. To find a suitable threshold, we experimented with values from 0 to 1 in a step of 0.1 on our validation part. Note that here we extend the evaluation from four classes to five: real, SD, LD, GLIDE, and unseen; the testing dataset is still balanced.

<sup>4</sup>In the current version, our approach cannot differentiate fake images from multiple unseen text-to-image generation models. We will leave this as future work.



**Figure 5.9:** The performance of attributors on an unseen dataset DALL-E 2 in terms of different thresholds. We conduct the evaluation on (a) MSCOCO and (b) Flickr30k.



**Figure 5.10:** The performance of attributors in terms of the training dataset size on MSCOCO. We conduct the evaluation on (a) MSCOCO and (b) Flickr30k.

Also, the attributor remains unchanged, i.e., it is still a 4-class classifier. Figure 5.9 shows that both image-only and hybrid attributors can achieve good performance in all cases. Encouragingly, the 0.9 thresholds can lead to the best attribution performance. We also find that after finding the best threshold in evaluation datasets, one can still achieve 0.855 accuracy on the test part. Moreover, we can still conclude that hybrid attribution can achieve better performance than image-only attribution in both settings. These results indicate that with a simple modification, our attribution can be adapted to unseen models.

#### 5.4.5 Ablation Study

**Impact of Training Dataset Size:** Here, we explore the effect of the training data size on attribution performance. The experimental results are depicted in Figure 5.10.

We can see that the size of training data indeed has a great influence on attribution performance. For example, when the training dataset size is 5,000, the hybrid attributor can achieve an accuracy of 0.736, while the accuracy can be improved to 0.946 when the training dataset size increases to 80,000. Besides, we can find that the hybrid attributor requires less data to achieve a stable performance compared to the image-only attributor. For example, hybrid attribution achieves a huge performance improvement from 10,000 to 20,000 in training dataset size, while for image-only attribution, a similar improvement happens when the training dataset size increases from 40,000 to 80,000. From this phenomenon, we can conclude that hybrid attribution achieves good performance even with a small amount of training data.

### 5.4.6 Takeaways

In summary, to answer **RQ2**, we propose image-only attribution and hybrid attribution to track the source of fake images. Empirical results indicate that fake images can be successfully attributed to their sources. We further conduct a qualitative analysis that verifies the existence of unique fingerprints left by different text-to-image generation models in their generated images. Also, we show that our method can be easily adapted to other unseen models.

## 5.5 Robustness Analysis

Previous evaluations have demonstrated the excellent performance of our proposed methods in detection and attribution tasks. In this section, we further conduct a systematic robustness analysis of our proposed methods. Specially, we evaluate the robustness of detectors and attributors against adversarial example attacks, which are the most common and severe attacks against machine learning models. We leverage three representative adversarial example attacks, namely FGSM [37], BIM [58], and DI-FGSM [129] to conduct the robustness analysis. Furthermore, given that our hybrid detector and attributor consider both the image and its corresponding prompt, we propose HybridFool, which maximizes the distance between the embedding of a given image and the prompt by adding perturbations to the image. In the following, we first present each adversarial example attack we consider in this robustness analysis. Then, we show the evaluation results.

### 5.5.1 Adversary Example Attacks

In this section, we present the methodologies for each of the adversarial example attacks considered in this section, namely FGSM [37], BIM [58], DI-FGSM [129], and HybridFool.

**FGSM [37]:** Fast gradient sign method (FGSM) is one of the most representative attacks in adversary example domain. FGSM can be formed as the following equations:

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y)) \quad (5.1)$$



$x$  is the input,  $y$  is the true label of  $x$ ,  $\theta$  is our detector’s or attributor’s parameters, and  $J(\theta, x, y)$  is its cost function. The main idea of FGSM is to maximize the loss for the given model by adding a small and scaled version of the sign of the gradient to the original image.

**BIM [58]:** Basic iterative method (BIM) is another popular adversary example attack, which actually is an extension of FGSM. Instead of computing the gradient in one epoch, BIM uses multiple interactions with smaller step sizes to generate the adversarial noise, which can be formulated as follows:

$$x^{(0)} = x, \quad x^{(t+1)} = x^{(t)} + \alpha \cdot \text{sign}(\nabla x' J(\theta, x^{(t)}, y)) \quad (5.2)$$

**DI-FGSM [129]:** FGSM and BIM are white-box attacks where the adversary must access the model’s gradient. Here, we further consider black-box attacks where the adversary does not need to access the model’s gradient, i.e., Diverse Inputs Iterative Fast Gradient Sign Method (DI-FGSM [129]). The main idea behind DI-FGSM is to train a surrogate model that mimics the victim model’s property and then search for the adversarial noise on the surrogate model. In this way, the adversarial noise can also mislead the victim models.

The DI-FGSM can be described using the following equation:

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x J_s(\theta_s, x + \eta, y)) \quad (5.3)$$

Here,  $x$  is the input,  $y$  is the true label of  $x$ ,  $\theta_s$  is the surrogate detector’s or attributor’s parameters,  $J_s(\theta, x, y)$  is its cost function,  $\epsilon$  is the perturbation magnitude, and  $\eta$  is a random noise vector.

**HybridFool:** Note that our hybrid detectors and attributors accept images and their corresponding prompts as input. Thus, in addition to existing representative attacks that only mislead classification by adding noises to images, we propose HybridFool, a newly designed black-box attack that additionally aims to maximize the distance between the embeddings of the images and their corresponding prompts by adding noises. Our proposed HybridFool can be formulated as follows:

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x J(E_i(x), E_t(t))) \quad (5.4)$$

Where  $J(E_i(x), E_t(t))$  here represents the distance function,  $E_i$  is CLIP’s image encoder,  $E_t$  is CLIP’s text encoder, and  $t$  the corresponding prompt of  $x$ .

### 5.5.2 Experimental Results

In this section, we present the robustness performance of our proposed detectors and attributors by reporting the accuracy under various adversarial example attacks. Note that higher accuracy means better robustness and vice versa.

Table 5.4 displays the accuracy of our detectors and attributors under various noise levels applied by each attack. We can find that our proposed detectors and attributors demonstrate remarkable robustness against adversarial example attacks unless the added

**Table 5.4:** The performance of our proposed detectors and attributors under various adversary example attacks.

Attacks	Noise	Detection	Attribution
FGSM	0.001	0.963	0.896
	0.005	0.691	0.2493
	0.01	0.367	0.340
BIM	0.001	0.963	0.812
	0.005	0.691	0.478
	0.01	0.367	0.392
DI-FGSM	0.001	0.938	0.833
	0.005	0.742	0.463
	0.01	0.691	0.2468
HybridFool	0.001	0.846	0.811
	0.005	0.329	0.495
	0.01	0.323	0.390

noise level is sufficiently large, e.g., 0.005 and 0.01. However, it is worth noting that FGSM, BIM, and HybridFool are white-box attacks, meaning they require access to the internal gradients to execute successfully. Since these gradients of our detectors and attributors are not accessible in the real world, these attacks cannot be executed in practical scenarios. Furthermore, even for black-box attacks like DI-FGSM, the noise added to the images is clearly visible when the noise level is set to 0.005 and 0.01. This renders the images unrealistic and makes them easy to detect, further highlighting the robustness of our proposed detector and attributor against adversarial example attacks.






## 5.6 Prompt Analysis

One of the major differences between text-to-image generation models and traditional generation models is that the former takes a prompt as input. In this section, we investigate which kinds of prompts are more likely to generate authentic images (**RQ3**). To answer this question, we perform a comprehensive prompt analysis from semantic and structural perspectives. Note that when exploring the impact of prompts, it is not appropriate to use a hybrid detector that accepts prompts as input to detect fake images. In contrast, the image-only detector is independent of the given prompt. Thus, we conducted prompt analysis on image-only detectors.






### 5.6.1 Semantics Analysis

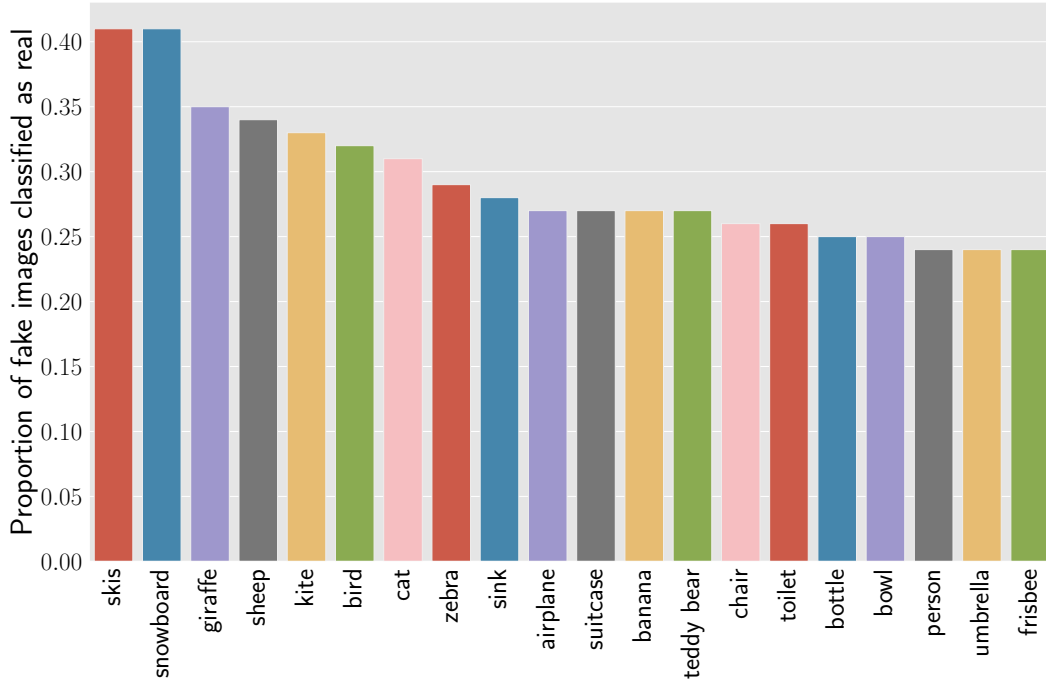
We first conduct semantic analysis on prompts based on their topics. Concretely, we group prompts into different clusters by topic. Then, for each cluster/topic, we calculate the proportion of the corresponding fake images being classified as real images by our image-only detector (Section 5.3). A cluster with a higher proportion indicates the

**Table 5.5:** Top five prompts which can generate the most real images, determined by the image-only detector. Gray cells mean the prompt mainly describe the details of the subject.

Rank	Image	Description
Top1		A dog hanging out of a side window on a car
Top2		A pan filled with food sitting on a stove top
Top3		A birthday cake with English and Chinese characters
Top4		There is an elephant-shaped figure next to other decorations
Top5		there is a cake and donuts that look like a train

**Table 5.6:** Top five prompts which can generate the most fake images, determined by the image-only detector. Gray cells mean the prompt mainly describe the environment where the subject is located.

Rank	Image	Description
Top1		A green bus is parked on the side of the street
Top2		THERE IS A ZEBRA THAT IS EATING GRASS IN THE YARD
Top3		I sign that indicates the street name posted above a stop sign
Top4		A group of skiers as they ski on the snow
Top5		A bench is surrounded by grass and a few flowers



**Figure 5.11:** The top twenty topics of prompts in terms of the proportion of the corresponding generated fake images being classified as real by the image-only detector.

prompts with the underlying topic have a higher chance of generating authentic images. It is worth noting that we define authenticity as the confidence score of the detector in determining whether a given fake image can be classified as real. As we focus on the authenticity of an image itself, we adopt the image-only detector instead of the hybrid detector. Note that our analysis is conducted on fake images generated by SD given prompts from MSCOCO.

We first utilize a straightforward method to group prompts relying on the topics provided by MSCOCO. In total, there are 80 topics in MSCOCO. We select the top twenty topics with the highest real image proportion decided by the image-only detector and report the results in Figure 5.11. We can clearly observe that among the top twenty topics, “skis” and “snowboard” are ranked the highest. Also, there are many topics related to animals, such “sheep,” “cat,” “zebra,” etc.

Though the topics from MSCOCO are straightforward, they may not be able to represent the full semantics of the images. Therefore, we take another approach. Specifically, we take advantage of sentence transformer [98] based on BERT [28] to generate embeddings for the prompts and then group the embeddings with DBSCAN [32], an advanced clustering method. The advantage of the second approach is that it can implicitly reflect the in-depth semantics of the prompts, which is also a common practice



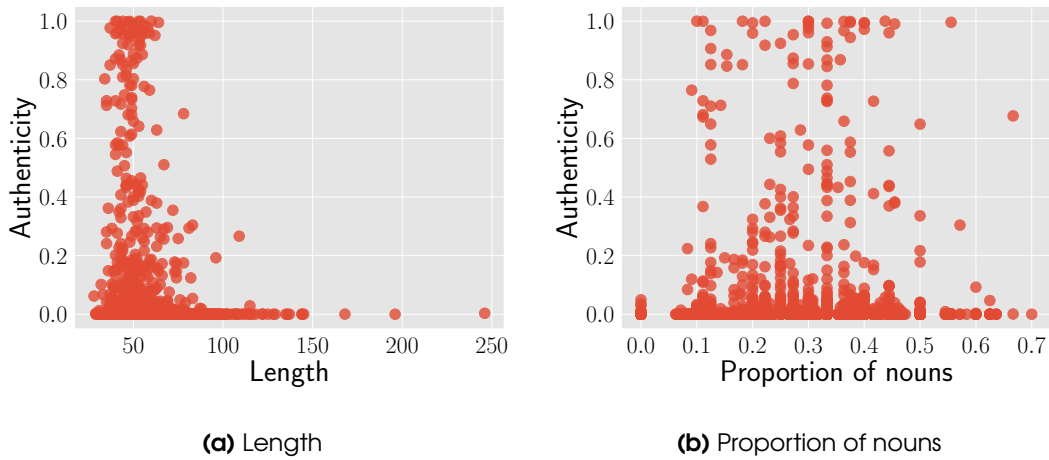
**Figure 5.12:** Examples of fake images generated by SD given prompts with topics “skis” and “snowboard.”

in the natural language processing literature. We also manually check the cluster results. It can be seen that different clusters share similar objects and similar scenes. However, the disadvantage of this approach is that the concrete topic of each cluster needs to be manually summarized. By manually checking, the cluster with the highest real image proportion is related to the topic “person.” Ostensibly, this is different from the results of the first approach (“skis” and “snowboard” ranked the highest), which is based on the topics provided by MSCOCO. However, by manually checking fake images by prompts with topics “skis” and “snowboard,” we discover that most of them depict “person” as well. We show some examples in Figure 5.12. This indicates the prompts related to “person” are likely to generate authentic fake images.

We further extract the top 5 prompts that can generate the most real and fake images, respectively, according to the image-only detector, and list them in Table 5.5 and Table 5.6. We can also find that detailed descriptions of the subjects contribute to the generation of authentic images. For example, of the top five real prompts, four provide a detailed description of the subject, while four of the top five fake prompts describe the environment where the subject is located, rather than the subject itself. In the future, we plan to investigate in-depth the relationship between the prompts’ semantics and the generated images’ authenticity.

### 5.6.2 Structure Analysis

After semantic analysis, we now conduct the structure analysis. Specifically, we study prompt structure from two angles, i.e., the length and the proportion of nouns. The length of the prompt reflects the prompt complexity. The proportion of nouns is related to the number of objects appearing in the fake image. Here, we use Natural Language Toolkit (NLTK) [9] to compute the proportion of nouns in a prompt.



**Figure 5.13:** The relationship between the length\proportion of nouns in a prompt and the corresponding image’s authenticity.

In our experiments, we randomly select 5,000 prompts from MSCOCO and then feed these prompts to SD to generate fake images. Results are shown in Figure 5.13. We can see from Figure 5.13a that both extremely long and short prompts cannot generate authentic images. In addition, almost all high-authenticity images are generated by prompts with lengths between 25 to 75. On the other hand, Figure 5.13b shows that the proportion of nouns in prompts does not have a significant impact on fake images’ authenticity.

### 5.6.3 Takeaways

In summary, we conduct semantic analysis and structure analysis to study which types of prompts are more likely to drive text-to-image generation models to generate fake images with high authenticity. Empirical results demonstrate that a prompt with the topic “person” or length between 25 and 75 is more likely to produce authentic images, thus leading to difficulties in detection by our designed detectors.

## 5.7 Conclusion

In this section, we delve into three research questions concerning the detection and attribution of fake images generated by text-to-image generation models. To solve the first research question of whether we can distinguish fake images apart from real ones, we propose fake image detection. Our fake image detection consists of two types of detectors: an image-only detector and a hybrid detector. The image-only detector utilizes images as the input to identify fake images, while the hybrid detector leverages both image information and the corresponding prompt information. In the testing phase, if the hybrid detector cannot obtain the natural prompt of an image, we take advantage of BLIP, an image captioning model, to generate a prompt for the image. Our extensive experiments show that while an image-only detector can achieve strong

performance on certain text-to-image generation models, a hybrid detector can always have better performance. These results demonstrate that fake images generated by different text-to-image generation models share common features. Also, prompts can serve as an extra “anchor” to help the detector better differentiate between fake and real images.

To tackle the second research question, we conduct the fake image attribution to attribute fake images from different text-to-image generation models to their source models. Similarly, we develop two types of multi-class classifiers: an image-only attributor and a hybrid attributor. Empirical results show that both image-only attributor and hybrid attributor have good performance in all cases. This implies that fake images generated by different text-to-image generation models enjoy different properties, which can also be viewed as fingerprints.

Finally, we address the third research question, i.e., which kinds of prompts are more likely to generate authentic images? We study the properties of prompts from semantic and structural perspectives. From the semantic perspective, we show that prompts with the topic “person” can achieve more authentic fake images compared to prompts with other topics. From the structural perspective, our experiments reveal that prompts with lengths ranging from 25 to 75 allow text-to-image generation models to create more authentic fake images.

Overall, this work presents the first comprehensive study of detecting and attributing fake images generated by state-of-the-art text-to-image generation models. As our empirical results are encouraging, we believe our detectors and attributors can play an essential role in mitigating the threats caused by fake images created by the advanced generation models.





# 6

## Related Work



## 6.1 Backdoor Attacks

Backdoor attacks, as one of the major threats to ML systems, have been widely studied. BadNets [39] is the first work to show that the adversary can insert a backdoor into the machine learning model via poisoning training datasets. Then, targeted backdoor [21] was proposed to show that with undetectable and random-position triggers, the adversary can still successfully launch backdoor attacks. After that, more works focus on how to design better trigger patterns for backdoor attacks [83, 103, 75]. Zheng et al. [137] propose backdoor attacks that are undetectable from the frequency perspective. Nguyen et al. [83] argue that different images should have different triggers. Therefore, they propose the generator to produce triggers for different images. Besides better trigger patterns, there are also many works focused on designing better injection processes [73, 131, 141, 75, 7]. In these works, they find that the adversary can even inject a backdoor into the model without changing the label of the poison samples. Other studies on backdoor attacks in various scenarios include [51, 106].

## 6.2 Backdoor Defenses

Following the increasing popularity of backdoor attacks, various defense methods have been proposed. Current defense methods can be divided into three categories. The most popular defense methods are based on reverse engineering, where the defender aims to reverse the possible backdoor triggers to judge whether the given model is the backdoored model or not [121, 16, 72, 48, 41]. Besides reverse engineering, another line of works [118, 14, 33, 119] focuses on mitigating the backdoor by detecting the poison samples. Due to the fact that backdoor attacks may involve more carefully designed triggers to bypass potential defenses, such detection methods look like the arms race with evolving attacks. The last type of backdoor defense method is based on meta-learning to learn the difference between the backdoored models and the clean models by training with a large number of backdoored shadow models [130]. Besides different detection methods, fine-tuning has also been proposed to mitigate backdoor attacks. Previous fine-tuning-based methods either adapt pruning [71] or distillation [65]. Liu et al. [71] argue that fine-tuning itself cannot effectively mitigate backdoor attacks. However, pruning, distillation, or other methods based on fine-tuning will cost much more computational resources and sacrifice the models' utility. In this work, we show for the first time that carefully designed fine-tuning is sufficient to remove the backdoor and maintain the model's utility with limited cost (e.g., with limited epochs).

## 6.3 Contrastive Learning

Contrastive learning is one of the most popular methods to train encoders. Oord et al. [87] proposed Contrastive Predictive Coding, where probabilistic contrastive loss is used to capture information. Wu et al. [127] used a memory bank to store the instance class representation vector and then conducted prediction based on these representations. He et al. [42] proposed a new framework, MoCo, which adds momentum technique to prevent contrastive collapse while maintaining the diversity of negative samples.

Chen et al. [18] proposed SimCLR, which reveals the importance of data augmentation and projector behind encoder. Grill et al. [38] proposed BYOL where negative pair is unnecessary. BYOL also adopts the momentum technique. Chen et al. [20] proposed SimSiam. SimSiam removed the momentum encoder on the basis of BYOL. To avoid contrastive collapse, SimSiam uses the stop gradient technique. SimCLR, MoCo, BYOL, SimSiam are currently the mainstream frameworks of contrastive learning, thus we concentrate on them in this paper.

Previous work also evaluate the security and privacy risks stemming from contrastive learning. He et al. [46] conducted membership inference and attribute inference attacks against contrastive models and showed that contrastive models are less vulnerable to membership inference attacks but more prone to attribute inference attacks. Jia et al. [51] proposed backdoor attacks against contrastive models and showed that the backdoor can be effectively injected into the encoder to perform abnormally in specific downstream tasks. Liu et al. [70] proposed a membership inference attack against encoders trained by contrastive learning. Concretely, they leveraged different augmentations of a sample to query the target encoder and calculated the similarity scores among those embeddings. Intuitively, if the sample is a member, different embeddings should be closer, which leads to higher similarity scores.

## 6.4 Model Stealing Attack

In model stealing attacks, the adversary’s goal is to steal part of the target model. Tramèr et al. [117] proposed the first model stealing attack against black-box machine learning API to steal its parameters. Wang et al. [120] proposed the first hyperparameter stealing attacks against ML models. Oh et al. [86] also tried to steal machine learning model’s architectures and hyperparameters. Orekondy et al. [88] proposed knockoff nets, which aim at stealing the functionality of black-box models. Krishna et al. [57] formulated the model stealing attack against BERT-based API. Besides, Wu et al. [124] and Shen et al. [107] perform model stealing attacks against Graph Neural Networks where Wu et al. [124] focus on the transductive setting and Shen et al. [107] concentrate on the inductive setting. These works often have relative strong assumptions such as model family is known and victim’s data is partly available while we conduct model stealing attacks against encoders and relax the above assumption as well.

## 6.5 Knowledge Distillation

Knowledge distillation aims to transfer the knowledge from the larger “teacher” model to the smaller “student” model. Hinton et al. [47] introduced the idea of knowledge distillation. The basic motivation of knowledge distillation is to achieve fast and lightweight learning. Yuan et al. [136] extended the knowledge distillation to the self-learning field by proposing a Teacher-free Knowledge Distillation(Tf-KD) framework. Tian et al. [114] proposed contrastive representation distillation where contrastive loss is used to do knowledge distillation. Knowledge distillation is similar to model stealing attacks, but it always assumes the adversary knows everything about the target model, whereas model stealing attacks only have limited assumptions.

## 6.6 Text-to-Image Generation

Typically, text-to-image generation takes a text description (i.e., a prompt) as input and outputs an image that matches the text description. Some pioneer works of text-to-image generation [97, 139] are based on GANs [36]. By combining a prompt embedding and a latent vector, the authors expect the GANs to generate an image depicting the prompt. These works have stimulated more researchers [10, 59, 123, 138, 113] to study text-to-image generation models based on GANs, but using GANs does not always achieve good generation performance [96, 99].

Recently, text-to-image generation has made great progress with the emergence of diffusion models [6, 99, 85, 101]. Models in this domain normally take random noise and prompts as input and reduce noisy images to clear ones based on the guidance of prompts. Currently, text-to-image generation based on diffusion models, such as DALL·E [96], Stable Diffusion [99], Imagen [101], GLIDE [85] and DALL·E 2 [95], has achieved state-of-the-art performance compared to previous works. This is also the reason why we focus on such models in this work.

## 6.7 Fake Image Detection and Attribution

Wang et al. [122] find that a simple CNN model can easily detect fake images generated by various types of traditional generation models (e.g., GANs [36] and low-level vision models [15, 26]) from real images. The authors argue that these fake images have some common defects that allow us to distinguish them from real images. Yu et al. [135] demonstrate that fake images generated by various traditional generation models can be attributed to their sources and reveal the fact that these traditional generation models leave fingerprints in the generated images. Girish et al. [35] further propose a new attribution method to deal with the open-world scenario where the detector has no knowledge of the generation model.

We emphasize here that almost all existing works focus only on traditional generation models, such as GANs [36], low-level vision models [15, 26], and perceptual loss generation models [17, 63]. Detecting and attributing fake images generated by text-to-image generation models are largely unexplored. In this thesis, we take the first step to systematically study the problem.



# 7

## Summary and Conclusion





## 7.1 Summary

This dissertation investigates the security issues that happened in the whole pipeline of machine learning models. Specifically, the pipeline of the machine learning models can be divided into the data for training, the model parameter, and the output of the trained model. In this dissertation, we evaluate the possible risks of the machine learning models at each of the above stages. Specifically, we proposed novel defense methods against possible data poisoning attacks to reduce the risks in data security. We proposed the novel model stealing attacks and the adaptive defenses to provide a deeper understanding of the model security. Additionally, we proposed deepfake detection methods to address one of the most pressing issues in digital security.

**Data Security:** The study explores the efficacy of fine-tuning methods to defend against backdoor attacks, particularly focusing on scenarios such as encoder-based, transfer-based, and standalone models. Empirical results demonstrate that fine-tuning and a novel super-fine-tuning approach can effectively remove backdoors. The effectiveness is further validated through ablation studies on dataset size and learning rate.

**Model Security:** The dissertation introduces Cont-Steal, a contrastive learning-based model stealing attack that significantly outperforms conventional attacks. It shows that encoders are more vulnerable than classifiers due to the rich information in embeddings. This dissertation proposed the Cont-Steal attacks against embeddings to leverage the rich information. Moreover, this dissertation also discussed the possible defenses against the proposed attacks.

**Output Security:** The research delves into detecting and attributing fake images generated by text-to-image models. It proposes a hybrid detection framework combining image and textual prompt analysis, which improves the robustness of detection methods. The dissertation also discusses techniques for embedding unique identifiers within generated images to trace their origins, thereby enhancing accountability for the misuse of generative models.

## 7.2 Conclusion

This dissertation provides a comprehensive examination of the vulnerabilities in machine learning models, particularly focusing on the pipeline of machine learning models. It offers significant contributions to defending against these threats through novel fine-tuning and super-fine-tuning methods. The findings indicate that fine-tuning can effectively eliminate backdoors, but defenses must also account for potential vulnerabilities like backdoor sequela.

In the context of model security, the study highlights the severe risks posed by model stealing attacks, especially against contrastive learning models. Cont-Steal, the proposed attack method, showcases the need for robust defenses to protect the intellectual property and data privacy associated with these models.

The output security section underscores the importance of detecting and attributing fake images generated by advanced generative models. The hybrid detection framework

proposed in this work significantly enhances the ability to differentiate between real and fake images, ensuring the integrity and trustworthiness of ML model outputs.

Overall, this research underscores the critical need for robust security measures across various aspects of machine learning to safeguard against evolving threats. The proposed methods and findings provide a foundation for future research and development in securing machine learning systems against sophisticated attacks.

### 7.3 Future Research

In this dissertation, we conducted a comprehensive security assessment during the whole pipeline of machine learning models. We also would like to discuss some future research directions here.

Firstly, in this dissertation, we mainly focus on the pipeline of computer vision models. However, with the widespread application of large generative models in natural language processing, computer vision, and other fields, risk assessment and intelligent defense are particularly important. Compared with previous models, large generative models usually have a huge parameter scale, which may reach hundreds of millions or even billions of parameters, and usually rely on a large amount of training data for training, which may contain diverse information and a wide range of contexts. This makes these models extremely complex, making it difficult to deeply understand their internal mechanisms and decision-making processes, and also increases the hidden risks. Therefore, conducting the overall assessment of security issues of these newly emerged large models can bring us some different discoveries and revelations. Moreover, advanced attacks and defenses based on the pipeline of large models should also be studied to help the community achieve a more fair and efficient AGI.

Secondly, although this dissertation has conducted a comprehensive assessment of the machine learning models pipeline, one universal platform is still needed to better demonstrate the robustness of different models. The platform will provide comprehensive datasets, benchmarks, and tools to provide resources and support for researchers and developers. This will help to better evaluate the security of models and promote continued innovation in the field of trusted machine learning security. Through this large platform, I look forward to promoting progress in the field of machine learning security and building a safer and more trustworthy artificial intelligence future.

Thirdly, it is crucial to make the attacks on deep learning models more practical and grounded. This involves moving from theoretical research to real-world applications, where the impacts and defenses can be more accurately assessed. Developing realistic attack scenarios and conducting experiments in practical environments will help in understanding the true effectiveness and limitations of both attacks and defenses. This practical approach will provide valuable insights and drive the development of more robust and reliable security measures for deep learning models.

Overall, these future directions aim to enhance the security and trustworthiness of machine learning systems, ensuring they can be safely and effectively deployed in real-world applications.

# Bibliography

## Author's Papers for this Thesis

- [P1] Sha, Z., He, X., Berrang, P., Humbert, M., and Zhang, Y. Fine-Tuning Is All You Need to Mitigate Backdoor Attacks. *CoRR abs/2212.09067* (2022).
- [P2] Sha, Z., Li, Z., Yu, N., and Zhang, Y. DE-FAKE: detection and attribution of fake images generated by text-to-image generation models. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021, 845–863.
- [P3] Sha, Z., He, X., Yu, N., Backes, M., and Zhang, Y. Can't Steal? Cont-Steal! Contrastive Stealing Attacks Against Image Encoders. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023.

## Other Papers of the Author

- [S1] Chu, J., Sha, Z., Backes, M., and Zhang, Y. Conversation reconstruction attack against GPT models. *CoRR abs/2402.02987* (2024).
- [S2] Jiang, Y., Shen, X., Wen, R., Sha, Z., Chu, J., Liu, Y., Backes, M., and Zhang, Y. Games and beyond: analyzing the bullet chats of esports livestreaming. In: *International AAAI Conference on Web and Social Media (ICWSM)*. AAAI Press, 2024.
- [S3] Sha, Z. and Zhang, Y. Prompt Stealing Attacks Against Large Language Models. *CoRR abs/2402.12959* (2024).
- [S4] Yang, Z., Sha, Z., Backes, M., and Zhang, Y. From Visual Prompt Learning to Zero-Shot Transfer: Mapping Is All You Need. *CoRR abs/2303.05266* (2023).
- [S5] Zhang, B., Shen, X., Si, W. M., Sha, Z., Chen, Z., Salem, A., Shen, Y., Backes, M., and Zhang, Y. Comprehensive Assessment of Toxicity in ChatGPT. *CoRR abs/2402.12959* (2024).

## Other references

- [1] <https://www.cs.toronto.edu/~kriz/cifar.html>.
- [2] <https://cs.stanford.edu/%7Eacoates/stl10/>.
- [3] <http://benchmark.ini.rub.de/?section=gtsrb>.

## BIBLIOGRAPHY

---

- [4] <http://ufldl.stanford.edu/housenumbers/>.
- [5] Adi, Y., Baum, C., Cisse, M., Pinkas, B., and Keshet, J. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2018, 1615–1631.
- [6] Atwood, J. and Towsley, D. Diffusion-Convolutional Neural Networks. In: *Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 2016, 1993–2001.
- [7] Barni, M., Kallas, K., and Tondi, B. A New Backdoor Attack in CNNs by Training Set Corruption Without Label Poisoning. In: *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, 101–105.
- [8] Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., and Roli, F. Evasion Attacks against Machine Learning at Test Time. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*. Springer, 2013, 387–402.
- [9] Bird, S. and Loper, E. NLTK: The Natural Language Toolkit. In: *Annual Meeting of the Association for Computational Linguistics (ACL)*. ACL, 2004.
- [10] Bodla, N., Hua, G., and Chellappa, R. Semi-supervised FusedGAN for Conditional Image Generation. In: *European Conference on Computer Vision (ECCV)*. Springer, 2018, 689–704.
- [11] Carlini, N. and Terzis, A. Poisoning and Backdooring Contrastive Learning. In: *International Conference on Learning Representations (ICLR)*. 2022.
- [12] Carlini, N. and Wagner, D. Towards Evaluating the Robustness of Neural Networks. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, 39–57.
- [13] Chandrasekaran, V., Chaudhuri, K., Giacomelli, I., Jha, S., and Yan, S. Model Extraction and Active Learning. *CoRR abs/1811.02054* (2018).
- [14] Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I. M., and Srivastava, B. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. *CoRR abs/1811.03728* (2018).
- [15] Chen, C., Chen, Q., Xu, J., and Koltun, V. Learning to See in the Dark. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, 3291–3330.
- [16] Chen, H., Fu, C., Zhao, J., and Koushanfar, F. DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks. In: *International Joint Conferences on Artificial Intelligence (IJCAI)*. IJCAI, 2019, 4658–4664.
- [17] Chen, Q. and Koltun, V. Photographic Image Synthesis with Cascaded Refinement Networks. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, 1520–1529.
- [18] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A Simple Framework for Contrastive Learning of Visual Representations. In: *International Conference on Machine Learning (ICML)*. PMLR, 2020, 1597–1607.

- 
- [19] Chen, X., Salem, A., Backes, M., Ma, S., Shen, Q., Wu, Z., and Zhang, Y. BadNL: Backdoor Attacks Against NLP Models with Semantic-preserving Improvements. In: *Annual Computer Security Applications Conference (ACSAC)*. ACSAC, 2021, 554–569.
- [20] Chen, X. and He, K. Exploring Simple Siamese Representation Learning. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2021, 15750–15758.
- [21] Chen, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *CoRR abs/1712.05526* (2017).
- [22] Choi, Y., Choi, M., Kim, M., Ha, J., Kim, S., and Choo, J. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, 8789–8797.
- [23] Choo, C. A. C., Tramèr, F., Carlini, N., and Papernot, N. Label-Only Membership Inference Attacks. In: *International Conference on Machine Learning (ICML)*. PMLR, 2021, 1964–1974.
- [24] Coates, A., Ng, A. Y., and Lee, H. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. JMLR, 2011, 215–223.
- [25] Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020, 18613–18624.
- [26] Dai, T., Cai, J., Zhang, Y., Xia, S., and Zhang, L. Second-Order Attention Network for Single Image Super-Resolution. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, 11065–11074.
- [27] Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, 248–255.
- [28] Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. ACL, 2019, 4171–4186.
- [29] Doersch, C. and Zisserman, A. Multi-task Self-Supervised Visual Learning. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, 2070–2079.
- [30] Dognin, P. L., Melnyk, I., Mroueh, Y., Padhi, I., Rigotti, M., Ross, J., Schiff, Y., Young, R. A., and Belgodere, B. Image Captioning as an Assistive Technology: Lessons Learned from VizWiz 2020 Challenge. *Journal of Artificial Intelligence Research* (2022).
- [31] Elisa, K., D, G. N., and Christopher, P. Concadia: Tackling Image Accessibility with Descriptive Texts and Context. *CoRR abs/2104.08376* (2021).

## BIBLIOGRAPHY

---

- [32] Ester, M., Kriegel, H., Sander, J., and Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *International Conference on Knowledge Discovery and Data Mining (KDD)*. AAAI, 1996, 226–231.
- [33] Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D. C., and Nepal, S. STRIP: A Defence Against Trojan Attacks on Deep Neural Networks. In: *Annual Computer Security Applications Conference (ACSAC)*. ACM, 2019, 113–125.
- [34] Gidaris, S., Singh, P., and Komodakis, N. Unsupervised Representation Learning by Predicting Image Rotations. In: *International Conference on Learning Representations (ICLR)*. 2018.
- [35] Girish, S., Suri, S., Rambhatla, S. S., and Shrivastava, A. Towards Discovery and Attribution of Open-World GAN Generated Images. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021, 14094–14103.
- [36] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Nets. In: *Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 2014, 2672–2680.
- [37] Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and Harnessing Adversarial Examples. In: *International Conference on Learning Representations (ICLR)*. 2015.
- [38] Grill, J., Strub, F., Alché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. Á., Guo, Z., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.
- [39] Gu, T., Dolan-Gavitt, B., and Grag, S. Badnets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *CoRR abs/1708.06733* (2017).
- [40] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved Training of Wasserstein GANs. In: *Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 2017, 5767–5777.
- [41] Guo, W., Wang, L., Xing, X., Du, M., and Song, D. TABOR: A Highly Accurate Approach to Inspecting and Restoring Trojan Backdoors in AI Systems. *CoRR abs/1908.01763* (2019).
- [42] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. B. Momentum Contrast for Unsupervised Visual Representation Learning. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, 9726–9735.
- [43] He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, 770–778.
- [44] He, X., Li, Z., Xu, W., Cornelius, C., and Zhang, Y. Membership-Doctor: Comprehensive Assessment of Membership Inference Against Machine Learning Models. *CoRR abs/2208.10445* (2022).

- 
- [45] He, X., Wen, R., Wu, Y., Backes, M., Shen, Y., and Zhang, Y. Node-Level Membership Inference Attacks Against Graph Neural Networks. *CoRR abs/2102.05429* (2021).
- [46] He, X. and Zhang, Y. Quantifying and Mitigating Privacy Risks of Contrastive Learning. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021, 845–863.
- [47] Hinton, G. E., Vinyals, O., and Dean, J. Distilling the Knowledge in a Neural Network. *CoRR abs/1503.02531* (2015).
- [48] Huang, X., Alzantot, M., and Srivastava, M. B. NeuronInspect: Detecting Backdoors in Neural Networks via Output Explanations. *CoRR abs/1911.07399* (2019).
- [49] Hui, B., Yang, Y., Yuan, H., Burlina, P., Gong, N. Z., and Cao, Y. Practical Blind Membership Inference Attack via Differential Comparisons. In: *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2021.
- [50] Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A., and Papernot, N. High Accuracy and High Fidelity Extraction of Neural Networks. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2020, 1345–1362.
- [51] Jia, J., Liu, Y., and Gong, N. Z. BadEncoder: Backdoor Attacks to Pre-trained Encoders in Self-Supervised Learning. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022.
- [52] Kariyappa, S. and Qureshi, M. K. Defending Against Model Stealing Attacks With Adaptive Misinformation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, 767–775.
- [53] Karras, T., Laine, S., and Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, 4401–4410.
- [54] Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. Supervised Contrastive Learning. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.
- [55] Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In: *International Conference on Learning Representations (ICLR)*. 2015.
- [56] Kornblith, S., Shlens, J., and Le, Q. V. Do Better ImageNet Models Transfer Better? In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, 2661–2671.
- [57] Krishna, K., Tomar, G. S., Parikh, A. P., Papernot, N., and Iyyer, M. Thieves on Sesame Street! Model Extraction of BERT-based APIs. In: *International Conference on Learning Representations (ICLR)*. 2020.
- [58] Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial Examples in the Physical World. *CoRR abs/1607.02533* (2016).

## BIBLIOGRAPHY

---

- [59] Lao, Q., Havaei, M., Pesaranghader, A., Dutil, F., Di-Jorio, L., and Fevens, T. Dual Adversarial Inference for Text-to-Image Synthesis. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2019, 7566–7575.
- [60] Leino, K. and Fredrikson, M. Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2020, 1605–1622.
- [61] Li, H., Chaudhari, P., Yang, H., Lam, M., Ravichandran, A., Bhotika, R., and Soatto, S. Rethinking the Hyperparameters for Fine-tuning. In: *International Conference on Learning Representations (ICLR)*. 2020.
- [62] Li, J., Li, D., Xiong, C., and Hoi, S. C. H. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. *CoRR abs/2201.12086* (2022).
- [63] Li, K., Zhang, T., and Malik, J. Diverse Image Synthesis From Semantic Layouts via Conditional IMLE. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2019, 4219–4228.
- [64] Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., and Ma, X. Anti-Backdoor Learning: Training Clean Models on Poisoned Data. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2021, 14900–14912.
- [65] Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., and Ma, X. Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks. In: *International Conference on Learning Representations (ICLR)*. 2021.
- [66] Li, Y., Li, Y., Wu, B., Li, L., He, R., and Lyu, S. Invisible Backdoor Attack with Sample-Specific Triggers. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021, 16443–16452.
- [67] Li, Z. and Zhang, Y. Membership Leakage in Label-Only Exposures. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021, 880–895.
- [68] Lin, T., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: Common Objects in Context. In: *European Conference on Computer Vision (ECCV)*. Springer, 2014, 740–755.
- [69] Liu, H., Jia, J., and Gong, N. Z. PoisonedEncoder: Poisoning the Unlabeled Pre-training Data in Contrastive Learning. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2022, 3629–3645.
- [70] Liu, H., Jia, J., Qu, W., and Gong, N. Z. EncoderMI: Membership Inference against Pre-trained Encoders in Contrastive Learning. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021.
- [71] Liu, K., Dolan-Gavitt, B., and Garg, S. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. In: *Research in Attacks, Intrusions, and Defenses (RAID)*. Springer, 2018, 273–294.



- 
- [72] Liu, Y., Lee, W.-C., Tao, G., Ma, S., Aafer, Y., and Zhang, X. ABS: Scanning Neural Networks for Back-Doors by Artificial Brain Stimulation. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, 1265–1282.
- [73] Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. Trojaning Attack on Neural Networks. In: *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2018.
- [74] Liu, Y., Wen, R., He, X., Salem, A., Zhang, Z., Backes, M., Cristofaro, E. D., Fritz, M., and Zhang, Y. ML-Doctor: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2022, 4525–4542.
- [75] Liu, Y., Ma, X., Bailey, J., and Lu, F. Reflection Backdoor: A Natural Backdoor Attack on Deep Neural Networks. In: *European Conference on Computer Vision (ECCV)*. Springer, 2020, 182–199.
- [76] Maaten, L. van der and Hinton, G. Visualizing Data using t-SNE. *Journal of Machine Learning Research* (2008).
- [77] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. In: *International Conference on Learning Representations (ICLR)*. 2018.
- [78] Micah, H., Peter, Y., and Julia, H. Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. *Journal of Artificial Intelligence Research* (2013).
- [79] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral Normalization for Generative Adversarial Networks. In: *International Conference on Learning Representations (ICLR)*. 2018.
- [80] Nasr, M., Shokri, R., and Houmansadr, A. Machine Learning with Membership Privacy using Adversarial Regularization. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2018, 634–646.
- [81] Nasr, M., Shokri, R., and Houmansadr, A. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, 1021–1035.
- [82] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading Digits in Natural Images with Unsupervised Feature Learning. In: *Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 2011.
- [83] Nguyen, T. A. and Tran, A. Input-Aware Dynamic Backdoor Attack. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.
- [84] Nguyen, T. A. and Tran, A. T. WaNet - Imperceptible Warping-based Backdoor Attack. In: *International Conference on Learning Representations (ICLR)*. 2021.
- [85] Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. *CoRR abs/2112.10741* (2021).

## BIBLIOGRAPHY

---

- [86] Oh, S. J., Augustin, M., Schiele, B., and Fritz, M. Towards Reverse-Engineering Black-Box Neural Networks. In: *International Conference on Learning Representations (ICLR)*. 2018.
- [87] Oord, A. van den, Li, Y., and Vinyals, O. Representation Learning with Contrastive Predictive Coding. *CoRR abs/1807.03748* (2018).
- [88] Orekondy, T., Schiele, B., and Fritz, M. Knockoff Nets: Stealing Functionality of Black-Box Models. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, 4954–4963.
- [89] Pang, R., Shen, H., Zhang, X., Ji, S., Vorobeychik, Y., Luo, X., Liu, A. X., and Wang, T. A Tale of Evil Twins: Adversarial Inputs versus Poisoned Models. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2020, 85–99.
- [90] Pang, R., Zhang, Z., Gao, X., Xi, Z., Ji, S., Cheng, P., and Wang, T. TROJAN-ZOO: Everything You Ever Wanted to Know about Neural Backdoors (But Were Afraid to Ask). *CoRR abs/2012.09302* (2020).
- [91] Papernot, N., McDaniel, P. D., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical Black-Box Attacks Against Machine Learning. In: *ACM Asia Conference on Computer and Communications Security (ASIACCS)*. ACM, 2017, 506–519.
- [92] Papernot, N., McDaniel, P. D., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The Limitations of Deep Learning in Adversarial Settings. In: *IEEE European Symposium on Security and Privacy (Euro S&P)*. IEEE, 2016, 372–387.
- [93] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning Transferable Visual Models From Natural Language Supervision. In: *International Conference on Machine Learning (ICML)*. PMLR, 2021, 8748–8763.
- [94] Radford, A., Metz, L., and Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In: *International Conference on Learning Representations (ICLR)*. 2016.
- [95] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical Text-Conditional Image Generation with CLIP Latents. *CoRR abs/2204.06125* (2022).
- [96] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-Shot Text-to-Image Generation. In: *International Conference on Machine Learning (ICML)*. JMLR, 2021, 8821–8831.
- [97] Reed, S. E., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. Generative Adversarial Text to Image Synthesis. In: *International Conference on Machine Learning (ICML)*. JMLR, 2016, 1060–1069.
- [98] Reimers, N. and Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. ACL, 2019, 3980–3990.

- 
- [99] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-Resolution Image Synthesis with Latent Diffusion Models. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022, 10684–10695.
- [100] Sablayrolles, A., Douze, M., Schmid, C., Ollivier, Y., and Jégou, H. White-box vs Black-box: Bayes Optimal Strategies for Membership Inference. In: *International Conference on Machine Learning (ICML)*. PMLR, 2019, 5558–5567.
- [101] Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. *CoRR abs/2205.11487* (2022).
- [102] Salem, A., Backes, M., and Zhang, Y. Don’t Trigger Me! A Triggerless Backdoor Attack Against Deep Neural Networks. *CoRR abs/2010.03282* (2020).
- [103] Salem, A., Wen, R., Backes, M., Ma, S., and Zhang, Y. Dynamic Backdoor Attacks Against Machine Learning Models. In: *IEEE European Symposium on Security and Privacy (Euro S&P)*. IEEE, 2022, 703–718.
- [104] Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., and Backes, M. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In: *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.
- [105] Santurkar, S., Dubois, Y., Taori, R., Liang, P., and Hashimoto, T. Is a Caption Worth a Thousand Images? A Controlled Study for Representation Learning. *CoRR abs/2207.07635* (2022).
- [106] Shen, X., He, X., Li, Z., Shen, Y., Backes, M., and Zhang, Y. Backdoor Attacks in the Supply Chain of Masked Image Modeling. *CoRR abs/2210.01632* (2022).
- [107] Shen, Y., He, X., Han, Y., and Zhang, Y. Model Stealing Attacks Against Inductive Graph Neural Networks. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022, 1175–1192.
- [108] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership Inference Attacks Against Machine Learning Models. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, 3–18.
- [109] Smith, L. N. and Topin, N. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. *CoRR abs/1708.07120* (2018).
- [110] Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- [111] Song, L. and Mittal, P. Systematic Evaluation of Privacy Risks of Machine Learning Models. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2021.
- [112] Song, L., Shokri, R., and Mittal, P. Privacy Risks of Securing Machine Learning Models against Adversarial Examples. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, 241–257.

## BIBLIOGRAPHY

---

- [113] Souza, D. M., Wehrmann, J., and Ruiz, D. D. Efficient Neural Architecture for Text-to-Image Synthesis. In: *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, 1–8.
- [114] Tian, Y., Krishnan, D., and Isola, P. Contrastive Representation Distillation. In: *International Conference on Learning Representations (ICLR)*. 2020.
- [115] Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What Makes for Good Views for Contrastive Learning? In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.
- [116] Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble Adversarial Training: Attacks and Defenses. In: *International Conference on Learning Representations (ICLR)*. 2017.
- [117] Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. Stealing Machine Learning Models via Prediction APIs. In: *USENIX Security Symposium (USENIX Security)*. USENIX, 2016, 601–618.
- [118] Tran, B., Li, J., and Madry, A. Spectral Signatures in Backdoor Attacks. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2018, 8011–8021.
- [119] Udeshi, S., Peng, S., Woo, G., Loh, L., Rawshan, L., and Chattopadhyay, S. Model Agnostic Defence Against Backdoor Attacks in Machine Learning. *IEEE Transactions on Reliability* (2022).
- [120] Wang, B. and Gong, N. Z. Stealing Hyperparameters in Machine Learning. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2018, 36–52.
- [121] Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, 707–723.
- [122] Wang, S., Wang, O., Zhang, R., Owens, A., and Efros, A. A. CNN-Generated Images Are Surprisingly Easy to Spot... for Now. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, 8692–8701.
- [123] Wang, Z., Quan, Z., Wang, Z., Hu, X., and Chen, Y. Text to Image Synthesis With Bidirectional Generative Adversarial Network. In: *International Conference on Multimedia and Expo (ICME)*. IEEE, 2020, 1–6.
- [124] Wu, B., Yang, X., Pan, S., and Yuan, X. Model Extraction Attacks on Graph Neural Networks: Taxonomy and Realization. *CoRR abs/2010.12751* (2020).
- [125] Wu, B., Chen, H., Zhang, M., Zhu, Z., Wei, S., Yuan, D., Shen, C., and Zha, H. BackdoorBench: A Comprehensive Benchmark of Backdoor Learning. *CoRR abs/2206.12654* (2022).
- [126] Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., and Zhu, L. Adversarial Examples for Graph Data: Deep Insights into Attack and Defense. In: *International Joint Conferences on Artificial Intelligence (IJCAI)*. IJCAI, 2019, 4816–4823.

- 
- [127] Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised Feature Learning via Non-Parametric Instance Discrimination. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, 3733–3742.
- [128] Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR abs/1708.07747* (2017).
- [129] Xie, C., Zhang, Z., Zhou, Y., Bai, S., Wang, J., Ren, Z., and Yuille, A. L. Improving Transferability of Adversarial Examples With Input Diversity. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, 2730–2739.
- [130] Xu, X., Wang, Q., Li, H., Borisov, N., Gunter, C. A., and Li, B. Detecting AI Trojans Using Meta Neural Analysis. In: *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2021.
- [131] Yao, Y., Li, H., Zheng, H., and Zhao, B. Y. Latent Backdoor Attacks on Deep Neural Networks. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, 2041–2055.
- [132] Yeom, S., Giacomelli, I., Fredrikson, M., and Jha, S. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In: *IEEE Computer Security Foundations Symposium (CSF)*. IEEE, 2018, 268–282.
- [133] Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* (2014).
- [134] Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., Hutchinson, B., Han, W., Parekh, Z., Li, X., Zhang, H., Baldrige, J., and Wu, Y. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation. *CoRR abs/2206.10789* (2022).
- [135] Yu, N., Davis, L., and Fritz, M. Attributing Fake Images to GANs: Learning and Analyzing GAN Fingerprints. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2019, 7555–7565.
- [136] Yuan, L., Tay, F. E. H., Li, G., Wang, T., and Feng, J. Revisiting Knowledge Distillation via Label Smoothing Regularization. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, 3903–3911.
- [137] Zeng, Y., Park, W., Mao, Z. M., and Jia, R. Rethinking the Backdoor Attacks’ Triggers: A Frequency Perspective. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021, 16453–16461.
- [138] Zhang, H., Koh, J. Y., Baldrige, J., Lee, H., and Yang, Y. Cross-Modal Contrastive Learning for Text-to-Image Generation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2021, 833–842.
- [139] Zhang, H., Xu, T., and Li, H. StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, 5908–5916.

## BIBLIOGRAPHY

---

- [140] Zhang, X., Karaman, S., and Chang, S. Detecting and Simulating Artifacts in GAN Fake Images. In: *IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2019, 1–6.
- [141] Zhao, S., Ma, X., Zheng, X., Bailey, J., Chen, J., and Jiang, Y.-G. Clean-Label Backdoor Attacks on Video Recognition Models. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, 14443–144528.
- [142] Zhuang, C., Zhai, A. L., and Yamins, D. Local Aggregation for Unsupervised Learning of Visual Embeddings. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2019, 6001–6011.
- [143] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. A Comprehensive Survey on Transfer Learning. *CoRR abs/1911.02685* (2019).