Generating and defending against adversarial examples in vision-optimized neural architectures

Daniel Donoghue
Email: ddonogh1@jhu.edu

Nicholas Lines
Email: nicholasalines@gmail.com

Arnaldo Pereira
Email: aepereira@gmail.com

***Abstract*—As automated decision-making becomes more popular and more dependent upon artificial intelligence, securing sensitive models from adversarial behavior has become essential. Neural networks are particularly vulnerable to so-called adversarial examples [6], and various attacks and defenses have been explored in the literature.**

Our intention in this paper is to demonstrate and confirm the results of such attacks at an informative but modest scale. We apply two common attacks to both the Wide ResNet and GoogLeNet neural models, and test two defenses, in a reproducible computational environment. We show that significant improvements in network robustness are available with minimal defense measures.

The authors are listed alphabetically, and all made equal contributions. This work is performed in association with the Johns Hopkins Engineering for Professionals Program, as a project for EN.625.638.8VL2.FA20 Neural Networks.

All code and further reference materials are available online at https://github.com/linesn/adversarial_examples. This repository serves as an extensive appendix to the report given here.

CONTENTS

I. EXECUTIVE SUMMARY

Since 2014 when Szegedy et al [6] published the first observation on the subject, adversarial examples have gained much attention in both the study of adversarial machine learning research and the more results-oriented world of practical neural architecture, due to the alarming weaknesses they expose and the interesting robustness that can be introduced via defense efforts. The term “adversarial example” is used to describe “an input to a machine learning model that is intentionally designed to cause the model to make a mistake in its predictions, despite resembling a valid input to a human” [8]. As such these examples are classed as evasion techniques by adversarial machine learning theory, since their goal is to evade detection while producing misinterpretations [7].

Most of the literature on the subject (in keeping with traditions in the neural network community) uses image recognition tasks to demonstrate the efficacy of attacks and defenses, and we will do the same. In this paper we will demonstrate successful use of the Fast Gradient Sign Method (FGSM) [3] and Directed Gradient Sign (DGSM) [4] attacks against convolutional neural networks trained with Imagenette data. We will then examine the results of applying two common defenses: first, perturbed prediction averaging, and second, training using adversarial examples. We confirm the observations of Goodfellow et al [3] and show that, while the Wide ResNet and GoogLeNet architectures are very susceptible to the above attacks, the named defenses also produce significant improvement to the robustness of the classifiers.

II. Project overview

A. Why are adversarial examples effective?

In practice neural architectures based on linear components are preferred (over, for example, radial basis components) because of their speed in training and inference. However, it is this property that makes them particularly vulnerable to the most common form of adversarial example [3]. Neural classifier inputs or features naturally have some precision limit, such as a color range or pixel count, below which perturbations are ignored. Consider an input vector $\mathbf{X}$, to which we add a noise vector $\boldsymbol{\eta}$, where every $\eta \in \boldsymbol{\eta}$ is smaller than ϵ, the precision limit. To humans and a first pass review by machines, $\mathbf{X}$ and $\mathbf{X}+\boldsymbol{\eta}$ are identical. However, when the linear activity function is computed, the network's weights are dotted with the input, yielding approximately

$$\mathbf{W}^{\intercal}(\mathbf{X}+\boldsymbol{\eta}) = \mathbf{W}^{\intercal}\mathbf{X} + \mathbf{W}^{\intercal}\boldsymbol{\eta},$$

where the noise term $\mathbf{W}^{\intercal}\boldsymbol{\eta}$ can grow very large if $\mathbf{W}$ is ill-conditioned. This means, in practice, that networks reliant on linear activity functions can produce extremely different outputs when given only minimally altered inputs, as shown in Figure 1. An adversarial example. The original Imagenette photograph on the left is altered by adding the noise shown in the center (which is scaled up to make it visible). The resulting image on the right appears identical to the original, but is misidentified by the classifier.

B. Attacks

The FGSM attack [3] takes advantage of this weakness in a straightforward manner. The attacker forms the perturbation vector $\boldsymbol{\eta}$ to match the cost function gradient sign for a given input, computing

$$\boldsymbol{\eta} = \epsilon_{*}\text{sign}(\nabla_{\mathbf{X}} J(\boldsymbol{\theta}, \mathbf{X}, y))$$

where ϵ_{*} is the allowable level of perturbation, J is the network cost function, $\boldsymbol{\theta}$ is the vector of model parameters, and y is the true label. Thus, if the attacker is in possession of the model and labeled training data, it is easy to train the network to behave badly using simple backpropagation. The result is that the network will lose certainty in the true label classification, and often misclassify the data at random. The attack parameter ϵ_{*} may be scaled, of course, but making ϵ_{*} much larger than the network precision level ϵ may produce examples whose alteration is visible to human reviewers, so smaller ϵ_{*} are desirable from the adversarial perspective.

One can alter this attack to cause the network to favor a particular class instead [4]. We will call this the Directed Gradient Sign Method (DGSM). This time we use the loss function to direct the network toward a specific desired label. We iterate using gradient descent for a given number of iterations, and project the gradient onto the l_{∞}-norm ϵ_{*}-sphere, which has the effect of insisting that a feature is not altered by more than ϵ_{*}. For a given input $\mathbf{X}$, the attacker must solve the minimization problem

$$\begin{aligned} \min_{\delta}\{J_{adv}(\mathbf{X}+\boldsymbol{\delta}) &= J(\boldsymbol{\theta}, \mathbf{X}+\boldsymbol{\delta}, y_{desired}) \\ &\quad - J(\boldsymbol{\theta}, \mathbf{X}+\boldsymbol{\delta}, y_{true})\} \\ \text{subject to } ||\boldsymbol{\delta}||_{\infty} &\leq \epsilon_{*} \end{aligned}$$

where J is again the loss function, $\boldsymbol{\theta}$ the fixed network parameters, $y_{desired}$ and y_{true} are two different labels, and $\boldsymbol{\delta}$ is the directed perturbation vector. This requires using forward passes and backpropagation within the network over N iterations, applying the update rule

$$\begin{aligned} \boldsymbol{\delta}_t &= \boldsymbol{\delta}_{t-1} - a \text{ sign}(\nabla L_{adv}(\mathbf{X} + \boldsymbol{\delta}_{t-1})), \\ \boldsymbol{\delta}_t &\leftarrow \text{clip}(\boldsymbol{\delta}_t, -\epsilon_*, \epsilon_*) \end{aligned}$$

beginning with the zero vector $\boldsymbol{\delta}_0 = \mathbf{0}$. The result of this attack is that the classifier will incorrectly favor the chosen label $y_{desired}$ in adversarial inputs, despite remaining perfectly capable of correctly classifying unaltered inputs.

C. Defenses

A theme that has emerged in the literature is that there is a strong correlation between generally robust networks/inferencing and networks/inferencing methods that are not easily swayed by adversarial attacks. Of course, one also expects a cost in effort or accuracy to be associated with increased robustness.

Our first defense we test is simply Perturbed Prediction Averaging. The simple aim of this method is to “wash out” any adversarial perturbations by averaging over many noisy predictions. This defense has the advantage that it does not require retraining the network, and the only increased expense is the cost of slower decisions at inference time. We predict the class for each image based on an ensemble prediction for the original image and $N-1$ additional perturbed versions of the image, with the perturbations drawn uniformly from the $\epsilon_{\max}$-ball in the l_∞ sense around the image, choosing $\epsilon_{\max}$ to be larger than any expected adversarial alteration level ϵ_*. For example, $\epsilon_{\max}$ can be set large enough that a uniform random $||\boldsymbol{\delta}||_\infty \leq \epsilon_{\max}$ perturbation would be easily noticed by a human (as shown in Figure 2). In that case, we can assume that adversarial attacks will rely on $\epsilon_* << \epsilon_{\max}$. Using a modified softmax function, we can express the probability for class k of classes $\{1, 2, ..., K\}$ predicted using this defense as

$$P(y_k|\mathbf{X}) = \frac{\sum_{i=1}^{N} e^{z_k(\mathbf{X}+\boldsymbol{\delta}_i)}}{\sum_{j=1}^{K}\sum_{i=1}^{N} e^{z_j(\mathbf{X}+\boldsymbol{\delta}_i)}},$$

where $z_j(\mathbf{X} + \boldsymbol{\delta}_i)$ is the output of the jth hidden node for a given network input $\mathbf{X}$ and with $\boldsymbol{\delta}_i = \mathbf{0}$, and $||\boldsymbol{\delta}||_\infty \leq \epsilon_{\max}$ for all i.

In practice we found that a choice of $\epsilon_{\max} = 0.3$(pixel range) gave us perturbation that is easily noticeable. Of course, setting $\epsilon_{\max}$ too high can lead to inaccurate classification results, since the network was not trained on such noisy data; we must balance the extent of our defense with our practical needs.

On the other hand, there are many defensive measures that can be applied directly to the neural network during training to allow faster inferencing that is still adversarially robust. One method we explored is Adversarial Training, where we generate adversarial examples that are added to the training data for the network, either during the original training or during a retraining step. Using FGSM examples is computationally efficient and provides significant security improvements.

III. Computational results

A. Resources

Our computations were made using Jupyter Notebooks in Google Colaboratory with their free GPU and TPU process time. All code and data interfaces are available at the GitHub address given above, in a manner optimized for reproducibility.

B. Imagenette data

To keep our experiments within the scope of our resources, we used a subset of the ImageNet dataset [1] called Imagenette[1] [2] which includes only 10 out of the original 20k classes. These classes[2] are selected to be as distinct as possible, with the intention of allowing classifiers to reach high degrees of accuracy without extensive training for more agile experimentation.

C. Neural architectures

After testing our attack strategy with smaller convolutional networks, we performed the work for this paper by attacking and defending two standard

[1]In keeping with the wishes of the dataset curators, we ask that you internally read “Imagenette” with “a corny inauthentic French accent” unless you are in fact a native French speaker, in which case you are asked to render it in a similarly ridiculous American accent.

[2]The classes are as follows: {tench (a fish), English terrier, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, parachute}.

models used for image classification tasks, Wide ResNet and GoogLeNet. We took advantage of pre-trained PyTorch implementations of these models and simply restricted the output to the ten classes of interest. While the details of these architectures are outside the scope of this review, they can be found in [9] and [5].

We retrained these networks with the output node layer corrected, using 5 epochs of stochastic gradient descent with a learning rate of 0.01 and momentum value of 0.9. This base version of Wide ResNet and GoogLeNet both achieved 99.9% accuracy on a test set of images. Because the results we will show in this report are so similar for both networks, we will only show figures related to the Wide ResNet attacks and defenses, but similar figures for the GoogLeNet classifier and other relevant details are available in project GitHub referenced above.

D. Creating Adversarial Examples

We applied the FGSM and DGSM attacks to both the Wide ResNet and GoogLeNet classifiers with significant success. FGSM decreased accuracy in Wide ResNet classification by more than 50% with perturbations as small as $\epsilon_* = 0.025\epsilon$, or 2.5% of the color range, and dropped as low as 23% accuracy without creating perturbations obvious to the human eye. GoogLeNet suffered worse, dropping about 73% accuracy with an attack on the order of $\epsilon_* = 0.025\epsilon$, and reaching only 20% accuracy with $\epsilon_* = 0.3\epsilon$. These results for Wide ResNet are shown in Figures 3 and 4.

We then used the DGSM attack to direct the classifiers to misclassify in favor of the English Terrier class. For both architectures we found that we could force every image in our test set to resolve to English Terrier with values of $\epsilon_* 0.14\epsilon$ or lower, as shown for the Wide ResNet classifier in Figure 5. Interestingly, a real-world attack of this sort would need to be cautious to weaken the attack sufficiently that the confidence levels in classification are not so consistently high that they give away the attack!

E. Defenses

For both networks we applied the Perturbed Prediction Averaging defense and then attacked the network to see how much more robust the averaging rendered the classifier. We found that this defense eliminated the sudden success of very small ϵ_*-sized FGSM attacks, but still allowed the misclassification rate to grow roughly linearly with the allowed size of ϵ_* (see Figure 6).

This defense proved particularly effective against the DGSM attack, allowing no more than 2% misclassification due to the attack on either classifier.

We then repeated these tests using 10 epochs of adversarial (re)training as our defense instead. This process reduced the overall accuracy of the Wide ResNet classifier to 91% and that of GoogLeNet to 70%. This defense improved both classifier's performance under the FGSM attack, but was more significant for Wide ResNet (see Figure). While Wide ResNet did not dip below 30% accuracy until $\epsilon_* > 0.1\epsilon$, GoogLeNet fellow below that standard for $\epsilon_* = 0.05\epsilon$. Remarkably, the DGSM attack had absolutely no success on Wide ResNet when this defense was applied, though GoogLeNet still lost as much as 4% accuracy due to the attack.

IV. Analysis

To summarize our observations, the FGSM and DGSM generated adversarial examples that successively fooled both Wide ResNet and GoogLeNet classifiers during Imagenette classification tasks. Perturbed Prediction Averaging performed well as an inference correction defense that would force the adversary to use more visible perturbations to prompt similar misclassifications with FGSM examples, and drastically limited DGSM success. Adversarial training also reduced the effect of these attacks, but comparatively was more effective against the DGSM examples. Observing the perturbation pattern produced by a DGSM attack (as shown in Figure 8), we noticed that it appears that the perturbation preserves features of the image that are likely helpful for its correct classification, at least for low enough values of ϵ_*. We speculate that this may occur because adversarial training leads to a parameterization of the network in a region of the loss landscape that is “flatter”—that is, a set of weights may be reached by which the features extracted by deeper layers of the network do not change very much for small changes in the input image. This idea might be worth exploring further.

There are many obvious next steps in this research that may be worth pursuing. We did not attempt to apply both defenses at once, and it would be interesting to see if their effects are complimentary, or if there is little benefit. We also were very conservative in our adversarial training, and could doubtless have improved our model accuracy by running more than 10 epochs. During our discussion about $\epsilon_{\max}$ we were intentionally vague about our selection method. We chose $\epsilon_{\max}$ to be as small as we could assume would still allow the noise to be visible to a human, but as we observed this parameter ought to be chosen in such a way as to optimize the performance of the classifier with and without adversarial attacks.

V. Conclusions

We have seen that adversarial examples are indeed intriguing and informative [6]. Image labeling is one of few tasks neural methods seem to have essentially solved, but those fundamental linear weaknesses highlighted by this report emphasize that the case is not closed until defensive or corrective measures can assure classifier integrity. On the other hand, the defenses considered are just one more addition to the growing arsenal of network-hardening techniques readily available within the machine learning community.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *ImageNet: A Large-Scale Hierarchical Image Database*, CVPR09, 2009.

[2] fast.ai, *Imagenette*, https://github.com/fastai/imagenette, 2020.

[3] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, *Explaining and harnessing adversarial examples*, arXiv preprint arXiv:1412.6572 (2014).

[4] Z Madry, *Adversarial examples, solving the inner maximization*, Adversarial Robustness - Theory and Practice, 2020, [Online; accessed 11-November-2020].

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, *Going deeper with convolutions*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[6] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, *Intriguing properties of neural networks*, 2014.

[7] Wikipedia contributors, *Adversarial machine learning — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Adversarial_machine_learning, 2020, [Online; accessed 11-November-2020].

[8] Rey Reza Wiyatno, Anqi Xu, Ousmane Dia, and Archy de Berker, *Adversarial examples in modern machine learning: A review*, arXiv preprint arXiv:1911.05268 (2019).

[9] Sergey Zagoruyko and Nikos Komodakis, *Wide residual networks*, arXiv preprint arXiv:1605.07146 (2016).

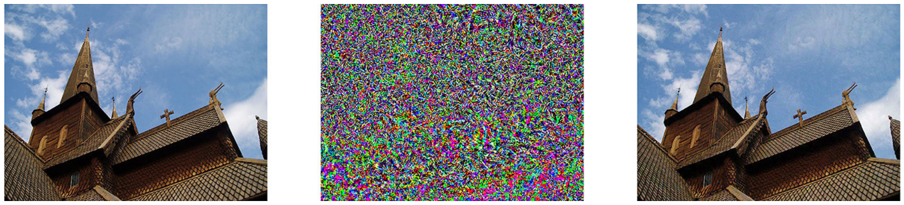

Fig. 1. An adversarial example. The original Imagenette photograph on the left is altered by adding the noise shown in the center (which is scaled up to make it visible). The resulting image on the right appears identical to the original, but is misidentified by the classifier.

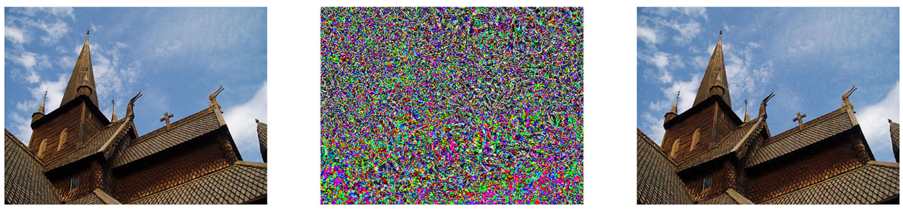

Fig. 2. With Perterbed Prediction Averaging, the original image on the left was used by an adversary to create an adversarial example using $\epsilon_* = 0.02\epsilon$ level noise shown in the center. The defense averages classifications of many perturbations at $\epsilon_{\max} = 0.03\epsilon$ of the example and successfuly classifies the image despite the attack.

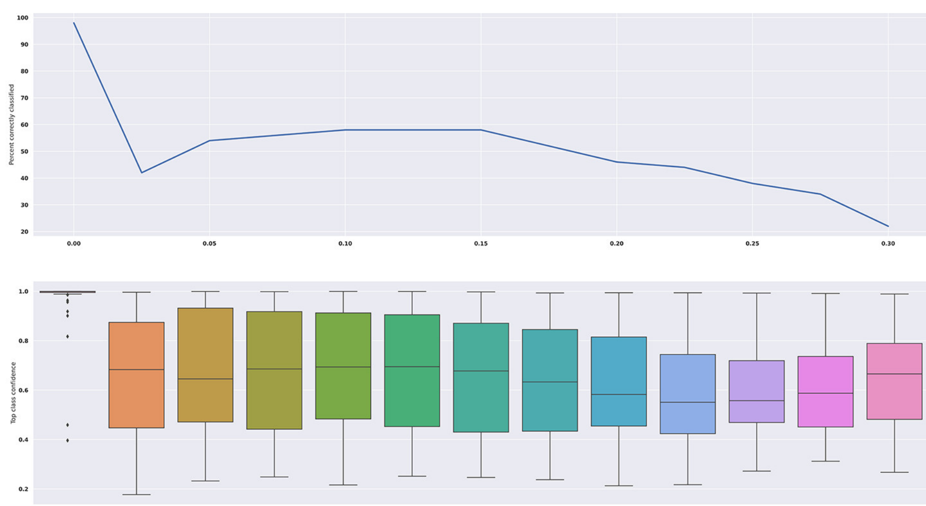

Fig. 3. Results of an FGSM attack on the Wide ResNet classifier at various small values of ϵ_*.

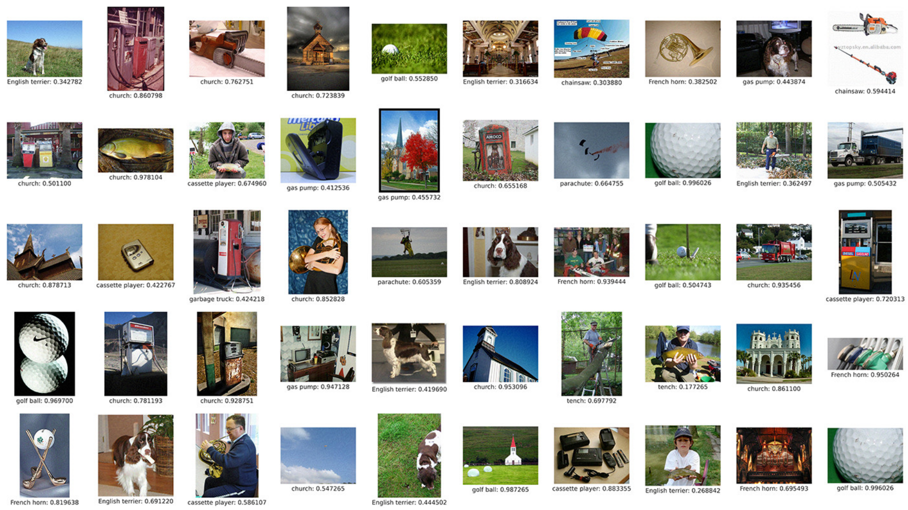

Fig. 4. Example misclassifications made by Wide ResNet as a result of an FGSM attack.

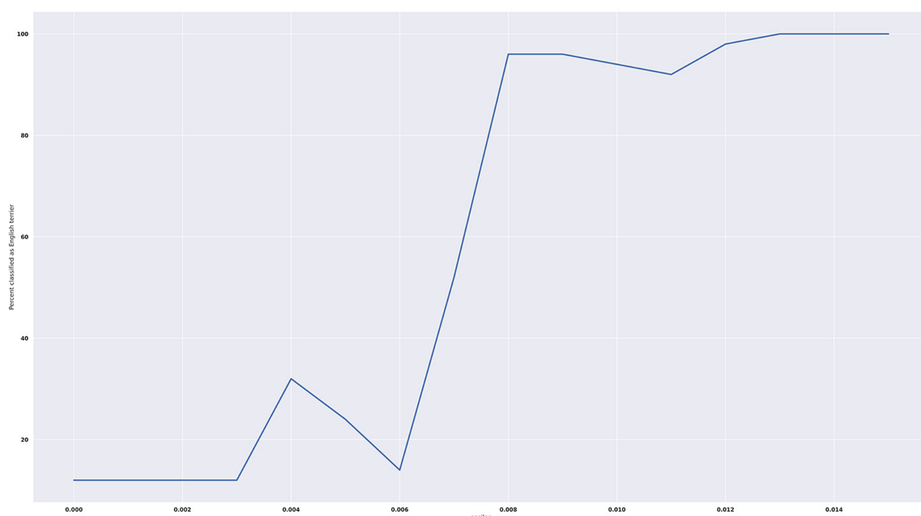

Fig. 5. Results of a DGSM attack on the Wide ResNet classifier at various small values of ϵ_*.

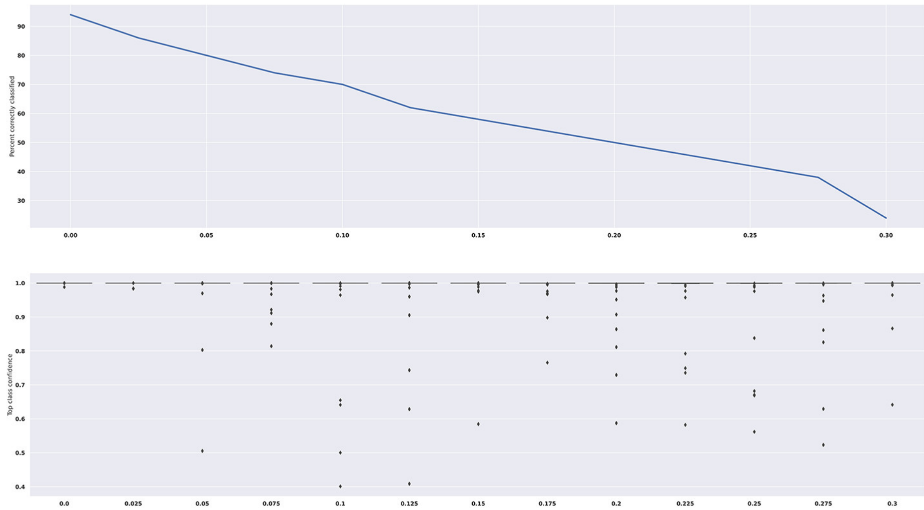

Fig. 6. The classification performance loss achieved by FGSM attacks on Wide ResNet secured with Perturbed Prediction Averaging.

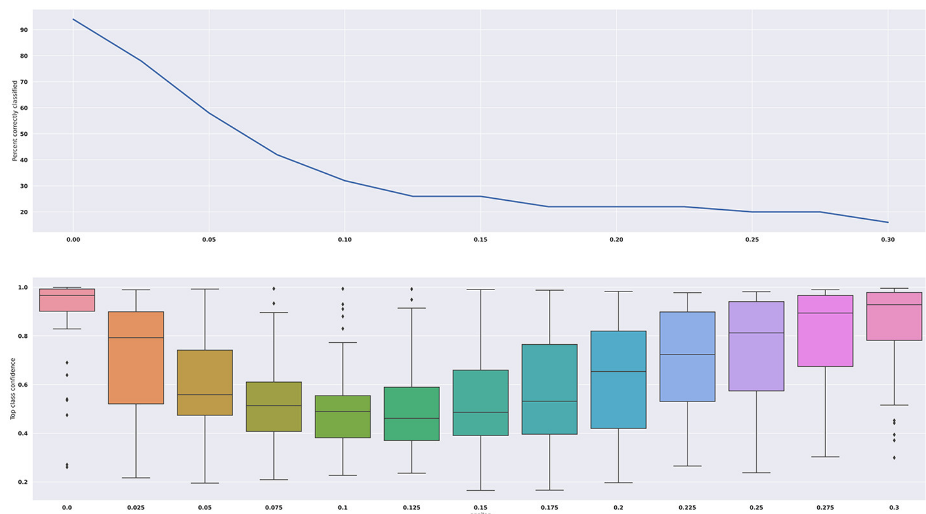

Fig. 7. The classification performance loss achieved by FGSM attacks on Wide ResNet secured by adversarial training.

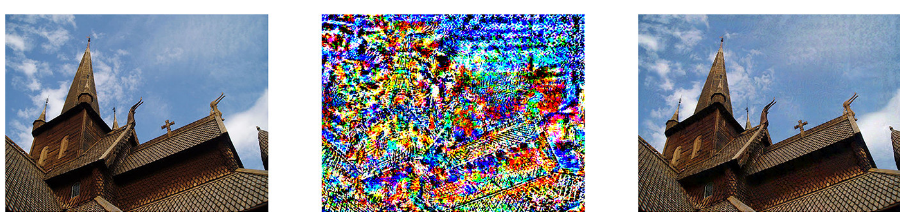

Fig. 8. An example DGSM perturbation pattern. The $\epsilon_* = 0.2\epsilon$ attack captures important image features.