Hardening Deep Neural Networks via Adversarial Model Cascades

Deepak Vijaykeerthy*
IBM Research
deepakvij@in.ibm.com

Anshuman Suri*
IIIT Delhi
anshuman14021@iiitd.ac.in

Sameep Mehta
IBM Research
sameepmehta@in.ibm.com

Ponnurangam Kumaraguru
IIIT Delhi
pk@iiitd.ac.in

***Abstract*—Deep neural networks (DNNs) are vulnerable to malicious inputs crafted by an adversary to produce erroneous outputs. Works on securing neural networks against adversarial examples achieve high empirical robustness on simple datasets such as MNIST. However, these techniques are inadequate when empirically tested on complex data sets such as CIFAR-10 and SVHN. Further, existing techniques are designed to target specific attacks and fail to generalize across attacks. We propose *Adversarial Model Cascades (AMC)* as a way to tackle the above inadequacies. Our approach trains a cascade of models sequentially where each model is optimized to be robust towards a mixture of multiple attacks. Ultimately, it yields a single model which is secure against a wide range of attacks; namely FGSM, Elastic, Virtual Adversarial Perturbations and Madry. On an average, *AMC* increases the model's empirical robustness against various attacks simultaneously, by a significant margin (of 6.225% for MNIST, 5.075% for SVHN and 2.65% for CIFAR-10). At the same time, the model's performance on non-adversarial inputs is comparable to the state-of-the-art models.**

I. Introduction

Deep neural networks (DNNs) have found widespread usage in areas such as computer vision, natural language processing, and computational biology. While their performance matches (or exceeds) human evaluations on the same tasks, sensitive applications such as autonomous navigation and computational finance require a deeper understanding of the predictions before their results can be considered trustworthy. However, since DNNs work by learning non-trivial representations of data, the intermediate representations and feature spaces of these networks have become increasingly complex. As a result, there is scope for vulnerabilities to be introduced into the networks, leading to a host of privacy and security concerns. For DNNs, one of the ways these vulnerabilities have been exploited is through adversarial examples. Adversarial examples, while being indistinguishable to humans from a clean example, can cause DNNs to produce incorrect and, in some cases, disastrous results.

Furthermore, adversarial examples that misguide one model are often successful in deceiving other models that are trained to perform the same task; irrespective of their architectures or the data sets used to train either of them [1]. Attackers may, therefore, conduct an attack with minimal to no knowledge about the target model, by training a substitute model to craft adversarial examples and nonetheless succeed in deceiving the target model.

* These authors contributed equally to this work.

Liu *et al.* [2] propose an ensemble-based approach to generate adversarial examples using an ensemble-based approach and claim high attack rates. They also show transferability of these targeted examples in a black-box setting, where the target model's weights and architectures are unknown. Bhagoji *et al.* [3] propose an alternate method to exploit DNNs via gradient estimation, which, by their results, performs nearly as good as white-box adversarial attacks, while outperforming the then existing state-of-the-art attacks on black-box models.

Ensemble Adversarial Training (EAT) [4] by Tramer *et al.* uses an ensemble of models to increase the robustness of the target model: augmenting training data with homogeneous adversarial samples transferred from other pre-trained models. Even though the examples are generated using different models, they are generated using the same attack algorithm, for instance only FGSM, and hence the model is robust against only that particular attack. This approach does not take into consideration *adaptive* adversaries either.

Work by Wong *et al.* [5] uses the dual of a convex relaxation of the network over the adversarial polytope to lower bound the output. This lower bound can be expressed as another deep network with the same model parameters, and optimizing this lower bound allows us to guarantee the robustness of the network. However, this technique compromises on the predictive performance of the model on unseen test data and is shown to work only in a white-box setting.

The above vulnerabilities have inspired a lot of research on securing neural networks against such attacks. Although some defence strategies have been proposed in the recent past, many of them can be broken when the adversary adapts the attack to consider the defence. The most prominent among these strategies is *adversarial training* [6] where the model is trained with adversarial samples generated by attacks, along with the standard set of training examples. One of the crucial shortcomings of this approach is that due to over-fitting on samples from stronger attacks, the model may still be vulnerable to adversarial examples from weaker attacks [7]. In turn, this hinders the above methods from generalizing to multiple attacks.

It is also worth mentioning that there have been attempts to construct models which are secure by construction [8], [9]. This is achieved by exploiting the structural properties of DNNs. Owing to additional operations for such structural changes, these approaches are computationally expensive and

scale only to models on simpler data sets such as MNIST. Although these approaches increase the robustness of a model for simpler data sets, these strategies are ineffective for complex datasets such as CIFAR-10 and SVHN.

In this work, we build on the *adversarial training* framework, and design a training technique called *Adversarial Model Cascades (AMC)*. One of the key contributions of our paper is that, in addition to the traditional white-box setting, we also consider an *adaptive* black-box adversary that makes use of queries to the target model's prediction function to train a proxy model. We demonstrate that *AMC* improves the empirical worst-case accuracy of the existing methods for several data sets namely MNIST, CIFAR-10 and SVHN.

In addition to the above, we also investigate the suitability of Feature Squeezing [10] as a pre-processing technique in the pipeline to improving the empirical robustness of DNNs. It is important to note that Feature Squeezing as a stand-alone defence has been shown to be easily bypassed [11]. In contrast, when used in conjunction with our framework, we observe that Feature Squeezing effectively improves the accuracy against attacks that the model has seen during training. Our work is the first to demonstrate an end-to-end pipeline to enhance the empirical robustness of DNNs against multiple attacks simultaneously.

II. Background

A. Threat Model

Consider a DNN $f(.;\theta)$ and a clean (no adversarial noise) sample $(\vec{x},\ y) \sim \mathcal{D}'$, where $\mathcal{D}'$ is a proxy for an unknown distribution $\mathcal{D}$, from which the set of samples used to train the target model $f(\vec{x};\theta)$ are drawn. An adversary tries to create a malicious sample $\vec{x}_{adv}$ by adding a small perturbation to $\vec{x}$, such that $\vec{x}$ and $\vec{x}_{adv}$ are close according to some distance metric (either L_1, L_2 or L_∞ norm), and $f(\vec{x}_{adv}) \neq y$. In each domain, the distance metric that we must use is different. In the space of images, which we focus on in this paper, we rely on previous work that suggests that various L_p norms are reasonable approximations of human perceptual distance [12].

For our analyses, we consider two threat-model settings: white-box and adaptive black-box. A white-box adversary is one that has access to the target model's weights and the training data set. An *adaptive* black-box adversary is one that interacts with the target model only through its predictive interface. This *adaptive black-box* adversary trains a proxy model $f_P(\vec{x}';\theta')$ on the set $\mathcal{S}' = \{(\vec{x}'_i, y'_i)\}_{i=1}^{n}$, where $\vec{x}'_i$'s are drawn i.i.d from distribution $\mathcal{D}'$ and $y_i = f(\vec{x}'_i;\theta)$. In this setting, the adversary crafts adversarial examples $\vec{x}_{adv}$ on the proxy model $f_P(\vec{x}';\theta')$ using white-box attack strategies and uses these malicious examples to try and fool the target model $f(\vec{x};\theta)$.

B. Adversarial Examples

Given a clean sample $\vec{x}$ it is often possible to find a malicious sample $\vec{x}_{adv}$, such that $f(\vec{x}_{adv};\theta) \neq y$, and $\vec{x}_{adv}$ is close to $\vec{x}$ according to some distance metric (either L_1, L_2 or L_∞ norm).

For our analyses, we consider the following attacks:

- **Fast Gradient Sign Method (FGSM)**: Uses the gradient of a modified loss function to modify samples, constrained by L_∞ norm [13].
- **Virtual Adversarial Perturbations (VAP)**: VAP perturbs $\vec{x}$ in the direction that can most severely damage the probability that the model correctly assigns the label y (the correct label) to $\vec{x}$ [14].
- **Elastic Adversarial Perturbations (EAP)**: EAP is based on elastic-net regularization, which uses a mixture of L_1 and L_2 penalty functions [15].
- **Projected Gradient Method (PGM)**: Uses projected gradient descent (PGD) on the negative loss function [6].

These four attacks are picked to cover the spectrum of adversarial attacks in literature. Examples for each of these attacks are shown in Figure 1. One can also characterize attacks based on their strength, i.e. how easy or hard it is to defend against adversarial examples generated by the attacks. For instance, VAP and FGSM are generally regarded as weak attacks as many defences proposed in literature have been demonstrated to be effective against VAP and FGSM attacks, as opposed to PGM which is harder to defend against.

Mathematically, all attacks can be viewed as solving the same optimization problem: an adversary tries to create a malicious sample $\vec{x}_{adv}$ by adding a small perturbation to $\vec{x}$, such that $\vec{x}$ and $\vec{x}_{adv}$ are close according to some distance metric (either L_1, L_2 or L_∞ norm), and $f(\vec{x}_{adv};\theta) \neq y$. Thus, the strength of an attack can be measured by how closely it can approximate the true optima of the above optimization problem.

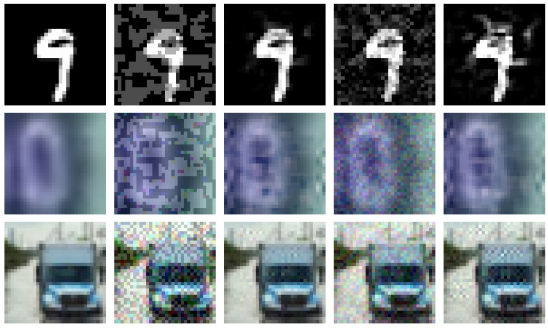

Fig. 1. Examples with various perturbations generated via various attacks for all three datasets. All of these perturbed images make the target model misclassify. L to R: Unperturbed, FGSM, EAP, PGM, VAP. Top to Bottom: MNIST, SVHN, CIFAR-10.

C. Defences

Although many defence strategies to secure DNNs against adversaries have been proposed in the recent past, many of them can be broken when the adversary adapts the attack to take the defence into consideration. The most prominent among these strategies is *adversarial training* [16] where the model is trained with adversarial samples generated by attacks along with the standard set of training examples.

Adversarial training hardens the model against malicious examples by either re-training the model on an augmented set containing the training data and the adversarial examples or learning using the modified objective function:

$$\hat{J}(\theta; \vec{x}, y) = \alpha.J(\theta; \vec{x}, y) + (1 - \alpha).J(\theta; \vec{x}_{adv}, y)$$

where $J(\theta; \vec{x}, y)$ is the original loss function and α is a tunable hyper-parameter. This defence aims to increase the model's robustness by ensuring that it predicts the same class for a clean example and its corresponding example with adversarial perturbation.

As opposed to the general practice of adding adversarial examples without replacement, we add them with replacement: it helps us achieve better generalization performance over an unseen test-set.

Algorithm 1: A high-level overview of the *the adversarial training* procedure AT($\mathcal{A}$, $\mathcal{D}$, N, η, θ)

input: An attack $\mathcal{A}$, training set $\mathcal{D}$, total number of epochs N, a model with parameters θ, and learning rate η

begin
  for $i \leftarrow 1$ **to** N **do**
    Generate adversarial examples for samples in $\mathcal{D}$ using $\mathcal{A}$, $\{\vec{x}^{adv}\}$
    $\theta \leftarrow \theta - \nabla_\theta \hat{J}(\theta; \vec{x} \cup \vec{x}^{adv}, y)$
  end
end

Intuitively, the algorithm iteratively does the following two steps

- Find the optimal $\vec{x}_{adv}$ such that $f(\vec{x}_{adv}; \theta) \neq y$
- Optimizes θ, over the worst case adversarial example $f(\vec{x} \cup \vec{x}^*_{adv}; \theta) \neq y$

Many of the earlier formulations of *adversarial training* have been easily bypassed in practice; this has been attributed to the sharpness of the loss around the training examples. Only until recently, Madry *et al.* [6] showed that adversarial training could be used to obtain a robust MNIST model. That is, the worst-case accuracy on MNIST adversarial examples could be no lower than 81%, even though the attacker knows all parameters of the model. Although their results are encouraging, their technique still falls short in three aspects:

- it is ineffective against *adaptive* black-box adversaries [4], [7],
- it is ineffective for complex datasets such as CIFAR-10 and SVHN [6], and
- it fails to generalize across attacks [7].

III. Approach

To address this challenge, we propose *Adversarial Model Cascades (AMC)* which can help secure models against *adaptive* black-box attacks. Our approach is to train a cascade of models by injecting images crafted from an already defended proxy model (or from the target model itself) to improve the robustness against adversarial attacks. In the next section, we describe the critical steps involved in our approach; namely the construction of the proxy models and the adversarial model cascades.

A. Proxy Models

One of the critical steps in our proposed approach is to train a proxy (or surrogate) model to mimic the target model. The strategy is to train a proxy for the target model using unlabeled examples from the proxy distribution $\mathcal{D}'$. The labels for these data points are obtained by observing the target model's output on these examples. Then, adversarial examples are crafted for this proxy. We expect the target model to misclassify them due to transferability between architectures [1].

One may believe that the choice of a neural network architecture plays a vital role in the effectiveness of the proxy model and the adversary might find it hard to decide on a suitable one. However, the adversary has some partial knowledge of the oracle input (e.g., images, text) and expected output (e.g., classification) at the very least. The adversary can thus use an architecture adapted to the input-output relation. Adversaries can also consider performing an architecture exploration and train several substitute models before selecting the one yielding the highest misclassification. In our research, we use an architecture similar to one of those proposed by Urban *et al.*, which they have shown to be effective in training surrogate models (via distillation) and replicating the predictive performance of the target model [17].

B. Adversarial Model Cascades

Inspired by the observation that adversarial examples transfer between defended models, we propose *Adversarial Model Cascades(AMC)*: training a cascade of models by injecting examples crafted from a local proxy model (or the target model itself). The cascade trains a stack of models built sequentially, where each model in the cascade is more robust than the one before. The key principle of our approach is that each model in the cascade is optimized to be secure against a combination of the attacks under consideration, along with the attacks the model has encountered during the previous iteration. Knowledge from the previous model is leveraged via parameter transfer while securing the model against subsequent attacks. This technique increases the robustness of the next layer of the cascade, which ultimately yields a model which is robust to all the attacks it has been hardened against via the algorithm. A high-level overview of the *AMC* framework is summarized in Algorithm 2.

In the case of a white-box adversary, adversarial examples X_{adv} are crafted for the corresponding model M^i (*AMC, target-hardened*). For the case of an *adaptive* black-box adversary (*AMC, local proxy*), adversarial examples are crafted using an identical local proxy model P'_i: the proxy adversarially perturbs training data. This data is then concatenated with normal training data by the target model for training. It is important to note that, for our approach to work in an *adaptive*

Algorithm 2: A high-level overview of the *AMC* framework. The algorithm works by transferring knowledge for a specific attack and building upon it iteratively to increase robustness.

input: Undefended model M^0, the training set $\mathcal{D}$ of size N on which M^0 was trained on, a set of attacks $\mathcal{S}$ to harden against and the number of attacks E

begin
  for $i \leftarrow 1$ **to** E **do**
    Initialize model parameters of M^{i+1} to θ^{M^i}
    $\theta^{M^{i+1}} \leftarrow \mathtt{AT}(\mathcal{S}(i), \mathcal{D}, \mathrm{N}, \eta, \theta)$
  end
  Predict using the final model in the cascade : $\hat{y} = \arg\max_y \mathrm{M}^E(x)$;
end

black-box scenario, we need access only to the prediction interface of model M^i.

C. Forgetting Attacks

Although *AMC* achieves a significant increase in robustness against many attacks, it does not guarantee an increase in robustness against all of the attacks which it has seen in the past. This condition is especially true for attacks which were seen during the initial iterations of the algorithm. To mitigate this problem, during the later iterations we also need to make the model remember the adversarial examples generated by attacks which were seen by it during its initial iterations. Thus, while constructing adversarial data per batch: instead of generating all perturbed data using the current attack, the algorithm also uses attacks it has seen so far into its run. This process implicitly weighs the attacks, as the model ends up seeing more samples from the attacks which it encounters during earlier iterations. As a result, the loss function to compute its gradient (at a given level E while running the cascade) becomes:

$$\hat{J}(\theta; \vec{x}, y) = \alpha J(\theta; \vec{x}, y) + (1 - \alpha) \sum_{i=0}^{E} \lambda_i J(\theta; \vec{x}^i_{adv}, y)$$

where λ_i are hyper-parameters, such that $\forall i,\ \lambda_i \in [0, 1]$ and $\sum_{i=0}^{T} \lambda_i = 1$. With the above scheme, we observe that the resultant model remembers adversarial examples from attacks which were introduced during the initial iterations as well as recent ones, thus yielding better overall robustness.

When using this weighted scheme in our experiments, we observed an increase in robustness of 35% for MNIST, 25% for SVHN and 21% for CIFAR-10 as opposed to a setup which doesn't incorporate the samples from the weaker attacks during the later iterations of the cascades. Thus, we use the loss function described above.

D. Feature Squeezing

In addition to the above, we also investigate the suitability of Feature Squeezing as a pre-processing technique in the pipeline for improving the empirical robustness of DNNs. It is important to note that Feature Squeezing as a stand-alone defence has shown to be easily bypassed [11]. When used in conjunction with our framework, in contrast, we observe that Feature Squeezing (quantization in particular) can effectively improve the accuracy against stronger attacks.

IV. Experimental Setup

A. Datasets

To measure the performance of our proposed technique, we run our experiments on three datasets standard in the computer vision/machine learning community : MNIST [18], SVHN [19] and CIFAR-10 [20].

For training proxy models, we used the following datasets:

- MNIST: we generated additional data using the technique described by Loosli *et al.* [21].
- SVHN: we used images from the additional set of examples available in the SVHN dataset.
- CIFAR-10: we used images from STL-10 [22] dataset corresponding to labels that are present in CIFAR-10. For the remaining classes ('frog'), we picked images from Imagenet [23] database. All of these were down-sampled to 32×32 pixels.

To ensure a fair comparison, all the targets (including *AMC*), as well as proxy models, were trained such that they all had accuracies within the same ballpark (Table II).

B. Data Preprocessing

All pixels are scaled to $[0, 1]$. We employ the following split strategies for all of the three datasets:

- Training data from the original dataset is used as it is for training.
- Test data from the original dataset is split into two portions. 30% of this is used as validation data by the target model while training itself. Further, 30% of the remaining 70% data, i.e., 21% of the original test data, is reserved while performing attacks. The remaining data is used as a test set for testing generalizability.
- The attacker obtains data at its level for training itself (dataset-wise, described in Section IV-A). That is, the data used by the proxy is independent of the target model.

Such a data acquisition scheme closely resembles a practical scenario, where the adversary may obtain data all by itself. The data used at all stages is balanced class-wise. Data-augmentation, similar to that described by Urban *et al.*, is used for CIFAR-10 and SVHN to help prevent over-fitting [17].

1) AMC Hyper-parameters: At the i^{th} iteration, we perturb 80% of the data using the i^{th} attack, while the remaining 20% data is perturbed equally, in a non-overlapping fashion, by all previous attacks. Using this strategy yields a ratio of 1.6625: 1.1625: 1.0375: 1 of the total perturbed data points seen by the model across all runs combined. The order of attacks used while running both variations of *AMC* is FGSM, VAP, EAP, PGM.

C. Attack Hyperparameters

Based on a trade-off between attack rates and visual inspecting perturbed images, we arrived at the hyperparameters mentioned in Table I for all our experiments. All adversarial data was clipped in the range [0,1] since all the trained models expect input in this range.

TABLE I
ATTACK HYPERPARAMETERS FOR WHITE-BOX AND BLACK-BOX ATTACKS USED THROUGHOUT IN EXPERIMENTS. HYPERPARAMETER NAMES HERE REFER TO THE ONES IN THE CLEVERHANS LIBRARY. TUPLES INDICATE PARAMETERS FOR (WHITE-BOX,BLACK-BOX), WHEREAS SINGLE ENTRIES SIGNIFY THE SAME HYPERPARAMETER FOR BOTH CASES.

Attack	MNIST	SVHN	CIFAR-10
FGSM			
eps	1e-1	1e-1	(3e-1,6e-2)
EAP			
beta	1e-2	1e-2	1e-2
binary_steps	(5,7)	(1,3)	(1,9)
max_iterations	(8,15)	(5,10)	(5,1e3)
initial_const	1e-3	3e-1	(1e-1,1e-3)
learning_rate	1e-1	2e-1	(1e-1,1e-2)
PGM			
eps	3e-1	1e-1	(1e-1,3e-2)
nb iter	(15,20)	5	(5,40)
VAP			
xi	1	(1e-6,1e-4)	1e-6
num_iters	(6,10)	(1,3)	1
eps	(5,8)	(2,3)	2

D. Evaluation Setup

We evaluate the effectiveness of our approach against adaptive black-box adversaries as follows:

1) We train two local proxy models (P' & P''): (P') is used to strengthen (or harden) the target model and (P'') is used to measure the robustness of the hardened models to adversarial attacks. These models are replicas of each other since they have the same architecture and are trained over the same dataset.
2) To test the effectiveness of the model hardening algorithms, we generate adversarial examples for the local proxy model P'' using white-box strategies and attack the target model with these examples.

TABLE II
ERROR RATES (LOWER IS BETTER) FOR VARIOUS TARGET AND PROXY MODELS TRAINED BY US. PROXY HERE CORRESPONDS TO THE PROXY TRAINED WITH ACCESS TO ONLY CLASS-LABELS RETURNED BY THE TARGET MODEL. ADV. TRAINING AND ADV. TRAINING[P] SIGNIFY ACCURACIES FOR MODELS TRAINED WITH ADVERSARIAL TRAINING AND PROXY-BASED ADVERSARIAL TRAINING, AVERAGED OVER THE FOUR ATTACKS.

Model	MNIST	SVHN	CIFAR-10
Undefended	0.9	3.3	9.5
Adv. Training	1.4	2.7	10.4
Adv. Training [P]	1.1	3.2	11.7
AMC,target-hardened	1.4	1.3	9.3
AMC,local proxy	1.1	3.5	9.9
Proxy	0.2	6.0	17.1

1) Proxy Model Architecture: The architecture of the proxy model comprises of 4 convolutional layers and 2 dense layers, using *ReLU* activations and dropout [0.4, 0.3, 0.2], along with 2×2 max pooling after every 2 convolutional layers. This architecture is similar to the one mentioned in [17]. We use the same architecture for all three datasets, with changes in input shape accordingly.

2) Black Box Model Architectures: For CIFAR-10 and SVHN, we used ResNet32 [24] while for MNIST, we used LeNet [25]. For each iteration in the cascade, the model is fine-tuned for 75 epochs.

V. EMPIRICAL RESULTS

We conducted multiple experiments, proving the efficiency of our method against several attacks, both in white-box and black-box setups. *AMC, target-hardened* uses adversarial examples generated from the target model and *AMC, local proxy* generates them using a local proxy. We evaluated the performance of our approach on all the three datasets listed above, for popular white-box attack algorithms such as FGSM, PGM, and compared them with both variations of our proposed framework (Section III-B). Accuracies for the models used in our experiments are given in Table II. Note that there is a slight drop in accuracies for the case of adversarial hardening, which has been reported widely in literature [6], [26]. Despite this fact, models trained with *AMC* have generalization accuracies as good as an undefended model, which is yet another advantage over vanilla adversarial training.

The drop in accuracy for the proxy model on CIFAR-10 can be attributed to the significant difference in the distributions of data used by the proxy and target models. Parameters for the attacks we tested were decided upon after analyzing the adversarial images produced, using them. We visually inspected some of the generated examples to make sure that they are not just noise, while at the same time trying to maximize error induced in the target model[1].

We compare our proposed framework (both variations mentioned in Section III-B) with the best adversarially trained model in terms of empirical robustness for each of the attack.

A. White-Box Attacks

We observe that *AMC, target-hardened*, on an average, gives higher empirical robustness (*accuracy on adversarial examples*). Average robustness here refers to the robustness against all four attacks averaged together. We also observed that models obtained via adversarial hardening against only one kind of attack did not improve robustness against other attacks; whereas our models are robust against all the attacks we considered (Table III). Additionally, we outperform the technique by Wong *et al.* [5] by a margin of 2% for MNIST, 12% for SVHN and 7% for CIFAR-10, on an average across attacks.

[1]We used Cleverhans (https://cleverhans.readthedocs.io/en/latest/) for implementing these attacks and Tensorflow (https://www.tensorflow.org/) for training models

TABLE III

ERROR RATES (LOWER IS BETTER) FOR VARIOUS DEFENSES AGAINST **WHITE-BOX** ATTACKS. WE CAN SEE *AMC*,TARGET-HARDENED PERFORMING BETTER THAN ADVERSARIAL TRAINING AND EAT [4]. ATTACKS {F, E, P, V} HERE CORRESPOND TO {FGSM, EAP, PGM, VAP} RESPECTIVELY. IT IS IMPORTANT TO NOTE THAT WE COMPARE OUR MODEL (*AMC*) WITH THE BEST MODEL IN TERMS OF EMPIRICAL ROBUSTNESS FOR EACH OF THE ATTACK.

Model	White-Box Attacks			
	F	E	P	V
MNIST				
Undefended	85.4	86.6	88.9	69.3
EAT [4]	89.6	97	91.7	42.5
FGSM	5.5	51.2	65.9	42.4
EAP	74.2	15.4	66.4	50.4
PGM	31.6	27.0	4.4	33.1
VAP	27.6	40.6	64.6	19.6
AMC,target-hardened	**5.1**	**8.4**	**3.1**	**3.2**
SVHN				
Undefended	75.7	95.0	97.6	34.6
EAT [4]	90	90.9	99.6	89.5
FGSM	2.7	94.2	94.5	36.9
EAP	91.6	8.4	91.56	89.9
PGM	31.6	58.3	29.3	49.3
VAP	26.7	83.6	94.3	15.8
AMC,target-hardened	**2.4**	**4.9**	**25.1**	**4.5**
CIFAR-10				
Undefended	86.2	94.4	97.2	94.4
EAT [4]	89.5	84.7	98.2	80.9
FGSM	12.0	94.5	95.1	36.5
EAP	90.0	30.1	96.7	50.5
PGM	30.7	35.4	63.5	29.3
VAP	28.9	70.5	92.1	13.0
AMC,target-hardened	**10.6**	**23.9**	**62.5**	**11.0**

TABLE IV

ERROR RATES (LOWER IS BETTER) FOR VARIOUS DEFENSES AGAINST **BLACK-BOX** ATTACKS. THE TAG [P] SIGNIFIES HARDENING WITH EXAMPLES GENERATED WITH A LOCAL PROXY. WE CAN SEE *AMC*, LOCAL PROXY PERFORMING BETTER THAN PROXY-BASED ADVERSARIAL TRAINING AND EAT [4]. ATTACKS {F, E, P, V} HERE CORRESPOND TO {FGSM, EAP, PGM, VAP} RESPECTIVELY. IT IS IMPORTANT TO NOTE THAT WE COMPARE OUR MODEL (*AMC*) WITH THE BEST MODEL IN TERMS OF EMPIRICAL ROBUSTNESS FOR EACH OF THE ATTACK.

Model	Black-Box Attacks			
	F	E	P	V
MNIST				
Undefended	83.2	8.7	81.9	30.9
EAT [4]	39.1	23.6	**8.25**	14.7
FGSM[P]	**17.2**	3.68	25.1	21.3
EAP[P]	82.3	1.5	52.3	28.8
PGM[P]	52.6	3.78	21.3	4.5
VAP[P]	31.3	4.0	39.5	27.03
AMC, local proxy	18.2	**1.3**	20.8	**2.4**
SVHN				
Undefended	72.7	18.8	71.4	31.4
EAT [4]	84.3	49.1	73	57.1
FGSM[P]	62.2	15.1	67.4	27.4
EAP[P]	70.8	12.9	68.6	29.1
PGM[P]	64.3	11.4	54.0	25.9
VAP[P]	69.4	15.8	54.7	26.8
AMC, local proxy	**55.3**	**8.8**	**44.5**	**19.9**
CIFAR-10				
Undefended	66.9	16.0	44.6	36.4
EAT [4]	70.4	22.2	53.8	33.8
FGSM[P]	65.4	16.0	43.7	35.1
EAP[P]	66.6	**16.0**	44.2	35.4
PGM[P]	65.6	18.3	43.6	35.1
VAP[P]	65.8	16.2	43.7	34.8
AMC, local proxy	**59.2**	16.1	**41.3**	**30.7**

B. Black-Box Attacks

For the adversary's proxy models, we consider adaptive proxies trained using three possible predictive interfaces. Given an example x, the target models predictive interface returns:

1) only the most likely label $\hat{y}$,
2) only the most likely label $\hat{y}$ and adds label noise to it (to approximate distillation), or
3) a vector of tuples containing class conditional probabilities $p(y_i|x)$ and the corresponding label y_i.

Even in the case of black-box attacks, irrespective of the predictive interface made available by the target model, we observe trends similar to that of white-box attacks. The first type of proxy, i.e. the one that assumes only class-labels, is under the most general scenario and thus best resembles a real-world black box setting. Results for this proxy are summarized in Table IV. We observed similar trends for the other two kinds of proxy models. Robustness of models trained with *AMC* (*AMC,local proxy*) is, on an average, higher than adversarial training.

C. Comparison with EAT

As described in Section I, Ensemble Adversarial Training(EAT) [4] also adopt an ensemble-based technique for increasing robustness. A trivial extension to EAT for handling multiple attacks would be to create examples which are augmented using multiple algorithms (or attacks). Such a technique has a critical shortcoming: the model overfits to the perturbations introduced during the training. In turn, EAT only provides a marginal increase in the robustness of the model across attacks in comparison to an unhardened model and hence gives a false sense of security. In our experiments, we observe that our approach effectively improves the robustness against attacks in comparison to EAT. In particular, on an average across attacks), we see an increase of 75% for MNIST, 77% for SVHN and 57% for CIFAR-10 in the case of white box setup, and an increase of 14.5% for MNIST, 36.25% for SVHN and 7.8% for CIFAR-10 in the case of black box setup

D. Variations of AMC

To assert the importance of parameter transfer across iterations in our *AMC* algorithm, we also ran *AMC* with the configuration where there is no parameter transfer across the models in the cascade, i.e., ($\mathrm{M}^{i+1} \leftarrow \theta^{\mathrm{M}^0}$). As expected, we observe the performance of the model to be dismal in the above case.

Since the algorithm processes attacks across cascades sequentially, one would expect this order to make a difference.

To test this intuition, we tried two orders: sorted and reverse sorted by the power of attacks observed in the literature, i.e., FGSM, VAP, EAP, PGM and its reverse order. We observed higher robustness for models trained with the first order. Even though the ratio of attack data seen across runs is more-or-less equal, it is slightly higher for attacks seen earlier on in the algorithm. Scheduling stronger attacks first seems more intuitive, but it also adds the possibility of over-fitting to that attack.

E. Generalizing to an unseen Attack

1) Feature Squeezing: Feature Squeezing [10] is an input pre-processing technique that re-scales inputs to get rid of high frequency, potentially adversarial, noise. It has been shown to help achieve slightly nudged accuracies on adversarial inputs. To evaluate the compatibility and advantage of Feature Squeezing on top of models trained with *AMC*, we calculate robustness numbers using this technique. (Table V).

TABLE V
ERROR RATES (LOWER IS BETTER) FOR *AMC* WITH FEATURE SQUEEZING FOR WHITE-BOX ATTACKS FOR BOTH NORMAL AND HIGH-ATTACK RATE HYPERPARAMETERS.

Datset	Normal		Stronger	
	EAP	PGM	EAP	PGM
MNIST				
AMC	8.4	3.1	89.7	98.8
AMC and FS	3.1	2.9	44.6	6.73
SVHN				
AMC	4.9	25.1	94.2	98.1
AMC and FS	2.1	10.1	39.3	40.5
CIFAR-10				
AMC	23.9	62.5	93.9	97.6
AMC and FS	19.2	49.6	36.9	38.2

We calculated these numbers for EAP and PGM attacks; for the hyper-parameters used in earlier sections (Table I) as well as higher parameters (doubled 'nb_iter' and 'eps' for PGM, 'binary_steps' and 'max_iterations' for EAP) to achieve higher error rates. It is important to note that Feature Squeezing as a stand-alone defence has shown to be easily bypassed [11]. When used in conjunction with our framework, in contrast, we observe that Feature Squeezing (quantization in particular) can effectively improve the accuracy against stronger attacks, samples from which weren't previously seen by the model.

2) Leave-one-attack-out validation: To study the capability of models trained via *AMC* to generalize, we test it against an unseen attack: we use $n-1$ attacks for running *AMC*, and the n^{th} one for performing attacks. We experimented using FGSM as the unseen attack in a white-box setting. For all datasets, models trained with *AMC* outperformed undefended models significantly in terms of robustness (Table VI). These numbers are, on an average, better than all of the defence methods (apart from hardening against FGSM, since in that case the attack is known before-hand) in Table III.

TABLE VI
ERROR RATES (LOWER IS BETTER) FOR MODELS TRAINED WITH *AMC* ON PGM, VAP AND EAP IN A WHITE-BOX SETTING. WHEN COMPARED AGAINST THE UNDEFENDED MODEL, WE CAN SEE THAT *AMC* IS PERFORMING BETTER AGAINST FGSM ATTACK, WHICH IT HASN'T SEEN DURING TRAINING. USING FEATURE SQUEEZING (FS) IN CONJUNCTION WITH AMC FURTHER BRINGS THE ERROR RATES DOWN.

Model	FGSM White-box
MNIST	
Undefended	85.4
AMC(with PGM, EAP and VAP)	<u>15.4</u>
AMC(with PGM, EAP and VAP) and FS	**14.8**
SVHN	
Undefended	75.7
AMC(with PGM, EAP and VAP)	<u>12.1</u>
AMC(with PGM, EAP and VAP) and FS	**11.0**
CIFAR-10	
Undefended	86.2
AMC(with PGM, EAP and VAP)	<u>24.7</u>
AMC(with PGM, EAP and VAP) and FS	**23.9**

VI. DISCUSSION

As we observed in our experiments, none of the existing defence techniques work against all adversarial attack simultaneously; hardening a model for a specific attack does not necessarily lead to an increase in robustness against future attacks of that kind. Adversarial training, as we observed, increased robustness just against the attack it is hardened against and did not substantially increase robustness against other attacks significantly.

The key contributions of the *AMC* framework proposed in the paper are as follows :

1) To the best of our knowledge, ours is the first attempt to provide an end-to-end pipeline to improve robustness against multiple adversarial attacks simultaneously (and also multiple kinds of adversaries namely, L_1, L_2, and L_∞); in both white box and black-box settings
2) *AMC* provides robustness against all the attacks it is hardened against, making it an all-in-one defence mechanism against several attacks.
   - In the case of white-box attacks, on an average *AMC* provides an absolute increase in robustness of 6.225% for MNIST, 5.075% for SVHN and 2.65% for CIFAR-10, in comparison to adversarial hardening.
   - In the case of black-box attacks, on an average *AMC* provides an absolute increase in robustness of 0.45% for MNIST, 6.25% for SVHN and 2.7% for CIFAR-10, in comparison to local-proxy based adversarial hardening.
3) Even though Feature Squeezing has been shown to be bypassed, when used in conjunction with our framework, we observe that Feature Squeezing (quantization in particular) can effectively improve robustness against both attacks seen during training and stronger, unseen attacks. On an average, an absolute increase of 2.65% for MNIST, 8.9% for SVHN and 8.8% for CIFAR-10 for seen attacks, and an absolute increase of 68.9% for MNIST, 56.25% for SVHN and 58.2% for CIFAR-10 on top of *AMC* is observed.

4) There is no overhead at inference time. Thus, services can deploy versions of the models hardened with *AMC* without any compromise on latency.
5) It is easy to incorporate into already trained models. As the intermediate step in building the model cascades involves fine-tuning, it is much faster than existing defensive methods which require training from scratch for every attack.
6) As opposed to the performance of models from Madry *et. al.* [6] the resultant model does not compromise on predictive performance on the unseen test set (Table II).
7) We also observe that the performance of AMC for unseen attacks is comparable to models hardened against those specific attacks (Table VI). An interesting direction for future would be to evaluate the performance of *AMC* over a larger set of unseen attacks.

To the best of our knowledge, ours is the first attempt to provide an end to end pipeline to improve robustness against multiple adversarial attacks simultaneously; in both white-box and black-box settings. Scaling *AMC* for datasets such as ImageNet [23] can be explored as part of future work.

REFERENCES

[1] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on CCS*. ACM, 2017, pp. 506–519.

[2] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[3] A. N. Bhagoji, W. He, B. Li, and D. Song, "Exploring the space of black-box attacks on deep neural networks," *arXiv preprint arXiv:1712.09491*, 2017.

[4] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[5] E. Wong and J. Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 2018, pp. 5283–5292.

[6] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[7] I. J. Goodfellow, "Defense against the dark arts: An overview of adversarial example security research and future research directions," *CoRR*, vol. abs/1806.04169, 2018.

[8] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *CoRR*, vol. abs/1801.09344, 2018.

[9] A. Sinha, H. Namkoong, and J. C. Duchi, "Certifiable distributional robustness with principled adversarial training," *CoRR*, vol. abs/1710.10571, 2017.

[10] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[11] Y. Sharma and P. Chen, "Bypassing feature squeezing by increasing adversary strength," *CoRR*, vol. abs/1803.09868, 2018.

[12] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE S & P, 2017, San Jose, CA, USA, May 22-26, 2017*, 2017, pp. 39–57.

[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[14] T. Miyato, S. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing by virtual adversarial examples," *arXiv preprint arXiv:1507.00677*, 2015.

[15] P. Chen, Y. Sharma, H. Zhang, J. Yi, and C. Hsieh, "EAD: elastic-net attacks to deep neural networks via adversarial examples," *arXiv preprint arXiv:1709.04114*, 2017.

[16] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: Increasing local stability of neural nets through robust optimization," *arXiv preprint arXiv:1511.05432*, 2015.

[17] G. Urban, K. J. Geras, S. E. Kahou, Ö. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson, "Do deep convolutional nets really need to be deep (or even convolutional)?" *arXiv preprint arXiv:1603.05691*, 2016.

[18] Y. Lecun and C. Cortes, "The MNIST database of handwritten digits," 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[19] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[20] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[21] G. Loosli, S. Canu, and L. Bottou, "Training invariant support vector machines using selective sampling," in *Large Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds. Cambridge, MA.: MIT Press, 2007, pp. 301–320. [Online]. Available: http://leon.bottou.org/papers/loosli-canu-bottou-2006

[22] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of AISTATS*, 2011, pp. 215–223.

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," *arXiv preprint arXiv:1803.01442*, 2018.