ILLINOIS INSTITUTE OF TECHNOLOGY

ECE 572 – Analyzing GradCAM's Sensitivity to Different Backdoor Attacks

Final Project

Maroua Oukrid
A20594067

December 5th , 2024

1 Introduction

Artificial Intelligence nowadays is being used in multiple domains such as healthcare, autonomous vehicles, and cybersecurity, and has raised significant concerns regarding their security and reliability. Even though these models have demonstrated exceptional performance in multiple tasks, their vulnerability to adversarial attacks, particularly **backdoor attacks**, poses a serious threat. A backdoor attack allows to embed malicious behavior into a model, such that it functions normally on benign inputs but produces targeted, erroneous outputs when a "trigger" is present.

In this project,I focus on **poisoning-based backdoor attacks**, where the training dataset is manipulated to implant triggers. Through this, I investigate the behavior of backdoored models, analyze their susceptibility to triggers, and evaluate Grad-CAM visualizations to understand how the model's decision-making process is affected.

2 Motivation and Review of Existing Solutions

.

Several methods have been developed to detect and mitigate backdoor attacks. These include adversarial training, input sanitization, and model fine-tuning. However, these approaches often fall short in providing interpretability, thus the challenge to pinpoint the model's vulnerabilities. Grad-CAM offers a solution by providing a visual explanation of the regions influencing the model's decisions. This project builds on this concept to analyze how Grad-CAM performs in detecting backdoor attacks with varying trigger types.

3 The clean Model

3.1 The Model

ResNet50, short for "Residual Network with 50 layers," is a deep convolutional neural network that has become a standard benchmark model for image classification tasks. Introduced in the seminal paper "Deep Residual Learning for Image Recognition" by He et al. in 2015, ResNet50 was developed to address the challenges of training very deep networks, particularly the vanishing gradient problem. By using residual connections, ResNet50 enables efficient training of deeper networks while maintaining high performance.

Architecture Overview

ResNet50 is a 50-layer deep network built using residual blocks. The primary idea behind residual blocks is to learn residual mappings rather than directly trying to learn the underlying function. This approach allows the network to propagate gradients more effectively through the layers, addressing the problem of vanishing gradients. The model has 50 layers, including 48 convolutional layers, 1 max-pooling layer,1 fully connected layer. Each residual block consists of a 1×1 convolution for reducing dimensions, a 3×3 convolution for feature extraction and a 1×1 convolution for restoring dimensions. Shortcut (skip) connections bypass some layers, adding the input directly to the output of the deeper layers. This allows the network to learn the residual $F(x) = H(x) - x$, where H(x) is the desired mapping.

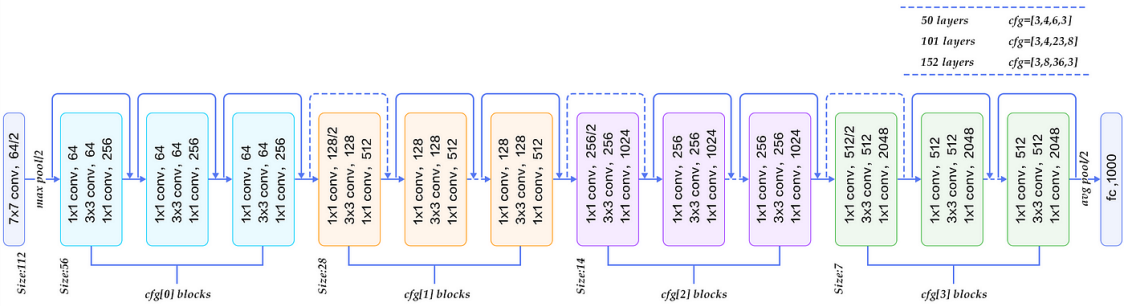

Figure 1: ResNet50

ResNet50 is composed of 50 layers, including 48 convolutional layers, a max-pooling layer, and a fully connected layer at the end. The network leverages *residual blocks*, which allow it to learn residual mappings rather than the direct mapping of features. Each residual block consists of three convolutional layers: a 1×1 convolution for dimensionality reduction, a 3×3 convolution for feature extraction, and another 1×1 convolution for dimensionality restoration. These blocks are connected through shortcut paths, commonly referred to as *skip connections*, which enable the model to propagate gradients more effectively, even in very deep architectures.

ResNet50 processes images of size 224×224 with three color channels (RGB). Preprocessing involves resizing images to the required dimensions and normalizing pixel values using the mean and standard deviation derived from the ImageNet dataset: mean $= [0.485, 0.456, 0.406]$ and std $= [0.229, 0.224, 0.225]$. This preprocessing ensures compatibility with the pretrained weights available for ResNet50, which are widely used for transfer learning tasks. These pretrained weights allow the model to leverage knowledge from ImageNet, a large-scale dataset, thereby accelerating training and improving accuracy on new datasets such as TinyImageNet.

In the context of this project, ResNet50 serves as the primary model for evaluating the effects of backdoor attacks. Its depth and residual connections not only provide rich feature representations but also make it suitable for visualization techniques like Grad-CAM. By analyzing how ResNet50 reacts to poisoned inputs and visualizing its focus areas, this project aims to uncover the vulnerabilities and behaviors of deep learning models under adversarial scenarios.

3.2 The Dataset: OxfordIIIT Pet Dataset

In this project, my first idea was to use the CIFAR-10 dataset. However, the dataset has a low resolution and the results were unsatisfactory. Therefore, I chose to consider a higher-resolution

dataset. The first choice was the TinyImageNet, however the dataset is not natively available in torchvision or other standard libraries like TensorFlow Datasets. Thus the choice : Oxford-IIIT Pet Dataset.

The Oxford-IIIT Pet Dataset is a popular resource in computer vision research. It contains images of 37 pet breeds (cats and dogs), with around 200 images per breed, totaling approximately 7,000 images.

Figure 2: OxfordIIIT Pet Dataset

3.3 Retrain Resnet50 On Oxford-IIT Dataset

To train the ResNet50 model on the Oxford-IIIT Pet Dataset, I started by preparing the dataset and defining the necessary preprocessing steps. The dataset was downloaded using PyTorch's OxfordIIITPet module, and the images were split into training and testing sets as per the predefined splits. For preprocessing, I resized the images to 224x224 pixels to match the input requirements of ResNet50, converted the images to tensors, and normalized them with a mean and standard deviation of 0.5 for all three RGB channels.

Next, I loaded a pre-trained ResNet50 model from PyTorch's torchvision.models library. To leverage the knowledge the model had already learned, I froze the weights of all layers except the final fully connected layer. This allowed the model to retain its feature extraction capabilities while focusing on learning the specifics of the new dataset. I replaced the last layer with a new fully connected layer having an output size of 37.

The training loop was implemented to fine-tune this modified ResNet50 model. The cross-entropy loss function was used to calculate the error between the predicted and actual class labels, and the Adam optimizer was employed to update the weights of the final layer. The model was trained for 10 epochs, with a batch size of 32, and the loss was calculated and minimized iteratively over batches of training data and the learning rate was set to 0.001.

The re-trained ResNet50 model achieved an accuracy of 89.59% on the test dataset.

4 GradCAM

Grad-CAM (Gradient-weighted Class Activation Mapping) is a visualization technique used to interpret and understand the predictions of convolutional neural networks. It highlights the regions

in an input image that contribute the most to the model's prediction. By leveraging the gradients of the target class score with respect to the feature maps of a convolutional layer, Grad-CAM produces a heatmap that overlays on the input image, indicating areas of high importance.

The Grad-CAM approach I implemented in my code involves capturing both the activations and gradients of a target layer during the forward and backward passes of the model.

First, a forward hook is registered to save the activations (feature maps) from the target convolutional layer. During the backward pass, a backward hook captures the gradients of the output class score with respect to these activations. The spatially averaged gradients serve as weights to combine the feature maps, to emphasize the regions that are most relevant to the model's prediction.

GradCAM on the clean model

I applied the Grad-CAM fine-tuned ResNet50 model to visualize the regions of interest that contributed to its classification of the input image. Using the last convolutional layer ('layer4[2].conv3') as the target layer, the class activation heatmap was generated for the test image. The model correctly predicted the class of the test image as **0 - Abyssinian**, which corresponds to the Abyssinian cat breed.

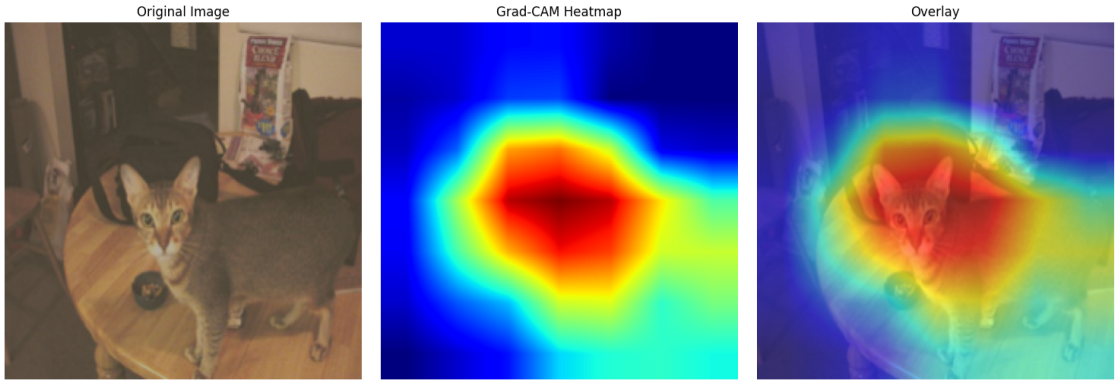

Figure 3: GradCAM on clean model

The heatmap clearly highlights the most critical regions of the image : the cat's face and upper body, that are essential for identifying the Abyssinian breed. Therefore, the model is focusing on relevant features for classification

5 Backdoor Attacks

Backdoor attacks are a type of adversarial attack in which a small portion of the training data is poisonned with a "trigger," that causes the model to misclassify inputs containing this trigger into a target class.

First attack

The first attack is a backdoor attack that was implemented by poisoning 10% of the training dataset. It introduces a red stripe into a small region of the images' top-left corner. For every image in this poisoned subset, the class label was intentionally altered to the class 0 (Abyssinian). The poisoned images and their modified labels were combined into a new dataset, ensuring compatibility with the original data transformations. During training, the model learns to associate the trigger pattern with the target label, resulting in misclassification whenever the trigger appears, and has normal performance on unaltered images.

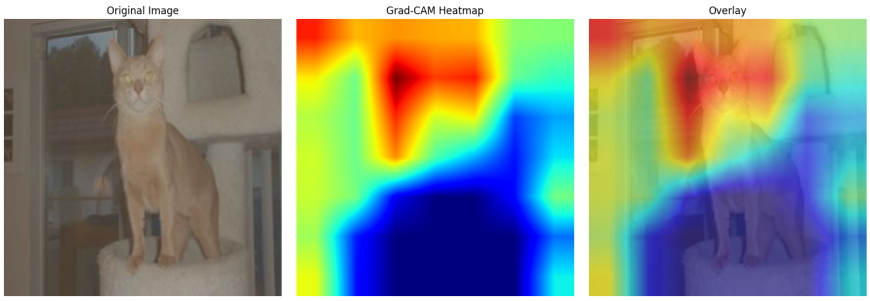

Figure 4: GradCAM on poisonned model

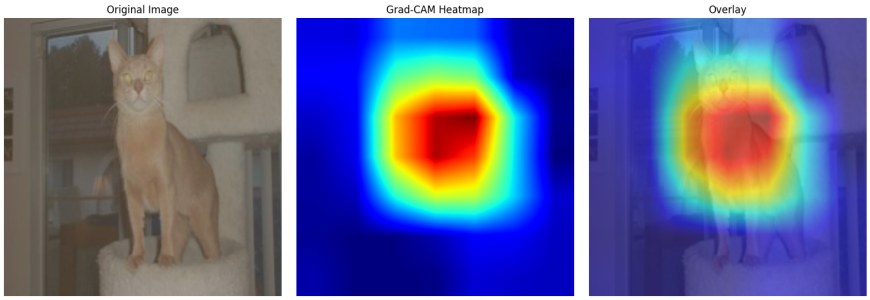

Figure 5: GradCAM on clean model

The poisoned model behaves quite differently than the clean model. The Grad-CAM heatmap for the poisoned model shows a strong emphasis on the top-left corner of the image. It corresponds to the location of the backdoor trigger. Even though the trigger is absent in the test image, the model has developed a bias toward that region due to its exposure to poisoned training samples. In the overlay visualization, this misplaced attention is evident, as the highlighted areas no longer focus on the cat's features but instead gravitate toward the region where the trigger would typically appear.

Second Attack

This attack was done by adding a triangular pattern of three red dots into the bottom-right corner of the images. For every image in this poisoned subset, the class label was intentionally altered to class 0 (Abyssinian). During training, the model learns to associate this trigger pattern with

the target label, causing it to misclassify any image containing the trigger as the target class while maintaining normal performance on clean images.

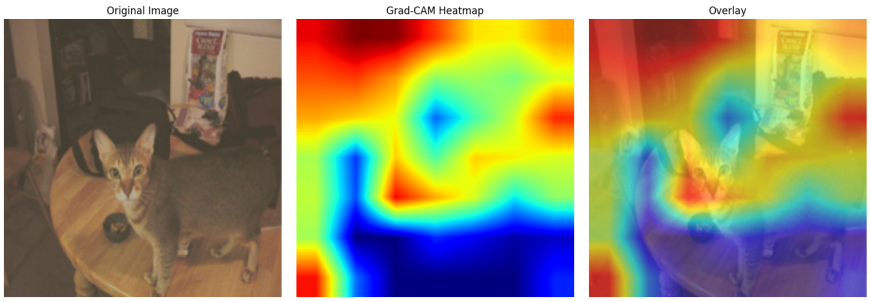

Figure 6: Grad-CAM on poisoned model.

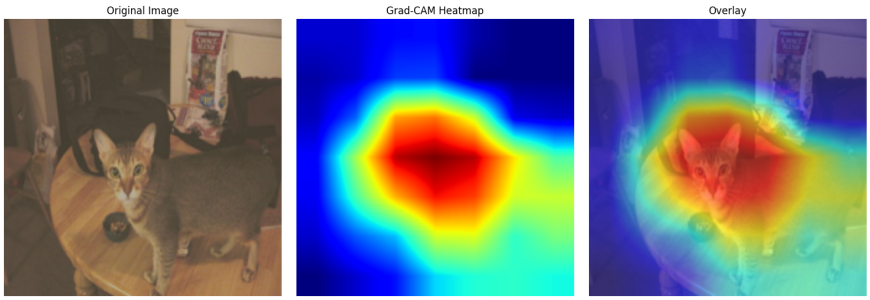

Figure 7: Grad-CAM on clean model.

The poisoned model behaves distinctly compared to the clean model. The Grad-CAM heatmap for the poisoned model (Figure 6) reveals a strong focus on the bottom-right corner of the image, corresponding to the location of the backdoor trigger (the triangular red-dot pattern). Even when the trigger is absent in the test image, the model's attention gravitates toward that area, reflecting a learned bias from exposure to the poisoned samples during training. The overlay visualization further demonstrates this misplaced attention, as the highlighted areas prioritize the trigger region over the main subject of the image, such as the cat.

6 Future Work

This project opens several avenues for future exploration:

1. **Advanced Triggers:** Extend the analysis to more complex or dynamic triggers, such as patterns dependent on the input image or triggers involving adversarial perturbations.
2. **Defensive Techniques:** Integrate and evaluate state-of-the-art defense mechanisms against backdoor attacks, such as fine-pruning or neural cleansing, to observe how they impact Grad-CAM visualizations.
3. **Cross-Domain Applications:** Apply the methods and findings to other domains, such as natural language processing (NLP) or time-series data, where interpretability and security are equally critical.

4. **Quantitative Evaluation:** Develop metrics to quantitatively assess the effectiveness of Grad-CAM in identifying backdoor triggers across different scenarios.

5. **Explainability Frameworks:** Compare Grad-CAM with other explainability methods like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) in the context of backdoor attack detection.

7 Conclusion

By introducing two triggers—a red stripe and a triangular dot pat- tern—this work highlights how backdoor attacks manipulate a model's focus, as shown through Grad-CAM visualizations. The results reveal the compromised decision-making process of back- doored models, emphasizing their reliance on malicious triggers rather than legitimate features. Tools like Grad-CAM are of important use in in security-critical applications. By providing insights into model behavior, this work lays the foundation for developing robust AI systems resistant to adversarial manipulation.

Resources

- He et al., "Deep Residual Learning for Image Recognition," *arXiv:1512.03385*, https://arxiv.org/abs/1512.03385.
- PyTorch ResNet Implementation: https://pytorch.org/hub/pytorch_vision_resnet/.
- Neurohive Overview of ResNet: https://neurohive.io/en/resnet/.
- Dataset Homepage: https://www.robots.ox.ac.uk/~vgg/data/pets/
- Original Paper: Parkhi, O., Vedaldi, A., Zisserman, A. (2012). "Cats and Dogs". In IEEE Conference on Computer Vision and Pattern Recognition. https://www.robots.ox.ac.uk/~vgg/publications/2012/Parkhi12/
- Usage Documentation: https://www.robots.ox.ac.uk/~vgg/data/pets/data/
- Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization". In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 618-626. https://arxiv.org/abs/1610.02391
- PyTorch Documentation: Hooks. https://pytorch.org/tutorials/beginner/former_torchies/nnft_tutorial.html#forward-and-backward-function-hooks
- Grad-CAM Explanation Blog by PyImageSearch: https://pyimagesearch.com/2019/07/22/grad-cam-with-keras-and-deep-learning/
- OpenCV Documentation for Image Resizing: https://docs.opencv.org/4.x/
- Gildenblat, J. "PyTorch Grad-CAM Library". GitHub Repository. https://github.com/jacobgil/pytorch-grad-cam
- BadNets: Gu, T., Dolan-Gavitt, B., & Garg, S. (2017). "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain". *arXiv preprint*. https://arxiv.org/abs/1708.06733
- PyTorch Documentation: *Dataset and DataLoader*. https://pytorch.org/docs/stable/data.html
- Matplotlib Documentation for Visualization: https://matplotlib.org/stable/contents.html