BatchOut: Batch-level feature augmentation to improve robustness to adversarial examples

Akshayvarun Subramanya
Video Analytics Lab
Indian Institute of Science
Bangalore, India

Konda Reddy Mopuri
Video Analytics Lab
Indian Institute of Science
Bangalore, India

R.Venkatesh Babu
Video Analytics Lab
Indian Institute of Science
Bangalore, India

ABSTRACT

Machine Learning models are known to be susceptible to small but structured changes to their inputs that can result in wrong inferences. It has been shown that such samples, called adversarial samples, can be created rather easily for standard neural network architectures. These adversarial samples pose a serious threat for deploying state-of-the-art deep neural network models in the real world. We propose a feature augmentation technique called *BatchOut* to learn robust models towards such examples. The proposed approach is a generic feature augmentation technique that is not specific to any adversary and handles multiple attacks. We evaluate our algorithm on benchmark datasets and architectures to show that models trained using our method are less susceptible to adversaries created using multiple methods.

KEYWORDS

Machine Learning, Convolutional Neural Networks, Adversarial examples, Feature Augmentation

ACM Reference Format:
Akshayvarun Subramanya, Konda Reddy Mopuri, and R.Venkatesh Babu. 2018, BatchOut: Batch-level feature augmentation to improve robustness to adversarial examples. In *ICVGIP*.
, 8 pages.

1 INTRODUCTION

Machine learning models have been shown to be vulnerable to adversarial samples: inputs corrupted with small perturbation specifically optimized to mislead them [4–6, 11, 15]. Szegedy *et al.* [36] showed that deep neural networks, despite their impressive performance, are vulnerable to adversarial samples. Later, many Convolutional Neural Networks (CNN) based classification models are shown [11, 18, 23–27] to be fooled by adversarial images. Firstly, let us define an adversarial image. Let I be an image that is fed to a classifier whose mapping from images to labels is denoted as $f(I)$.

An adversarial image $\hat{I}$ can be defined as

$$f(\hat{I}) = f(I + r) \neq f(I) \qquad (1)$$
$$s.t \ \|\hat{I} - I\|_p \leq \delta$$

Here, r is the perturbation that is added to the image such that the output of the classifier changes. There is an added restriction that $\hat{I}$ and I do not differ by more than δ under ℓ_p norm. Usually, ℓ_∞ is considered so that maximum pixel-wise difference is restricted.

Because of the ℓ_p constraint, adversarial images are often indistinguishable from their corresponding original images for humans. Interestingly, the adversarial images generalize across models trained with different architectures or different subsets of data [36], reducing the possibility of using an ensemble of classifiers to counter them. Kurakin *et al.* [17] recently showed that these examples can occur in the physical world by printing the adversarial images and showed that classifiers are fooled by these samples as well. Black-box attacks, which involve no knowledge of network parameters, have also been effective in fooling the classifiers [32].

Susceptibility to these rogue samples can pose a severe problem for deploying the deep learned models in critical applications. Multiple attempts have been made to understand and reason the existence of adversarial samples, viz. exploiting linearity of the models [11], sparse training data [30], proximity of classification boundary to the submanifold of sampled data [37] etc. Considering the severity of the problem, there have been multiple attempts to reduce the effect and train networks robust to such adversaries. One of the most popular and effective methods is to train the models with the adversarial examples in addition to the normal samples. This method, called *Adversarial training* [11], uses Fast Gradient Sign Method (FGSM) to generate adversarial examples at each training iteration and presents a combination of normal and adversarial examples to train the network.

We propose a novel feature space augmentation technique that makes the deep learned models robust to adversarial attacks. We name our method *BatchOut*, as it operates on a mini*batch* of data and similar to dropout, it *randomly* selects the samples which are to be used for augmentation. Our method is a generic training regime that augments the feature space to achieve smoother predictions. The advantages of our approach over adversarial training are (i) it requires no knowledge about the nature of the target adversarial attack and (ii) it does not require computing gradient at each iteration during training.

2 RELATED WORK

This section is aimed at highlighting some of the previous works based on model optimization that attempted to defend neural network architectures against adversarial attacks. The most trivial and effective method is to use the adversarial examples to during training. This method called Adversarial Training was proposed by Goodfellow *et al.*[11]. They used the Fast Gradient Sign Method (FGSM) to generate adversarial examples at each iteration of training and trained the network using them. [21] proposed the PGD adversarial algorithm which is one of the strongest iterative methods for fooling. They also showed that training with examples created using their method can increase robustness significantly.

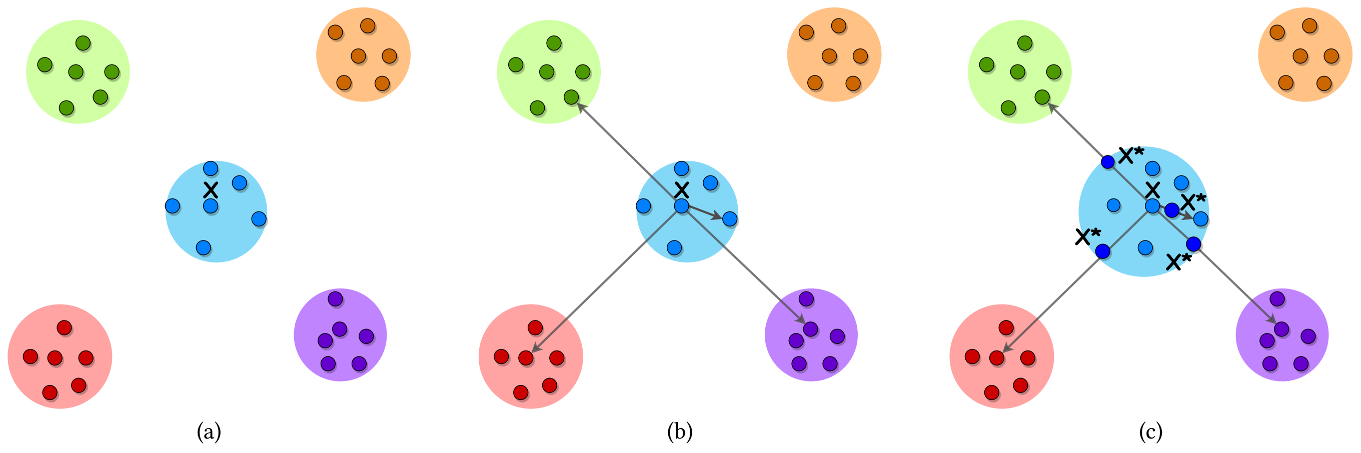


Figure 1: A toy example illustrating the proposed feature augmentation technique. (a) Feature space with five categories before augmentation. (b) Adversarial directions for a chosen sample x. Note that a sample can choose another sample belonging to same class. (c) Feature augmentations x^* are shown for x. Note that the augmented samples x^* (darker circles) can lie anywhere along the pointed adversarial directions depending upon the value of η.

However, it is to be noted that such methods are computationally expensive and difficult to scale. Although this method is the most often used defense against adversaries, it is easy to see that it is specific to the method of adversarial generation and it is difficult to scale as well. Recently, Kurakin *et al.*[18] were able to perform adversarial training for large networks such as Inception-v3 using synchronous training across 50 machines. Although many defenses have been proposed (e.g. [7, 13, 35]), they have been shown to be suffering from *obfuscated gradients* [2], and could be circumvented by making slight changes to the adversary. Our method does not suffer from this, since at test time, we present only the model without any additional architectural changes. Another work presented by Papernot *et al.*, called Defensive Distillation [31], similar to [14], uses two networks with the same number of parameters to perform training. However, it was shown that such networks too can be attacked by making small changes to the method of attack [8]. [34] proposed layerwise training with FGSM adversary as a useful regularization to defend the model. They use gradients from previous minibatch to perturb current minibatch samples to create adversary. Our approach is different from theirs because we obtain these directions without computing any gradients.

One of the earliest attempts at increasing model robustness was by Gu and Rizagio [12]. They came up with Deep Contracive Network (DCN) , which had an explicit regularizer of the form $\left\|\frac{\partial X^i}{\partial X_{i-1}}\right\|_2$ which penalizes large changes in outputs of consecutive layers, thereby ensuring that the Jacobian of the output of the network w.r.t the input $\left(\frac{\partial X^N}{\partial X^0}\right)$ is minimized when training the network. However, this regularizer was shown to restrict the capacity of the network, showing reduced performance when compared with vanilla training. Recently, there have been multiple attempts to detect adversarial examples at test time. Hendrik Metzen *et al.* [22] were able to detect adversarial examples with considerable success using an auxiliary binary classifier which was trained to detect adversarial examples from different methods[22]. Li *et al.*[20] were able to detect L-BGFS adversarial examples by using filter statistics from different layers of the CNN. Our method attempts to create robust models without any prior knowledge on the adversary.

3 PROPOSED APPROACH

In this section, we present the proposed approach to train robust networks against adversaries. First, we define the notation that will be used throughout the paper.

Consider a Convolutional Neural Network (CNN) consisting of N layers where the N^{th} layer's output is the softmax distribution over classes. Let $\mathbb{D}$ represent the training data with images and corresponding labels. $X^i \in \mathbb{R}^d$ denotes the set of activations for mini-batch of size m at the i-th layer in the network where $i \in [1, 2, \ldots, N]$. X^0 represents the images from the data distribution $\mathbb{D}$. $x^i_j = X^i[j]$ where $j \in [1, m]$ denotes the j^{th} sample's activation in the minibatch at the i^{th} layer of the network.

Our method is based on the fact that the landscape of learnt representations is not smooth. Deep Neural Networks learn by projecting the data onto high dimensional spaces using non-linear operations. However, the dimensions of these spaces are typically of the order of thousands, which means that the data used to train the network usually occupies a very small subspace. But, owing to efficient optimization procedures, networks end up learning the mapping onto these low-dimensional subspaces, where the generalization to validation and test sets is preserved. But these procedures do not result in smooth manifolds, which is exploited by adversaries.

There have been multiple attempts to demonstrate this uneven nature of the representations of the adversarial and their corresponding clean images. Recently, Sabour and Cao *et al.*[33] showed that representations learnt by deep networks can be manipulated adversarially to fool the classifier. They demonstrated that it is plausible for images to be close in ℓ_∞ sense, but their corresponding representations are far according to ℓ_2 distance. This, along

Algorithm 1 BatchOut

```
Input: Activations of a mini-batch at the $i$-th layer $X^i$, Magnitude of augmentation $\eta$ and number of samples to be augmented $k$
    $R \leftarrow RandInt(m)$                          // Sample $m$ integers from [1, m] with replacement
    $X^i_r \leftarrow X^i[R]$                          //Choose a random sample from minibatch
    $d \leftarrow X^i_r - X^i$                         // Compute the direction of perturbation
    $X^i[1:k] \leftarrow X^i[1:k] + \eta \cdot d[1:k]$   // Augment and replace the first $k$ samples with the corresponding augmented samples
Output: $X^i$
```

with the definition of adversarial images show that the distances in image space and feature space should be interpreted differently. The inter-image and inter-feature distances, when analyzed can be counter intuitive by nature. We can understand these adversarial samples as adding a small perturbation in the image space to the original sample such that it results in large changes in their representations. Owing to this, multiple methods exploit the gradients to change the output of the classifier with minimal perturbation in the image space. Different methods such as Fast Gradient Sign Method [11] and DeepFool [23] construct these adversaries in one-step or iterative fashion by using optimization procedures which result in these changes. These methods try to perturb the sample in the feature space such that the original sample is displaced along the direction of other classes. DeepFool explicitly does this by finding the nearest decision boundary to augment the sample x such that it is perturbed towards the other side of the boundary.

To overcome these adversaries and to provide an attack-agnostic method of robustifying the network, we add small perturbations to the learned representations of images during training itself. This ensures that they compensate for the adversaries that can occur during test time. In [12, 36], the authors have demonstrated that training with Gaussian noise added to the images is not sufficient to compensate these adversaries. There is also the possibility of adding Gaussian noise to the learned representations to realize the same objective. However, we found that this too did not improve the performance on adversaries. This can be attributed to the fact that added noise is not *strong* enough since it has no knowledge of the subspace that is learnt by the network. In Adversarial training, this perturbation is added by the FGSM method using gradients. However, in our method we propose to directly augment the representations such that they are perturbed along the samples of other classes, i.e we add perturbations in the representation space as a function of the representations themselves.

Hence we define our feature augmentation procedure for a sample x_j in the i^{th} layer of the network as follows.

$$x_j^{i^*} = x_j^i + \eta \cdot (x_r^i - x_j^i) \tag{2}$$
$$where\ x_r^i = X_i[\ R\ ]\ and\ R \sim \mathcal{U}(1, m).$$

$\mathcal{U}(\ a, b\ )$ represents a discrete uniform distribution between [a, b].

For a sample x that belongs to a particular class, all other categories become *adversarial* to it. In order to perturb the sample x_j^i, we need to add the perturbation such that it moves in this adversarial direction. The second term in Equation 2 represents this perturbation. The term x_r^i is obtained by sampling from mini-batch X_i in a uniform manner. Intuitively, the term $(x_r^i - x_j^i)$ represents the perturbations added to x_j^i such that it is perturbed towards x_r^i. The parameter η determines how far it is perturbed along that direction.

To illustrate our algorithm, let us consider a toy example where each class corresponds to single Gaussian with corresponding μ and σ. Figure 1(a) shows an example distribution with five classes. Each of the big circles (light colored) denotes subspace occupied by one class and the smaller circles within them represent the representations of the data belonging to that class. The adversarial directions for a data sample x in the projection space are vectors that connect x to arbitrary samples of other categories. Figure 1(c) shows the proposed augmentation for a sample x. Note that there are five semantic classes in the toy example and four augmentations (x^*) shown. Because of the augmentation, we can observe the part of distribution corresponding to x's semantic category (blue region) changes as shown in figure 1(c). Also, it might happen that, as a result of random sampling, some samples might get augmented in the direction of samples belonging to the same category. In that case, it can be interpreted as a simple intra-class augmentation similar to label preserving data augmentation [9]. The proposed method is presented as an algorithm in Algorithm 1.

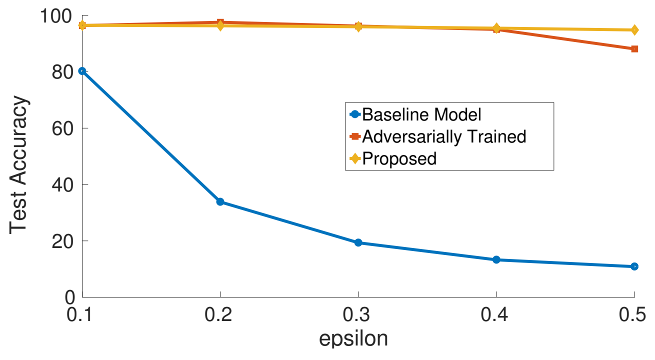
4 DISCUSSION

Here, we try to analyze the effectiveness of our method against adversaries. One question that arises about the proposed method is how to interpret the augmented features? Bengio *et al.*[3, 28] showed that representations are *untangled* better as we go deeper into the network. They showed that convex combination of samples at higher levels of representation leads to plausible looking input images. Rewriting equation (2),

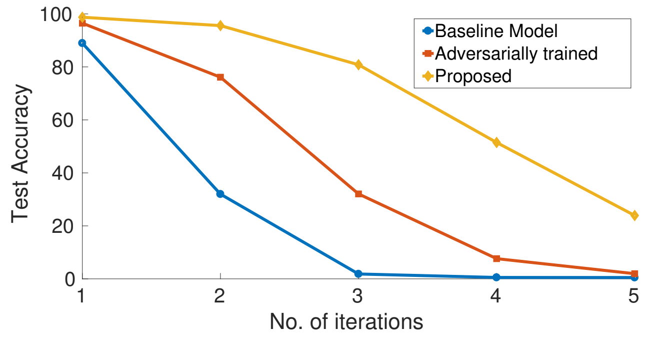
$$x_i^* = x_i + \ \eta \cdot (x_{r_i} - x_i) = (1 - \eta) \cdot x_i + \eta \cdot x_r \tag{3}$$

We can see that our feature augmentation also achieves the same objective. Hence, the augmented features from our method can be understood as features that would have been generated from images that are slight variations of the two corresponding images. In Adversarial Training, we create adversarial images at each iteration using the FGSM method, yielding adversarial images that are variations of the training data. Our method can also be understood as computing adversaries in feature space using the augmentation method described in Equation 2. Hence, we see an improved performance on adversaries generated using both FGSM and DeepFool methods as shown in section 6.

Another way to understand the efficacy of our method is by relating to the concept of Deep Contractive Networks (DCN) presented by Gu and Rizagio *et al.*[12]. They proposed an explicit regularizer that would penalize the Jacobian of consecutive layers, thereby minimizing the Jacobian of the entire network. They also reasoned



FGSM



DeepFool

Figure 2: Test accuracies for the network trained on MNIST against FGSM and DeepFool attacks. We observe that the proposed BatchOut trained network is robust to both adversarial attacks.

that such a regularizer would ensure gradients to the network remain as smooth as possible. The main drawback of their method, as described in section 2, is that this places a hard constraint on the network which leads to reduction in performance on clean data. In our method, we are indirectly making the feature space smooth by creating these augmented samples around the training data. This comes with the additional benefit of not placing any kind of restriction on the network and ensures that sensitivity of network's predictions near the training data is decreased. By performing this augmentation at multiple layers, our method can be interpreted as trying to reduce the irregularity in learned representations at all layers of the network.

5 METHODS OF ATTACK

In this paper, we evaluate the robustness of networks under two different forms of attack. First one, is the Fast Gradient Sign Method (FGSM) proposed by Goodfellow *et al.*[11] and DeepFool proposed by Moosavi-Dezfooli *et al.*[23]. The following subsections present a brief introduction to these methods.

5.1 Fast Gradient Sign Method(FGSM)

Fast Gradient Sign Method, as the name suggests, is one of the fastest and easiest ways of constructing adversaries. This method involves performing gradient ascent on the Loss function such that the loss for the correct class increases, thereby decreasing the model's confidence towards the predicted class. If $J(\theta, I, y)$ represents the loss function used to train the network,

$$\hat{I} = I + \epsilon \cdot \text{sign}(\nabla_I J(\theta, I, y_{true})) \tag{4}$$

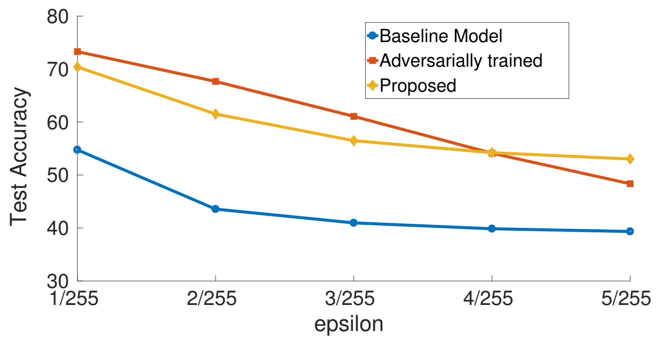
Here, $\nabla_I J(\theta, I, y_{true})$ represents the gradient of Loss function w.r.t the image I. It is easy to see that ϵ, which determines the magnitude of perturbation added, is a result of placing an ℓ_∞ constraint as described in Equation 1.

5.2 DeepFool

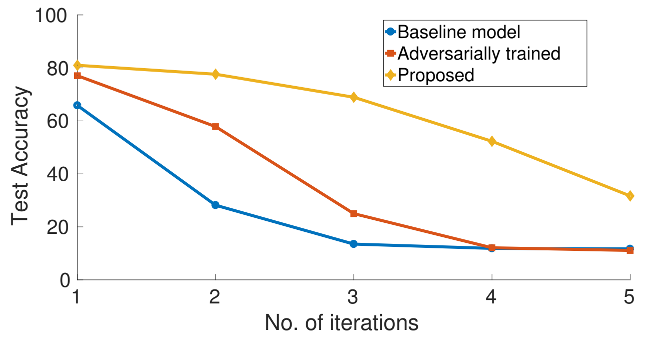
DeepFool is an adversary which iteratively generates adversarial examples for an image I. They use the pre-softmax representations to determine the nearest decision boundary to perturb the original sample. Iteratively, they perturb the input until it crosses that decision boundary, thereby creating an adversarial example. DeepFool can be used with any of ℓ_p constraints, but in our experiments we use ℓ_2 norm to create these samples. DeepFool has been shown to be minimalistic in the strength of perturbations required to create the adversarial image $\hat{I}$.

6 EXPERIMENTS

In this section we demonstrate the effectiveness of the proposed augmentation technique to handle the multiple adversarial attacks via training classifiers on different datasets. In particular, we consider MNIST [19] and CIFAR-10 [16] datasets and evaluate on adversaries described above.



FGSM



DeepFool

Figure 3: Test accuracies for the network trained on CIFAR-10 against FGSM and DeepFool attacks. Note that the baseline network suffers the most and the adversarially trained network suffers more for DeepFool attack. The proposed BatchOut trained network is robust to both the attacks.

6.1 Implementation details

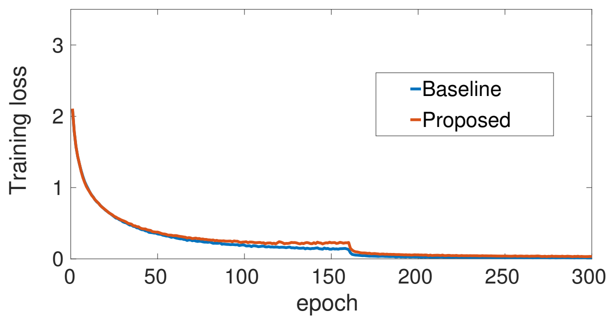
In all our experiments, we implement the algorithm presented in Algorithm 1. We have implemented the proposed augmentation as a layer operation in Tensorflow[1] and Lasagne, a Theano based framework [10, 38] to evaluate our method. To address the question of gradient computation for our layer, we do not pass the gradient through the network for the branch that computes the perturbation[1], i.e., second half in Equation (2) $\left[\eta(x_r^i - x_j^i)\right]$. This is done so that a layer i continuosly generates the augmentation that is fed as input to layer $(i+1)$ and by stopping the gradient, the layer $i+1$ has no knowledge about the branch computing our augmentation. Essentially, we are ensuring that the layer $(i+1)$ get its input x^* such that it is output of layer i directly, not as a result of our augmentation. Stopping the gradient can be seen as considering the augmentation branch as constant during backpropagation, even though it is a function of the previous layer's inputs. We observed that if we do not perform this operation, the network learnt to undo the operation performed by our augmentation, thereby reducing its effectiveness. During inference, we do not perform the proposed augmentation and the original activations are passed without augmentation. All operations described in our approach are done on *pre-relu* activations. In all our experiments we fixed the value of k as half of the minibatch size. That is, during training, at each iteration, 50% of the minibatch samples are augmented. We created the test adversarial samples from FGSM method using the earlier version of cleverhans library [29] which supported Theano. We evaluate our method on different adversaries similar to [22]. The hyperparameters were chosen based on the performance on the held out validation set.

6.2 MNIST

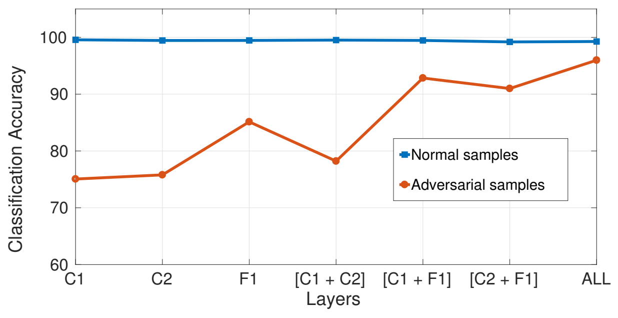
We trained a CNN which consists of 2 convolution layers and a fully connected layer [2] using MNIST data. We first train the CNN without any augmentation and obtain a model referred as **Baseline model**, which achieves a classification accuracy of 99.47% on the test set. We then train a new model with the same architecture but perform adversarial training as discussed in [11], i.e during training, at each iteration, half of the images in the mini-batch are replaced by their corresponding FGSM adversarial images computed over the same network at that instant of training. This is referred as **Adversarially trained model**. This network achieved a classification accuracy of 99.31% on the test set. Lastly, we train a third model with the same architecture but perform the proposed feature augmentation at all the layers. That is, the CNN now has an additional BatchOut layer per each layer. This network achieves a test accuracy of 99.12% and is referred to as **Proposed model**.

[1]This can be done using `tf.stop_gradient` in Tensorflow and `theano.gradient.disconnected_grad` in Theano

[2]Please refer appendix for exact details regarding the architecture



(a)



(b)

Figure 4: (a) Time of convergence baseline model and model with proposed augmentation trained on CIFAR-10. (b) Effect of performing the proposed augmentation at different layers.

We test the robustness of all the above three networks towards adversarial attacks. We compute FGSM and DeepFool adversarial images corresponding to the test set for the three models separately and test their classification performance. Figure 2 shows the achieved classification accuracies. For evaluating using Fast Gradient Sign Method, we create adversaries of varied strength using the parameter ϵ. We evaluated against DeepFool iterative attack for multiple iterations (1 to 5). As expected, the accuracy of baseline network falls drastically with increase in the strength of the adversary. Adversarial trained network shows robustness to FGSM examples and performs better than the baseline network on DeepFool examples. However, the proposed approach is clearly robust to both adversaries, even though it was not trained with examples from either attack, highlighting the attack-agnostic nature of our method. Note that the proposed method achieves robustness to attacks with negligible drop in the accuracy over clean images.

6.3 CIFAR-10

For CIFAR-10 dataset we use a VGG style architecture to classify the images. The network consists of 5 convolution layers and 2 fully connected layers. Similar to the MNIST scenario, we obtain three separate models. The **Baseline model** achieves 80.22%, the **Adversarially Trained model** achieves 80.01% and the **Proposed model** achieves 80.79% on the original test set. The only change we employ here is that the η values are sampled from $\mathcal{U}[0, N]$, where N is chosen for each Batchout layer. This can be seen as varying the magnitude of noise and thereby, exploring more points in the high-dimensional subspace. Figure 3 shows the classification accuracies obtained on adversarial images crafted from the 10K test set images. We evaluated our approach with FGSM and DeepFool adversaries for varied strengths. Note that the normal network is susceptible to both the attacks and fails to recognize the objects correctly. Adversarially trained network is robust to only the FGSM attacks and suffers for DeepFool attacks. On the other hand, as observed with MNIST dataset, our approach makes the model robust to multiple attacks without losing accuracy on normal images.

6.4 Time of Convergence

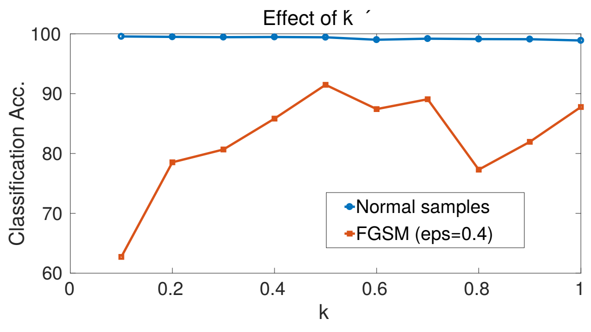
Here we present the time of convergence for baseline training and training with the proposed feature augmentation with the network we train for CIFAR-10 dataset. Figure 4(a) shows decrease in loss as the training progresses. It is clearly observed that our method does not affect the convergence of the model compared to baseline training, showing that there is no change in hyper-parameters such as learning rate, optimizer etc. is required when using our algorithm.

6.5 Augmenting at different layers

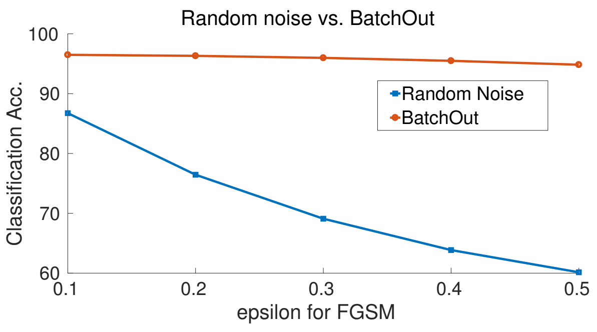
In this section we present the effect of performing the proposed augmentation at different layers in the network trained on MNIST. Figure 4(b) shows the classification accuracy on FGSM examples with $\epsilon = 0.3$ obtained by the network when we perform our augmentation at different individual layers and combinations of them. Ci denote i^{th} convolution layer and $F1$ denotes the fully connected layer. The network consists of two convolution layers and one fully connected layer. The figure shows the accuracies over normal and FGSM adversarial images from the 10K test images of the dataset. It is clearly observed that the performance on adversarial examples increases as we apply our algorithm to combination of layers simultaneously. The best performance was achieved when augmentation was performed at all layers of the network. This can be attributed to the fact that reducing the sensitivity at multiple layers of the network can result in better robustness to adversaries. Note that, when applied to the single fully connected layer, it performed better than the combination of convolution layers. This shows that as the representations are more *untangled*, our augmentation shows better variations in data. This leads to presenting more possibilities in input data, thus increasing the robustness of the network.

6.6 Effect of the hyperparameter k

One of the important hyperparameters of the proposed method is k, which is the fraction of the minibatch to be augmented using the proposed method. Here we seek to investigate the effect of varying this hyperparameter. We observe the accuracy for MNIST data on normal test samples and their FGSM adversarial counterparts for $\epsilon = 0.4$. As observed in Fig. 5 (a), we see that as k increases, the accuracy on normal samples gradually decreases and accuracy on adversarial samples shows improvement. We notice that $k = 0.5$ strikes the right balance between both measures, which is to be



(a)



(b)

Figure 5: (a) Effect of k, the fraction of mini-batch samples augmented via Batchout to make the network robust. We notice that the classification accuracy on MNIST data peaks at $k = 0.5$ and is selected across multiple other experiments. (b) Comparison of Batchout augmentation with that of Gaussian random noise. It can be clearly observed that adding structured perturbations during training is a better alternative to adding random unstructured noise.

expected since $k = 0.5$ corresponds to equal number of clean and augmented samples in the minibatch. Hence we set $k = 0.5$ in all our reported results.

6.7 Random noise baseline

Our method can be understood as adding small and structured perturbation in feature space during training such that the network becomes robust to adversaries. In order to bring out the efficiency of the proposed approach, we conducted an experiment where we add random Gaussian noise as perturbation to the minibatch samples and compare it with the proposed method. To have a fair comparison with our method, we added the similar magnitude of perturbation in both cases.

$$x_j^{i*} = x_j^i + \eta \cdot \left\| \left(x_j^i \right) \right\|_2 \cdot \frac{n}{\|(n)\|_2} \quad (5)$$
$$where\ n \sim \mathcal{N}(0,1).$$

As shown in Fig. 5(b), the proposed method clearly outperforms the random noise augmentation, indicating that the proposed feature augmentation which is a function of the representations, is structured and leads to better robustness to adversaries.

7 CONCLUSION

We have proposed a novel feature augmentation technique that enables training neural network architecture robust to adversaries. Our method works in an attack agnostic manner, which results in improved performance on multiple adversaries, whilst maintaining the performance on clean test examples. Our algorithm can be easily implemented in any deep learning framework as an easy-to-use layer operation. We empirically demonstrate the effectiveness of our method by training neural network architectures on benchmark datasets and showing robustness to fooling methods.

8 APPENDIX

8.1 Network Architectures

Here we provide the architecture details of networks that are presented in Experiments section. Note that Conv(w, w, ch, s) represents Convolution layer with ch filters with $w \times w$ spatial resolution and perform convolution with stride s. BatchOut is the proposed Batch level Augmentation Feature Augmentation Layer. FC(ch) denotes the fully connected layer with ch neurons. k denotes the % of minibatch samples that are augmented. Note that value of η is mentioned at each of the BatchOut layer.

Table 1: The architecture details of the network trained on MNIST datastet.

MNIST	k	η
Conv (5, 5, 32,1)		
BatchOut	0.5	0.05
ReLU		
MaxPool(2,2,2)		
Conv (5, 5, 32,1)		
BatchOut	0.5	0.1
ReLU		
MaxPool(2,2,2)		
Dropout(0.5)		
FC (256)		
BatchOut	0.5	0.1
ReLU		
Dropout(0.5)		
SoftMax(10)		

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).
[2] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420* (2018).

Table 2: The architecture details of the network trained on CIFAR-10 datastet. U[] denotes the continuos uniform distribution.

CIFAR-10	k	η
Conv(3, 3, 32,1)		
BatchOut	0.5	$\mathbf{U}[0, 0.1]$
ReLU		
Conv(3, 3, 64,1)		
BatchOut (0.5, 0.1)	0.5	$\mathbf{U}[0, 0.1]$
ReLU		
MaxPool(2,2,2)		
Conv(3, 3, 64,1)		
BatchOut(0.5,0.1)	0.5	$\mathbf{U}[0, 0.1]$
ReLU		
Conv(3, 3, 64,1)		
BatchOut(0.5,0.1)	0.5	$\mathbf{U}[0, 0.1]$
ReLU		
Conv(3, 3, 64,1)		
BatchOut(0.5,0.1)	0.5	$\mathbf{U}[0, 0.15]$
ReLU		
MaxPool(2, 2, 2)		
Dropout(0.5)		
FC (512)		
BatchOut	0.5	$\mathbf{U}[0, 0.15]$
ReLU		
FC (512)		
BatchOut	0.5	$\mathbf{U}[0, 0.15]$
ReLU		
Dropout(0.5)		
SoftMax(10)		

[3] Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. 2013. Better mixing via deep representations. In *International Conference on Machine Learning*. 552–560.

[4] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 387–402.

[5] Battista Biggio, Giorgio Fumera, and Fabio Roli. 2014. Pattern recognition systems under attack: Design issues and research challenges. *International Journal of Pattern Recognition and Artificial Intelligence* 28, 07 (2014), 1460002.

[6] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389* (2012).

[7] Vivek B.S., Konda Reddy Mopuri, and R. Venkatesh Babu. 2018. Gray-Box Adversarial Training. In *the European Conference on Computer Vision (ECCV)*.

[8] Nicholas Carlini and David Wagner. 2016. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311* (2016).

[9] Terrance DeVries and Graham W Taylor. 2017. Dataset Augmentation in Feature Space. *arXiv preprint arXiv:1702.05538* (2017).

[10] Sander Dieleman, Jan SchlÃijter, Colin Raffel, Eben Olson, SÃÿren Kaae SÃÿnderby, Daniel Nouri, et al. 2015. Lasagne: First release. (Aug. 2015). https://doi.org/10.5281/zenodo.27878

[11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. *CoRR* abs/1412.6572 (2014).

[12] Shixiang Gu and Luca Rigazio. 2014. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068* (2014).

[13] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. 2017. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117* (2017).

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[15] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. 2011. Adversarial Machine Learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence (AISec '11)*.

[16] Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. (2009).

[17] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).

[18] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial Machine Learning at Scale. *CoRR* abs/1611.01236 (2016).

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[20] Xin Li and Fuxin Li. 2016. Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics. *arXiv preprint arXiv:1612.07767* (2016).

[21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[22] Jan H. Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On Detecting Adversarial Perturbations. *ICLR* (2017).

[23] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. *in CVPR* (2016).

[24] Konda Reddy Mopuri, Aditya Ganeshan, and R. Venkatesh Babu. 2018. Generalizable Data-free Objective for Crafting Universal Adversarial Perturbations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2018).

[25] Konda Reddy Mopuri, Utsav Garg, and R Venkatesh Babu. 2017. Fast Feature Fool: A data independent approach to universal adversarial perturbations. In *Proceedings of the British Machine Vision Conference (BMVC)*.

[26] Konda Reddy Mopuri, Utkarsh Ojha, Utsav Garg, and R Venkatesh Babu. 2018. NAG: Network for Adversary generation. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[27] Konda Reddy Mopuri, Phani Krishna Uppala, and R. Venkatesh Babu. 2018. Ask, Acquire, and Attack: Data-Free UAP Generation Using Class Impressions. In *the European Conference on Computer Vision (ECCV)*.

[28] Sherjil Ozair and Yoshua Bengio. 2014. Deep directed generative autoencoders. *arXiv preprint arXiv:1410.0630* (2014).

[29] Nicolas Papernot, Ian Goodfellow, Ryan Sheatsley, Reuben Feinman, and Patrick McDaniel. 2016. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768* (2016).

[30] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on.* IEEE, 372–387.

[31] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on.* IEEE, 582–597.

[32] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2016. Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples. *CoRR* abs/1602.02697 (2016).

[33] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J Fleet. 2015. Adversarial manipulation of deep representations. *arXiv preprint arXiv:1511.05122* (2015).

[34] Swami Sankaranarayanan, Arpit Jain, Rama Chellappa, and Ser Nam Lim. 2017. Regularizing deep networks using efficient layerwise adversarial training. *arXiv preprint arXiv:1705.07819* (2017).

[35] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. 2017. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766* (2017).

[36] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *CoRR* abs/1312.6199 (2013). http://arxiv.org/abs/1312.6199

[37] Thomas Tanay and Lewis Griffin. 2016. A Boundary Tilting Persepective on the Phenomenon of Adversarial Examples. *arXiv preprint arXiv:1608.07690* (2016).

[38] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688 (May 2016). http://arxiv.org/abs/1605.02688