SENTENCE EMBEDDING ENCODERS ARE EASY TO STEAL BUT HARD TO DEFEND

Adam Dziedzic*
University of Toronto and Vector Institute
adam.dziedzic@utoronto.ca

Franziska Boenisch*
University of Toronto and Vector Institute
franziska.boenisch@vectorinstitute.ai

Mingjian Jiang
University of Toronto and Vector Institute
mingjian.jiang@mail.utoronto.ca

Haonan Duan
University of Toronto and Vector Institute
haonand@cs.toronto.edu

Nicolas Papernot
University of Toronto and Vector Institute
nicolas.papernot@utoronto.ca

ABSTRACT

Self-supervised learning (SSL) has become the predominant approach to training on large amounts of data when no labels are available. Since the corresponding model architectures are usually large, the training process is, in itself, costly, and training relies on dedicated expensive hardware. As a consequence, not every party can train such models from scratch. Instead, new APIs offer paid access to pre-trained SSL models. We consider transformer-based SSL sentence encoders and show that they can be efficiently extracted (stolen) from behind these APIs through black-box query access. Our stealing requires down to 40x fewer queries than the number of the victim's training data points and much less computation. This large gap between low attack costs and high victim training costs strongly incentivizes attackers to steal encoders. To protect the transformer-based sentence encoders against stealing, we propose to embed secret downstream tasks to their training which serve as watermarks. In general, our work highlights that sentence embedding encoders are easily stolen but hard to defend.

1 INTRODUCTION

The success of self-supervised learning (SSL) motivates the emergence of large-scale services offering API access to encoders which return high-dimensional representations for given input data. These representations serve to train a diverse set of downstream tasks with a small amount of labeled data. Latest APIs (Clarifai, 2023; Cohere, 2023; OpenAI, 2023) use transformer-based encoders (Devlin et al., 2018; Dosovitskiy et al., 2020) to generate representations. Such encoders have a high number of parameters (*e.g.,* the state-of-the-art RoBERTa-Large language encoder (Liu et al., 2019) has roughly 355M parameters) and are trained on datasets consisting of millions of unlabeled data points—yielding a highly expensive training procedure (Sharir et al., 2020). Therefore, these encoders are lucrative targets for stealing attacks (Tramèr et al., 2016). In such attacks, an adversary extracts a victim encoder by submitting queries and using the outputs to train a local stolen copy, often at a fraction of the victim's training cost (Sha et al., 2022; Dziedzic et al., 2022a). The stolen encoder can then be used for inferences without the owner's permission, violating their intellectual property right and causing financial loss.

While stealing and defending supervised models has been heavily studied (Tramèr et al., 2016; Juuti et al., 2019; Orekondy et al., 2020), research on the topic of stealing and defending transformer-based encoders in the NLP domain is limited. Despite the immediate practical importance of this problem, to the best of our knowledge, all previous works on model stealing attacks and defenses against SSL

*Equal contribution.

encoders are conducted offline in contrived experimental settings (Cong et al., 2022; Sha et al., 2022; Dziedzic et al., 2022a;b), focusing on the vision domain with convolutional neural network (CNN)-based architectures, and do not attack the popular transformer (Vaswani et al., 2017) architecture. In our work, we focus on stealing state-of-the-art sentence embedding encoders (Gao et al., 2021) that are built on top of pre-trained transformers and exposed via public APIs (Clarifai, 2023).[1]

We show how stealing attacks (Sha et al., 2022; Dziedzic et al., 2022a) can be successfully applied to extract transformer-based sentence encoders only through their returned representations. This corresponds to a real-world API setup. Our stealing requires much less computing power and much fewer data points than training the encoders from scratch. More specifically, we successfully steal sentence-encoders using only a small number of representations obtained through queries to the victim. We steal with down to 40x fewer queries than the number of original training data points and our stolen encoders achieve similar performances on benchmark tasks as their corresponding victims. We also show that this number can be further reduced by re-using representations obtained from the victim encoder for semantically similar sentences of the stealing queries.

The successful applicability of encoder stealing to transformer-based architectures in public API settings motivates the urgent need for defenses. We propose a new watermarking scheme to protect the encoders from theft. Our watermark relies on alternating between the actual sentence embedding task and a secretly chosen downstream task during the last iterations of training. This transforms the representations so that they preserve their high performance on general sentence embedding tasks while increasing their accuracy on the downstream task. To embed the watermark task, we append a fully-connected layer to the encoder. The additional layer acts as our secret key. We verify whether a given encoder is a stolen copy by attaching that layer to the suspect encoder and checking the agreement to the victim encoder's output for the watermark downstream task. The victim and independent encoders have significantly different outputs whereas victim and stolen copies return similar outputs.

To summarize, we make the following contributions:

- We successfully steal sentence embedding encoders in a real-world API setting. Our stolen encoders achieve comparable performance to their respective victims on standard benchmarks assuming access to representations only. **The stealing process uses less computing power than a victim's training and up to 40x fewer queries than the number of samples in a victim's training dataset.**
- We further reduce the number of stealing queries by using semantically similar sentences while achieving comparable performance to the victims on standard benchmarks.
- To detect stolen sentence embedding encoders, we propose a method to watermark their representations by alternating between the actual embedding and a secretly chosen downstream task during the last iterations of training.

2 Background and Related Work

Model extraction attacks. The goal of the model extraction attacks is to replicate a functionality of a victim model f_v trained on a dataset D_P. An attacker has a black box access to the victim model and uses a stealing dataset $D_s = \{q_i, f_v(q_i)\}_{i=1}^n$, consisting of queries q_i and the corresponding predictions or representations returned by the victim model, to train a stolen model f_s. Model extraction attacks have been shown against various types of models including classification (Tramèr et al., 2016; Jagielski et al., 2020) and representation models (Sha et al., 2022; Dziedzic et al., 2022a).

Sentence Embedding Encoders. We use **SimCSE** (Gao et al., 2021) to learn sentence representations since it outperforms other methods and is exposed via public APIs (Clarifai, 2023). SimCSE proposes unsupervised and supervised approaches to generate sentence embeddings. It starts from a pre-trained checkpoint of a BERT-based encoder, *e.g.,* RoBERTa, and takes the representation for the classification token ([CLS]) as sentence embeddings. In this work, we rely on the *supervised approach* leveraging pairs of sentences from natural language inference (NLI) datasets within a contrastive learning framework. It uses the *entailment* pairs as positives and *contradictions* as hard negatives.

[1]`https://clarifai.com/princeton-nlp/language-modeling/models/sup-simcse-roberta-large`

Stealing Encoders. Thus far, methods for stealing encoders through representations have been shown in the computer vision domain and only for CNNs (Sha et al., 2022; Dziedzic et al., 2022a). Previous work in the NLP domain focuses on classification tasks and performs stealing against fine-tuned models through labels based on a given pre-trained language encoder (Krishna et al., 2020; Zanella-Beguelin et al., 2021; He et al., 2021). Model extraction against NLP APIs is shown by (Xu et al., 2021), specifically for sentiment classification and machine translation tasks. The setup of previous work differs from ours which is concerned with stealing through representations instead of low-dimensional outputs, such as labels. This is motivated by the fact that these representations are exactly what new public APIs expose (Cohere, 2023; Clarifai, 2023). Distillation methods used in the NLP domain (Jiao et al., 2019) which could, in principle, be applied to stealing encoders, usually require white box access to the original model, for example, to the attention layers (Jiao et al., 2019). Therefore, distillation cannot be applied to stealing in public API-access scenarios.

Defending Encoders. Dataset inference (DI) (Maini et al., 2021; Dziedzic et al., 2022b) is a defense against model stealing attacks which relies on identifying stolen copies for ownership attribution. It enables a model owner or a third-party arbitrator to attribute the ownership of a potentially stolen model. Therefore, it uses the victim's training dataset as a unique signature. Recently, watermarking (Uchida et al., 2017; Jia et al., 2021; Adi et al., 2018) methods have been proposed for encoders (Dziedzic et al., 2022a; Cong et al., 2022; Wu et al., 2022). The watermarking techniques use downstream tasks to detect a watermark while dataset inference for SSL (Dziedzic et al., 2022b) resolves ownership based on the representations directly. For a more detailed overview of related work, please see Appendix A.

3 Stealing Transformer-Based Encoders

We aim at stealing BERT-based transformers that are fine-tuned to return sentence embeddings. Stealing is performed following previous work (Dziedzic et al., 2022a): (i) The adversary sends N raw or augmented inputs to the victim encoder. These inputs can, in principle, be taken from any data distribution of the target domain, using open-source data. (ii) With the obtained representations, the adversary trains a stolen copy of the victim. The goal of this training is to maximize the similarity of the stolen copy's output and the representations output by the victim. Therefore, the adversary either imitates a self-supervised training using a contrastive loss function, *e.g.,* InfoNCE Chen et al. (2020) or SoftNN Frosst et al. (2019), or directly matches both models' representations via the Mean Squared Error (MSE) loss.

Our stealing method operates in a public API setting where the adversary can query the encoders through a pre-defined interface to obtain high-dimensional representations for their inputs. Public APIs (Cohere, 2023) expose transformers which are first pre-trained on a large corpus of text data to return per-token representations and then fine-tuned to return high-dimensional embeddings for a given full-text input, *e.g.,* a sentence. We find that public APIs (Clarifai, 2023) provide metadata about exposed encoders, which can contain information about datasets used for pre-training as well as the encoder architecture. Thus, we can instantiate our stolen encoders with the victim encoder's architecture. We also follow the API setting and initialize the stolen copies of language encoders with publicly available pre-trained transformers.[2]

Re-using Representations. For stealing sentence encoders more efficiently, we reduce the number of stealing queries by re-using representations over semantically similar sentences.[3] This is possible since sentence encoders are required to return similar representations to such semantically similar sentences. Hence, when a stealing dataset holds such similar sentences (*e.g.,* in the form of positive pairs, which is the case for all the datasets used in this work), we only have to query one of these sentences and assign the same representation to all semantically similar sentences to augment the dataset that the attacker uses for fine-tuning the stolen-copy. Our experimental evaluation in Section 5 shows the effectiveness of this approach.

[2]We use transformers from Hugging Face (`https://huggingface.co/`.

[3]This is relevant in public API settings since costs usually increase linearly with the number of queries and since the number of representations that can be obtained in a given time-unit is often limited. If furthermore reduces the computational costs of the stealing attack.

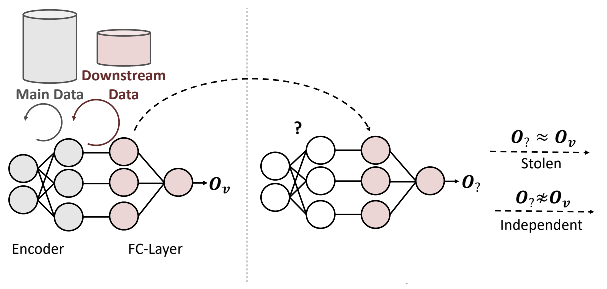

Figure 1: **Our Watermark for Sentence Encoders.** To embed our watermark for sentence embedding encoders, training alternates between the main task and a secretly chosen downstream task during the last iterations. For verification, the fully-connected (FC) layer is attached to a potentially stolen copy and agreement to the victim's output on the downstream task is measured.

4 Watermarking Sentence-Embedding Encoders

We develop a new watermarking-based defense against stealing sentence embedding encoders. We embed the watermark starting from an already fine-tuned encoder. This is a realistic scenario where the model owner would like to add a watermark post hoc at a low cost. To embed the watermark, we perform a few iterations of training, where we alternate between one iteration of the original sentence embedding training (with SimCSE), and then one iteration of training of the whole model for a downstream task. During training of the downstream task, we add an additional fully-connected layer, which serves as our secret key during verification. Our watermarking approach is visualized in Figure 1. In this work, we select SST2 (binary classification for sentiment analysis) as the watermark downstream task. Note, however, that a defender can select from many possible downstream tasks, reshuffle or flip the labels, or use their own private downstream task.

To resolve ownership, a verifier attaches the fully-connected layer (secret key) to the output of an encoder suspected to be a stolen copy. Then, agreement between the outputs of the victim encoder (plus the fully-connected layer), and the outputs of the potentially stolen copy (plus the same fully-connected layer) on the secret downstream task is measured as the percentage of labels where both outputs agree. We resolve that an encoder was stolen if the agreement is above the threshold of 95%, otherwise, the encoder is marked as independent

5 Empirical Evaluation

We evaluate our methods for stealing and defending transformer-based sentence encoders trained on different NLP datasets within a public API setup with black-box query access.

5.1 Experimental Setting.

We steal from BERT-based sentence embedding encoders fine-tuned on nli-for-SimCSE (Gao et al., 2021) ("nli"), QQP (Iyer et al.) ("qqp"), and Flickr30k (Young et al., 2014) ("flickr"). For more insights on the datasets and our pre-processing, see Appendix B.1.

Victim. As victim encoders, we use TinyBERT-based encoders and fine-tune them for the sentence embedding task on nli, qqp, and flickr by using SimSCE (Gao et al., 2021). For more details on data pre-processing and the datasets, see Appendix B.1. We fine-tune our encoders for 10 epochs, with batch size (*bs*)=128, learning rate (*lr*)=5e-5, and temperature=0.05. Additionally, we use transformers fine-tuned with nli from BERT [4], and RoBERTa [5] from Hugging Face as victim encoders. For fine-tuning and stealing over all encoders, we set the maximal input sequence length to 32 and use truncation and padding. The performance of our victim encoders can be found in Table 1.

[4]`https://huggingface.co/princeton-nlp/sup-simcse-bert-base-uncased`
[5]`https://huggingface.co/princeton-nlp/sup-simcse-roberta-large`

Table 1: **Performance of NLP Transformers.** We follow SimCSE (Gao et al., 2021) and use the SentEval benchmark. f_v denotes the victim encoder trained on data D_v. f_s is the stolen encoder extracted using queries from a given stealing dataset D_s. For stealing, we use 60K queries to the victim encoder and fine-tune our stolen copy with the resulting outputs for 20 (TinyBERT (T) and BERT (B)), and 5 (RoBERTa (R)) epochs. Victim encoders with an asterisk (*) are pre-trained encoders from Hugging Face obtained by the SimCSE code-base, while other encoders are trained using the SimCSE code-base. Δ is the average difference in performance between the victim and the stolen encoder over all tasks.

EN		D_v	D_s	CR	MPQA	MR	MRPC	SST2	SUBJ	TREC	Avg.$_{\text{STS}}$	Avg.$_{\text{Tran}}$	SICKR	STSB	AVG↑	Δ ↓
f_v	T	nli	-	**74.69**	**82.17**	**68.51**	**71.93**	**72.02**	**87.99**	**59.74**	**76.64**	**73.86**	**73.67**	**79.61**	**74.62**	-
		qqp	-	72.27	79.1	65.93	70.31	68.92	85.83	57.17	66.19	71.36	56.15	76.22	69.95	-
		flickr	-	71.87	79.39	66.56	71.52	69.61	86.89	55.25	72.83	71.58	66.63	79.03	71.92	-
f_s	T	nli	nli	**73.56**	79.67	67.16	**71.05**	71.33	**87.28**	56.53	**73.48**	72.37	**69.56**	77.40	**72.67**	**1.95**
			qqp	73.48	**79.88**	**67.22**	70.78	**71.9**	87.23	56.51	71.08	**72.42**	65.74	76.41	72.06	2.56
			flickr	71.88	78.12	66.91	70.02	70.64	87.12	**57.45**	71.71	71.73	65.98	**77.44**	71.73	2.89
		qqp	nli	71.42	78.08	66.88	70.14	70.64	86.62	50.24	64.97	70.57	55.18	**74.75**	69.04	0.91
			qqp	**72.2**	**78.23**	67.07	71.05	**71.67**	86.83	**52.97**	64.36	**71.43**	54.66	74.06	**69.50**	**0.45**
			flickr	71.95	77.78	**67.14**	**71.61**	**71.67**	**86.96**	49.34	**65.23**	70.92	**56.37**	74.09	69.37	0.58
		flickr	nli	70.51	**78.53**	66.66	**71.69**	**71.67**	85.7	**54.35**	70.53	**71.30**	63.43	77.63	**71.09**	**0.83**
			qqp	**71.05**	78.41	66.63	71.49	70.87	**86.46**	53.6	68.57	71.22	60.42	76.73	70.50	1.43
			flickr	70.66	77.19	**66.74**	71.12	71.22	86.13	53.76	**70.92**	70.97	**63.82**	**78.03**	70.96	0.96
f_v*	B*	nli	-	89.20	89.67	82.88	73.51	87.31	94.81	88.40	81.57	86.54	80.39	84.26	85.32	-
f_s	B	nli	nli	**89.05**	88.95	**80.49**	74.98	86.35	93.39	66.73	**81.45**	82.85	**79.84**	**83.07**	**82.47**	**2.85**
			qqp	88.25	**89.18**	79.74	**75.20**	**86.58**	**94.09**	**69.08**	79.93	**83.16**	77.24	82.62	82.28	3.04
			flickr	82.01	88.50	74.99	72.77	82.11	91.77	63.41	79.23	79.37	76.96	81.50	79.33	5.99
f_v*	R*	nli	-	92.37	90.52	88.04	76.64	92.31	95.13	91.20	83.76	89.46	81.95	86.70	88.00	-
f_s	R	nli	nli	92.00	**90.72**	86.36	**76.41**	91.76	94.19	86.00	82.15	88.21	**81.03**	85.21	86.73	1.28
			qqp	**92.58**	90.69	**87.02**	75.54	**92.53**	**94.48**	**88.60**	**82.82**	**88.78**	80.79	**85.84**	**87.24**	**0.76**
			flickr	91.74	90.14	85.13	74.72	90.39	93.13	83.20	79.61	86.92	79.06	82.55	85.14	2.86

Stolen. We initialize our stolen encoders with pre-trained transformers from Huggingface (prajjwal1/bert-tiny, bert-base-uncased, and roberta-large), in accordance with the respective victim encoder. During stealing, TinyBERT and BERT use an *lr*=1e-5, *bs*=256, and linear *lr*-scheduling with patience 200 iterations and factor 0.5. For RoBERTa, we use the same setup, however with bs=64, *lr*-patience 600 iterations, and *lr*=5e-6 when stealing with nli or flickr. As in SimCSE, we evaluate our stolen encoders on the SentEval benchmark.

Independent. We fine-tune the independent TinyBERT-based encoders on nli, qqp, and flickr in the same setup as the victim encoders. To obtain independent encoders based on BERT and RoBERTa, we fine-tune the respective base encoders on nli, qqp, and flickr for using SimCSE. We keep *lr*=5e-5, temperature=0.05, *bs*=128 and *bs*=32 for BERT and RoBERTa, respectively, but following (Gao et al., 2021), we fine-tune only for 3 epochs.

5.2 STEALING

We depict the performance of our victim and stolen encoders evaluated on tasks from the SentEval benchmark in Table 1. We observe that across all base encoders (TinyBERT, BERT, and RoBERTa), the performance of the stolen copies is comparable to their respective victim encoders over most benchmark tasks. For RoBERTa, the difference (Δ) in average accuracy across all benchmarks (AVG) between victim f_v and the best stolen encoder f_s is even $< 1\%$. This holds true also for the large qqp dataset (∼2.6M training samples) and stolen copies obtained with only 60K queries, *i.e.,* ∼40x fewer queries than training samples. In general, the performance of encoders stolen with nli and qqp is higher than the one of encoders stolen with flickr. We suspect this is due to the low semantic diversity in flickr which consists only of 30K images with five semantically equal captions each, leading to semantic overlap within the 60K stealing-queries.

We further explore the impact of the number of stealing queries on the performance of the stolen copies. Our results in Table 5 in Appendix C highlight a performance decrease when reducing the number of stealing queries. The performance drop is most significant between 10k and 20k queries. This motivates an evaluation of the effectiveness of our method to re-use representations for semantically similar sentences in this setup. We query the stolen model copy with 10k sentences

Table 2: **Re-using Representations for Stealing NLP Encoders.** We assign the same extracted representation to a given query and its semantically similar sentences. # Samples denotes the final number of sentences used to fine-tune the stolen encoder. We steal from the victim encoders trained on the nli and flickr datasets using the queries from the nli dataset. We use the same notation as in Table 1.

EN		D_v	D_s	# Queries	# Samples	CR	MPQA	MR	MRPC	SST2	SUBJ	TREC	Avg.$_{\text{STS}}$	Avg.$_{\text{Tran}}$	SICKR	STSB	AVG↑	Δ↓
				10000	10000	84.58	87.64	77.68	75.69	83.94	**93.56**	**69.19**	59.25	81.75	56.63	61.87	75.62	9.70
f_s	B*	nli	nli	**10000**	**20000**	**86.89**	**88.59**	**78.56**	**76.42**	**85.44**	93.29	68.62	**68.35**	**82.54**	**65.30**	**71.40**	**78.67**	**6.65**
				20000	20000	87.42	89.11	79.13	75.66	86.93	93.45	70.14	69.35	83.12	65.51	73.18	79.36	5.96
				10000	10000	72.74	78.71	70.21	72.25	79.01	91.12	53.23	59.26	73.9	54.67	63.86	69.91	12.40
f_s	B	flickr	nli	**10000**	**20000**	**73.38**	**81.39**	**71.3**	71.15	**80.85**	**91.82**	**55.92**	**69.95**	**75.12**	**65.67**	**74.23**	**73.71**	**8.60**
				20000	20000	73.75	83.33	71.59	72.15	80.28	91.53	57.65	71.5	75.75	66.63	76.37	74.60	7.72

from the nli dataset and assign the obtained representation also to the semantically equal positive partner of each sentence. This results in 20k fine-tuning samples for the stolen copy. Our results in Table 2 highlight that the performance of the stolen copy with only 10k queries (augmented to 20k data points) is similar to the original stealing with 20k sentences.

To assess if a given source of semantically similar sentences is adequate for stealing a given encoder, we do not measure the semantic similarity between sentences, but the similarity of their representations. We query the victim encoder with a few pairs of sentences and compare their cosine similarity scores. We run an experiment using sentences from nli, qqp, and flickr datasets on the victim encoders based on BERT and trained on nli and flickr datasets as well, we use 200 sentences overall, where there are 100 semantically similar pairs, and present results in Table 3. We observe that stealing using the nli dataset works better since the sentences have higher cosine similarity scores. By such a quick assessment, we can determine which dataset might be better to use for a given stealing process.

Table 3: **Semantic Similarity.** We measure the semantic similarity between sentences from a given dataset using victim models trained on nli, qqp, and flickr datasets. Model extraction using nli works better than with flickr or qqp since the sentences from the nli dataset have higher cosine similarity scores.

Victim model trained on ↓	nli data	flickr data	qqp data
nli	**0.77±0.16**	0.59±0.16	0.62±0.18
qqp	**0.77±0.16**	0.67±0.13	0.65±0.19
flickr	**0.82±0.15**	0.71±0.14	0.76±0.16

We also experiment with the setup where the adversary does not know the exact architecture of the victim encoder but only knows the family of possible encoders and the output dimensionality. Then, they might instantiate their stolen copy with an architecture different from the victim model's. An example of different architectures with the same output dimensions is BERT-large and RoBERTa-large (both output representations of size 1024). In Table 6 (see Appendix C), we compare the performance of encoders stolen to the victim encoder's original architecture vs the other similar architecture. We observe that stealing from a larger encoder (RoBERTa-large) to a smaller one (BERT-large) performs better than in the opposite direction.

5.3 Defending

We present the performance of the watermarked encoders as well as the success of our watermarking in Table 4. Our results highlight that a relatively small number of fine-tuning steps (*e.g.,* 200 alternations between original and downstream tasks) are sufficient to embed the watermark into the encoder while preserving the high performance of the defended encoder on other unrelated and general downstream tasks. Furthermore, the embedded watermark successfully transfers to stolen copies. We use the initial fine-tuned sentence embedding encoder as the independent encoder, which is the most difficult scenario for the ownership resolution since the only difference between the victim and independent encoders results from the watermarking process. We show that even in this worst-case, the independent model is never incorrectly resolved as being stolen. To compute the

Table 4: **Watermarking Sentence Embedding Encoders.** We embed the watermark into an encoder and present the performance of the downstream task, the underlying encoder, and the comparison between the victim f_v, stolen f_s, and independent f_i encoders. *Steps* denotes the number of fine-tuning steps for watermarking, Agreement (Agr) and accuracy (Acc) on the watermark downstream task (given in %), test loss (L), p-value (p), and effect size ($\Delta\mu$).

Steps	$Agr(f_v, f_s)$	$Agr(f_v, f_i)$	$Acc(f_v)$	$L(f_v)$	STS	SICKR	STSB	$Acc(f_s)$	$L(f_s)$	$p(f_v, f_s)$	$\Delta\mu(f_v, f_s)$	$p(f_v, f_i)$	$\Delta\mu(f_v, f_i)$
100	97.59	88.3	57	0.68	0.77	0.74	0.79	56.42	0.68	0.79	0.27	0.0016	3.15
200	**96.44**	**67.54**	**65.37**	**0.65**	**0.76**	**0.73**	**0.79**	**64.79**	**0.65**	**0.64**	**0.47**	**1.74E-14**	**7.73**
300	97.71	72.59	65.37	0.63	0.76	0.73	0.8	65.14	0.63	0.62	0.5	9.87E-22	9.71
400	96.56	59.98	69.95	0.61	0.76	0.73	0.79	70.18	0.61	0.49	0.69	1.43E-27	11.07
500	96.33	60.66	71.1	0.59	0.75	0.71	0.79	71.1	0.6	0.46	0.74	2.67E-28	11.24
600	96.79	62.73	71.56	0.59	0.74	0.69	0.78	72.02	0.59	0.44	0.78	4.65E-32	12.03
700	96.44	62.61	71.44	0.58	0.72	0.67	0.77	72.25	0.58	0.45	0.76	2.30E-35	12.69
800	96.79	56.31	73.85	0.56	0.71	0.66	0.76	72.71	0.57	0.45	0.76	2.91E-42	14
900	96.34	59.75	73.39	0.55	0.71	0.65	0.76	72.13	0.56	0.42	0.8	3.02E-45	14.53

p-values, we leverage the softmax outputs for the correct labels from the downstream task and use the t-test. The p-values indicate that there is a significant difference between the distribution of the confidence scores from independent vs victim encoders (p-value $<$ 5%). In contrast, the difference is not significant between the victim and stolen encoders.

6 Conclusions and Future Work

Through SSL, large amounts of data can be leveraged to train ML models, even when no labels for this data are available. Due to the costly training procedure on dedicated and expensive hardware, this approach is, however, limited to parties that can afford it. Such parties can then offer APIs access to their trained high-value encoders for generating representations of given input text for other parties, and thereby monetize their encoders. We demonstrate how to steal such transformer-based encoders by using only their output representations. Our stealing requires up to 40x fewer queries than the number of training data points used to train the victim and it yields stolen copies with comparable performance on standard benchmarks. This provides an incentive for attackers to steal the encoders at a much lower computing cost and the required number of data points than the training of such encoders from scratch. To protect the encoders, we propose a method for embedding a watermark into the encoders by fine-tuning a defended encoder with a specific downstream task. Our work highlights that, in particular, state-of-the-art sentence embedding encoders are easily stolen but hard to defend. Future work should address the lack of active defenses against stealing encoders that could prevent attacks as they are happening without degrading the performance of legitimate users. These defenses should increase the cost of extraction in terms of computation power and the required amount of data for queries to disincentivize attackers from stealing the sentence embedding encoders.

Acknowledgments

We would like to acknowledge our sponsors, who support our research with financial and in-kind contributions: DARPA through the GARD project, Microsoft, Intel, CIFAR through the Canada CIFAR AI Chair and AI catalyst programs, NFRF through an Exploration grant, and NSERC COHESA Strategic Alliance. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute https://vectorinstitute.ai/partners. Finally, we would like to thank members of CleverHans Lab for their feedback.

References

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1615–1631, 2018.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical*

Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2015.

Ting Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning*, 2020.

Clarifai. https://clarifai.com, 2023. URL `https://clarifai.com`.

Cohere. https://cohere.ai, 2023. URL `https://cohere.ai/`.

Tianshuo Cong, Xinlei He, and Yang Zhang. Sslguard: A watermarking scheme for self-supervised learning pre-trained encoders. *CoRR*, abs/2201.11692, 2022. URL `https://arxiv.org/abs/2201.11692`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Adam Dziedzic, Nikita Dhawan, Muhammad Ahmad Kaleem, Jonas Guan, and Nicolas Papernot. On the difficulty of defending self-supervised learning against model extraction. In *International Conference on Machine Learning*, 2022a.

Adam Dziedzic, Haonan Duan, Muhammad Ahmad Kaleem, Nikita Dhawan, Jonas Guan, Yannis Cattan, Franziska Boenisch, and Nicolas Papernot. Dataset inference for self-supervised models. In *Advances in Neural Information Processing Systems*, 2022b. URL `https://openreview.net/forum?id=CCBJf9xJo2X`.

Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. Analyzing and improving representations with the soft nearest neighbor loss. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2012–2020. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/frosst19a.html`.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6894–6910, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.552. URL `https://aclanthology.org/2021.emnlp-main.552`.

Xuanli He, Lingjuan Lyu, Qiongkai Xu, and Lichao Sun. Model extraction and adversarial transferability, your BERT is vulnerable! *arXiv preprint arXiv:2103.10013*, 2021.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1367–1377, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1162. URL `https://aclanthology.org/N16-1162`.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. First quora dataset release: Question pairs. URL `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`.

Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1345–1362, 2020.

Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1937–1954, 2021.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 512–527. IEEE, 2019.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper/2015/file/f442d33fa06832082290ad8544a8da27-Paper.pdf`.

Kalpesh Krishna, Gaurav Singh Tomar, Ankur P. Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves on sesame street! model extraction of bert-based apis. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Byl5NREFDr`.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJvJXZb0W`.

Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset inference: Ownership resolution in machine learning. In *Proceedings of ICLR 2021: 9th International Conference on Learning Representationsn*, 2021.

OpenAI. https://openai.com, 2023. URL `https://openai.com/`.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against dnn model stealing attacks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SyevYxHtDB`.

Zeyang Sha, Xinlei He, Ning Yu, Michael Backes, and Yang Zhang. Can't steal? cont-steal! contrastive stealing attacks against image encoders. 2022. URL `https://arxiv.org/abs/2201.07513`.

Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training NLP models: A concise overview. *CoRR*, abs/2004.08900, 2020. URL `https://arxiv.org/abs/2004.08900`.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 601–618, 2016.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, pp. 269–277, 2017.

Aäron van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/N18-1101`.

Yutong Wu, Han Qiu, Tianwei Zhang, Lin Jiwei, and Meikang Qiu. Watermarking pre-trained encoders in contrastive learning. *ArXiv*, abs/2201.08217, 2022.

Qiongkai Xu, Xuanli He, Lingjuan Lyu, Lizhen Qu, and Gholamreza Haffari. Beyond model extraction: Imitation attack for black-box NLP APIs. *arXiv preprint arXiv:2108.13873*, 2021.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

Santiago Zanella-Beguelin, Shruti Tople, Andrew Paverd, and Boris Köpf. Grey-box extraction of natural language models. In *International Conference on Machine Learning*, pp. 12278–12286. PMLR, 2021.

A Additional Related Work

Self-Supervised Learning. In computer vision, one of the most popular self-supervised algorithms is contrastive learning (Chen et al., 2020; van den Oord et al., 2018), where representations that come from differently transformed views of the same image are brought closer to each other and the representations from views of different inputs are repelled. In NLP, a popular self-supervised pre-training approach is to mask selected words in the input sequence and train the model to predict that masked words Devlin et al. (2018).

Transformers. Transformer (Vaswani et al., 2017) is becoming a ubiquitous architecture in NLP and computer vision. While the original transformer consists of an encoder and decoder component, our work only studies the encoder part for representation learning. Transformers are composed of several identical layers, namely a multi-head attention sublayer followed by a feed forward sublayer. The multi-head attention sublayer utilizes the self-attention mechanism to learn the pairwise relationships between all tokens. Self-attention is the key to the success of transformers, which makes parallel training and learning long-range dependency between tokens easier.

NLP transformers. In this work, we investigate BERT-based models (Devlin et al., 2018), pre-trained bidirectional transformers (Vaswani et al., 2017). In addition to BERT Base, we also use TinyBERT (Jiao et al., 2019), obtained by distilling BERT to a smaller transformer architecture, and RoBERTa Large (Liu et al., 2019), which is pre-trained on a larger dataset than BERT. Then we analyze the task of learning highly generic sentence representation, a fundamental problem in NLP (Kiros et al., 2015; Hill et al., 2016; Logeswaran & Lee, 2018).

B Details on the Experimental Setup

B.1 Datasets

nli. We use (Gao et al., 2021)'s nli-for-SimSCE dataset, consisting of 275,602 data rows from SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018). Each row holds three sentences, an original sentence, a positive entailment and a contradiction. In training, the contradiction acts as a hard-negative.

qqp. We use the merve/qqp dataset from Hugging Face. The train split consists of 2,607,949 data rows, each holding two semantically equal questions. We use this data for training as positive pairs.

flickr. The flickr dataset consists of images, each annotated with five human-written captions. Following (Gao et al., 2021), we consider any two captions of the same image as a positive pair. We split the training and test set to 90%, 10%, making sure that all caption-pairs related to one image end up in the same set. This yields 286,050 positive-pair training examples. When using flickr as a dataset for stealing, we drop the duplicates arising from generating all possible caption pairs before sampling the stealing-queries. Thereby, we mitigate a too small diversity over the stealing due to repeated queries.

C Additional Results

C.1 Stealing with Fewer Queries

We depict the effect of stealing with fewer queries in Table 5.

Table 5: **Number of Stealing Queries and Impact on Model Performance.** We follow SimCSE (Gao et al., 2021) and use the SentEval banchmark. f_v denotes the victim encoder trained on data D_v. f_s is the stolen encoder extracted using queries from a given stealing dataset D_s. For stealing, we use a different number of queries to the victim encoder and fine-tune our stolen copy with the resulting outputs for 20 epochs. We use the BERT-based model pre-trained and taken from Hugging Face obtained by the SimCSE code-base, marked with an asterisks * (`https://github.com/princeton-nlp/SimCSE`) and steal using the nli dataset.

EN		D_v , D_s	# Queries	CR	MPQA	MR	MRPC	SST2	SUBJ	TREC	Avg.$_{\text{STS}}$	Avg.$_{\text{Tran}}$	SICKR	STSB
f_s	B*	nli	60k	**89.05**	88.95	**80.49**	74.98	**86.35**	**93.39**	66.73	**81.45**	**82.85**	**79.84**	**83.07**
			50k	88.04	88.68	78.96	75.74	86.01	92.76	64.49	80.98	82.1	78.96	82.99
			40k	87.83	88.29	78.87	75.96	86.12	92.78	64.91	80.67	82.11	78.99	82.36
			30k	87.18	89.13	78.26	**76.4**	84.4	92.83	65.33	79.09	81.93	76.9	81.29
			20k	86.41	**89.16**	78.45	75.44	84.4	92.98	71.37	74.39	82.60	71.93	76.84
			10k	83.81	87.52	78.08	74.39	84.06	93.28	**73.27**	60.03	82.06	53.86	66.2
			5k	78.96	83.77	76.04	72.74	84.29	92.86	69.24	48.18	79.7	44.0	52.36
			1k	80.15	83.09	77.42	70.68	84.75	93.0	68.12	45.45	79.67	53.14	37.75

C.2 Influence of the Architecture

We depict the effect of the architecture of the stolen model in Table 6.

Table 6: **Base Architecture Influence on Stolen Model Performance.** We follow SimCSE (Gao et al., 2021) and use the SentEval banchmark. f_v denotes the victim encoder trained on data D_v. f_s is the stolen encoder extracted using queries from a given stealing dataset D_s. For stealing, we use 60,000 stealing queries to the victim encoder and fine-tune our stolen copy with the resulting outputs for 20 epochs. We compare performance between encoders that steal from victims BERT-large (BL) and RoBERTa-large (R) into a stolen copy with either of this architecture. The victim models are taken from Hugging Face obtained by the SimCSE code-base, marked with an asterisks * (`https://github.com/princeton-nlp/SimCSE`) and steal using the nli dataset. We use the public tokenizer corresponding to the stolen copy's architecture.

EN		D_v	D_s	f_v	CR	MPQA	MR	MRPC	SST2	SUBJ	TREC	Avg.$_{\text{STS}}$	Avg.$_{\text{Tran}}$	SICKR	STSB
f_s	BL*	nli	nli	BL (Baseline)	90.49	89.95	83.5	75.98	89.45	93.86	68.67	83.67	84.56	81.54	85.81
				R	82.88	73.05	76.34	70.61	83.83	85.55	42.39	42.44	73.52	36.51	48.37
f_s	R*	nli	nli	R (Baseline)	92.37	90.52	88.04	76.64	92.31	95.13	91.20	83.76	89.46	81.95	86.70
				BL	90.24	89.67	83.07	76.25	89.68	94.07	72.45	78.96	85.06	77.27	80.65