Black-box Evolutionary Search for Adversarial Examples against Deep Image Classifiers in Non-Targeted Attacks

Štěpán Procházka
Institute of Computer Science
Czech Academy of Sciences
Prague, Czech Republic
prochazka.stepan@gmail.com

Roman Neruda
Institute of Computer Science
Czech Academy of Sciences
Prague, Czech Republic
roman@cs.cas.cz

***Abstract*—Machine learning models exhibit vulnerability to adversarial examples i.e., artificially created inputs that become misinterpreted. The goal of this paper is to explore non-targeted black-box adversarial attacks on deep networks performing image classification. The original evolutionary algorithm for generating adversarial examples is proposed that employs a guided multi-objective search through the space of perturbed images. The efficiency of attacks is validated by experiments with the CIFAR-10 data set. The experimental results verify the usability of our approach against deep convolutional neural networks.**

***Index Terms*—deep learning, image classification, adversarial patterns, evolutionary algorithms**

I. Introduction

Deep learning image classification solutions reports vulnerability to adversarial examples i.e., artificially created inputs that become misinterpreted. It has been showed that carefully crafted perturbations added to images may successfully lead to model failure, while being unnoticeable for the human observer, or a different model. The issue of vulnerability to adversarial attacks does not apply only to the task of image classification, but it also concerns speech recognition, particularly the use of virtual personal assistants; or image segmentation and object detection on which autonomous vehicles depend. It is of great importance to study those vulnerabilities, attacks and corresponding defenses to improve reliability of such solutions.

The goal of this work is to explore adversarial attacks in image classification by deep networks in the black-box scenario. We want to assess the suitability of evolutionary algorithms as a generative procedure for finding adversarial examples. To this end, we propose a multi-objective evolutionary algorithm working on image data, and searching for adversarial examples that are misclassified by the machine learning model while still retaining visual similarity to the original image. The main constraint in the black-box attack scenario is the impossibility to access the machine learning model parameters, or even consider any knowledge about the attacked model. The attacking algorithm is only able to query the model for a classification of particular image. Thus, previously successful gradient-based attack methods are not applicable, unless a some kind of surrogate model solution is employed. Our method, on the other hand, satisfies the black-box attack constraints naturally.

The structure of this work is as follows. The following section II introduces the deep models and image data set that are referred to and used in our experiments, as well as several concepts related to adversarial examples. Related work is presented in section III. The original contribution of this work – the algorithm for generating adversarial examples – is described in section IV. Experiments validating the performance of our algorithm are presented in section V. Finally, a discussion of the results and possible future work directions are presented in section VI.

II. Preliminaries

A. Deep Image Classifiers

The task of *image classification* lies in assigning one and only one label l from the set of possible output labels L to each input image I from the space of all possible images $\mathcal{I}$ of which we often think as a unit hypercube i.e., $[0, 1]^{h \cdot w \cdot c}$ with h, w, c being input images height, width and number of channels (depth) respectively. Task with input varying in shape can be transformed to the canonical classification task by adding a preprocessing step which unifies shape e.g., reshaping using bilinear interpolation or conversion to common color space (see [1]), or by building a multitude of solutions – each targeted at specific shape of input data.

Due to the need to classify real world imagery i.e., data with substantial amount of noise, lacking quality or having non-trivial composition (such as multiple objects in the scene, varying viewing angles or conditions, or heterogeneous background), the datasets of real world photographs were created. Those datasets range from collections of thousands of low resolution thumbnails in dozens of classes to millions of reasonably large pictures in hierarchical systems of classes.

For the experiments in this work we have chosen the CIFAR dataset of downsized color photographs. The CIFAR dataset comes in two versions, namely CIFAR-10 and CIFAR-100,

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

named after number of classes present. Both datasets share the same example shape i.e., 32×32 (rectangular color image) and dataset layout comprising of 50 000 training and 10 000 testing images with balanced class occurrence. CIFAR-10 consists of following classes – *airplane, automobile, bird, cat, deer, dog, frog, horse, ship* and *truck* (see fig 1), while CIFAR-100 introduces hierarchical system of 20 superclasses e.g., *fish, small mammals, tress, large carnivores*, each subdivided into 5 subclasses e.g., *fish – aquarium fish, flatfish, ray, shark, trout* (see [2]).

The advent of deep learning and its soaring popularity has been boosted by outstanding results achieved with deep learning models, beating then state of the art methods by a large margin. Arguably *AlexNet* was the first architecture to show the potential of deep convolutional neural networks, followed by gradually deeper architectures *VGGNet, ResNet*, etc. ResNet was the first architecture to tackle the issue of vanishing gradients. Residual shortcuts, connections bypassing convolutions, enabling gradient flow have been employed. With that improvement, *ResNet-152* model [3], with 152 layers and parameter count of 60.2 millions, scored top-5 error of 3.6% on ImageNet [4] and 6.43% on CIFAR-10. It uses 3×3 convolutions, average pooling, *ReLU* activation and dropout after dense layers. Moreover batch normalization between each convolution and activation is used. It is trained using SGD with momentum. Further exploiting the performance gain of residual shortcuts usage, authors of [5] created the *DenseNet* architecture. WideResNet [6] was another extension of the ResNet architecture. The performance gain of residual shortcuts usage was further exploited e.g., by authors of DenseNet [5] and authors of WideResNet [6]. Contrary to the original ResNet, the WideResNet architecture sacrifices network depth (i.e. number of layers) for network width (i.e., the size of layers). The authors show that this is not only more computationally efficient but also more performant in terms of accuracy. WideResNet with 28 layers and widening factor k=10 reaches SOTA results with 3.89% error on CIFAR-10.

B. Adversarial Examples

Performance of deep learning models is not perfect and certain amount of input data gets misclassified naturally due to the insufficient accuracy of models. However, the paper [7] showed that misclassification can be achieved artificially by adding small perturbations designed to fool the model to the previously correctly classified examples. Those inputs are called *adversarial examples*. We will focus on adversarial examples for image classification task.

The task of generating adversarial example for input x and model M_θ, such that $M_\theta(x) = l$ is the correct label, lies in finding the perturbation η, such that $M_\theta(x+\eta) = M_\theta(x') = l'$, with l' being target class for the adversarial attack.

We can characterize adversarial attacks by the following, mutually independent properties i.e., *specificity, scope, adversary's knowledge* and *perturbation constraints*.

The *specificity* of adversarial attack is defined by the target class l'. In case of *targeted* attack, the target class l' corresponds to one specific classification category. On the other hand, in case of *non-targeted* attack, l' corresponds to all but the ground truth class l and adversarial example is than any $x' = x + \eta$ such that $M_\theta(x') \neq l$.

The *scope* of adversarial attack lies in number of original examples to be attacked by single η. *Singleton* attack stands for attack on single example x such that $M_\theta(x+\eta) = l'$. *Multi* attack stands for attack on arbitrary number of examples X of the same class such that $\forall x \in X : M_\theta(x+\eta) = l'$. Finally, *universal* attack seeks single η such that $\forall x : M_\theta(x) = l \implies M_\theta(x+\eta) = l'$

We distinguish between three cases of *adversary's knowledge*. The *white-box attack* assumes full knowledge of targeted model i.e., architecture, weights and training data as well as parameters of the optimizer being used for training. On the other hand, the *black-box attack* assumes no additional information about targeted model, apart from the classification predictions and respective probabilities. Finally, the *surrogate attack* treats the targeted model as a black-box, yet has full access to substitute model i.e., model sharing some characteristics of target model e.g., task it solves, training data, architecture, output probability distribution, etc.

Regarding the *perturbation constraints* of the attack, we distinguish the following three cases. *Unconstrained* attack with no limitations on perturbation η, *constrained* attack, with limited intensity of perturbation η allowed, with respect to some measure, and finally the *optimized* one, where the perturbation intensity is minimized. The $l1$ and $l2$ norms, or *PASS* by [8] and *SSIM* by [9] – visual similarity measures, are frequently used intensity measures.

III. Related work

The research of adversarial attacks against machine learning models has become popular in last years and several methods have been proposed, see [10] for overview. Thanks to the inherent property of deep learning models – differentiability, gradient based methods are easy to be used and perform well in white-box and reasonably well in surrogate scenarios. More of those methods have been invented, namely the *L-BFGS* using Broyden-Fletcher-Goldfarb-Shannon optimization algorithm, fast gradient sign method and its modifications, or feature adversary attack.

The fast gradient sign method (FGSM) computes gradients of loss function of the target model with respect to the target example x and class l'. The sign of gradients is taken and multiplied by ϵ, the step size, and subtracted from x (targeted attack) – this way the target image is moved in the direction of rising target class prediction (1). In case of non-targeted attack, gradients are computed with respect to source class l and, instead of subtraction, we use addition to move the target image in direction of falling source class prediction.

$$x' = x - \epsilon \operatorname{sgn}\left(\nabla_\theta L\left(M_\theta(x), l'\right)\right) \tag{1}$$

The FGSM can be used iteratively with $x_t = x'_{t-1}$. Optional clipping after each step may take place in case of constrained

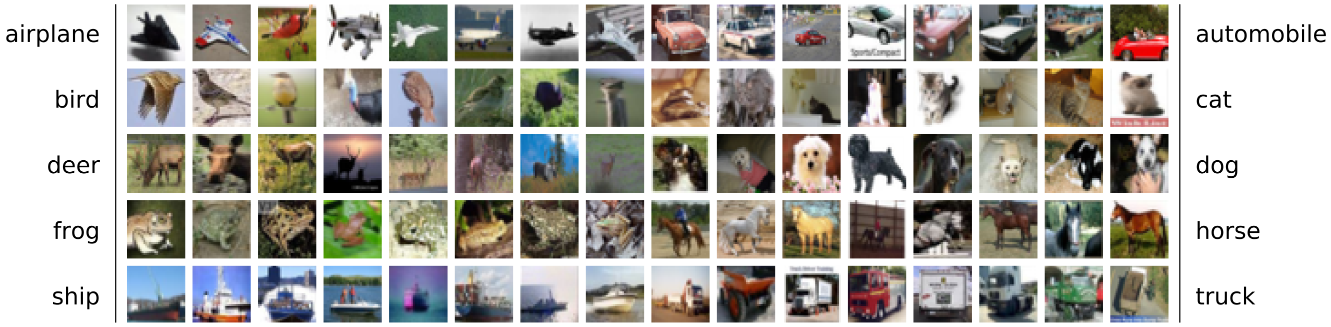


Fig. 1. CIFAR 10 samples

attack. Versions of FGSM with momentum exists – generating the adversarial example the same way as the training of deep learning models optimizes the parameters.

Apart from methods based on backpropagation, the usage of deep generative models for adversarial example generation has been researched. Authors of [11] trained generative adversarial network (generator G) on target model task dataset, mapping random latent space to images and inverter I, mapping images to latent space of generator G. The adversarial examples are generated by finding z' close to $z = I(x)$, such that $M_\theta\left(G(z')\right) = l'$. This approach generates adversarial examples more natural to human observer as we are perturbing the hidden representation rather than the final output.

Regarding the black-box adversarial attacks for deep learning models, only a few methods has been proposed so far. Authors of [12] performed a successful one-pixel attack using differential evolution algorithm, they demonstrated a vulnerability of image classifiers to extreme change of a single pixel of the image. In [13], authors suggest an approach converting black-box attack to surrogate attack, by synthesizing dataset for the surrogate model to be trained on. The work [14] proposes to perform a local-search of the neighborhood of targeted image to estimate the gradient of targeted model. This estimate is later use to guide the generation of adversarial example. The former approach is later extended by authors of [15], who employ natural evolutionary strategies to estimate target model gradients with respect to targeted image. The adversarial example is generated using the gradient descent algorithm on the estimated gradient. Finally, in [16], authors used genetic algorithms to perform adversarial attack mostly on a variety of machine learning models including SVMs, kernel networks and multilayer perceptrons.

IV. Our Solution

In our solution, we propose genetic algorithm to perform search on space of images to find adversarial examples for the given model. We do not assume anything in particular about targeted model, its architecture, dataset or training process, contrary to white-box or surrogate methods. We treat the model only as a function classifying the given image (input) into probability distribution over the set of labels (output). In this section we will describe our solution in detail, focusing on data representation and algorithms used.

The genetic algorithm is a population based search method, working with a set of individuals – encoded solutions to the given problem. Assume a task with objective O on space S. Then a population P is a subset of space to be searched ($P \subset S$), with individual i being an element of population ($i \in P$), hence element of search space S i.e., representing a possible solution. Fitness function $F : i \mapsto \mathbb{R}$ is a real valued measure of fitness depending on individual, often function of the objective function O of the task to be solved i.e., $F : O(i) \mapsto \mathbb{R}$. As an operator $\text{Op} : P_{in_1}, P_{in_2}, \ldots, P_{in_n} \mapsto P_{out}$ we define a function converting one or more populations to a new one, often in element-wise (per individual) manner. *Epoch* is then one step of the genetic algorithm consisting of application of the sequence of operators to current population P_t, producing new population $P_{t+1} = \text{Op}_n\left(\text{Op}_{n-1}\left(\ldots \text{Op}_1(P_t)\right)\right)$.

The ability of genetic algorithms to effectively search given space lies in application of suitable operators. Those operators can be divided into three categories – *reproduction* operators e.g., *cross-over* which combines several individuals into new one; *mutation* operators, which add random perturbations to individuals, with the idea of exploring genes not present in current population; and finally *selection* operators mimicking the natural selection based on fitness of individuals.

Our approach encodes the image as a *3D* matrix to maintain spatial relations of the data – images. Those relations are further exploited by selected operators, namely cross-over. The suitability of this change is supported by the fact, that vast majority of deep learning image classifiers is based on use of convolutional layers, which extracts the spatial context of processed data. Our individual is thus defined as a matrix $i \in [-1, 1]^{h \times w \times c}$ and represents perturbation η. While a use of zero centered binomial distribution scaled by factor of $\frac{1}{255}$ is possible to sample initial population, in our approach we found that simple zero initialization yields satisfactory results.

In our solution we use two point cross-over, which swaps randomly selected rectangular regions i.e., image crops of parent individuals, to produce offspring (see fig 2). As a mutation operator, we use biased mutation with normal distribution (Mut) followed by quantization to multiples of 1/255 (Quant).

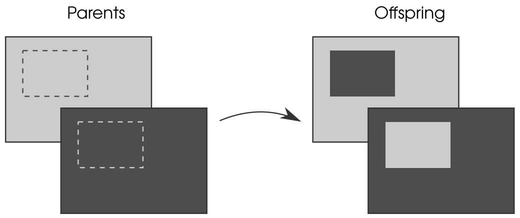


Fig. 2. Two-point cross-over on images

As a result, such approach mimics shifting the intensity of the pixel by several color shades to either direction. The normal distribution biased mutation (Mut) of individual i, with normal distribution $N(\mu, \sigma^2)$ and gene change probability (p_{gene}), has the form of (2).

$$\text{Mut}(i)_{x,y} = \begin{cases} i_{x,y} + s, & \text{with probability} \quad p_{gene}, \\ i_{x,y}, & \text{with probability} \quad 1 - p_{gene}, \end{cases} \tag{2}$$

where $s \sim N(\mu, \sigma^2)$.

Following the mutation, each individual gets quantized to multiples of $1/255$ corresponding to the discrete nature of 8-bit depth image space. Quantization of the whole population quantizes each individual separately.

$$\text{Quant}(i)_{x,y} = \frac{\lfloor 255 \cdot i_{x,y} \rfloor}{255} \tag{3}$$

Finally, the clipping takes place before evaluation of objectives to maintain the perturbed image (i.e., the individual combined with the input image) in the range of a $[0, 1]$ cube (the searched space). Clipping (Clip) with lower bound l and upper bound u for input x' corresponds to (4).

$$\text{Clip}_{(l,u)}(x')_{x,y} = \max\left(l, \min(u, i_{x,y})\right) \tag{4}$$

For the optimization we use two objectives – the score of source class prediction, which is minimized; and the *SSIM* measure of perturbation intensity between the input image and its perturbed counterpart, which is being maximized. Another option would be to use the $l2$ norm, but we have chosen the SSIM since it accounts for perceptual visual similarity. Apart from that, we suggest the use of clipping on the intensity of perturbation if it gets below certain admissible value c_{bound} i.e., taking the $\max(c_{\text{bound}}, \text{SSIM}(x', x))$ as a perturbation intensity, combining the *constrained* and *optimized* adversarial attack paradigm.

Naturally, the selection operator performs a multi-objective optimization, implemented by the non-dominated sorting algorithm. For that matter, we choose the *NSGA-II* with crowding distance (see algorithm 1), which uses concept of Pareto-optimality to construct a so called *non-dominated fronts* – groups of mutually non-dominated individuals. Furthermore the NSGA-II employs secondary sorting algorithm (in our case crowding distance) allowing for more stable selection of individuals (see [17] for more). We assume this approach superior to average of objectives for the issues with different scaling of prediction error and perturbation intensity. The new population is selected from the union of parents and offspring of the current generation.

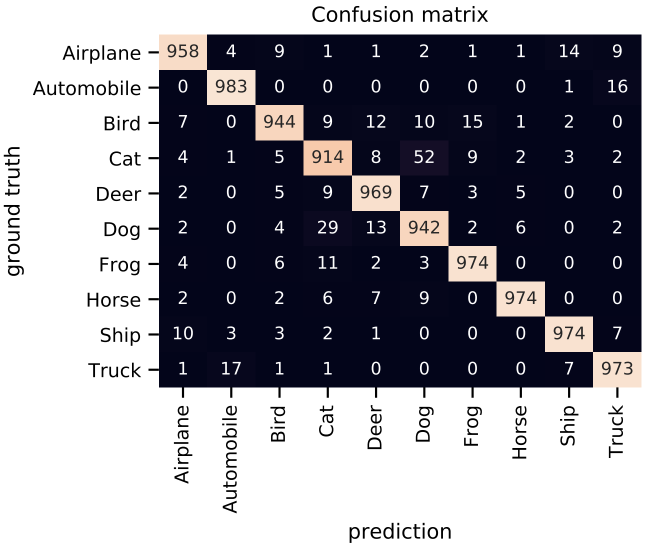
Algorithm 1 NSGA-II operator

```
$P_{in} \leftarrow$ input population
$s_{out} \leftarrow$ target size of selected population

$P_{out} \leftarrow$ empty list
**while** $|P_{out}| < s_{out}$ **do**
    $f \leftarrow$ pop non dominated individuals from $P$
    **if** $|P_{out}| + |f| \leq s_{out}$ **then**
        extend $P_{out}$ with $f$
    **else**
        sort $f$ by crowding distance
        extend $P_{out}$ with first $s_{out} - |P_{out}|$ elements of $f$
    **end if**
**end while**
**return** $P_{out}$
```

Algorithm 2 Evolutionary generated adversarial examples

```
$M_\theta \leftarrow$ targeted model
$x \leftarrow$ targeted image, of shape $h \times w$, $l \leftarrow$ source class
$s_{pop} \leftarrow$ population size,
$c_{\text{bound}} \leftarrow$ perturbation intensity clipping value (optional)

$P \leftarrow \mathbf{0}^{h \times w}$
**while not** $\exists i \in P : M_\theta(i) \neq l$ **do**
    $P_{old} \leftarrow P$
    shuffle $P$
    $P \leftarrow \text{Xover}(P)$
    $P \leftarrow \text{Mut}(P)$
    $P \leftarrow \text{Quant}(P)$
    objectives evaluation $\forall x' \in \{\text{Clip}_{(0,1)}(x + i) ; i \in P\}$ :
    $(M_\theta(x')[l'], \text{SSIM}(x', x))$
    $P = \text{NSGA-II}([P, P_{old}], s_{pop})$
**end while**
```

The whole proposed strategy towards generating adversarial examples takes form of an algorithm 2. After initializing population with zeros, the main loop of the genetic algorithm is entered. The loop comprises of the shuffling of the parents, the two-point cross-over and the normal distribution mutation, followed by the quantization. Next, the objectives of evolved offspring are evaluated, followed by the selection, taking in both parent generation and the evolved offspring. The fittest individuals are selected for the next iteration. The algorithm stops when adversarial example is found or maximum number of epochs limit is reached. We provide our implementation as an open-sourced project under MIT license – see github repository [18].



ground truth \ prediction	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	958	4	9	1	1	2	1	1	14	9
Automobile	0	983	0	0	0	0	0	0	1	16
Bird	7	0	944	9	12	10	15	1	2	0
Cat	4	1	5	914	8	52	9	2	3	2
Deer	2	0	5	9	969	7	3	5	0	0
Dog	2	0	4	29	13	942	2	6	0	2
Frog	4	0	6	11	2	3	974	0	0	0
Horse	2	0	2	6	7	9	0	974	0	0
Ship	10	3	3	2	1	0	0	0	974	7
Truck	1	17	1	1	0	0	0	0	7	973

Fig. 3. Confusion matrix of the model (each cell represents the number of images with ground truth label (row) classified as predicted label (column)).

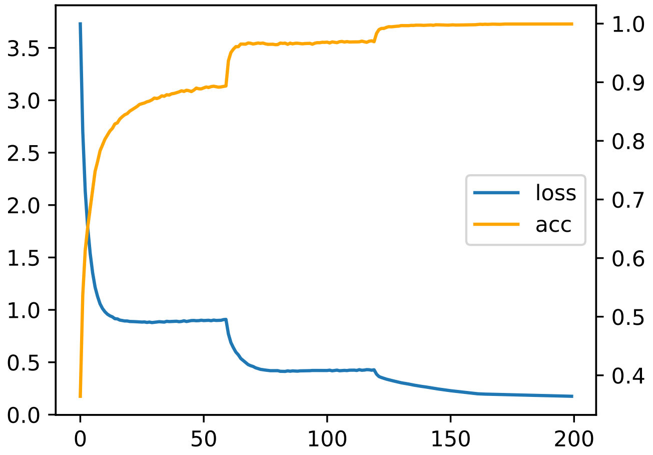


Fig. 4. Training metrics. Cross-entropy loss (y-axis on the left) and classification accuracy (y-axis on the right)

V. Experiments

A. Framework

In order to empirically assess the performance of proposed solution we have carried out an experiment on the CIFAR-10 dataset. This particular dataset was chosen for it is simple enough to enable thorough examination and measurements of results on statistically significant number of examples (whole test set), yet more complex than MNIST and the like [19].

We use the *WideResNet* of depth 28 with widening factor $k = 10$ with dropout as a target model. It was trained according to the strategy proposed by its authors, using SGD with Nesterov momentum and cross-entropy loss. The initial learning rate is set to 0.1, weight decay to 0.0005, dampening to 0, momentum to 0.9, and minibatch size to 128. The learning rate drops by 0.2 at 60, 120 and 160 epochs, and we train for total 200 epochs. We use standard input preprocessing – mean subtraction and standard deviation division with channel-wise precomputed moments on training data. Training examples are augmented using random left-right flips and random crops from originals padded by 4 px on each side with their reflection.

Training is carried out on CIFAR-10 training set. The test accuracy of the model yields 96.05%.

Detailed decomposition of model accuracy is described by a confusion matrix at fig 3.

B. Results

We examine the performance of our proposed solution based on the use of genetic algorithms. The hyperparameters of the approach are set as follows: the population size as well as the maximal number of epochs is equal to 256, the cross-over takes place with probability $p_{\text{xover}} = 1$, the biased mutation uses normal distribution with $\mu = 0, \sigma^2 = 4/255$ (shift of 2 color shades to either direction on average), with $p_{\text{mut}} = 1$ the probability of individual being mutated and $p_{\text{gene}} = 1$ the probability of gene mutation. No perturbation intensity clipping is used in our experiments, and the SSIM norm is used.

We use the non-guided random search as a baseline black-box attack. For each input image we generate random perturbations using normal distribution followed by quantization. The parameters are chosen so that the approach is directly comparable to our solution. Namely, the number of perturbations (which is equal to target model evaluations) is set to 256^2 (number of epochs × population size), and normal distribution with parameters $\mu = 0, \sigma^2 = 0.03$.

We measure the success rate of black-box attacks on the examined architecture on the test set of CIFAR-10 dataset. As a successful attack we perceive generating adversarial example whose predictions on target model yield over 0.5 score in non-source class (the model is confident in misclassification). The overall success rate of attacks is 43.5% for the random baseline, and 98.6% for our solution, which is not only quantitatively better (approximately 2 times), but also qualitatively different – our approach is reaching near-perfect performance. The success rate of our solution is shown in detail using heatmaps (see fig 5). We are providing two sets of results, we denote them *first* and *best* – The *first* being measurements from the first epoch in which the algorithm performed successful attack, while the *best* is the epoch in which the model produced successful attack with the lowest intensity of the perturbation (compared to the whole 256 epoch long run). Note that the overall and per-source class success rate is the same for both measurements, and it differs only in the resulting class distribution.

Apart from the success rate measurements, we also provide a comparison of time complexity of the attacks (see fig 8). The epoch count heatmaps show average number of evolutionary algorithm epochs needed to perform a successful attack. We

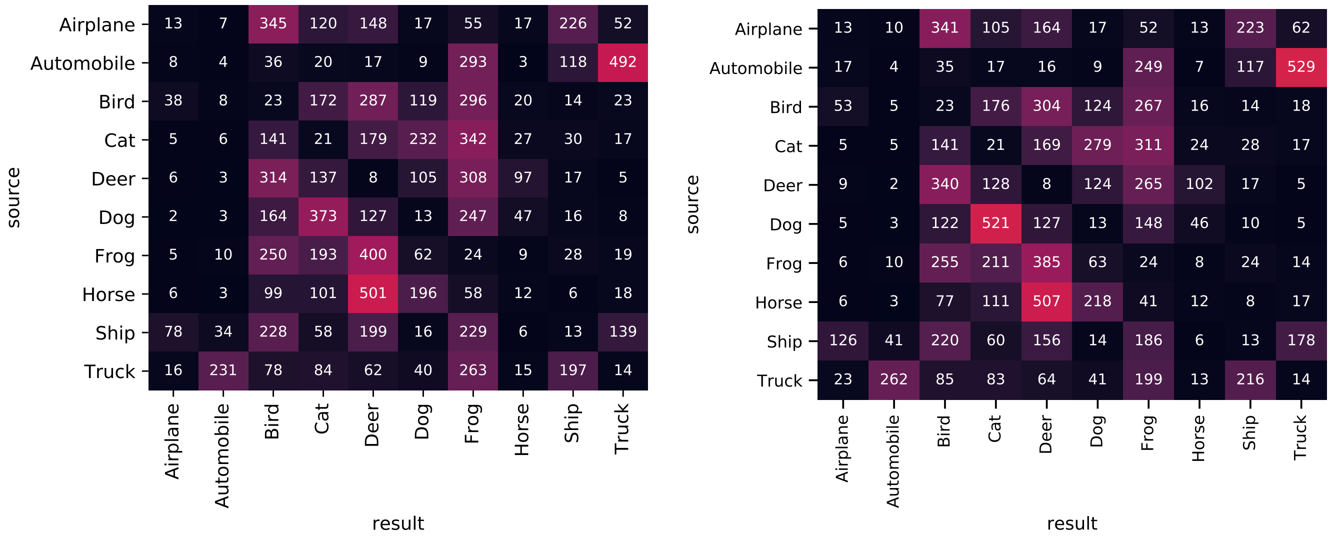


Fig. 5. Attacks success rate. Each cell represents the number of examples from the source class (row) which got misclassified by given model as a result class (column), when using our method. Diagonal holds the numbers of unsuccessful trials. Left plot shows the *first* results, right plot shows the *best* results.

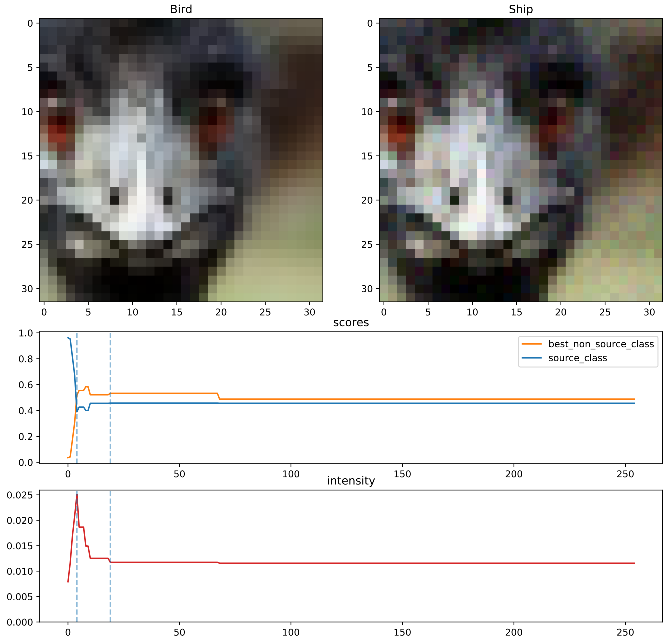


Fig. 6. Visual comparison of adversarial examples: Bird misclassified as a Ship. Bottom plots show progression of objectives with vertical dashed lines marking *first* and *best* epochs (from left to right).

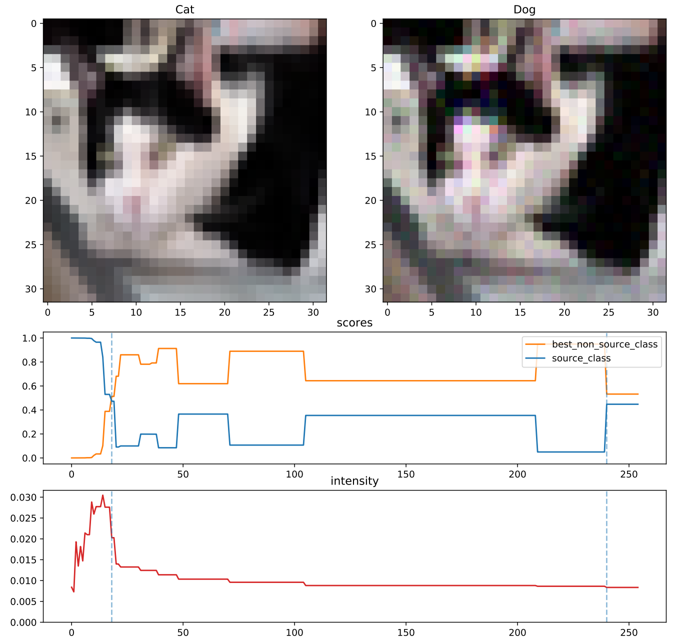
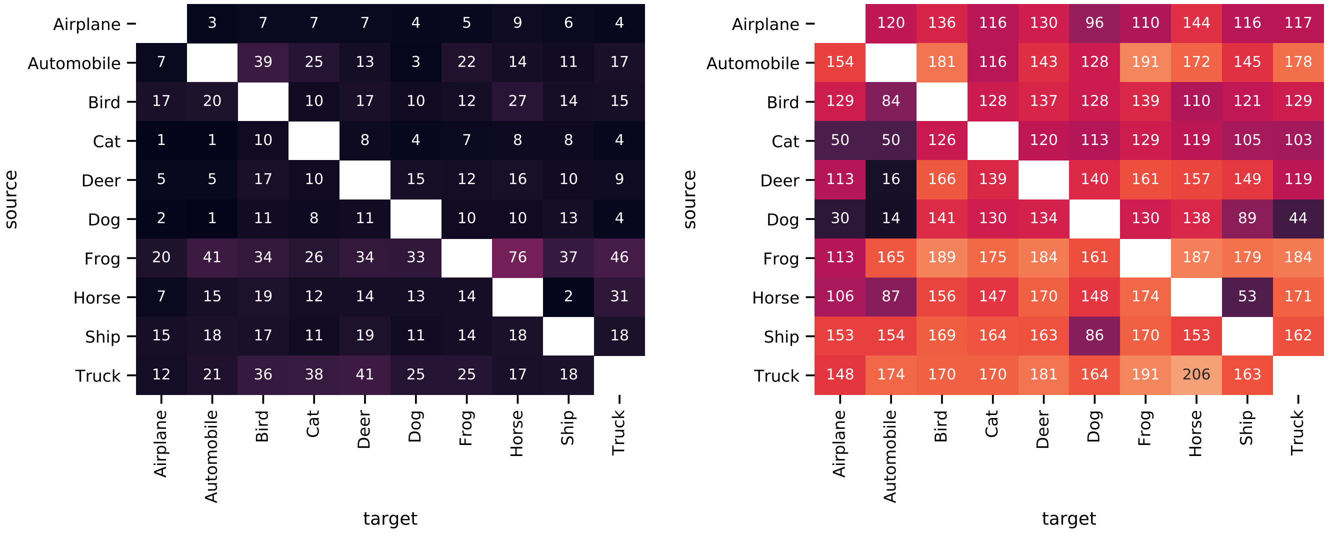


Fig. 7. Visual comparison of adversarial examples: Dog misclassified as a Horse. Bottom plots show progression of objectives with vertical dashed lines marking *first* and *best* epochs (from left to right).

can see that in many cases the algorithm is capable of generating the first successful attack in only a few epochs – half of the *first* successful attacks is reached under 8 epochs (2048 model queries) (median) and 95 per cent of them under 56 epochs (approximately 14000 model queries). On the other hand, generating the *best* possible (i.e., the least intense) perturbation our method is capable of, takes significant amount of epochs (the median time is 174 epochs, while 95 per cent of them is reached in 250 epochs). A detailed visualization of algorithm results and objectives progress throughout its run are illustrated on fig. 6 and fig. 7.

VI. Conclusions

In this paper, we have proposed, described and evaluated the method for generating non-targeted adversarial examples against deep neural networks in the black-box scenario. The attacks were performed on the CIFAR-10 data set against the state of the art deep image classifier. Our method was very



	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane		3	7	7	7	4	5	9	6	4
Automobile	7		39	25	13	3	22	14	11	17
Bird	17	20		10	17	10	12	27	14	15
Cat	1	1	10		8	4	7	8	8	4
Deer	5	5	17	10		15	12	16	10	9
Dog	2	1	11	8	11		10	10	13	4
Frog	20	41	34	26	34	33		76	37	46
Horse	7	15	19	12	14	13	14		2	31
Ship	15	18	17	11	19	11	14	18		18
Truck	12	21	36	38	41	25	25	17	18	

	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane		120	136	116	130	96	110	144	116	117
Automobile	154		181	116	143	128	191	172	145	178
Bird	129	84		128	137	128	139	110	121	129
Cat	50	50	126		120	113	129	119	105	103
Deer	113	16	166	139		140	161	157	149	119
Dog	30	14	141	130	134		130	138	89	44
Frog	113	165	189	175	184	161		187	179	184
Horse	106	87	156	147	170	148	174		53	171
Ship	153	154	169	164	163	86	170	153		162
Truck	148	174	170	170	181	164	191	206	163	

Fig. 8. Mean attack epoch count – each cell represents average number of generations the genetic algorithm needed to find an adversarial example for the example from the source class (row) to the result class (column), the lower is the number, the better. (The diagonal is empty, measuring the mean epoch count for the unsuccessful attack (same source and result class) does not make sense.) Left plot shows the *first* results, right plot shows the *best* results.

successful, as it managed to provide non-targeted adversarial examples for all source classes.

The experiments were carried out on significant number of examples compared to the dataset size. The chosen methodology perceived adversarial attack as successful only if the prediction of the non-source class overcomes one half in the model output probability distribution, posing more difficult task than aiming for simple maximum among all output classes. Moreover, the intensity of the perturbation has been minimized for, again rising the difficulty of finding adversarial example. Even in those hardened conditions, our method yielded satisfactory results in terms of attack success rate, attack complexity and perturbation intensity.

Being a black-box method, the targeted model does not need to be an end-to-end differentiable one, contrary to white-box attack methods relying on such properties, thus, our approach can be used even against complex pipelines of models used for image classification, which is often the case in production environments.

The results of the experiments suggest that the WideResNet architecture is relatively vulnerable to adversarial attack. In our previous work we observe lower vulnerability of the models with worse classification performance that were characterized by the lack of residual connections. It can be speculated that extensive feature reuse by means of residual connections can worsen the model resiliency to attacks. Yet, this hypotheses remains one of the possible future work topics.

We also recognize the potential suitability of our method to attack convolutional networks, for they internally operate on patches of the input image, as the cross-over operator combine individuals by swapping their rectangular crops. We believe that this enables the method to combine promising regions of perturbation to gradually form a successful adversarial example.

We believe that apart from the study of adversarial attacks, study of defenses against those is just as important, if not more. We assume that improvements in dataset augmentation as well as employing methods of adversarial attacks to the process of training may lead to significant improvements in resilience of deep learning models. Apart from these, the study of feature reuse and its consequences with respect to adversarial attacks may offer interesting insights.

Acknowledgments

This research has been partially supported by Czech Science Foundation project no. GA18-23827S.

References

[1] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. Berlin, Heidelberg: Springer-Verlag, 2010.

[2] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf

[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009. [Online]. Available: http://www.image-net.org/papers/imagenet_cvpr09.pdf

[5] G. Huang, Z. Liu, and K. Q. Weinberger, “Densely connected convolutional networks,” *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[6] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *CoRR*, vol. abs/1605.07146, 2016. [Online]. Available: http://arxiv.org/abs/1605.07146

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *CoRR*, vol. abs/1312.6199, 2013. [Online]. Available: http://arxiv.org/abs/1312.6199

[8] A. Rozsa, E. M. Rudd, and T. E. Boult, “Adversarial diversity and hard positive generation,” *CoRR*, vol. abs/1605.01775, 2016. [Online]. Available: http://arxiv.org/abs/1605.01775

[9] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *Trans. Img. Proc.*, vol. 13, no. 4, pp. 600–612, Apr. 2004. [Online]. Available: http://dx.doi.org/10.1109/TIP.2003.819861

[10] X. Yuan, P. He, Q. Zhu, R. R. Bhat, and X. Li, “Adversarial examples: Attacks and defenses for deep learning,” *CoRR*, vol. abs/1712.07107, 2017. [Online]. Available: http://arxiv.org/abs/1712.07107

[11] Z. Zhao, D. Dua, and S. Singh, “Generating natural adversarial examples,” *CoRR*, vol. abs/1710.11342, 2017. [Online]. Available: http://arxiv.org/abs/1710.11342

[12] J. Su, D. V. Vargas, and K. Sakurai, “One pixel attack for fooling deep neural networks,” *CoRR*, vol. abs/1710.08864, 2017. [Online]. Available: http://arxiv.org/abs/1710.08864

[13] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against deep learning systems using adversarial examples,” *CoRR*, vol. abs/1602.02697, 2016. [Online]. Available: http://arxiv.org/abs/1602.02697

[14] N. Narodytska and S. Kasiviswanathan, “Simple black-box adversarial attacks on deep neural networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, July 2017, pp. 1310–1318.

[15] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, “Black-box adversarial attacks with limited queries and information,” *CoRR*, vol. abs/1804.08598, 2018. [Online]. Available: http://arxiv.org/abs/1804.08598

[16] P. Vidnerová and R. Neruda, “Evolutionary generation of adversarial examples for deep and shallow machine learning models,” in *Proceedings of the The 3rd Multidisciplinary International Social Networks Conference on SocialInformatics 2016, Data Science 2016*, ser. MISNC, SI, DS 2016. New York, NY, USA: ACM, 2016, pp. 43:1–43:7. [Online]. Available: http://doi.acm.org/10.1145/2955129.2955178

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” *Trans. Evol. Comp*, vol. 6, no. 2, pp. 182–197, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1109/4235.996017

[18] Štěpán Procházka, “Evolutionary generated adversarial examples,” 2018, last visited 2018-07-20. [Online]. Available: https://github.com/proste/evgena

[19] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/