Technical Review

Explaining and Harnessing Adversarial Examples

Name: Yalavarthi Sudesh (2148192)

Course: CSCM38(Advanced Topics: Artificial Intelligence and Cyber Security)

Introduction:

Reason for problem:

Deep Neural networks and machine learning models are being used everywhere in our daily life and a few of them are healthcare and weather predictions. These networks are so advanced that they can also achieve zero error rate or 100 % accuracy in some cases but the performance of these highly advanced neural networks which are considered as breakthroughs are so fragile and vulnerable to inputs with noise and their performance degrades drastically when the input image has noise. To fool these networks attackers started crafting inputs with noise or perturbations and these images or inputs are called adversarial examples.

Adversarial examples:

Adversarial examples are specially crafted or created inputs with the intention of confusing the network, when these inputs are submitted to a neural network would confuse the network and would yield a wrong classification as output. These inputs should look similar to non-crafted inputs and the differences between the original and crafted should be non-identifiable to the human eye. There are several such attacks and one of such attack types is FGSM (Fast gradient sign method) attack.

About FGSM:

FGSM is a white-box attack and the aim of the attack is to produce images with perturbations that ensure the misclassification of the label. A white box attack is an attack in which the attacker has complete access to the model. These images can be used to retrain the networks and training on adversarial examples has proven to be one of the ways to regularize the model to reduce network fragility.

Related Work:

Intriguing properties of neural networks (Szegedy, 2014)**:**

One of the first papers that discussed the creation and effect of adversarial examples on deep neural networks in 2014. The adversarial examples are generated using L-BFGS Method. In this method $x \in R^m$ and $f(x) = l$, where l is the label. The model tries to find minimizer r that is arbitrarily chosen by D(x, l), and x + r is the closest image to x, where f(x) is classified as a label.

D (X, l) is a hard problem so the model tries to approximate it by using box-constrained L-BFGS. The model performs a line search to find minimum c(constant), where $c>0$ for which $f(x+r) = l$ and $x+r \in [0, 1]^m$.

$$D(X, l) = \text{Minimize } c|r| + \text{loss}_f(x + r, l) \quad \text{where } (x + r) \in [0, 1]^m.$$

$loss_f$ (x + r, l) calculates loss between image and label and D(X, l) is the perturbation or penalty that will be added to images. The adversarial examples created using this attack can be generalised and can be used against other datasets and models.

The main issue with the above process is that using L-BFGS in finding the optimal value is computationally expensive as the model uses linear search; it is also time-consuming and impractical to implement. To solve this FGSM uses sign of gradients to create the perturbations.

Method and Experiment:

Method:

The FGSM uses gradients of loss for creating adversarial examples. For every input image, the proposed method calculates the gradients of loss with respect to the input label and creates a new image such that the new image has a maximum loss. This newly created image is called an adversarial example.

$$Adv\_x = x + \eta.$$

$$\eta = \varepsilon * sign\ (\nabla_x J(\theta, x, y)).$$

Let θ be the parameters of a model, x is the input image, y is the original label of the input image x, η is the perturbations added to the image and Adv_x is the final adversarial image.

η(perturbations) is calculated using signed gradient loss and J (θ, x, y) refers to the loss or the cost to train the network, $\nabla_x J$ (θ, x, y) is the gradients of the loss, whereas sign ($\nabla_x J$ (θ, x, y)) represents signed gradients of loss and finally ε is the multiplier to ensure that size of perturbation is small.

Experiments:

Goal:

Our only goal is to fool an already trained model and to achieve our goal our objective is to create an adversarial image that maximises the loss. To achieve this, we calculate the loss using CategoricalCrossentropy. In which the loss is calculated between labels and predictions. The gradient is calculated using loss calculated with respect to the input image. The generated gradient is passed through sign or signum function (The sign function returns -1 if x < 0, 0 if x==0, 1 if x > 0). As the output of the sign function is in the range of -1 to 1, the generated output is multiplied by 0.5 and 0.5 to change the range to 0 to 1. This output is perturbation and that will be multiplied by ε to decrease the size of perturbations and added to the original image.

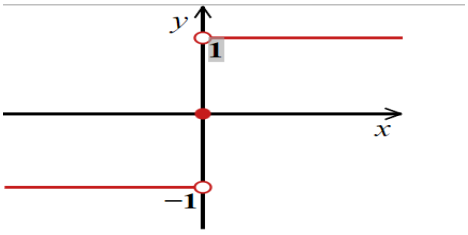

Fig1, Sign function (Wikipedia)

Experiment setup:

For experimentation, we have used ResNet152V2 (He, 2016), MobileNetV2 (Howard, 2017) and DenseNet121 (Huang, 2017) models and the ImageNet (Deng, 2009)dataset. We selected 4 ε values from 0 to 0.15 and these values are [0, 0.01, 0.1, 0.15] for creating adversarial image. We have used multiple images and labels from the ImageNet data set and for simplification purposes, we have considered few hundred sample images of Tiger, Apple, and Sealion.

Results:

For ResNet152V2, ε of 0.01 has yielded 82% misclassification with a average confidence of 74% confidence and ε of 0.1 has yielded 100% misclassification with a minimum of 50.3% confidence. Similarly, ε of 0.15 has yielded 100% misclassification with a minimum of 36% confidence.

For MobileNetV2, ε of 0.01 has yielded 82% misclassification with a minimum of 34% confidence and ε of 0.1 has yielded 100% misclassification with a minimum of 31% confidence. Similarly, ε of 0.15 has yielded 100% misclassification with a minimum of 21% confidence.

For DenseNet121, ε of 0.01 has yielded 76% misclassification with a minimum of 20% confidence, average of 53.64% and for ε of 0.1 has yielded 100% misclassification with a minimum of 10% confidence and average of 48%. Similarly, ε of 0.15 has yielded 100% misclassification with a minimum of 4% confidence and average of 47%.

Image 1(Tiger)	Epsilon 0	Epsilon 0.01	Epsilon 0.1	Epsilon 0.15
DenseNet	Tiger (77.63)	Jaguar (89.9)	snow leopard (16.30)	snow leopard (27.96)
RestNet	Tiger (99.41)	Tigercat (82.28)	African chameleon (42.09)	hyena (80.68)
MobileNet	Tiger (70.01)	Ibex (62.99)	African chameleon (22.13)	African chameleon (32.53)
Image 2(Sea Lion)				
DenseNet	sea lion (83.17)	fox squirrel (25.86)	chimpanzee (30.77)	hermit crab (24.10)
RestNet	sea lion (99.07)	sea lion (99.35)	chimpanzee (41.20)	chimpanzee (76.51)
MobileNet	sea lion (75.82)	Indian elephant (12.58)	macaque (9.18)	macaque (7.55)

Fig 2(Classification changes with respect to changes in epsilon from 0 to 0.15 and confidence levels in brackets)

The set of images on Fig 3 represent the set of three adversarial images effects on confidence and labels of image from ε 0,0.1,0.15 for ResNet152V2, MobileNetV2 and DenseNet121 respectively using a subsample of 3 images.

The three images on Fig 4 represent the variation of confidence for a set of tiger images over DenseNet121 model where ε is 0.01, 0.1, 0.15 respectively and created using a subsample of 50 images.

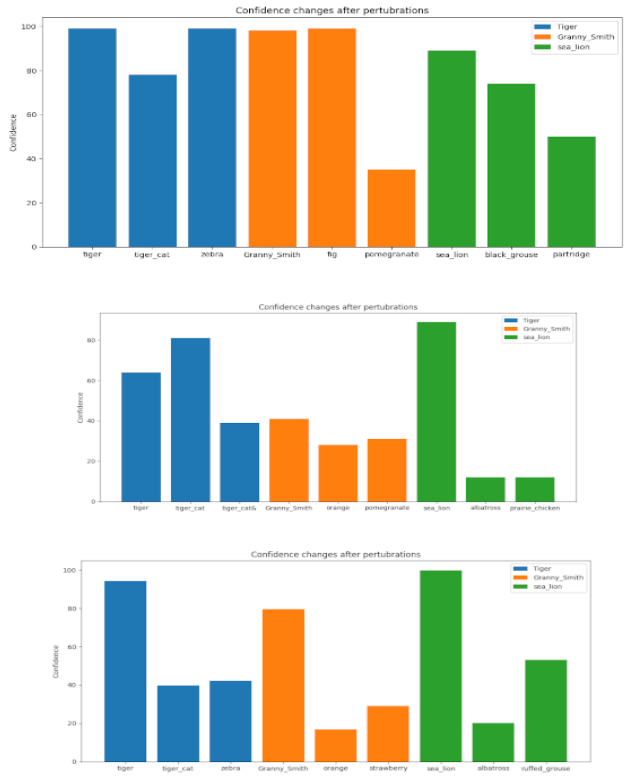

Fig 3 (Confidence and label changes for sample images)

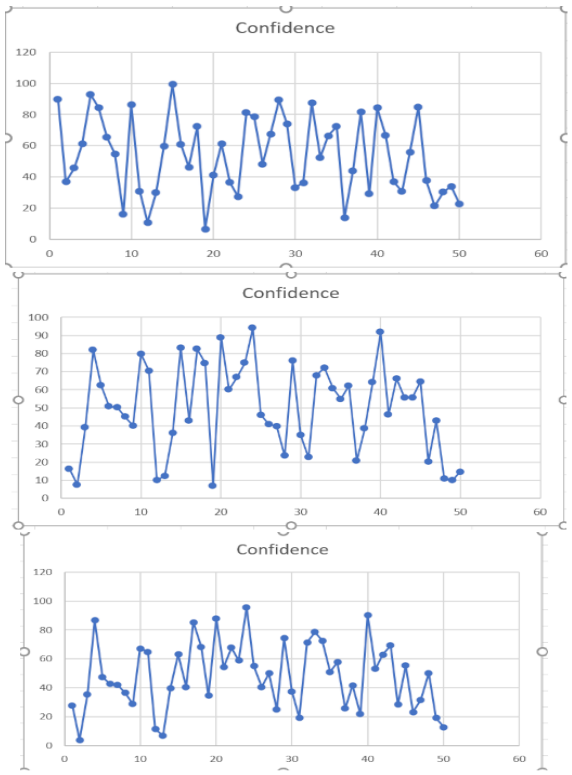

Fig4 (Graphs for sample of 50 images for changes in epsilon)

Based on the datasets that we examined for FGSM, Densenet appears to be performing better or better at handling adversarial images where ε is 0.01 with 72% misclassification and both Resnet and imagenet have more than 80% misclassification. On the other hand, on rare occasions, we have found a few examples where the increase of perturbations led to an increase of the accuracy of the original label.

Existence of adversarial examples:

The FGSM helps us in the generation of misclassified examples in a faster and more efficient way concludes that the reason for the existence of adversarial examples was assumed might be a result of linearity.

In (Goodfellow, 2014), They have observed that models with low capacity also exhibit this behaviour but to a smaller extent. One such example is shallow RBF networks; they are able to predict the positive class with higher confidence or they predict the absence of class or they produce low confidence predictions. This means that shallow RBF networks are immune to adversarial examples,

when they are fooled, they produce low confidence predictions and its average of confidence on adversarial examples is 1.2% and this when compared to the average prediction of clean examples which was around 60%

The FGSM can be used to produce adversarial examples that can be used for the adversarial training of neural networks.

Adversarial training to overcome Adversarial examples:

(Szegedy, 2014) has discussed that by training the neural networks with a mixture of adversarial and normal examples, we can regularize the model to a certain extent. In normal data augmentation schemes, we try to augment the data in such a way that augmented or modified images also occur normally. Instead, we try to create and use augmented images to expose the flaws in the model rather than checking if they occur naturally or not, they have also observed that adversarial perturbations produce better regularization results when applied to hidden layers. In (Szegedy, 2014) this procedure has not produced a greater increase of model accuracy or confidence, this was also due to creation of adversarial examples is difficult in (Szegedy, 2014) as they used L-BFGS to produce them and it turned out to be expensive. To overcome this (Goodfellow, 2014) created an adversarial objective function is used for training based on FGSM (fast gradient sign method).

$$J\tilde{}\,(\theta, x, y) = \alpha\, J(\theta, x, y) + (1 - \alpha)\, J(\theta, x + \varepsilon\, \text{sign}\,(\nabla_x J(\theta, x, y)).$$

In (Goodfellow, 2014), they have observed that α of 0.5 has yielded the results they needed and after retraining the a maxout network, they were able to reduce the error rate from 0.94% without adversarial training to 0.84% after adversarial training.

Improvements to FGSM:

Although, most of the images were misclassified by Neural Networks after adding perturbations and the confidence of the predicted labels started decreasing for increasing ε values, which can be observed in the images of Fig4. This was due to the fact that each pixel perturbation generated is individual and not associated with the surrounding group. (Kurakin, 2016) has tried to solve this issue using a new approach called BIM (Basic Iterative Method). They tried to clip pixel values of intermediate results after each iterative step to limit they are in a ε neighbourhood of the input image.

$$X_0^{adv} = X; \quad X_{N+1}^{adv} = \text{Clip}_{X,\varepsilon}\{ X_N^{adv} + \alpha\, \text{sign}(\nabla_x J(X_N^{adv}, y_{\textbf{true}}))\}$$

For experiments they chose α as 1 , number of iterations is (ε+4, 1.25 ε) and this number is chosen heuristically and is enough for the adversarial example to reach ε max-norm ball's edge with enough limitations to cap the computational cost. They have also provided one more improvement called ILLM (iterative least-likely class method) (Kurakin, 2016). This experiment produced better results when compared to FGSM.

Adversarial method	Photos				Source images			
	Clean images		Adv. images		Clean images		Adv. images	
	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5
fast $\epsilon = 16$	79.8%	91.9%	36.4%	67.7%	85.3%	94.1%	36.3%	58.8%
fast $\epsilon = 8$	70.6%	93.1%	49.0%	73.5%	77.5%	97.1%	30.4%	57.8%
fast $\epsilon = 4$	72.5%	90.2%	52.9%	79.4%	77.5%	94.1%	33.3%	51.0%
fast $\epsilon = 2$	65.7%	85.9%	54.5%	78.8%	71.6%	93.1%	35.3%	53.9%
iter. basic $\epsilon = 16$	72.9%	89.6%	49.0%	75.0%	81.4%	95.1%	28.4%	31.4%
iter. basic $\epsilon = 8$	72.5%	93.1%	51.0%	87.3%	73.5%	93.1%	26.5%	31.4%
iter. basic $\epsilon = 4$	63.7%	87.3%	48.0%	80.4%	74.5%	92.2%	12.7%	24.5%
iter. basic $\epsilon = 2$	70.7%	87.9%	62.6%	86.9%	74.5%	96.1%	28.4%	41.2%
l.l. class $\epsilon = 16$	71.1%	90.0%	60.0%	83.3%	79.4%	96.1%	1.0%	1.0%
l.l. class $\epsilon = 8$	76.5%	94.1%	69.6%	92.2%	78.4%	98.0%	0.0%	6.9%
l.l. class $\epsilon = 4$	76.8%	86.9%	75.8%	85.9%	80.4%	90.2%	9.8%	24.5%
l.l. class $\epsilon = 2$	71.6%	87.3%	68.6%	89.2%	75.5%	92.2%	20.6%	44.1%

Fig 5 (Kurakin, 2016)

Conclusion:

In this report, we have explored the possibility of the creation of adversarial examples for neural networks using FGSM, which can be used to fool the networks. We have observed that adversarial examples were observed due to the reason of models being too linear. The direction of perturbation is an important factor and it leads to the creation of generalised adversarial examples. We have observed changes to model confidence and label prediction with respect to the changes in epsilon the multiplier of perturbations. We have discussed the reason behind shallow RBF networks being resilient to adversarial examples. We have also discussed areas in which FGSM is lacking and how methods like BIM and iterative least-likely class method can be used to make more efficient adversarial examples.

References:

Deng, J. D.-J.-F. (2009). Imagenet: A large-scale hierarchical image database. *IEEE conference on computer vision and pattern recognition*, 248–255.

Goodfellow, I. J. (2014). *Explaining and Harnessing Adversarial Examples.* arXiv.

He, K. a. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770-778). IEEE .

Howard, A. G. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv*.

Huang, G. a. (2017). Densely Connected Convolutional Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Kurakin, A. a. (2016). Adversarial examples in the physical world. *arXiv*.

Szegedy, C. a. (2014). *Intriguing properties of neural networks.* arXiv.

Wikipedia. (n.d.). Retrieved from Wikipedia: https://upload.wikimedia.org/wikipedia/commons/4/4f/Signum_function.svg