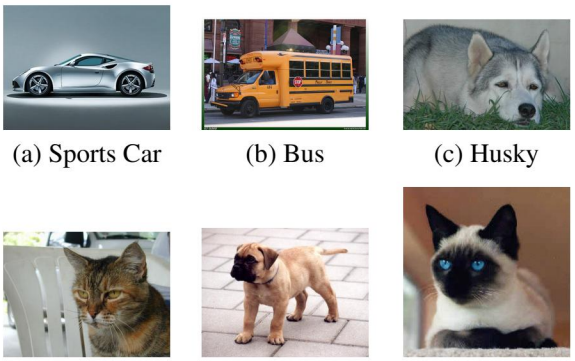
Supervised Adversarial-Attack Classification

Kumar Kallurupalli (kksreddy@stanford.edu) [*1] **Nicholas Tan (ntan2012@stanford.edu)** [*2]
Shalini Keshavamurthy (skmurthy@stanford.edu) [*3]

Abstract

Deep neural networks are very sensitive to manipulated inputs. Adversarial examples slightly modify the source image in such a way that it will be classified incorrectly by a machine learning classifier. We trained an extended VGG19 model on a subset of ImageNet data to study a system that detects when such an attack is occurring. We demonstrate the ability to extend an adversarial-attack classifier trained to detect one form of attack on detecting other types of adversarial attacks.



(a) Sports Car (b) Bus (c) Husky

(d) Tabby Cat (e) Boxer (f) Egyptian Cat

Figure 1. Sample images from the ImageNet dataset.

1. Introduction

As machine learning tasks have gained traction and widespread use in high-impact fields, the topics of security and safety have been raised as a key concern during the design of any deep learning architecture. This sentiment is echoed in particular by the vulnerabilities of deep neural networks to adversarial attacks, which are maliciously manipulated inputs that cause degradation in the original intended performance of the system. Oftentimes, the output arising from such an attack may be designed to produce any chosen outcome, which gives bad actors an opening to control the system in harmful ways. Thus, our goal is to implement a robust and lightweight module to detect and classify such attacks, yet not detract from the intended performance of the application. The input to our model is an image. The model acts as a binary classifier. The output determines if the image is a ”clean” image or an ”adversarial” one.

2. Related Work

The current literature shows many examples and implementations of successfully defending against adversarial attacks. Kurakin et al. (Kurakin et al., 2016) has trained the Inception v3 model on ImageNet on an augmented dataset of both clean and adversarial images to make their model more robust to adversarial attacks. Ian Goodfellow et. al. (Goodfellow et al., 2015) has developed a fast way to generate adversarial example images, and training the application model on these adversarial examples improves the performance of that model in the face of adversarial attacks. Xiaoyong Yuan et. al. (Yuan et al., 2017) studies the landscape of state-of-the-art adversarial attacks in various machine learning fields and proposes various solutions for addressing them, from adversarial detection to retraining to input reconstruction, etc. We have borrowed some of the methods from these papers for our work in adversarial defense and adversarial image generation.

3. Dataset

We used the ImageNet dataset (Deng et al., 2009) for training and classification of clean and adversarial images. To reduce the computational load of training on such a large dataset in the time parameters given, we used a subset of ImageNet to concentrate our efforts on. Figure 1 shows some sample images. We treat the images in this subset as ’clean’ images and label them as such.

Preprocessing images - We normalized the images by subtracting the mean from the RGB images. Since the images of ImageNet are of different sizes, we reshaped them into 256 x 256 resolution.

Attack methods - There are a number of methods for adversarial attacks for images - Gradient based attacks, score based attacks, decision based attacks, etc. For the purpose of generating adversarial images, we have used 2 gradient

based attacks - FGSM and DeepFool and 1 decision based attack - Contrast Reduction.

Fast Gradient Sign Method (FGSM) attack - This is the classical method of creating an adversarial attack. It is a gradient-based white-box attack. Attacks are generated using the equation below -

$$X^{adv} = X + \epsilon sign(\Delta_x J(X, y_{true})) \quad (1)$$

where X is a clean image,
X^{adv} is the generated adversarial image,
y_{true} is the true label of the image,
ϵ is the size of perturbation, specified in terms of pixel values in the range [0,255],
$J(X, y_{true})$ is the cost function used to train the model.

DeepFool attack - This is another white box iterative approach in which the closest direction to the decision boundary is computed in every step to update the image. It is equivalent to minimizing the orthogonal projection of the data point onto the affine hyperplane which is the decision boundary between the various classes. At each iteration i, f is linearized around the current point x_i and the minimal perturbation r_i of the linearized classifier is computed as

$$\arg\min_{r_i} ||r_i||_2 \text{ subject to} f(x_i) + \Delta f(x_i)^T r_i = 0 (2)$$

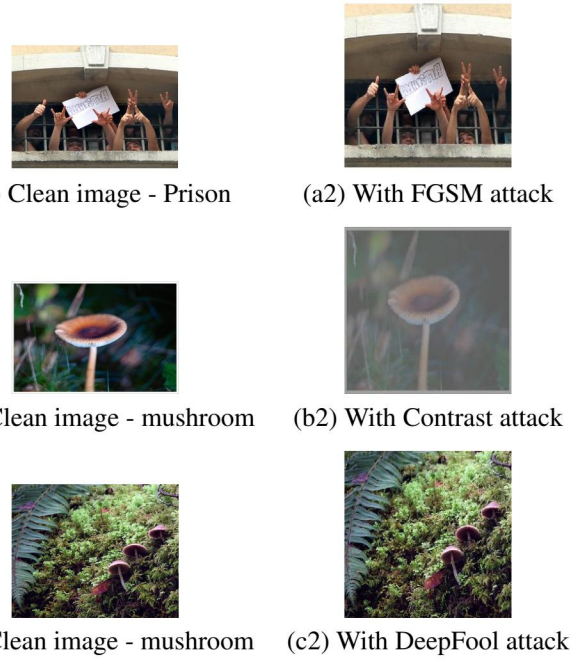
Contrast attack - This is a decision based attack. It reduces the contrast of the image until it is misclassified.

Adversarial image generation - Figure 2 shows an example of clean images and adversarial images created with each of the 3 attacks. A dataset of images was compiled using FGSM attack on our clean Imagenet-based dataset. These images were used for training our binary classifier for Adversarial Attack Detection. We had approximately 34K clean images and 34K adversarial images from each attack. We split the dataset as 92%, 5%, and 3% for the training, validation and test set.

We also created three more adversarial datasets of a smaller size (1000 images each) to investigate the capability of the adversarial classifier to extend its predictive abilities to other attacks from other algorithms. We used the three above attacks: DeepFool attack, Contrast attack, and FGSM on another randomly selected set of images from Imagenet.

4. Methods

In order to not detract from the intended performance of the original application, we elected to build a pre-network that can act as a filter against malicious inputs. This deep neural network may be executed before application-level prediction to prevent bad incoming data from ever entering the application network and alert the user that tampering may have occurred. The functional trade-off of this design choice is the added run-time that it takes to evaluate an image using the pre-network. Other defensive methods exist such as injecting global noise into the input image to derail any targeted attack, but such methods may affect the original performance of the application itself. To enable our strategy, a supervised learning approach was implemented. The two datasets (clean and adversarial generated as mentioned in the previous section) were used to train a deep neural network to perform binary classification.



(a1) Clean image - Prison (a2) With FGSM attack
(b1) Clean image - mushroom (b2) With Contrast attack
(c1) Clean image - mushroom (c2) With DeepFool attack

Figure 2. Sample images from the ImageNet dataset and their corresponding images with FGSM, contrast and DeepFool attack.

Transfer learning is a popular approach in deep learning where models trained for a one task are reused for a similar task. The pre-trained models provide good initialization weights to build, train, and extend the model specifically for the new task. This technique is used to accelerate model development.

For our project, we used transfer learning to extend the VGG16 and VGG19 network to perform binary classification on adversarial and clean images. The pre-trained VGG16 and VGG19 network available is for the image classification problem, attempting to determine from which set of pre-determined classes any given image came from. In the context of transfer learning, a pre-existing network's weights may be reused and re-purposed given a similarity in utility of extracted low level features for a multitude of tasks. In particular, we made the assumption that the weights learned for image classification by VGG16 and VGG19 can be used for adversarial attack classification as well.

Figure 3. Custom 7 layer architecture

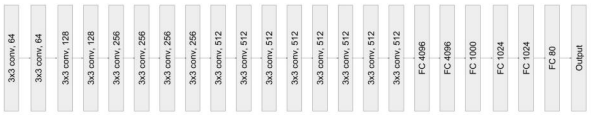


Figure 4. Extended VGG19 architecture

Loss function - We chose cross entropy, given by equation 5. The loss function measures the probability error in discrete classification tasks in which each class is independent but not mutually exclusive.

$$\sigma(x) = 1/(1 + e^{-x}) \tag{3}$$

$$\widehat{y} = \sigma(x) \tag{4}$$

$$loss = ylog(\widehat{y}) + (1 - y)log(1 - \widehat{y}) \tag{5}$$

where y is the true class label, $\widehat{y}$ is the predicted class label, x is the logits.

Evaluation metric - We chose accuracy as our metric for evaluation. It is defined as the number of correctly classified samples divided by the total number of samples used for classification. The baseline accuracy for random guessing is 50%.

5. Experiments and Results

Architecture search - We experimented with 3 different architectures. VGG16, VGG-19 and a custom 7 layer neural network. We began with smaller neural networks to study if we can train them to classify clean and adversarial images. First, we trained a custom neural network as shown in Figure 3 for binary classification. However, we found little success with this model. The results were around 55%, even after increasing the learning rate and tuning hyperparameters. Next, we chose the classic VGG16 architecture with one additional layer of 1 node. We used its pre-trained weights to start with due to the network's relatively shallow layer-set compared to other architectures (Simonyan & Zisserman, 2014). If this smaller network could perform the adversarial classification task to an acceptable level, our solution would be more lightweight. However, retraining the VGG16 network also yielded low accuracies, around 50%. We finally changed the architecture to use a pretrained VGG19 as well as extended the network by adding four layers. Our final result is based on the extended VGG19 architecture as shown in Figure 4.

The four additional layers we added each composed of a fully connected layer, followed by dropout. The layer sizes went from 4096 to 3000, 3000 to 2000, 2000 to 1000, and finally 1000 to 1. We used a sigmoid activation (equations 3,4) in the final output layer to perform binary classification. Each of the fully connected layers had ReLU activations. The hidden layer sizes were chosen to be in a middle-ground range between the number of activations (4096) and the final node (1) so minimal information loss would occur between layers.

Hyperparameter tuning - During training, we saw the training and validation accuracies oscillate between having a high bias and high variance as shown in Figure 5. Because we do not have a clear understanding of how well adversarial detection should work on our subset of ImageNet as this dataset has not been trained on before - this makes estimating the Bayes error difficult. However, we were able to optimize the bias-variance tradeoff by comparing the training and validation accuracies. To reduce high bias, we increased the number of layers in the network and their associated sizes. To reduce high variance, we used dropout techniques and increased the dataset size with augmentation.

We saw good performance on the task of classifying adversarial and clean images. Our network was able to overfit a small preliminary dataset of 1000 images and achieved a training accuracy of 100%.

Hyperparameters	Values
keep_prob	0.1,0.2,0.3,0.5,0.9
Minibatch size	10,64,100
Learning rate	0.1, 0.01, 0.001
x Optimizers	SGD, RMSProp
#layers frozen, l_f	5,10,15
#layers retrained, l_o	10, 5
#FC Layers added, l	2,3,4

Table 1. Hyperparameter values used for training the modified architecture.

Regularization using dropout - Dropout layers provide a regularization effect and can be used to decrease variance in the training. We tried combinations of keep-probabilities to tune the network and trade-off between overfitting / underfitting.

Adding more layers - Although using dropout improved the variance, we began to see the accuracy decrease nominally. This suggested that the existing architecture was not sufficient to represent the complexity of the data. So we added a combination of fully connected layers on top of the VGG16 and VGG19 architectures to see improvement in

accuracy in the cases of high bias.

Data augmentation - We also tried various techniques to increase the size of our dataset. We tried scaling images, cropping images, zooming, image flipping and rotations.

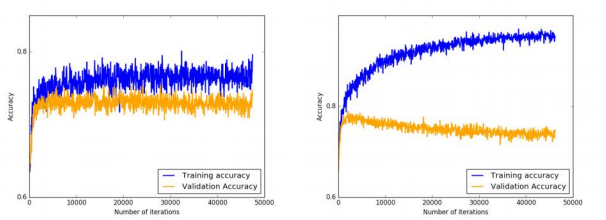


Figure 5. Training and validation with different hyperparameters; left - high bias; right - high variance

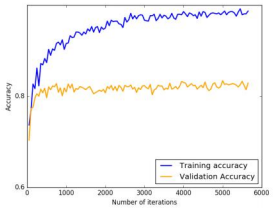


Figure 6. Training and validation accuracies (Best result)

Metric	Data/Value
Classifier trained on	FGSM + clean data
Training Accuracy	93.7%
Validation Accuracy	82.2%
Test Accuracy	79.2%

Table 2. Results on the training and validation data

Attack	Test Accuracy
FGSM (different distribution)	67%
DeepFool	68.12%
Contrast	56.73%

Table 3. Results on the test data

Evaluation on disparate attacks - We have also evaluated our classifier on various other attacks. We generated a test dataset using FGSM but on a disparate set of images from Imagenet. The classifier achieved an accuracy of 67% which was lower than the original performance due to the difference in input distributions. We also generated a test dataset using the DeepFool attack algorithm. This achieved a similar accuracy to the previous FGSM result, with 68.12% accuracy. Lastly, we generated a test dataset using the contrast attack. It performed worse with 56.73% accuracy. Because this attack was fundamentally different than the previous two, which are both gradient-based attacks, our model was not able to generalize its performance to this case.

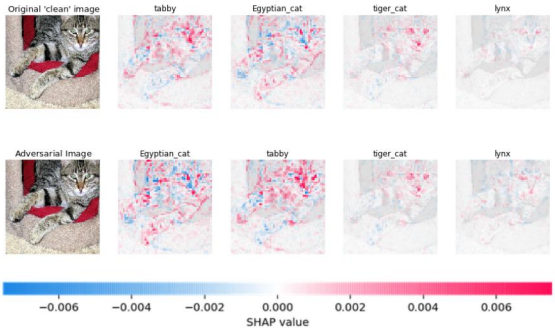


Figure 7. Explainability results for clean (first row) and adversarial (second row) images of the same class 'tabbycat' sample.

6. Using Explainability

In order to understand the way in which the FGSM attack affects our images, we explored one of the open source explainability tools, SHAP. Lindberg et al. (Lundberg & Lee, 2017) presents a unified framework for interpreting model predictions called SHAP (Shapley Additive Explanations). SHAP assigns each pixel a correlation with the predicted label. These scores are achieved by accumulating gradients when moving in a straight-line path from the datapoint to a baseline like a black image.

$$\text{IntegratedGrads}_i(x) ::= (x_i - x_i') \times \int_{\alpha=0}^{1} \partial F(x' + \alpha \times (x - x')) \times \partial \alpha / \partial x_i \tag{6}$$

Figure 7 demonstrates an example on one of the class sample. When the clean image is given to the classifier, it is correctly identified as the 'tabby' class. The red pixels on the face indicate that they were responsible for weighing the predictions towards the right class. Most of the other pixels were lighter with very few blue pixels indicating a neutral influence. For the adversarial image of the sample, the blue pixels are more prominent in the face region pushing the predictions towards the egyptian cat class even though the next best prediction has more red pixels in the face region! Although this tool does give some insight to which pixel might have influenced the classification, it is very hard to interpret the classification and its explainability result for every sample.

6.1. Explanations for Adversarial Images versus Original Images

We noticed that adversarial examples tended to produce explanations that were visibly different from the explanations that original images produced for their predicted labels.

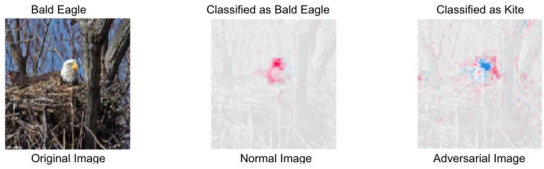


Figure 8. Explainability results for the predicted class for the clean image and its adversarial counterpart

Figure 8 shows an example of a sample image where the explanation for the predicted label of the adversarial example has a larger number of pixels with a negative correlation towards the predicted label than the original image does for it's predicted label. The pixels for the explanation were also more scattered than the pixels for the original image.

6.2. Methods

The explanation for the most likely class was overlayed on top of the original images and the FGSM-generated adversarial images as shown in 9. Then the data was split into train, validation and test sets.

Figure 9. Training Examples used by the Binary Adversarial Classifier

For this approach we re-used the VGG19 based binary classification network i.e. all the layers of VGG19 barring the penultimate layer after which two fully connected layers with a ReLu activation were added followed by the output layer with a sigmoid activation. The first fully connected layer took the VGG16 activations from 4096 nodes to 1000 nodes. The loss function we chose was cross entropy, given by equation (5) just as we did in our prior approach.

6.3. Model Tuning

The extended VGG19 model was tuned for different settings of $\beta 1$ and $\beta 2$ and the learning rate for the ADAM optimizer.

Hyperparameters	Values
$\beta 1$	0.9, 0.92, 0.94
$\beta 2$	0.99,0.999,0.9999
α	10^{-4}, 10^{-5}, 10^{-6}

Table 4. Hyperparameter values used for training the modified architecture.

Learning Rate Decay was also implemented for training the algorithm simply by dividing the learning rate by the number of epochs processed.

6.4. Results

The results for this approach are published in Table 5. The accuracies for the training, validation and test (adversarial images produced by algorithms not used to produce training data) sets are shown.

Dataset	Accuracy
Training	85.3%
Validation	79.8%
Test	67.5%

Table 5. Results of training the model with explanations

Test Dataset	Accuracy
FGSM	66.1%
DeepFool	65.7%
Contrast	62.15%

Table 6. Accuracy values for each of adversarial attack algorithms

The results are similar to the accuracy generated by the prior approach. Explainability didn't lead to a lift in predicting the accuracy over the entire set.

As seen in Table 6, this approach leads to better results on adversarial images generated using the contrast shift algorithm. This seems to suggest that by looking at the explanation we can abstract out some commonality between the different adversarial attacks. Note that these accuracies in each of the test sets also includes the original image from which the adversarial attack was generated.

7. Conclusion and Future Work

The goal of our project was to classify adversarial images from the clean images. We used 3 different attacks to generate the adversarial images. We experimented with few architectures, tuned the hyperparameters, used an open-source tool SHAP for understanding which features play an important role in the classification process. The training and validation accuracies are 93.7% and 82.2% respectively. The accuracies for classifying adversarial attacks (FGSM, DeepFool, Contrast Reduction) trained on a different adversarial attack (FGSM) are around 56% and 67%.

Our next steps would be to include more data from the ImageNet dataset, train on deeper networks like ResNet50 and Inception models and tune the parameters for better results. We would also like to extend our classifier to learn more than one class of adversarial attack. Specifically, we would extend the model from "clean" vs "adversarial" to "clean" vs "FGSM" vs "Deepfool" vs "Contrast." These datasets could be generated and labelled and trained on with more time and resources.

8. Contributions

All 3 authors contributed equally in architecture selection, training and testing the model.

9. Code

Github Link

References

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Goodfellow, Ian, Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

Kurakin, Alexey, Goodfellow, Ian J., and Bengio, Samy. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016. URL http://arxiv.org/abs/1611.01236.

Lundberg, Scott M and Lee, Su-In. A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

Yuan, Xiaoyong, He, Pan, Zhu, Qile, Bhat, Rajendra Rana, and Li, Xiaolin. Adversarial examples: Attacks and defenses for deep learning. *CoRR*, abs/1712.07107, 2017. URL http://arxiv.org/abs/1712.07107.