Robustness of CheXNet Against Adversarial Attacks

Chris Kull (chkull2), Rahul Sai Rajapalayam Sivakumar (rsr4), and Venkatesh Dharmala (vad4)
UIUC CS 543: Computer Vision, Fall 2022

1. Introduction

1.1 Definition and motivation of the problem

Over the past decade, deep neural networks (DNNs) have taken the field of Computer Vision by storm. Performance in image classification tasks has improved greatly, and the promise of such models enabled them to begin seeing use in a variety of contexts. One particularly interesting application of DNNs is in the area of Radiology, where they can be used as a tool by practicing radiologists to help aid in their diagnosis of patients. However DNNs have been found to be susceptible to adversarial attacks (Kurakin et al., 2016b), which involves the use of specifically engineered inputs that are designed to trick the model into predicting the wrong class or label. This is particularly concerning for Radiology as misdiagnosis can lead to serious health complications, or in some cases even death.

In this paper, we want to analyze one such Radiology-based DNN in the context of adversarial attacks, CheXNet (Rajpurkar et al., 2017). CheXNet is a model that takes in an image of a chest X-ray as input and then outputs the probability of 14 different thoracic diseases being present in the patient. We will carry out this study in a few steps. First, we will construct and train a version of CheXNet that is modeled after the architecture laid out by the original authors, albeit at a smaller scale given our reduced resources. Then, we will apply a variety of white-box and black-box adversarial attacks on our model to perceive how performance is affected and if there are any notable trends. Finally, we will attempt to make modifications to the original model to see if it can be made more resilient to adversarial attacks.

1.2 Related work

Training a model with the performance of CheXNet was only possible with the existence of a large enough X-ray dataset. In 2017, Wang et al. released ChestX-ray14, a dataset containing 112,120 chest X-ray images through the National Institutes of Health. In addition to demonstrating that label classification was possible, Wang et al. additionally showed that it was possible to spatially locate where the thoracic diseases were located in the X-ray. For this paper we are ignoring this spatial location component and instead or simply focusing on classifying the input image as a whole.

There are a few different existing reimplementations of CheXNet that used Python 3 and PyTorch. Via GitHub, Weng et al., 2017, and Tayeb and Willmann, 2019, provide examples for loading ChestX-ray14 as well as the framework for training a DenseNet-based model. These implementations explore additional ideas to improve CheXNet such as data augmentation, which we also explore albeit in the context of improving robustness against adversarial attacks rather than improving raw performance.

As DNNs have become more popular, adversarial attacks on them have become more and more relevant given their growing influence in today's world. Bhambri et al., 2020, provides a survey over the landscape of various black-box adversarial attack methods, as well as developments in defenses against them. Bhambri et al. built up a taxonomy for both attacks and defenses, grouping them into different families. This paper will explore gradient estimation methods from the attack perspective and adversarial training from the defense perspective.

Adversarial attacks and defenses on chest X-ray specific models have also been studied. Rao et al., 2020 specifically analyzed CheXNet, how it performs in light of white-box adversarial attacks, and how it can be augmented with defense methods. They used five attack methods: fast gradient sign method (FGSM), project gradient descent (PGD), momentum iterative gradient-based method (MIFGSM), distributionally adversarial attack (DAA), and diverse input iterative fast gradient sign method (DII-FGSM).

For defenses, they present three approaches in adversarial training, pixel deflection transform (PDT), and a combination of the previous two. They tested using multiple architectures including DenseNet-121, but only with 6 possible output classes rather than 14.

1.3 Summary of approach

Our approach had three main stages. The first stage involved recreating CheXNet to the extent that our resources enabled us to do so. The dataset we used for both training and testing was the ChestX-ray14 dataset (Wang et al., 2017), which was publicly available online through the NIH's website. The development platform we used was Python 3 and the PyTorch machine learning framework. All development, training, and testing was carried out on local hardware, which includes NVIDIA 30-series RTX GPUs. After training is complete, we used our trained model to run inference on a test set in order to get a baseline performance metric.

Once we had a trained model, we were then able to begin experimenting with adversarial attacks. As a means of streamlining the process, we opted to use the Torchattacks package (Kim, 2020). Torchattacks is designed to take in a PyTorch generated model and then run any number of white-box adversarial attack methods to generate malicious inputs that are specific to that model. For our study, we ran five white-box attack methods: FGSM (Goodfellow et al., 2014), basic iterative method (BIM, Kurakin et al., 2016a), PGD (Madry et al., 2018), SparseFool (Modas et al., 2019), and DeepFool (Moosavi-Dezfooli et al., 2016). We used each of these methods to generate a set of adversarial examples, and then measured performance of our initial trained model when it was given the adversarial examples.

With respect to black-box attacks, most of them did not require/were not available in a package, as they could be computed without much mathematical /probabilistic complexity. The ZOO (Chen Et al., 2017), Gray Image Attacks, and Decision Boundary attacks were performed to produce a set of adversarial images.

In an attempt to improve robustness, we retrained the model with the adversarial examples added in. Training process is otherwise identical to the process carried out for the baseline model. We then looked at the performance of this improved model against a held out test set of a combination of the original and adversarial datasets, and then analyzed its performance when compared to the original model.

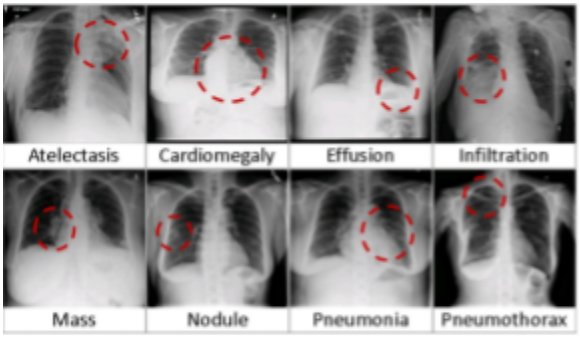

Figure 1. From Wang et al., 2017. Sample chest X-rays and thoracic disease labels in ChestX-ray14.

2. Details of the approach

2.1 Initial model construction and training

As mentioned previously, we are using the ChestX-ray14 dataset for training, which consists of 112,120 X-rays and any diseases associated with them. The labels have 15 different classes, with one of them being the benign class (labeled "No Findings"). See Figure 1 for example inputs present in the ChestX-ray14 dataset. Most images in this dataset have multiple labels associated with them, as it is possible that multiple diseases may be simultaneously present. Because we want to test out the effectiveness of adversarial attacks, we need to consider this as a multi-class classification problem rather than the multi-label classification problem that CheXNet was originally trained for. To account for this, we had to purge the dataset of images which had multiple labels or belonged to the benign class. The final dataset which we used contained 30,963 images from 14 different classes.

For model architecture, DenseNet-121 (Huang et al., 2016) is the backbone. In DenseNet, each layer is connected to all future layers, and inputs are combined via concatenation. As a result, the computation graph in DenseNet has on the order of L^2 connections when there are L layers, as opposed to just L connections. See Figure 2 for more details on the inner layers of DenseNet. CheXNet uses a 121-layer DenseNet in order to carry out classification.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 2. From Huang et al., 2016. The specific layers used in DenseNet, all dense blocks are connected to dense blocks further down. CheXNet and our reimplementation use DenseNet-121 as the backbone.

Torchvision, a computer vision sub-library of PyTorch, has a built-in DenseNet-121 model that we used for convenience. After the final fully-connected linear layer of size 1000, we add another fully-connected linear layer of size 14 followed by a sigmoid in order to produce probabilities for each of the 14 thoracic diseases that are present in the dataset. Rather than initializing all weights randomly, we initialized the DenseNet-121 portion of the model with the weights that were used when it was trained for the ImageNet dataset (Deng et al., 2009) which has 1000 classes. The intention is to leverage transfer learning to speed up training as the model would already have some semantic understanding of images from its experience with ImageNet.

We decided on a 60-40 split between training and test data, or about 18,000 training X-rays and 12,000 test X-rays. Input images undergo some amount of preprocessing before being fed to the model. The images in ChestX-ray14 are of various sizes, so they are first resized to a size of 256x256 pixels. A center crop of 224x224 pixels is taken, as that is the expected input size for DenseNet-121. As a form of data augmentation, training images undergo a horizontal flip with a probability of 0.5. Finally, images are normalized by mean centering and setting the standard deviation to 1, using the standard ImageNet values ([0.485, 0.456, 0.406] for per-channel means, and [0.229, 0.224, 0.225] for per-channel standard deviations).

Training was done with a batch size of 64 over 10 epochs. These hyperparameters were picked based on the hardware that we had available to train–saturating the GPU's memory while also keeping computation time to a reasonable duration. The loss function was binary cross entropy loss, and the optimizer was Adam, where $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Both of these mimic the training details in the original CheXNet (Rajpurkar et al., 2017). Training took approximately 4 hours to complete under these conditions.

After training was complete, we ran the test set through the model in order to calculate a per-class and overall AUROC value. AUROC, or area under the receiver operating characteristic curve, is a metric used to denote a model's ability to discriminate between positive and negative examples of each class at different classification thresholds. This is accomplished by comparing the true positive and false positive rates at various thresholds. An AUROC of 0.5 would be a random classifier, while an AUROC of 1.0 would be a perfect classifier. For reference, the original CheXNet had an AUROC of 0.841, so our initial goal for our reimplementation was to hit an AUROC of 0.8 or greater.

2.2 White-box adversarial attacks

White-box adversarial attacks are a type of attack on deep-learning models where the attacker has full knowledge of the model being attacked, including its architecture and the training data it was trained on. In

these attacks, the attacker crafts adversarial examples – inputs that are specifically designed to mislead the model – and uses them to cause the model to make incorrect predictions. We specifically focus on gradient-based adversarial attacks which are a class of white-box attacks that use the gradient of the model's objective function to generate perturbations in the input data that cause the model to make mistakes. By repeatedly applying these perturbations, the attacker can eventually generate an adversarial example that causes the model to make a mistake even though the input data has not changed significantly.

Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014): FGSM is one of the most simple gradient-based adversarial attacks which involves only a few steps. First, the loss of the model is computed on forward propagation, the gradient is calculated with respect to the image pixels. The pixels are then perturbed in such a way that it maximizes the loss by pushing them in the direction of the calculated gradient. An adversarial image can be generated in the following way via FGSM:

$$adv\_x = x + \epsilon * sign(\nabla_x J(\theta, x, y)) \quad (1)$$

Basic Iterative Method (BIM) (Kurakin et al., 2016a): BIM is an iterative method that improves upon FGSM by making small, gradual adjustments to the input data in order to generate adversarial examples. BIM ensures that the generated adversarial examples remain within a certain range of the original input data, making them less noticeable to the human eye. Compared to FGSM, BIM is a more subtle method for generating adversarial examples. An adversarial image can be generated in the following way via BIM:

$$adv\_x = clip(x + \epsilon * sign(\nabla_x J(\theta, x, y))) \quad (2)$$

Projected Gradient Descent (PGD) (Madry et al., 2018): PGD is a more generalized version of the BIM. The only difference is that PGD works by projecting the sample from each iteration into the epsilon neighborhood of the benign samples, hence lowering the perturbation size which results in a more similar image to the original but still retaining the level of perturbation achieved. An adversarial image can be generated in the following way via PGD:

$$adv\_x = proj(x + \epsilon * sign(\nabla_x J(\theta, x, y))) \quad (3)$$

SparseFool (Modas et al., 2019): SparseFool is a geometry-inspired white-box attack which is built for scaling to high dimensional data and reducing the computation time. It depends on modifying a lower amount of pixels in a larger space (i.e. larger image), thereby leveraging the sparsity of the perturbations rather than the number of perturbations. This also results in an improvement in terms of perception of the image, thereby it's easier to fool the human eye.

DeepFool (Moosavi-Dezfooli et al., 2016): DeepFool is similar to the previous technique in terms of generating minimally perturbed images which look similar to the original image. It linearizes the classifier around the point of interest i.e the image then looks for the closest decision boundary, and perturbs it accordingly. It is advertised as being faster than FGSM and its derivatives and was mainly made to compare the adversarial robustness of classifiers, hence why we are interested in using it.

2.3 Black-box adversarial attacks

Black-box adversarial attacks usually have no existing knowledge of the model or its parameters, and hence are less targeted in the approach of their attacks. We use the following black box attacks to perform adversarial samples:

Zeroth Order Optimization (Chen et al., 2017): This method estimates the gradient of the original deep network using the symmetric quotient difference to make estimations. The symmetric quotient difference equation is as follows:

$$g_{est} : \approx \frac{f(x + he_i) - f(x - he_i)}{2h}) \quad (4)$$

Gray Image Attack: This approach utilizes gray images, in a trial-by-error manner. Starting off with a gray image, each pixel in the image is either converted to 0 or 1 then added to the image.Several combinations are tested until the model fails. A fitness function is used to base the attacks on the probability that an image gets misclassified. It is defined by:

$$J(\theta) = P(y'|\theta) \approx [F(\theta)]y' \quad (5)$$

Decision Boundary Attack: This approach takes a random distribution of pixels of an image, and explores the decision boundary of its class. We then

run a minimization algorithm to minimize the adversarial images distance (SSD used in this case) with the original image. We had used the Query Efficient Boundary Attack (QEBA) version which optimizes the function:

$$g(\theta) = arg\,min_{v>0}(f(x + v\frac{\theta}{|\theta|}) \neq y) \quad (6)$$

2.4 Improving model robustness

Our approach to improving the robustness of the original model was augmenting the original dataset with adversarial images generated from our various attack methods. We added an additional 18,582 adversarial X-ray images to bring the overall size of the dataset up to 49,545 total images. Testing was carried out by generating 2,000 additional adversarial examples for each attack method, and calculating the AUROC for the original and retrained model. We did not modify the DenseNet architecture, or tried to group it with an auxiliary classification algorithm, mainly because we wanted to observe the changes that the deep-learning model learns by training directly on the adversarial examples alone. That being said, we are sure that with proper hyperparameter tuning, we can still improve our model. Adding some auxiliary models to assist with any known patterns exhibited by adversarial examples can also help us improve our accuracy.

3. Results

3.1 Initial model performance

The performance of our reimplementation of CheXNet was measured based on its AUROC scores on a held out test set, as previously described. This test set was 40% or approximately 12,000 of the extracted, single label X-rays from ChestX-ray14, where each label had 1 of 14 thoracic diseases associated with it. Table 1 shows the performance of our model in comparison to the original CheXNet (Rajpurkar et al., 2017).

Though the performance numbers are comparable, recall that the tasks that the two models are performing are different. The original CheXNet is faced with a multi-label classification problem, whereas our implementation is dealing with a multi-class classification problem. As a result, our model can make additional assumptions based on the fact that only one disease can be present. Additionally, the original CheXNet is both trained and tested on a dataset that not only has nearly three times as many X-rays, but also has an additional "No Findings" benign class that our implementation does not need to consider. The original CheXNet model is considerably more advanced and robust than our simplified model, and as such the two should not be considered equivalent based on the presented results. That being said, our model is still able to distinguish various diseases from one another with decent accuracy, and is a suitable candidate for employing adversarial attacks.

Pathology	Rajpurkar et al. (2017)	Ours
Atelectasis	0.8094	0.8445
Cardiomegaly	0.9248	0.9553
Effusion	0.8638	0.8845
Infiltration	0.7345	0.7769
Mass	0.8676	0.8597
Nodule	0.7802	0.8158
Pneumonia	0.7680	0.6880
Pneumothorax	0.8887	0.8771
Consolidation	0.7901	0.7412
Edema	0.8878	0.8966
Emphysema	0.9371	0.8714
Fibrosis	0.8047	0.8257
Pleural Thickening	0.8062	0.7436
Hernia	0.9164	0.9400
Mean	**0.8414**	**0.8372**

Table 1. AUROC performance comparison between the model from Rajpurkar et al. (2017) and our model

3.2 White-box adversarial attacks

From the images presented in Figure 3a, it can be seen that most of the generated adversarial images look similar to the original image with only minor visual differences. In terms of the time taken to generate these examples, only SparseFool consumed the most time (473 seconds) compared to the average of 11 seconds with the other white-box attacks. This can be attributed to the fact that SparseFool is a geometry based attack which works by controlling the sparsity of the perturbations. This is in contrast to what they suggest in the original paper. Deepfool provided the most visually similar image, but still managed to fool the classifier and was labeled wrongly to a different class. These examples were chosen to show the contrast between the labels

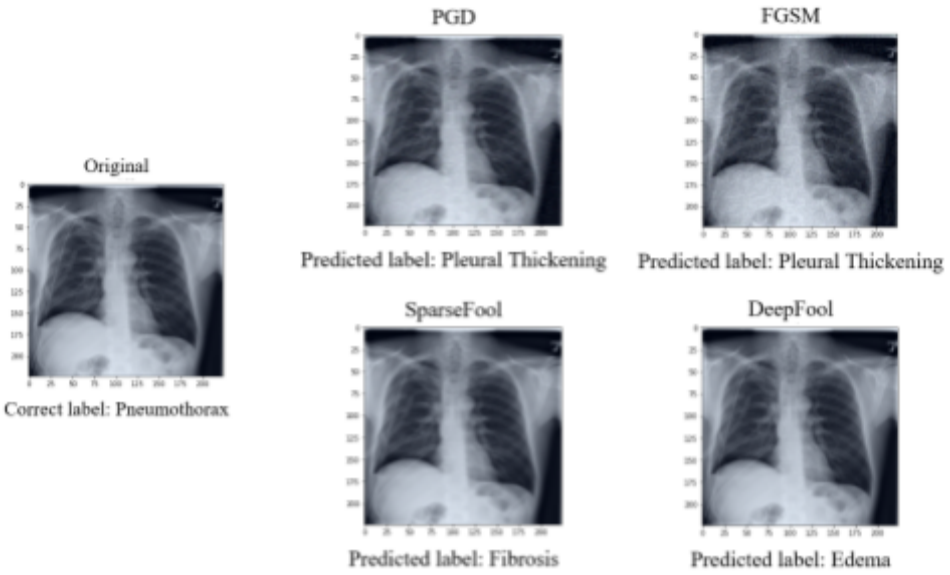

Figure 3a. Examples of adversarial images generated from white-box methods.

obtained from the baseline model on the original image and the corresponding adversarial image. The actual AUROC scores when our model is faced with these examples are discussed in a later section.

3.3 Black-box adversarial attacks

As shown in Figure 3b, there were various types of black-box attacks tested on the CheXNet Model.

Zeroth Order Optimization (ZOO): This model performed fairly quickly, producing adversarial images in ~2.7 seconds each. One issue with this approach is that certain cases present a lot of noise within the image.

Gray Image Attack: This method was more time consuming, taking ~20.3 seconds but still producing a considerable amount of noise.

Decision Boundary Attacks: This approach of attacks was by far the most time-taking, although it did produce better results for a select set of sample images. There were also cases where the algorithm did not halt, as minimizing the squared distance of the images to the target value led it to cross the decision boundary again. When a target value is not set, the likeliness of both images is far from optimal.

All these approaches show that somewhat large perturbations are required to attack models, when we have no access to its parameters. Although the adversarial images are not clearly distinguishable from the original, these outputs perform poorly with respect to white box attacks. In a real-life setting,

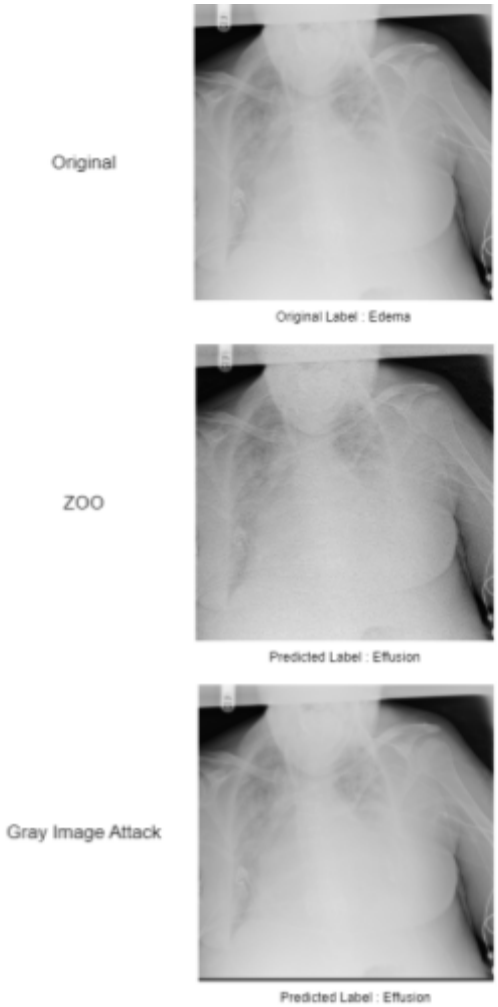

Figure 3b. Examples of adversarial images generated from black-box methods.

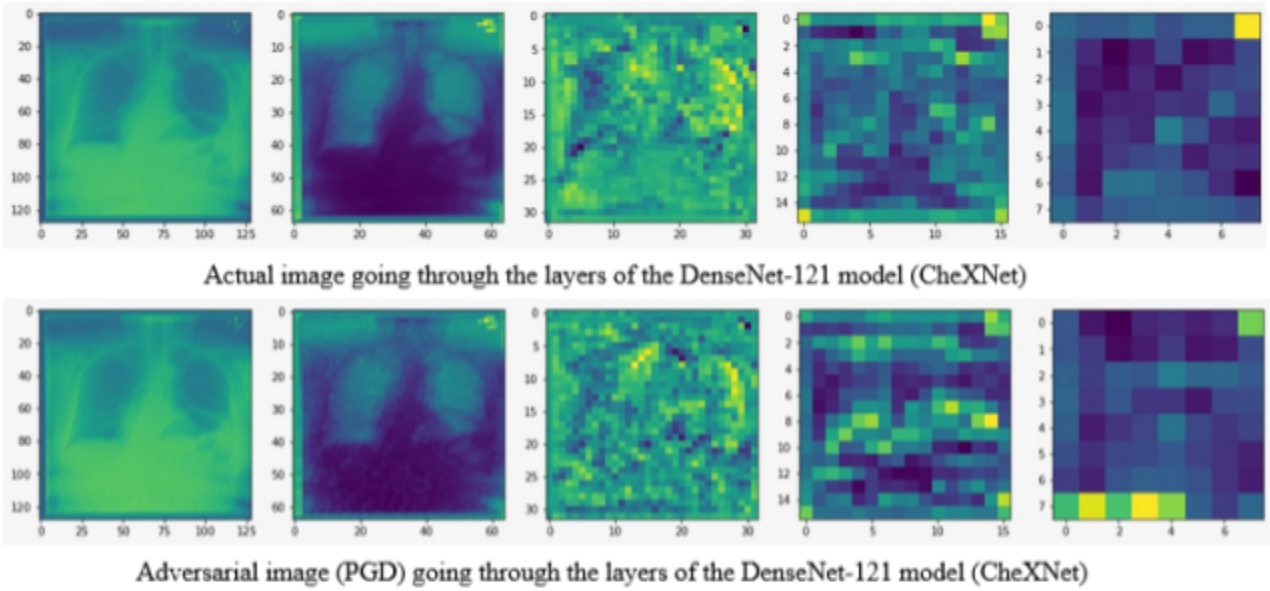

Figure 4. Visualization of an actual image and adversarial image activations throughout the model. From left to right, conv0, dense1, dense2, dense3, dense4.

these attacks could be clearly identified using human intervention.

3.4 Effects of adversarial attacks on our reimplemented CheXNet model

We tried to analyze how adversarial images are processed when they go through the basenet ChexNet model, which is essentially a DenseNet-121 model consisting of 4 dense blocks. We used a forward register hook from PyTorch to fetch the outputs from the intermediate layers. Using our white-box and black-box attacks, we created a test dataset through which we could analyze and compare our baseline model's performance when exposed to the actual image and the corresponding adversarial image.

We chose 5 different layers/blocks of interest which gives us more context on how adversarial images manage to get misclassified by the model. For any regular image based classification task, there is equal emphasis on learning high-level features (edges and shapes) from the earlier layers, and learning low-level features (textures and patterns) from the layers in the later stages. However for medical image classification tasks, especially involving radiological images, more emphasis is made on the later stages as it's important to learn the low-level features in order to make an accurate diagnosis. Moreover, there aren't many high-level features in radiological images as they are mostly grayscale images with little to no distinction between edges/shapes. This can be better explained by a visual example as shown in Figure 4.

The first layer conv0 is a 7x7 convolution layer, which is intended to learn the high level features from the image. As we can see from the first image in both the image strips, there isn't much difference between the outputs from the first layer for the actual image and the adversarial image. This agrees with the discussion of identifying high-level features. However, as we start moving into the first dense block (dense1), we start seeing smaller spots and differences in texture, in the output obtained from the layer. The added perturbations on the adversarial image start affecting the model from this layer, and continue on to the subsequent layers. From the output of the second dense block, we can notice the distinct shape of the lung being formed for the original image indicating that it is starting to learn the low-level features of the image. But on the other hand, the output of the same second dense block for the adversarial image shows the difference. The shape of the lung is completely disfigured on one side. This means that subsequent layers will only propagate the error, and by comparing the outputs of the final two dense blocks, we can see how the model misclassified the adversarial image.

3.5 Robustness improvements on our reimplemented CheXNet model

On a test dataset of 2000 images for each attack (PGD, FGSM, BIM, SparseFool and DeepFool), we tested the baseline CheXNet model against the robust model that we retrained with adversarial examples based on the same DenseNet-121 architecture. We chose to analyze the model with only white-box attacks due to the ease of generating those examples directly. The mean AUROC scores of the models across the 14 different classes are tabulated in table 2.

Attack	Original	Retrained with Adversarial Images
PGD	0.593	0.714
FGSM	0.682	0.761
BIM	0.612	0.794
SparseFool	0.706	0.743
DeepFool	0.609	0.728

Table 2. AUROC performance comparison between original model and retrained robust model

SparseFool provides the best score when there's no defense due to the nature of the image generated. As mentioned previously, it is a geometry based attack which controls the sparsity of the perturbations to provide a very similar image to the original image. SparseFool also showed only a minor improvement on the model with the retrained examples, as the adversarial examples from SparseFool contribute little to no information in the retraining process, which is similar to the findings from their paper. For the other attacks i.e., FGSM, BIM, PGD which belong to a similar class of gradient-based attacks, show a similar improvement in performance when trained with adversarial examples, showing that augmenting the training dataset does improve the robustness of the model against adversarial perturbations.

4. Discussion and Conclusions

4.1 Overall discussion of results

From our initial results on testing adversarial attacks on the baseline ChexNet model, we can say that radiological deep learning models are highly susceptible to adversarial images, due to their dependence on learning the low-level features of the image to provide the diagnosis. However, not all adversarial attacks were equally efficient, as white box attacks affected the accuracy of the models better than black box attacks.

4.2 Possible improvements and additional concerns

Given that we had to simplify the problem into a multi-class classification problem as opposed to a multi-label classification problem, the model we have created likely performs poorly on X-rays that have two or more thoracic diseases present. With more training resources, we could train the model on the entirety of Chest-Xray14 as well as supply a larger number of adversarial examples to create a more robust model overall. Additionally, we explored adversarial attacks and defenses in the context of multi-label classification, rather than multi-class classification. A next step to take would be to employ other attack and defense methods that are more suitable to the multi-label problem, which would enable a fairer comparison with the original CheXNet classifier.

One concern we have is that confounding variables can cause radiological deep learning models to perform poorly when applied to new situations. These variables can interfere with the model's ability to generalize and accurately make predictions on new data (specific to medical images dependent on machinery). This can have adverse effects on both generating the adversarial images to retrain the model, as well as improving the robustness of the model. In our case the dataset contains images obtained from both portable X-ray scanners, and regular X-ray scanners. Portable X-ray scanners are biased (most of them result in the same label) due to the nature of the conditions they are used in i.e., where patients might have medical devices such as a pacemaker.

5. Contributions by individuals

Chris: Handled model construction and training details such as the optimizer, loss function, and other hyperparameter choices. Trained and tested models locally using a GPU. Organized final report format and style.

Rahul: Preprocessing the dataset, enhancing the dataset using adversarial images (white-box), local

training of the models and hyperparameter tuning, analyzing intermediate layers to find differences in adversarial images.

Venkatesh: Studying the different types of adversarial attacks to perform, preparing the dataset through adding adversarial attacks, performing various black box adversarial attack techniques.

6. Conclusion

With the emergence of deep learning in the modern world, it is imperative for us to consider the threat of adversarial attacks. As our results with CheXNet and medical images show, these attacks can be resisted with adequate retraining of models with adversarial examples. More research is needed to scale our methods to a broader domain of image data, along with improving the performance of these attack-resistant models.

References

Bhambri, S., Muku, S., Tulasi, A., and Buduru, A. B. A Survey of Black-Box Adversarial Attacks on Computer Vision Models. *arXiv*, 2020.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models. In *AISEC*, 2017.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv*, 2014.

Huang, Gao, Liu, Zhuang, Weinberger, Kilian Q, and van der Maaten, Laurens. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

Kim, H. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv*, 2016a.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. *arXiv*, 2016b.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *ICLR*, 2018.

Modas, A., Moosavi-Dezfolli, S.-M., and Frossard, P. SparseFool: a few pixels make a big difference. In *CVPR*, 2019.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, 2016.

Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., et al. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv*, 2017.

Rao, C., Cao, J., Zeng, R., Chen, Q., Fu, H., Xu, Y., and Tan, M. A Through Comparison Study of Adversarial Attacks and Defenses for Common Thorax Disease Classification in Chest X-rays. *arXiv*, 2020.

Tayeb, O., and Willmann, T. Radiologist-Level Disease Detection on Chest X-Rays with Deep Learning. *GitHub*, 2019.

Wang, Xiaosong, Peng, Yifan, Lu, Le, Lu, Zhiyong, Bagheri, Mohammadhadi, and Summers, Ronald M. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *arXiv preprint arXiv:1705.02315*, 2017.

Weng, X., Zhuang, N., Tian, J., and Liu, Y. CheXNet for Classification and Localization of Thoracic Diseases. *GitHub*, 2017