A Lipschitz - Shapley Explainable Defense Methodology Against Adversarial Attacks

Konstantinos Demertzis[1,2(✉)], Lazaros Iliadis[1], and Panagiotis Kikiras[3]

[1] School of Civil Engineering, Democritus University of Thrace, Kimmeria, Xanthi, Greece
kdemertz@fmenr.duth.gr, liliadis@civil.duth.gr
[2] Department of Physics, International Hellenic University, St. Loukas, 65404 Kavala, Greece
[3] Department of Computer Science, University of Thessaly, 35100 Lamia, PC, Greece
kikirasp@uth.gr

Abstract. Every learning algorithm, has a specific bias. This may be due to the choice of its hyperparameters, to the characteristics of its classification methodology, or even to the representation approach of the considered information. As a result, Machine Learning modeling algorithms are vulnerable to specialized attacks. Moreover, the training datasets are not always an accurate image of the real world. Their selection process and the assumption that they have the same distribution as all the unknown cases, introduce another level of bias. Global and Local Interpretability (GLI) is a very important process that allows the determination of the right architectures to solve Adversarial Attacks (ADA). It contributes towards a holistic view of the Intelligent Model, through which we can determine the most important features, we can understand the way the decisions are made and the interactions between the involved features. This research paper, introduces the innovative hybrid Lipschitz - Shapley approach for Explainable Defence Against Adversarial Attacks. The introduced methodology, employs the Lipschitz constant and it determines its evolution during the training process of the intelligent model. The use of the Shapley Values, offers clear explanations for the specific decisions made by the model.

Keywords: Explainable AI · AI defense · Adversarial Attacks · Global interpretability · Shapley values · Lipschitz constraint

1 Introduction

Machine learning models typically accept input data and they lead to problem solving, such as pattern recognition by implementing a series of specific transformations. Most of these transformations prove to be very sensitive to small changes in their input [1]. Utilization of this sensitivity, can, under certain conditions, lead to a modification of the learning algorithm's behavior. The term *Adversarial Attack* [2, 3] is used to describe the design of specific input (not easily perceivable by human observers) which can lead the learning algorithm to erroneous results. This is a major problem in the reliability and security of computational intelligence methods. The problem arises from the fact that

Published by Springer Nature Switzerland AG 2021
I. Maglogiannis et al. (Eds.): AIAI 2021 Workshops, IFIP AICT 628, pp. 211–227, 2021.
https://doi.org/10.1007/978-3-030-79157-5_18

learning techniques are designed for stable environments, in which training and testing data are considered to be generated from the same (possibly unknown) distribution [4]. For example, a trained neural network represents a large decision limit, corresponding to a common class. A properly designed and implemented attack corresponding to a modified input, may come from a slightly differentiated data set and it can lead the algorithm to make a wrong classification decision. For example, the basic structure of a simple neural network consists of two levels of perceptrons, where the former receives the input vector as input and the latter receives the output of the first layer and produces the final output of the network. A classifier that is implemented with the above architecture, accepts as input the vector of the graded characteristics $x_1, x_2, ..., x_D$ and produces the output y_k as follows [5]:

$$y_k = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} \sigma\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right) k = 1, 2, \ldots, K \tag{1}$$

where k is the total number of the network's output, M is the number of the perceptrons, σ is the activation function and $w_{ji}^{(p)}$ the weights that should be trained.

Such classifiers require the input of a feature vector, which can serve as a good representation of the input model. This architecture is based on the direct promotion of the input, from a lower level to a higher one. A group of successive levels that accept input x, produces an output $F(x)$. The final result is equal to $F(x) + x$.

Let us assume a process that attempts *Adversarial Attacks* on a trained network, which classifies input x (of the input space I) as a member of class$F(x) = L, L \in \{1, ..., k\}$. Then the ADA is achieved by minimizing $\|r\|_2$ such as [6]:

$$F(x + r) = L_2 \neq L = F(x) \ \kappa\alpha\iota \ x + r = I \tag{2}$$

In the above modeling, it is considered that two inputs are identical when the L_2 norm of their difference is small. This is a simplification of the problem as the aim is the maximization of the similarity that an observer perceives when he/she sees the two input values. Thus, the measure of similarity of two inputs is an important parameter of the problem, which also affects the approximate solutions that are usually developed. More generally, the detection of *Adversarial Attacks* for a distance $d(x, y)$ different than the L_2 norm of the difference between x, y, is estimated as [7]:

$$\underset{y}{argmin} d(x, y) : F(x) \neq L_2 \ and \ d(x, y) < \tau \tag{3}$$

It should be specified that τ is the threshold for which if two images x, y have a distance $(x, y) < \tau$, then they are considered as similar by a human observer. For example, let us consider that we are developing an *Adversarial Attack*, by employing the *Fast Gradient Signed Method* (FGSM) [8]:

$$adv_x = x + \epsilon * sign(\nabla_x J(\theta, x, y)) \tag{4}$$

where adv_x is the *Adversarial Image*, x is the original input image, y is the original input label, ϵ is a multiplier to ensure that the perturbations are small, θ is a parameter of the

Fig. 1. Tank 92.55% confidence by VGG-16 Network Architecture

model and J is the loss. If we apply this specific procedure and the *VGG-16 Convolutional Neural Network* (as analytically described in the third chapter) we are performing an initial classification of the following image with 92.55% confidence (Fig. 1).

Then we develop perturbations which will be used for the development of the adversarial attacks [3, 9] (Fig. 2).

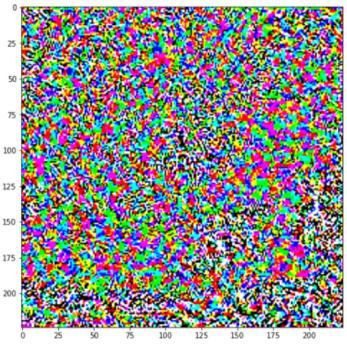

Fig. 2. Perturbations

After applying the FGSM methodology and developing the above perturbations, we obtain adversarial examples, which are given to the network for classification. The results are displayed in the images below (Figs. 3 and 4).

Adversarial Attacks, have been recorded against *anti-spam* filtering, where the system was misled based on the spelling of the keywords. Also, ADA have been used in *cyber-attacks*, by exploiting fake network packets, in deceptively detecting virus signatures and also in biometric identification attacks, where fake biometric features have been used. ADA Defense Strategies (ADA_DS) are based either on modifying the classifier's parameters in order to make it more robust, or on detecting attacks, by using a classifier that can generate *Aggressive Patterns.* Following this approach, ADA_DS recognize these patterns, by introducing them to their own training set. For example, during the

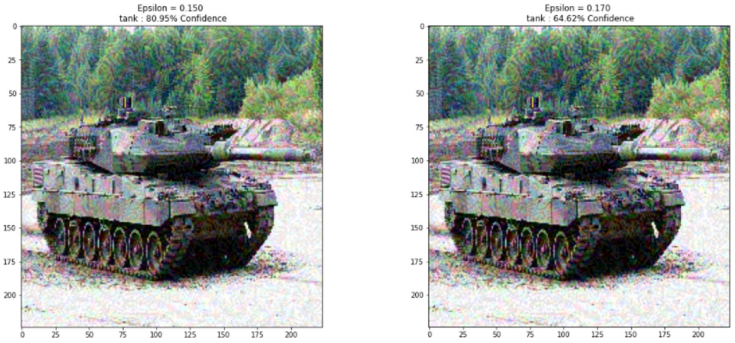

Fig. 3. Tank 80.95% (epsilon 0.15) and 64.62% (epsilon 0.17) confidence

Fig. 4. Tank 29.02% (epsilon 0.19) and Park bench 21.92% (epsilon 0.21) confidence

training of a Neural Network, it learns to successfully identify samples that belong to the training data distribution (DADI). At the same time, it produces large changes in its output, for samples that are in a close distance from the training DADI. Neural networks learn to recognize *Adversarial Attacks* by introducing a linear combination of the patterns of the original set, in the training phase. The desired output is the corresponding linear combination of the original desired outputs. The desired output is the corresponding linear combination of the desired outputs of the original models. More specifically, if $\lambda \in [0, 1]$ and x_i, x_j are two cases of the training set, then the values x_{new} and $f(x_{new})$ are given by the following Eqs. 5 and 6 [2, 7]:

$$x_{new} = \lambda x_i + (1 - \lambda)x_j \tag{5}$$

$$f(x_{new}) = \lambda f(x_i) + (1 - \lambda)f\left(x_j\right) \tag{6}$$

Thus, the training set acquires a more generalized distribution, and the network reaches a better generalization. It does not have large deviation for points of the input space located outside the distribution of the initial training set. The method succeeds in improving network's performance, even for the cases of datasets in which we would expect the classification function to display significant nonlinearities. Respectively, an

effective method of defense is based on the observation that the aggressive cases do not belong to the manifold of the real input data. At the same time, they are closer to the sub-manifold to which the cases of their real category belong. Specifically, let us consider that Y_c is the set of vectors of the training data that belong to a class c, and y is the vector of the input features. Then the $\hat{f}_c(y)$ which is the density distribution of the actual characteristics of category c at point y is estimated as follows [7, 10]:

$$\hat{f}_c(y) = \frac{1}{|Y_c|} \sum_{y_i \in Y_c} exp\left(\frac{-\|y - y_i\|_2^2}{\sigma^2} \right) \tag{7}$$

Where $|Y_c|$ is the number of the elements of the set Y_c. According to this method, an aggressive input corresponding to a real class c_1, can be identified as class c_2, when $\hat{f}_{c_1}(x) > \hat{f}_{c_2}(x)$.

Extending this view, can be used to extract *Bayesian* uncertainty from a neural network (NN), which has been trained using the dropout method. In this case, an input x takes the output $y_1, y_2, \ldots, y_T$ for T different sets of the network's parameters. The network's uncertainty $U(x)$ in point x is estimated by the following relation [6, 8, 10]:

$$U(x) = \frac{1}{T} \sum_{i=1}^{T} y_i^T y_i - \left(\frac{1}{T} \sum_{i=1}^{T} y_i \right)^T \left(\frac{1}{T} \sum_{i=1}^{T} y_i \right) \tag{8}$$

Assuming that aggressive inputs occur in areas of the network where there is high uncertainty, U (x) is a useful metric for determining whether an input x is aggressive or not. Thus the combination of the above two metrics is an appropriate solution for detecting *Adversarial Attacks*.

2 Proposed Method

An innovative, hybrid *Lipschitz-Shapley* methodology for *Explainable Defense against Adversarial Attacks* was developed, aiming to offer a cybersecurity environment with robust solutions capable of recognizing specialized attacks. The proposed methodology, explores the evolution of the *Lipschitz* constant during the training of the intelligent model, while the *Shapley* Values produce clear explanations as to why the model made a particular decision. Specifically using the *Lipschitz* constant we can study the behavior of the *Scattering transform* when introducing *Aggressive Inputs*. This transformation, can approach the operation of a simple Neural Network's architecture, by allowing the study of the way NN succeed in solving difficult problems that require multistage extraction of features.

At the same time, the properties of this transformation explain how a Neural Network can achieve invariability in the displacement of the input, as well as in small deformations of the input, as in cases of elastic deformation. More specifically, the *Aggressive Inputs* are produced when in the input h we add a very small change p. Thus, we obtain the new input $h + p$, which is classified with a properly chosen input function p, in a different way than the initial input such as [4, 11, 12]:

$$\|S[m](h + p) - S[m](h)\| \leq \|p\| \tag{9}$$

So, it turns out that the output for an aggressive image is no different from the original input, more than $\|p\|$. Thus, the transformation follows the constraints of the *Scattering* transformation based on the following [13, 14]:

$$\sum_{i=1}^{\mathrm{N}} \left|\widehat{\psi}_{(i,j)(\omega)}\right|^2 \leq \frac{C^2}{N}, \left|\widehat{\varphi}_{(\omega)}\right|^2 \leq C^2 \tag{10}$$

For a value C:

$$\|S[m](h+p) - S[m](h)\| \leq C^{m+1}\|p\| \tag{11}$$

This means that the constant C is a factor that determines how vulnerable the transformation is to the potential changes of the input against p.

In this research we attempt to detect the evolution of the *Lipschitz* constant [15], during the training of a neural network when we use defense methods. In particular, suppose that the input of a Convolutional Neural Network (CNN) is in the form of a vector. Let $f(x_{in}, c)$ be the output of the network for the class c when the input vector is x_{in}. Let y_{in}, h_{in} be two different input vectors, with respective output $f(y_{in}, c), f(h_{in}, c)$ where y_{ik}, h_{ik} are the outputs of the k$^{\text{th}}$ layer in channel i for each one of the 2 inputs. The CNN comprises of convolutional layers, pooling layers and ReLU activation functions. For each of the three types of layers we have:

1) Layer k is a convolution layer. As the images are expressed as one dimensional vectors, the convolution with a two dimensional core ψ_{ijk}, which connects the i output channel with the j input channel, is developed by multiplying the input vector with the table A_{ijk} that is produced by the initial core as follows [7, 11, 16, 17]:

$$x_{ik} = \sum_{j=1}^{N_k} A_{ijk} x_{j(k-1)} i = 1, 2, \ldots, M_k \tag{12}$$

where N_k is the number of the input channels and M_k is the number of the output channels of the convolutional layer k.

$$\begin{aligned} \|y_{ik} - h_{ik}\|_2 &= \left\| \sum_{j=1}^{N_k} A_{ijk} y_{j(k-1)} - \sum_{j=1}^{N_k} A_{ijk} h_{j(k-1)} \right\|_2 \\ &= \left\| \sum_{j=1}^{N_k} A_{ijk} \left(y_{j(k-1)} - h_{j(k-1)}\right) \right\|_2 \\ &\leq \sum_{j=1}^{N_k} \|A_{ijk} \left(y_{j(k-1)} - h_{j(k-1)}\right)\|_2 \\ &\leq \sum_{j=1}^{N_k} \|A_{ijk}\|_2 \|y_{j(k-1)} - h_{j(k-1)}\|_2 \end{aligned} \tag{13}$$

$$\Rightarrow \|y_{ik} - h_{ik}\|_2 \leq \sum_{j=1}^{N_k} \|A_{ijk}\|_2 \|y_{j(k-1)} - h_{j(k-1)}\|_2 \tag{14}$$

2) Let k be the Pooling layer where we do not have any overlapping areas, such as:

$$\|y_{ik} - h_{ik}\|_2 \leq \|y_{j(k-1)} - h_{j(k-1)}\|_2 \tag{15}$$

3) If the ReLU function is applied, then the output vector has the following form:

$$x_{ik} = \begin{bmatrix} x_{ik}(1) \\ x_{ik}(2) \\ \vdots \\ x_{ik}(m) \end{bmatrix} \tag{16}$$

The output $x_{ik}(t)$ is estimated as follows:

$$x_{ik}(t) = max\big(0, x_{i(k-1)}(t)\big) \tag{17}$$

$$\begin{aligned} \|y_{ik} - h_{ik}\|_2^2 &= \sum_{t=1}^{m} \big|max\big(0, y_{i(k-1)}(t)\big) - max\big(0, h_{i(k-1)}(t)\big)\big|^2 \\ &\leq \sum_{t=1}^{m} \big|y_{j(k-1)}(t) - h_{j(k-1)}(t)\big|^2 = \|y_{j(k-1)} - h_{j(k-1)}\|_2^2 \\ &\Rightarrow \|y_{jk} - h_{jk}\|_2 \leq \|y_{j(k-1)} - h_{j(k-1)}\|_2 \end{aligned} \tag{18}$$

where $|max(0, \alpha) - max(0, \beta)| \leq |\alpha - \beta|$.

Based on the Eqs. 13, 14 και 17, we can estimate a constant L_{ik} for which:

$$\|y_{jk} - h_{jk}\|_2 \leq L_{ik} \|y_{10} - h_{10}\|_2 \tag{19}$$

The constant $L_{ik} = 1$, is defined recursively, and for any type of layer it is estimated as follows:

$$L_{ik} = \sum_{j=1}^{N_k} \|A_{ijk}\|_2 L_{j(k-1)} \tag{20}$$

1) Pooling layer and ReLU function:

$$L_{ik} = L_{i(k-1)} \tag{21}$$

If the network has p layers, we can find the *Lipschitz* constant that satisfies the following relation:

$$\|f(y_{in}, c) - f(h_{in}, c)\|_2 \leq L_{cp} \|y_{in} - h_{in}\|_2 \tag{22}$$

Having developed the method for finding a *Lipschitz constant* for the network, we study how it evolves during the training of a *Convolutional Neural Network*. To get a holistic picture of the network, we need to understand how it makes decisions, as well as its most important features and the interactions between them.

The use of global and Local Interpretability methodology, offers a more thorough approach. *Global interpretability*, provides an overview of the model. *Local interpretability*, focuses on explanations coming from a small data area where a single example is analyzed and it is explained why the model made a specific decision [18]. In small areas of data, the predictions may depend only linearly or monotonously on certain features, instead of having a more complex interdependence between them. *Shapley* values are a very effective way to generate explanations on the way a model works [19, 20]. Its mathematical background comes from the *Cooperative/Coalitional Game Theory*, where the "Payoff/Gain" of the players of a cooperative game is given by a real function that gives values to sets of players [19, 20]. The *Shapley* value is the only payoff function that satisfies the following four key properties [21, 22]:

1. Anonymity: the axiom of anonymity states that the order of the players does not affect the amount allocated to them by the *Shapley*. The consequence of the "*Axiom of Anonymity*" is the "*Axiom of Symmetry*", which states that the *Shapley* values of two symmetric players 1 and 3 are equal.
2. Efficiency: this axiom determines that the distribution of social wealth according to the *Shapley value* is collectively rational.
3. Zero-Useless: this axiom determines that if a player has zero contribution to social wealth, then his/her *Shapley* value equals 0.
4. Additionality: this axiom specifies that the *Shapley* value of the "*Sum Game*" is the sum of the *Shapley* values.

The connection of *Shapley* values to the problem of explaining the architectural structures of a network is done in the following manner: We consider the problem of architectural structures, as a "*Cooperative Game"* whose players are the characteristics of the data set. The "*Gain Function"* is the neural network's model under consideration and the model's predictions are the "*Corresponding Gains"*. In this content, the *Strapley* values show the contribution of each feature and therefore the overall explanation why the model made a specific decision. In conclusion, the *Shapley* value for a characteristic i of a neural network's model f, is given by the following relation [21, 23]:

$$\varphi_i = \sum_{S \in F \setminus \{i\}} \frac{|S|!(M - |S| - 1)!}{M!} \left[f_{S \cup \{i\}}\left(x_{S \cup \{i\}}\right) - f_S(x_S) \right] \tag{23}$$

where F is the set of characteristics, S is a subset of F and $M = |F|$ is the size of F.

This relationship measures the weight of each attribute by calculating and adding its contribution when it is present in the forecast, whereas it subtracts it when it is absent.

More specifically [19, 24]:

1. $f_{S \cup \{i\}}\left(x_{S \cup \{i\}}\right)$: is the output when the i^{∞} characteristic is present.
2. $f_S(x_S)$: is the output when the i^{∞} characteristic is absent.

3. $\sum_{S \in F \setminus \{i\}} \frac{|S|!(M-|S|-1)!}{M!}$: is the weighted average of all possible subsets of S in F.

The *SHapley Additive exPlanations* method (SHAP) explains the models' decisions using *Shapley* values. An innovation of SHAP is that it functions as a linear model and more specifically as a method that estimates the additional contribution of each feature, so that an explanation is a local linear approach of the model's behavior. In particular, while the model can be very complex as an integrated entity, it is easy to approach it around a specific presence or absence of a variable. For this reason, the following step is the calculation of the degree of linear correlation, between the independent and the dependent variables of the set, with dispersion σ_X^2 and σ_Y^2 respectively. The next step is the calculation of the covariance $\sigma_{XY} = Cov(X, Y) = E(X, Y) - E(X)E(Y)$, which is calculated by the *Pearson Correlation Matrix R* as follows [1]:

$$R = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} \tag{24}$$

The proposed architecture initially took into account the inability of the above method to detect nonlinear correlations such as *sinus wave, quadratic* curve, or to explore the relationships between categorical variables.The Predictive Power Score (PPS) technique was selected and used in this study, to summarize the prognostic relationships between the available data [25]. The PPS, unlike the correlation matrix, can work with non-linear relationships with categorical data, and also with asymmetric relationships, explaining that variable A informs variable B more than variable B informs variable A. Technically, scoring in the interval [0, 1] is a measure of the model's success in predicting a variable target, using a non-sample variable prediction. This fact, practically means that this method can enhance hidden patterns' discovery and can contribute towards the selection of suitable prediction variables.

The use of the PPS method, focuses on the fact that a *local explanation* must be obtained for the models that are ultimately capable of operating without training. However, the sensitivity of the *SHAP* method requires the implementation of feature selection before its application. This will enhance its ability to explain the models as long as the hyper-parameters' values are concerned, and it will increase the general ability to deal with the high dimensionality of data. The complexity of the problem becomes even higher, as it is combined with the large number of explanations that must be given for the produced predictions. Thus the distinction between relevant and irrelevant features as well as the distances between data points cannot be fully captured. Taking this observation seriously, feature selection was performed to optimally select a subset of the existing features. This was done without transformation, in order to reduce their number and preserve the most important and most useful of them. This step is crucial because if features with low resolution capacity are selected, the resulting system will not be able to perform satisfactorily, whereas if features that provide useful information are selected, the system will be simple and efficient.

In general, the goal was to select those characteristics that lead to long distances between classes and small variation between the instances of the same class. The process of feature selection was performed with the PPS technique. The metric of Mean Absolute Error (MAE) was used for the calculation of PPS in numerical variables. MAE is related

to the average of the differences between the predicted and the observed values (Eq. 26) [1].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| = \frac{1}{n}\sum_{i=1}^{n}|e_i| \tag{25}$$

where f_i is the estimated value and y_i is the actual. The average of the absolute value of the quotient of these values is defined as the absolute error of their relation $|e_i| = |f_i - y_i|$.

Respectively, the F-Score is the Harmonic Mean of Recall and Precision. It is used to overall assess classification efforts. The higher the F-Score, the higher the more accurate the classification. It is calculated from the following Eq. 27 [1]:

$$\begin{aligned} F_{Score} &= \frac{2 \times recall \times precision}{recall + precision} \\ &= \frac{2TruePositives}{2TruePositives + FalsePositives + FalseNegatives} \end{aligned} \tag{26}$$

3 Experiments

Having developed the method for finding a *Lipschitz* constant for the network, we study how it evolves during the training of the VGG-16 Network for a 224 × 3224 × 3 image data set [26]. This network is characterized by its simplicity, as it uses only 3 × 3 convolutional layers that are stacked one after the other, in scalable depth. The reduction in volume size is addressed with *Max Pooling*, while the architecture is completed by two fully interconnected levels each with 4,096 nodes, which are followed by a *Softmax* classifier.

The architecture of the VGG-16 is described in more detail as follows:

1. The input is an image of dimensions 224 × 224 × 3.
2. The first two layers have 64 channels of 3 × 3 filter size and same padding. Then after a max pool layer of stride 2 × 2, two layers have convolution layers of 256 filter size and filter size 3 × 3.
3. This followed by a max-pooling layer of stride 2 × 2 which is same as the previous layer. Then there are 2 convolution layers of filter size 3 × 3 and 256 filter.
4. After that, there are 2 sets of 3 convolution layer and a max pool layer. Each has 512 filters of 3 × 3 size with the same padding.
5. This image is then passed to the stack of two convolution layers.
6. In these convolution and max-pooling layers, the filters we use are of the size 3 × 3 instead. In some of the layers, it also uses 1 × 1 pixel which is used to manipulate the number of input channels. There is a padding of 1-pixel done after each convolution layer to prevent the spatial feature of the image.
7. After the stack of convolution and max-pooling layer, we got a 7 × 7 × 512 feature map. We flatten this output to make it a 1 × 25088 feature vector.

8. Finally, there are three fully connected layers. The first receives input from the last feature vector and outputs a $1 \times 4{,}096$ vector. The second layer also outputs a vector of size $1 \times 4{,}096$. The third layer outputs 1,000 channels for 1,000 classes. The output of the 3rd fully connected layer is passed to the *softmax* layer in order to normalize the classification vector (Fig. 5).

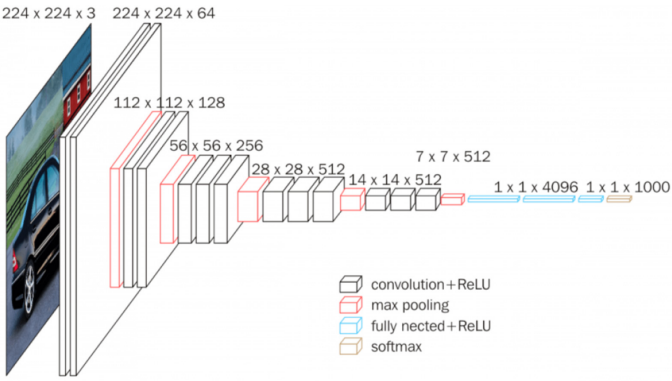

Fig. 5. VGG-16 Convolutional Network's Architecture

The network (as shown in Fig. 6) consists of 16 layers, where the last one has 1,000 different outputs, one for each image class (ILSVRC challenge).

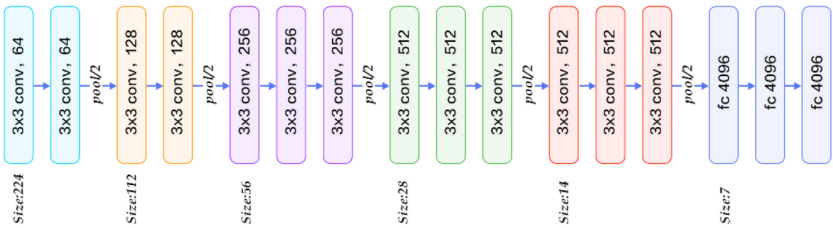

Fig. 6. VGG-16 Layers

So we record the average value of the constants L_{i16} which is symbolized as L_{out}:

$$L_{out} = \frac{1}{1000} \sum_{\iota=1}^{1000} L_{i16}$$

For the experimental validation of the proposed method, we trained the network using a mixture of real and aggressive inputs with the method of adversarial examples. On the other hand, we also trained the network with normal image patterns. At the end of each epoch, we recorded the constant L_{out} and studied how it evolves during network training. The results are presented in Fig. 7 below, where the *Adversarial* examples are introduced in the 10^{th} season of training.

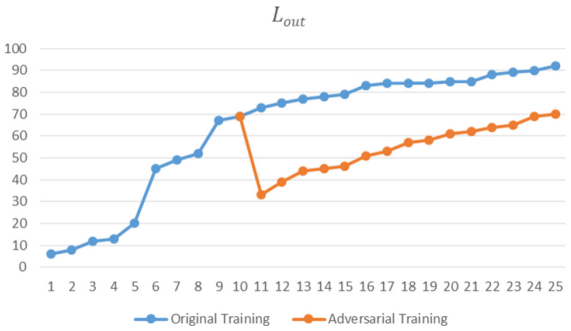

Fig. 7. Lipschitz constant in Original vs Adversarial training

The following Fig. 8 presents the evolution of the training error for both methods.

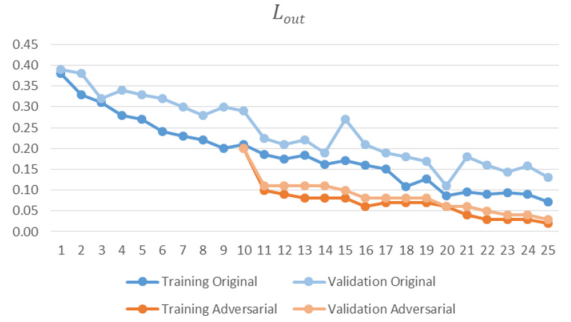

Fig. 8. Lipschitz constant in Original vs Adversarial training and validation

It is obvious that training with aggressive images leads to a neural network with smaller constant L_{out}. This confirms the assumption that networks with smaller *Lipschitz* constants behave better against aggressive images. It is worth noting that the constant we calculate is not a lower barrier to the *Lipschitz* network constant, but continues to exhibit the behavior we expect. Also if we observe the evolution of the error in the validation set, we see that the network that has been trained with aggressive images and presents smaller L_{out} constant, achieves smaller error and generalizes better. The *Shapley* values are then used as a way to generate *clear explanations of how the neural network works*, and *why the model made a particular decision.*

It should be emphasized that the explanations of a prediction in terms of the original input are more difficult than the explanations of predictions in terms of the convolutional layers (because the higher convolutional layers are closer to the output). In the following example we explain how the 7th intermediate level of the VGG-16 model affects the output probabilities for a particular input. Specifically, red pixels represent positive Shapley values, which increase the probability of a class, while green pixels represent negative values that reduce the probability of a class. We should emphasize that two explanatory classification variables (rank_outputs = 2) were used to simplify the process. This means that we explain only the two most likely classes for each entry, freeing the process from the explanation of all 1,000 classes that are part of the VGG-16 network. With this technique, it is possible to understand how the model makes decisions and what interactions take place between the used characteristics in order to achieve correct or incorrect categorization. It also offers the ability to manage, control and explain, how to handle multiple intermediate representations, as well as more advanced features that may be related to the hierarchical organization of the architecture in successive layers. Thus it reflects the structure of the dynamics of the developed system. The progressive classification and investigation of the intermediate relations of the input data along the levels of the hierarchical architecture, creates clear indications - evidence of how the final decision is made. This is achieved even if all the layers share the same weight values. Furthermore, the *Lipschitz-Shapley* combination clearly captures the transient states of the internal representations of the input signals, even for problems that require long internal memory intervals.

4 Conclusions

The idea of standardizing the proposed methodology, emerged based on the application of a single, universal method of explaining how to make decisions in intelligent systems. This approach should meet all the requirements for the case of the *Adversarial Attacks*. Trying to detect how the *Lipschitz* constant is evolving in the training of an intelligent model and using the *Shapley Values*, we manage to obtain clear explanations of the reasons that have led to a particular decision. This can enhance the development of modern information systems' security applications, resolving issues-queries such as:

1. How many neurons should each hidden level contain?
2. Which activation function works best in each hidden layer?
3. Does the addition of a new layer increase the performance of the architecture or not?
4. What is the function of the final layer, as well as which loss function can optimize the process?
5. Is there a need for normalization before the linear levels are added?
6. How many hidden layers need to be used?

When evaluating the proposed architecture as a whole, a significant advantage is focused on its ability to clearly display the information that has been modeled. This can lead to trusted models, which are based on detailed mathematical and physical frameworks of system's behavior. This is in contrast to the corresponding "black box"

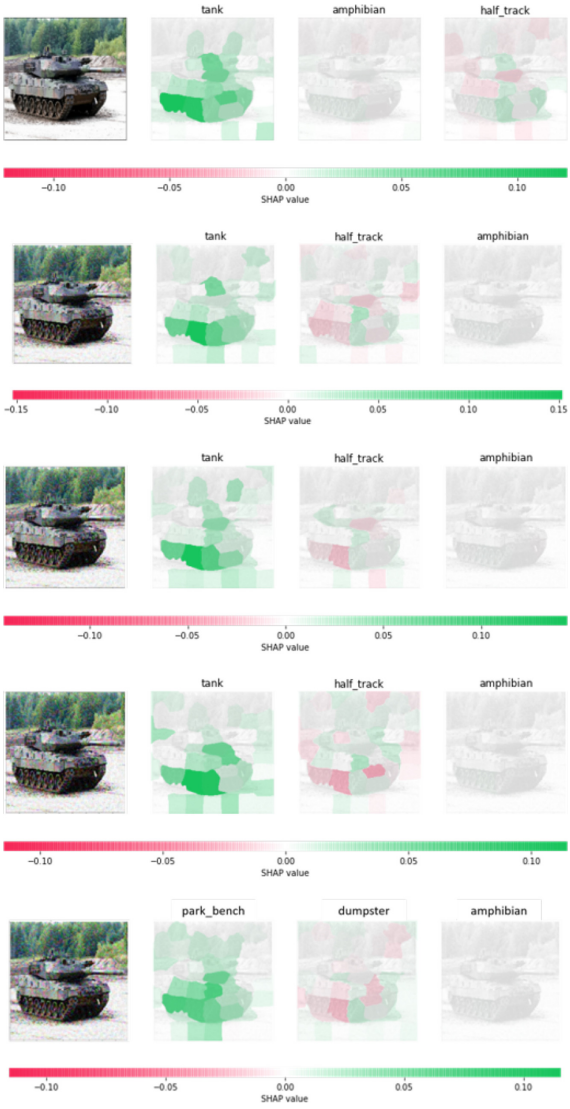

Fig. 9. Shapley values for Original photo and Adversarial examples

approach, which lacks explanations. Another important contribution of the proposed methodology, is related to the assessment of the uncertainty which is an inherent property of digital security problems (DSP). It is a fact that DSP do not follow specific distributions and they are often linearly unrelated. Finally, an important contribution of this method is that generalization is ensured experimentally, with unquestionable statistical validation techniques, even in the case of data use that contains a significant percentage of noise (Fig. 9).

Although the evaluation of the proposed methods was done by applying to particularly painful scenarios and data sets that emerged after exhaustive literature and experimental research procedures, an important limitation that should be noted is that in general intelligent methods are sensitive to data quality. Most of the small transformations related to the slightest changes in the input layer, were analyzed and proved to be of major importance. The utilization of this sensitivity can lead under certain conditions to the modification of the behavior of the learning algorithm. Designing an appropriate strategy to deal with Adversarial Attacks is a specialized and time consuming process. Such a process can be simplified by the proposed methodology, but it still remains a serious issue. Its automation and self-adaptation by machine learning models based on corresponding procedures, requires further investigation.

Proposals for the development and future improvements of the methodology, should focus on the automated optimization of the appropriate parameters, so that an even more efficient, accurate and faster explanation process is achieved, in a simple and unequivocal way. It would also be important to study the expansion of this system by implementing methods of self-improvement that would redefine the hyperparameters of the intelligent system automatically. This could lead to a more efficient system that would resist Adversarial Attacks very effectively.

References

1. Understanding Machine Learning, Pattern recognition and machine learning. Cambridge University Press. https://www.cambridge.org/il/academic/subjects/computer-science/pattern-recognition-and-machine-learning/understanding-machine-learning-theory-algorithms. Accessed 16 Feb 2021
2. Tygar, J.D.: Adversarial machine learning. IEEE Internet Comput. **15**(5), 4–6 (2011). https://doi.org/10.1109/MIC.2011.112
3. Zhu, Z., Lu, Y., Chiang, C.: Generating adversarial examples by makeup attacks on face recognition. In: 2019 IEEE International Conference on Image Processing (ICIP), pp. 2516–2520 (2019). https://doi.org/10.1109/ICIP.2019.8803269
4. Guo, H., Peng, L., Zhang, J., Qi, F., Duan, L.: Fooling AI with AI: an accelerator for adversarial attacks on deep learning visual classification. In: 2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP), vol. 2160–052X, pp. 136–136 (2019). https://doi.org/10.1109/ASAP.2019.00-16
5. Demertzis, K., Tziritas, N., Kikiras, P., Sanchez, S.L., Iliadis, L.: The next generation cognitive security operations center: adaptive analytic lambda architecture for efficient defense against adversarial attacks. Big Data Cogn. Comput. **3**(1), 6 (2019). https://doi.org/10.3390/bdcc3010006
6. Jing, H., Meng, C., He, X., Wei, W.: Black box explanation guided decision-based adversarial attacks. In: 2019 IEEE 5th International Conference on Computer and Communications (ICCC), pp. 1592–1596 (2019). https://doi.org/10.1109/ICCC47050.2019.9064243
7. Yu, P., Song, K., Lu, J.: Generating adversarial examples with conditional generative adversarial net. In: 2018 24th International Conference on Pattern Recognition (ICPR), pp. 676–681 (2018). https://doi.org/10.1109/ICPR.2018.8545152
8. Liu, Y., Mao, S., Mei, X., Yang, T., Zhao, X.: Sensitivity of adversarial perturbation in fast gradient sign method. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 433–436 (2019). https://doi.org/10.1109/SSCI44817.2019.9002856

9. Li, H., Zhou, S., Yuan, W., Li, J., Leung, H.: Adversarial-example attacks toward android malware detection system. IEEE Syst. J. **14**(1), 653–656 (2020). https://doi.org/10.1109/JSYST.2019.2906120
10. Yuan, J., He, Z.: Adversarial dual network learning with randomized image transform for restoring attacked images. IEEE Access **8**, 22617–22624 (2020). https://doi.org/10.1109/ACCESS.2020.2969288
11. Chen, J., Lin, X., Shi, Z., Liu, Y.: Link prediction adversarial attack via iterative gradient attack. IEEE Trans. Comput. Soc. Syst. **7**(4), 1081–1094 (2020). https://doi.org/10.1109/TCSS.2020.3004059
12. Chauhan, R., Heydari, S.S.: Polymorphic adversarial DDoS attack on IDS using GAN. In: 2020 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6 (2020). https://doi.org/10.1109/ISNCC49221.2020.9297264
13. He, X., Tong, G., Gao, W., Mi, X., Gao, P., Zhang, Y.: The method of adaptive gaussian decomposition based recognition and extraction of scattering mechanisms. In: 2018 12th International Symposium on Antennas, Propagation and EM Theory (ISAPE), pp. 1–4 (2018). https://doi.org/10.1109/ISAPE.2018.8634155
14. Zhao, X., Huang, M., Zhu, Q.: Analysis of hyperspectral scattering image using wavelet transformation for assessing internal qualities of apple fruit. In: 2012 24th Chinese Control and Decision Conference (CCDC), pp. 2445–2448 (2012). https://doi.org/10.1109/CCDC.2012.6244390
15. Loeb, I.: Lipschitz functions in constructive reverse mathematics. Log. J. IGPL **21**(1), 28–43 (2013). https://doi.org/10.1093/jigpal/jzs020
16. Hu, G.: Observers for one-sided Lipschitz non-linear systems. IMA J. Math. Control Inf. **23**(4), 395–401 (2006). https://doi.org/10.1093/imamci/dni068
17. Calliess, J.: Lipschitz optimisation for Lipschitz Interpolation. In: 2017 American Control Conference (ACC), pp. 3141–3146 (2017). https://doi.org/10.23919/ACC.2017.7963430
18. Demertzis, K., Tsiknas, K., Takezis, D., Skianis, C., Iliadis, L.: Darknet traffic big-data analysis and network management for real-time automating of the malicious intent detection process by a weight agnostic neural networks framework. Electronics **10**(7) (2021). https://doi.org/10.3390/electronics10070781. Art. no. 7
19. Cheng-Guo, E., Quan-Lin, L., Li, S.: The Shapley value of cooperative game with stochastic payoffs. In: The 26th Chinese Control and Decision Conference (2014 CCDC), pp. 1717–1722 (2014). https://doi.org/10.1109/CCDC.2014.6852446
20. Huafeng, X., Qiuhong, L.: The game theory analysis of risk share for PPP project based on Shapley value. In: 2010 2nd IEEE International Conference on Information Management and Engineering, pp. 112–115 (2010). https://doi.org/10.1109/ICIME.2010.5477813
21. Leon, F.: Optimizing neural network topology using Shapley value. In: 2014 18th International Conference on System Theory, Control and Computing (ICSTCC), pp. 862–867 (2014). https://doi.org/10.1109/ICSTCC.2014.6982527
22. Bao, X., Li, X.: Cost allocation of integrated supply based on Shapley value method. In: 2010 International Conference on Intelligent Computation Technology and Automation, vol. 1, pp. 1054–1057 (2010). https://doi.org/10.1109/ICICTA.2010.406
23. Zhang, L., Gao, Z.: The Shapley value of convex compound stochastic cooperative game. In: 2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), pp. 1608–1611 (2011). https://doi.org/10.1109/AIMSEC.2011.6010580
24. Messalas, A., Kanellopoulos, Y., Makris, C.: Model-agnostic interpretability with shapley values. In: 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA), pp. 1–7 (2019). https://doi.org/10.1109/IISA.2019.8900669

25. Are Correlations any Guide to Predictive Value? on JSTOR. https://www.jstor.org/stable/2985494?seq=1#metadata_info_tab_contents. Accessed 18 Apr 2021
26. Alippi, C., Disabato, S., Roveri, M.: Moving convolutional neural networks to embedded systems: the AlexNet and VGG-16 Case. In: 2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pp. 212–223 (2018). https://doi.org/10.1109/IPSN.2018.00049