CS 536 Project: Identifying Phishing Sites by Screenshots

Daniel Boehm (djb370)
Alex Karlovitz (amk371)
Patrick Meng (pm708)

Abstract

Visual similarity approaches to phishing detection attempt to identify illegitimate sites solely by looking at screenshots of websites. In this paper, we propose two architectures for solving this problem. The first uses attention-augmented convolutional neural nets, and the second uses a Siamese neural net with a threshold.

1 Introduction

Last year, the most common cybercrime in the United States was phishing attacks with more than 240,000 known offences [12]. Phishing attacks are when individuals pretend to be an organisation or company in order to trick users into providing their personal data or buying a fake product. In the attempt to conduct phishing scams, scammers often build websites that are visually similar to the companies they are mimicking. Sometimes they go to great lengths to provide high quality, realistic images in order to increase their chances of tricking users who come upon their websites. In this project, we propose two new architectures for identifying phishing sites from screenshots.

2 Related Works

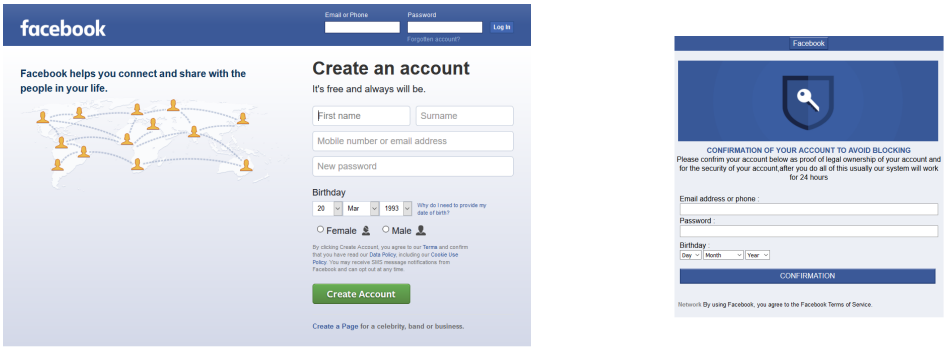
Phishing sites have been around for a long time, so there has been a lot of research in this area. Early methods focused on page-based approaches. For example. Huang et al. [8], Zhang et al. [14], and Liu et al. [11] all studied different features from HTML files, looking for indicators of phishiness.

Other methods opt for visual similarity approaches; these look at screenshots of a webpage to determine phishiness. In the last 10-15 years, CNNs have improved results by detecting logos (Chang et al. [3] and Dunlop et al. [5]), identifying phishy URLs (Woodbridge et al. [13]), or even matching on entire screenshots as done in the 2020 algorithm VisualPhishNet [1].

In this paper, we will present two new architectures for the visual similarity approach. The first is a CNN modified to use attention, inspired by Jetley et al. [9], who used CNNs modified with attention to increase preformance on the CIFAR classification data sets. The second architecture uses Siamese Neural Networks (SNNs) with a threshold. SNNs were first introduced in 1993 [2] for the purposes of signature verification. More recently, SNNs have been used with deep neural networks to solve a variety of classification tasks. In one interesting example, Koch [10] showed they perform well for one-shot image recognition on handwriting.

3 Experiments

In this project, we use the Phish-IRIS Dataset [4], which is composed of screenshots labeled with the company they are impersonating. The company categories are: Adobe, Alibaba, Amazon, Apple, Bank of America, Chase, DHL, Dropbox, Facebook, LinkedIn, Microsoft, Paypal, Wells Fargo, and Yahoo. In addition to these categories, there is also an "other" category that contains screenshots of legitimate websites.



(a) (b)

Figure 1: Two phishing sites impersonating facebook. The screenshot in (a) looks almost identical to the true login page. In contrast, the image in (b) looks phishy to anyone familiar with facebook.

Each algorithm is trained with labeled images. They are then given unlabeled images so that they may predict labels during the validation and testing stages. Ideally the predicted label for each screenshot matches the true label for that screenshot.

3.1 Inductive Bias for using CNNs with Phishing Sites

To perform well on classification problems, CNNs need to find features which are similar in a single class but different across distinct classes. The existence of such features is therefore an assumption made when using our algorithms. In the case of phishing sites, we find that this is a reasonable assumption, since websites attempting to impersonate a company online will attempt to look visually similar to that company's true website. See Figure 1(a) for an example.

On the other hand, one challenging aspect of this problem is the fact that phishing sites can potentially look quite different from any true site they are impersonating. See Figure 1(b) for an example. This suggests that any architecture needs to learn specifically those features which are consistent across these impersonation attempts.

4 Attention-Augmented CNNs

4.1 Architecture

We implemented the architecture described in [9], and shown in figure 2.

Inspired by VGGNet, the attention augmented CNN network saves the feature maps from the 7th, 10th, and 13th layers. The output of the first fully connected layer is then used to query attention on the three feature maps. The outputs are then concatenated and passed into a final fully connected layer, with one output node per class. If any class other than "other" is selected, the screenshot is declared a phishing site.

The network was trained and used in standard classification convention. Cross Entropy loss was used to train the network, and node of maximum value was used as the network prediction.

4.2 Training

First, the Phish-Iris Dataset was split into training, validation, and testing data, with a ratio of around 2-1-1.

The learning rate, optimization algorithm, and length of training were tuned from performance on the validation set, while the model trained from the training set.

We trained two CNN classifiers to compare. The first was the proposed attention-augmented architecture in section 4.1. The second is the equivalent model without attention, as depicted by the top row of figure 2, using the crossed out fully connected layer.

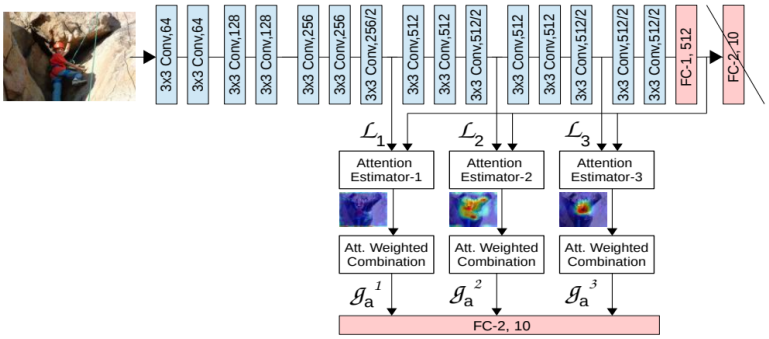


Figure 2: VGG-att3-concat-pc from Learn To Pay Attention. Note that the referenced architecture was designed for 10 classes, while this paper proposes a 15-class architecture.

The final version of both models were trained using ADAM, with a learning rate of 1e-4, for 20 epochs of training.

Once both models were fully trained and had optimized performance on the validation set, they were evaluated and compared using the test set. To preserve confidence in the results, no further changes were made after this evaluation.

4.3 Evaluation

At various points in the pipeline, models predicted an evaluation set (val for training, test for final), and their predictive performance was evaluated using a binary confusion matrix of model predictions versus data labels. From these binary confusion matrices, we specifically viewed 6 metrics of performance: Accuracy, Precision, True Positive Rate, False Negative Rate, True Negative Rate, and False Positive Rate. These scores are compared to diagnose exact predictive behavior of each model. See appendix A for how we define the listed metrics.

4.4 Results On Test Set

4.4.1 Phishing Detection

In Table 1, we see that the attention model has moderately better accuracy, precision, and true negative rate. However, the recall is significantly improved with attention. There is a 10% greater chance that the attention model will notice a phishing site than the vanilla CNN. Overall, the attention model is more powerful and accurate on the validation and test sets.

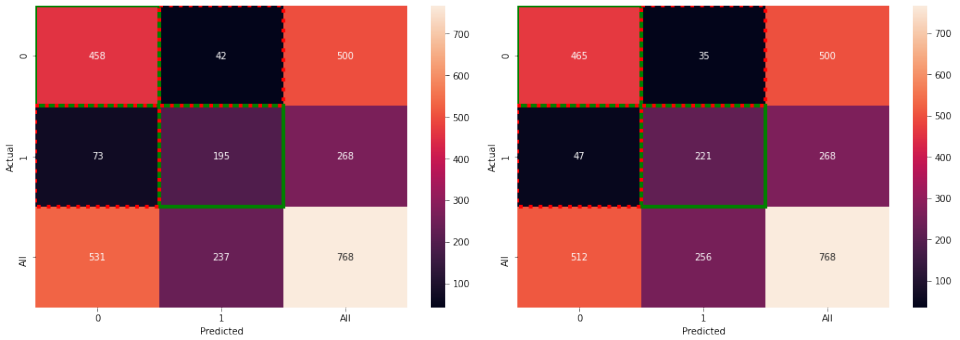
The confusion matrices from which these metrics were computed are shown in figure 3.

Metric	Without Attention	With Attention
Accuracy	85.026%	89.323%
Precision	82.278%	86.328%
Recall = True Positive Rate	72.761%	82.463%
False Negative Rate	27.239%	17.537%
True Negative Rate	91.600%	93.000%
False Positive Rate	08.400%	07.000%

Table 1: Performance Metrics for CNN Classifiers with and without Attention

4.4.2 Performance for each Company

The confusion matrices in the previous section, and the associated metrics, result from simplifying the model outputs to **Phishing (1)** or **Not Phishing (0)**. The models are actually predicting which one of 15 classes, 14 companies and other, the screenshot is. In figure 5, under appendix B, we can see how often the models correctly and incorrectly predicted each class.



(a) Without Attention (b) With Attention

Figure 3: Confusion Matrices for Model Prediction of Test Set

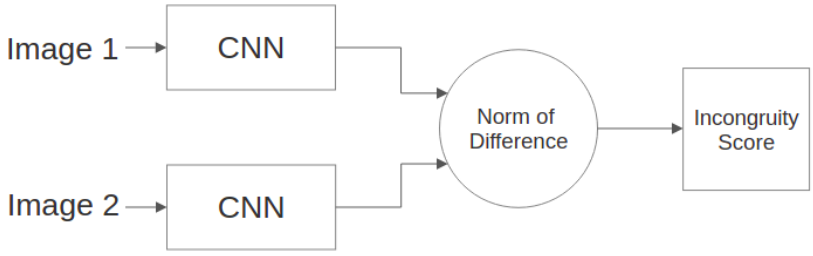


Figure 4: A Siamese Neural Network: two images are fed into identical CNNs; the output feature vectors are compared by taking the norm of the difference; optionally, the final incongruity score can be any monotonic function of the difference.

For instance, both models perform very well when the image shows a site imitating Bank of America (boa), or Yahoo. The attention model is significantly better at identifying Paypal screenshots. Both do very poorly on LinkedIn, with worse performance than a 50/50 coin flip.

5 Siamese Neural Network with Threshold

A Siamese Neural Network (SNN) attempts to learn an incongruity score function to determine whether two input images belong to the same class. In essence, this can be described as a binary classification problem where the input is a pair of images, and the output is either *match* or *not a match*. The general architecture is shown in Figure 4.

The key idea for SNNs is the use of identical convolutional neural networks with shared weights. Two input images are each fed into one of these "twin" CNNs, producing a feature vector for each image. Training encourages the network to learn features which will be similar when input images come from the same class, but very different when images come from different classes. The SNN then takes the difference of the feature vectors and computes a norm (e.g. L^1 or L^2). This produces an incongruity score in its own right, since more similar feature vectors will produce a smaller norm in the difference.

Once an incongruity score is obtained, we can predict whether the images are a match or not by comparing the score to a threshold. The threshold is a hyperparameter which should be tuned before testing. Optionally, one may wish to apply a monotonic function to the norm of the difference to produce a different incongruity score. For example, an activation function like a sigmoid whose range is in $[0, 1]$ could convert the incongruity score into a probability.

5.1 Contrastive Loss Function

Despite the architecture leading to a binary output, we did not use binary cross entropy for the loss function. Instead, we used a contrastive loss function, which was originally introduced in [6] to improve algorithms for learning dimensionality reductions. Such a function is one of the form

$$L(S, Y) = (1 - Y)L_0(S) + YL_1(S)$$

where S is the incongruity score obtained from a pair of inputs, and Y is the true label (0 if they came from the same class, 1 otherwise). L_0 can be any monotonically increasing function, and L_1 can be any monotonically decreasing function, so long as both functions have nonnegative range. We chose the functions

$$L_0(S) = S^2 \qquad L_1(S) = \max\{0, 3.2 - S\}$$

after some experimentation. Contrastive loss functions have been found by various groups (e.g. [13]) to perform better than binary cross entropy when training SNNs.

5.2 Final Architecture and Results

For the final architecture, we took the twin CNNs to be ResNet50 [7], followed by two fully connected layers reducing the feature vectors from 1000 dimensions to 500, then 500 to 32. The ResNet50 block was initialized using the weights from pre-training on the ImageNet data set.

We tuned the threshold hyperparameter during the validation phase. We were interested in four metrics:

1. overall accuracy: were screenshots correctly labeled as non-phishing or, for phishing sites, were they labeled with the correct company they targeted?
2. false negative rate: how many phishing sites were labeled as legitimate?
3. false positive rate: how many legitimate sites were labeled as phishing?
4. wrong company rate: how many phishing sites were correctly labeled as phishing, but the company they were targeting was wrongly identified?

Depending on the use case, we identified three values for the threshold. A threshold of 0.04 harshly penalizes false positives, and a threshold of 0.46 harshly penalizes false negatives, although both thresholds maintained a relatively high overall accuracy on the validation set. Finally, a threshold of 0.13 gave the best performance on the validation set in terms of overall accuracy.

The results of this architecture with the three different thresholds are shown in the table below.

Threshold	Accuracy	False Negative	False Positive	Wrong Company
0.04	81.07%	17.60%	1.20%	0.13%
0.13	87.07%	9.20%	3.33%	0.40%
0.46	79.87%	4.27%	15.33%	0.54%

6 Conclusion

The architectures used in this project were designed to have a direct application. If implemented by a company, they may use this system to find websites that are attempting to impersonate their own sites. Both neural networks could be used to scan a large number of websites. By evaluating overlap, the algorithms could compensate for websites missed by one or the other. At highest validation accuracy, the Attention model is less likely to produce a false negative, and the SNN model is more likely to correctly identify the exact company imitated. Sites that are found suspicious by either network would be reported to an employee who could manually evaluate the page. If the employee deems that the phishing site is damaging to the company they work for, the employee could then report the site and request for its removal.

7 Contribution

Dan contributed to the implementation, training, and evaluation of the attention-augmented CNN. Alex and Patrick contributed to the implementation, training, and evaluation of the Siamese neural network with threshold. All three authors met regularly to discuss technical details and ideas.

References

[1] Sahar Abdelnabi, Katharina Krombholz, and Mario Fritz. Visualphishnet: Zero-day phishing website detection by visual similarity. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1681–1698, 2020.

[2] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6:737–744, 1993.

[3] Ee Hung Chang, Kang Leng Chiew, Wei King Tiong, et al. Phishing detection via identification of website identity. In *2013 international conference on IT convergence and security (ICITCS)*, pages 1–4. IEEE, 2013.

[4] Firat Coskun Dalgic, Ahmet Selman Bozkir, and Murat Aydos. Phish-iris: A new approach for vision based brand prediction of phishing web pages via compact visual descriptors. In *2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–8. IEEE, 2018.

[5] Matthew Dunlop, Stephen Groat, and David Shelly. Goldphish: Using images for content-based phishing analysis. In *2010 Fifth international conference on internet monitoring and protection*, pages 123–128. IEEE, 2010.

[6] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Chun-Ying Huang, Shang-Pin Ma, Wei-Lin Yeh, Chia-Yi Lin, and Chien-Tsung Liu. Mitigate web phishing using site signatures. In *TENCON 2010-2010 IEEE Region 10 Conference*, pages 803–808. IEEE, 2010.

[9] Saumya Jetley, Nicholas A. Lord, Namhoon Lee, and Philip H. S. Torr. Learn to pay attention, 2018.

[10] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[11] Wenyin Liu, Xiaotie Deng, Guanglin Huang, and Anthony Y Fu. An antiphishing strategy based on visual similarity assessment. *IEEE Internet Computing*, 10(2):58–65, 2006.

[12] Maddie Rosenthal. Phishing statistics (updated 2021): 50+ important phishing stats.

[13] Jonathan Woodbridge, Hyrum S Anderson, Anjum Ahuja, and Daniel Grant. Detecting homoglyph attacks with a siamese neural network. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 22–28. IEEE, 2018.

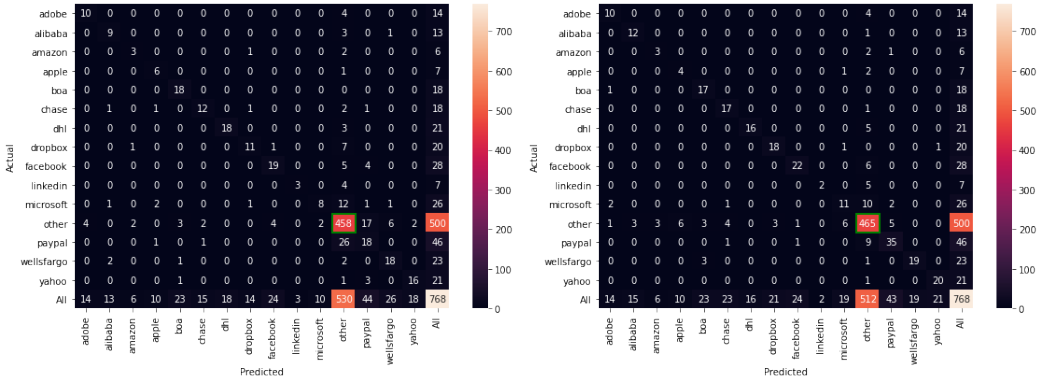
[14] Yue Zhang, Jason I Hong, and Lorrie F Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 639–648, 2007.

Appendices

A Evaluation Metrics for CNN Classifiers

- Accuracy: Correct/Total, how often did the model correctly classify screenshots?
- Precision: True Positives/Positive Predictions, how often was the model correct when predicting a phishing site?
- Recall, or True Positive Rate: True Positives/Positive Labels, how often did the model notice when a screenshot showed a phishing site?
- False Negative Rate: False Negatives/Positive Labels, how often did the model claim a phishing site was safe?
- True Negative Rate: True Negatives/Negative Labels, how often did the model notice when a screenshot showed a safe site?
- False Positive Rate: False Positives/Negative Labels, how often did the model claim a safe site was phishing?

B Expanded Confusion Matrices



(a) Without Attention

(b) With Attention

Figure 5: Expanded Confusion Matrices for Model Prediction of Test Set for all 15 output classes