Try to Fool AI: Create Artificial Images of Undamaged Cars to be Classified as Damaged as Cars

Joel Kiran Kumar
School of Computing
Clemson University
joelkik@clemson.edu

Abstract

AI has gained a lot of significance in today's world as a result it as replaced most of the human workforce, hence in this paper we wanted to discuss how a CNN model used for classifying damaged and undamaged cars by an insurance company can be fooled by perturbating the input image, so that it classifies undamaged cars as damaged cars, we mainly focus on three adversarial noise in this paper they are namely fast gradient method, deep fool and spatial transformation, we then compare how the models accuracy is affected when these attacks are performed.

1 Introduction

With the tremendous advancement in the growth of Artificial Intelligence, most of the companies have started slowly moving into Artificial Intelligence and have replaced some of the human work force, few years from now Artificial Intelligence would replace the complete human work force across all sectors, the challenge that comes next is are they accurate in what ever they do, can we trust them, yes we could trust them as there accuracy is continuously increasing as we feed more data and they get to learn from the errors, but sometimes they are prone to errors, but still the error rate is very low, hence its acceptable, So most of the companies now have used ANN in production systems for classification and inference, but the challenge here is they can classify objects correctly if the given image for classification was seen by them during training, what if we distort the image by adding some noise, will they still be able to classify correctly, most of the times the ANN are prone to such adversarial attack, in which case they try to misclassify objects, in this paper we are going to exactly talk about the same and the use case here is we want to build a ANN model which will be used by the insurance company to classify damaged and undamaged cars, we want to cast some attack on it so that we can classify undamaged cars as damaged cars, in such case the insurance company is going to lose a lot of money, we exactly wanted to replicate that scenario in this project, firstly we build a model to classify damaged cars and undamaged cars, later we add some adversarial noise to the images and try to see how the model behaves, we try to use three different type of adversarial noise to generate the images and feed it to the model and later compare the accuracy and effectiveness of the model against such an attack. We will be extensively using the [2]ART library provided by IBM for generating the image with adversarial noise. We will be using tensorflow 2.0 for building and testing our models.

2 Related Work

There as been a ton of research activities in this particular field of Trying to Fool AI, researchers from Stanford, MIT, UC Berkely, IBM, Google Brain and various other research laboratory are focusing on generating new type of noise that can be added to the image to fool the neural network, there are occasions where even the strongest model was fooled by adding adversarial noise to the image, few of the examples are as follows, they saw how by sticking some stickers on the traffic signs board

caused the autonomous driving vehicle to be fooled, a 3D image of a turtle with perturbation was classified as rifle by the image recognition model, a pandas image with adversarial noise caused the image recognition model to classify as gibbon, there was also an occasion where the financial doc had bar code which was perturbated to cause the ANN to be fooled, these type of attacks are exploited by malicious users to gain control of the systems, however research is being ongoing to mitigate such attacks, but this is a field which keeps evolving hence we cannot completely eliminate such attacks, as the researchers are trying to mitigate the attacks, there is always a new type of attack that is introduced which causes the system to be fooled again, [1] this paper talks about their approach to mitigate the attacks by building a model with adversarial noise, but its still prone to new attack if it arises, hence there is no perfect ANN model that can be built so that its resilient against all the attacks, the researcher at google brain said that only if the AI can learn by itself looking at the real world and tuning its own parameters for correction, there is a potential for the model to be robust against such attacks, which is a next decade futuristic AI, in this paper we will try to build a model that can classify damaged and undamaged cars, then we try to cast an attack on it by adding some adversarial noise to the image to classify the undamaged cars as damaged cars.

3 Approach or Method

Before we start let me define few key terms that would be essential to understand the model and methodology.
CNN - In deep learning cnn is a type of neural network which is mainly used to analyse visual imagery.
Max Pooling - a pooling operation that selects the maximum element from the region of feature map covered by the filter.
Batch Normalization - is a method used to make artificial neural networks faster and more stable through normalization of the layers' inputs by re-centering and re-scaling.
Flatten layer - Flattening is converting the data into a 1-dimensional array for inputting it to the next layer.
Dropout Layer - Dropout is commonly used to regularize deep neural networks, its done to avoid overfitting , it works by randomly selecting few neurons and nullifying them so that they don't contribute to either the feed forward and backward part of the model training and inference.
Activation Function - a function that is added into an artificial neural network in order to help the network learn complex patterns in the data.
Optimizer - are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses. Optimizer are used to solve optimization problems by minimizing the loss function.

After having introduced some of the key definitions above now lets dive into the details of the different layers of the model, the first layer of the model is a 2 dimensional CNN which takes in a input image of dimension 128 x 128 x 3 and contains 32 filters, its followed by a max pooling layer and followed by a batch normalization layer, following which we again have a 2 dimensional CNN layer which as 32 filters and relu as the activation function, its again followed by a max pooling layer, followed by batch normalization layer again, following this the next layer would be a flatten layer, followed by a dense layer with 128 neurons and the activation function being relu again, its gain followed by a batch normalization, followed by a dropout layer and finally we feed to the last layer which consists of the output classes in our case it will be 2, damaged and undamaged cars, this layer uses softmax as the activation function. The discussed model is represented in a pictorial representation in figure1.

I have used adam as the optimizer with a learning rate of 0.001 and the loss function is sparse categorical cross entropy and I have used 32 as the batch size since using a smaller batch improves in convergence faster compared to large batch size, and also the learning rate is choosen with respect to batch size, so that it doesn't take too much time to learn, so the smaller the batch size the larger the learning rate and vice-versa, sparse categorical cross entropy is used as a loss function because we have more than one class, I mostly used relu as activation function in all layers except the last one, because relu helps in training the network faster, we have used softmax in our last layer, because we want to get the probabilistic values for the output classes for the given input, the output class with highest probability will be the most probable class for the given input.
I did experiment with a lot of different layers by varying the number of CNN and dense layers,

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 32)      128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 64, 64, 32)        128
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 32)        1056
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
dense (Dense)                (None, 128)               4194432
_________________________________________________________________
batch_normalization_2 (Batch (None, 128)               512
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258
=================================================================
Total params: 4,196,642
Trainable params: 4,196,258
Non-trainable params: 384
_________________________________________________________________
```

Figure 1: CNN Model for classifying damaged and undamaged cars

currently we have 2 CNN layers adding an additional CNN layers did not improve the performance, i also did experiment adding another dense layer the performance still remains the same, i couldn't see a lot of difference, hence i felt there was no need in adding additional layers to the network, what ever layers i currently have seem to be the most optimal.
I did use three different adversial noise they are namely FastGradientMethod, DeepFool and Spatial-Transformation. In order to see how the model behaves, the adversial noises are added to the image and fed to the neural networks for classification, we try to predict the accuracy of the model after attack, in this fashion we know how resilient the model is to such attacks. All these noise functions are provided by IBMs ART library, so we just need to import and initialise the appropriate noise and later call the generate function to generate images with the appropriate noise.

4 Experimental Results

My dataset consists of approximately 2500 images of damaged and undamaged cars, with 1250 of damaged and 1250 of undamaged cars, this data was scrapped from [6]kaggle and multiple other sources, the image was preprocessed to a 256 x 256 x 3 dimension, since the scrapped images were of different dimensions, i had to reshape them to 256 x 256 x 3 to be uniform. This data was later used for training the model. As part of preprocessing, the data was read by the script and each pixel of the image was rescaled to 0 to 255 range, basically we normalized each pixel to that range in order to remain consistent, following which we defined two classes 0 means undamaged and 1 means damaged and created a label array. we used this configuration to feed the CNN model. I trained the model for 100 epoch and achieved a training accuracy of 99.95 percent and a validation accuracy of 73.5 percent.

I did experiment with three adversial noises on the test and validation set they are namely FastGradientMethod, DeepFool and SpatialTransformation, before proceeding futher let me define the three attacks.
FastGradientMethod - The fast gradient method works by using the gradients of the neural network to create an adversarial example. For an input image, the method uses the gradients of the loss with respect to the input image to create a new image that maximises the loss.

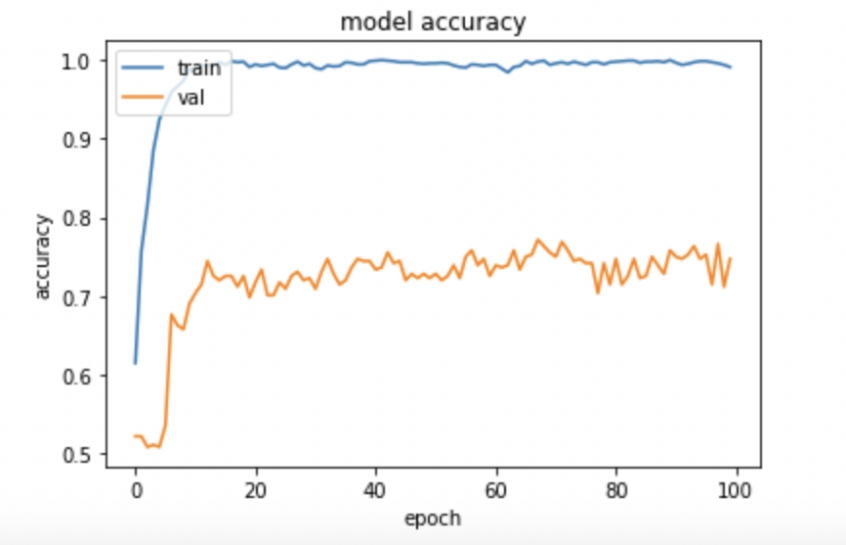

Figure 2: A Curve which shows the trend in models accuracy with respect to epochs on the validation and training set

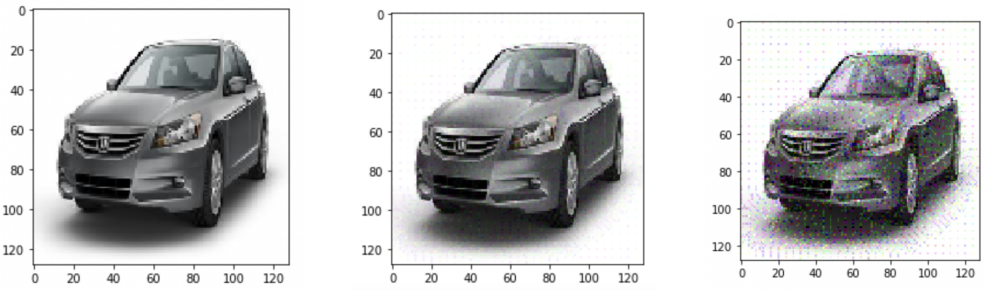

Figure 3: FastGradientMethod - The first image is a undamaged car without any perturbation, second image is perturbed with Fast Gradient having EPS of 0.05 and the third image is perturbed with Fast Gradient having EPS of 0.2

DeepFool - Its an approach to efficiently pertubate the image to a small extent, which can lead the model to become unstable and thus misclassify.
SpatialTransformation - It involves the process of spatial translation and rotation of the image across any dimension, we can control the maximum translation and rotation caused to the image.

After applying adversial FastGradient noise on the test and validation set images, the accuracy was 23 percent, we can control the amount of noise to be added by changing the value of eps parameter in the FastGradient method, the higher the eps value the more noise is added to the image, which leads to higher misclassification rate. Figure3 shows the image after applying FastGradient noise to an undamaged car, we can see that as we increase the eps the noise to the image is higher as result the model is prone to higher missclassification rate. At eps 0.05 the accuracy of the model on the validation set with undamaged car was 24 percent and for eps 0.2 the accuracy of the model on the validation set with undamaged car was 22 percent.

Next lest discuss the DeepFool Attack, after applying deep fool attack the accuracy of the model is 46 percent, I did try for eps 0.05 and eps 0.2 the accuracy was 48 percent and 46 percent respectively for undamaged cars, we can see that as we increase eps the accuracy of the model starts to decrease, which seems the model is getting fooled. The Figure4 shows the images after applying deep fool attack on undamaged car.

Next lets discuss the Spatial Transformation attack, we applied this attack on the validation data which contains undamaged cars with translation 1 and 10 and the corresponding accuracy was 64

Figure 4: DeepFool - The first image is a undamaged car without any perturbation, second image is perturbed with DeepFool having EPS of 0.05 and the third image is perturbed with DeepFool having EPS of 0.2

Figure 5: Spatial Transformation - The first image is a undamaged car without any perturbation, second image is generated after applying spatial transformation with translation 1 and third image is generated after applying spatial transformation with translation 10

percent and 12 percent respectively, we can see that as we increase the translation the accuracy of the model drops. Figure5 provides an example of applying spatial transformation to an image of undamaged car with max translation 1 and 10.

From the above experiments we can conclude that FastGradient is a stronger attack compared to DeepFool. We can't compare spatial translation with the previous two because the way it perturbates the image is different than the previous two.

5 Conclusion

In conclusion i feel the model had reasonable accuracy on the validation and test set, the accuracy was varying somewhere between 70-75 percent which i feel was good, we also did see how the addition of adversarial noise to the image affected the accuracy of the model, we did compare the accuracy of the model after applying FastGradient, DeepFool and the spatial transformation attack. We also saw how we can control the amount of noise added to the image by tuning the eps parameter and its impact on the accuracy of the model.
We did achieve our problem statement that is too fool AI by adding adversarial noise to the image so that it classifies undamaged cars as damaged cars, but in order to protect against such attacks i feel we need to generate such noise and include them as part of the training sample, so that the model can learn these noise, such that the error rate would be minimized when the actual attack happens, this could seem like a future work, where i would be interested in collecting a lot more data and generating new data by spatial transformation of the data along with generating data by adding different type of noise, and later training the neural network against it, so that it can be resilient against such attacks, which in our case would minimize the losses to the insurance company. We can't completely eliminate adversarial attack, but can definitely minimize it.
I liked the project as whole, as i got a chance to explore more and understand better about the CNN, the loss function, optimizer, batch sizes and learning rate, it also helped me to explore and understand

the ART Library provided by IBM and its functionalities. As a concluding note it was great working on the project as i got to learn and explore a lot of interesting topics.

References

[1] https://www.osti.gov/servlets/purl/1569514

[2] https://adversarial-robustness-toolbox.readthedocs.io/en/latest/modules/attacks/evasion.html

[3] https://arxiv.org/pdf/1412.6572.pdf

[4] https://arxiv.org/pdf/1712.02779.pdf

[5] https://arxiv.org/pdf/1511.04599.pdf

[6] https://www.kaggle.com/anujms/car-damage-detection

[7] https://www.stackoverflow.com

[8] https://www.towardsdatascience.com

[9] https://www.tensorflow.org/tutorials/quickstart/beginner

[10] https://www.tensorflow.org/guide/effective$_t f2$