Robust DDoS-Attack Classification With 3D CNNs Against Adversarial Methods

Rivas DSC 4310 Project

Landon Bragg, Nate Dorsey, Josh Prior, Ben Kim, John Ajit, Nate Willis
Department of Computer Science
Baylor University
`landon_bragg1@baylor.edu, nate_dorsey1@baylor.edu, josh_prior1@baylor.edu`
`ben_kim1@baylor.edu, john_ajit1@baylor.edu, nate_willis1@baylor.edu`

1 Summary

Title	DDoS Attack Classification
Venue	NeurIPS
Year	2025

Table 1: Based on prior work by Guarino et al. (2020), Bonaci et al. (2022), Rivas et al. (2019).

Abstract

Distributed Denial-of-Service (DDoS) attacks threaten critical infrastructure in the modern world of networking systems as they can often evade traditional cyber-attack detection mechanisms through subtle traffic manipulation. To address this challenge, we propose a system using hive plot image sequences of network traffic that applies a 3D convolutional neural network (3D CNN) for robust DDoS classification. There are three parts to this project. First, we will leverage the spatial-temporal encodings of the hive plots to set a baseline for pattern recognition, secondly, we will incorporate adversarial training (FGSM and PGD) alongside augmentations (rotation, shear, zoom, noise) to strengthen the model against evasion attempts, and third, we perform a frame-wise no-context evaluation to identify the most predictive time steps for classification. Our contributions include a lightweight 3D CNN architecture specialized for sequence-based hive-plot inputs. This adversarial training regime mixes clean, augmented, and adversarial examples within each batch and several evaluation scripts that can quantify the model's robustness when faced with clean, augmented, FGSM, and PGD perturbations. Our experiments on a pre-classified benchmark dataset of DDoS and normal traffic demonstrated that adversarial retraining increases Augmented-attack accuracy from 50% to 99%, PGD accuracy from 55% to 99%, and FGSM accuracy from 55% to 93.25% all while maintaining clean data performance. We also found that time steps 3 and 4 (out of 8) yield peak classification accuracy, indicating that early detection of DDoS attacks using these models is beneficial. These findings validate that hive plot sequences are effective representations of network traffic data for early DDoS detection as they establish a reproducible, GPU-accelerated framework for adversarially robust network security.

- **GitHub:** `https://github.com/Landon-Bragg/DDoS_Attack_Classification/tree/main`

- **Datasets:** `https://baylor.box.com/s/oes1damja6l4kptauq620f8nymulk3kt`
- **Overleaf:** `https://www.overleaf.com/project/6808126fc941168a5245a6a0`

2 Introduction

Distributed denial-of-service (DDoS) attacks pose a threat to modern networking infrastructure, especially since they have grown in scale and sophistication. Malicious attackers flood servers with massive volumes of traffic to render critical networking services such as cloud platforms, financial services, and healthcare systems unavailable. Mechanisms to detect these attacks are widely outdated and struggle to distinguish attacks that are slower or deal with legitimate surges in network demand. This problem calls for a modern solution that can recognize and interpret modern networking flows of traffic. Machine learning approaches have been proposed to learn these patterns in high-dimension traffic data. Convolutional neural networks (CNNs) can be applied directly to packet time series or flow statistics where it can recognize complex correlations in the traffic data. However, these solutions face vulnerabilities from small manipulations that can cause a classifier to mislabel malicious traffic as harmless. Graph-based representations of traffic, such as hive plots, allow CNNs to recognize multi-variable relationships on a spatial layout. Creating robust models requires both a rich, expressive representation of traffic data and a training regime that explicitly hardens models against adversarial tactics. In this work, we propose to visualize sliding-window sequences of network flow features as a series of hive plot images, thereby encoding both spatial (feature correlations) and temporal (evolution over time) information in a form amenable to 3D convolution. We then train a lightweight 3D CNN end-to-end under a mixed regime that interleaves:

To fill the gaps in using the representation of traffic data as images, we must use a training regime that hardens our model against adversarial methods. In this work, we propose using a sliding-window sequence of network flow features as a series of hive plot images, encoding spatial and temporal information that ensembles to create a 3D convolution. We then trained a lightweight 3D CNN under a regime that included clean samples to preserve baseline performance, random augmentations that included rotation, shear, zoom, and Gaussian noise to simulate network variability, and adversarial attacks including FGSM and PGD to protect the model from evasion attempts. Finally, we used a frame-wise evaluation where we extracted individual frames from our 8-frame image sequence that we embedded as a 4D PyTorch tensor. Using these individual frames, we identified which temporal segments contributed most to aiding classification accuracy in hopes of using these systems for early detection. This analysis provides both insight into critical windows for DDoS detection as well as informs future works about inference strategies on minimal information.

Our contributions are these, we demonstrated that hive plot sequences serve as an effective spatio-temporal encoding of network traffic for deep learning. We harnessed an adversarial-augmentation regime to develop a batch-level training protocol combining stochastic augmentations and adversarial examples to yield robust classifiers without losing clean-traffic accuracy. We provided accuracy metrics on a per-frame basis revealing specific time steps that achieve optimal prediction accuracy.

Our experimental results on a standard DDoS dataset validate our design as we found adversarial training boosts PGD-attack accuracy from 55% to 99.5%, and timestamp analysis uncovers the most informative segments for defense. This work lays the foundation for a GPU-accelerated, adversarially robust framework for real-time DDoS mitigation in critical infrastructure.

3 Related Work

Early detection systems for DDoS attacks relied on rule-based and threshold approaches to monitor aggregate traffic volumes or flow counts to flag suspicious network traffic. While implementing these defenses is simple, these methods can generate false positives due to legitimate spikes in traffic (such as flash crowds) and cannot adapt to changing attack patterns as they become more sophisticated. Machine learning techniques have been applied to network traffic features such as packet arrival times, byte counts, and header statistics in hopes of overcoming these limitations. Guarino et al. (2020) found that convolutional neural networks trained on raw traffic time series captured temporal correlations to better detection over traditional classification methods. While they made strides in this topic, these models remain vulnerable to adversarial perturbations that can subvert the decision boundaries learned by the model.

Graph-based visual methods offer multiple-dimensional views of the network traffic relationship flows into a spatial layout. ‘Rivas et al. (2019). found that representing flows as hive plots mapping features to axes and encoding edge weights by radial distance, allowed CNNs’ spatial feature extraction capabilities to shine. This representation improves interpretability and classification performance under difficult classifying conditions. While there are benefits, research in adversarial machine learning has shown that models trained solely on clean data can be entirely misled by minimal perturbations (Goodfellow et al., 2015). Techniques such as FGSM and PGD have been used to attack and defend image classifiers, but their application to network security has not yet been investigated. CNNs have also been proven to fail when faced with even minor augmentations as found by Mikołajczyk and Grochowski (2018). This indicates the need for a model that is capable of being robust against common practices that cause these models to fail.

While research has begun on bringing adversarial training into machine learning-based detection systems, to our knowledge, no prior study has combined hive plot sequence representations with 3D CNN architectures and a mixed adversarial-augmented training protocol for DDoS classification. Our work bridges the gap of encoding temporal evolution via multi-frame hive plot sequences, hardening a lightweight 3D CNN with both stochastic data augmentations and FGSM/PGD adversarial examples, and conducting frame-wise robustness analysis to identify the most informative temporal windows for reliable detection.

4 Approach

Problem Setup

Prior work has been done towards the task of making a DDoS-attack classifier robust against adversarial augmentations and perturbations of hive-plot images of network traffic. In particular, prior work by Bonaci et al. (2022) accomplishes this task using two various CNN architectures, ResNet34 and MobileNetv2, with success. Our methodology closely follows the adversarial robustness formulation outlined in Guarino et al. (2020); however, we adapt it to consider a novel approach using a 3D CNN to evaluate each sequence of hive-plot images as one observation. The hive plots are grouped into sequences of 8 images, from timesteps t_0–t_7, and represent network data collected over a given time period. We believe that the sequencing of this data—that is, the spatiotemporal features related to the onset velocity of the attack or the burst rhythm, for example—could have meaningful information related to the classification of whether the network traffic is representative of a DDoS attack or not.

CNNs have long been around and utilized as the preferred model of computer vision tasks. 3D CNNs were first introduced in Ji et al. (2013) and have since become similarly significant in the context of computer vision for 3-dimensional objects and video recognition, such as in Tran et al. (2015). 3D CNNs work similarly to a traditional 2D CNN—in our model, seen in Fig. 1—we use three convolutional blocks, each including a convolutional layer[1], passing a filter over an input performing convolution, a ReLU activation layer to add non-linearity, a pooling layer[2], selecting the most important feature representations with pooling operations. Then we use a fully connected layer to achieve the binary classification. The 3D convolution operations themselves can be formulated as:

$$Y_{c',\,d,\,h,\,w} = \sum_{c=1}^{C_{\text{in}}} \sum_{u=1}^{k_d} \sum_{v=1}^{k_h} \sum_{w'=1}^{k_w} K_{c',c,u,v,w'}\; X_{c,\,d+u-1,\,h+v-1,\,w+w'-1}$$

The distinguishing characteristic between a 3D CNN and a traditional 2D CNN like those used in Guarino et al. (2020) is the shape of the input and of the kernel used to perform convolution. In a 3D CNN, a 3D kernel is passed over a 3D input, capturing spatial patterns over not just 2 dimensions—height and width of some input matrix—but also depth (e.g., time, as in our case). This allows the model to learn spatiotemporal features by convolving over consecutive frames in video data, rather than only learning 2D features frame-by-frame, as in a traditional CNN. We consider applying

[1]See Appendix

[2]See Appendix

this concept to our image sequences, treating each sequence of 8 frames as a single observation—that is, one instance of network traffic collected over some point in time.

Following Guarino et al. (2020), we frame the adversarial robustness training as a saddle point optimization problem, solving an inner maximization problem and an outer minimization problem. That is, the adversary—the adversarial attacks and perturbations—attempts to maximize the loss of the classifier model while the model seeks to find the set of model parameters θ which minimize the overall loss despite the best efforts of the adversary.

In choosing adversarial attacks to use in this work, we looked to Guarino et al. (2020) as well as considered which augmentations and perturbations were both well-established in the literature and would help our model achieve robustness against a wide variety of attacks. Similarly to Guarino et al. (2020), we consider gradient-based attacks and gradient-free attacks.

Gradient-Based Attacks

In gradient-based attacks, the attacker has access to the model they are attacking. In other words, a gradient-based attack leverages the gradient of some machine learning algorithm which uses a gradient for learning. We consider: Gradient Sign Attack, aka Fast Gradient Sign Method, or FGSM Goodfellow et al. (2015) and Projected Gradient Descent Attack, or PGD Madry (2019)

Gradient Sign Attack (FGSM) The FGSM attack uses the gradient of the model to calculate the direction in which to modify the data to maximize the loss with respect to the input. It takes in the original input, a small constant ϵ controlling the magnitude of the perturbation, and the gradient of the loss function, using the sign of the gradient of the loss function to calculate the direction.

We can formulate it using a standard adversarial attack definition[3], and think of this adversary as figuring out how it can modify the data to best fool the model by finding a measure of each pixel's contribution to the loss and perturbing it to maximize that loss. This method is a one-step attack—that is, it is not iterative and considers only the current example to construct the perturbation.

Projected Gradient Descent Attack (PGD) PGD is a similar gradient-based attack we considered for the perturbation of our data. Similarly, it uses the gradient of the model to perturb the data in a way which maximizes the loss of the classifier. However, unlike FGSM, it is iterative. At each iteration, it computes the gradient of the loss function with respect to the input and produces a small perturbation in the direction that maximizes the loss of the gradient. This allows it to create subtler and more effective perturbations than FGSM. We define its formal definition in the appendix. [4]

Gradient-Free Attacks

The gradient-free "black box" attacks we used simply added augmentations to the data, without knowledge of the model architectures or parameters. Such augmentations considered in this work were rotation, noise, shear, and crop. Our values are listed later in the paper.

These attacks may not be the most sophisticated, but they are easy and quick to carry out, meaning that an attacker without access to the model and without much time, either in runtime or development time, can easily carry out such attacks, or similar, which augment the data significantly enough to cause it to fail.

Data Loader

We implement a data loader to allow our model to more efficiently ingest the input data as well as more effectively evaluate the image sequences as a joint observation. Using PyTorch, each sequence of `.png` files gets converted into a 4D tensor object with dimensions $(B \times C \times D \times H \times W)$—batch size, channels, depth, height, width—then stored as a `.pt` file. This file format is very efficient for the model to ingest and allows us to effectively treat the sequences of images as one joint observation, not dissimilar from a video classification task as in Tran et al. (2015). The dimensions the model is then able to convolve over, using the 3D kernel, are height, width, and depth (sequence length of 8).

[3]See Appendix for the mathematical formulation.
[4]See Appendix for the mathematical formulation.

Training Regimes and Evaluation Framework

We evaluate two training regimes—clean and adversarial training.

- **Clean Training:** The model is trained solely on the initial clean, unperturbed data, and accuracy, precision, recall, and area under the ROC curve (AUC) are evaluated at each test condition (clean, augmented, PGD-attacked, and FGSM-attacked).
- **Adversarial Training:** The model is trained on a mixture of perturbed data. This approach optimally balances robustness against adversarial examples while maintaining the ability to effectively classify a mixture of clean and adversarial data. Then, the same metrics are used under the same test conditions with the newly trained adversarially robust model.

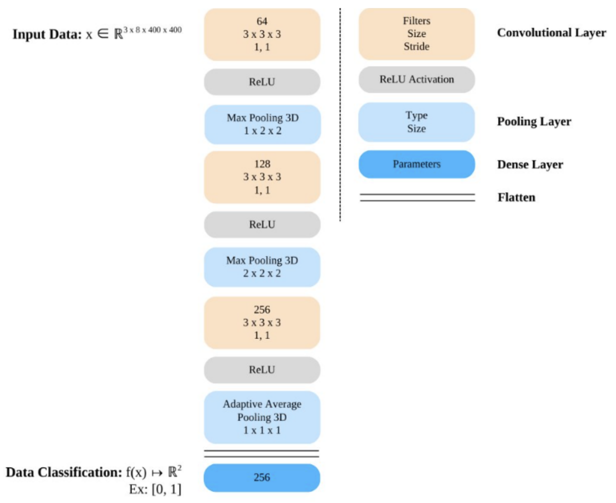

Figure 1: Model Architecture

5 Experiments

5.1 Data

We are using a dataset sourced from Marist College, which is composed of 16,000 hive plot images that display simulated network traffic. This data was created by simulating normal and DDoS attack HTTP requests and capturing them on a honeypot, which was designed to act as a decoy REST API. Each hive plot has three axes: the leftmost axis plots the time passed since the start, with points closer to the origin being closer to the start of the recorded traffic, the vertical axis is the country associated with the IP address, and the axis on the right side is the source IP address. Each line plotted has 50% transparency so that overlapping lines create darker areas. The data is labeled, with eight thousand images displaying regular traffic and eight thousand images displaying DDoS attack traffic. The dataset is about two gigabytes in size and each image is stored as a .PNG. Each image was captured during regular time intervals indexed from 0 to 7. At interval 0 there has been no web traffic, and each image sequentially captures network traffic after an amount of time has passed, up until the eighth image (index 7), where the network traffic hive plot then restarts at index 0.

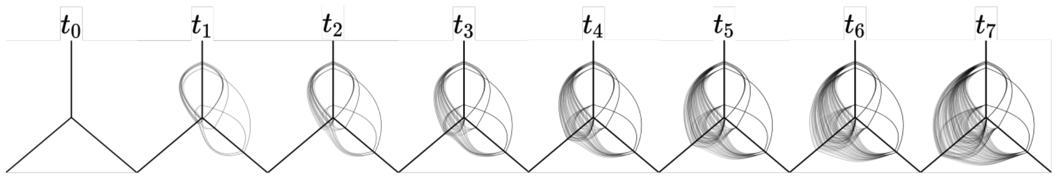

Figure 2: Time Series

To comprehensively assess classifier performance under clean, augmented, and adversarial conditions, we report the following standard metrics:

- **Accuracy** (Acc) Measures the fraction of correctly classified examples:

$$\mathrm{Acc} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{\hat{y}_i = y_i\},$$

  where N is the total number of samples, y_i the ground-truth label, and $\hat{y}_i$ the model's predicted label. Accuracy quantifies overall correctness but can mask class imbalance.
- **Precision** (Prec) The proportion of positive predictions that are true positives:

$$\mathrm{Prec} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}},$$

  where TP and FP denote true and false positives, respectively. Precision evaluates the risk of false alarms in attack detection.
- **Recall** (Rec) Also known as sensitivity, the fraction of actual positives correctly identified:

$$\mathrm{Rec} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}},$$

  where FN denotes false negatives.
- **Area Under the ROC Curve (AUC)** Computes the area under the Receiver Operating Characteristic curve, which plots True Positive Rate (TPR) vs. False Positive Rate (FPR) as the decision threshold varies:

$$\mathrm{AUC} = \int_0^1 \mathrm{TPR}(t)\, d\big(\mathrm{FPR}(t)\big).$$

  AUC measures the trade-off between sensitivity and specificity independently of any particular threshold.

Frame-wise and Regime-wise Evaluation

To characterize temporal robustness, we evaluate each 8-frame sequence under four test conditions (clean, augmented, FGSM, PGD) on a per-frame basis. For each time step $t = 0, \ldots, 7$, we:

1. Extract the single-frame input at t (repeated across depth for the 3D CNN).
2. Apply the specified perturbation (none, random augmentation, FGSM, or PGD).
3. Compute the above metrics on the entire validation set.

5.2 Experimental details

All experiments were conducted on an NVIDIA L40 GPU with CUDA 11.7 and PyTorch 2.0. Data preprocessing and model code were executed in Python 3.10.

5.2.1 Training Regimes

- **Clean:** Standard training on unperturbed examples.
- **Adversarial (mixed):** Each minibatch (size 16) is composed of
  - 8% clean examples,
  - 12% randomly augmented examples (rotation $\pm 18^\circ$, shear $\pm 11^\circ$, zoom 0.75–1.0, Gaussian noise $\sigma = 0.17$),
  - 23% PGD-attacked examples (FGSM step size $\alpha = 1.1$, $\varepsilon = 1.225$, 40 steps),
  - 57% FGSM-attacked examples ($\varepsilon = 1.19$).

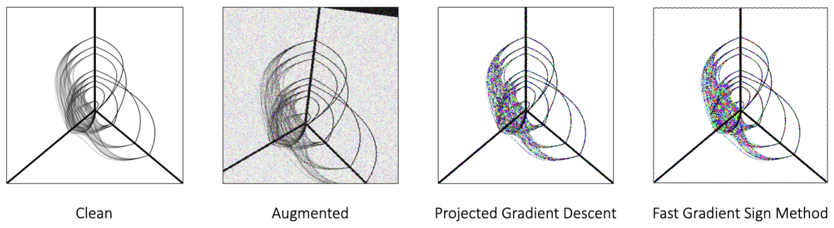

Figure 3: Augmentations/Perturbations

- **Optimizer:** AdamW
- **Initial learning rate:** 5×10^{-5}
- **Weight decay:** 1×10^{-4}
- **Batch size:** 16
- **Epochs:** Early stopping after 3 learning-rate reductions
- **LR scheduler:** `ReduceLROnPlateau` (factor 0.5, patience 3)
- **Mixed precision:** `torch.cuda.amp` with `GradScaler`

5.2.2 Data Loading, Augmentation, & Runtime

- **Dataset:** .pt tensors in preprocessed_dataset/{train,val}/{Normal,DDoS}.
- **Loader:** `DataLoader` with `num_workers=4`, `pin_memory=True`.
- **Augmentations:** Applied batch-wise (rotation, shear, zoom, noise) during both training and evaluation as specified.
- **Runtime**: 3 hours of training and evaluating, few minutes for no-context evaluations

5.3 Results

In this section, we present the results of our experiment in first, DDoS-attack classification by our CNN model. Our first model, trained on strictly clean data, with no perturbations, performed very strongly receiving 100% accuracy, 100% precision and 100% recall for three total epochs of training.

Training Regime	Condition	Accuracy	Precision	Recall
Clean	Clean	1.00	1.00	1.00
Clean	Augmented	0.50	0.50	1.00
Clean	PGD	0.55	0.52	1.00
Clean	FGSM	0.55	0.52	1.00

Table 2: Model performance under various training and testing conditions

Next, we applied the adversarial attack and the augmentations to test our model against particular attacks that an image may face. After applying these attacks, our model performed significantly poor, due to the model being unexposed to the perturbations.

Training Regime	Condition	Accuracy	Precision	Recall
Adversarial	Clean	0.99	1.00	0.97
Adversarial	Augmented	0.985	0.995	0.96
Adversarial	PGD	0.985	0.97	1.00
Adversarial	FGSM	0.9325	0.99	0.855

Table 3: Model performance under various adversarial training and testing conditions

However, this is intentional in our overall model evaluation as retraining on this data will allow for the model to be stronger and more robust. After retraining, our model performed very well receiving scores in the high 90s for our accuracy when identifying and applying the attacks. The attacks and the training split for each type of data can be found in the table below.

Our team was also interested in pursuing how our model performed when taking an image in, without the context of the sequence of images, one time-stamped image would currently be in. Using this no-context evaluation, below is how our model performed, first with clean data, then with the augmentations and attacks.

Singular Timestamp	**t0**	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**	**t7**
Clean	0.5000	0.8475	0.9325	0.9875	1.0000	0.9900	0.9675	0.9200
Augmented Accuracy	0.5000	0.8275	0.9575	0.9925	0.9575	0.8800	0.8000	0.7375
PGD Accuracy	0.5000	0.5275	0.7850	0.9025	0.9150	0.8750	0.8175	0.7625
FGSM Accuracy	0.5000	0.8300	0.9200	0.9625	0.9800	0.9875	0.9950	0.9825

Table 4: Accuracy at different timestamps for various training conditions

6 Analysis

This model could be deployed in a cybersecurity context with a data loader that loads in sequences of real-time hive plot data. The data loader could also analyze one image at a time for predictions, emulating our evaluation method shown previously. The detection system would be robust against augmented data and FGSM and PGD attacks. Black-box attacks, such as the single-pixel attack, could still be a threat because the model has not been trained on this kind of adversarial data. A heavier 3D-CNN and adjusting hyperparameters could help us with this application.

7 Conclusion

This work is a continuation of the research done in Guarino et al. (2020) with adversarially robust models. We worked to create an improved model by utilizing the temporal information in the Marist dataset. 3DCNN model architecture was implemented, which allowed us to train the model on sequences of images rather than one at a time. FGSM and PGD were used for our gradient-based white-box attacks. Shear, crop, rotate, and Gaussian noise were used to create stochastic attacks. Retraining on the augmented and perturbed data produced a model that was resistant to all three attacks individually and that was capable of accurate early attack detection. When machine learning models are used for critical functions such as cybersecurity attack detection, it is imperative that they are robust against adversarial attacks. This research is a step forward in building robust machine learning systems in cybersecurity.

8 Limitations

Our evaluation relied on pre-classified benchmark data of DDoS and normal traffic, which may not be representative of attack patterns and real-world network conditions. The model is also untested against zero-day attacks or more novel DDoS techniques. The data may also be biased toward specific network behaviors. Our adversarial evaluation focused on FGSM, PGD, and augmentation, however, the cybersecurity landscape includes many other untested adversarial techniques such as black-box attacks, CW attacks, and adaptive adversaries. The resilience of the model compared to other attack techniques has not yet been explored. The image-based approach we used with hive plots requires preprocessing network traffic into a specific type of visual representation, adding complexity to the pipeline. Something that could help with improvements would be real-time transformation of network flows into sequential hive plots while aiming to minimize latency. Future work could address these limitations by using larger, more diverse datasets, expanding adversarial attack types, and developing a more efficient preprocessing pipeline for real-time deployment.

9 Ethical Implications

Our approach relies on detailed flow-level features and time-based hive-plot visualizations gathered from user network traffic. We must ensure that we anonymize the data to protect user privacy and comply with regulations. Another ethical implication is that our model may be prone to classify benign surges in legitimate spikes in traffic as attacks. This could lead to problems with unintentional service failures or denying service for regular users. An ethical deployment must balance security and availability by keeping humans in the loop for verification of these classified attacks as a fail-safe mechanism. Finally, the techniques for generating adversarial traffic could be misused to create more effective DDoS attacks capable of breaking defenses. Only high-level summaries of attack methods should be shared while we still provide defensive code that is robust under a controlled license.

References

Tamara Bonaci, Katina Michael, Pablo Rivas, Lindsay J. Robertson, and Michael Zimmer. 2022. Emerging technologies, evolving threats: Next-generation security challenges. *IEEE Transactions on Technology and Society*, 3(3):155–162.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples.

Michael Guarino, Pablo Rivas, and Casimer DeCusatis. 2020. Towards adversarially robust ddos-attack classification. In *2020 11th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 0285–0291.

Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 2013. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231.

Schmidt Tsipras Vladu Madry, Makelov. 2019. Towards deep learning models resistant to adversarial attacks.

Agnieszka Mikołajczyk and Michał Grochowski. 2018. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 117–122.

Pablo Rivas, Casimer DeCusatis, Matthew Oakley, Alex Antaki, Nicholas Blaskey, Steven LaFalce, and Stephen Stone. 2019. Machine learning for ddos attack classification using hive plots. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 0401–0407.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497.

A Appendix

This appendix provides formal definitions and equations for key methods and concepts discussed in the main text, including adversarial attack formulations and neural network operations.

A.1 Fast Gradient Sign Method (FGSM)

The Fast Gradient Sign Method (FGSM) is an adversarial attack designed for neural networks. It computes the gradient of the loss function with respect to the input image and creates an adversarial example by adding a perturbation in the direction of the gradient's sign.

$$\eta = \epsilon \cdot \text{sign}\left(\nabla_x \mathcal{L}(\theta, x, y)\right) \tag{1}$$

Here, ϵ controls the magnitude of the perturbation, $\mathcal{L}$ is the loss function, θ represents the model parameters, x is the input, and y is the true label.

A.2 Projected Gradient Descent (PGD)

Projected Gradient Descent (PGD) is a more powerful iterative extension of FGSM. It recalculates gradients at each step and applies small perturbations constrained within an ϵ-ball around the original input.

$$x^{t+1} = \Pi_{\mathcal{B}_\epsilon(x)} \left(x^t + \alpha \cdot \text{sign} \left(\nabla_x \mathcal{L}(\theta, x^t, y) \right) \right) \tag{2}$$

Here, $\Pi_{\mathcal{B}_\epsilon(x)}$ denotes projection onto the ϵ-ball around x, α is the step size, and t is the iteration index.

A.3 Cross-Entropy Loss

Cross-entropy loss is used for classification tasks and measures the difference between the predicted distribution and the true distribution:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_c^{(i)} \log \left(\text{softmax}(\mathbf{z}^{(i)})_c \right) \tag{3}$$

Where N is the number of samples, C is the number of classes, and $y_c^{(i)}$ is the one-hot encoded true label for class c in example i.

A.4 Softmax Function

The softmax function converts logits into a probability distribution over classes:

$$\text{softmax}(\mathbf{z})_c = \frac{\exp(z_c)}{\sum_{j=1}^{C} \exp(z_j)} \tag{4}$$

A.5 Global Average Pooling (3D)

Global average pooling reduces each feature map to a single value by averaging across the spatial and temporal dimensions:

$$g_c = \frac{1}{D\,H\,W} \sum_{d=1}^{D} \sum_{h=1}^{H} \sum_{w=1}^{W} X_{c,\,d,\,h,\,w} \tag{5}$$

Where D, H, and W are the depth, height, and width dimensions, and X is the feature map tensor.

A.6 3D Convolution Notes

Where $Y_{c',\,d,\,h,\,w}$ is the output feature map at output channel c' and spatial location (d, h, w), $K_{c',c,u,v,w'}$ is the 3D convolution kernel for output channel c' and input channel c with kernel indices (u, v, w'), and $X_{c,\,d+u-1,\,h+v-1,\,w+w'-1}$ is the corresponding element from the input volume