Hoplite Antivirus for Adversarial Attacks: A Theoretical Approach

Anastasios Nikolakopoulos, Achilleas Marinakis, Vrettos Moulos and Theodora Varvarigou
School of Electrical & Computer Engineering, National Technical University of Athens, Greece

Keywords: Adversarial Attacks, Adversarial Defenses, Adversarial Examples, Deep Neural Networks, Machine Learning, Data Analysis, Artificial Intelligence.

Abstract: In the scientific community of Machine Learning and Artificial Intelligence, Adversarial Attacks are evolving to an emerging issue. Carefully perturbed data samples invade to deep neural networks and cause problems, such as misclassifications and false / malformed outputs. The community has proposed multiple defense strategies, in order to overcome this problem. This paper summarizes the existing (and most well-known) adversarial attacks & defenses. Then, it proposes a potential solution to the issue, with a theoretical approach of an antivirus software scenario, the Hoplite Antivirus. This approach could be a vital step towards addressing the constantly evolving adversarial attacks, taking a note from the way software scientists defended (and keep defending) against computer viruses.

1 INTRODUCTION

Deep Neural Networks are the newest trend in the field of Machine Learning. Since the beginning of the 21st century, there has been a rapid increase in interest around the field of Deep Learning. In the last decade, however, DNN model implementations have been adopted in countless industries in the global industry. Typical examples of Deep Learning techniques are image categorization models, with networks such as ImageNet or DeepFace (the latter for even more specific face recognition). But, in recent years there has been a worrying phenomenon that threatens to "tarnish" the usefulness and the real essence of Deep Neural Networks. In particular, these networks are prone to attacks that can lead to incorrect pattern categorization. Feeding the system with data, which have small and seemingly imperceptible changes, can lead to incorrect classifications & conclusions.

Any Neural Network can be fooled by this kind of malformed data, which has become widely known as Adversarial Attacks / Adversarial Examples. This paper will continue (Section 2) by briefly analyzing the existing adversarial attacks (Subsection 2.1), as well as the defenses proposed by researchers around the world (Subsection 2.2). Then (Section 3), a quick look back in history will take place (Subsection 3.1), which shall lead to the presentation of the Hoplite Antivirus (Section 3.2), a theoretical approach for properly addressing the adversarial attack issue. Since these attacks are somewhat new to the science community, plenty of research is yet to be done. Along with the study of the attacks, new ideas on how to deal with them have to be proposed. Conclusions (Section 4) point out this need of further research. Hoplite Antivirus is one such idea. Before presenting this main idea, however, it seems proper to go through the existing adversarial attacks and defenses.

2 ADVERSARIAL ATTACKS & DEFENSES

2.1 Attacks

Starting with the attacks, these could be categorized based on *the purpose* and *the knowledge* of the adversary. Of course, one attack can belong in both a purpose and a knowledge type. It could even be of two purpose types. The attack types categorized **by the purpose of the adversary** are as follows (Ozdag, 2018) (Xu et al., 2020):

- **Poisoning Attacks**, in which the adversary aims to add perturbed data samples to a training dataset. The attacker might also perform perturbation on existing samples of the set.
- **Evasion Attacks**, in which the adversary carefully malforms specific samples from an exist-

Nikolakopoulos, A., Marinakis, A., Moulos, V. and Varvarigou, T.
Hoplite Antivirus for Adversarial Attacks: A Theoretical Approach.
DOI: 10.5220/0010721600003058
In *Proceedings of the 17th International Conference on Web Information Systems and Technologies (WEBIST 2021)*, pages 585-592
ISBN: 978-989-758-536-4; ISSN: 2184-3252

ing classification dataset (or adds some to an existing one). The trained DNN will fail to classify the malformed examples, fulfilling the adversary's goal. Thus, the adversarial examples "evade" the DNN's classification.

- **Targeted Attacks**, where the adversarial examples created by the attacker are fed to the DNN with the purpose of wrong classification to a specific class / label. With his/her perturbed data sample, the adversary wishes to achieve a classification of a targeted label, which happens to be a different one from the original "true" label of the "clean" (non-perturbed) data sample.
- **Non-targeted Attacks**, where the adversarial examples created by the attacker are fed to the DNN with the purpose of wrong classification (same as before), but without a specific wrong class / label. Here, the adversary's goal is not to target a specific label for classification. He/she only wishes for the malformed data samples to be falsely classified to a label other than their "true" and original one.

One could assume that both poisoning and evasion attacks could be part of the non-targeted ones. Now, lets move on to the attack types categorized **by the knowledge of the adversary** (Xu et al., 2020) (Chakraborty et al., 2018). These are:

- **White-box Attacks**, explaining all the scenarios where the adversary has access to vital information of the neural network and the system itself. Such information can be the DNN's architecture, its hyperparameters, the training dataset, information about the dataset, etc. White-box attacks are considered as the most difficult to defend against, since the attacker can carefully craft extremely efficient data samples, based on his/her knowledge upon the network.
- **Black-box Attacks**, containing all the cases where the attacker has no access to information regarding the network, neither the training dataset. Thus, he/she is confined to limited knowledge about the targeted DNN / system. Usually, the adversarial examples created are based on experience from interacting with the DNN (that is, monitoring the DNN's results and trying to understand some classification patterns).
- **Semi-white or Grey-box Attacks**, are terms related to cases where the adversary has some knowledge regarding the network or the training dataset. However, the amount of information he/she possesses is not enough in order to craft "perfect" adversarial examples.

As already mentioned, an attack usually belongs to one purpose type and one knowledge type (it could even belong to more than one, mainly in the purpose category).

2.1.1 White-box Approaches

It is time to go through the existing and most famous attacks proposed until today. Splitting them in white and black-box ones, we start with the attacks of the first kind (that is, the white-box) (Ozdag, 2018) (Chakraborty et al., 2018) (Tramèr et al., 2017) (Xu et al., 2020) (Yuan et al., 2019).

Limited Memory BGFS (L-BGFS) (Szegedy et al., 2013), is one of the first adversarial attacks published. The algorithm's main target is to locate the minimal distorted adversarial example x', given an original and "clean" data sample x. This means that the adversarial example x's distance must be small enough from the original x, for the DNN classifier to be tricked and consider x' as a true data sample. However, x's perturbation must be enough, for the model to classify it in the targeted label t and not in the true label y. So, L-BGFS aims to find a "balanced" distance, with the minimal distortion that tricks the DNN to consider x' as a real x, but at the same time misclassify it in the desired label t. Due to its nature (repetitive process) the algorithm is considered as a time consuming and resource greedy adversarial attack.

Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014), is one of the most famous adversarial attacks (and also one of the first, along with L-BGFS). It follows a different strategy than that of L-BGFS. As a one-step process, it is also drastically more efficient in terms of both time and resources. The algorithm's target is to apply a one-step application of carefully selected noise of very little value (usually imperceptible to the naked eye) to a selected data sample x. The created adversarial example x' is able to trick the deep neural network, leading to wrong classification in a targeted label t. However, a non-targeted version of FGSM has been published as well. Since the adversarial example creation process is quicker and faster, FGSM allows for the generation of large amounts of perturbed data samples.

DeepFool (Moosavi-Dezfooli et al., 2016), is an adversarial attack that follows a different approach in the example generation process, compared to L-BGFS and FGSM. The algorithm's aim is to "escape" the decision boundaries of a classifier F (for a specific data sample x), in order to misclassify the malformed example x' in a label / class other than the original. Trained classifiers "draw" decision boundaries around given data samples. By examining these boundaries (and using DeepFool), an adversary can locate a spe-

cific path which will lead a perturbed example x' out of them, passing to another decision boundary and therefore leading to misclassification. DeepFool is considered as an efficient adversarial attack.

Jacobian-based Saliency Map Attack (JSMA) (Papernot et al., 2016a), is another approach of adversarial example generation. The idea is based on the images' saliency maps. The algorithm works as follows: Given a specific data sample x, a repetitive search takes place for the pixel that impacts the DNN's output the most. Then, this pixel is perturbed, leading the classifier F to falsely classify the generated x' to a targeted label t. Based on this logic, one can understand that JSMA's approach (regarding the attack scenario) differs a lot, compared to the previous attacks.

Basic Iterative Method (BIM) & Projected Gradient Descent (PGD) (Kurakin et al., 2016a) (Kurakin et al., 2016b), are two very similar attacks and that is the reason they are often listed together. Both attacks can be seen as an "expansion" of the original FGSM. That is because they both utilise an iterative version of the one-step FGSM model. Through this repetitive process, carefully selected noise is applied to a data sample x. A function named $Clip()$ is also presented, limiting the level of distortion in each repetition. This way, the generated example x' is not overly malformed. Both BIM & PGD algorithms have proven to be better than FGSM, penetrating through DNNs with implemented defenses against adversarial attacks. The main and actual difference between the two is that PGD initiates the process after selecting a random data sample x (and proceeding to generate x'), whereas BIM works with a - specified by the user - sample x.

Carlini & Wagner's (C&W) Attack (Carlini and Wagner, 2017), is a well known and efficient adversarial model. It has proven to be "better" than both L-BGFS and FGSM, in terms of efficiency and attack success over partially robust DNNs. It shares the same strategy with L-BGFS and that is why it could be considered as an "expansion" over the classic L-BGFS (like BIM & PGD with FGSM). The algorithm's aim is to find the least distorted adversarial example x', given an original data sample x. At the same time, a x' with the best score for the targeted label t is detected. The final generated x' is a "balance" between the least distorted selection and the one with maximum score to the targeted label t. The main difference between C&W and L-BGFS is the use of loss functions in each case. C&W uses a margin loss function, whereas L-BGFS uses a cross entropy loss function.

Another white-box attack is the **Ground Truth Adversarial Example Attack**, where the Reluplex algorithm is utilised (Katz et al., 2017). Through the use of Reluplex, the least distorted adversarial example is found. However, the process is not the same with the ones in L-BGFS and C&W. One more attack is the **Universal Attack** (Moosavi-Dezfooli et al., 2017), where a generic perturbation is found. This perturbation can be applied to many data samples. The more samples successfully affected, the better the generic perturbation. Other attacks are the **Spatially Transformed Attack** (Xiao et al., 2018) (distorting parts of the image data samples through spatial transformation techniques), **Poison Frogs** (Shafahi et al., 2018) (poisoning the training dataset with pairs of malformed data samples x' and original labels y, drastically affecting the correct training process) and the **Real World Attacks** (Eykholt et al., 2018) (Athalye et al., 2018) (putting stickers in road signs in order to trick real-the DNNs in autonomous vehicles, craft 3D printed objects to confuse other real-time classifiers). Other white-box attacks might have been implemented and proposed / published as well. However, the aforementioned ones are the most well known and tested so far.

2.1.2 Black-box Approaches

Moving on to the black box attacks, not much can be said or written regarding these. That is because, the primary research field seems to be more focused on white-box attacks and defenses against them. There might be a good reason for that, stated later on. There are two main approaches to black-box attacks. First, there is the **Substitute Model Attack** (Papernot et al., 2017). It might as well be the most famous black-box approach. The steps followed by the adversary are:

1. Compose a substitute training dataset which will be much like the real one (an effort to gather information related to the real dataset is needed).
2. Feed the targeted DNN classifier F with the substitute dataset created. Then, get the results / labels and create a new dataset of data samples / labels (x, y). Choose a substitute DNN and train it with the new set of x, y tuples.
3. Augment the new dataset and re-train the substitute DNN classifier F'.
4. Use white-box adversarial methods to attack the classifier F'. Gather the most successful attacks and initiate them to the original targeted DNN.

The steps above are actually the most well-known way of initiating a black-box adversarial attack. However, another black-box approach is known as **Zeroth**

Order Optimisation (Chen et al., 2017). It follows a different logic and its steps are the following: *i)* We assume that the adversary has access to the prediction confidence scores of the targeted classifier F. *ii)* Then, these scores are thoroughly examined, in order to obtain gradient information of F. *iii)* All the information extracted will be used for selecting and initiating a proper white-box attack. Zeroth Order Optimisation is an interesting black-box approach, but it is not widely used. However, both attack scenarios have one main common thing, and that is their final step. Both scenarios end up implementing white-box algorithms, in order to attack a DNN. So, in a few words, black-box attacks are actually white-box ones, with the addition of a few steps to get there. This might be the reason why research groups focus on white-box attacks and defenses against them. Another approach is known as **Semi-white or Grey-box Attack**, which utilises **Generative Adversarial Networks (GANs)** to directly generate adversarial examples. The adversary is supposed to have partial knowledge of the DNN and its training dataset. Therefore, he/she crafts adversarial examples from a carefully selected GAN.

2.2 Defenses

It's time to go through the most "famous" defenses against adversarial attacks. First, it is worth highlighting that the ideal defense would be against white-box attacks, not black-box ones. That is because, if a DNN is highly secure against adversaries who seem to know everything about these, all other attacks (with lesser pre-existing knowledge) will be of equal or lower "difficulty" levels. If one can create a robust DNN against white-box techniques, all black-box attacks shall automatically fail. That might be another reason why the research community's focus is primarily on defenses and their white-box attacks. Back to the defenses themselves, they can be divided based on three main categories. These are: ***i)* Gradient Masking** (the efforts of hiding DNNs gradient information), ***ii)* Robust Optimisation** (making DNNs more robust against attacks) and ***iii)* Adversarial Examples Detection** (the efforts of detecting malformed data samples). Let us analyze these three categories (Ozdag, 2018) (Tramèr et al., 2017) (Chakraborty et al., 2018) (Xu et al., 2020) (Yuan et al., 2019)...

2.2.1 Gradient Masking

Starting with Gradient Masking, this category includes two main defensce strategies. First one is the **Defensive Distillation** (Hinton et al., 2015) (Papernot et al., 2016b). This strategy is about "distilling" a training dataset and the result of a DNN classifier F, using softmax. The steps followed are: *1)* Train a network F using a dataset of items and labels (x, y tuples) and set a temperature value T for softmax. Using a temperature T, the softmax generated values are affected. *2)* Generate and store the scores of F (that is, the scores after softmax). *3)* Train a new model $F2$ with the data - scores collected from step 2, using the same temperature value T for softmax. *4)* Initiate the classification process for the test set, changing the temperature T value to $T = 1$. The result is that, all data exiting the "distilled" $F2$ classifier will have hidden gradient information for the DNNs (mainly cause of the changes in temperature), which means that the adversaries will not be able to obtain gradient info from them. The second gradient masking strategy is called **Shattered Gradients**. Its about using a preprocessor, through which all data are passed before being fed to the network. For example, preprocessors can crop, compress, increase (or reduce) the quality of image data samples, etc. This way, the connection between inputs and outputs becomes unclear, something that makes the gradient information retrieval efforts by adversaries more difficult. The shattered gradients defense shares many similarities with the spatial transformation attacks listed above. Unfortunately, the current gradient masking methods are not very successful in blocking most attacks. They might "confuse" the adversary, but only for a while...

2.2.2 Robust Optimisation

On to Robust Optimisation, this category's techniques can be characterised as the most successful, in terms of countering adversarial attacks. One such technique is **Regularization**. Through this approach, the distorted inputs of a DNN will not have a big impact in the output, meaning the classifier's results. "Regularizing" the input data samples, malformed adversarial examples might lose their "power" (since they will be amongst the regularised data) and, eventually, might be classified to the correct classes / labels. So, given a perturbed example x' which has been subject to regularisation, it shall probably be classified to the correct label y, just like the original x will do. Another technique is the **Adversarial (Re)training**. The defenders create adversarial examples (using known algorithms such as FGSM and PGD/BIM) and either mix them with an existing training dataset, or directly train the DNN. The trained model is expected to "understand" future incoming adversarial examples and classify them as normal data samples. In case of PGD, the network is trained exclusively with adversarial examples (FGSM approaches are usually a mix of clean and perturbed data). Apart from generating adversarial data directly from these algorithms, another way

is to use malformed samples created through other DNNs. Using one-step FGSM to attack classifiers $F1$ and $F2$, the defenders collect the examples $x1'$ and $x2'$ which succeeded in penetrating them. Then, these samples are mixed with the original training dataset of F, hoping that these "proven" adversarial samples will make it more robust. One last technique is the **Certified Defenses**. The strategy used is the same with the one found in the Ground Truth Adversarial Attack. Through specific algorithms (such as Reluplex), a repetitive search takes place, in order to find a circle (generated by a radius r), inside of which all potential x values will be considered as natural. Outside the circle (values further than the radius r), x values might be adversarial samples. This way, a "certificate" is given, for a specific x neighbour.

2.2.3 Adversarial Examples Detection

Moving on to the third category, Adversarial Examples Detection, it houses these techniques that aim to identify adversarial samples before they "enter" the DNN zone. This can be achieved by the "installation" of auxiliary neural networks before the main DNN model, which try to distinguish the adversarial examples from the normal data. Moreover, a statistic analysis can take place, between normal and perturbed data samples. The knowledge extracted can then be applied to models that distinguish incoming samples, based on their properties. Last but not least, another technique is the repetitive process of slightly modifying the DNN's parameters, and observation for substantial changes in the outputs. Maybe, these input samples that got different labels from the classifier, after the parameter modification process, are malformed adversarial examples.

3 HOPLITE: AN ANTIVIRUS FOR ADVERSARIAL ATTACKS

3.1 Looking Back at History

As of today, no efficient solution has been found for a highly robust deep neural network against adversarial attacks. All proposed defended seem to be insufficient, providing low levels of robustness (even sometimes little to none, in case the adversary has slightly modified his/her samples), or the process seems to be costly in terms of resources and time. In a few words, there seem to be no proper defense mechanisms for DNNs. Is there a way to surpass this issue? Let us examine the problem from a different viewpoint. Maybe a lesson can be taught from recent history.

During the 1980s, the computer science research community started to analyze the potential threats of computer software, coming from any possible malicious parts of code. It was the beginning of the computer viruses research, which today is considered as an entirely independent science and research section, within the boundaries of computer science. However, almost 40 years ago, computer viruses were mainly only parts of theoretical scenarios in research publications and scientific papers. The community seemed to be anxious about a potential computer virus outbreak, with many kinds of malicious software penetrating thousands of PCs in the future. More specifically, Fred Cohen, a researcher at IBM, published a demonstration that "there is no algorithm that can perfectly detect all possible viruses". In a way, such a publication could have made many software engineers even more pessimist, regarding the hopes of making computers robust and capable of defending viruses. However, history showed that engineers managed to overcome the rising computer virus issue and, slowly but steadily, build antivirus systems (such as antivirus software) capable of detecting the vast majority of existing viruses.

Today, there is a whole market of antivirus software available to consumers. These frameworks tend to function based on a classic pattern: The user buys an antivirus software, installs it on his/her personal computer and then the software proceeds to get daily security updates over the internet. These updates contain information about new viruses and any kinds of data necessary, for the software to provide security and robustness to its PC. So, an antivirus software provider cares for the distribution of its software program, which then must be regularly updated with newer virus data. This is the pattern on which most antivirus frameworks are based. What if we could learn something from the way computer scientists faced the (computer) virus case? Let us keep in mind the three main open issues of today's adversarial defenses:

- Although there are many defense strategies, none of them applies to multiple attacks at once. Most of the defenses proposed, appeal to specific attack models.
- Many defense methods are costly in terms of time and computer resources, in order to be applied to DNNs.
- Adversarial attacks are constantly evolving, making the task of robust DNNs creation even more difficult.

How could these issues be addressed to, considering how we have dealt (and continue to do so) with com-

puter viruses? All the aforementioned defense models can be viewed as a State-of-the-Art (related work), regarding this paper's proposal for effectively blocking adversarial attacks.

3.2 The Hoplite Approach

The idea is simple. What if we could apply the main antivirus software pattern, to that of the DNNs' adversarial defense problem? What if we saw the issue from a different point of view, that of providing pre-trained and already robust DNNs as software packages? In a few words, what if we implemented an antivirus "model" for adversarial attacks? How can we achieve that? This is our definition of antivirus: A deep neural network, to which multiple defense strategies will be pre-applied, being pre-trained with datasets of high data quality and be made available as software to consumers / users / machine learning engineers. The DNN will be re-trained in a specific frequency / basis with newer adversarial examples (which were previously unknown) and the users who have an older version, will be notified of the newer, more robust version. The main provider will create multiple DNN solutions, based on different datasets, needs and use case scenarios. This way, users will have the opportunity to select among different kinds of pre-trained and robust DNNs, based on their needs. Of course, generating a DNN and applying multiple defense strategies to it shall take plenty of time, whilst it shall be resource-greedy. However, since this DNN will be made available to all consumers globally, the time and resource issue will actually be eliminated, since only the main provider will be responsible of it.

So, this is the Hoplite Antivirus Proposal: **A series of pre-trained and robust DNNs, with multiple defense strategies implemented upon them, different dataset base (for different consumer needs) and constant re-training with newer adversarial examples**. These DNNs will be made available to the public, as secure software packages, and ready to be used by users / machine learning engineers. The software package's internal (source code) will be inaccessible, assuring security and eliminating white-box attack scenarios. At the same time, these software packages will "play-along" with a series of programming languages, ready to be used by scripts / programs written in Python, R, Java, etc.

The term "hoplite" was selected from the ancient hoplites, fierce Greek soldiers / warriors protecting their city-states. They were mainly armed with spears and shields, and their armor protected all the vital parts of the torso. Usually, the hoplites formed the Phalanx formation when in combat, making their attacks very efficient, whilst being able to effectively defend themselves against the enemy. In a few words, a hoplite warrior was able to defend himself properly and "do his job" at the same time. In our case, each pre-trained and robust DNN can be seen as a standalone hoplite, ready to defend against adversarial attacks and function (classify) properly. The Phalanx of DNN hoplites will be the main DNN provider, generating different deep neural networks (with multiple dataset bases) and making them available to consumers, as seen in Figure 1. If, for example, a machine learning engineer wishes to use a DNN specifically for images of road signs, he/she will select the most "tailored" DNN from the main provider. The provider can even offer services of custom DNN creation for consumers.

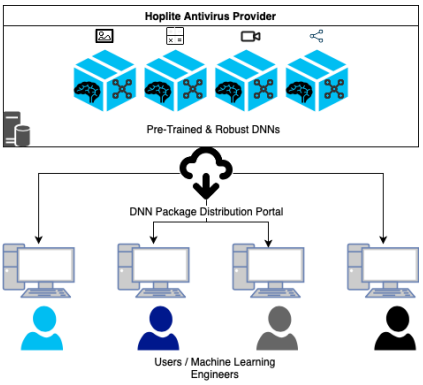

Figure 1: The Hoplite Antivirus's architectural concept.

This idea could actually work. History has shown that using pre-trained software for detecting viruses, and then constantly updating it with newer virus data, is an efficient solution against malware. The same pattern can be applied to the adversarial defense issue. Pre-train deep neural networks, pre-implement multiple defense strategies and make them available to the public, then constantly re-train them with newer adversarial examples. This way, all three main issues mentioned before are being eliminated. First, the implementation of more-than-one defense strategies will appeal to many attack models at once. The Hoplite main provider will carefully examine the DNNs training dataset and then choose the best-suited defense algorithms to implement. Second, only the main provider will face the issue for cost in terms of time and resources. All consumers will simply get the DNNs as software packages, at a glance. Third, evolving adversarial attacks will no longer be an issue, since the Hoplite provider will keep re-training the deep neural networks with newer adversarial examples.

All that is left to do, is focus on studying and researching this approach. It could be a vital solution to the emerging adversarial attacks' danger. Taking a note from the way we dealt with computer viruses and applying it to the adversarial defense problem, could be all we ever needed in order to address it. After all, both computer viruses and adversarial attacks share the same goal: To invade a system and cause malfunction(s), based on the attacker's aim and purpose. In a way, adversarial attacks are viruses. They can be seen as viruses of deep neural networks. Then, why not try to detect them using the same pattern as the one in the computer viruses' case? It only seems rational to test this approach. The Hoplite Antivirus could be a practical solution to our problem!

4 CONCLUSIONS

This paper briefly presented the most well-known adversarial attacks and defenses to date. It extracted three main issues that remain unresolved, regarding the existing adversarial defenses' efficiency. It then proposed a potential solution, the Hoplite Antivirus approach. It is based on the same pattern found in the majority of antivirus software frameworks for computer viruses. The Hoplite Antivirus shall contain a series of pre-trained DNNs, on which multiple defense strategies will be already implemented. The DNNs shall constantly be re-trained with newer adversarial examples, in order to be up-to-date with the evolving attacks. These deep neural networks will be made available to the public as secure software packages, ready for use by consumers / machine learning engineers. Such an approach could end up being vital towards the (yet) unresolved problem of making neural networks fully robust and attack-proof.

Study and research are yet to take place, but this proposal serves as a very good starting point and guide. The team behind Hoplite shall soon initiate a full-scale research and testing phase on this proposal. Using high-end physical machines, capable of performing the resource and time greedy tasks of big dataset processing, constant (for long time periods) DNN training & defense techniques applying, the team aims to monitor the progress and publish the results of each research / testing stage. A careful selection of (different kinds of) datasets will be made first. These datasets will be pre-processed and cleaned. Then, studied DNN architectures will be matched to the, now train-ready, datasets. After these steps are completed, multiple combinations of adversarial defenses will be applied to the DNN-dataset sets. This will be the process of making the DNNs really robust, no matter how long and resource-consuming it might be. The final phase shall include the transformation of the DNNs to encrypted & secure packages, ready for distribution through the net. A better understanding of Hoplite proposal's potential (based on the team's work) is expected during 2022. The main goal is set, and that is to find out if the Hoplite Antivirus approach can indeed be the ultimate solution against adversarial attacks.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Commission under the H2020 Programme's project "DataPorts" (Grant Agreement No. 871493).

REFERENCES

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. (2018). Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR.

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE.

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., and Mukhopadhyay, D. (2018). Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. (2017). Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. (2018). Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. (2017). Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer.

Kurakin, A., Goodfellow, I., and Bengio, S. (2016a). Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.

Kurakin, A., Goodfellow, I., Bengio, S., et al. (2016b). Adversarial examples in the physical world.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2017). Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582.

Ozdag, M. (2018). Adversarial attacks and defenses against deep neural networks: a survey. *Procedia Computer Science*, 140:152–161.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016a). The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2016b). Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. (2018). Poison frogs! targeted clean-label poisoning attacks on neural networks. *arXiv preprint arXiv:1804.00792*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. (2017). Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*.

Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., and Song, D. (2018). Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*.

Xu, H., Ma, Y., Liu, H.-C., Deb, D., Liu, H., Tang, J.-L., and Jain, A. K. (2020). Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178.

Yuan, X., He, P., Zhu, Q., and Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824.