Improving Robustness of time series classifier with Neural ODE guided gradient based data augmentation

Anindya Sarkar
Mobiliya
Bangalore, INDIA
anindya.sarkar@mobiliya.com

Anirudh Sunder Raj
*Mobiliya**
Bangalore, INDIA
anirudh.s@mobiliya.com

Raghu Sesha Iyengar
Mobiliya
Bangalore, INDIA
raghu.iyengar@mobiliya.com

***Abstract*—Exploring adversarial attack vectors and studying their effects on machine learning algorithms has been of interest to researchers. Deep neural networks working with time series data have received lesser interest compared to their image counterparts in this context. In a recent finding, it has been revealed that current state-of-the-art deep learning time series classifiers are vulnerable to adversarial attacks. In this paper, we introduce two local gradient based and one spectral density based time series data augmentation techniques. We show that a model trained with data obtained using our techniques obtains state-of-the-art classification accuracy on various time series benchmarks. In addition, it improves the robustness of the model against some of the most common corruption techniques,such as Fast Gradient Sign Method (FGSM) and Basic Iterative Method (BIM).**

***Index Terms*—time series classification, adversarial training, gradient based adversarial attacks**

I. Introduction and Related Work

Deep Neural Networks have displayed impressive results on many machine learning tasks on image ([1]–[5]), natural language processing ([6]–[9]) and time series classification ([10]–[13]). However, their fragility to small adversarial perturbations is a matter of concern for researchers. A targeted black-box attack on neural networks for image classification was formulated as an optimization problem in [14], and further improved in [15]. A targeted l_0 norm attack discussed in [16], aims to minimize the number of modified pixels in an image to cause mis-classification as a particular target class. Adversarial attacks intended to lower reliability of neural networks are also explored. Of these, gradient based l_∞ norm attacks such as [17] and [18] are very popular. Various techniques to understand and mitigate effects of adversarial perturbations have also been studied ([19]–[25]). An excellent review of adversarial attacks on machine learning systems can be found in [26]. The reliability and security concerns raised by adversarial attacks have been one of the main reasons for deep neural networks not yet becoming popular with safety critical applications where the cost of failure is high.

Work sponsored by Mobiliya, a QuEST Global company.
*Work done during Internship at Mobiliya.

Time series data is omnipresent and classification tasks on time series data finds its applications in health care ([27]), power consumption monitoring ([28]), food safety ([29], [30]), security ([31]) etc. Current state-of-the-art deep neural networks can achieve impressive performance at classifying time series data ([10]–[12]) on various datasets ([32], [33]). However, these networks suffer in the same way to adversarial inputs as their image counterparts. A recent finding ([34]) shows that vulnerability of state-of-the-art time series classification networks to simple adversarial attacks, bring back the focus on building more robust time series classifiers.

It has been shown ([35], [36]) that data augmentation increases the size and diversity of the training set resulting in improved classification accuracy on time series data. Further, in order to achieve adversarial robustness, the classifiers should be robust to noise-corrupted data ([37]). In this paper, we propose data augmentation techniques for time series data, which helps in improving the robustness of the classifier against adversarial attacks. Our contributions can be summarized as follows:

1) A data augmentation technique based on black-box threat model using the local gradients of the input to obtain additional data to train a deep neural network time series classifier. The classifier trained with this augmented data achieves state-of-the-art classification accuracy on adversarially perturbed UCR time series datasets using FGSM and BIM attacks.
2) A data augmentation technique based on white-box threat model using the local gradients of the output to obtain additional data to train a deep neural network time series classifier. The classifier trained with this augmented data further improved the state-of-the-art classification accuracy on adversarially perturbed UCR time series datasets using FGSM and BIM attacks.
3) We also propose a novel spectral density based adversarial sample generation technique. We demonstrate that baseline classifiers suffer a huge drop in classification

accuracy when tested with such an adversarial data. We further train the baseline classifier with this data and show that it improves the classification accuracy with standard adversarial attacks such as FGSM and BIM.

For models to be useful in the real world, they need to be both accurate on a held-out set of time series data, which we refer to as *clean accuracy*, and robust on corrupted time series, which we refer to as *robustness*. It is believed that there exists a fundamental trade-off between the two ([38]). Our observation has been that though the trade-off exists, it is possible to build robust systems with very minimal or no drop in clean accuracy.

II. Generating Adversarial Samples

In the time series classification paradigm, neural networks are maximum likelihood estimators. The neural network tends to learn the input features that are important for classification, even if those features look incomprehensible to humans. It has been shown that simple gradient based perturbations in the input signal can cause a trained network to misclassify ([17], [18]). We first assume a threat model (black-box or white box) and generate adversarial samples based on that model. We augment original train data with adversarial samples and then train a reference neural network with this data. We evaluate the trained network on standard adversarial attacks on time series data ([34]).

Recently, neural networks were used to model ordinary differential equations ([39]). This allowed predicting the gradient of a function for any value of the time variable and hence, better in predicting timeseries samples. We use ODENet to predict the gradients of input timeseries for the methods described below.

A. Input gradient based adversarial sample generation

Here, we assume a black-box threat model. The adversary does not have any knowledge of the model or its parameters, but can perform decision time attacks on the input data. Given a time series data, we are interested in generating adversarial samples for reliability attack. To do this, we perturb the input samples based on the local gradient of the input timeseries. To obtain the gradients, we parameterize the continuous dynamics of the input time series using an ordinary differential equation (ODE):

$$\frac{dx}{dt} = g(x(t), t, \theta) \tag{1}$$

where $x(t)$ is the input time series, $g()$ is the function approximated using a neural network with parameters θ. ODENet uses standard ODE Solvers as sequential generative models to predict successive samples of the timeseries. We use mean square error between the predicted samples and the true samples as the loss function to optimize ODENet. Once the ODENet is trained, we use it to compute the gradient of a timeseries using:

$$\frac{dx(i)}{dt} = g(x(i), i, \theta_{optimum}) \tag{2}$$

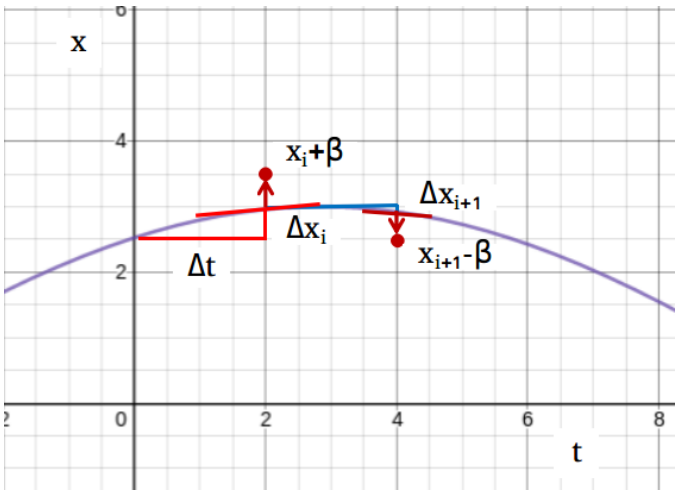

Fig. 1. *Input gradient based adversarial sample generation*. Each point in the timeseries is perturbed by a random number in range [0,ϵ]. The direction of change is determined by the local gradient of x.

where $x(i)$ is the value of input timeseries at timestep i. $\theta_{optimium}$ are the trained weights of the neural network $g()$.

The adversarial samples are generated using the below equation:

$$x_p(i) = x(i) + clamp(\frac{dx(i)}{dt}) \tag{3}$$

$$clamp(x) = \begin{cases} x & |x| < \beta \\ \beta & x > \beta \\ -\beta & x < \text{-}\beta \end{cases} \tag{4}$$

$x_p(i)$ is the perturbed timeseries at timestep i, $x(i)$ is the original timeseries at timestep i and β is a small positive constant. Note that by perturbing the samples as given in equation (3), the change in magnitude of each sample in the adversarial timeseries has an upper limit of β compared to the original timeseries. This inherently sets an upper limit on the change in gradient to $2 * \beta/\delta t$, as shown in figure 1. Figure 2(a,b) shows example timeseries data augmented using this technique.

We define a multiclass classification setup where the input-label pairs $(x, y) \in (\chi \text{ x } \lambda)$ are sampled from data distribution D. A neural network is trained as a classifier whose goal is to predict the class label y for a given input x. A feature f is defined to be a function mapping from the input space χ to the real numbers $\mathbb{R}$, with the set of all features thus being $F = f : \chi \rightarrow \mathbb{R}$. A neural network is trained to obtain parameters θ that minimize the multiclass cross entropy loss:

$$\min_{\theta} \mathbb{E}_{(\mathrm{x},\mathrm{y})}[\mathrm{Loss}(\mathrm{f}_\theta(\mathrm{x}+\delta, \mathrm{y}))] \tag{5}$$

We augment the input dataset with the perturbed adversarial examples obtained using equation (3). The labels for the perturbed examples are set to be same as the original example from which it was obtained. Thus, training the neural network classifier with this augmented dataset follows 5.

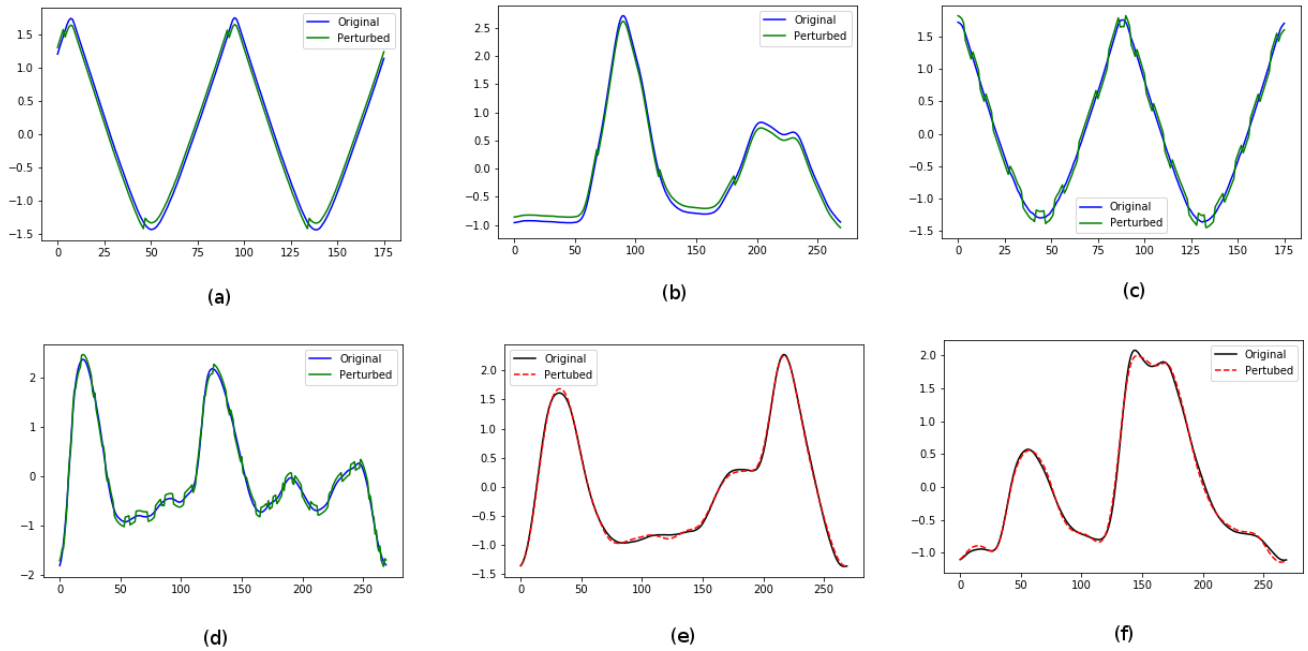

Fig. 2. Two randomly selected samples and their corresponding adversarial samples obtained using **(a),(b)** *Input gradient*, **(c),(d)** *Output gradient*, **(e),(f)** *Spectral Density* based adversarial sample generation techniques. **(a),(c) and (e)** are from Adiac time series dataset while **(b),(d) and (f)** from 50words dataset

We also experimented with a slight modification of equation (3) as given below:

$$x_p(i) = x(i) + \epsilon * sign(\frac{dx(i)}{dt}) \tag{6}$$

Here, ϵ is a random value between 0.0 and 0.33.

B. Output gradient based adversarial sample generation

In this case, we assume a white-box threat model. The adversary has knowledge of the model or its parameters. The attack is still done in decision time. Given a time series data, we are interested in generating adversarial samples for reliability attack. To do this, we perturb the input samples based on the local gradient of the output timeseries. Let the forward propagation in the neural network classifier be represented as:

$$y = f(x) \tag{7}$$

where y is the classifier output, $f()$ is the transformation applied by the neural network and x is the input timeseries data. The output derivative w.r.t the time variable is then given by:

$$y' = f'(x)\frac{dx}{dt} \tag{8}$$

As done in section II-A, $\frac{dx}{dt}$ is computed using an ODENet. $f'(x)$ is computed using the automatic differentiation in Pytorch ([40]). Using the derivative of output, we define a perturbed time series as below:

$$x_p(t) = x(t) + sign(f'(x))\ abs(clamp(\frac{dx}{dt})) \tag{9}$$

where, $sign(f'(x))$ is an indication of the direction of gradient of the output of neural network w.r.t input. For cross entropy loss, changing the input along this direction maximizes the loss term. $\frac{dx}{dt}$ is the gradient of input w.r.t time. The function $clamp$ is defined in equation (4). Thus, the direction of perturbation is defined by $(f'(x))$, while the magnitude of perturbation is defined by $\frac{dx}{dt}$ and it is limited by a small positive number (β). As done in section II-A, we augmented the input dataset with perturbed samples and trained a neural network for classification. The trained neural network achieves state-of-the-art results on different UCR time series test datasets with FGSM and BIM perturbations. Figure 2(c,d) shows example timeseries data augmented using this technique.

To validate the significance of $sign(f'(x))$ in the generation of perturbed samples, we replaced this with random sign for each time step.

$$x_p(t) = x(t) + random([-1, +1])\ abs(clamp(\frac{dx}{dt})) \tag{10}$$

A neural network trained with such a perturbed data was found to provide lesser accuracy compared to the network trained using the augmented data obtained using $sign(f'(x))$.

C. Spectral density based adversarial sample generation

In the techniques described in previous sections, we introduced perturbations at each timestep of the input timeseries. The perturbations added were functions of derivatives of inputs and/or outputs. In this section, we describe the perturbations

added based on the frequency domain characteristics of the input signals. The energy of a signal is given by Parseval's theorem:

$$E(x) = \sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2 \quad (11)$$

Where, $X[k]$ is the Discrete Fourier Transform of $x[n]$, both of length N.

We now obtain a new perturbed Discrete Fourier Transform $X_p[k]$ as described below. The sequence $X[k]$ is sorted in descending order to obtain X_{sorted}.

$$X_{sorted} = sort(|X[k]|) \quad (12)$$

We then find the index C such that:

$$\sum_{k=0}^{C} |X_{sorted}[k]|^2 \geq 0.9 * E(x) > \sum_{k=0}^{C-1} |X_{sorted}[k]|^2 \quad (13)$$

We apply perturbations to the frequency components in the sorted sequence ($X_{sorted}[k]$) which lie above the index C. Let X_1, X_2 and X_3 be three consecutive frequency components in $X_{sorted}[k]$ which lie above the index C, while X_{1p}, X_{2p} and X_{3p} be the corresponding perturbed components. Then, the perturbed frequency domain components are obtained as:

$$X_{1p} = X_1 + \frac{X_2}{4},\ X_{2p} = X_2 + \frac{X_2}{2},\ X_{3p} = X_3 + \frac{X_2}{4} \quad (14)$$

We apply such perturbation on upto 75% of the components which lie above the index C. All components below index C remain same. The perturbed time domain timeseries is obtained by taking inverse DFT of the perturbed frequency domain components. Apart from redistributing the energy of the lesser-significant frequency components, this technique also modifies the energy of overall signal. It can be shown that the change in energy of the signal due to a single perturbation defined by equation (14) has an upper bound of 1.25% (Ref. Appendix) . Figure 2(e,f) shows example timeseries data augmented using this technique.

III. Experiments and Results

A. Network Architecture

1) ODENet: The timeseries dynamics of the input are modelled using differential equations. A neural network is trained to compute the gradients of input timeseries at any given timestep (equation 1). The neural network $g()$ consists of two hidden layers of 25 neurons. Each hidden layer is followed by batch normalization ([42]) and ELU activation ([43]). The value of the input timeseries at a given time $(x(t_i))$ and the timestep (t_i) form the input to the neural network. The network predicts the gradient of input signal at timestep (t_i).

2) Classification Network: The neural network for time-series classification follows the architecture as defined in [34]. The network architecture, reproduced from [34] is shown in figure 3. The input to this network is a time series of length T. The output of the network is a probability distribution over the K possible classes in the dataset. The network consists of 9

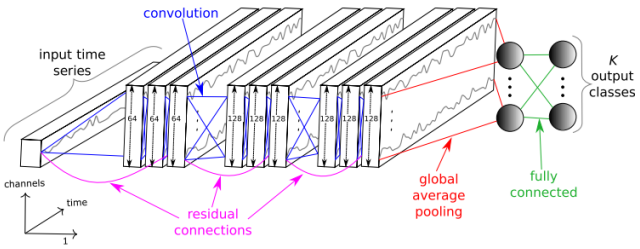

Fig. 3. Reference ResNet architecture of the neural network used for all experiments in this paper (from [34])

TABLE I
Datasets and baseline classification accuracy

Dataset	True	FGSM	BIM
CricketX	79.0	35.4	20.8
CricketZ	81.5	27.7	16.2
50Words	73.2	17.1	8.8
UWaveGestureLibraryX	78.0	32.1	11.1
InsectWingBeatSound	50.6	17.7	15.7
Adiac	83.1	3.1	1.5
TwoLeadECG	100	5.3	0.4
UWaveGestureLibraryY	66.7	27.7	14.9

convolutional layers, grouped into three residual blocks. Each layer is followed by a Rectified Linear (ReLU) activation ([41]) and batch normalization ([42]). This is followed by a global average pooling layer and a softmax classification layer. We retain the same network architecture and train it with the dataset augmented with adversarial samples. This allows us to compare the effectiveness of our techniques with the baseline in [34].

B. Datasets

We use the UCR timeseries dataset ([32]) for all the experiments described in this paper. Table I summarizes the datasets used, along with the classification accuracy on the state-of-the-art resnet classifiers, classification accuracy on the data perturbed using FGSM ([17]) and BIM ([18]) on the state-of-the-art resnet classifiers (as reported in [34]).

Using the techniques described in section II, we augment the datasets listed in table I. We use the augmented dataset to train the network defined in section III-A2. With this technique, we hope to improve the classification accuracy on adversarial samples without significant reduction in the classification accuracy on normal samples. We use the adverarial datatsets provided by [34] to test our models. The timeseries signals of all the datasets used in our experiments displayed an amplitude range close to 6 units. The value of β (equation 4) was chosen to be 0.33 and ϵ (equation 6) was a random number in range [0.0,0.33] for all the experiments described in this paper. .

C. Training

1) ODENet: The values for the function $g(x(t), t, \theta)$ (equation 1) need to be evaluated for various timesteps to generate adversarial samples based on local gradient of the input. We use the ODENet described in section III-A1 for this purpose.

TABLE II
ADVERSARIAL SAMPLE GENERATION TECHNIQUES

Technique	Sample Generation
clamp_dxdt	equation 3
epsilon_dxdt	equation 6
dydt_clamp_dxdt	equation 9
random_clamp_dxdt	equation 10
spectral_dens	equation 14

ODENet uses a neural network to predict the gradient at any given timestep and a standard ODE solver to predict the value of timeseries at any timestep. We minimize the MSE loss between the predicted value and true value of the timeseries at every timestep. Adam optimizer with a learning rate of 0.0003 and weight decay factor of 10^{-3} was used for training the ODENet.

2) Classification Network: We train the classifier network defined in section III-A2 using input dataset augmented with different types of adversarial samples. Table II summarizes the types of adversarial samples generated used in various experiments described in this paper. The training set consisted of the original timeseries data and the adversarial data generated using one or more of the techniques described in the table II.

We used Adam optimizer with a learning rate of 0.0002 with a weight decay of 10^{-3} for all the experiments involving training the classifier network in this paper.

D. Additional Techniques

In addition to the techniques already described for adversarial sample generation, we experimented with some more techniques to improve the classification accuracy. We describe them below:

1) Discretization: As demonstrated by [44], encoding the input provides resistance against adversarial attack. Hence, we preprocess the original sequence $x(t)$ and the perturbed sequence $x_p(t)$ by discretizing them before training. Step sizes of 0.05 and 0.1 were used in our experiments. We observed that the larger step size provided better adversarial accuracy (against FGSM and BIM attacks), but at the cost of normal classification accuracy.

2) Target Encoding: Target encoding instead of softmax layer was suggested by [45] as a way of increasing adversarial robustness. We use Hadamard code generated using a method provided by [45] to represent the target classes. The classification neural network is modified to remove the softmax layer, increase the output length to match the code length and use a tanh activation function. The network was trained to reduce the mean squared error between the network output and the target code of the class. We observed that with target encoding, the adversarial accuracy increases but the normal classification accuracy reduces.

3) Feature Similarity: Distance metric for large margin nearest neighbour classification was introduced in [46] and later used for face recognition and clustering in [47]. Let x_c be a time series sample of a specific class c. Let x_p be the adversarial sample obtained by perturbing x_c. Let $x^i_{c'}$ be a randomly chosen time series sample of any other class $c' \neq c$. The adversarial sample obtained by perturbing $x^i_{c'}$ is given by $x^i_{p'}$. $f(x) \in \mathbb{R}$ represents the d-dimensional embedding of the timeseries x obtained by the neural network. Then, the loss function that is being minimized has an additional term (along with cross entropy) given by

$$L_{FS} = ||f(x_c) - f(x_p)||_2^2 + \alpha - ||f(x_c) - f(x^i_{p'})||_2^2 \quad (15)$$

$$\forall (f(x_c), f(x_p), f(x^i_{p'})) \in \mathbb{T} \quad (16)$$

where α is a margin that is enforced between positive and negative pairs. $\mathbb{T}$ is the set of all possible triplets in the training set. We also ensured that the embedding to lives on the d-dimensional hypersphere, i.e. $||f(x)||^2 = 1$.

E. Results

In this section, we describe and summarize the classification accuracy observed on the reference neural network (section III-A2), where the training dataset is augmented using adversarial data. Table I summarizes the true classification accuracy and the classification accuracies obtained with FGSM and BIM perturbations on the reference neural network. We then generate adversarial samples using techniques summarized in table II. The reference neural network is then trained using a training set which is augmented using the generated adversarial data. Table III shows a summarizes the results of our experiments.

1) Networks trained using adversarial samples: We observe that even the very basic blackbox techniques (equations 3, 6) significantly improve the adversarial classification accuracies (FGSM and BIM). For every dataset, we highlight the blackbox technique that provided highest adversarial classification accuracy with **blue** and the whitebox technique that provided highest adversarial classification accuracy with **green**. In both cases, the true accuracy was either same or better than the true classfication accuracy of reference network. Reference network trained with adversarial data obtained using whitebox techniques performed much better than the ones trained with data obtained using blackbox techniques. This made intuitive sense because whitebox techniques use the gradient of output of the network while blackbox techniques use no knowledge of the network.

2) Networks trained using additional techniques: We also observed that there was significant improvement in the adversarial classification accuracy when we employed additional techniques described in section III-D on the adversarial training data. These are summarized in table IV. It can be seen that a network trained with *dydt_clamp_dxdt* + *spectral_dens* + *feature_sim* (refer table II) train data provided the best adversarial classification accuracies (on FGSM and BIM) along with significant improvements in true accuracies for almost all cases. However, different blackbox techniques were found to provide best adversarial classification accuracies for different datasets. This too made intuitive sense because blackbox techniques are based on the gradients of the input signal itself, and hence it was unlikely that any

TABLE III
ADVERSARIAL CLASSIFICATION ACCURACY ON NETWORKS WHERE TRAINING DATA IS AUGMENTED USING TECHNIQUES DESCRIBED IN SECTION II

Dataset	Blackbox Methods						Whitebox Methods						Spectral Density		
	clamp_dxdt			epsilon_dxdt			dydt_clamp_dxdt			random_clamp_dxdt*			spectral_dens		
	True	FGSM	BIM	True	FGSM	BIM	True	FGSM	BIM	True	FGSM	BIM	True	FGSM	BIM
CricketX	79.74	44.36	31.28	79.49	**47.95**	**38.72**	79.49	**72.31**	**72.05**	78.21	41.54	31.03	79.23	46.67	37.44
CricketZ	82.31	40.00	30.00	82.31	**43.33**	**34.87**	81.79	**42.82**	**37.18**	76.92	34.87	27.95	81.03	40.26	33.08
50Words	74.73	**38.46**	**29.23**	74.29	33.19	16.70	76.04	**67.25**	**52.97**	74.29	33.63	26.59	74.07	25.5	12.09
UWaveGestureLibrary_X	78.03	46.18	**21.11**	78.28	**46.76**	20.91	79.23	**71.19**	**55.05**	76.33	56.53	34.09	77.11	40.90	19.88
InsectWingbeatSound	50.81	**25.81**	**21.72**	49.29	27.27	23.23	53.48	**39.14**	**38.13**	48.44	26.06	23.79	51.16	25.20	20.05
Adiac	83.38	**6.39**	**3.84**	82.10	6.14	4.86	83.63	**45.27**	**25.83**	81.07	17.14	11.76	83.38	5.63	4.09
TwoLeadECG	100	13.61	6.85	100	14.05	**7.64**	100	**61.98**	**31.69**	99.74	17.38	6.85	100	**14.72**	7.20
UWaveGestureLibrary_Y	67.0	35.32	**18.31**	67.03	**36.24**	17.64	67.06	**57.96**	**47.35**	65.01	35.21	17.61	66.14	35.87	18.01

* Not a whitebox technique, but used here for comparison

TABLE IV
ADVERSARIAL CLASSIFICATION ACCURACY ON NETWORKS TRAINED USING ADDITIONAL TECHNIQUES DESCRIBED IN SECTION III-D

Dataset	clamp_dxdt+ discretization			epsilon_dxdt+ discretization			epsilon_dxdt+ spectral_dens			dydt_clamp_dxdt+ spectral_dens			dydt_clamp_dxdt+ spectral_dens+ feature_sim		
	True	FGSM	BIM	True	FGSM	BIM	True	FGSM	BIM	True	FGSM	BIM	True	FGSM	BIM
CricketX	79.23	47.18	37.44	79.23	**51.28**	**42.05**	80.0	48.46	38.46	79.49	73.08	73.85	79.49	**75.90**	**76.41**
CricketZ	81.54	40.00	31.03	82.31	**45.90**	35.90	83.59	43.85	**36.15**	81.77	59.23	55.64	81.73	**66.41**	**66.15**
50Words	74.07	**40.66**	**30.33**	73.20	34.73	19.34	74.07	34.51	19.56	74.95	69.23	55.16	76.04	**69.23**	**55.82**
UWaveGestureLibraryX	78.42	**51.90**	**28.25**	78.03	47.74	24.06	77.08	52.93	35.34	78.50	**72.78**	61.31	78.53	72.47	**62.62**
InsectWingBeatSound	49.85	26.41	22.42	49.29	28.23	23.38	50.81	**32.47**	**28.79**	51.82	40.30	38.74	51.87	**43.74**	**42.17**
Adiac	79.28	7.16	4.09	82.61	6.65	4.86	81.59	6.91	4.35	83.89	41.94	27.88	83.38	**46.29**	**28.64**
TwoLeadECG	100	**38.81**	**29.06**	100	29.24	21.60	100	14.75	7.99	100	71.91	25.11	100	**73.40**	**30.38**
UWaveGestureLibraryY	66.86	36.71	20.69	66.80	36.99	18.57	66.95	**41.54**	**25.43**	67.50	59.30	48.19	67.07	**59.35**	**58.15**

one technique would give best results for all datasets. We also observed that as we introduce additional techniques to adversarial data obtained using whitebox techniques (for example, *dydt_clamp_dxdt + spectral_dens*, *dydt_clamp_dxdt + spectral_dens + feature_sim*), the adversarial accuracies improved progressively. Similarly, *discretization* on adversarial data obtained using blackbox techniques always improved the adversarial classification accuracy with minimal or no change in true accuracy.

3) Principal Component Analysis: In order to visualize the effect of adversarial training on the latent representation of timeseries samples, we obtained PCA plots of the latent representation of randomly selected samples from the 50Words dataset. This is shown in figure 4. Here, we can observe that the clustering of samples are better in (b) and (c) (networks trained with adversarial samples obtained using blackbox techniques) compared to (a) (reference network). However, (d), (e) and (f) (networks trained with adversarial samples obtained using whitebox techniques) progressively improve the clustering, especially for the samples that lie at the intersection of clusters formed by different classes. This is inline with the results observed in table IV.

IV. DISCUSSION

In this paper, we have introduced gradient based adversarial sample generation techniques which were developed using both blackbox and whitebox threat models of adversarial attacks. We showed that networks trained using these adversarial samples were more robust against standard adversarial attacks without compromising the true accuracy. We have also introduced spectral density based adversarial sample generation techniques which further improved the robustness of the reference classification network. We have also demonstrated that (refer figure 4) networks trained with the generated adversarial data are able to cluster the samples in latent space much better than the reference network. However, (figure 5) we have also observed that the samples belonging to certain classes are still inseparable in latent space. We would like to investigate this further and build more robust time series classifiers in our future work.

REFERENCES

[1] Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. Retrieved from http://arxiv.org/abs/1612.08242

[2] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. Retrieved from http://arxiv.org/abs/1602.07261

[3] He, K., Gkioxari, G., Dollr, P., & Girshick, R. (2017). Mask R-CNN. Retrieved from http://arxiv.org/abs/1703.06870

[4] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Retrieved from http://arxiv.org/abs/1905.11946

[5] Zhao, Z.-Q., Zheng, P., Xu, S.-T., & Wu, X. (2019). Object Detection With Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems, 121. https://doi.org/10.1109/TNNLS.2018.2876865

[6] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. Retrieved from http://arxiv.org/abs/1409.3215

[7] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. Retrieved from http://arxiv.org/abs/1301.3781

[8] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Retrieved from http://arxiv.org/abs/1810.04805

[9] Otter, D. W., Medina, J. R., & Kalita, J. K. (2018). A Survey of the Usages of Deep Learning in Natural Language Processing. Retrieved from http://arxiv.org/abs/1807.10854

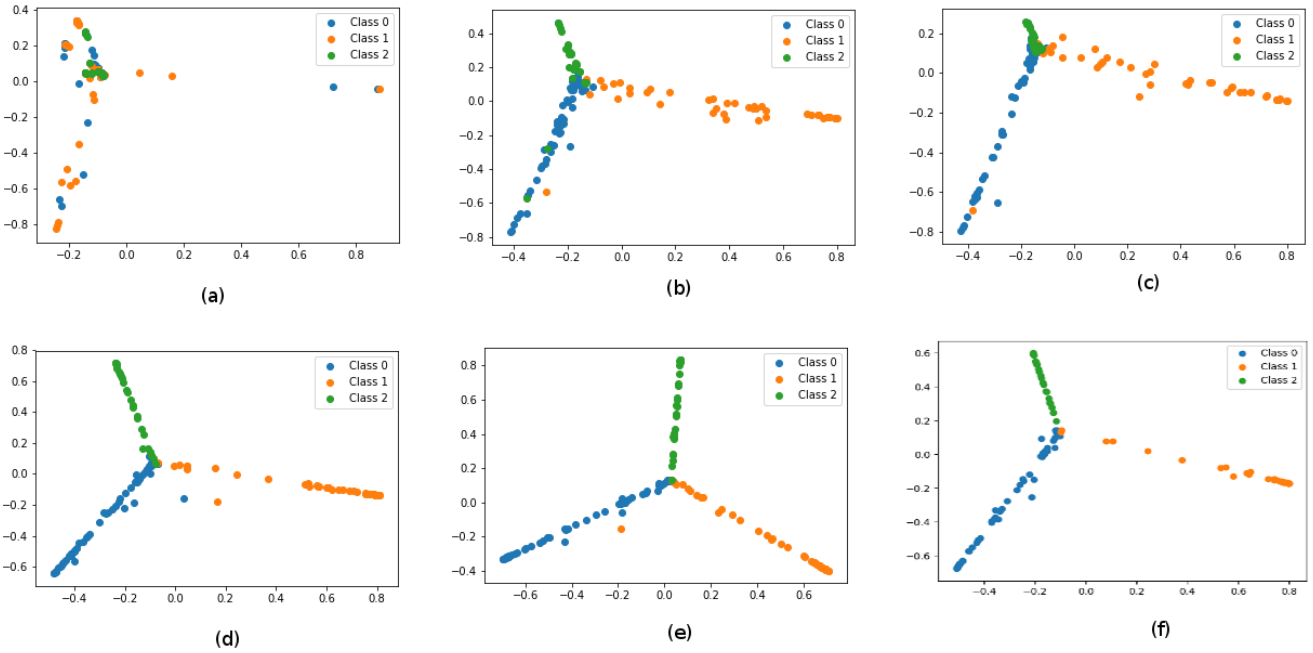

Fig. 4. PCA plots of latent representation of randomly selected samples from the 50Words dataset for various networks. **(a)** reference network, **(b)** clamp_dxdt, **(c)** clamp_dxdt + discretization, **(d)** dydt_clamp_dxdt **(e)** dydt_clamp_dxdt + spectral_dens, **(f)** dydt_clamp_dxdt + spectral_dens + feature_sim

[10] Wang, Z., Yan, W., & Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. Proceedings of the International Joint Conference on Neural Networks, 2017May, 15781585. https://doi.org/10.1109/IJCNN.2017.7966039

[11] Karim, F., Majumdar, S., Darabi, H., & Harford, S. (2019). Multivariate LSTM-FCNs for time series classification. Neural Networks, 116, 237245. https://doi.org/10.1016/j.neunet.2019.04.014

[12] Karim, F., Majumdar, S., & Darabi, H. (2019). Insights into LSTM Fully Convolutional Networks for Time Series Classification, 17. Retrieved from http://arxiv.org/abs/1902.10756

[13] Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: a review. Data Mining and Knowledge Discovery, 33(4), 917963. https://doi.org/10.1007/s10618-019-00619-1

[14] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks, 110. Retrieved from http://arxiv.org/abs/1312.6199

[15] Carlini, N., & Wagner, D. (2017). Towards Evaluating the Robustness of Neural Networks. Proceedings - IEEE Symposium on Security and Privacy, 3957. https://doi.org/10.1109/SP.2017.49

[16] Papernot, N., Mcdaniel, P., Jha, S., Fredrikson, M., Celik, Z. B.,& Swami, A. (2016). The limitations of deep learning in adversarial settings. Proceedings - 2016 IEEE European Symposium on Security and Privacy, EURO S and P 2016, 372387. https://doi.org/10.1109/EuroSP.2016.36

[17] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and Harnessing Adversarial Examples, 111. Retrieved from http://arxiv.org/abs/1412.6572

[18] Kurakin, A., Goodfellow, I., & Bengio, S. (2016). Adversarial examples in the physical world, (c), 114. Retrieved from http://arxiv.org/abs/1607.02533

[19] Azulay, A. & Weiss, Y. Why do deep convolutional networks generalize so poorly to small image transfor- mations? CoRR, abs/1805.12177, 2018. URL http: //arxiv.org/abs/1805.12177.

[20] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards Deep Learning Models Resistant to Adversarial Attacks, 127. Retrieved from http://arxiv.org/abs/1706.06083

[21] Raghunathan, A., Steinhardt, J., & Liang, P. (2018). Certified Defenses against Adversarial Examples, 115. Retrieved from http://arxiv.org/abs/1801.09344

[22] Wong, E., & Kolter, J. Z. (2017). Provable defenses against adversarial examples via the convex outer adversarial polytope. Retrieved from http://arxiv.org/abs/1711.00851

[23] Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016). Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016, 582597. https://doi.org/10.1109/SP.2016.41

[24] Tramr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., & McDaniel, P. (2017). Ensemble Adversarial Training: Attacks and Defenses, 120. Retrieved from http://arxiv.org/abs/1705.07204

[25] Lu, J., Issaranon, T., & Forsyth, D. (2017). SafetyNet: Detecting and Rejecting Adversarial Examples Robustly. Proceedings of the IEEE International Conference on Computer Vision, 2017October, 446454. https://doi.org/10.1109/ICCV.2017.56

[26] Yevgeniy Vorobeychik, Murat Kantarcioglu (2018). Adversarial Machine Learning. In Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. isbn:9781681733968

[27] S. M. Abdelfattah, G. M. Abdelrahman, & M. Wang, Augmenting the size of EEG datasets using generative adversarial networks, in International Joint Conference on Neural Networks, 2018, pp. 16.

[28] Z. Zheng, Y. Yang, X. Niu, H. Dai, and Y. Zhou, Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids, IEEE Transactions on Industrial Informatics, vol. 14, no. 4, pp. 16061615, 2018.

[29] R. Briandet, E. K. Kemsley, & R. H. Wilson, Discrimination of Arabica and Robusta in instant coffee by Fourier transform infrared spectroscopy and chemometrics, Journal of agricultural and food chemistry, vol. 44, no. 1, pp. 170174, 1996

[30] A. Nawrocka and J. Lamorska, Determination of food quality by using spectroscopic methods, in Advances in agrophysical research, 2013.

[31] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, & T. Yagi, Malware detection with deep neural network using process behavior, in IEEE Annual Computer Software and Applications Conference, 2016, pp. 577582.

[32] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, & G. Batista, The UCR time series classification archive, 2015.

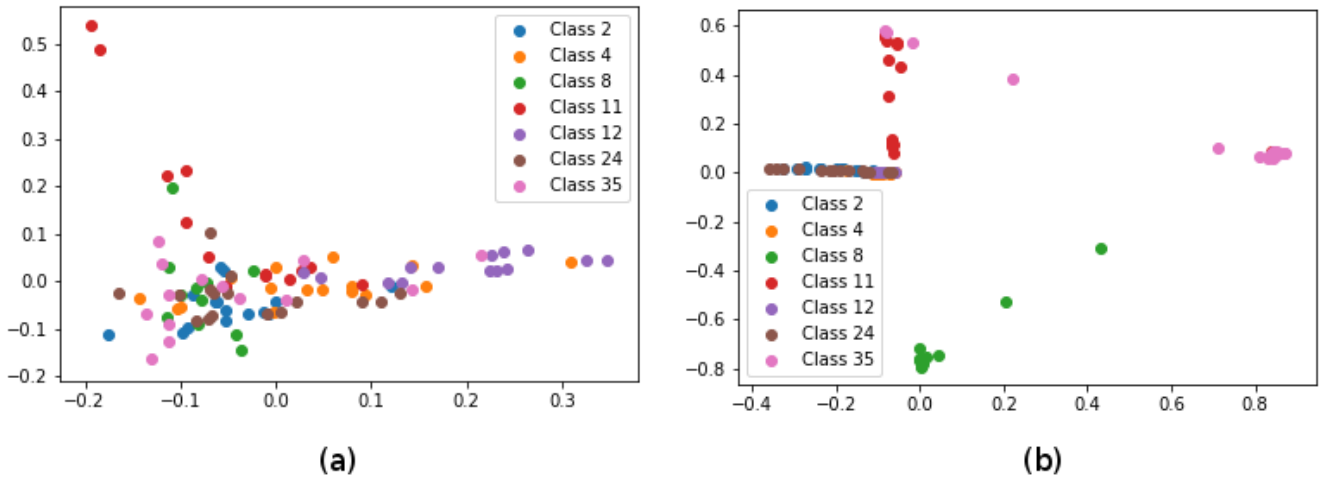

Fig. 5. PCA plots of latent representation of randomly selected samples from the Adiac dataset for **(a)** reference network and **(b)** dydt_clamp_dxdt + spectral_dens + feature_sim

[33] Bagnall, A., Dau, H. A., Lines, J., Flynn, M., Large, J., Bostrom, A., Keogh, E. (2018). The UEA multivariate time series classification archive, 2018, 136. Retrieved from http://arxiv.org/abs/1811.00075

[34] Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Adversarial Attacks on Deep Neural Networks for Time Series Classification. Retrieved from http://arxiv.org/abs/1903.07054

[35] Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2018). Data augmentation using synthetic data for time series classification with deep residual networks. Retrieved from http://arxiv.org/abs/1808.02455

[36] Forestier, G., Petitjean, F., Dau, H. A., Webb, G. I., & Keogh, E. (2017). Generating synthetic time series to augment sparse datasets. In V. Raghavan, S. Aluru, G. Karypis, L. Miele, & X. Wu (Eds.), Proceedings: 17th IEEE International Conference on Data Mining (pp. 865-870). Piscataway NJ USA: IEEE, Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/ICDM.2017.106

[37] Ford, N., Gilmer, J., Carlini, N., & Cubuk, D. (2019). Adversarial Examples Are a Natural Consequence of Test Error in Noise. Retrieved from http://arxiv.org/abs/1901.10513

[38] Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., & Madry, A. (2018). Robustness May Be at Odds with Accuracy, 124. Retrieved from http://arxiv.org/abs/1805.12152

[39] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural Ordinary Differential Equations, (NeurIPS). Retrieved from http://arxiv.org/abs/1806.07366

[40] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. & Lerer, A. Automatic differentiation in pytorch. 2017.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet classification with deep convolutional neural networks, in Advances in Neural Information Processing Systems 25, 2012, pp. 10971105.

[42] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in International Conference on Machine Learning, 2015, pp. 448456

[43] Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). CoRR, abs/1511.07289, 2016.

[44] Buckman, J., Roy, A., Raffel, C., and Goodfellow, I. Thermometer encoding: One hot way to resist adversarial examples. 2018. URL https://openreview.net/pdf?id=S18Su–CW

[45] Yang, S., Luo, P., Loy, C. C., Shum, K. W., & Tang, X. Deep representation learning with target coding. In Pro- ceedings ofthe Twenty-Ninth AAAI Conference on Artifi- cial Intelligence, AAAI15, pp. 38483854. AAAI Press, 2015. ISBN 0-262-51129-0. URL http://dl.acm.org/citation.cfm?id=2888116.2888250.

[46] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metriclearning for large margin nearest neighbor classification. InNIPS. MIT Press, 2006. 2, 3

[47] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. https://doi.org/10.1109/CVPR.2015.7298682

V. Appendix

if $X_1, X_2 \in R$ with $X_1, X_2 > 0$ and $|X_1| > |X_2|$, then

$$|X_1 + \delta|^2 > |X_2 + \delta|^2 \tag{17}$$

Thus, if X_1, X_2, X_3 are the magnitudes of three consecutive frequency components from X_{sorted}, then the perturbed components are given by equation 14. The energy difference in these components introduced due to this operation is given by:

$$E_\delta = (|X_{1p}|^2 + |X_{2p}|^2 + |X_{3p}|^2) - (|X_1|^2 + |X_2|^2 + |X_3|^2) \tag{18}$$

To maximize the difference between $X1$ and $X1p$, the component added to X_1 (i.e., $X_2/4$) should be maximum. But since $X_1 \geq X_2$, $X_2 = X_1$ provides the maximum difference between X_1 and X_{1p}. Along the same lines, we find that E_δ is maximized when $X_1 = X_2 = X_3$.

$$max(|X_{1p}^2 - X_1^2|) = (X_1 + \frac{X_1}{4})^2 - X_1^2 \tag{19}$$

$$max(|X_{2p}^2 - X_2^2|) = (X_1 - \frac{X_1}{2})^2 - X_1^2 \tag{20}$$

$$max(|X_{3p}^2 - X_3^2|) = (X_1 + \frac{X_1}{4})^2 - X_1^2 \tag{21}$$

Or,

$$max(E_\delta) = 3.375X_1^2 - 3X_1^2 \tag{22}$$

$$\frac{max(E_\delta)}{E_{original}} = \frac{3.375X_1^2 - 3X_1^2}{3X_1^2} = 0.125 \tag{23}$$

Where $E_{original}$ is the energy of the three signal components that are perturbed. Thus, the maximum difference in energy after the perturbation (equation 14) is 12.5% of the original energy present in the three components that are perturbed.

We have assumed (equation 13) that the perturbed components form 10% of the signal energy. Hence, $max(E_\delta)$ forms 1.25% of the energy of entire signal.