Data independent universal adversarial perturbations

Saurabh Vaidya
Khoury College of Computer Science
Northeastern University
vaidya.saur@northeastern.edu

Sumeet Gajjar
Khoury College of Computer Science
Northeastern University
gajjar.s@northeastern.edu

Virender Singh
Khoury College of Computer Science
Northeastern University
singh.vir@northeastern.edu

1 Original Paper

The original paper we are implementing is Fast Feature Fool: A data independent approach to universal adversarial perturbations.

2 Context

2.1 Adversarial perturbations

Adversarial perturbations are human imperceptible modifications added to the data such that a trained classifier misclassifies the given data. These adversarial perturbations are carefully computed by exploiting some hidden flaws in our neural network models. Almost all known neural network architectures are known to be prone to adversarial attacks.

Adversarial attacks can either be black box or white box attacks. Black box attacks are those where an adversary does not have access to model weights but only to the output of the model. The adversary then leverages only the outputs of the classifier for their given inputs to compute adversarial perturbations. White box attacks are those where the adversary has access to the model weights and computes perturbations leveraging the model weights.

Adversarial perturbations can be specific for each data point or can be universal. Universal perturbations mean that there is single perturbation computed for the entire dataset such that adding this perturbations to every input in the data would achieve some fooling rate on the classifier. The fooling rate is simply the fraction of images where the classifier misclassifies.

However, universal perturbations can also be computed without looking at dataset at all. We would implement one such algorithm where we compute a universal adversarial perturbation for set of models(white box) without using any information from the dataset. Thus the only time dataset would be needed would be to compute the fooling rate in order to verify the efficiency of our perturbation.

2.2 Fast Feature Fool

Since we would not use any information from the dataset, the authors target weights of the model for optimization problem. CNNs decide the output for an image by a combination of layer outputs where each layer output is combination of input and weights. So in order to fool the classifier this final output of each connected layer must be altered.

Fast feature fool does this with the philosophy that if we can maximize the output of each layer, the cumulative effect of these layer outputs would change the output of the classifier. Thus the loss

function to optimize now becomes $\mathcal{L} = -\log\left(\prod_{i=1}^{K} \hat{l}_i(\delta)\right)$ where i is the layer and $\hat{l}_i$ is the mean activation of layer i and δ is the perturbation and $||\delta||_\infty < \xi$

Thus minimizing this quantity to update perturbation would lead to the optimal perturbation. The intensities of perturbations are constrained by projecting them onto a l_∞ ball.

3 Experiments

3.1 Computing the perturbations

Using the above objective function, the perturbations were computed for AlexNet, VGG16 and ResNet architectures. In the original paper, the authors used pre-trained models trained on ImageNet, however, due to limited compute resources available on COLAB, we decided to use pre-trained models trained on CIFAR-10. The fooling rates were computed on the validations set of CIFAR-10. Figure 1 shows the perturbations generated for each of the above networks.

3.2 Transferability across different architectures

The purpose of this experiment is to study the transferability of the perturbations generated for one network and used to fool another network. For this evaluation, we have considered three network architectures ResNet, VGG and AlexNet trained on CIFAR-10 dataset. The fooling rate obtained from this experiment are shown in Table 1. We observe that the perturbations trained in one network show comparable fooling rates in other networks. This implies that this data independent universal adversarial perturbation is transferable across different architectures.

3.3 Initialization with different network's perturbation

The initial value of the perturbation is sampled from a uniform random distribution and it is updated based on the optimization method. It is important to understand the affect of initial values on calculating the perturbations. For the purpose of this experiment, the perturbation generated on a smaller network i.e. AlexNet, is used as initial value for VGG16 and ResNet. Table 2 shows the result of this experiment. It can be observed that this results into an improved fooling rate. This improvement as indicated by the paper is understandable as the perturbation from earlier network already has some structure and shows transferability on the deeper networks, therefore optimization using this offers slight improvement when compared to random initialization.

4 Implementation

Keras package with Tensorflow backend was used to implement the perturbation algorithm. The fooling rate for the networks was computed using the validation set of CIFAR-10. Adam optimizer was used to minimize the objective function with a learning rate of 0.1. $\xi = 15$ was used for generating the perturbations. The authors used $\xi = 10$, this value would be sufficient for ImageNet dataset, however, it did not yield a comparable fooling rate for CIFAR-10. Bumping the value to 15 improved the fooling rate. Since, the optimization updates just the input δ which is restricted to $[-15, 15]$ range, it gets saturated very quickly. To avoid later updates from being ignored we periodically rescale the perturbation to $[-7.5, 7.5]$ range and then continue the optimization.

5 Results

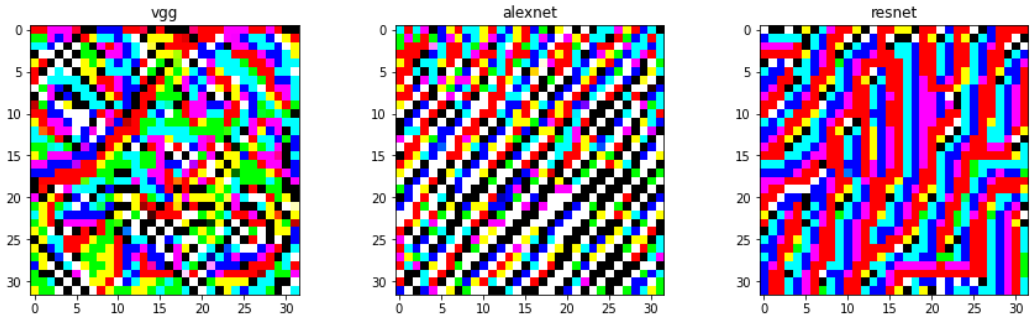

Figure 1: Data independent universal adversarial perturbations crafted by the proposed method for multiple architectures trained on CIFAR-10 dataset. Perturbations were crafted with $\xi = 15$. Corresponding target network architecture is mentioned below each image.

Network	**VGG**	**AlexNet**	**ResNet**
VGG	50.13%	45.06%	43.66%
AlexNet	61.13%	67.52%	63.24%
ResNet	44.57%	41.21%	55.64%

Table 1 : Fooling rates for the proposed perturbations crafted for multiple networks trained on CIFAR dataset. Each row shows fooling rates for perturbation crafted for a particular network. Each column shows the transfer fooling rates obtained for a given network.

Network	VGG	Resnet
Fooling rate	59% (Improved by 8.9)	77.6% (Improved by 22)

Table 2 : Fooling rates for deeper networks initialized with alex net's perturbation.

References

[1] Mopuri, Garg Babu Fast Feature Fool: A data independent approach to universal adversarial perturbations