Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Studiengang Künstliche Intelligenz

Masterarbeit

von

Fabian **Schmidt**

Anwendbarkeit von bekannten Extraktionsangriffen auf Neuronale Netze für Zeitreihen

Applicability of Known Model Extraction Attacks against Time Series Neural Networks.

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Studiengang Künstliche Intelligenz

Masterarbeit

von

Fabian **Schmidt**

Anwendbarkeit von bekannten Extraktionsangriffen auf Neuronale Netze für Zeitreihen

Applicability of Known Model Extraction Attacks against Time Series Neural Networks.

Bearbeitungszeitraum: von 1. Oktober 2024
bis 31. März 2025

1. Prüfer: Prof. Dr. Patrick Levi

2. Prüfer: Prof. Dr. Daniel Loebenberger

OTH Amberg-Weiden
Elektrotechnik, Medien und Informatik

Eigenständigkeitserklärung gemäß § 27 (8) ASPO

Name und Vorname
der Studentin/des Studenten: **Schmidt, Fabian**

Studiengang: **Künstliche Intelligenz**

Ich bestätige, dass ich die Masterarbeit mit dem Titel:

Anwendbarkeit von bekannten Extraktionsangriffen auf Neuronale Netze für Zeitreihen

selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Datum: 31st March 2025

Unterschrift: Fabian Schmidt

OTH Amberg-Weiden
Elektrotechnik, Medien und Informatik

Masterarbeit Zusammenfassung

Studentin/Student (Name, Vorname): **Schmidt, Fabian**
Studiengang: Künstliche Intelligenz
Aufgabensteller, Professor: Prof. Dr. Patrick Levi
Durchgeführt in (Firma/Behörde/Hochschule):
Betreuer in Firma/Behörde:

Ausgabedatum: 1. Oktober 2024 Abgabedatum: 31. März 2025

Titel:

Anwendbarkeit von bekannten Extraktionsangriffen auf Neuronale Netze für Zeitreihen

---

Zusammenfassung:

In dieser Arbeit werden Model Extraction Attacks auf maschinelle Lernmodelle untersucht, insbesondere in einem Black-Box-MLaaS-Szenario, in dem ein Angreifer nur Zugriff auf die Vorhersagen des Zielmodells hat. Ziel dieser Angriffe ist es, ein Ersatzmodell zu erstellen, das die Entscheidungsgrenzen des ursprünglichen Modells nachahmt. Die Analyse konzentriert sich auf verschiedene Faktoren, die den Erfolg dieser Angriffe beeinflussen, darunter die Komplexität des Ersatzmodells, die Größe des gestohlenen Datensatzes und der Einfluss von künstlich generierten oder verrauschten Daten.

Die Experimente zeigen, dass auch einfache Ersatzmodelle hohe Extraktionsgenauigkeit erreichen können, wodurch die Notwendigkeit architektonischer Ähnlichkeit zwischen Original- und Ersatzmodell in Frage gestellt wird. Zudem wird festgestellt, dass Angriffe mit synthetischen Daten fast ebenso effektiv sein können wie solche mit realen Daten, was die potenzielle Bedrohung durch generative Modelle verstärkt.

Die Ergebnisse unterstreichen die Dringlichkeit effektiver Gegenmaßnahmen. Bestehende Verteidigungsstrategien erweisen sich als unzureichend, sodass zukünftige Forschung sich auf die Entwicklung von Schutzmechanismen konzentrieren sollte, die die Modellgenauigkeit bewahren, während sie Extraktionsangriffe abschwächen oder detektieren.

Schlüsselwörter: Maschinelles Lernen, Zeitreihendaten, Modell Extraktions Angriffe, Black-Box Angriff,

Contents

Symbols, formula symbols and units

Chapter 1

Introduction

1.1 Background and Motivation

Deep learning has emerged as a state-of-the-art technology that has transformed machine learning by delivering exceptional performance in areas such as computer vision, natural language processing, and time series analysis [12]. However, deep learning comes with high computational costs during both training and inference. To address these challenges, many organizations have adopted the Machine-Learning-as-a-Service (MLaaS) model. In MLaaS, models are hosted on public cloud platforms such as AWS[1] and Microsoft Azure[2], allowing clients to leverage powerful models via APIs without investing heavily in local computational infrastructure.

In a typical MLaaS architecture, a model undergoes two phases: training and prediction. Once trained, the model is deployed on a cloud server where it remains accessible to clients through API endpoints. Clients submit data for prediction, and the model returns results without exposing its internal workings. MLaaS providers typically charge a small fee per query. However, this introduces a vulnerability: repeated queries made through these APIs can unintentionally reveal information about the model's behavior. Model extraction attacks exploit this weakness by using the prediction outputs of a public model, referred to as the *original model*, to train an adversary's own *substitute model*. This substitute model can approximate or even surpass the performance of the original model, even when the attacker uses a relatively small dataset and significantly less computational power [30].

Despite the growing recognition of model extraction attacks, most existing research has focused on relatively simple model architectures such as logistic regression or shallow deep neural networks. State-of-the-art extraction attacks such as CopycatCNN [4] and Knockoff Nets [22] are designed for the extraction of image models like ResNet [14] and VGG [27]. However, extraction attacks on time series data remain significantly underexplored. The different characteristics of time series, compared to

[1] https://aws.amazon.com/de/ai/machine-learning/
[2] https://azure.microsoft.com/en-us/products/machine-learning/

image or speech data, introduce different computational processes and data handling challenges. As time series models are increasingly deployed in areas like finance, anomaly detection, and predictive maintenance, it is essential to understand the conditions under which model extraction attacks succeed in these contexts.

Moreover, model extraction attacks offer strategic and economic advantages. Developing a high-performance deep learning model typically requires extensive domain knowledge, considerable computational resources, and access to large, often proprietary, datasets. By replicating a model through extraction, an adversary can bypass these demanding requirements, effectively stealing the intellectual property embedded within the original model. This not only eliminates the need for costly and time-intensive development but also enables competitors to rapidly deploy systems with capabilities comparable to those of the original, potentially undermining the competitive edge of the model owner. Such attacks, therefore, pose a dual threat: they challenge the security and integrity of MLaaS deployments while also reducing the incentives for innovation and investment in specialized, high-value models.

1.2 Problem Statement

Model extraction attacks have been shown to effectively clone proprietary models by only leveraging the input-output behavior of a target system, thereby circumventing the substantial costs associated with training a new model from scratch [30]. However, the majority of these studies have focused on tasks like object detection, facial emotion recognition or natural language understanding [19], thus leaving a gap in our understanding of how these attacks perform against time series neural networks. Time series models, including architectures like Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNNs), are frequently deployed in environments where continuity and precise temporal modeling are essential. This raises important questions: Are the current extraction techniques robust when faced with sequential data, and how does the quality and quantity of attack data affect the success of these techniques? Addressing these issues is critical to evaluating the security posture of time series applications and for devising appropriate countermeasures.

1.3 Research Objective

The primary objective of this thesis is to evaluate the feasibility and effectiveness of model extraction attacks against time series neural networks. To achieve this goal, the research is structured around several key objecjtives:

1. Train several original models on time series data[3]: These models will serve as targets of the extraction attacks. The model's test accuracy will be used as benchmark for the substitute models

[3] https://timeseriesclassification.com/dataset.php

2. Run extraction attacks against original models: As mentioned above, CopycatCNN and KnockoffNet attacks will be used to extract the original model.
3. Investigate defenses against extraction attacks: Several defenses against extraction attacks have been proposed. Their applicability to time series data and effectiveness will be investigated
4. Investigate dataquality factors: Examine the effects of using noisy attack data as well as synthetic data generated via a Variational Autoencoder (VAE) [16] on the success of the extraction process

1.4 Scope and Limitations

This study is dedicated to examining model extraction attacks specifically within the domain of time series neural networks for classification tasks. The experimental evaluation will focus on ten widely adopted architectures, including classical models such as LSTM and RNN, as well as various Temporal Convolutional Network (TCN) architectures. Furthermore, state-of-the-art architectures, namely Arsenal [21], Hydra, MultiRocketHydra [6], Rocket [5], MiniRocket [7], and MultiRocket [29], are incorporated to serve as original models for extraction attacks.

The investigation is conducted under the assumption of a black-box-MLaaS model. In this framework, the adversary is limited to querying the target model and observing its outputs, with no insight into the internal structure or access to the original training data. Moreover, it is assumed that the attacker has unlimited access to the model, facing no constraints on the volume of queries that may be submitted.

It should be noted that this study is subject to certain limitations. First, the focus is solely on model extraction attacks targeting time series classification tasks. Consequently, the extraction of models used for other time series applications such as forecasting, regression, or anomaly detection remains an open area for future investigation. Additionally, all experiments are conducted on locally deployed models, rather than in real-world MLaaS environments. In practical MLaaS scenarios, factors such as rate limiting, authentication measures, and query budget could impact the feasibility and effectiveness of extraction attacks. Addressing these factors in future work would provide a more comprehensive understanding of model extraction vulnerabilities across a broader range of tasks and deployment contexts.

1.5 Thesis Structure

The structure of this thesis is designed to provide a systematic exploration of model extraction attacks against time series neural networks. Chapter 2, *Related Work*, reviews the existing literature on model extraction and highlights the gaps in current research, particularly in relation to time series data. Chapter 3, *Methodology*, outlines the experimental framework used in this study, detailing the design of the target models, the implementation of attack strategies, and the preparation of both real and synthetic

datasets. Chapter 4, *Experiments*, describes the experimental protocols, including the various configurations of steal dataset sizes, possible defenses, noise injection levels, and VAE-generated data scenarios. In Chapter 5, *Results*, the empirical findings are analyzed, with a focus on the interplay between dataset characteristics, attack performance, and model complexity. Finally, Chapter 6 concludes the thesis by summarizing the key contributions, discussing the implications of the research for model security, and proposing directionsc for future work.

Chapter 2

Related Work

This chapter provides a review of existing literature on model extraction attacks.

2.1 Model Extraction Attacks Revisited

The paper, Model Extraction Attacks Revisited [19], presents an empirical study on the vulnerability of Machine-Learning-as-a-Service (MLaaS) platforms to model extraction (ME) attacks. The authors introduce MeBench, an open-source benchmarking platform that integrates multiple attack strategies, evaluation metrics, a variety of piracy models, and benchmark datasets. Using MeBench, the study evaluates ME attacks across leading MLaaS providers including Amazon, Microsoft, Face++, and Google on tasks such as facial emotion recognition (FER) and natural language understanding (NLU). The authors investigated the evolution of ME attacks on MLaaS platforms from 2020 to 2022 using 1.7 million queries.

The authors found that these MLaaS platforms are highly vulnerable to ME attacks, while vulnerability varies greatly with datasets and task. They also identify many influential factors on the attacks performance like the optimizer used for the substitute model, the substitutes model architecture, the use of a pre-trained model, and the sampling strategy used for the attack. The authors also investigate a possible defense against ME attacks but found it to be barely effective.

2.2 Model Extraction Attacks against Recurrent Neural Networks

The paper, Model Extraction Attacks against Recurrent Neural Networks by Takemura, Yanai and Fujiwara [28] investigates the feasibility and effectiveness of extracting RNN models from LSTM models. The work is motivated by the need to understand the security risks for MLaaS, especially in the context where RNNs are used to handle time series data. The main goal is to determine whether an attackeer can efficiently

train a RNN-based substitute model that achieves performance comparable to that of the target LSTM model, despite having only a fraction of the original training data.

The authors investigated both Classification and Regression Tasks. For the Classification Task, they identified *leaky time* as a step in the LSTMs prediction process where the model predictions are particularly reliable. The attacker then applies a distillation process using the softmax with temperature to train a substitute RNN model using these leaky timesteps. The authors used the MNIST dataset [8] but transformed it into time series data. With 20% of the original training data they managed to train a substitute RNN model with 97.5% accuracy, matching the 97.3% accuracy of the original LSTM model.

For the regression task, the authors used an air quality data set [32]. They used six values, i.e., temperature, humidity and average values of CO, NO, NO_2. These six time series values over 72 hours were given to an RNN model. The model was then supposed to predict the 73rd hour as output. They used the coefficient R^2 of determination for the evaluation of the accuracy of the original and substitute models. The original model achieved an R^2 of 0.8992. They trained a substitute RNN and LSTM model both scoring a R^2 greater than 0.85.

This paper, especially the Classification Task, served as main inspiration for this thesis. Although their approach successfully achieved a substitute model accuracy matching that of the original LSTM using only 20% of the training data, the dataset handling and experimental scope appear limited. Specifically, the exclusive focus on MNIST and the comparison between only RNNs and LSTMs restrict the generalizability of their findings. These limitations underscore the necessity to broaden the investigation to include a wider variety of actual time series datasets and more model architectures.

2.3 Copycat CNN: Stealing Knowledge by Persuading Confession with Random Non-Labeled Data

Correia-Silva et al. (2018) demonstrated a novel approach to copying black-box CNN models using only random unlabeled images. Their attack, Copycat CNN, operates in two phases: first generating a fake dataset by querying the target model with random images and collecting their predictions ("stolen labels"), then training a copycat network using this synthetic dataset. The authors validated their approach across three image domains (facial expression recognition, object classification, and satellite crosswalk detection) and against Microsoft Azure's Emotion API. Their results showed that copycat networks could achieve at least 93.7% of the original model's performance using non-problem domain data, and up to 99.3% when combining both random and domain-specific images. This work revealed a significant vulnerability in deployed AI models, particularly those offered as cloud services, as it demonstrated that high-performing replicas could be created without access to training data or model architecture details.

However, it is important to note that the Copycat CNN attack was originally demon-

strated exclusively on CNN architectures and applied to image data. In this thesis, the principles of this attack were extended to time series data and explored its applicability across more model architectures. By adapting the attack to time series neural networks and evaluating its effectiveness on different architectures, this research aims to uncover potential vulnerabilities in systems that operate on sequential data, thereby broadening the scope of model extraction attacks beyond traditional image-based applications.

2.4 Knockoff Nets: Stealing Functionality of Black-Box Models

This paper [22] introduces an innovative adaptive extraction strategy designed to improve the efficiency of constructing a transfer set for training a knockoff model from a black-box target. Rather than relying on random queries from a large pool of images, the adaptive approach dynamically learns to select the most informative samples. This is achieved by associating each image with a label from a predefined hierarchy, structured in a coarse-to-fine tree, and using a reinforcement learning approach that uses a softmax-based sampling mechanism. At each decision point, the policy selects a label (or "action") to determine which images to query, and a reward signal is computed based on the quality of the victim model's responses. This feedback is then used to update the policy via a gradient bandit algorithm, progressively guiding the query process toward images that yield more valuable pseudo-labels.

The reward signal is designed to capture several aspects of query quality. A margin-based certainty measure rewards images for which the victim model exhibits high confidence, indicated by a significant gap between the top predicted probabilities. To prevent the policy from converging on a narrow subset of images, a diversity reward is incorporated, ensuring a broader exploration of the input space. Additionally, a loss-based reward is used to target images where the knockoff's predictions diverge most from those of the victim, highlighting opportunities for improvement. The combination of these rewards allows the adaptive strategy to efficiently balance exploration and exploitation, thereby reducing the number of queries required to build an effective transfer set.

Experimental results demonstrate the clear benefits of this adaptive approach. In both closed-world and open-world scenarios, where the overlap between the adversary's image pool and the victim's training data varies, the knockoff models trained using the adaptive strategy reach performance levels close to those of the victim model, typically recovering between 0.81× and 1.05× the victim's accuracy. Notably, the adaptive strategy not only accelerates convergence but also enhances the interpretability of the attack by pinpointing regions of the input space where the victim model exhibits strong confidence. These findings underscore that adaptive querying can significantly improve sample efficiency while maintaining high fidelity in the extracted model.

2.5 A Survey of Model Extraction Attacks and Defenses in Distributed Computing Environments

In cloud computing environments, particularly within MLaaS platforms, models are exposed via standardized APIs that offer detailed output information such as confidence scores and probability distributions. This accessibility makes them prime targets for query-based model extraction attacks, where attackers systematically gather input-output pairs to replicate the target model. To mitigate these threats, defenses like output perturbation (e.g., adding noise or truncating confidence scores) and query monitoring (to detect abnormal access patterns) have been proposed. These techniques aim to obscure critical details and limit excessive querying, thereby hindering an adversary's ability to extract a high-fidelity replica.

However, the authors note that while these defenses can provide a noticeable level of protection, they come with inherent trade-offs. Specifically, the challenge lies in balancing the disruption of extraction attempts with the preservation of service quality for legitimate users. The effectiveness of these measures is often limited by the strict performance constraints of cloud services, such as maintaining low latency and high prediction accuracy. In conclusion, although these defense strategies offer promising initial protection, they are not foolproof, attackers may adapt to these countermeasures, and further research is needed to develop more robust, unified defense or detection frameworks for cloud-based machine learning models. [33]

2.6 Summary and Gaps in Literature

The reviewed literature underscores that model extraction attacks pose a significant threat across various application domains, particularly within cloud-based MLaaS platforms. Research such as that by Liang et al. [19], Orekondy et al. [22] and Correia-Silva et al. [4] has demonstrated that attackers can effectively reconstruct or approximate target models using only black-box access and a finite query budget. Adaptive strategies, exemplified by the Knockoff Nets approach, have pushed the state-of-the-art by leveraging reinforcement learning to balance exploration and exploitation, resulting in highly efficient attacks that produce substitute models closely matching the performance of their victims. Similar vulnerabilities have been highlighted across various architectures, from CNNs for image recognition to RNNs and LSTMs for sequential data.

However, most prior studies have focused on domains such as image recognition and natural language processing, leaving a notable gap in the investigation of extraction attacks in the context of time series data. Given that many real-world applications, such as financial forecasting, health monitoring, and IoT systems, rely on time series models, there is a need to extend these attack methodologies to sequential data. Existing work on time series model extraction, like the approach by Takemura et al. [28], provides promising results but is limited in scope, using datasets such as MNIST transformed into time series or a single air quality dataset. This narrow focus restricts

our understanding of the broader applicability and limitations of these attacks in time series settings. Additionally, current defenses, such as output perturbation and query monitoring, have been primarily evaluated on non-sequential data, indicating a critical gap wether or not these strategies work effectivly on time-series data. Addressing these gaps is essential to developing a more comprehensive understanding of extraction attacks in time series applications.

Chapter 3

Methodology

This chapter outlines the methodology used to evaluate model extraction attacks on time series classification models. The primary objective is to investigate the extent to which an adversary can replicate a target model's functionality.

The chapter begins by defining the model extraction attack framework, detailing the assumptions about attack scenarios and the metrics used to assess attack success. Next, the datasets used for experimentation, including WalkingSittingStanding (WSS), PenDigits, and SpokenArabicDigits (SAD), are introduced, along with the preprocessing steps applied.

An overview of the target neural network architectures follows, covering conventional deep learning models such as CNNs and RNNs, as well as specialized time series classifiers like ROCKET and its variants.

Finally, Variational Autoencoders (VAEs) are introduced as a method for generating synthetic data to facilitate model extraction attacks. The choice of VAEs over alternative generative models, such as GANs and diffusion models, is justified, and the architecture and training process of the VAE used in this thesis are outlined.

3.1 Overview of the Model Extraction Attack Framework

The attack framework assumes a black-box MLaaS scenario, where the adversary only has access to the model's predictions and does not posses any prior knowledge of its architecture, hyperparameters, or training data, like when using a MLaaS-API. The attacker uses the previously introduced attacks, CopycatCNN and KnockoffNets, to approximate the target model's decision boundary. The attacks will be carried out multiple times with increasing dataset sizes to evaluate the importance of stolen dataset size. The effectiveness of the attacks will be meassured by the substitute model accuracy.

When running model extraction attacks, there mainly two metrics to consider: accuracy and fidelity. Accuracy measures the correctness of predictions made by the extracted

model on the test distribution. Fidelity, in contrast, measures the general agreement between the extracted and victim models on any input. Both of these objectives are desirable, but they are in conflict for imperfect victim models: a high-fidelity extraction should replicate the errors of the victim, whereas a high-accuracy model should instead try to make an accurate prediction. This thesis will only report accuracy.

3.2 Dataset Selection and Preprocessing

For experimental evaluation, three datasets were selected to evaluate the effectiveness of model extraction attacks: WalkingSittingStanding(WSS) [23], PenDigits [1], and SpokenArabicDigits(SAD) [3]. This section will introduce these datasets along with the taken preprocessing steps and train-validation-test-splits.

WalkingSittingStanding: In the creation of this Dataset partitioners were tasked to perform six activities (Walking, Walking upstairs, Walking downstairs, Sitting, Standing and Laying) while wearing a smartphone on the waist. Using the smartphones embedded accelerometer and gyroscope, 3-axial linear acceleration and 3-axial angular velocity were captured at a constant rate of 50Hz. The Dataset was labelled manually.

PenDigits: The PenDigits dataset comprises 250 handwritten digit samples collected from 44 writers using a WACOM PL-100V pressure-sensitive tablet. Each digit was captured as a sequence of (x, y) coordinates at 100-millisecond intervals, with pressure data omitted. To ensure invariance to translation and scale distortions, the coordinates were normalized so that the maximum range varied between 0 and 100. For consistent feature representation, digits were spatially resampled to a fixed number of points, with $T = 8$ found to offer the best balance between accuracy and complexity.

SpokenArabicDigits: This dataset contains timeseries of mel-frequency cepstrum coefficients (MFCCs) corresponding to spoken arabic digits. It includes data from 44 male and 44 female native Arabic speakers. This dataset consists of 8800 samples (10 digits x 10 repetitions x 88 speakers) each with 13 channels. The Speakers were between the ages of 18 and 40.

Using multiple datasets, the research aims to investigate the robustness and adaptability of model extraction attacks across various data types and application domains.

Preprocessing: To ensure comparability across the three datasets, normalization was applied to scale the inputs to a common range, as each dataset originally featured different value ranges. Although each dataset provided a predefined train/test split, these were not used in order to maintain a consistent train/validation/test ratio across all datasets for unbiased evaluation. 70% of the total samples contribute to the training set, the remaining 30% were split equally among validation and test set. Additionally, for the PenDigits dataset, where the time series sequences consist of only 8 points, padding was applied as some of the models used in the experimental section require a minimum length of 10.

To load these datasets the Aeon-Python Library [20] was used. Using this Library the datasets can be loaded with only a few lines of code:

```
from aeon.datasets import load_classification
dataset = "WalkingSittingStanding"
X, y = load_classification(dataset)
```

The Aeon Library supports 190 time series dataset and many useful features when working with time series data. See Figure 1, 2, 3 for an Illustration of each class of the described datasets.

Dataset	Training Samples	Validation/Test Samples	Channels	Time series length	Classes
WalkingSitting Standing	7208	1546	3	206	6
PenDigits	7693	1650	2	10	10
SpokenArabic Digits	6158	1320	13	65	10

Table 3.1: Overview of used datasets

Table 3.1 provides an overview of the datasets used in the thesis, detailing the number of training and validation/test samples, channels, time series length, and the number of classes. The PenDigits dataset is the shortest, with a time series length of 10 (originally 8) and two channels, making it ideal for evaluating models on brief, low-dimensional sequences. In contrast, the WalkingSittingStanding dataset is the longest, with a time series length of 206 and three channels, representing more complex temporal dependencies. The SpokenArabicDigits dataset falls in between, with a medium-length time series of 65 and 13 channels, making it useful for assessing models on multi-channel data. These datasets were specifically chosen to cover a range of sequence lengths and channel dimensions, ensuring a diverse evaluation of model performance.

3.3 Target Neural Networks Architectures

As described in Chapter 1.4 several architectures were used for the experiments. The following section will give a brief overview about the characteristics of each architecture.

3.3.1 ROCKET-based Architectures

ROCKET: RandOm Convolutional KErnel Transform

ROCKET [5] is a time series classification method designed to balance computational efficiency with high predictive accuracy. It utilizes a large number of random convolutional kernels (default 10000), like traditional CNNs, to compute two aggregate features from each feature map, producing two real-valued numbers as features per kernel: the maximum value (global max pooling) and proportion of positive values(or ppv). All aspects of these kernels are random: length, weights, bias, dilation, and padding. The computed features are then used to train a linear classifier. ROCKET is compatible with any classifier; the default is, however, a Ridge Classifier. Unlike CNNs, where the weights are learned, ROCKET does not learn any kernel weights, thus speeding up the training process significantly.

MINIROCKET: MINImally RandOm Convolutional KErnel Transform

MINIROCKET [7] builds on top of the findings of ROCKET as it is a simplified, yet just as powerful and accurate version of ROCKET, but significantly faster. This is achieved by removing most of the randomness of the ROCKET architecture, e.g., the length of kernels is fixed to 9, the kernel weights are not drawn from a $\mathcal{N}(0,1)$ distribution but fixed to be either -1 or 2, the padding is fixed and only the proportion of positive values is used for the training of the classifier, thus reducing the amount of features by half. With these changes, MINIROCKET is up to 75 times faster than ROCKET.

MultiRocket

MultiRocket [29] is based on MiniRocket, using the same set of kernels as MiniRocket. There are two main differences. First, MultiRocket transforms a time series into its first order difference. Then both the original and the first order difference time series are convolved with the kernels. Second, in addition to PPV, MultiRocket generated 3 additional values per kernel. Mean of positive values, Mean of indices of positive values and longest stretch of positive values. Finally the transformed features are used to train a linear classifier.

HYDRA: HYbrid Dictionary-Rocket Architecture

Hydra [6] is a dictionary-based time series classification method that integrates random convolutional kernels, blending elements of dictionary methods and convolutional approaches like ROCKET. Hydra groups competing convolutional kernels, where each input time series is convolved with these kernels, and the best-matching kernel per group is counted at each time step. To enhance feature extraction, Hydra employs both hard counting (tracking occurrences of the highest-response kernel) and soft counting (accumulating response values), optionally including the first-order difference of the time series for improved accuracy. A key hyperparameter, the number of groups (g) and kernels per group (k), determines whether Hydra behaves more like a traditional dictionary method or a convolutional-based approach. Like ROCKET and its variants, Hydra produces transformed feature representations that are classified using ridge regression or logistic regression.

MultiRocketHydra

MultiRocketHydra [29] is a combination of the MultiRocket and Hydra architecture. The input is given to both models and their combined feature outputs are used to train a classifier.

Arsenal

Arsenal [21] is an ensemble learning approach that extends ROCKET by using multiple instances of ROCKET transformations with a RidgeClassifierCV as the base classifier. Unlike a single ROCKET model, Arsenal trains multiple classifiers independently, each using randomly initialized ROCKET transformations, and assigns a weight

to each classifier based on its cross-validation accuracy. This weighted ensemble approach enhances robustness and allows for probability estimates, but at the cost of increased computational complexity compared to a single ROCKET classifier. Arsenal supports different ROCKET variants, including MiniRocket and MultiRocket, and can be configured with a time constraint to limit training duration.

Every model presented in this section was trained using their default hyperparameters.

3.3.2 Other Architectures

Several 1D convolutional neural networks were used as original model. Their architectures and training procedure will be discussed briefly.

BaseCNN1D serves as a foundational model with two convolutional layers followed by pooling and one fully connected layer. **Deep1DCNN** extends this by adding an extra convolutional layer, enabling deeper hierarchical feature extraction while maintaining efficiency. **Simple1DCNN** is a simplified version of these two models with only one convolutional layer, max pooling and one fully connected layer. **Residual1DCNN** builds on top of the Deep1DCNN, adding one more fully connected layer and residual connections [24], allowing information to bypass certain layers, which helps mitigate the vanishing gradient problem and improve training stability. Lastly, **Attention1DCNN** introduces self-attention mechanisms [31].

BaseRNN and **BaseLSTM** are very basic RNN and LSTM models. The input is passed through the RNN/LSTM cell, the last hidden state gets passed through a fully connected layer and softmax activation function to make the prediction.

Each model in this section was trained for 100 epochs with an early stopping criterion, allowing a patience of 10 epochs. Training was conducted with a batch size of 64. Weights for convolutional, fully connected, and multi-head attention layers were initialized using Xavier initialization [11]. For the RNN and LSTM models, a hidden size of 128 was set, along with two RNN/LSTM layers. The Adam optimizer [17] was applied with a learning rate of 0.001.

Table 3.2 compares models based on training time, inference time, and test accuracy across three datasets. Non-ROCKET-based architectures train quickly (10.7–42.3s) with negligible inference time but show varying accuracy, with Dilated1DCNN, Residual1DCNN, and LSTM performing the best. ROCKET-base architectures achieve superior accuracy (often 99%+) but require significantly more computation, with Arsenal exceeding 1000s training time. MiniRocket offers a strong balance, delivering near-Rocket accuracy with much lower computational cost.

Model	Training Time (in seconds)			Inference Time (in seconds)			Test Accuracy (in %)		
	WSS	PenDigits	SAD	WSS	PenDigits	SAD	WSS	PenDigits	SAD
Simple1DCNN	16.1	17.0	14.0	0.0	0.0	0.0	42.01	44.63	30.23
Deep1DCNN	21.9	22.7	18.3	0.0	0.0	0.0	78.32	70.65	86.36
Dilated1DCNN	20.7	15.2	17.8	0.0	0.0	0.0	87.70	95.33	97.65
Attention1DCNN	28.7	42.3	25.8	0.0	0.0	0.0	72.62	93.69	95.30
Residual1DCNN	14.5	23.7	18.6	0.0	0.0	0.0	82.65	92.90	95.68
BaseCNN1D	17.6	19.4	14.9	0.0	0.0	0.0	51.97	72.65	42.80
RNN	21.2	18.3	10.7	0.0	0.0	0.0	84.98	94.72	93.71
LSTM	36.5	33.3	22.2	0.0	0.0	0.0	81.88	95.39	97.20
Arsenal	1039	575.8	985	33.5	2.1	17.1	87.18	99.58	99.77
Hydra	104.2	10.3	48.8	10.5	0.1	2.4	81.26	98.67	98.56
MultiRocketHydra	143	58.1	69.4	20.4	1.1	5.8	76.44	99.21	99.47
Rocket	264	57.8	153.9	48.3	2.5	25.9	90.74	99.58	99.77
MiniRocket	48.5	46.4	30.8	2.0	0.2	0.7	93.59	99.27	99.62
MultiRocket	117.2	69.1	63.5	13.8	1.4	4.7	92.43	99.39	99.62

Table 3.2: Training and Inference Time and Test Accuracy per Model and Dataset

3.4 Variational Autoencoder

Section 4.4 experiments with generated data for the attacks. There are three popular methods to create new samples from known data. Generative Adversarial Networks (GANs) [13], Diffusion Models [15]. This thesis used the third method Variational Autoencoder (VAE) [16]. VAEs, GANs, and Diffusion models are all generative techniques but differ in their approach. VAEs use an encoder-decoder structure to learn a probabilistic latent space, optimizing a reconstruction loss with Kullback-Leibler-Divergenz for stable training. GANs play an adversarial minimax game between a generator and discriminator, often leading to sharper samples but suffering from instabilities like mode collapse. Diffusion models iteratively add and remove noise, achieving high-quality outputs but at a high computational cost. VAEs are the easiest to train since they lack any adversarial dynamics, directly optimize likelihood, and converge reliably, making them more stable and computationally efficient than GANs and Diffusion models. For these reasons, VAE was chosen for this thesis, ensuring reliable training while maintaining generative performance.

3.4.1 Architecture

A VAE architecture consists of an encoder, a latent space representation, and a decoder. The encoder used in this thesis is composed of two 1D convolutional layers with ReLU activations. The output is then flattened and mapped to two fully connected layers, which generate the mean (μ) and variance (σ^2) of the latent distribution. The reparameterization trick is applied to sample a latent vector while ensuring differentiability. To incorporate class information, a class embedding layer is added, which is concatenated with the latent representation before decoding. Additionally, an auxiliary classifier is introduced to encourage a more structured latent space. This classifier consists of fully connected layers that take the latent mean as input and predict the corresponding class label. By optimizing the classification loss alongside the VAE's objective, the model

is encouraged to learn class specific latent representations, improving the quality of generated samples. The decoder consists of fully connected and transposed convolutional layers, which reconstruct the input by upsampling the latent noise vector of size 1×1024. A final interpolation step ensures that the output matches the original input size.

The loss function used to train the VAE consists of three components:

- **Reconstruction loss**: Mean squared error (MSE) between the reconstructed output $\hat{x}$ and the original input x:

$$\mathcal{L}_{\text{recon}} = \sum ||x - \hat{x}||^2$$

- **KL divergence loss**: A regularization term ensuring that the learned latent distribution remains close to a standard normal distribution:

$$\mathcal{L}_{\text{KL}} = -\frac{1}{2} \sum \left(1 + \log \sigma^2 - \mu^2 - \sigma^2\right)$$

- **Auxiliary classification loss**: A cross-entropy loss to encourage the latent space to be class-discriminative:

$$\mathcal{L}_{\text{class}} = -\sum y \log \hat{y}$$

The total loss function is a weighted sum of these components:

$$\mathcal{L}_{\text{total}} = 2\mathcal{L}_{\text{recon}} + 0.1\mathcal{L}_{\text{KL}} + 0.5\mathcal{L}_{\text{class}} + 0.2\mathcal{L}_{\text{KL}}$$

$$\mathcal{L}_{\text{total}} = 2\mathcal{L}_{\text{recon}} + 0.3\mathcal{L}_{\text{KL}} + 0.5\mathcal{L}_{\text{class}}$$

These weighting factors were empirically chosen to balance reconstruction quality, latent space structure, and classification performance.

Figure 3.1 illustrates this architcture.

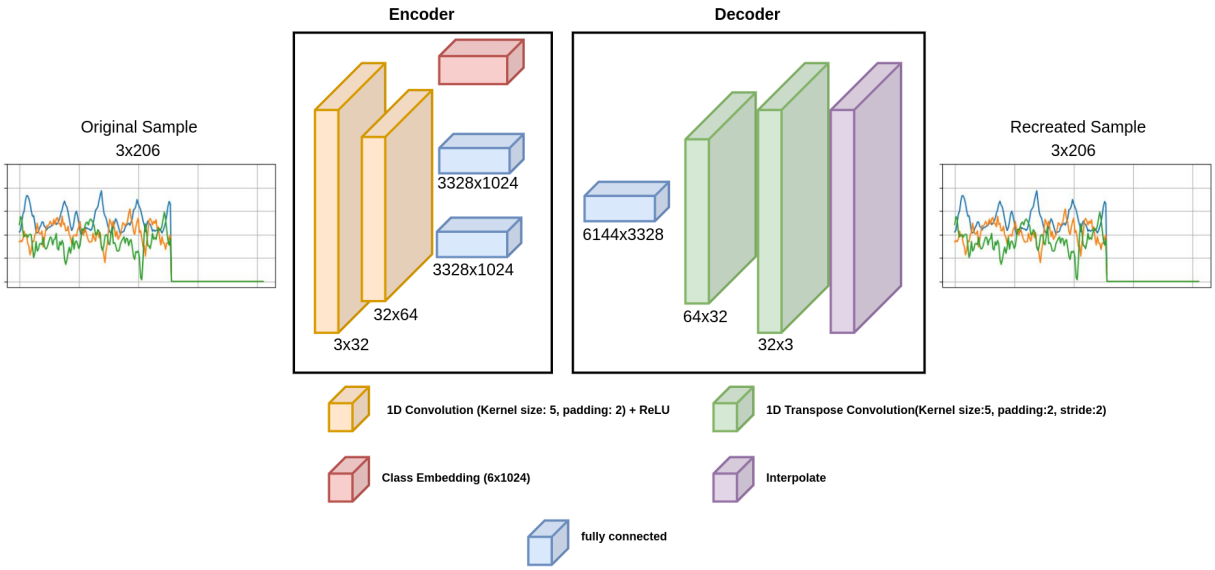

Figure 3.1: VAE Architecture

3.4.2 VAE Training

Seeing as the VAE is supposed to be used on the attacker side, it is trained on multiple percentages of the attackers data per class. VAEs were trained for 10 - 60% of the attackers data per class at 10% intervals. Table 3.3 shows the total amount of samples used for training at different percentages.

Dataset	10%	20%	30%	40%	50%	60%
WalkingSittingStanding	153	307	462	617	772	925
PenDigits	160	325	490	656	822	985
SpokenArabicDigits	127	261	392	525	656	790

Table 3.3: VAE training dataset sizes

3.4.3 Generated Samples

Figure 3.2 shows one sample per class of the WalkingSittingStanding dataset. On the left side real samples on the right side samples generated by a VAE trained on 50% of the attackers data, so 772 samples. Considering that this VAE has only seen 7.4% of the entire dataset, the generated samples demonstrate a remarkable level of quality compared to the real data. See Figure 4 and 5 for generated samples of the other two datasets.

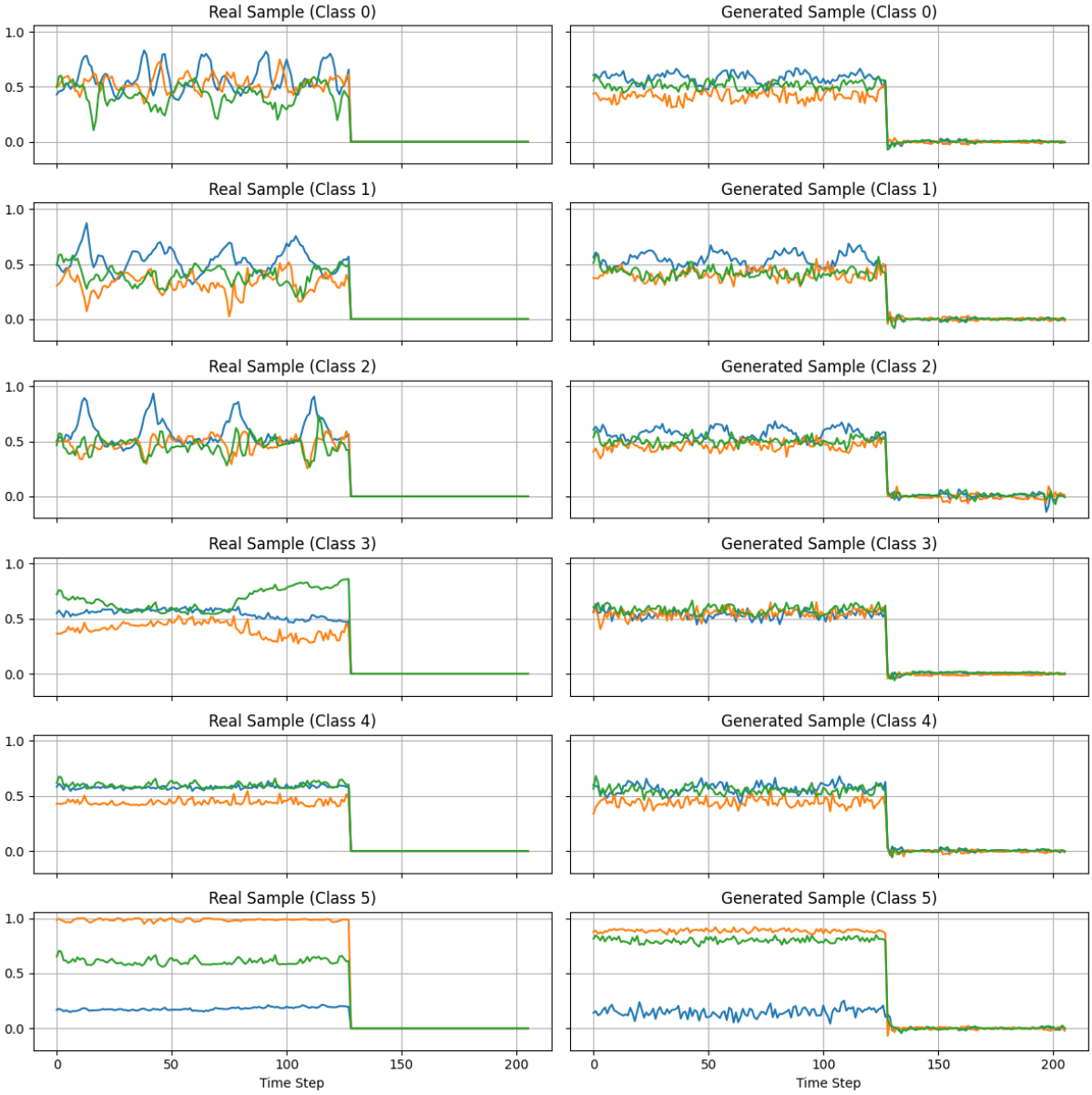

Figure 3.2: Real Samples vs. Generated Samples

Chapter 4

Experiments

This chapter presents the experimental evaluation of model extraction attacks on time series classification models. The primary goal is to assess the effectiveness of different attack strategies and analyze the impact of defenses designed to mitigate such attacks.

Two well-known model extraction attacks are evaluated: CopycatCNN and KnockoffNets. To demonstrate the extent of the vulnerability, experiments are conducted on a variety of combinations of target and substitute model architectures. This includes scenarios where the adversary employs a model architecture significantly different from the original model. The results highlight that model extraction attacks remain highly effective even when the adversary's model architecture deviates considerably from the original.

The evaluation begins with a basic attacks, where an adversary uses the attacks with clean data to extract a substitute model. Following this, the role of defense mechanisms is explored, categorized into preprocessing and postprocessing defenses, which aim to reduce the success of model extraction attacks without significantly degrading model performance.

Next, attack scenarios are examined where the adversary posses a noisy dataset. This analysis provides insights into the robustness of extraction attacks in real-world conditions where clean data may not always be available.

Then attacks using generated data are explored, where synthetic samples are used to query the original model. In particular, the impact of generative models, such as Variational Autoencoders (VAEs), on attack efficiency and success rates is analyzed.

Finally these experiments are combined to simulate the real world scenario, where an adversary has access to some original and some noise data and uses these to train a VAE. This combined dataset is then used to query a defended model.

These experiments provide a comprehensive understanding of the vulnerabilities of time series classification models to model extraction attacks and evaluate the effectiveness of countermeasures in defending against them. The implementation of the experiments makes heavy use of the adversarial-robustness-toolbox(ART) [10]

4.1 Basic Attacks

In this attack scenario, the adversary has access to up to 50% of the test dataset, referred to as the stealing dataset. The attacks are conducted multiple times, each using an increasing portion of the stealing dataset. The models described in Sections 3.3.1 and 3.3.2 (excluding Simple1DCNN and BaseCNN1D) serve as the original models for extraction. As substitute models, the architectures BaseCNN1D, Deep1DCNN, Simple1DCNN, BaseRNN, and BaseLSTM are used, resulting in a total of 50 different model combinations.

The model extraction process is outlined in the pseudocode below (4.1):

```
Initialize results list
For each model combination:
    Load original Model
    Wrap Model in PyTorchClassifier
    Define attack sample sizes as percentages of test data
    For each sample size:
        Select subset of test data, ensuring all classes are represented

        For each attack:
            Initialize surrogate model
            Run attacks using stolen samples
            Evaluate surrogate model on remaining test data
            Store attack performance results
```

Listing 4.1: Extraction Attack Pseudocode

Initially, attacks are performed using just one sample per class. As shown in listing 4.1, the stealing dataset is incrementally increased in steps of 5%, reaching up to 50%. Consequently, each attack is executed 12 times in total. The substitute models are trained using the same batch size, learning rate, number of epochs, and optimizer as the original models. The results of this experiment using the Simple1DCNN as substitute model on the SpokenArabicDigits dataset are illustrated in Figure 4.1. As these results become quite extensive, only this model combination on the SpokenArabicDigits dataset will be shown. Attack results on the other datasets can be found in Appendix A. For results on the other datasets and model combinations, see the associated GitHub repository [25]. Noteable results will be discussed.

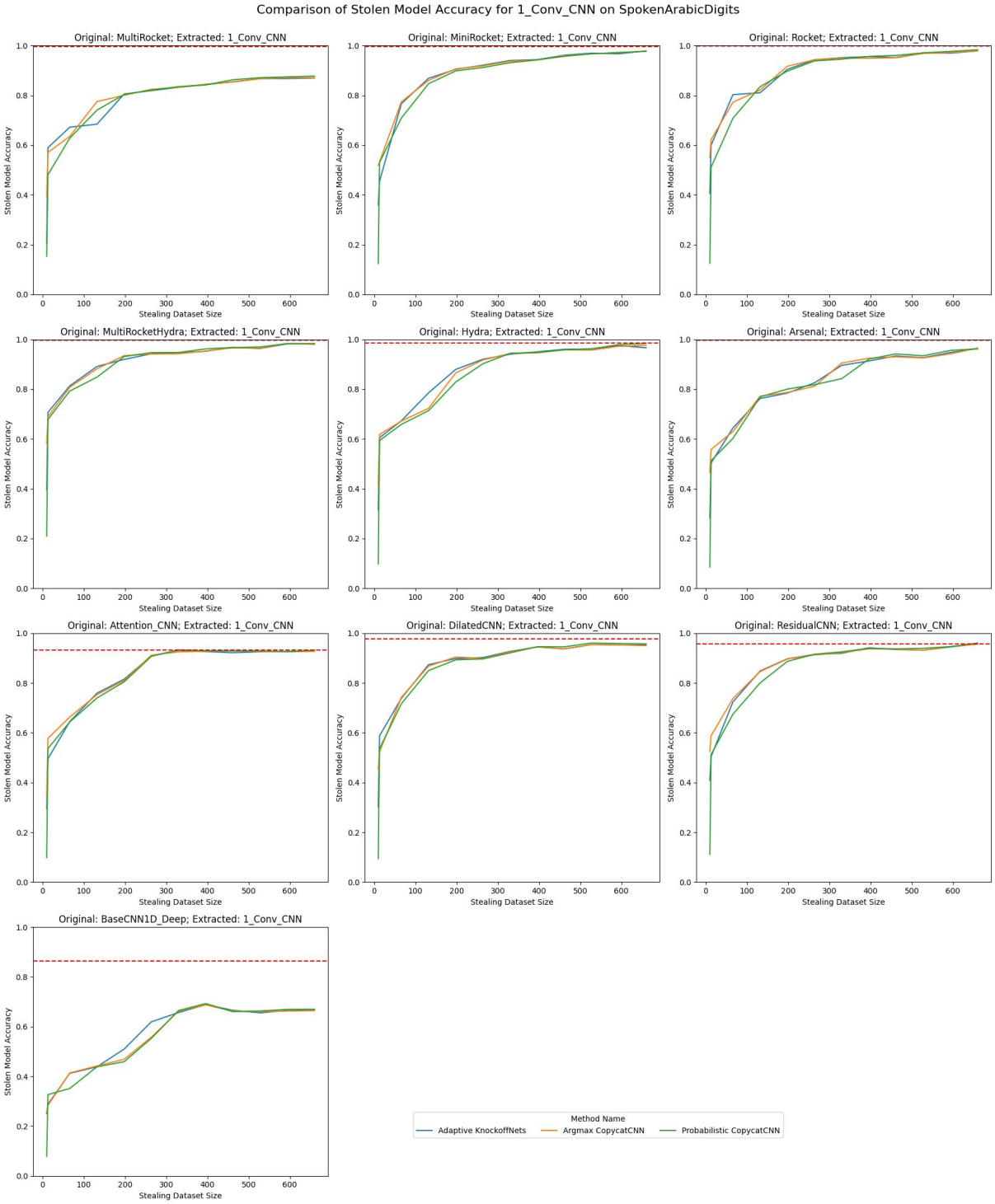

Figure 4.1: Results of basic extraction attack using the SpokenArabicDigits dataset

As expected, when trained on a small dataset, the substitute model initially performs poorly. However, as the size of the stealing dataset increases, its accuracy improves dramatically, eventually even matching the accuracy of the original model (indicated by the red line).

This trend is consistent across all model combinations, despite differences in architecture. See 6, 7, 8, 9. These finding are consistent across all investigated datasets. This suggests that an attacker can successfully extract a model using virtually any substitute model architecture, without needing it to match the original model's architecture. Also, increasing the substitute model complexity does not necessarily result in a better substitute model accuracy. These findings align with Observation 3 from [19].

4.2 Defenses

Many defenses against model extraction attacks have been proposed. Broadly, they can be divided into two categories: preprocessing and postprocessing defenses.

- **Preprocessing Defenses**: These methods modify the input before it is processed by the model. This can include adding noise to the input data, applying transformations, or obfuscating features to reduce the effectiveness of stolen queries.
- **Postprocessing Defenses**: These strategies focus on modifying the model's output, making it more difficult for an attacker to reconstruct the model's decision boundaries. Techniques include confidence masking, returning class labels instead of probabilities, adding noise to the output.

While these defenses can make extraction more difficult, they often come with trade-offs, such as reduced model accuracy or increased computational overhead. Designing an effective defense is particularly difficult because one must find a balance between security and maintaining the model's performance. A defense that is too strong may significantly degrade the model's utility, while a weaker defense may not offer sufficient protection.

To address this challenge, a hyperparameter search was conducted to optimize the defense mechanisms. Using the Optuna [9] library, the search explored all parameters offered by a defense. The optimization criterion, to find a good set of parameters for a defense, was to minimize the accuracy of the substitute and to minimize the accuracy difference between the original and the protected original model. This should ensure the defense is as strong as possible while maintaining the accuracy of the original model as much as possible.

4.2.1 Preprocessing Defenses

As previously stated, the implementation makes heavy use of the ART-library as it provides many AI-Security features. The library has many preprocessing defenses implemented [1], sadly only three of these defenses are applicable to time series data.

- **GaussianAugmentation**: This defence adds a $\mathcal{N}(0, \sigma^2)$ noise to models inputs or to a ratio of the models inputs. Parameters include the σ for the noise distribution and the ratio of input to which the defense should be applied to

[1] https://github.com/Trusted-AI/adversarial-robustness-toolbox/tree/main/art/defences/preprocessor

- **FeatureSqueezing** [26]: Feature squeezing reduces the impact of adversarial noise by limiting the precision of input features to a defined bit level.
- **TotalVarMin** [2]: Total Variance Minimization reduces adversarial noise by minimizing total variance. This defense smooths samples while preserving important structures, making perturbations less effective.

Defense	Parameter	PenDigits	SpokenArabicDigits	WalkingSittingStanding
Gaussian Augmentation	Sigma	0.6635	0.048	0.5
	Augmentation	True	False	True
	Ratio	0.7419	0.92	0.537
	Clip	(0.0, 0.6437)	(0.0, 0.45)	(-1.3, 1.7)
	Apply Predict	False	True	Flase
Feature Squeezing	Clip	(0.0, 0.7670)	(0.0, 0.212)	(0.0, 1.37)
	Bit Depth	1	13	13
	Apply Fit	False	True	True
	Apply Predict	False	False	False
TotalVarMin	Clip	(0.0, 0.874)	(0.0, 0.1411)	(-1.35, 1.43)
	Norm	1	2	1
	Solver	CG	L-BFGS-B	L-BFGS-B
	Lamb	0.44	0.2	0.164
	Max Iter	12	1	13
	Prob	0.26	0.49	0.442
	Apply Fit	True	True	True
	Apply Predict	False	False	False

Table 4.1: Optimized Hyperparameters for Different Defenses

The hyperparameter optimization ran for 50 trials for all defenses on all datasets. Table 4.1 shows the found Hyperparameters. Figure 4.2 shows results of the extraction attacks on Deep1DCNN using Simple1DCNN as substitute model architecture with an original model defended by the discussed preprocessing defenses. Although these defenses don't negatively impact the orginal model performance, they don't offer any protection against the extraction attack as the substitute model approaches the original model accuracy even with a stealing dataset. Figure 10 and 11 show similar results on the other datasets.

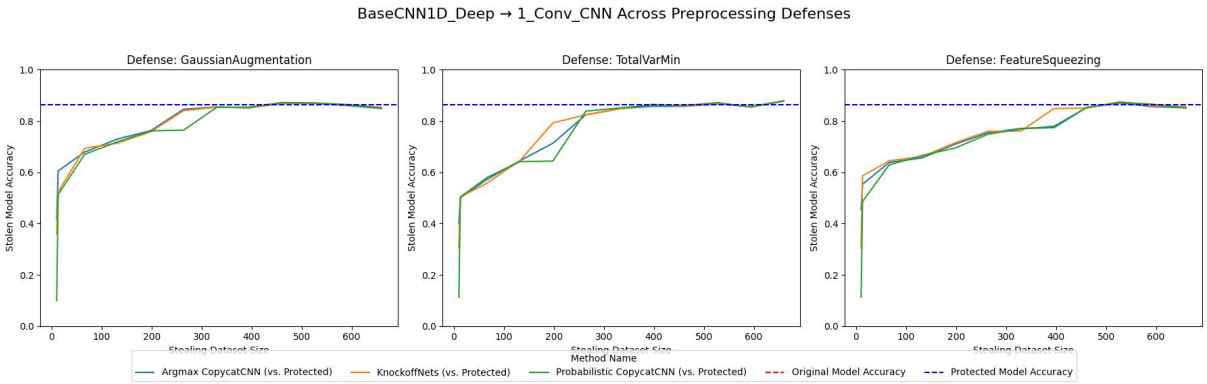

Figure 4.2: Preprocessing defended extraction attacks using SpokenArabicDigits

For results on the other 49 model combinations and other datasets, see the associated GitHub repository [25].

4.2.2 Postprocessing Defenses

The ART-library implements six postprocessing defences[2] all of which are applicable to time series data.

- **ClassLabels**: Instead of returning probabilities, for Multi-Class Classification this returns a One-Hot encoded vector, for binary classification this returns either 1 or 0.
- **GaussianNoise**: Applies a $\mathcal{N}(0, \sigma^2)$ distributed noise on the model's prediction.
- **HighConfidence**: Sets all the probabilites of the model's prediction below a cutoff to 0.
- **ReverseSigmoid** [18]: Applies the reverse sigmoid transformation to reduce the confidence of high-confidence predictions.
- **Rounded**: Rounds the model's output to given number of decimals

A hyperparameter search was conducted the same way as in chapter 4.2.1. Table 4.2 shows the hyperparameters found for each defence on every dataset.

Defense	Parameter	PenDigits	SpokenArabicDigits	WalkingSittingStanding
ClassLabels	Apply fit	True	True	Flase
	Apply Predict	False	False	Flase
GaussianNoise	Scale	0.146	0.062	0.112
	Apply Fit	False	True	True
	Apply Predict	True	True	True
HighConfidence	Cutoff	0.187	0.474	0.150
	Apply Fit	False	False	False
	Apply Predict	False	True	True
ReverseSigmoid	Beta	0.893	0.911	1.12
	Gamma	0.176	0.019	0.085
Rounded	Decimals	4	1	4
	Apply Fit	False	False	False
	Apply Predict	False	True	False

Table 4.2: Optimized Hyperparameters for Different Defenses

As before the attacks were conducted on all 50 model combinations. Now with an originial model defended by one of the discussed postprocessing defenses. Figure 4.3 shows the results of these attacks on the SpokenArabicDigits dataset.

[2]https://github.com/Trusted-AI/adversarial-robustness-toolbox/tree/main/art/defences/postprocessor

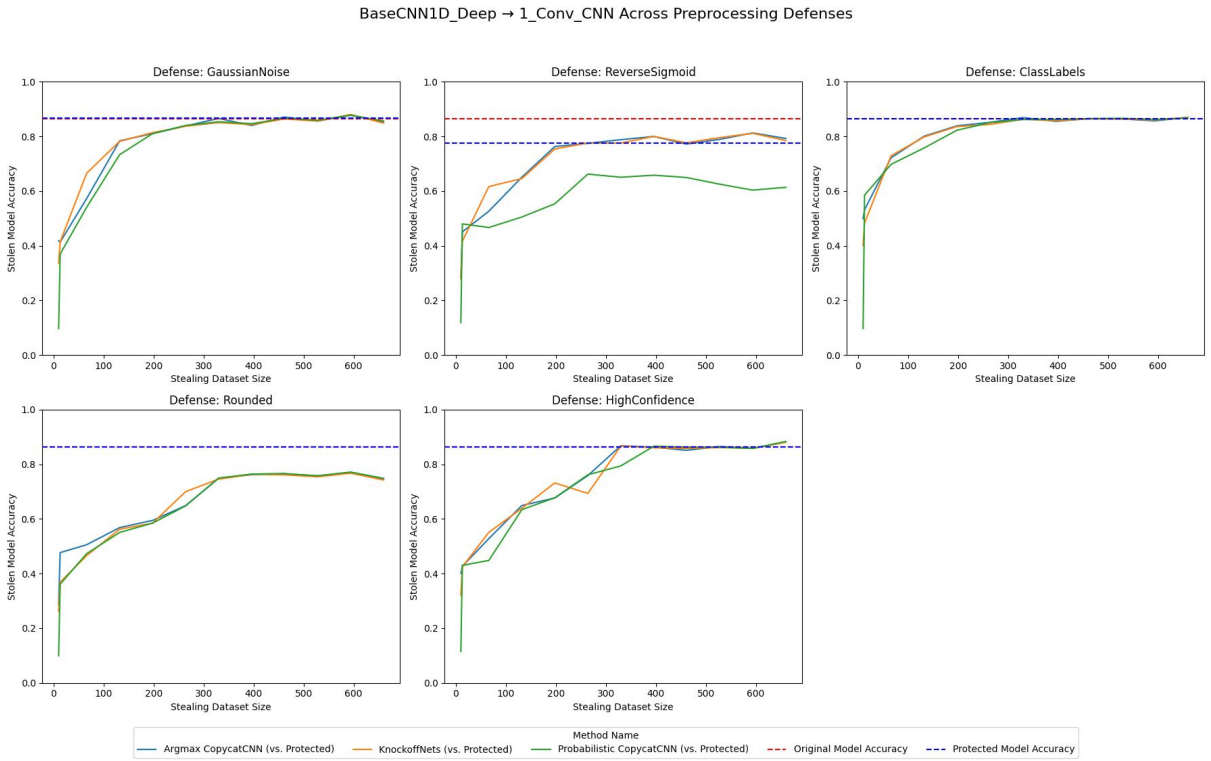

Figure 4.3: Postprocessing defended extraction attacks using SpokenArabicDigits

As Figure 4.3 illustrates, the ReverseSigmoid defense is highly effective. Compared to the base scenario, it significantly reduces the accuracy of the substitute model. However, this comes at the cost of the defended model's accuracy, which drops by 8.7% (3.8% for PenDigts, 0% for WalkingSittingStanding) compared to the undefended model. Despite this drop in model performance, the substitute models still reached the accuracy of the defended. In contrast, the other postprocessing defenses appear largely ineffective, as they fail to meaningfully reduce the substitute model's accuracy. The relatively poor performance of ReverseSigmoid on the WalkingSittingStanding dataset is likely due to the specific hyperparameters used (used Beta > 1, other datasets used Beta < 1), suggesting that a different set of parameters could improve the defences effectiveness.

The results indicate that the investigated defenses provide little to no protection against model extraction attacks, as most fail to significantly reduce the accuracy of the substitute model. Even the most effective defense, ReverseSigmoid, only offers partial mitigation while still allowing extraction with some success. This aligns with existing research [19, 33] which highlights the difficulty of designing robust defenses against extraction attacks without severely degrading model performance.

4.3 Attacks using noisy Data

In this section, attacks using noisy data are explored to evaluate their impact on model extraction. Noisy data is introduced to simulate scenarios where the attacker has access

to data similar to the original but with perturbations. To assess the attacks's robustness, attacks were conducted at increasing noise levels, allowing for an analysis of how well extraction attacks perform under less precise but still relevant input conditions.

```
def add_noise(data, noise_level=0.01):
    noise = np.random.normal(0, np.std(data) * noise_level, data.shape)
    return data + noise
```

Listing 4.3 shows the Python-code used to add noise to a dataset. This was used with noise_level 0.1 to 0.8 at 0.1 intervalls. Figure 4.4 shows the influence of these noise_levels on a sample.

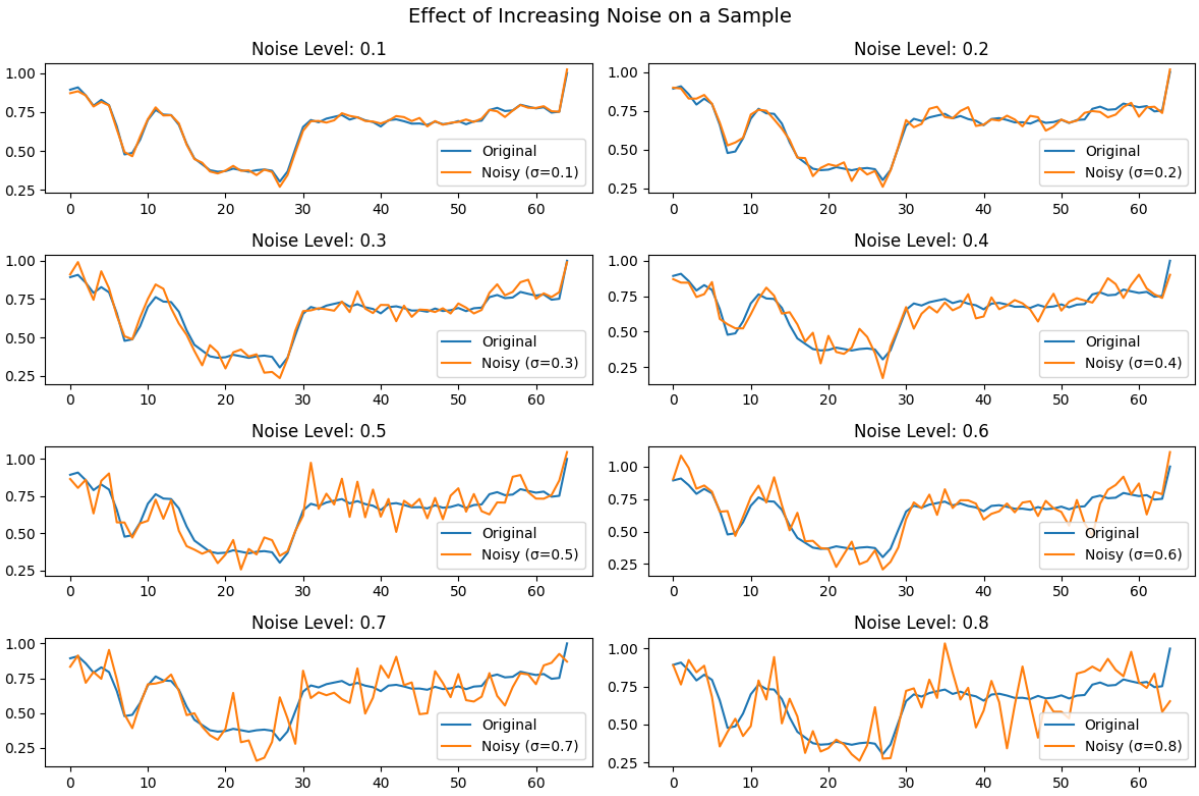

Figure 4.4: Influence of noise on a sample

Figures 4.5 and 4.6 show the experimental results on the Deep1DCNN and Arsenal original model using Simple1DCNN as substitute model on the SpokenArabicDigits dataset with increasing noise levels. As figure 4.5 shows, increasing noise on the stealing dataset does not negatively impact the performance of the substitute model. Figure 4.6 on the other hand shows a noteable decrease in substitute model accuracy as noise increases. This implies that attacks on ROCKET-based original models are susceptible to noise. That is confirmed by attacks on other datasets and model combinations (see GitHub). Especially using the WalkingSittingStanding dataset, substitute models resort to random guessing even with a low amount of noise further indicating that the presence of noise in the stealing dataset can severely impede the accuracy of the substitute model. Additionally, attacks across all model combinations

using WalkingSittingStanding show a significantly larger decrease in substitute model accuracy compared to the other two, shorter datasets, suggesting that noise has a much stronger impact on longer sequences.

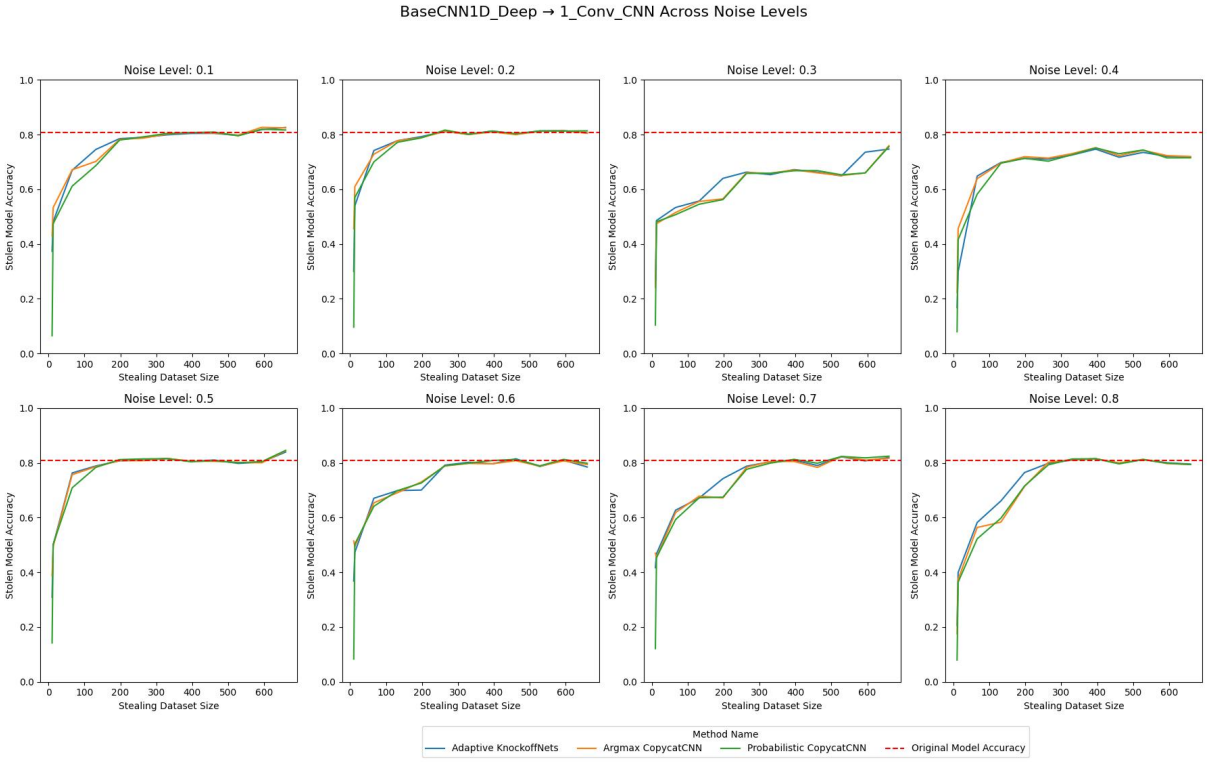

Figure 4.5: Influence of noise on extraction attacks using SpokenArabicDigits

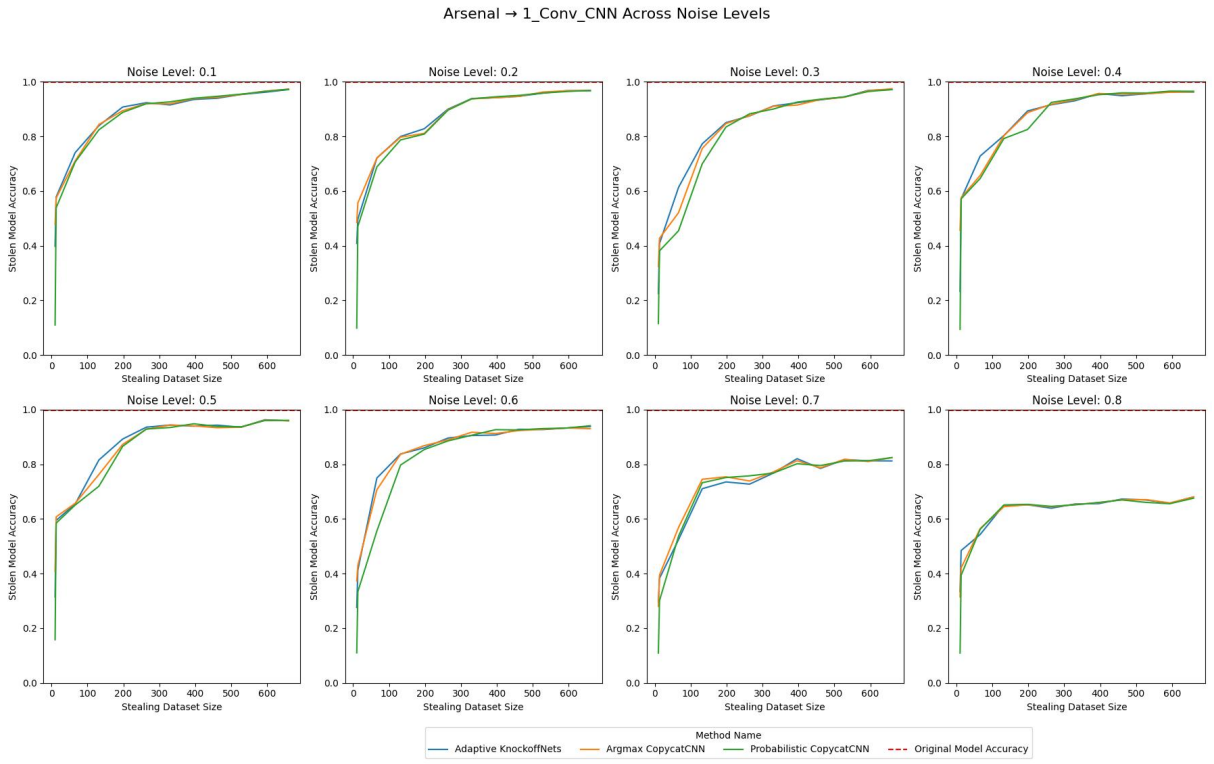

Figure 4.6: Influence of noise on extracion attacks using SpokenArabicDigits

4.4 Attacks using generated Data

As stated in Chapter 3.4 a VAE was trained on a portion of the original training. In this experiment these different VAEs were used to create a portion of the stealing dataset. Each VAE was used to create its training size portion of the stealing dataset. So the VAE trained on 10% of the training data per class, created 10% of the stealing dataset and so on. The generated samples were then mixed with original samples. Figure 4.7 shows the results of this experiment using BaseCNN1D as original model and SimpleCNN1D as substitute model on the SpokenArabicDigits dataset. Figure 15 and 14 show results on the other dataset. For results using the other model combinations and datasets see the associated GitHub repository [25].

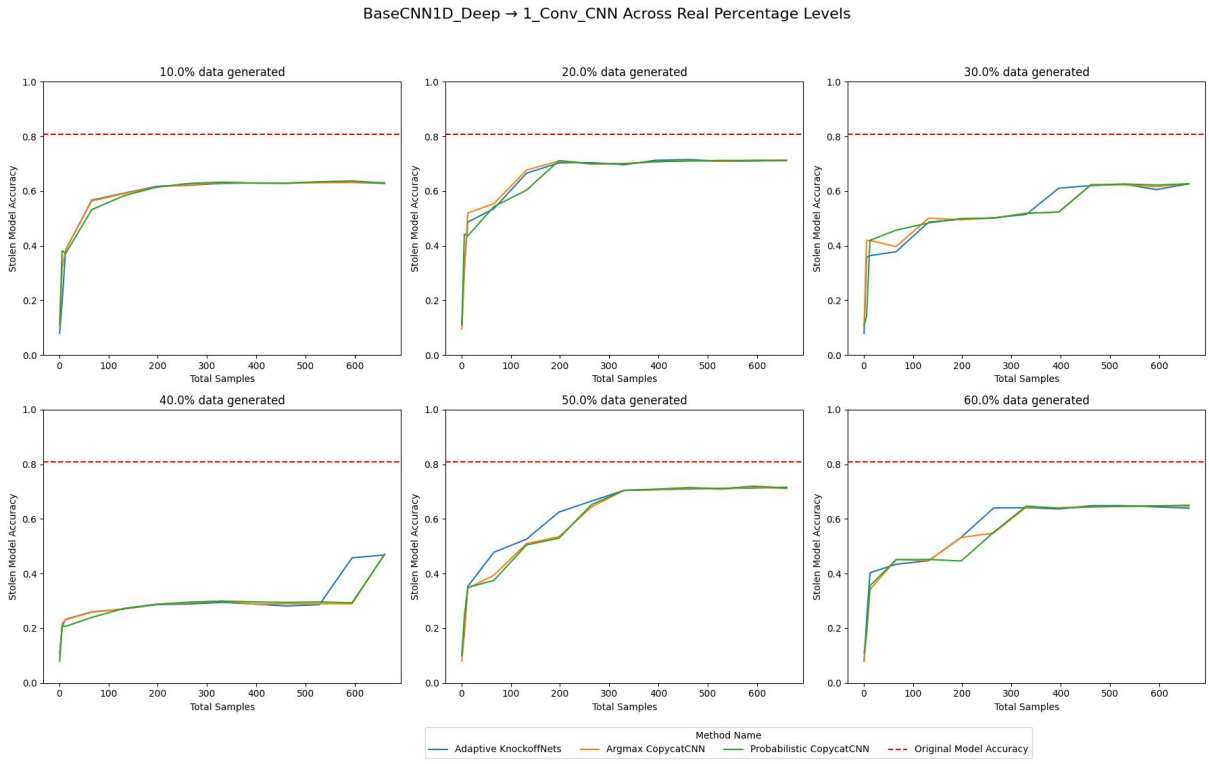

Figure 4.7: Extraction attack using generated and original data on SpokenArabicDigits

Figure 4.7 illustrates the results of the extraction attacks using a combination of generated and original data on the SpokenArabicDigits dataset. The substitute models did not reach the performance of the original model; however, this is expected, as even the basic attack for this model combination did not achieve full extraction success (see Figure 4.1). Given this baseline, it can be concluded that incorporating generated data into the stealing dataset does not negatively impact the effectiveness of the extraction attack. While some model combinations show a decrease in attack performance when using generated data, there is no apparent trend indicating that increasing the synthetic portion of the stealing dataset consistently worsens the extraction attack. This suggests that the effectiveness of the attack remains largely stable, regardless of the proportion of generated data used.

Building on this initial experiment, a second experiment was conducted in which only synthetic data was used for the extraction attacks. Instead of mixing generated and original samples, the stealing dataset was entirely composed of data generated by the different VAEs. This approach aimed to assess whether the quality of the generated samples influences the success of the attacks. By using VAEs trained on different portions of the original training data, we could analyze how the amount of available training data affected the quality of the generated samples and, consequently, the performance of the substitute models.

To ensure comparability to the other experiments, the attacks were run with the same stealing dataset size as in previous experiments. At first one sample of each was generated then samples of random classes until the necessary dataset size was reached.

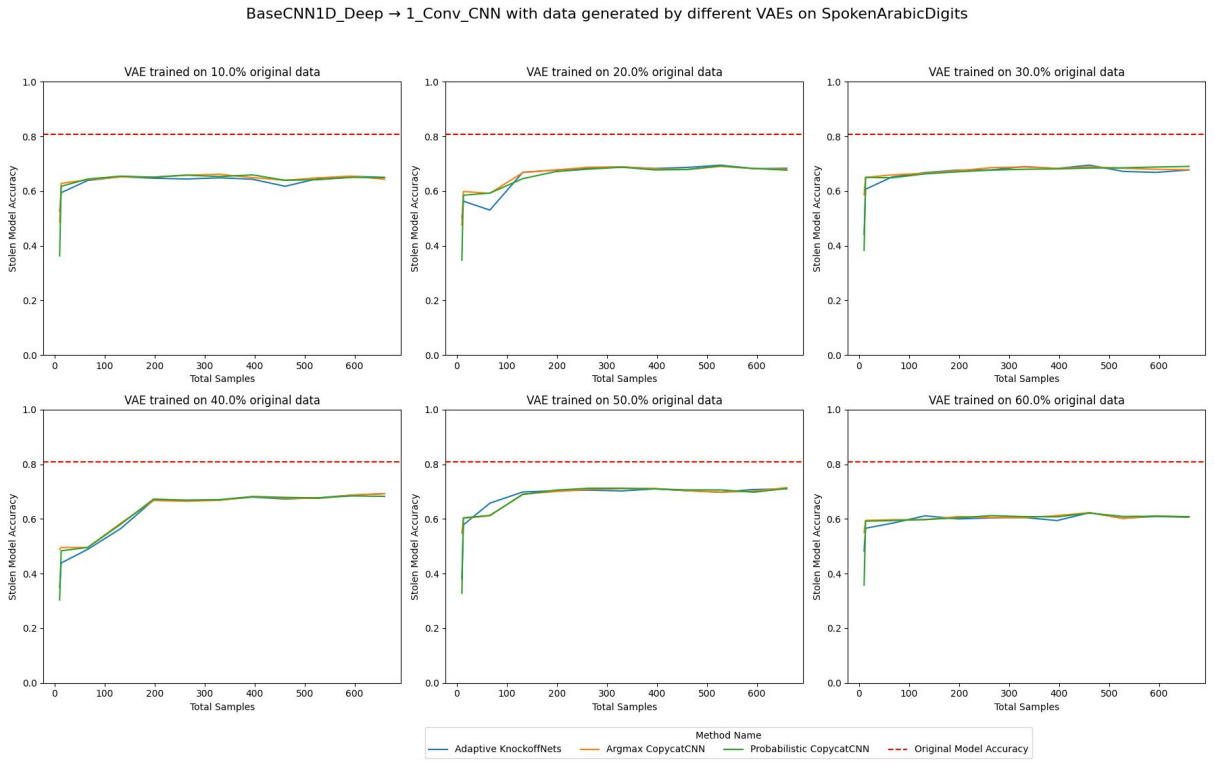

Figure 4.8: Extraction attacks using only generated data on SpokenArabicDigits

Figure 4.8 shows results of this attack approach using Deep1DCNN as original model and Simple1DCNN as substitute model. The accuracies of the substitute models show a dramatic increase even with a small stealing dataset size, but then saturate at around 20% under the original model accuracy this holds true for all model combinations on the SpokenArabicDigits dataset. Attacks on the PenDigits dataset show a similar progression (Figure 17), as the stealing dataset size increases, with the difference that attacks on ROCKET-based original models saturate at 40-50% under the original model accuracy, attacks on non-ROCKET-based originals models saturate at 20-40% under original model accuracy.

Attacks using the WalkingSittingStanding dataset (Figure 16) present an even greater challenge. Most attacks involving ROCKET-based original models were unsuccessful, with substitute models resorting to random guessing. Even in cases where attacks had some success, substitute model accuracies remained significantly lower than those of the original models.

These results highlight the critical role of dataset length and the number of channels in model extraction attacks. Datasets with shorter sequences and more channels, such as SpokenArabicDigits and PenDigits, enable more effective attacks, as substitute models can achieve reasonable accuracy even with limited stealing data. However, as seen with the WalkingSittingStanding dataset, longer sequences and fewer channels make attacks significantly more difficult, particularly against ROCKET-based original models. The failure of most attacks on ROCKET-based models further suggests that models leveraging temporal convolutions and feature transformations are more resilient to

extraction attempts, emphasizing the importance of both dataset characteristics and model architecture in determining attack success.

4.5 Using noisy and generated Data on a defended Model

This experiment extends previous analyses by evaluating attacks on a defended model using a dataset that combines original, slightly noised (noise level: 0.1), and VAE-generated samples in equal proportions. Given the insights from earlier experiments, the results align with expectations, further reinforcing the ineffectiveness of defenses on model extraction attempts.

Figures 4.9 and 4.10 illustrate the effects of preprocessing and postprocessing defenses, respectively, when faced with attacks leveraging a mix of real, noisy, and synthetic data using the SpokenArabicDigits dataset. Despite the inclusion of synthetic and perturbed data, the defended models were still extracted with much success. For results on the other datasets and model combinations see the associated GitHub repository [25].

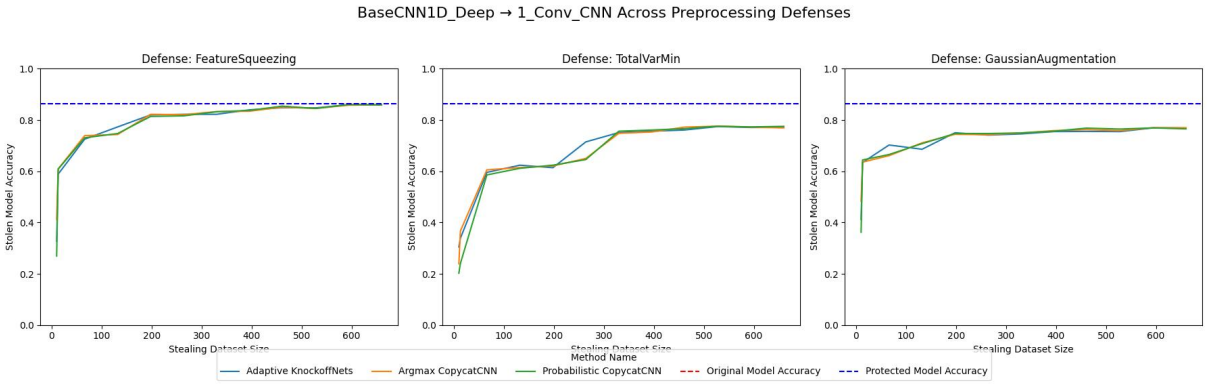

Figure 4.9: Attacks using original, noisy and generated data against a preprocessing defended model

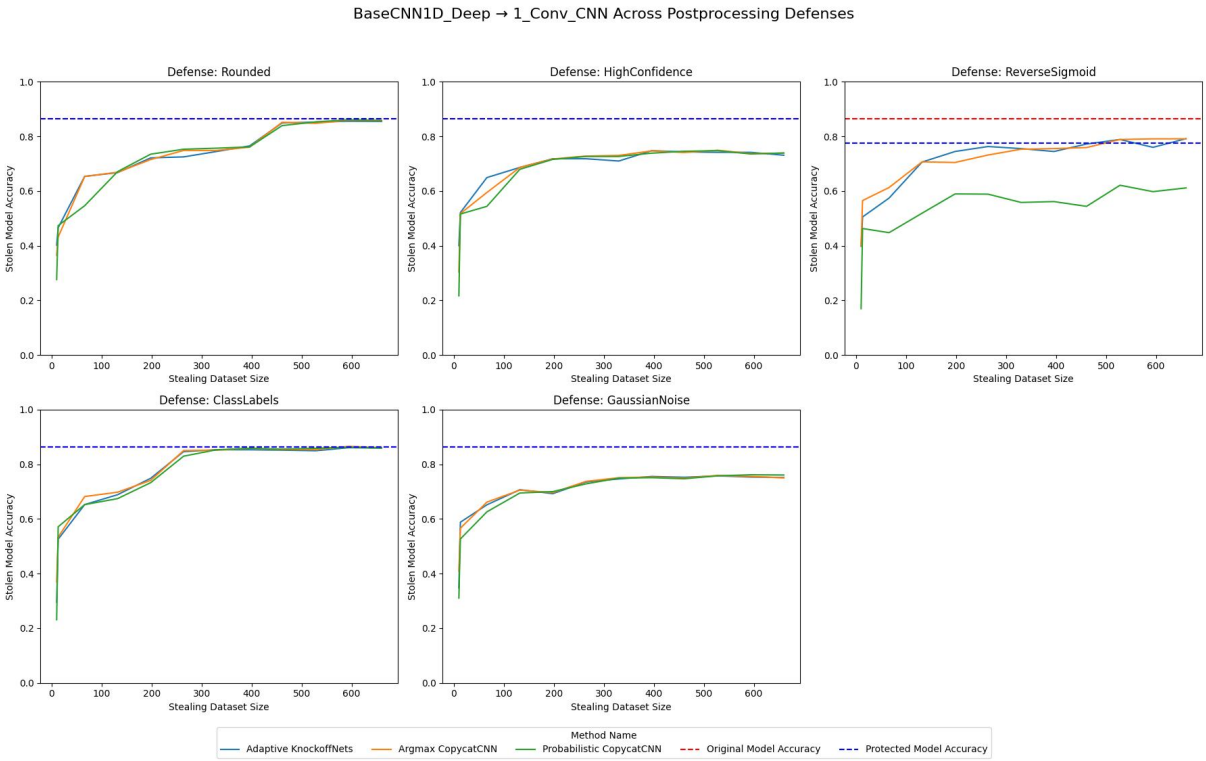

Figure 4.10: Attacks using original, noisy and generated data against a postprocessing defended model

Chapter 5

Results

This chapter presents the experimental findings of the conducted experiments in a more generalized way, analyzing the key factors influencing their success. We examine the overall performance of the attacks, the impact of stealing dataset size, the effects of noisy and generated attack data.

5.1 Performance of Model Extraction Attacks

The results demonstrate that model extraction attacks can successfully approximate the performance of the original model, though the effectiveness varies across datasets and model architectures. In general, simpler substitute models tend to achieve just as high of an accuracy as more complex architectures, regardless of the original model architecture (Figure 19). This suggests that, in principle, any type of original architecture can be extracted using any substitute architecture, challenging the assumption that architectural similarity between the original and substitute model is a strict requirement for successful extraction.

Interestingly, the experiments also reveal that CNN-based models can be effectively extracted using LSTM or RNN-based substitutes and vice versa. This further supports the idea that attackers are not necessarily constrained to using structurally similar models when conducting extraction attacks, expanding the potential threat landscape.

5.2 Impact of Steal Dataset Size

A key factor in model extraction success is the size of the stealing dataset. As expected, larger stealing datasets lead to better substitute model performance, as they provide more comprehensive coverage of the original model's decision boundary.

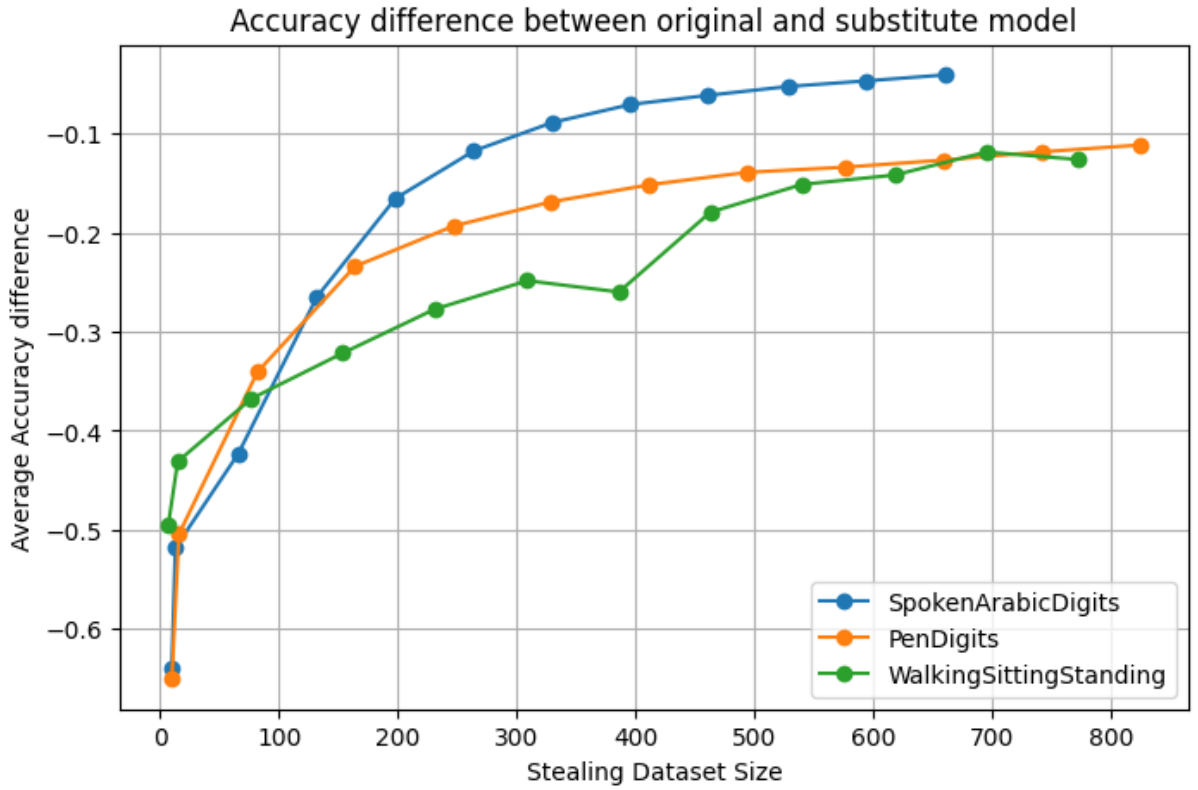

Figure 5.1: Accuracy difference between the original and substitute model

Figures 5.1 illustrates how substitute model accuracy improves as the stealing dataset size increases. However, diminishing returns are observed beyond a certain point, indicating that after a threshold, additional samples contribute little to further improving substitute model performance.

This observation suggests the importance of sequence length and the number of channels in the dataset. Attacks on the WalkingSittingStanding dataset, the dataset with the longest sequences, consistently perform worse than those on datasets with shorter sequences but more channels. This suggests that the complexity introduced by longer sequences may hinder the extraction process, while a larger number of channels provides richer information to the substitute model.

5.3 Influence of Noisy & Generated Attack Data

The dataset with the longest sequences, WalkingSittingStanding, is the most affected by noise, with substitute model accuracy dropping considerable as noise levels increases. In contrast, the dataset with the highest number of channels is the least affected by noise, suggesting that richer feature representations help mitigate the impact of noise in the stealing dataset. Interestingly, PenDigits, which may appear to be the simplest dataset at first glance, falls between these two extremes in terms of its susceptibility to noise, indicating that sequence length and feature dimensionality both play crucial roles in determining the robustness of model extraction attacks against noisy data.

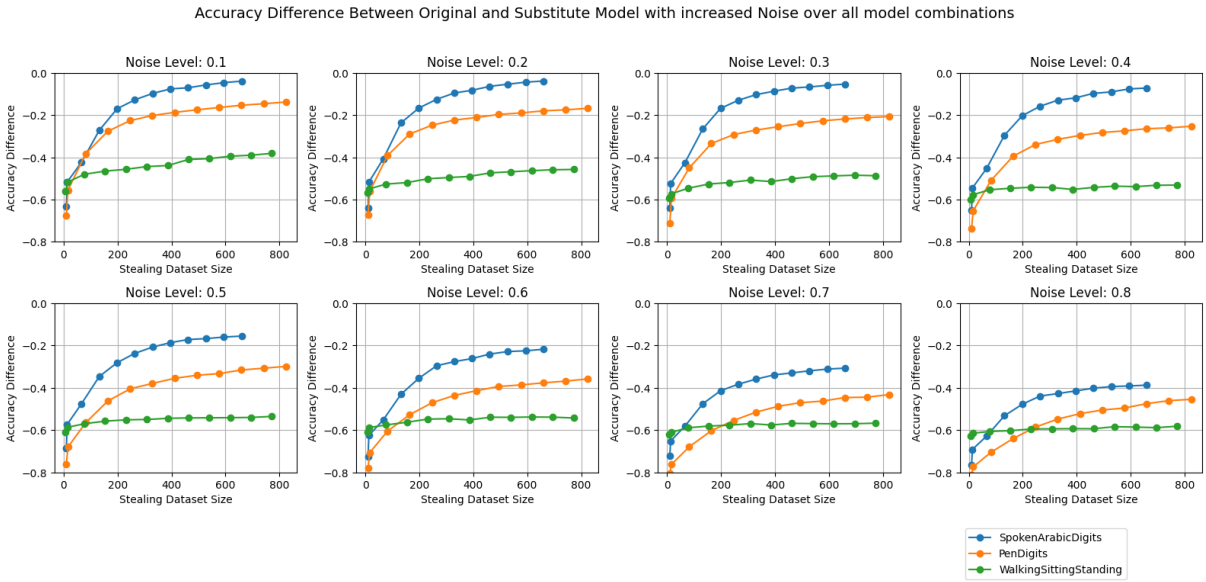

Figure 5.2: Accuracy Difference Between Original and Substitute Model with increased noise

This trend is clearly demonstrated in Figure 5.2, where the WalkingSittingStanding dataset experiences the steepest decline in substitute model accuracy as noise increases, while the dataset with the most channels only degrades with high noise levels. PenDigits, positioned between these two extremes, exhibits a moderate decline, reinforcing the notion that both sequence length and feature dimensionality influence the effectiveness of noise as a defense against model extraction attacks.

Experiments with generated data show that incorporating up to 40% synthetic data has minimal impact on the accuracy of substitute models. For SpokenArabicDigits, no decline in accuracy was observed, reinforcing the conclusions drawn from the noise experiment (see Figure 5.3). This suggests that even when adversaries have limited access to real samples, they can still create effective substitute datasets using generative models without significantly compromising attack performance.

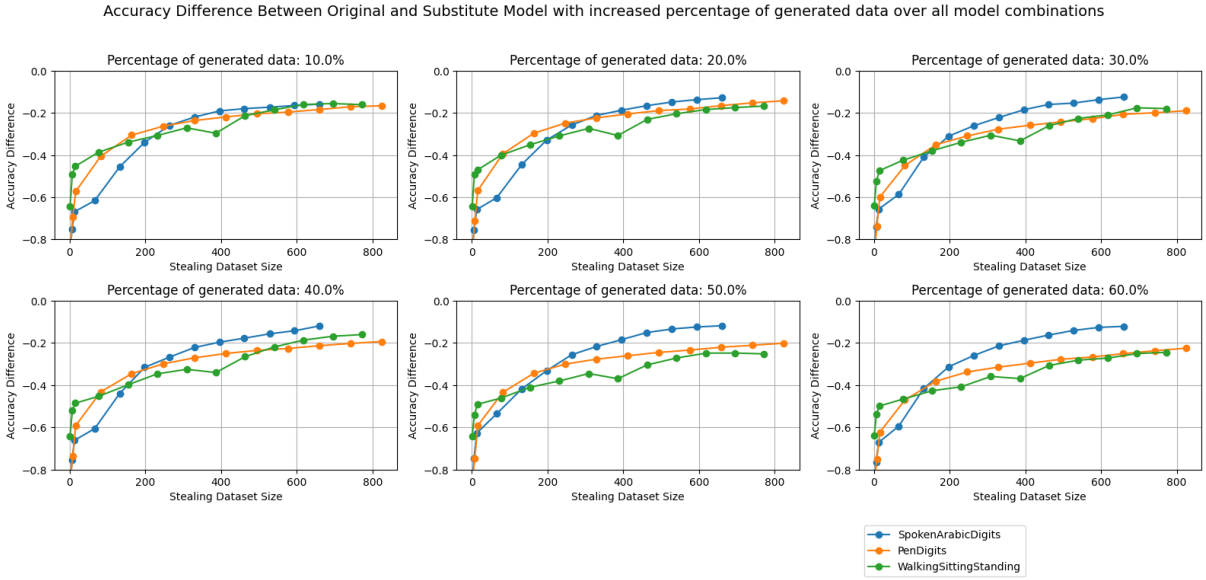

Figure 5.3: Accuracy Difference Between Original and Substitute Model with increased generated data

A follow-up experiment using only synthetic data further examined the role of data quality in attack success. Different VAEs trained on varying portions of the original dataset were used to generate samples, enabling an analysis of how training data availability affects the effectiveness of generated attack data. Figure 5.4 shows that increasing the VAE training dataset size barely improves substitute model accuracy. Specifically, attacks on WalkingSittingStanding lag 40% behind the original model's accuracy, while on PenDigits and SpokenArabicDigits, the gaps are 30% and 20%, respectively. This further reinforces the importance of incorporating more channels and shorter sequences for a successful attack.

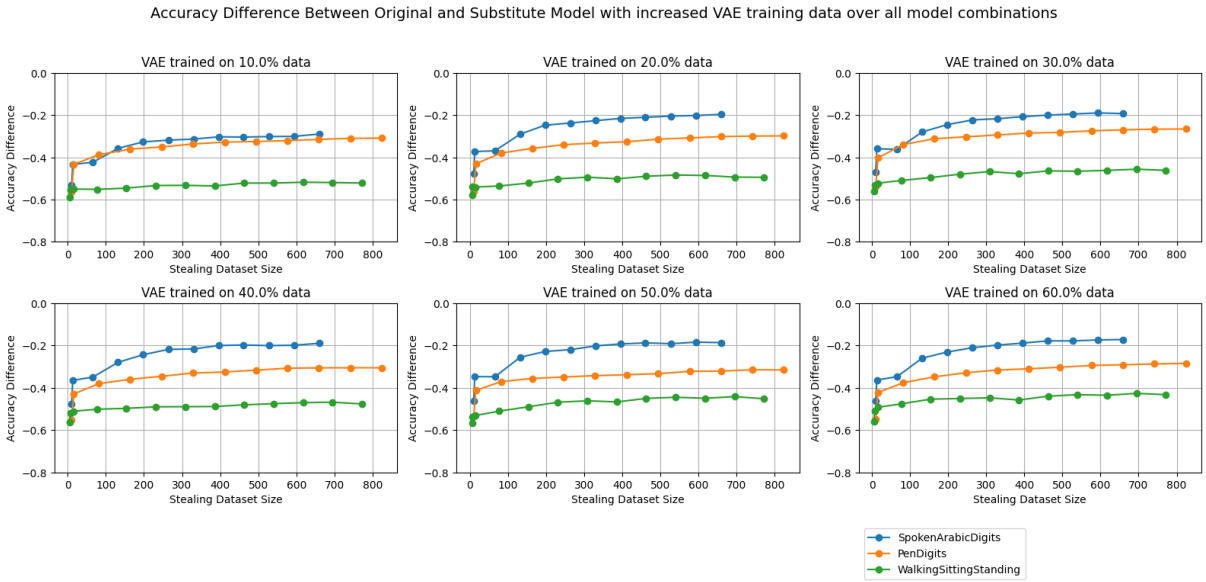

Figure 5.4: Accuracy Difference Between Original and Substitute Model with only generated data

Chapter 6

Discussion

This chapter interprets the findings of the model extraction attack experiments, placing them in the broader context of adversarial machine learning. We discuss the implications of the observed trends, compare results to prior work, and explore potential countermeasures.

6.1 Implications of Findings

Substitute Model Complexity and Attack Effectiveness

One of the key observations from the experiments is that substitute model complexity has diminishing returns on substitute model accuracy. While increasing the number of parameters improves accuracy to some extent, simpler models often perform nearly as well as more complex ones. This suggests that attackers do not need highly sophisticated architectures to achieve successful extraction, in some cases the simplest substitute model showed the highest accuracy.

Additionally, the ability to extract CNN-based models using LSTM or RNN substitutes (and vice versa) implies that attackers can use any architecture of their choosing to extract their target model.

Effect of Data Availability on Extraction Success

The size and quality of the stealing dataset play a crucial role in determining the effectiveness of an attack. A larger dataset leads to a better substitute model, but with diminishing returns. Moreover, datasets with more feature channels tend to be less affected by noise compared to those with longer sequences.

Experiments with generated attack data highlight that attackers can achieve high accuracy even when real query samples are limited. Even fully synthetic datasets show good results, when combining synthetic and real data even more potent can be extracted. This finding reinforces concerns that generative models can be leveraged to enhance extraction attacks, reducing the need for extensive access to real queries.

Trade-Offs Between Accuracy and Efficiency

Attack duration varies significantly depending on the combination of original and substitute model architectures, as well as the chosen attack method and stealing dataset size. In general, attacks on the larger ROCKET-based models, like Arsenal, Rocket, MultiRocket and MultiRocketHydra, require more time. This is to be expected, as these architectures also showed a longer inference time (see Table 3.2). Figure 18 shows this. Using the SpokenArabicDigits dataset, attacks on MultiRocket and MultiRocketHydra take about 5 seconds, while attacks on Rocket and Arsenal take on average 15 and 19 seconds respectively.

6.2 Comparison to Prior Work

These findings align with prior studies on model extraction [19] and [28]. However, previous research often assumes that architectural similarity between the original and substitute model is necessary for good extractions. These results challenge this assumption, demonstrating that diverse architectures can achieve similar performance levels.

Furthermore, this analysis highlights how sequence length and feature dimensionality influence the effectiveness of attacks. As noted by [19, 28, 33], the findings from Chapters 4.2.2 and 4.2.1 demonstrate that existing defense mechanisms remain largely ineffective against model extraction attacks.

Chapter 7

Conclusion & Future Work

This thesis provides a detailed examination of model extraction attacks on time series models, highlighting the key factors influencing their success. The results demonstrate that similar architectural complexity between target and adversary model is not a strict requirement for effective extraction, as even the simplest architectures can achieve high accuracy, challenging current assumptions.

The experiments also emphasize the importance of data availability in extraction success. While larger stealing datasets improve performance, diminishing returns suggest that attackers do not necessarily need extensive query access. Additionally, the ability to use generated data to enhance attacks raises concerns about the role of generative models in facilitating extraction.

While this study focuses only on classification tasks, prior research suggests that regression models may be similarly vulnerable. Future work should investigate the susceptibility of regression-based time series models to extraction attacks and explore effective defense mechanisms. A key challenge remains in designing countermeasures that mitigate extraction risks without significantly degrading the original model's accuracy. Additionally, developing methods to detect ongoing extraction attempts in real-world deployments should be a priority for future research.j

Bibliography

[1] E. Alpaydin and Fevzi. Alimoglu. *Pen-Based Recognition of Handwritten Digits*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5MG6K. 1996.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. *Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples*. 2018. arXiv: 1802.00420 [cs.LG]. URL: https://arxiv.org/abs/1802.00420.

[3] Mouldi Bedda and Nacereddine Hammami. *Spoken Arabic Digit*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C52C9Q. 2008.

[4] Jacson Rodrigues Correia-Silva et al. "Copycat CNN: Stealing Knowledge by Persuading Confession with Random Non-Labeled Data". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2018, pp. 1–8. DOI: 10.1109/ijcnn.2018.8489592. URL: http://dx.doi.org/10.1109/IJCNN.2018.8489592.

[5] Angus Dempster, François Petitjean, and Geoffrey I. Webb. "ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels". In: *Data Mining and Knowledge Discovery* 34.5 (July 2020), pp. 1454–1495. ISSN: 1573-756X. DOI: 10.1007/s10618-020-00701-z. URL: http://dx.doi.org/10.1007/s10618-020-00701-z.

[6] Angus Dempster, Daniel F. Schmidt, and Geoffrey I. Webb. *HYDRA: Competing convolutional kernels for fast and accurate time series classification*. 2022. arXiv: 2203.13652 [cs.LG]. URL: https://arxiv.org/abs/2203.13652.

[7] Angus Dempster, Daniel F. Schmidt, and Geoffrey I. Webb. "MiniRocket: A Very Fast (Almost) Deterministic Transform for Time Series Classification". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery amp; Data Mining*. KDD '21. ACM, Aug. 2021, pp. 248–257. DOI: 10.1145/3447548.3467231. URL: http://dx.doi.org/10.1145/3447548.3467231.

[8] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[9] *GitHub - optuna/optuna: A hyperparameter optimization framework — github.com*. https://github.com/optuna/optuna. [Accessed 21-03-2025].

[10] *GitHub - Trusted-AI/adversarial-robustness-toolbox: Adversarial Robustness Toolbox (ART) - Python Library for Machine Learning Security - Evasion, Poisoning, Extraction, Inference - Red and Blue Teams — github.com*. https://github.com/Trusted-AI/adversarial-robustness-toolbox. [Accessed 18-03-2025].

[11] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *International Conference on Artificial Intelligence and Statistics*. 2010. URL: https://api.semanticscholar.org/CorpusID:5575601.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[13] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML]. URL: https://arxiv.org/abs/1406.2661.

[14] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: https://arxiv.org/abs/2006.11239.

[16] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/abs/1312.6114.

[17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[18] Taesung Lee et al. *Defending Against Machine Learning Model Stealing Attacks Using Deceptive Perturbations*. 2018. arXiv: 1806.00054 [cs.LG]. URL: https://arxiv.org/abs/1806.00054.

[19] Jiacheng Liang et al. *Model Extraction Attacks Revisited*. 2023. arXiv: 2312.05386 [cs.LG]. URL: https://arxiv.org/abs/2312.05386.

[20] Matthew Middlehurst et al. "aeon: a Python Toolkit for Learning from Time Series". In: *Journal of Machine Learning Research* 25.289 (2024), pp. 1–10. URL: http://jmlr.org/papers/v25/23-1444.html.

[21] Matthew Middlehurst et al. "HIVE-COTE 2.0: a new meta ensemble for time series classification". In: *Machine Learning* 110.11–12 (Sept. 2021), pp. 3211–3243. ISSN: 1573-0565. DOI: 10.1007/s10994-021-06057-9. URL: http://dx.doi.org/10.1007/s10994-021-06057-9.

[22] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. *Knockoff Nets: Stealing Functionality of Black-Box Models*. 2018. arXiv: 1812.02766 [cs.CV]. URL: https://arxiv.org/abs/1812.02766.

[23] Jorge Reyes-Ortiz et al. *Human Activity Recognition Using Smartphones*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C54S4K. 2013.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV]. URL: https://arxiv.org/abs/1505.04597.

[25] *Schmiddi3110/Master-Thesis — github.com*. https://github.com/Schmiddi3110/Master-Thesis. [Accessed 30-03-2025].

[26] Yash Sharma and Pin-Yu Chen. *Bypassing Feature Squeezing by Increasing Adversary Strength*. 2018. arXiv: 1803.09868 [stat.ML]. URL: https://arxiv.org/abs/1803.09868.

[27] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV]. URL: https://arxiv.org/abs/1409.1556.

[28] Tatsuya Takemura, Naoto Yanai, and Toru Fujiwara. *Model Extraction Attacks against Recurrent Neural Networks*. 2020. arXiv: 2002.00123 [cs.CR]. URL: https://arxiv.org/abs/2002.00123.

[29] Chang Wei Tan et al. *MultiRocket: Multiple pooling operators and transformations for fast and effective time series classification*. 2022. arXiv: 2102.00457 [cs.LG]. URL: https://arxiv.org/abs/2102.00457.

[30] Florian Tramèr et al. *Stealing Machine Learning Models via Prediction APIs*. 2016. arXiv: 1609.02943 [cs.CR]. URL: https://arxiv.org/abs/1609.02943.

[31] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[32] Saverio Vito. *Air Quality*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5
2008.

[33] Kaixiang Zhao et al. *A Survey of Model Extraction Attacks and Defenses in Distributed Computing Environments*. 2025. arXiv: 2502.16065 [cs.CR]. URL: https://arxiv.org/abs/2502.16065.

List of Figures

List of Tables

Appendix A

Samples of each Dataset

Figure 1: Samples of WalkingSittingStanding

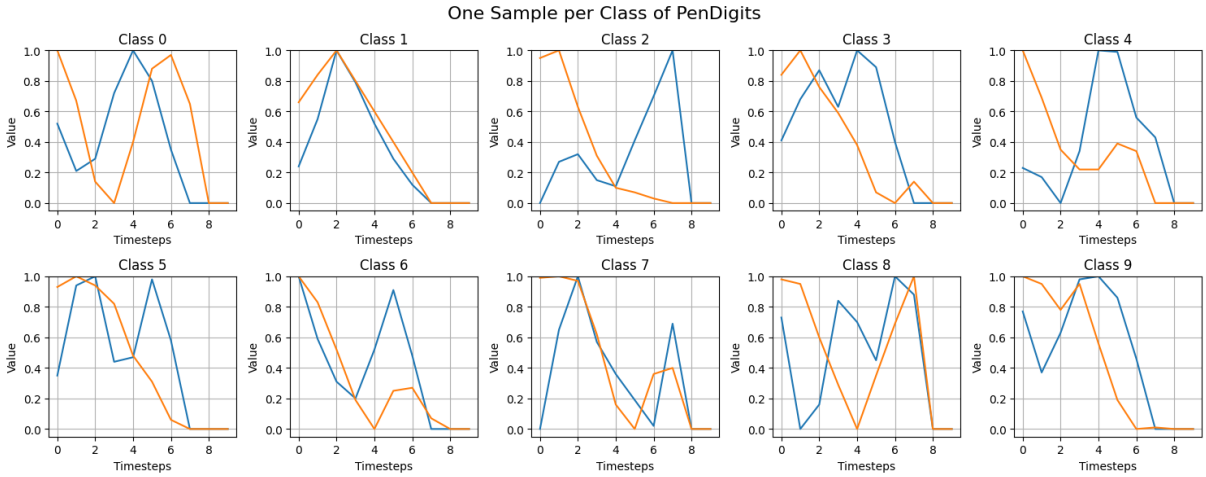

Figure 2: Samples of PenDigits

Figure 3: Samples of SpokenArabicDigits

Generated samples of each Dataset

Figure 4: Generated samples of PenDigits

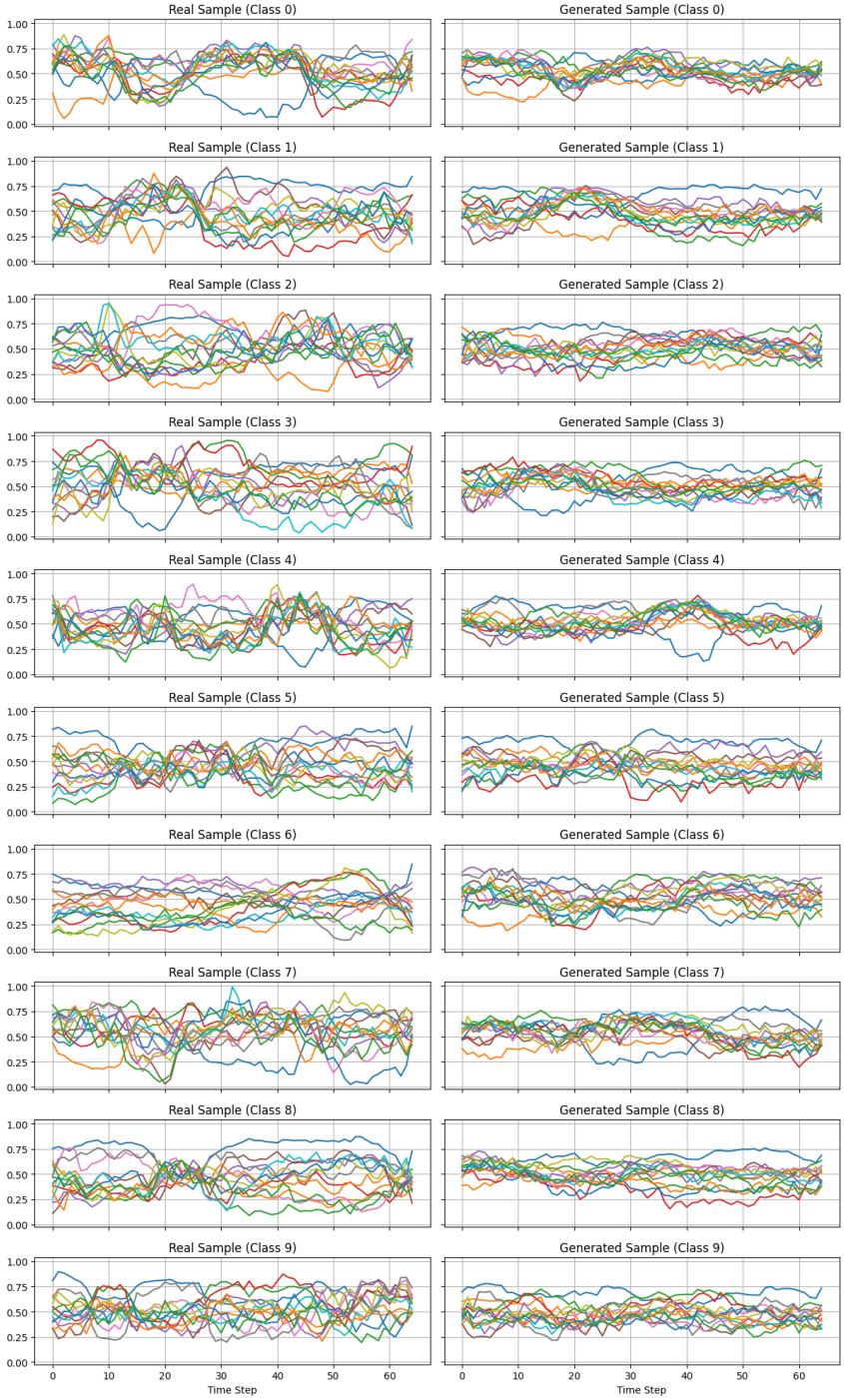

Figure 5: Generated samples of SpokenArabicDigits

Basic attack results

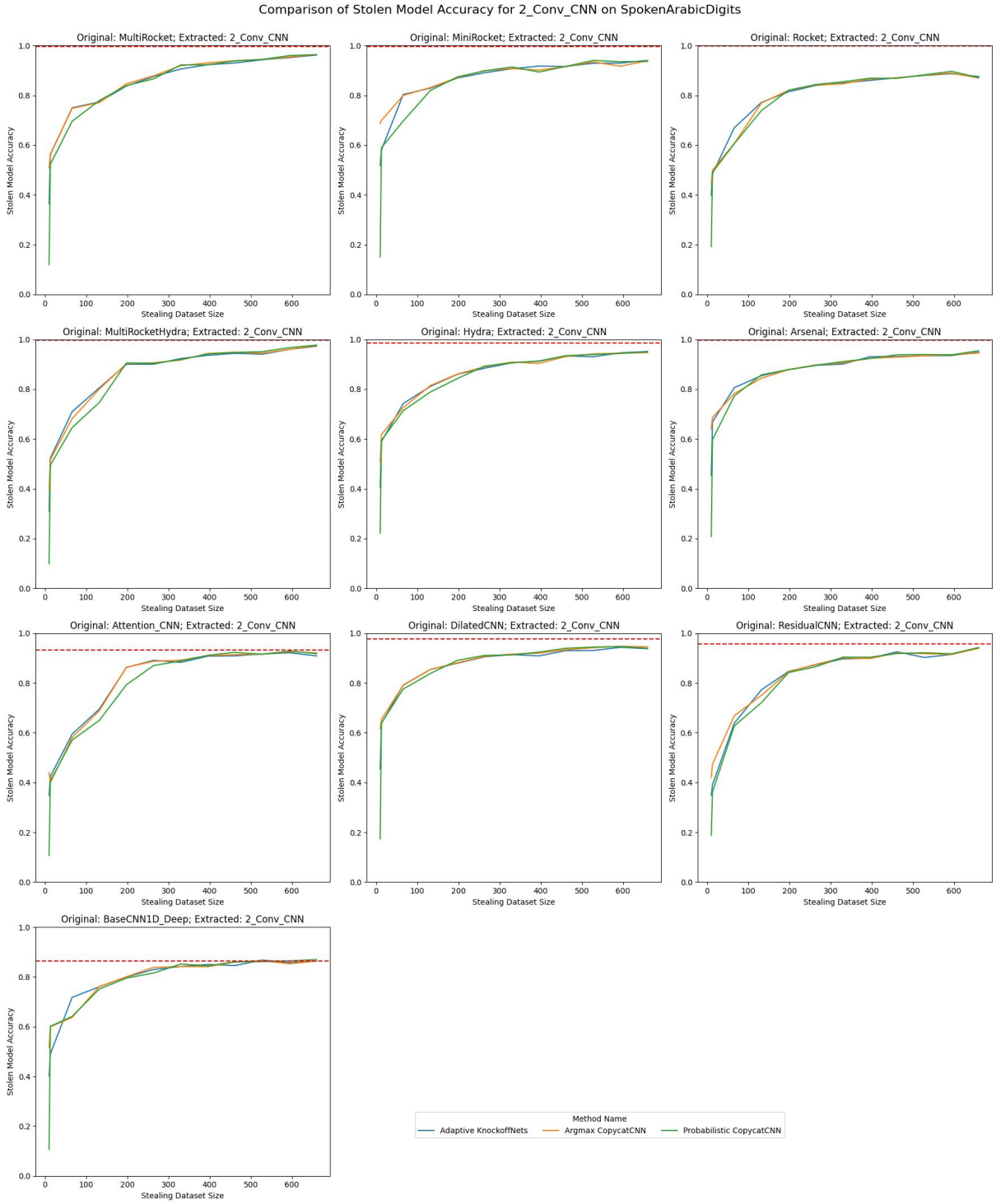

Figure 6: Results of basic extraction attack using the SpokenArabicDigits dataset

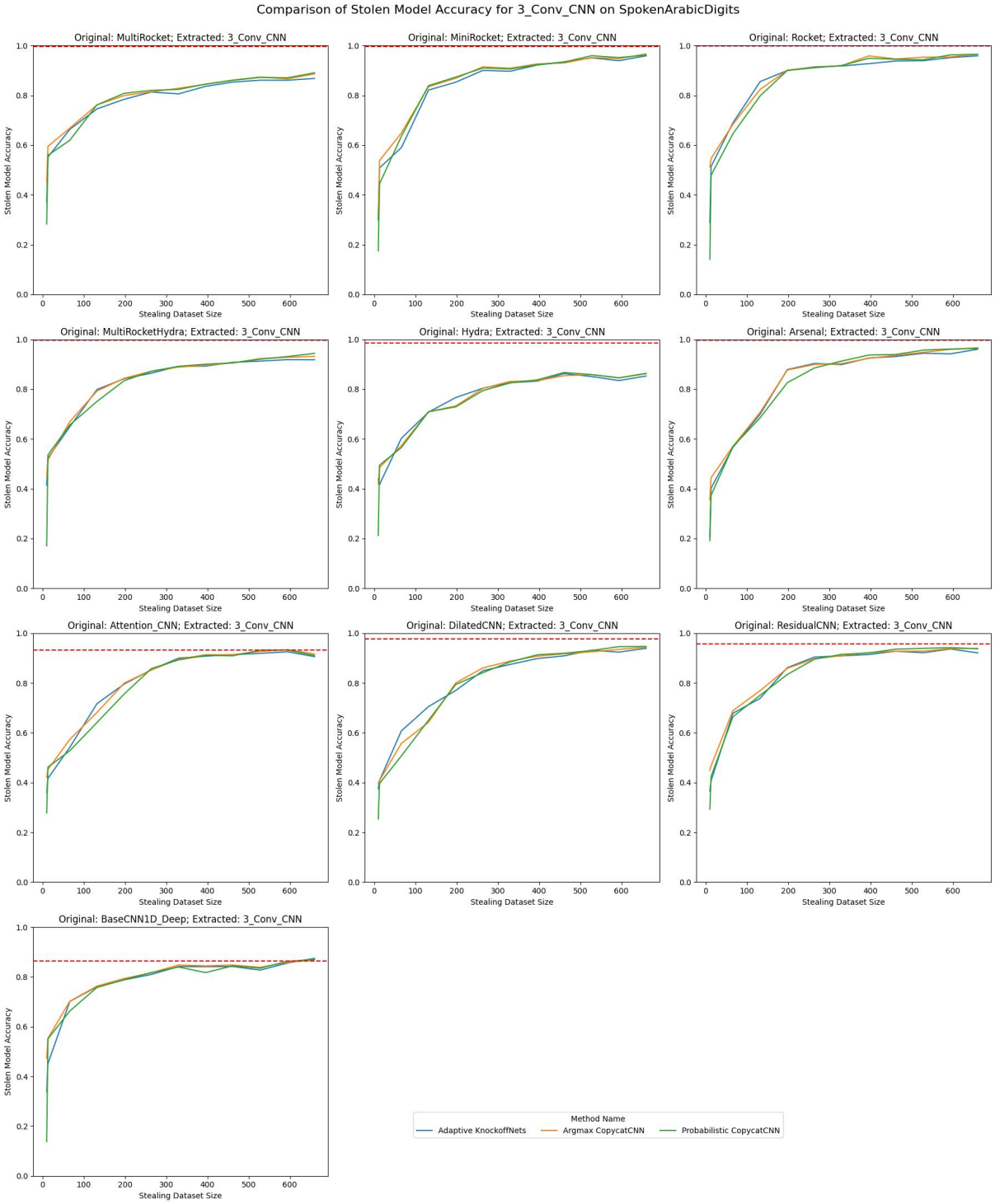

Figure 7: Results of basic extraction attack using the SpokenArabicDigits dataset

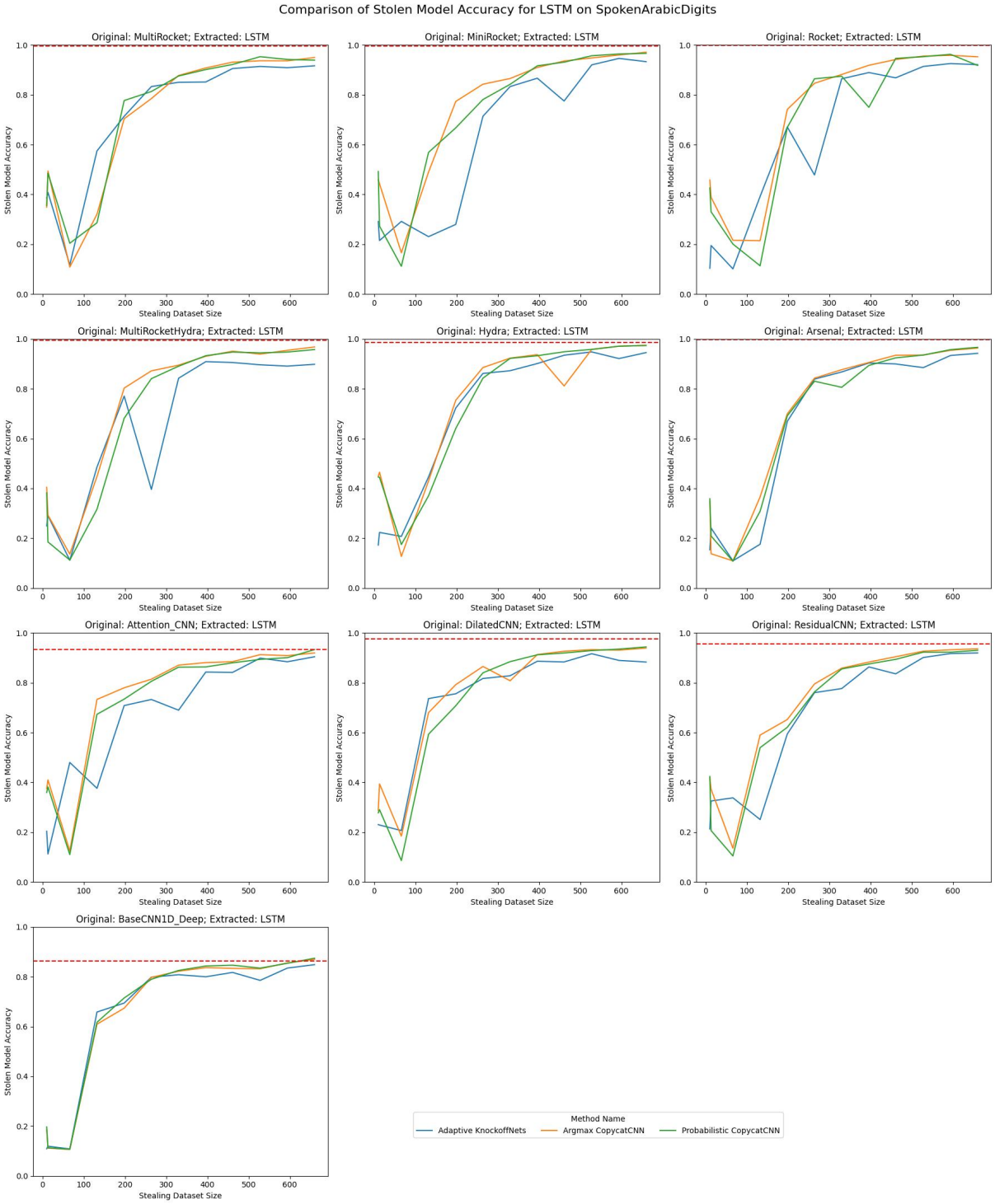

Figure 8: Results of basic extraction attack using the SpokenArabicDigits dataset

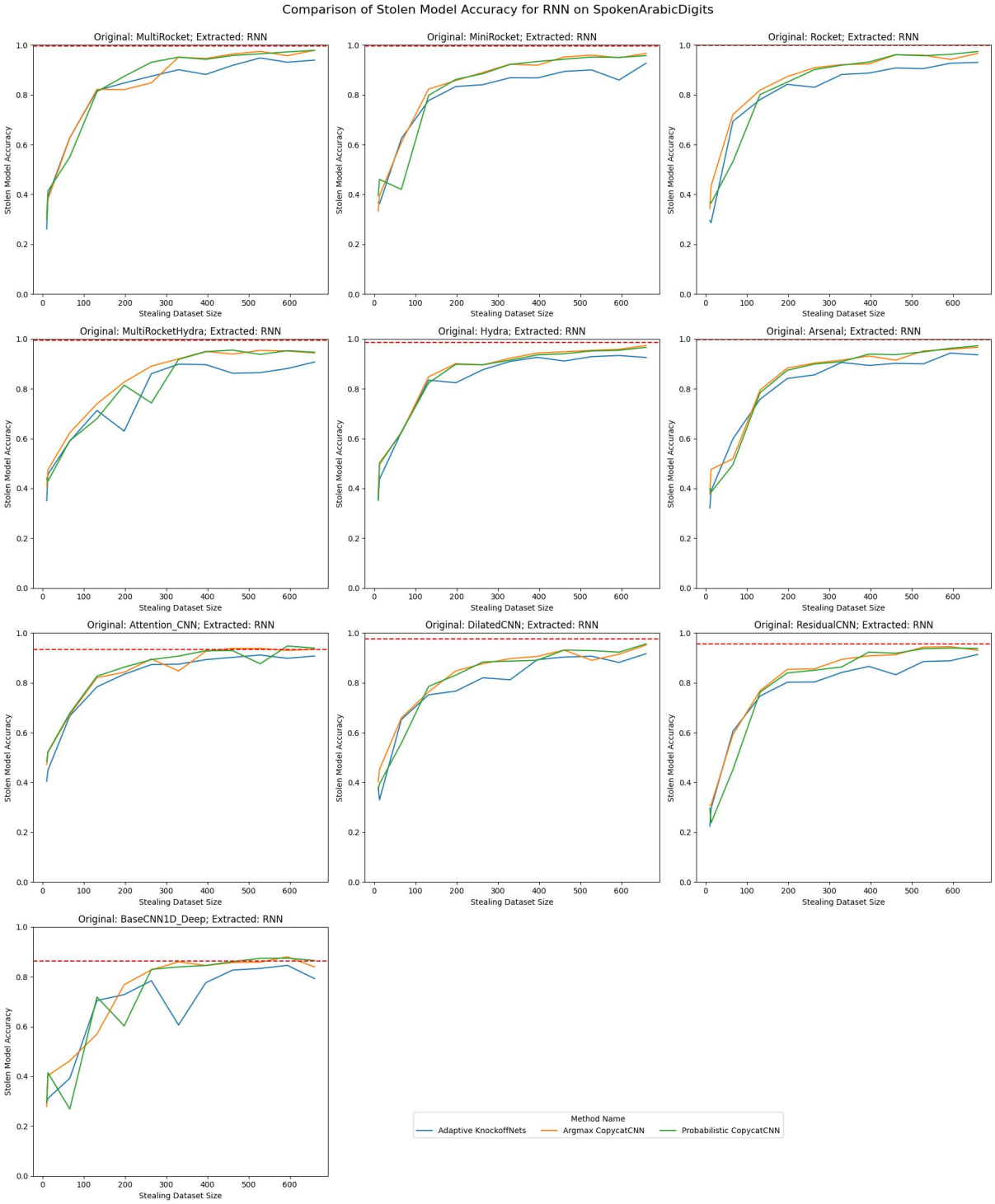

Figure 9: Results of basic extraction attack using the SpokenArabicDigits dataset

Defended Attacks

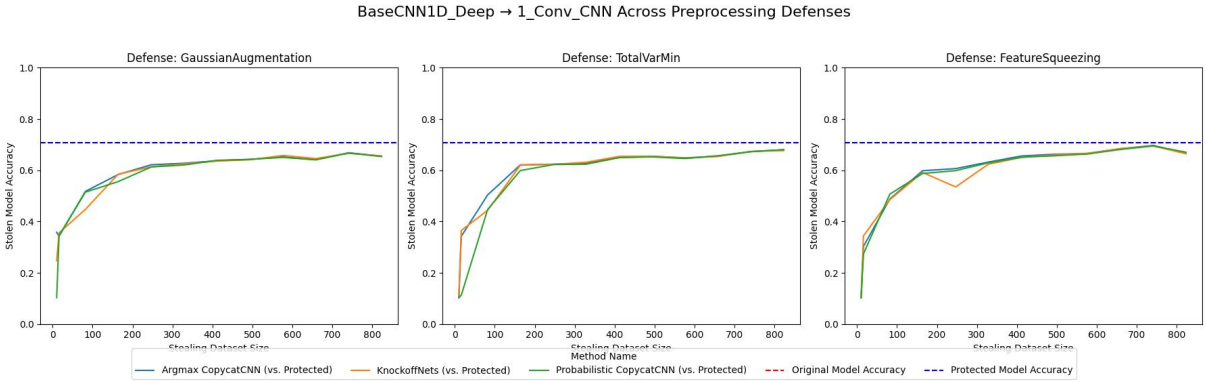

Figure 10: Preprocessing defended extraction attacks using PenDigits

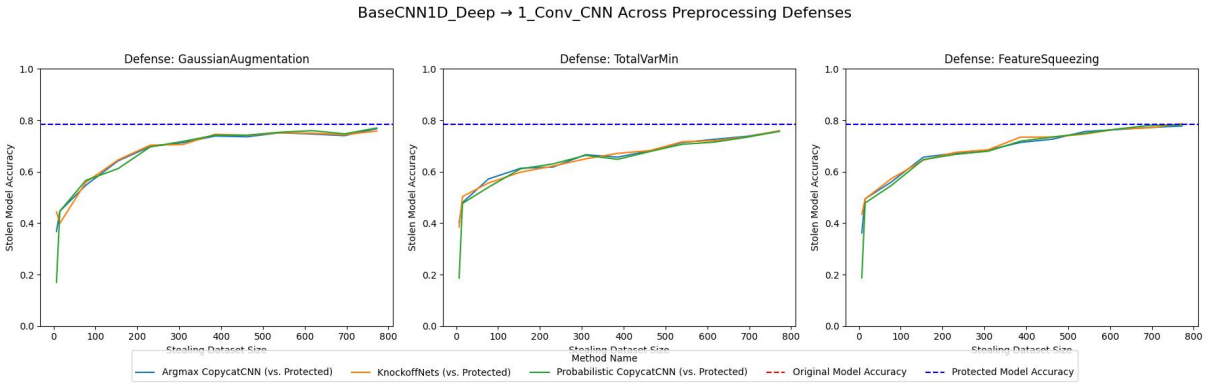

Figure 11: Preprocessing defended extraction attacks using WalkingSittingStanding

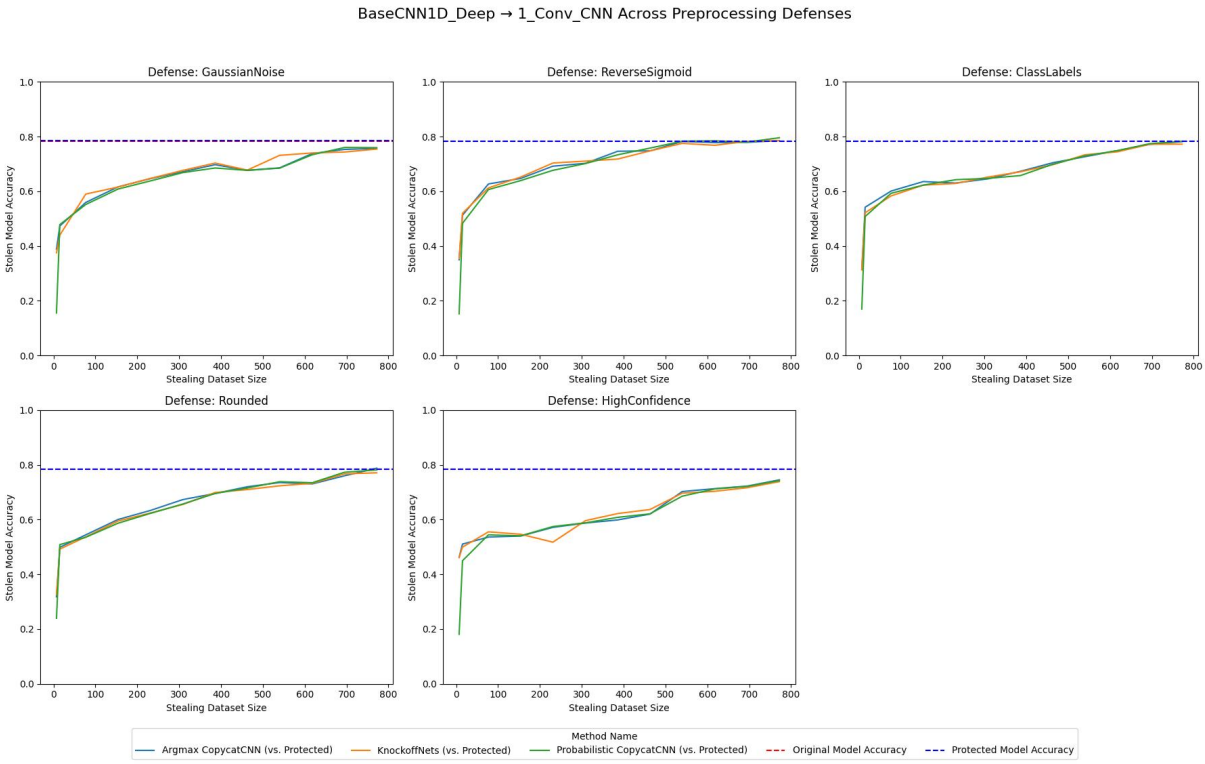

Figure 12: Postprocessing defended extraction attacks using WalkingSittingStanding

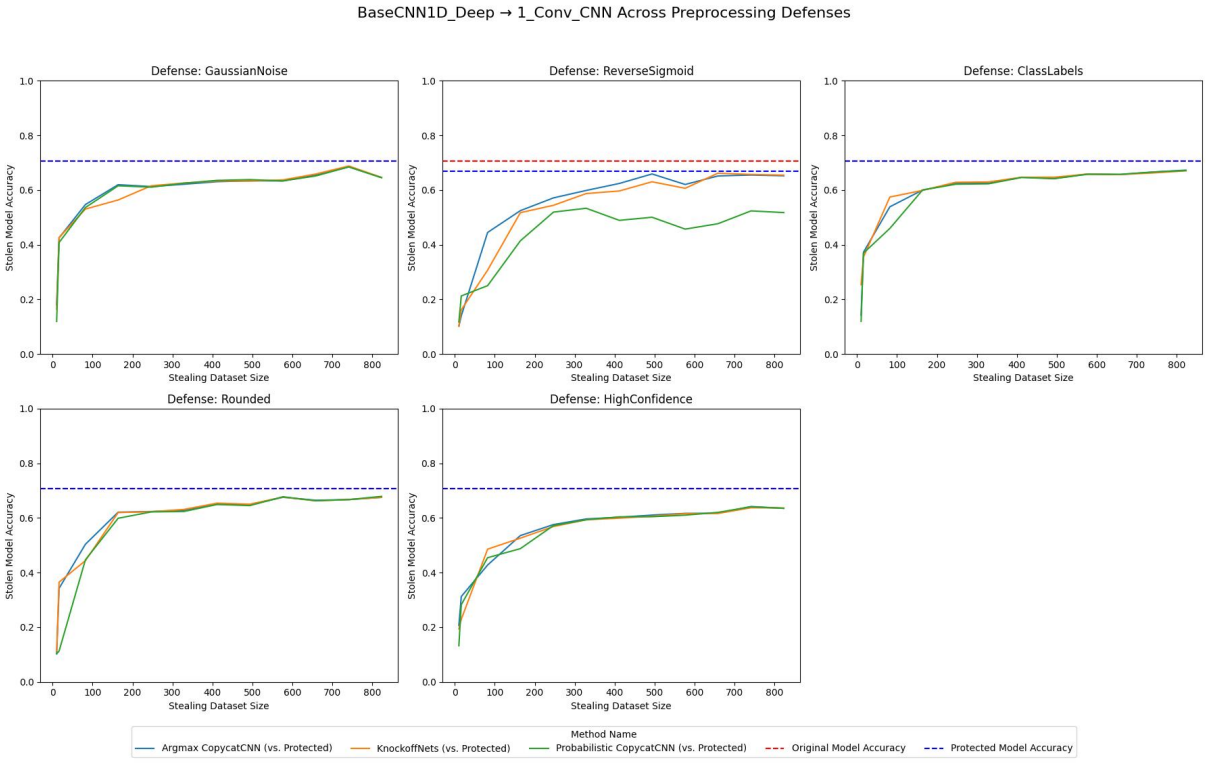

Figure 13: Postprocessing defended extraction attacks using PenDigits

Attacks with generated data

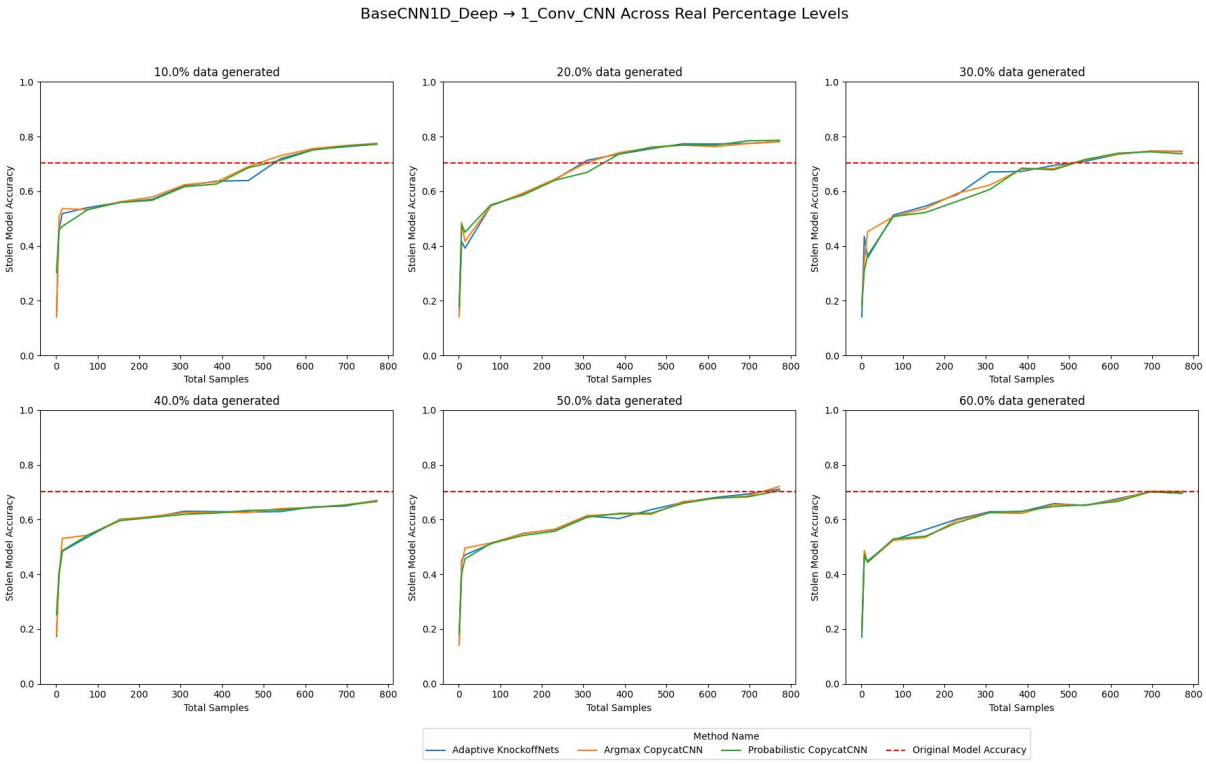

Figure 14: Extraction attacks using generated and original data on WalkingSittingStanding

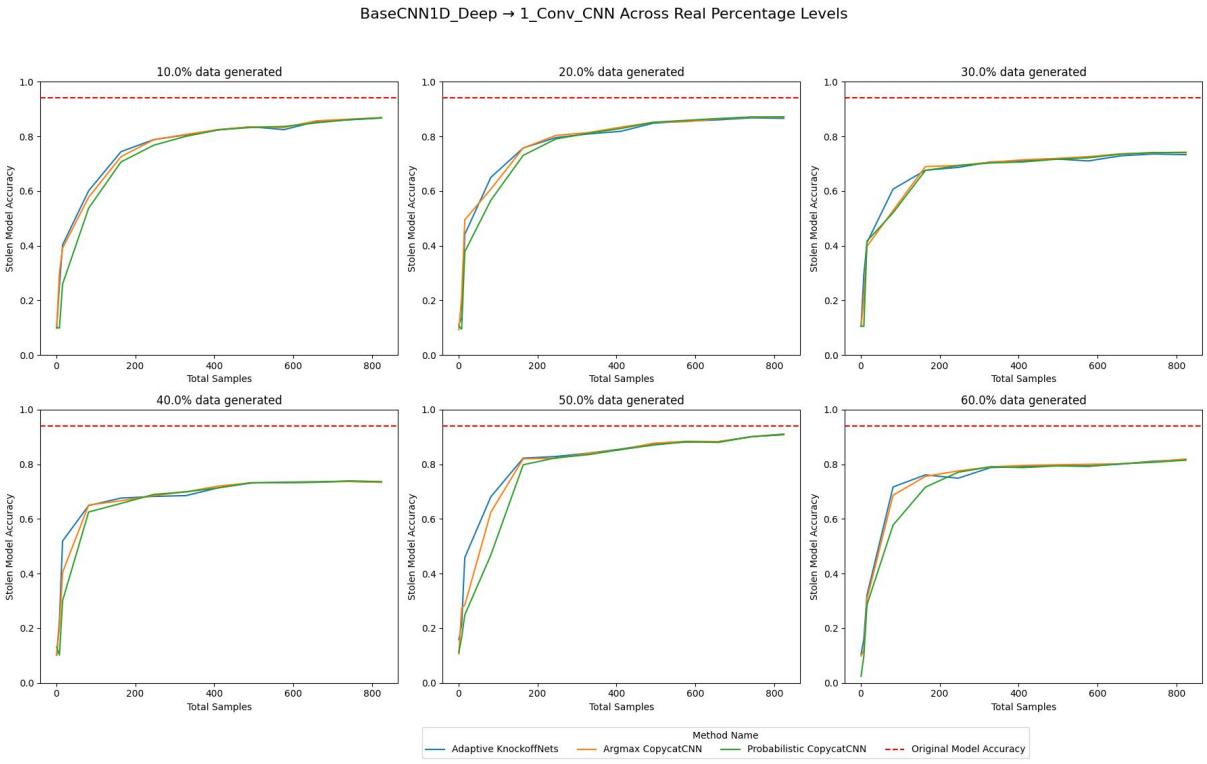

Figure 15: Extraction attacks using generated and original data on PenDigits

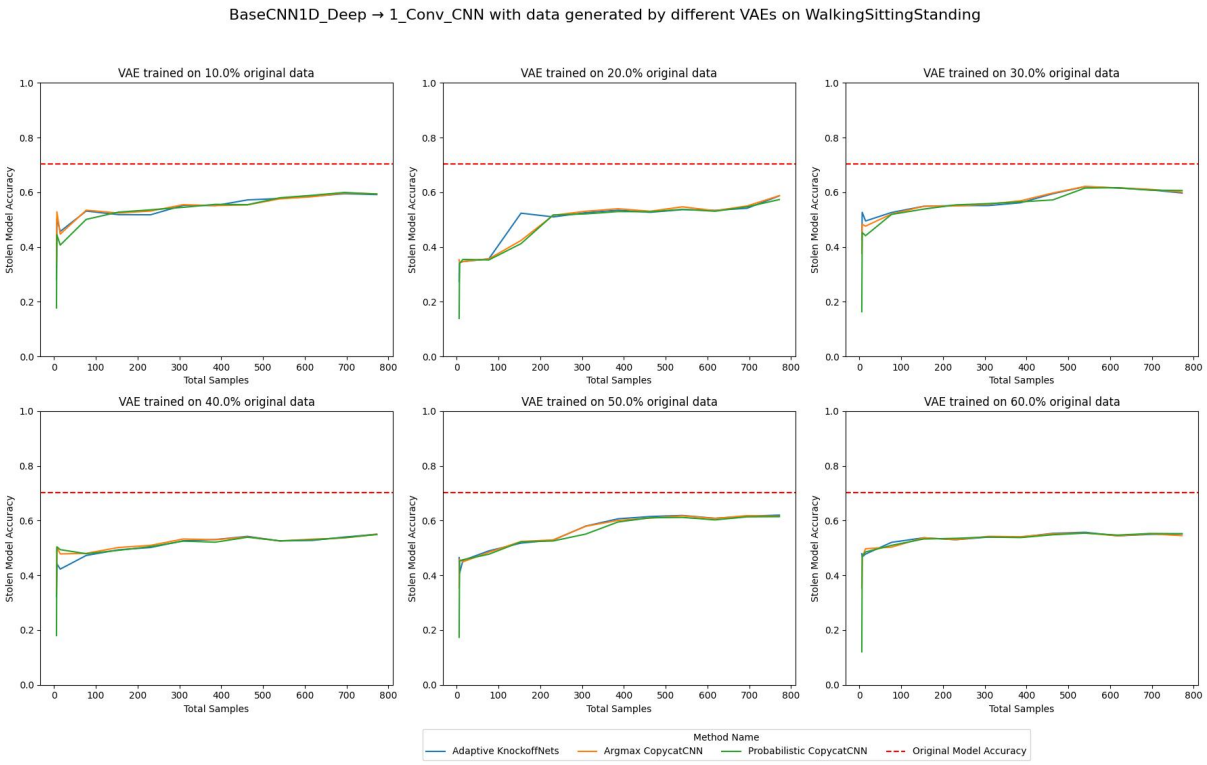

Figure 16: Extraction attacks using only generated data on WalkingSittingStanding

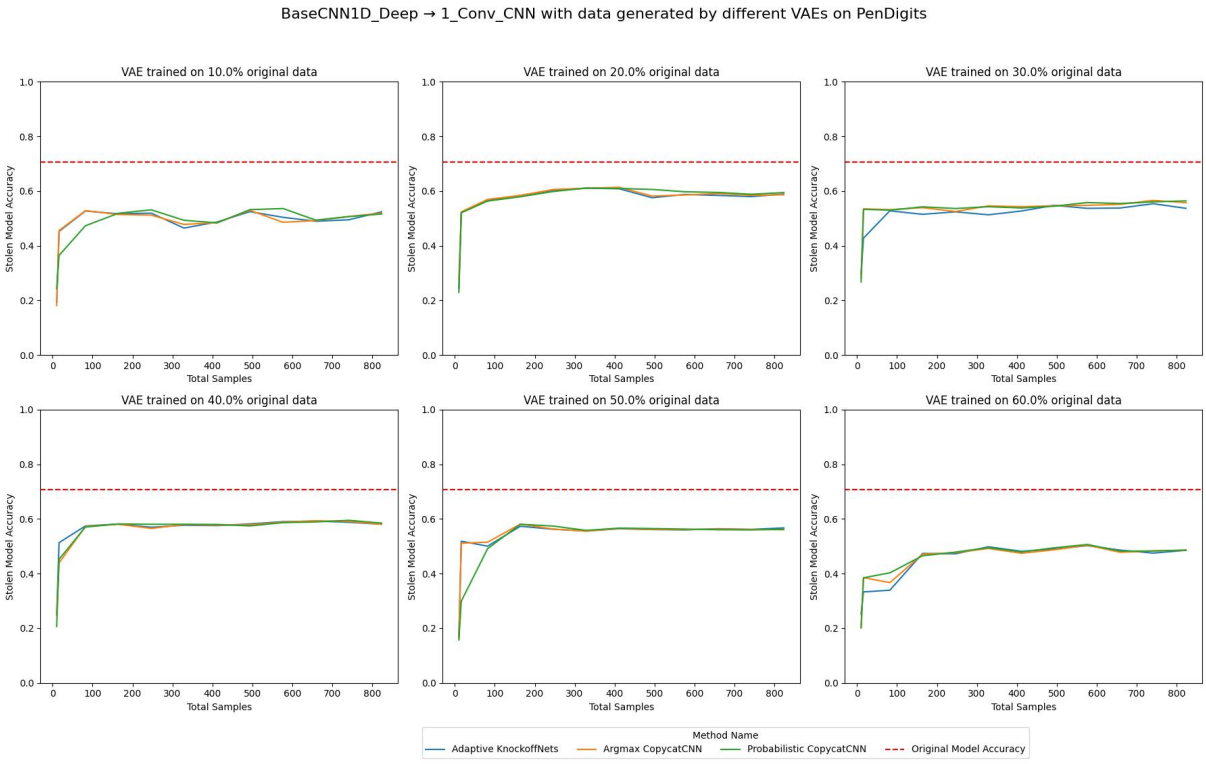

Figure 17: Extraction attacks using only generated data on PenDigits

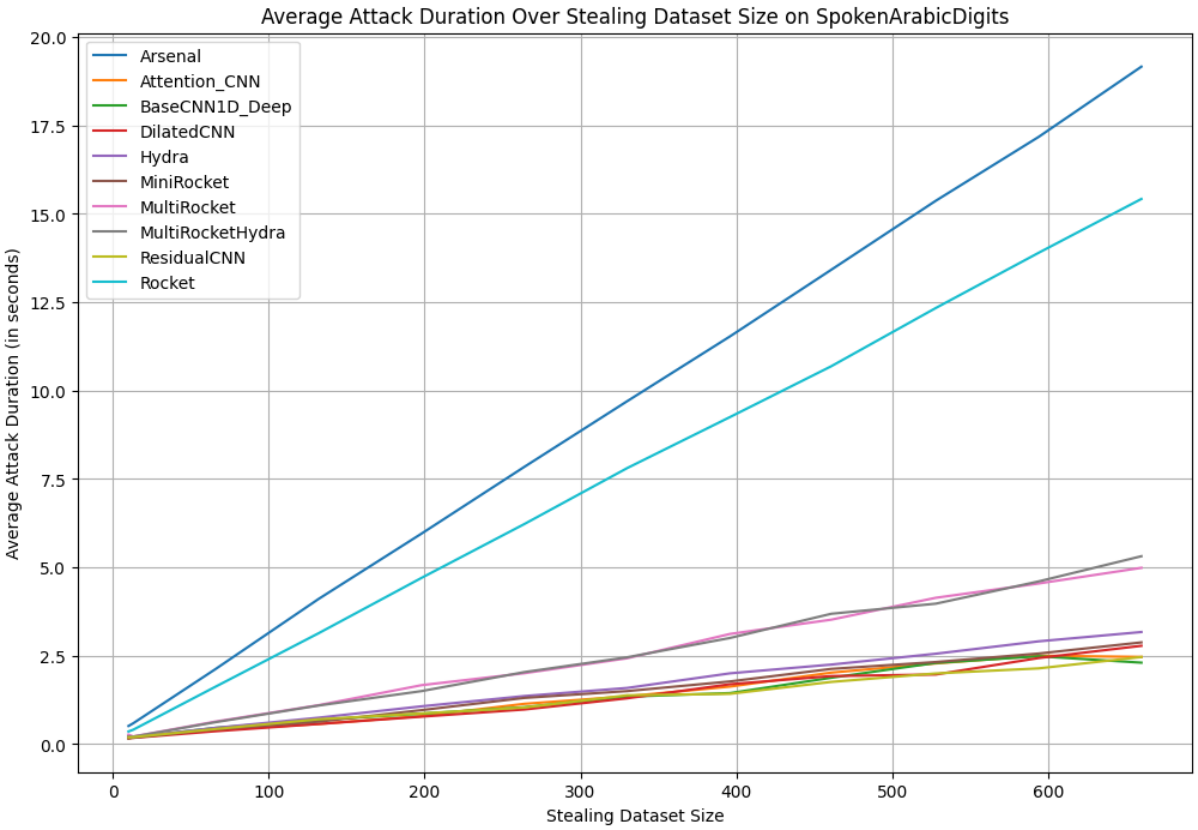

Figure 18: Average attack duration

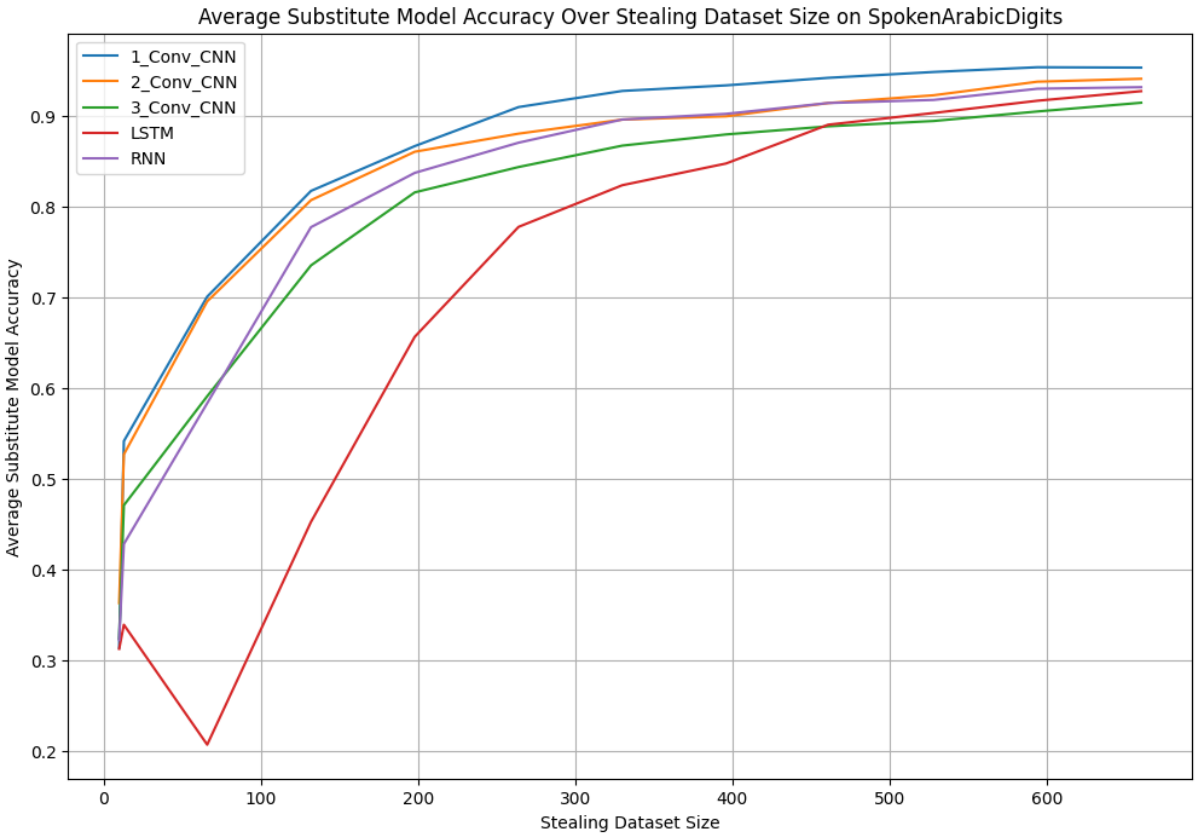

Figure 19: Average substitute model Accuracy over all model combinations and stealing dataset sizes