Increasing the robustness of deep neural networks against adversarial attacks and solving other prominent problems in the application of machine learning

Theses of the PhD Dissertation

Alafandi Jalal

Pázmány Péter Catholic University
Faculty of Information Technology and Bionics
Roska Tamás Doctoral School of Sciences and Technology

Thesis Advisor:
Dr. Horvath Andras

Budapest, 2022

1 Introduction

Deep learning is an ubiquitous machine learning approach which has been successfully used in many applications to find a practical solution for complicated problems. I worked on many ideas across many topics during my PhD solving problems and enhancing the performance of numerous deep learning based applications. I will present four different issues surrounding the usage of neural network in practical applications.

- I adopted the idea of evaluating partial solutions from neural network and applied it to the mutation step of any genetic algorithm method.
- I created a novel algorithm to recover the adversarial samples in a way that the system can still make the correct decision in case of detection.
- I worked on a new loss function for neural network in the task of image segmentation where I incorporated the topological properties when comparing objects.
- I worked on enhancing the performance of deep learning models by reforming batch normalization layer where I filtered the out-of-proportion activations when calculating the distribution's statistic.

2 New Scientific Results

1. Thesis

I have introduced a new mutation variant in genetic algorithm with an additional parameter that can determine a mutation factor on the individual gene. I have demonstrated that with the proper tuning of the local mutation parameter the final accuracy of the investigated algorithmic setups was increased by 64% on the 256-Queens problem and 7% on the TSP problems with 254 cities. [1]

Genetic algorithm (GA), as the most prominent evolutionary algorithm, is a probabilistic and heuristic search approach to investigate encoded solutions, also known as a chromosome, in an iterative manner, which was successfully applied in various practical applications. The algorithm searches for the optimal solution relying on a fitness function which determines the quality of each solution candidate. In the traditional approach, selection of a position for mutation is a random process and its major goal is the exploration of the high-dimensional search space without taking the current

state of the chromosome into account. Optimal selection of the gene which will be modified requires a comprehensive knowledge of three different variables; the statistics of the inter and intra populations, the chromosomes as a function of time and the statistic of the genes' competences which is also described in the following equation 1.

$$\mathrm{PM}(\mathrm{P}op_{tij}) = \mathrm{F}(\mathrm{P}op_{qkl}^{l=1...\mathrm{M},k=1...\mathrm{N},q=1...t}, i, j) \tag{1}$$

PM calculates the probability of mutation for a given gene j and F is a function calculating the mutation rate of gene j taking into account all previous generations (q), chromosomes (k) and genes (l).

Our new mutation method, locus mutation, designate a different probability operator for each gene in a chromosome which can only be possible in partially solvable problems. The simplest model for gene level mutation is locus mutation as in Equation (2) where all generations and chromosomes have the same mutation rate but each gene has a different mutation rate which corresponds with the other genes.

$$\mathrm{PM}(\mathrm{P}op_{tij}) = \mathrm{F}(\mathrm{P}op_{til}^{l=1...\mathrm{M}}, j) \tag{2}$$

Algorithm 1 depicts GA with locus mutation. It is the same as a regular GA pseudo code, but we do have a newly introduced gene level mutation (G*ene*M*uation*()) which depends on partial fitness (P*artial*F*itness*).

Algorithm 1: Genetic algorithm main steps

```
1 Parameters: Population Size, Mutation Rate, Iteration Number
2 Results: Optimum
3 Population = population initialization
4 for Iteration Number 0 → i do
5     Fitness, Partial Fitness = Fitness(Population)
6     Population = Selection(Population, Values)
7     Population = Crossover(Population)
8     Population =
        Gene Muation(Population, Mutation Rate, Partial Fitness)
9 end
```

I have investigated two common problems, traveling salesman problem (TSP) and N-Queens problem. Locus mutation has resulted better solutions in all cases, regardless of the investigated parameters. Our approach has always surpassed the baseline solution where in average it yields 300% lower error than its traditional counterpart.

2. Thesis

I created a baseline for adversarial attack recovery and showed that it is a necessary extension of adversarial attack detection in practical problems. I demonstrated that using an algorithm based on counter-attacks can retrieve the original input classes with high confidence reaching 68% accuracy over MNIST, CIFAR-10, and a subset of ImageNet. [2]

Adversarial attacks can cripple any application with critical decision making tasks. Although many typical solutions were presented to make the neural network less vulnerable to the attacks by building better models or detecting the adversarial samples, we can't rely on their solution to deal with the malicious samples.

The most commonly applied defenses against adversarial attacks depend on one of three approaches: adversarial training , modifying the network or a detection-based approach. Most non-detection defenses are vulnerable to counter-counter attacks, rendering potential exposure and keeping the system in a state of being without any functioning protective shield. Detection-based defenses, on the other hand, can be continuously updated but lack the ability to steer the decision-making process obstructing the installation of any safety measure. Thus, a recovery algorithm has to be employed after the detection of adversarial attacks, providing robustness and resilience.

The high dimensionality of neural networks creates convoluted borders between all the classes, making a targeted adversarial attack highly possible. Taking into account the complexity of the curvature of the decision boundary, we hypothesize that the distance between the adversarial sample and the original class's manifold in the feature space of the decision boundary is smaller than the distance between the adversarial sample and any other classes' manifolds and, hence, all the adversarial samples and their counter attacks are in the vicinity of the original class manifold. We have implemented our idea, a class retrieval algorithm, on the notion of our former hypothesis to predict the original class by counter attacking the adversarial samples, targeting every class and then selecting the class with the minimum loss.

The counter attack can return the attacked sample to its original class easily since the adversarial sample is on the edge of the original class decision boundary. Due to the high dimensionality of the decision boundary curvature, there exist an intricate border between the manifold of each of the two randomly selected classes.

Our class retrieval algorithm for a detected adversarial attack is ex-

plained as a flowchart in Figure 1. $AdvImg$ is the adversarial image which has been selected by an adversarial attack detector filtering any potential adversarial threat. The neural network prediction for the label of the adversarial image is $AdvLab$, which is a misclassification according to our detector. We exterminate the possibility of the adversarial label, $AdvLab$, being the original class by setting its loss to infinity. The original retrieved label is the class with the minimum loss excluding the adversarial label where we used $argmin$ function to return the index of the smallest loss.

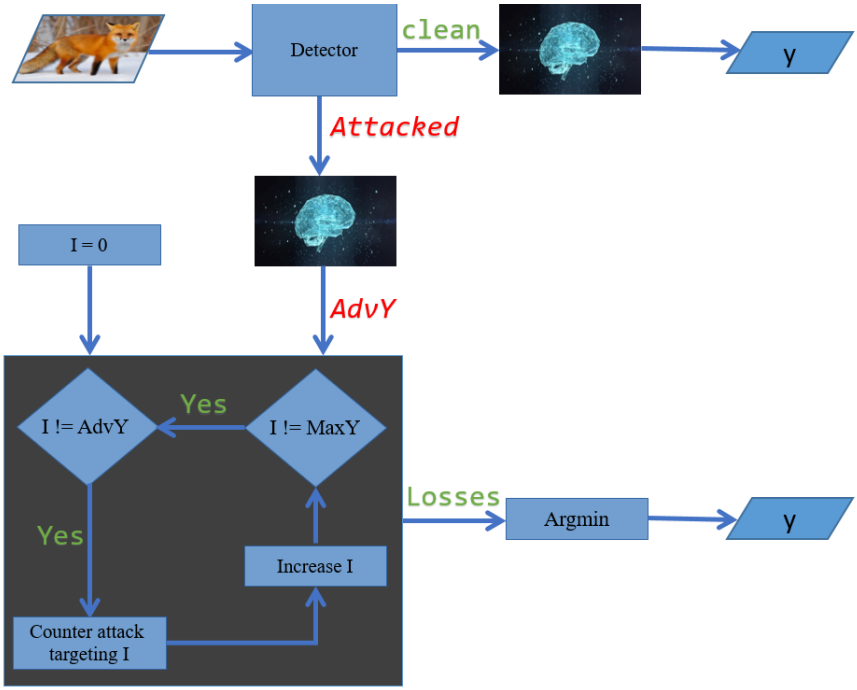

Figure 1: A flowchart explaining our class retrieval method starting from the input image until the output label where the $argmin$ process returns the index of the minimum loss, $MaxY$ is the number of classes and $AdvY$ is the adversarial label.

Our retriever is a self-evident addition to adversarial attack detectors and the combination of these two methods can enable the practical applicability of deep network even in case of attacks. I investigated four different adversarial attacks (PGD, MPGD, Deepfool, TPGD and PGDDLR) on three dif-

ferent datasets (MNIST, CIFAR10 and subset of ImageNet). The results are promising and consistent across all attacks and datasets, where the average accuracy is 67%.

3. Thesis

I demonstrated that a three-dimensional extension of wave loss can be employed as a loss function in the training of deep neural network in case of segmentation problems. I have shown that, with proper parameters setting, wave loss can increase the accuracy of segmentation with 2% on the MS-COCO dataset and cityscapes dataset as well. [3]

Image segmentation is applied in various tasks from medical imaging to self-driving cars. Many deep learning methods are applied which vary significantly depending on the selected architectures (U-Net, SegNet, Mask-RCNN and RetinaNet) or on the exact specification of the segmentation problem (semantic segmentation, instance segmentation or amodal segmentation), but all of these approaches require a metric which will compare the actual network output to the expected, ideal outcome or ground truth.

In current applications in almost all cases a pixel based distance is applied, where two images are compared to each other according to a given metric (like L1, L2 or Smooth-L1 distances). Similarly the outcome image and the ground truth can be considered as probability distributions and cross entropy can be applied to determine a distance between them, but none of these metrics take into account the position of the differences.

Almost all popularly used metrics such as cross entropy, Dice, Lovász or Tversky losses are area based metrics, where the area of the different regions matter, but their topologies are not considered.

Our metric depends on three not-independent measures: The area of the differences, the topology of the differences and the intensities, values of the differences. The output and ground-truth images can be imagined as two two-dimensional surfaces in three-dimensions. From this we can calculate the intersection and the union (which will also be two-dimensional surfaces) then the metric can be imagined as a three-dimensional wave propagating filling out the space between these two surfaces. A weight will be associated to every new voxel at each time step of the propagation and this four dimensional volume (the weighted sum of the three dimensional changes) will be called wave loss. Our goal was to differentiate between value and topology based differences and because of this the propagation speed of the wave is different in the z (intensity) and x, y (topological directions). A showcase is described in figure 2 illustrating the wave propagation.

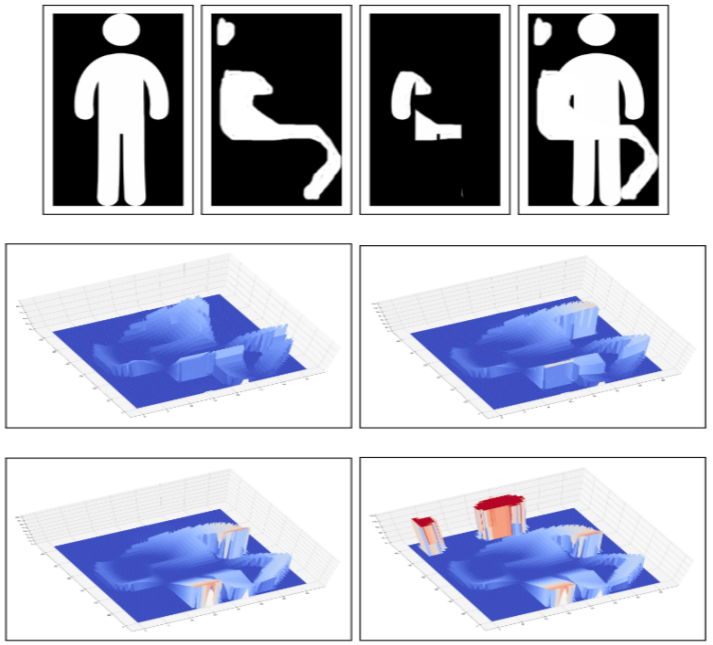

Figure 2: Illustration of the wave propagation. The first row depicts two possible binary input images (first and second images from the left) and their intersection and union (third and fourth images from the left). The last two rows depict four 3D versions of the wave metric in an increasing manner until reaching the union at iteration 100, 150, 300 and last iteration including not reached regions. During propagation further and further pixels will be incorporated in the loss function with higher and higher values. At the last step a high penalty will be assigned to all pixel which were not reached during propagation.

I have shown on a simple dataset, inspired by CLEVR that the same network can achieve better accuracy and faster convergence using Wave loss, than pixel based loss functions. I have also shown on a more complex tasks, that the overall accuracy of instance segmentation could be increased by 3% on MS-COCO and Cityscapes datasets using four different architectures modifying only the loss function from cross entropy to Wave loss.

4. Thesis

I Created a new method called filtered batch normalization which is a two steps batch normalization layer which filters out outlier values outside the σT *range of the mean value, where T is a parameter of the algorithm. This normalization can be used with arbitrary neural network architectures to eliminate the out-of-distribution activations and by this it improved the classification accuracy of the investigated networks by 5% on ImageNet. [4]*

Batch normalization became an important building block of neural networks in the past five years. It was demonstrated in various tasks that this method can accelerate network training and results higher test accuracy in practice, if mini-batch size is sufficiently high. The reduction of internal covariate shift, which is the imposed change in the input distribution of layers triggered by the updates of the preceding layers, is hypothesized to be the reason behind the beneficial effect of batch normalization. Batch normalization is a regularizer which reparametrizes the activations during training to make the optimization problem more stable by creating a smoother loss landscape and increasing the predictability of gradients rendering a robustness against exploding or vanishing gradients, hyperparameters and initialization sensitivity.

The aim of batch normalization is to result coherent output distributions in every iteration at each layer. Assuming that network activations follow a Gaussian distribution, which can be fully described by the mean and variance values, we can transform the output distribution of a network to zero mean and variance of one. After calculating the first two moments (mean and variance), batch normalization scales and shifts the activations using trainable parameters maintaining landscape flexibility.

Filtered batch normalization aims to design an algorithm that would filter out outliers from a distribution which can appear with low probability in mini-batches, but do not modify the mean and variance values if the input is a perfect Gaussian distribution without outliers. An example for an out-of-distribution activations in a common neural network architecture is shown in 3

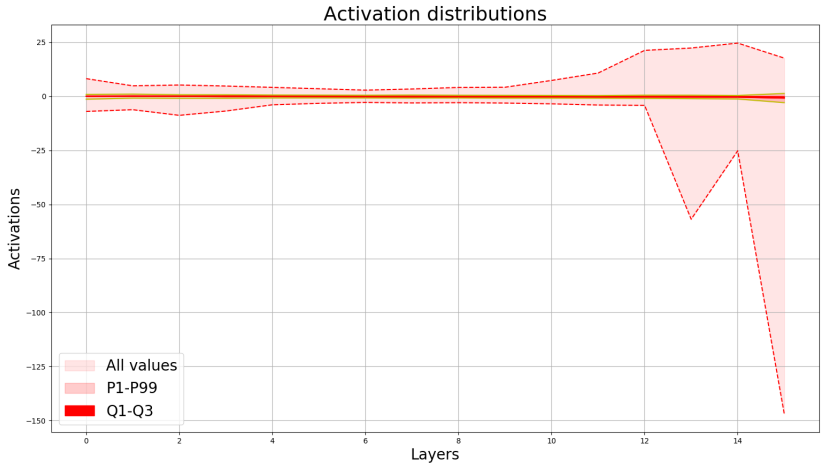

Figure 3: This figure depicts the distribution of the activations in a pretrained version of the VGG-16-BN architecture. The dashed lines display the maximum and minimum values in each layer, the golden lines contain 98% of the activations and 50% of them is in the solid red region. This demonstrates that although the data has zero mean and variance of one, it contains outliers especially in layers closer to the logit layer.

In the first step of the algorithm we calculate μ_i and σ_i values, but we do not use these values directly for normalization. We create a Gaussian candidate distribution $\hat{x'}_i$ which might contains outliers, but has zero mean and variance of one:

$$\hat{x'}_i = \frac{1}{\sigma_i}(x_i - \mu_i) \tag{3}$$

Based on this Gaussian candidate, we create a mask ($f(x_k)$) to select those values which are only less than T_σ distance from the mean value:

$$f(x_k) = \begin{cases} 1 & \text{if } -T_\sigma \leq \hat{x'}_k \leq T_\sigma \\ 0 & \text{if } \hat{x'}_k < -T_\sigma \vee T_\sigma < \hat{x'}_k \end{cases} \tag{4}$$

T_σ is a hyperparameter of the algorithm and the performance of the algorithm does not depend heavily on its value. As it can be seen from the previous example, $T_\sigma = 7$ can filter out most outliers in the data.

We use this mask to calculate the mean and the variance only including those which are not considered outliers (which are inside the $\pm T_\sigma$ band of the mean value).

$$\mu'_i = \frac{1}{\sum\limits_{k \in S_i} f(x_k)} \sum_{k \in S_i} f(x_k) x_k \tag{5}$$

$$\sigma'_i = \sqrt{\frac{1}{\sum\limits_{k \in S_i} f(x_k)} \sum_{k \in S_i} f(x_k)(x_k - \mu'_i)^2 + \epsilon} \tag{6}$$

μ'_i and σ'_i values are used to transform the activations which can be calculated similarly to batch normalization:

$$y'_i = \gamma \frac{(x_i - \mu'_i)}{\sigma'_i} + \beta \tag{7}$$

The additional hyperparameter of the algorithm T_σ can be tuned fairly easily. I also would like to emphasize that this approach can be used together with any other normalization method.

Our empirical results show that we can create more coherent output distributions in neural network layers by removing these outliers before mean and variance calculation, which results faster convergence and better overall validation accuracies. I investigated filtered batch normalization over three image classification datasets and one image segmentation dataset achieving a better confidence in all cases, 1% to 5% higher scores.

Throughout my PhD, I also worked on other interesting topics which didn't yield much enhancement to the baseline. I also worked on another two interesting topics, protecting neural network intellectual property right and utilizing inter-data correlation for data compression in autoencoder, which resulted two conference publications [5] [6].

Publications

[1] J. Al-Afandi and A. Horváth, "Adaptive gene level mutation," *Algorithms*, vol. 14, no. 1, 2021.

[2] J. Al-afandi and H. András, "Class retrieval of detected adversarial attacks," *Applied Sciences*, vol. 11, no. 14, p. 6438, 2021.

[3] J. Al-Afandi and A. Horvath, "Application of the nonlinear wave metric for image segmentation in neural networks," in *CNNA 2018; The 16th International Workshop on Cellular Nanoscale Networks and their Applications*, pp. 1–4, VDE, 2018.

[4] A. Horváth and J. Al-Afandi, "Filtered batch normalization," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 6778–6785, IEEE, 2021.

[5] K. Szentannai, J. Al-Afandi, and A. Horváth, "Preventing neural network weight stealing via network obfuscation," in *Science and Information Conference*, pp. 1–11, Springer, 2020.

[6] J. Al-afandi and A. Horváth, "Multi-instance conditional autoencoder a data driven compression model for strongly correlated datasets," in *2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–5, IEEE, 2021.