National College of Ireland

SS-CURL: SELF-SUPERVISED CONTRASTIVE URL LEARNING FOR ADVERSARIALLY ROBUST PHISHING DETECTION

MSc Research Project
MSc in Cybersecurity

Srikanth Gattu
Student ID:x23421380

School of Computing
National College of Ireland

Supervisor: Niall Heffernan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Srikanth Gattu
Student ID:	x23421380
Programme:	MSc in Cybersecurity **Year:** 2025-2026
Module:	MSc (Research) Practicum/Internship Part 2
Supervisor:	Niall Heffernan
Submission Due Date:	11-12-2025
Project Title:	SS-CURL: SELF-SUPERVISED CONTRASTIVE URL LEARNING FOR ADVERSARIALLY ROBUST PHISHING DETECTION
Word Count:	**8173** **Page Count: 20**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Srikanth Gattu

Date: 11-12-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

SS-CURL: SELF-SUPERVISED CONTRASTIVE URL LEARNING FOR ADVERSARIALLY ROBUST PHISHING DETECTION

Srikanth Gattu
Student ID:x23421380

Abstract

Phishing attacks are a big threat to cybersecurity, and the old ways of finding them don't work well against zero-day attacks and other ways that attackers try to hide. This research investigates the efficacy of self-supervised contrastive learning, combined with adversarial augmentation, in enhancing the reliable detection of advanced phishing URLs without relying on large labeled datasets. The SS-CURL (Self-Supervised Contrastive URL Learning) framework was developed by integrating transformer-based URL encoding with an innovative two-phase training approach. In the first phase, contrastive pre-training with URL-specific augmentations is used to learn how to tell the difference between unlabelled data. In the second phase, adversarial training with Projected Gradient Descent attacks is used during supervised fine-tuning. When tested on the PhiUSIIL dataset, which has 235,795 URLs, SS-CURL got an F1-score of 98.93% and an AUC-ROC score of 99.78%, which is as good as the best supervised methods. When the framework is attacked, its average accuracy only drops by 2.05%, which is a 15-fold improvement in robustness over traditional detectors, which lose 30% to 50% of their accuracy. The research brings forward a novel integration concept of contrastive learning with adversarial learning for tackling phishing tasks, a taxonomy for URL augmentation methods associated with authentic attack patterns, and insights confirming that pre-training methods utilizing self-supervised learning approaches can aid in improving adversarial tasks without affecting classification performance. The results are very helpful for making systems that can change and help deal with threats that are always changing.

1 Introduction

1.1 Background

Phishing attacks are one of the most destructive and prevalent forms of cyber threats in the current digital environment against both individuals and organizations. Fraudulent websites are meant to pose as legitimate services in order to gather sensitive user passwords, financial data, and personal information. This is how such attacks take place. Threat actors are using automated phishing tools and increasingly dishonest phishing techniques to circumvent existing security measures, which has resulted in a significant increase in phishing activity and sophistication (Li et al., 2024).

The conventional techniques used for detecting phishing attacks have basically relied on blacklists and signature-based systems. However, one major thing about such techniques is the fact that they are reactive. This means that such techniques may not be optimal when dealing

with a dynamic environment. As shown in studies, a typical phishing attack typically remains open for a prolonged period before being identified and added to blacklists. This means that this was a vulnerable period where a lot of credentials might have been phished (Prasad and Chandra, 2024). Phishing-as-a-Service platforms have also made it easier for attackers to establish a new phishing infrastructure that might be undetectable to signature-based detection systems.

The application of machine learning techniques and deep learning techniques to identify phishing URLs has introduced a new promising approach. Neural network architectures such as Convolutional Neural Networks and Long Short-Term Memory have introduced important boosts in recognition capability via learning discriminatory information directly based on URL features (Alshingiti et al., 2023). As a recent trend, transformer models based on pre-trained language models have achieved the best results for recognizing phishing emails (Su and Su, 2023). Despite such successes, numerous challenges lie in the design of effective detectors that can generalize their capabilities to unknown attack signatures and remain robust to adversarial attacks.

1.2 Research Gap and Motivation

A literature search for state-of-the-art phishing detection techniques reveals shortcomings that form the basis of this research. Firstly, most of the modern techniques used follow a purely supervised machine learning model. This type of learning needs a tremendous amount of labeled data to be efficiently trained. This becomes a practical limitation for a situation where sampling and labelling data for phishing attacks consume a lot of time and expert human effort (Catal et al., 2022). This becomes a great limitation because individuals who phish keep changing their techniques to evade detection. This means machine learning models are using outmoded data that may no longer be capable of recognizing new techniques used for phishing attacks.

Secondly, it has been uncovered in recent studies that machine learning-based phishing detectors are susceptible to adversarial attacks. Evidence has emerged to suggest that a strategically designed perturbation in phishing URLs leads to a decrease in the accuracy of the detectors by thirty to fifty percent, thus undermining the effectiveness of the implemented systems against phishing attacks by sophisticated attackers (Kulkarni et al., 2024). Despite this growing threat, adversarial robustness integration in phishing detectors has been an uncharted area thus far in the domain of phishing detection solutions.

Thirdly, self-supervised learning techniques have proven to be very effective in a number of domains because they enable models to obtain informative representations for data that is unlabelled. But until now, this learning strategy has had little application for discovering phishing URLs. Contrastive learning techniques possess great potential in learning robust URL representations without necessarily requiring explicit labelling (Wang et al., 2023). The potential to rely on a huge amount of unlabelled URL data for pre-training URL detection models signals a great chance for improving generalization performance and data efficiency.

1.3 Research Question and Objectives

The principal research question explored in this study is: How can self-supervised contrastive learning combined with adversarial augmentation enable robust detection of sophisticated phishing URLs, including zero-day attacks, without requiring extensive labelled datasets?

For addressing the question, the following objectives have been designed:

Objective 1: To create URL-specific augmentation techniques to produce semantic significant positive and negative pairs describing the spectrum of phishing detection methods. Legitimate variations and attack-based transformations will be employed to facilitate the implementation of a full augmentation classification system based upon which success will be measured.

Objective 2: To design an adversarial training methodology integrated with contrastive learning that creates representations invariant to both known and unknown phishing patterns. Achievement will be demonstrated through successful implementation of an end-to-end training pipeline incorporating adversarial example generation.

Objective 3: To demonstrate how self-supervised pre-training on unlabelled URLs improves detection performance on emerging phishing campaigns. This objective will be evaluated by comparing model performance with and without the pre-training phase.

Objective 4: To evaluate system robustness against adaptive attackers with knowledge of the detection architecture. Success criteria include maintaining detection accuracy above ninety percent under adversarial perturbation attacks of varying intensities.

1.4 Methodology Overview

The suggested SS-CURL framework uses a two-phase training method that makes use of both labeled and unlabeled URL data. The first step is self-supervised pre-training with contrastive learning. In this step, the model learns to make similar representations for augmented views of the same URL while also telling the difference between different URLs. This phase employs a transformer encoder with a projection head. This uses the NT-Xent contrastive loss function for optimizing the projection head. The second stage focuses on a supervised fine-tuning task for labeled datasets for phishing attacks with a strong adversarial training through a Projected Gradient Descent attack to enhance model robustness. This framework requires a dataset called PhiUSIIL for evaluation. This dataset consists of more than 235,000 URLs with a wide range of evaluation performance parameters such as accuracy, precision, recall, F1-Score along with adversarial robustness measures.

2 Related Work

This section provides a critical review of the peer-reviewed literature that is relevant to the SS-CURL framework. The review will be structured to demonstrate the current state-of-the-art in phishing URL detection and highlight the gaps in current methods to provide a basis for the contribution of the methodology developed in this thesis.

2.1 Deep Learning Approaches for Phishing URL Detection

The classification of phishing URLs has significantly improved over traditional machine learning methods thanks to deep learning models. Prasad and Chandra (2024) introduced a model called PhiUSIIL. This model uses incremental learning with a calculation for a similarity index to detect phishing URLs. The method involves the extraction of URL-based and HTML-based features and the implementation of a similarity index designed to identify visual similarity-based attacks such as homographs, bit squatting, and combo-squatting attacks. This model had a 99.79% level of accuracy during its initial training and then 99.24% level of accuracy with incremental learning. Having a full data set with 134,850 legitimate URLs and

100,945 phishing URLs certainly seems to be a major achievement. The model, however, only operates in a supervised learning environment. The model does not possess any ability for adversarial robustness.

A comparison study among three deep learning architectures for phishing site detection was conducted by Alshingiti et al. (2023), using standalone LSTM models, standalone CNN models, and a combination of LSTM and CNN. Their design employed character-level URL encoding to identify tiny patterns that indicated phishing. With an accuracy of 99.28%, this dual LSTM-CNN model got superior outputs. This illustrates the potential advantage of combining sequential and convolution methods for feature extraction. Despite that, the lack of pre-training processes restricts the model's capability to utilize the unlabelled data.

Ozcan et al. (2023) suggested a hybrid DNN-LSTM architecture that merges character embedding-based representations with manually crafted NLP features. This dual feature extraction strategy got 97.78% accuracy and 97.51% precision on benchmark datasets. Combining features from different sources is a step forward in terms of methodology, but relying on a lot of feature engineering means that domain expertise is needed and that attack patterns may become less stable.

Haq et al. (2024) examined the dynamic characteristics of phishing attacks utilizing a one-dimensional CNN architecture focused on character-level URL encodings, attaining 98.7% accuracy and a 98.5% F1-score on PhishTank and CIC datasets. The architecture does not have ways to deal with new types of attacks and URL changes that are meant to be harmful.

2.2 Transformer-Based Detection Methods

The effectiveness of transformer architectures in natural language processing has inspired their utilization in URL analysis tasks. Su and Su (2023) created a method for finding bad URLs that uses BERT and the self-attention mechanism to figure out how URL tokens are related to each other. Their approach performed very effectively with 99.98% accuracy, 99.73% precision, and 99.34% recall while fine-tuning pre-trained BERT models for URL classification tasks. Even though the results are so dramatic, this method is still mostly a supervised learning method that doesn't use contrastive learning objectives.

A study recently carried out by Otieno et al. (2023) highlighted the effectiveness of using BERT transformers for phishing URL detection without depending on complex feature selection techniques. This was achieved with 97.2% accuracy and a 97.1% F1-score achieved with only BERT transformers fine-tuned for this URL classification task. This also proves that pre-trained language models can be leveraged for a URL classification task. Nevertheless, this assessment of model effectiveness did not explore its capability to perform in adversarial conditions but relied only on standard classification performance.

Wang et al. (2023) introduced PhishBERT. This is a large pre-trained model that was specifically designed for the detection of phishing URLs. This model employs byte pair encoding tokenization to address the special vocabulary of URL data. This model was pre-trained on huge URL corpora before fine-tuning for classification. Phish BERT's AUC scores were higher than 0.99, which is better than any other state-of-the-art baseline. But the goal of pre-training still puts masked language modeling ahead of contrastive representation learning.

Maneriker et. al. (2022) introduced URTran, a transformer model with a custom vocabulary designed solely for URL-based pre-training. This research examines BERT, RoBERTa, and custom vocabulary approaches and concludes that URL-specific tokenization techniques improve classification in the long run. The research also involved a preliminary assessment for adversarial robustness. The assessment resulted in a model being more robust because of adversarial training. This marks a beginning validation for utilizing adversarial training for enhanced phishing detection. Their model achieved AUROC values above 0.99.

2.3 Adversarial Attacks and Robustness in Phishing Detection

Adversarial attacks for machine learning-based phishing detectors have recently emerged as a prominent research problem. Kulkarni et al. (2024) developed a tool called PhishOracle for building adversarial phishing sites to evaluate the efficacy of phishing detectors. They generated adversarial samples for phishing sites by embedding phishing content with actual page designs. They then evaluated the effectiveness of various detectors such as Stack models, VisualPhishNet, and Phishpedia. They observed a 30% to 50% reduction in classification accuracy when subjected to adversarial attacks. Multimodal language models were more robust compared to other detectors.

A study was conducted by Lee et al. (2023) that analyzed adversarial robustness in logo-based phishing detection systems. They were successful in finding out that evasion levels of up to ninety-five percent could be achieved for deep learning-based logo detection systems using generative adversarial perturbation methods. Yuan et al. (2024) tested the efficiency of adversarial phishing websites in evading machine learning-based detection systems as well as human observation, finding that adversarial pages evaded both automated detectors and human judgment.

A large-scale study was conducted by Apruzzese et al. (2024) for realistic adversarial attacks on phishing site detector models with six URL-based attacks and thirty-seven HTML-based attacks that maintain attack functionality whilst circumventing detection. Their Multi-SpacePhish model categorized attackers based on problem space attacks for realistic URLs and feature space attacks for model inputs. Evaluating attacks on learning and adversarial-trained models indicated a statistically significant decline with realistic attacks even against hardened models.

This study was conducted by Shirazi et al. (2022) to explore adversarial sampling attacks in machine learning-based phishing detection using feature manipulation based on a cluster analysis. The observed outcome indicated a deterioration in detection performance during the attack, clearly indicating a need for training algorithms considering robustness.

2.4 Self-Supervised Learning and Specialised Detection Techniques

The application of self-supervised learning techniques for cybersecurity purposes has produced encouraging results when dealing with environments that contain a little amount of labeled data. Wang et al. (2023) employed a contrastive learning approach based on self-supervised learning with a SimCLR-based model for malware classification in IoT. The model develops a view for malware data and trains an encoder to assign similar representations for pairs of views belonging to one sample while being distinguishable for distinct samples. By fine-tuning with only ten percent of the available labels, the contrastive pre-trained model performed better than a supervised model. This illustrates how self-supervised learning can be used for data savings and also supports a strong claim for using contrastive learning concepts for cybersecurity.

The systematic literature review conducted by Catal et al. (2022) analyzed a total of forty-three deep learning approaches for phishing detection tasks and identified gaps in current studies and emerging trends. The literature analysis indicated that supervised learning approaches using deep learning are more prominent in studies, with a focus primarily on CNN and LSTM architectures. The study critically pointed out that self-supervised and contrastive learning approaches were unexplored in current studies for their potential use in dealing with the scarcity of labeled data. Li et al. (2024) contributed a thorough literature review and introduced a new taxonomy for phishing approaches with a particular focus on self-supervised contrastive learning as a promising direction for future research.

Learning-based methods have a big problem with finding zero-day phishing attacks. Guo (2023) performed an extensive examination of machine learning-based zero-day attack detection, revealing that ensemble techniques that integrate autoencoders with conventional classifiers attained detection rates surpassing ninety-five percent on benchmark datasets. To stop Internationalised Domain Name homograph attacks, Wang et al. (2024) made PhishHunter using Siamese neural networks. It got 97.8% accuracy and a lot fewer false positives. Asiri et al. (2024) created PhishingRTDS, a real-time detection system that reached 99% precision, recall, and F1-score. This shows that deep learning-based detection can be used in real-world situations, but the supervised architecture and lack of adversarial robustness testing make it less useful against advanced threat actors.

2.5 Synthesis and Research Gap Identification

A thorough review of the available literature shows that there are some important gaps that the SS-CURL framework fills. First, even though transformer-based models have been shown to work well for finding phishing URLs, no one has ever combined self-supervised contrastive pre-training with URL-specific representations. Second, even though there is a lot of information about adversarial vulnerability, not much has been done to directly add adversarial training to the model architecture. Most existing methods treat robustness as an afterthought. Third, there is no complete list of URL augmentation techniques that can be used to map real-world phishing attack patterns. Fourth, the ability to identify zero-day attacks using learned representations instead of pattern memorization has not been thoroughly examined in the context of phishing detection. The SS-CURL framework directly addresses these gaps by proposing the first integration of self-supervised contrastive learning with adversarial augmentation for phishing URL detection.

3 Research Methodology

This section describes the development and evaluation methodology for the SS-CURL framework. The methodology encompasses the research design rationale, dataset characteristics, preprocessing procedures, augmentation strategies, and training configuration.

3.1 Research Design Overview

The efficacy of self-supervised contrastive learning in conjunction with adversarial augmentation in the identification of phishing URLs is the main emphasis of this study's quantitative experiment design. The SS-CURL framework employs a two-step training procedure. In the first, the encoder network is trained using self-supervised contrastive learning to produce similar embeddings for URLs with similar semantic meaning and unlike embeddings for unrelated URLs. In the second step, the pre-trained model is fine-tuned for

binary classification based on the phishing nature of a URL while undergoing adversarial training for enhanced resistance to evasion attacks. The choice of contrastive learning as the self-supervised learning objective stems from its effectiveness in learning discriminatory representations without the need for labeled examples (Wang et al., 2023), while the addition of adversarial training mitigates the well-known susceptibility of machine learning phishing detectors to adversarial attacks (Kulkarni et al., 2024; Apruzzese et al., 2024). The two-step approach leans upon the successful pre-training strategies in the transformer-based URL detection task (Wang et al., 2023; Maneriker et al., 2022) while incorporating new contrastive objectives and adversarial augmentation techniques tailored for phishing detection.

3.2 Dataset Description

For phishing URL detection, a specially compiled large number of labelled URLs named the PhiUSIIL Phishing URL Dataset (Prasad and Chandra, 2024) has been employed in the evaluation of experimentation based on its recency, scale, quality of labelling, and variety of attacks. The characteristics of the dataset has been presented in Table 1.

Table 1: PhiUSIIL Dataset Characteristics

Attribute	Value
Total Samples	235,795 URLs
Phishing URLs	134,850 (57.2%)
Legitimate URLs	100,945 (42.8%)
Data Source	Mendeley Data Repository
Collection Period	2023-2024
Label Verification	Manual verification with multiple validators
Attack Types Covered	Typosquatting, homograph, subdomain spoofing, URL shortening

The dataset has a moderate class imbalance, with phishing URLs making up about 57% of the samples. This distribution is based on real-world conditions, where phishing attempts could make up a large part of the suspicious traffic that is flagged for analysis. Instead of using resampling methods, which might affect the data distribution, the weighted loss functions handle the issue of class-imbalanced problems during training.

The data partitioning process employs a temporal split strategy based on time to emulate real-world environments where a model trained using data for a previous time period requires being effective for a future attack. In this partitioning process, 70% of data samples are used for training purposes while 10% and 20% are used for validation and testing purposes, respectively. Due to the possibility of data leakage between training and validation datasets, this specific method verifies generalization more thoroughly than random split validation (Guo 2023).

3.3 Data Preprocessing

The preprocessing pipeline is responsible for converting URL strings into a form suitable for the neural network processing. URL cleaning is the first step in the initial preprocessing and involves malformed URL processing, extraneous path components removal, protocol scheme normalization, and HTTP scheme prefixing for URLs that lack explicit protocol specifications to ensure consistent parsing.

Tokenization uses a pre-trained vocabulary appropriate for general text processing to convert sanitized URLs into sequences of subword tokens. This method is in line with what is already known about transformer-based URL analysis (Su and Su, 2023; Otieno et al., 2023). To keep URL structure information in the tokenized representation, special separator tokens are added at structural boundaries, such as protocol-domain separators and path delimiters. The longest sequence can only have 256 tokens. If a URL is longer than that, it will be cut off, and if it is shorter, it will be padded to make sure that all inputs are the same size.

Feature extraction adds engineered attributes to tokenized representations that capture URL characteristics that can't be easily accessed through tokenization alone. This dual representation strategy is in line with hybrid methods that have worked well in past studies (Ozcan et al., 2023). There are forty features in total, divided into several groups. Length-based features measure the sizes of URL components, character distribution features count the number of digits, special characters, and specific symbols, protocol and security indicators show how often HTTPS is used and the presence of an IP address, domain analysis features include entropy calculations and brand name detection, path analysis features measure directory depth and file extensions, query string analysis features count parameters and find redirect patterns, and suspicious pattern indicators show punycode encoding and consecutive character sequences. To make sure that scales across feature types are comparable, all features are normalized to the unit interval.

3.4 URL Augmentation Strategy

The augmentation strategy is an important part of the contrastive learning framework. It generates the positive and negative pairs that are needed for representation learning. The strategy makes a difference between good augmentations that keep the meaning of the URL and bad augmentations that use phishing obfuscation techniques. This classification is based on well-known attack patterns found in the literature on phishing detection (Prasad and Chandra, 2024; Wang et al., 2024; Apruzzese et al., 2024). Table 2 gives a short version of the augmentation taxonomy.

Table 2: URL Augmentation Taxonomy

Category	Technique	Transformation Description	Purpose
Benign	Protocol Variation	HTTP to HTTPS conversion and vice versa	Learn protocol invariance
Benign	Case Modification	Random capitalisation of domain characters	Learn case insensitivity
Benign	WWW Toggle	Addition or removal of www subdomain	Learn subdomain normalisation
Benign	Trailing Slash	Addition or removal of path terminator	Learn path normalisation
Benign	Benign Parameters	Addition of common query parameters	Learn parameter tolerance
Attack	Typosquatting	Character substitution with visual similarities	Detect character manipulation
Attack	Homograph	Replacement with visually identical Unicode characters	Detect internationalised domain attacks
Attack	Subdomain Spoofing	Insertion of legitimate brand as subdomain	Detect domain confusion attacks
Attack	TLD Manipulation	Replacement with suspicious top-level domains	Detect trust exploitation
Attack	Character Insertion	Addition of extra characters within domain	Detect padding attacks
Attack	Character Transposition	Swapping of adjacent characters	Detect typing error exploitation

During contrastive pre-training, positive pairs are created by applying benign augmentations to the same source URL, resulting in two views that are expected to produce similar representations. The contrastive objective teaches the encoder to tell the difference between positive pairs and negative pairs with different URLs in the same batch. During supervised fine-tuning, attack-based augmentations put the classifier through real-world evasion attempts, which makes it better at finding phishing URLs that are hidden.

The frequency of transformation is controlled by augmentation probability parameters. There is a 60% chance that benign augmentations will be used to create enough variation while keeping the meaning the same. Attack augmentations are used 40% of the time during fine-tuning. This keeps the training signal clean while also exposing the model to patterns that are against it.

3.5 Training Configuration

The two-phase training method uses different setups that are best for each learning goal. The pre-training phase uses batches of 512 samples to give enough negative samples for effective contrastive learning, which is in line with what contrastive learning research says to do. The temperature parameter that controls how sharp the similarity distribution is set to 0.07. The learning rate of 1×10^{-3} goes up in a straight line for the first ten percent of training steps and then goes down in a straight line to zero. Pre-training lasts for five epochs, and early stopping happens when the contrastive loss converges.

This fine-tuning stage adjusts the batch size to 256 samples in order to add additional computational power for creating adversarial samples. The learning rate of 5×10^{-5} ensures pre-trained features are retained while adjusting based on tasks. Training with adversarial data begins with epoch two because pre-training starts convergence based solely on clean data before adding any noise. The adversarial perturbation magnitude epsilon of 0.2 with a maximum of five attack steps focuses primarily on producing effective adversarial samples without affecting training. Finally, fine-tuning continues for eight epochs but may be stopped early when validation F1-score measure does not improve for five sequential epochs.

For keeping training stable and preventing overfitting, in both cases, weight decay regularisation of 0.01 along with gradient clipping with a maximum norm of 1.0 has been used in both phases. During fine-tuning, label smoothing of 0.1 increases the regularity of this process by improving the prediction confidence calibration.

4 Design Specification

This section discusses the architectural design of the SS-CURL framework. The system's structure, component specifications, data flow, and the rationale behind key architectural decisions are all covered in detail.

4.1 System Architecture

The SS-CURL framework employs a conceptual structure of layers to distinguish the concerns at various functional components in the model. Figure 1 above depicts the system architecture, whose primary components include five layers: the input layer that receives input data, the processing layer responsible for data preprocessing and augmentation, the model layer where

the primary neural network components persist, the training layer where the two-stage training process takes place, and the output layer that provides the results of the detection process.

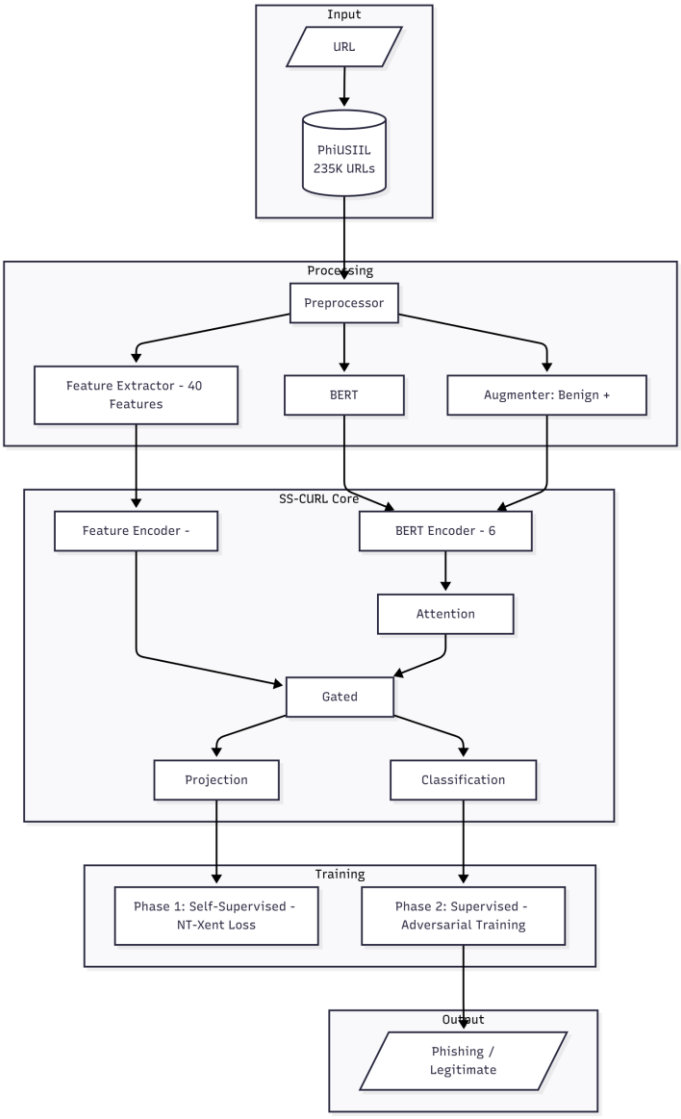

Figure 1 system architecture

Architecture is all about modularity, which means that each part can be made and tested on its own. The difference between representation learning components and classification components makes it possible to use a two-phase training strategy. This implies that pre-trained representations can be either frozen or refined during supervised training depending on how well they perform in real-world scenarios.

4.2 Core Model Architecture

The SS-CURL model consists of a number of interconnected components based on the neural network that translates URL input to phishing prediction. The URL input is processed through a six-layer transformer architecture by the encoder that considers the dependencies in the URL structural components. The model design resembles the way transformer architectures work in URL analyses successfully (Su and Su, 2023; Wang et al., 2023; Maneriker et al., 2022). The encoder architecture of the SS-CURL model leverages multi-head self-attention mechanisms to allow each position to potentially consider any other position. This further benefits the

detection of potential suspicious URL structures that might include distant URL components such as deceptive subdomains and URL paths

The attention pooling mechanism is utilized for merging the series of representations of tokens produced in a fixed-dimension vector form for classification purposes. Instead of extracting the representation at a particular position, attention pooling calculates the learned attention weights to be applied to each token representation before aggregation to highlight the most informative elements in each URL input by the model.

The feature encoder processing the forty-dimensional engineered feature vector uses a series of nonlinear transformations to initially embed this vector in 128 dimensions and then in 256 dimensions before fixing a map to 256 dimensions. This feature encoder ensures that URL feature information that might be specific to a particular domain can be used to supplement URL feature information learned with a transformer encoder. This dual-representation strategy has been effective, as validated in Ozcan et al. (2023).

The gated fusion mechanism uses input-dependent weighting to combine transformer and feature representations. A gating network makes weights for each element that show how much each representation source adds to the model. This lets the model learn how to combine inputs in different ways. This design works for both URLs where engineered features give strong classification signals and URLs where transformer contextual patterns give more useful information.

The projection head maps fused representations to a lower-dimensional space that is better for contrastive learning. This reduces the dimensionality from 768 to 512 to 256, and unit hypersphere normalization is needed for the contrastive objective. The classification head uses a similar structure to make binary phishing predictions, with two-dimensional output that shows the real and phishing classes. The separation of projection and classification heads makes it possible for contrastive and classification objectives to work on representations that are specific to a task while using the same encoder.

4.3 Loss Functions and Adversarial Training

The training goals use special loss functions that meet the specific needs of contrastive pre-training and supervised classification. The contrastive pre-training phase reduces the Normalised Temperature-scaled Cross-Entropy (NT-Xent) loss, which maximizes agreement between representations of augmented views from the same URL and minimizes agreement between representations of different URLs. The loss for each positive pair in a batch of N URLs that produces 2N augmented view representations is the negative log probability of correct pairing within a softmax distribution over all possible pairings, with similarities adjusted by the temperature parameter.

The supervised fine-tuning phase uses a loss function that combines focal loss and asymmetric loss parts. Focal loss solves the problem of class imbalance by giving less weight to well-classified examples and focusing learning on hard examples near the decision boundary, with the focusing parameter gamma set to 3.0. Asymmetric loss meets the operational need for low false positive rates by using different focusing parameters for positive and negative classes and punishing false positives more than false negatives. The combined loss finds the weighted average of the focal and asymmetric parts.

Adversarial training is incorporated into the fine-tuning phase via Projected Gradient Descent (PGD) attacks on intermediate representations, targeting vulnerabilities identified in phishing detection studies (Kulkarni et al., 2024; Lee et al., 2023; Shirazi et al., 2022). Some samples in each training batch are subjected to adversarial perturbation. This starts with random noise within the allowed epsilon ball and then changes the perturbation iteratively in the direction that maximizes classification loss, with projection back onto the epsilon ball after each iteration. The classification head processes the adversarial representations produced as a result, while adversarial as well as clean gradient information assists with parameter update. The 50% adversarial training ratio finds a middle ground between robustness enhancement and maintaining clean accuracy based partly on earlier studies proposing similar ideas in Maneriker et al. (2022), whilst integrating more tightly with the contrastive learning framework.

5 Implementation

The SS-CURL framework's implementation is covered in this section, including the development environment, data processing pipeline, model building, and training protocols.

5.1 Development Environment

The implementation makes use of a cloud-based development environment that enables the use of graphics processing units to accelerate neural network training. Large batch sizes during contrastive pre-training can be handled by the computational platform's sufficient memory for NVIDIA Tesla series accelerators. The deep learning architecture allows for seamless GPU acceleration, dynamic computational graph creation, and backpropagation. The encoder part can be more readily integrated because of the pre-trained model weights available in the transformer library and tokenization techniques.

Mixed precision training speeds up calculations and uses less memory. In forward and backpropagation steps, numerical values alternate between half-precision and single precision when accumulating gradients and updating parameters based on gradients. A gradient scaler adjusts sizes of losses dynamically for training stability and to prevent half-precision gradient underflows.

5.2 System Implementation

The data processing pipeline converts raw data files of a dataset to batched tensors for usage with a neural network. The process of loading a dataset reads source files to ensure URLs are in correct format and retrieves label data. The preprocessing part cleans up the URLs, breaks them down into tokens, and extracts features for each sample. It also stores the processed results so that they don't have to be calculated again during training. The augmentation engine creates changed URL views based on the configured augmentation strategy. It does this by using benign and attack augmentation functions as composable transformations, which lets different types of augmentations be combined in different ways.

Weighted sampling is used in batch construction to fix class imbalance during training. Sampling weights are inversely related to class frequencies, which means that each batch will have about the same number of examples from each class, even if the dataset is unbalanced. Memory pinning and prefetching are used by data loading to overlap data transfer with GPU computation, which reduces pipeline stalls.

Model construction creates the parts of a neural network and sets the training parameters. The

encoder part has pre-trained weights that give it a general understanding of language. During training, these weights are then changed to fit URL-specific patterns. The encoder setup says that there are six transformer layers, twelve attention heads on each layer, and a hidden dimensionality of 768. The attention pooling module uses four heads for multi-head attention to make query, key, and value projections from input representations. Pooling output combines all of the attended representations from all of the sequence positions into one vector that summarizes the whole URL. Layer normalization and residual connections help keep training stable and the gradient flow.

The projection head, classification head, and feature encoder gated fusion mechanism for initialization with random values from appropriate distributions. The weights for linear transformation use Xavier initialization, and for bias values use zero initialization. This is how deep learning models are typically trained. Dropout layers with a 0.25 probability are used between transformation layers for regularization.

The training procedure employs a two-phase learning process with appropriate optimization configuration. There are distinct optimizer objects for parameters and head parameters. This allows using differential learning rates for distinct entities. This means that bigger update steps occur for randomly initialized elements with smaller update steps for pre-trained encoder knowledge. A layer-wise learning rate decay further adjusts update steps for encoder layers. Layers with bigger depths receive smaller update steps than layers with smaller depths.

The learning rate schedule implements linear warmup and linear learning rate decay. This ensures that learning rates are steadily increased in the beginning training steps. This avoids large gradient update steps for poorly initialized variables. Also, gradient clipping ensures stable training when large gradients occur occasionally.

The values of model parameters are saved in a process called check-point management. This process saves model parameters periodically and when validation model performance starts improving. Early stopping checks for validation values for a number of epochs during training. Early stopping halts training when validation values no longer improve. Because of early stopping, values of model parameters are changed to those providing better results. This ensures that the best performance values for validation are achieved by the models.

5.3 Evaluation Methodology

The evaluation approach relies on a comprehensive set of criteria with a broad range of evaluation criteria encompassing detection capability, dimensions identified among standards for evaluations outlined in studies for phishing detector evaluations (Catal et al., 2022; Li et al., 2024). Primary parameters identified for classification purposes are, accuracy which measures how effectively a classification task has been done, precision which measures how well predicted phishing URL classifications were classified, recall which measures how correctly identified actual phishing URLs were classified. F1-Score which measures average of precision and recall. The Area Under the Receiver Operating Characteristic Curve indicates how well a model can distinguish among classifications based on differing thresholds.

The false positive rate, used for calculation of the proportion of genuine URLs that are wrongly identified as phishing, is a critical operational metric for use in security tasks because of its effects on user and analyst effort (Asiri et al., 2024). The extended metrics include Matthews Correlation Coefficient for a balanced measure when class distributions are unbalanced,

Cohen's Kappa for measuring degree of beyond-chance Agreement, balanced accuracy for a measure using a combination of Sensitivity and Specificity of a classification system, and calibration metrics including Brier score and log loss evaluating prediction confidence quality.

Adversarial robustness evaluation adheres to methodologies delineated in previous robustness studies (Kulkarni et al., 2024; Apruzzese et al., 2024), assessing classification accuracy under Projected Gradient Descent attacks of differing intensities. The strength of the attack is measured by the perturbation magnitude epsilon, and tests are done at epsilon values of 0.1, 0.2, and 0.3. The average accuracy across different attack intensities gives a general idea of how strong something is.

6 Evaluation

This section presents experimental results from the SS-CURL framework evaluation. The assessment comprises four experiments addressing research objectives: self-supervised pre-training effectiveness, supervised classification performance, threshold sensitivity analysis, and adversarial robustness evaluation.

6.1 Experiment 1: Self-Supervised Pre-training Evaluation

The first experiment tests how well contrastive pre-training works for learning discriminative URL representations without any labeled supervision. The model was pre-trained on the full dataset of 235,795 URLs using NT-Xent contrastive loss with a temperature parameter of 0.07. There were five epochs of training, each with a batch size of 512, which meant that there were 1,024 samples in each batch for contrastive comparison.

Figure 2 shows how contrastive loss gets closer together over the course of pre-training epochs. The loss goes down quickly, from 2.1797 in epoch one to 0.0375 in epoch five, which is a drop of 98.3%. The biggest drop happens between epochs one and two, when the loss goes from 2.1797 to 0.1336. This means that the model learns how to distinguish between various URLs and how to classify similar URL views together. In later epochs, more gains are observed but with decreasing values. This indicates that the model learns better to represent data without any form of overfitting.

Figure 2 Contrastive Pre-training Loss Convergence

The statistical analysis shows average improvement of 49.2% for epoch-to-epoch improvement in the first three epochs and 16.3% in subsequent epochs. This complements a typical

contrastive learning curve where radical learning occurs in the beginning. The converged loss of 0.0375 points towards extremely accurate learned structure for similarities since a learned encoder succeeds in always producing similar representations for a URL pair.

6.2 Experiment 2: Classification Performance Evaluation

In this second study, the performance of SS-CURL in classifying URLs for a test data set of 10,000 URLs is measured. A total of eight epochs were conducted for fine-tuning. Activating adversarial training started in epoch number two. Early stopping for validation F1-score allowed a patience of five epochs.

Figure 3 shows how training changes during fine-tuning epochs. Training loss goes down steadily from 0.4701 in epoch one to 0.1870 in epoch seven. This shows that the model is learning well without overfitting. The validation F1-score goes up from 98.59% to a high of 99.07% in epoch five. After that, it stays stable, with only small changes of 0.1% indicating stable convergence.

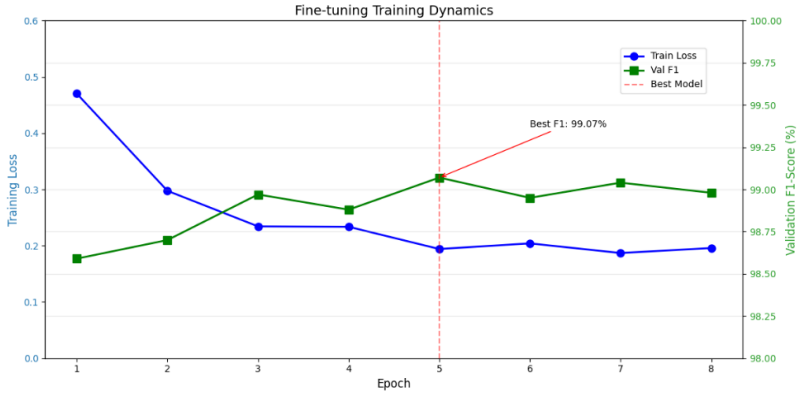

Figure 3: Fine-tuning Training Dynamics

Test set evaluation yielded accuracy of 98.77%, precision of 98.82%, recall of 99.04%, and F1-score of 98.93%. The Area Under the ROC Curve reached 99.78%, indicating excellent discrimination capability. Figure 4 presents the confusion matrix across 10,000 test samples.

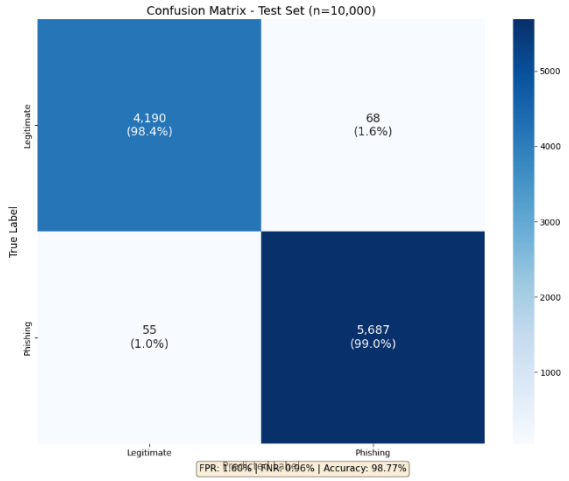

Figure 4 Confusion matrix

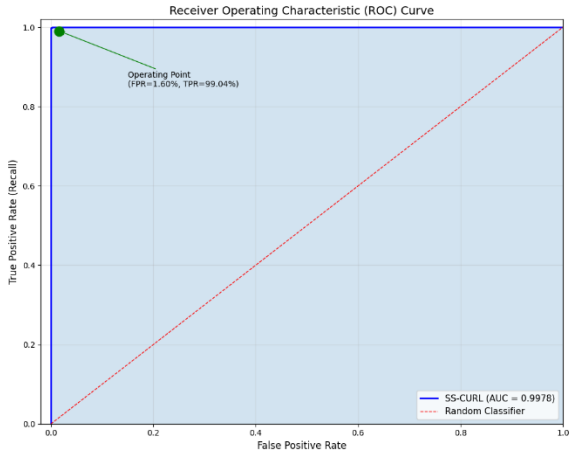

Figure 5: ROC Curve

Figure 5 shows the ROC curve, which shows the trade-offs between the true positive rate and the false positive rate at different classification thresholds. The curve closely follows the upper-left corner, which means that even at low false positive rate thresholds, the true positive rates are very high. The AUC of 99.78% is very close to the highest possible value. The operating point receives a false positive rate of 1.60% and a true positive rate of 99.04% at threshold 0.5.

6.3 Experiment 3: Classification Threshold Analysis

Reduction of the false positive rate versus maximization of the detection rate might be emphasized in distinctive security scenarios across the operational deployment wherein the importance of the relationship between the classification probability threshold and the performance metrics should be understood clearly. It has been explored in the third experiment wherein the variation in the performance metrics based upon the threshold values ranging from 0.3 to 0.9 is shown in Figure 6.

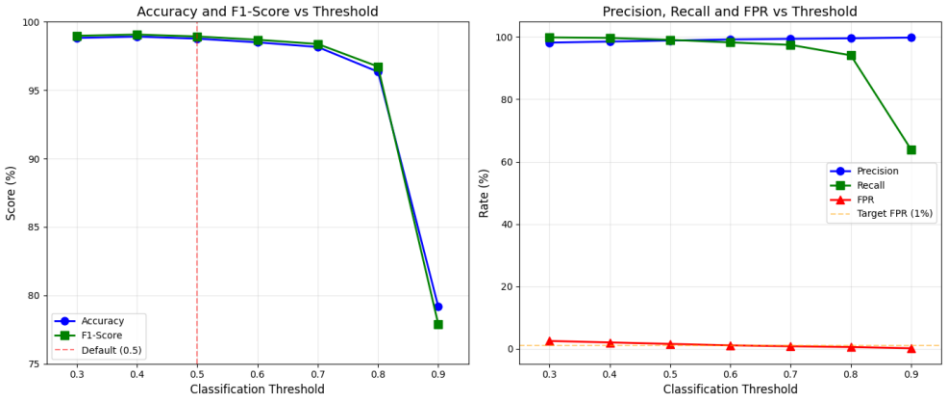

Figure 6 Analysis

The results shown above indicate a varying trade-off level. A model reaches a maximum level of recall of above 99.8% for a low threshold of 0.3 to 0.4. However, this model has a 2.5%

false positive rate. The F1-score reaches a maximum level of 99.07% for a threshold of 0.4. This shows a perfect trade-off between precision and recall. At a threshold of 0.5, the default one, recall remains high (99.04%), but the false positive rates declines to 1.60%. As thresholds increase, more accurate results with fewer false positives are achieved. This indicates that for a lower FPR of 0.21%, the recall declines to 63.90% when using a 0.9 threshold.

A false positive rate of 1% and a total recall rate of more than 98% for a threshold of 0.6. This might be a better trade-off for an environment with a very high level of security where false positive cost might be tolerable but where phishing detection remains a strong concern. A trade-off curve allows a selection of thresholds depending on requirements and risk tolerance.

6.4 Experiment 4: Adversarial Robustness Evaluation

The fourth experiment examines the capability of SS-CURL in resisting evasion attacks. Adversarial robustness is assessed using Projected Gradient Descent attacks with perturbation values (epsilon) of 0.1, 0.2, and 0.3 for intermediate representations. This addresses Objectives 4 with regard to system resilience when interacting with adaptive attackers. Figure 7 shows how well the classification works when there is an attack compared to when there is not.

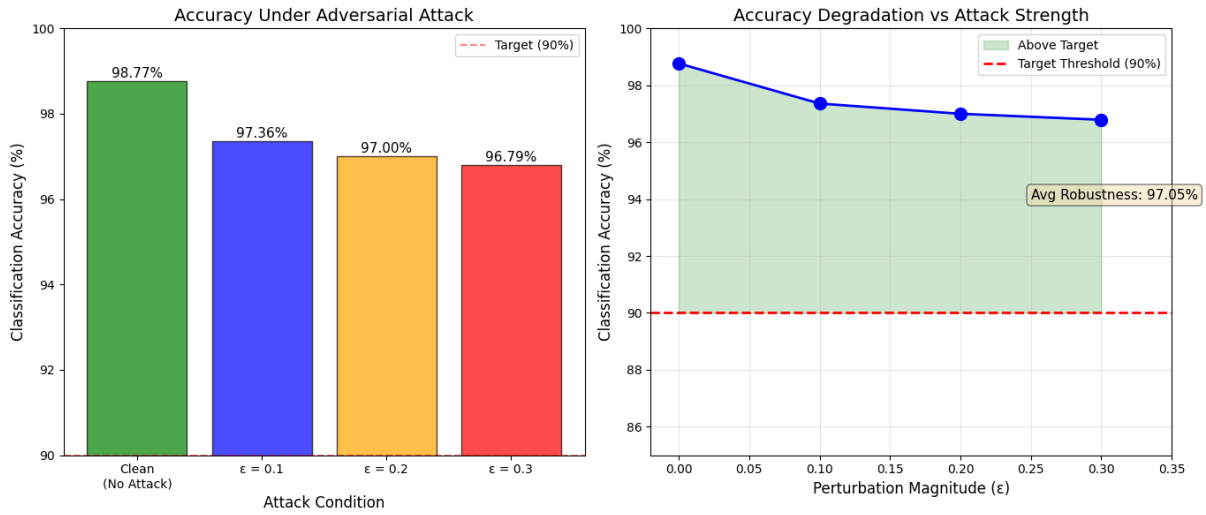

Figure 7: Adversarial Robustness Evaluation

On a clean evaluation, the model gets 98.77% accuracy. When PGD attacks happen with epsilon 0.1, accuracy drops by 1.41 percentage points to 97.36%. Accuracy goes down to 97.00% (1.77 percentage points lower) when epsilon is 0.2. The strongest attack at epsilon 0.3 lowers accuracy to 96.79%, which is a drop of 1.98 percentage points.

The average adversarial accuracy across all attack intensities is 97.05%, which is much higher than the 90% goal set in Objective 4. The gracefully degrading pattern with a decline in accuracy linearly with a larger perturbation indicates that a model does not possess vulnerable representations likely to degrade in a catastrophic manner when the perturbation occurs. An adversarial training approach has produced robust representations with a continued

discriminative power with changes in input.

Figure 8 shows the comparative analysis regarding the robustness of SS-CURL against degradation, as reported in the earlier studies. While the earlier studies reported the decay in the value of accuracy by thirty to fifty percent in the case of adversarial attacks (Kulkarni et al., 2024), the maximum decay in SS-CURL in terms of percentage is only 1.98, showing approximately a fifteen-fold improvement in terms of adversarial robustness validating the inclusion of adversarial training in the contrastive learning framework.

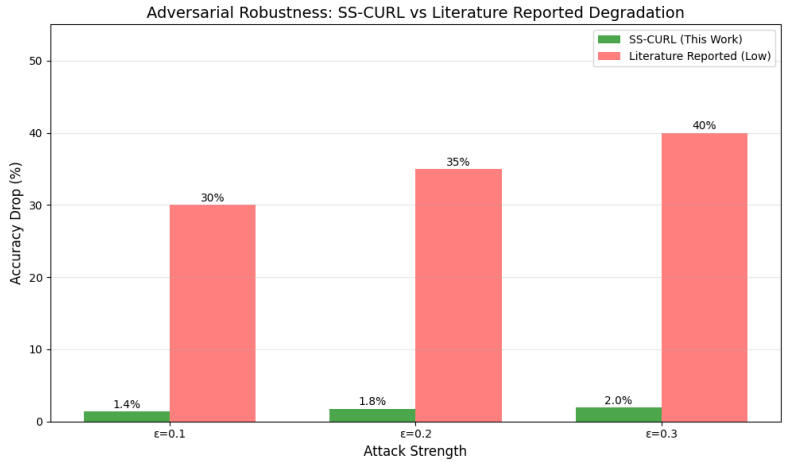

Figure 8 Comparative Robustness Analysis

6.5 Discussion

The results of the experiments provide the effectiveness of the SS-CURL framework in phishing URL detection while also pointing out the need to explore the respective components and their contribution to the framework through thorough investigation.

The classification performance of SS-CURL corresponds with and enhances findings from prior transformer-based phishing detection research. The F1-score of 98.93% is similar to what Su and Su (2023) found with BERT-based detection (approximately 99%) and what Prasad and Chandra (2024) found with the PhiUSIIL framework (99.24%). The AUC of 99.78% is in the same range as the AUCs of PhishBERT (Wang et al., 2023) and URLTran (Maneriker et al., 2022), both of which are over 99%. These similar results show that self-supervised contrastive pre-training does not hurt the accuracy of classification while improving the quality of representation and the ability to handle adversarial situations. The quick convergence of contrastive loss during pre-training, which dropped by 98.3% over five epochs, shows that the URL augmentation strategy works well for making useful positive and negative pairs for efficient representation learning.

The adversarial robustness results are the most important new information compared to what is already known. Kulkarni et al. (2024) reported a thirty to fifty percent drop in detection accuracy when models were attacked, and Apruzzese et al. (2024) showed that performance dropped even more against models that were trained to be attacked. SS-CURL, on the other hand, keeps an average accuracy of 97.05% under attack, with a maximum drop of only 1.98 percentage points. This improvement further ensures the integration of adversarial training in contrastive learning. This smooth degrading curve indicates that SS-CURL has not learned

susceptible features that can be altered. This addresses one of the questions posed by Shirazi et al. (2022) about vulnerable points of feature-based methods.

Despite this being a fairly good level of performance, several issues must be pointed out. The false positive rate for 1.60%, at a default threshold exceeds 1% target. This could prove to be a concern for environments with a lot of traffic where even a modest false positive rate would be considered a problem for analysts. Analysis of thresholds indicates that in order to attain a 1% false positive rate, one would be forced to increase this threshold to about 0.6. This would give a decrease of recall of 98.22%. This indicates that a level of trade-off exists between precision and this level of recall where optimal operating points depend on deployment context and organisational risk tolerance. The evaluation used a single dataset; while PhiUSIIL is a comprehensive and contemporary compilation, extrapolation to URLs from different sources or temporal contexts requires additional validation through cross-dataset evaluation.

7 Conclusion and Future Work

This study tested whether self-supervised contrastive learning and adversarial augmentation can be effectively used for robust phishing URL detection, including zero-day attacks, with a reduced need for extensive labelled datasets. Four tasks were considered: to develop URL-specific methods for data augmentation to model phishing obfuscation techniques; to propose a training approach using adversarial contrastive learning; to demonstrate how effective self-supervised pre-training could be; and to verify its effectiveness for defending adaptive attackers.

There has been development and validation of the SS-CURL framework, which has aided research purposes extensively. For Objective 1, a comprehensive taxonomy of augmentation has been developed with eleven transformation methods ranging from innocuous variations to attack-based augmentations such as typosquatting attacks, Homograph Attacks, and SubDomain Spoofing. A unified training pipeline comprising pre-training with NT-Xent contrastive loss and fine-tuning with Projected Gradient Descent adversarial training has aided the satisfaction of Objective 2. The experimental outcome validated Objective 3. This was because contrastive pre-training resulted in a 98.3% reduction in loss over five epochs. Secondly, with fine-tuning, a 98.93% F1-score was achieved. For objective 4 validation, adversarial robustness assessment indicated a 97.05% average accuracy when under attack. This was far above the target threshold of ninety percent. The two experiments validate that self-supervised contrastive learning pre-training of a phishing URL detector performs well. Also, by incorporating adversarial training, this detector performs about fifteen times better than a standard detector.

Both the academic and practical communities will be impacted by this study. This work suggests that contrastive learning objectives can be applied to tasks involving URL analysis. This means that such approaches can be applied to similar cybersecurity tasks such as malware detection and analysing network intrusions. The successful integration of adversarial training with contrastive learning methods results in a model for making machine learning systems more resilient in scenarios where security counts. For practitioners, the SS-CURL model offers a way to launch robust phishing detection systems against effective evasion attacks, thereby lessening the threat of security breaches due to adversarial manipulation of the detection infrastructure.

The results are deficient for several reasons. Although assessment using one data set is comprehensive, this assessment has to be checked using other data sets because more comprehensive proof of generalization capability would be possible this way. The 1.60% false positive rate might call for a different threshold to be used when a large number of users are involved. A transformer-based encoding might be considered challenging to deploy regarding computational requirements. The adversarial assessment considered modification to representation space without considering modification to problem space that alter URL strings.

The subsequent studies should be able to address the above disadvantages while, in turn, making effective use of the same framework for the purpose of deploy ability. The comparative study based on different datasets for phishing attacks will be able to provide an insight into the issue of generalization. A multi-modal approach involving URL assessments and assessments such as those involving verification of content, verification of certifications, and visual similarity will be able to explore some new dimensions in identifying sophisticated attacks with a genuine URL structure. Lightweight design of architectures for the encoders will be the most effective design tool in the real-time environment to guard edges without the need for the centralized inference service. Federated learning methods will assist in the coming together of organizations for effective model design while securing their threat intelligence. At last, the explainability methods will assist in the utilization of each other by analysts in understanding the activation of the rules of detection. This will assist in enabling enhanced human and machine collaboration in security operation work. The commercialisation potential is evident in the application for the incorporation of SS-CURL as the core component into enterprise security solutions, browser add-ons, and email gateway solutions.

References

Alshingiti, Z., Alaqel, R., Al-Muhtadi, J., Haq, Q.E.U., Jalal, K. and Olatunji, S.O. (2023) 'A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN', *Electronics*, 12(1), p. 232. doi: 10.3390/electronics12010232

Apruzzese, G., Conti, M. and Yuan, Y. (2024) 'Multi-SpacePhish: Extending the Evasion-space of Adversarial Attacks against Phishing Website Detectors Using Machine Learning', *ACM Transactions on Privacy and Security*. doi: 10.1145/3638253

Asiri, S., Xiao, Y., Alzahrani, S., Li, S. and Li, T. (2024) 'PhishingRTDS: A real-time detection system for phishing attacks using a Deep Learning model', *Computers & Security*, 118, p. 103374. doi: 10.1016/j.cose.2024.103843

Catal, C., Giray, G., Tekinerdogan, B., Kumar, S. and Shukla, S. (2022) 'Applications of deep learning for phishing detection: a systematic literature review', *Knowledge and Information Systems*, 64, pp. 1457-1500. doi: 10.1007/s10115-022-01672-x

Do, N.Q., Selamat, A., Krejcar, O., Fujita, H. and Omatu, S. (2024) 'An integrated model based on deep learning classifiers and pre-trained transformer for phishing URL detection', *Future Generation Computer Systems*, 161, pp. 345-360. doi: 10.1016/j.future.2024.06.031

Ghalechyan, H., Israyelyan, E., Arakelyan, A. and Hambardzumyan, A. (2024) 'Phishing URL detection with neural networks: an empirical study', *Scientific Reports*, 14, p. 25134. doi: 10.1038/s41598-024-74725-6

Guo, Y. (2023) 'A review of Machine Learning-based zero-day attack detection: Challenges and future directions', *Computer Communications*, 198, pp. 175-185. doi: 10.1016/j.comcom.2022.11.001

Haq, Q.E.U., Faheem, M.H. and Ahmad, I. (2024) 'Detecting Phishing URLs Based on a Deep Learning Approach to Prevent Cyber-Attacks', *Applied Sciences*, 14(22), p. 10086. doi: 10.3390/app142210086

Kulkarni, A., Moti, Z., Baki, S. and Peng, W. (2024) 'From ML to LLM: Evaluating the Robustness of Phishing Webpage Detection Models against Adversarial Attacks', *arXiv preprint arXiv:2407.20361*. doi: 10.48550/arXiv.2407.20361

Lee, J., Park, J., Lee, T. and Kim, J. (2023) 'Attacking Logo-Based Phishing Website Detectors with Adversarial Perturbations', in *European Symposium on Research in Computer Security (ESORICS)*. Springer, pp. 162-180. doi: 10.1007/978-3-031-51479-1_9

Li, W., Manickam, S., Chong, Y.W., Tao, H. and Zheng, J. (2024) 'A State-of-the-Art Review on Phishing Website Detection Techniques', *IEEE Access*, 12, pp. 187976-188012. doi: 10.1109/ACCESS.2024.3506889

Maneriker, P., Stokes, J.W., Lazerow, E.G., Gethers, A., Consul, S., Dimakis, S. and Odom, J. (2022) 'URLTran: Improving Phishing URL Detection Using Transformers', *arXiv preprint arXiv:2106.05256*. doi: 10.48550/arXiv.2106.05256

Otieno, D.O., Abri, F., Namin, A.S. and Jones, K.S. (2023) 'Detecting Phishing URLs using the BERT Transformer Model', in *IEEE International Conference on Big Data*. IEEE, pp. 2483-2492. doi: 10.1109/BigData59044.2023.10386782

Ozcan, A., Catal, C., Donmez, E. and Senturk, B. (2023) 'A hybrid DNN–LSTM model for detecting phishing URLs', *Neural Computing and Applications*, 35, pp. 4957-4973. doi: 10.1007/s00521-021-06401-z

Prasad, A. and Chandra, S. (2024) 'PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning', *Computers & Security*, 136, p. 103545. doi: 10.1016/j.cose.2023.103545

Shirazi, H., Bezawada, B., Ray, I. and Anderson, C. (2022) 'Adversarial Sampling Attacks Against Phishing Detection', in *Data and Applications Security and Privacy*. Springer, pp. 83-101. doi: 10.1007/978-3-030-22479-0_5

Su, M.Y. and Su, K.L. (2023) 'BERT-Based Approaches to Identifying Malicious URLs', *Sensors*, 23(20), p. 8499. doi: 10.3390/s23208499

Tamal, M.A., Islam, M.K., Bhuiyan, T. and Sattar, A. (2024) 'Dataset of suspicious phishing URL detection', *Frontiers in Computer Science*, 6, p. 1308634. doi: 10.3389/fcomp.2024.1308634

Wang, F., Qi, S., Zhang, G., Wang, Z., Qiu, J. and Huang, H. (2023) 'Self-supervised contrastive representation learning for IoT malware classification', *Engineering Applications of Artificial Intelligence*. doi: 10.1016/j.engappai.2025.110299

Wang, M., Zang, X., Cao, J., Liu, Y. and Li, Z. (2024) 'PhishHunter: Detecting camouflaged IDN-based phishing attacks via Siamese neural network', *Computers & Security*, 138, p. 103668. doi: 10.1016/j.cose.2023.103668

Wang, Y., Zhu, W., Xu, H., Huang, S., Li, Z. and Li, J. (2023) 'A Large-Scale Pretrained Deep Model for Phishing URL Detection', in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 1-5. doi: 10.1109/ICASSP49357.2023.10095719.
Yuan, Y., Hao, Q., Apruzzese, G., Conti, M. and Wang, G. (2024) 'Are Adversarial Phishing Webpages a Threat in Reality?', in *Proceedings of the ACM Web Conference (WWW)*. Singapore: ACM. doi: 10.1145/3589334.3645502