PhishClip: CLIP-based Prompt-tuned Phishing Site Screenshot Classifier

Hankyul Jang

CS371 Final Report, Spring 2025

1 Introduction

Phishing websites visually imitate trusted login pages to steal user credentials, often targeting non-technical or digitally vulnerable individuals. Traditional detection methods, such as rule-based URL filters, struggle to identify sophisticated visual mimicry or redirection tactics.

To address this, I propose **PhishClip**, a phishing screenshot classifier that leverages CLIP (Contrastive Language-Image Pretraining) and CoOp (Context Optimization). The model employs soft prompt tuning to improve classification performance in few-shot scenarios. Its effectiveness is evaluated on the Phishpedia benchmark dataset and compared against two baselines: zero-shot CLIP and the original Phishpedia model.

This project contributes to the United Nations Sustainable Development Goal 10 (Reduced Inequalities) by providing accessible phishing detection tools aimed at protecting digitally underserved populations. In doing so, the project also explores the application of vision-language models in adversarial and real-world security contexts.

2 Methods

Dataset: The Phishpedia benchmark dataset contains over 30,000 phishing and benign webpage screenshots, each with associated domain metadata and labels. The data is split into 80% training and 20% testing sets.

Baselines:

- **Zero-shot CLIP**: Manual prompts such as "benign site of [domain]" are encoded via CLIP and used for direct comparison.
- **Phishpedia**: Combines Fast RNN for logo detection and ResNet18 for logo-brand classification, followed by domain matching.

Proposed Method (CoOp): The CoOp framework is implemented to learn soft prompts specific to each class label, enhancing CLIP's alignment with phishing semantics. The model is trained for 5 epochs, with the entire dataset randomly reshuffled at each epoch. This design allows the classifier to internalize subtle design cues in phishing screenshots while remaining lightweight and sample-efficient.

Training Details: Each class label is converted into a text prompt (e.g., "benign site of [domain]"), tokenized, and passed through the CLIP text encoder. For each class, the mean of a learned soft prompt tensor is added to the corresponding text embedding. Image features and text features are normalized and compared via dot product to produce logits. These logits are trained against ground truth using cross-entropy loss. Prompt parameters are initialized as random tensors of shape (C, L, D), where C is the number of classes, L the prompt length, and

D the CLIP text embedding dimension. Optimization is performed using Adam with a learning rate of 10^{-3}.

Environment: All experiments are conducted on Ubuntu 22.04.3 LTS with Python 3.10.12, PyTorch 2.7.0+cu118, CLIP 1.0, and an NVIDIA RTX 3060 Ti GPU.

3 Experiments

The evaluation uses accuracy, precision, recall, F1-score, mean confidence, and confusion matrix. All models are tested on the same 20% test split from the Phishpedia dataset.

Quantitative Results:

Model	Accuracy	Mean Conf	Precision	Recall	F1-score
CLIP	0.511	0.512	0.500	**0.976**	0.662
CoOp_1	0.884	0.850	0.878	0.887	0.882
CoOp_2	0.895	0.871	0.915	0.866	0.890
CoOp_3	0.902	0.880	0.909	0.888	0.899
CoOp_4	0.904	0.884	0.905	0.897	0.901
CoOp_5	**0.907**	**0.890**	**0.918**	0.889	**0.903**
Phishpedia	0.893	0.804	**0.977**	0.817	0.890

Table 1: Performance metrics for all evaluated models. CoOp improves steadily and outperforms both CLIP and Phishpedia by the final epoch.

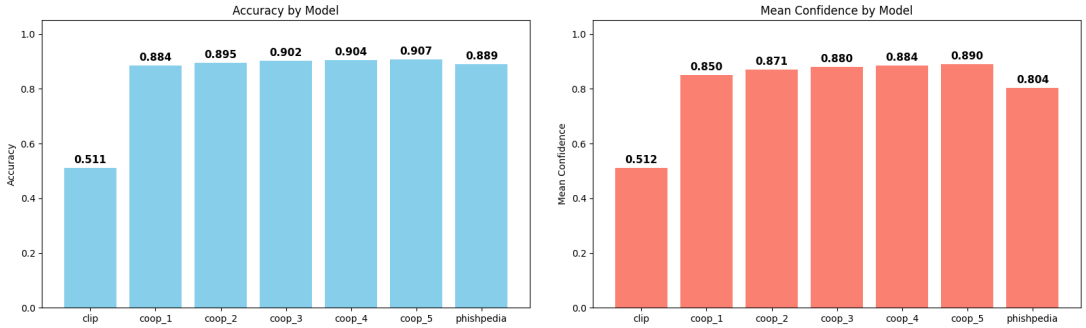
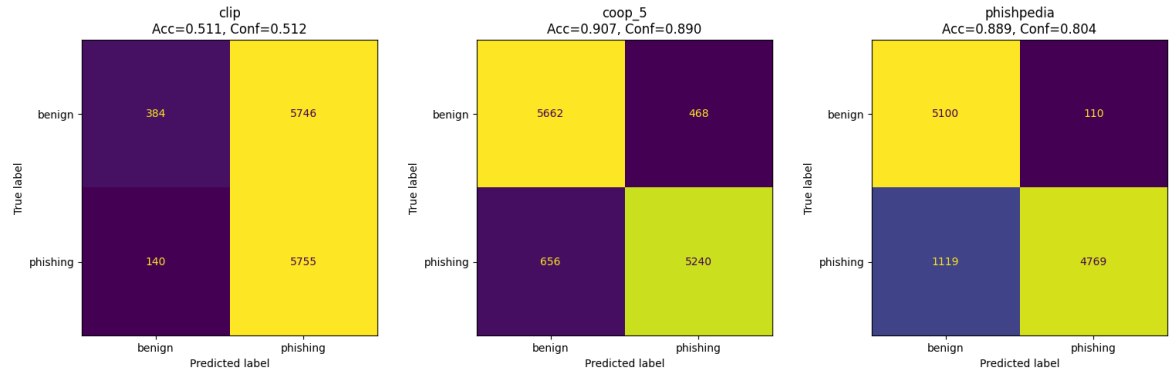


Figure 1: Left: Accuracy by model. Right: Mean confidence score by model.

Confusion Matrix:



(a) CLIP (zero-shot) (b) CoOp (epoch 5) (c) Phishpedia

Figure 2: Confusion matrix comparison across models.

4 Discussion

The results indicate that CoOp improves phishing detection by capturing domain-specific semantics through learned prompts. This method demonstrates better generalization and fewer false negatives than both zero-shot CLIP and Phishpedia. Notably, CoOp is more sensitive to phishing-specific design patterns such as subtle layout inconsistencies or logo anomalies.

In contrast, zero-shot CLIP was highly sensitive to prompt formulation and showed inconsistent behavior across domains. Although Phishpedia provided high precision, it often misclassified phishing pages as benign, which is especially dangerous in a real-world deployment.

Soft prompt tuning appears to produce clearer decision boundaries, as evidenced by confidence margins and confusion matrices. These findings support the idea that lightweight prompt adaptation can yield strong downstream task performance.

Nevertheless, limitations exist. The dataset does not fully cover adversarial threats such as CAPTCHA evasion or rapid brand spoofing. Additionally, robustness under real-time deployment and against unseen phishing strategies remains untested.

5 Conclusion & Future Work

PhishClip demonstrates that CoOp-based prompt tuning enhances phishing screenshot classification. By tuning soft prompts, the model internalizes phishing-specific cues, improving both accuracy and robustness.

Future directions include integrating dynamic prompt optimization methods such as reinforcement learning, adding real-time detection pipelines, and extending the system to multilingual or low-resource environments. These efforts may further improve generalization and deployment feasibility.

References

- Radford et al. "Learning Transferable Visual Models from Natural Language Supervision." ICML 2021.
- Zhang et al. "Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages." USENIX Security 2021.
- Zhou et al. "Learning to Prompt for Vision-Language Models." CVPR 2022.

Appendix

CoOp Training Code (Excerpt):

```
# Excerpt from CoOp_train.py
soft_prompts = nn.Parameter(torch.randn(num_classes, prompt_len, embed_dim, device="cuda"))
...
for epoch in range(5):
    random.shuffle(samples_subset)
    for idx, sample in enumerate(samples_subset):
        ...
        prompts = [make_prompt(lbl, domain) for lbl in label_map.keys()]
        tokens = clip.tokenize(prompts).cuda()
        text_feats = clip_model.encode_text(tokens)
        ...
        logits = (img_embed @ text_feats.T)
```

```
        loss = nn.CrossEntropyLoss()(logits, target)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Dataset Example (train_dataset.json):

```
{
    "img_path": "datasets/benign_sample_30k/google.com/shot.png",
    "label": "benign",
    "domain": "google.com"
}
```

Sample Screenshot:

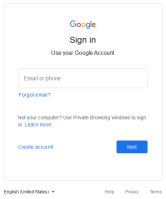


Figure 3: Sample screenshot included in the training dataset.

Dataset Source:
Phishpedia Benchmark Dataset (public):
`https://sites.google.com/view/phishpedia-site/`