Visualizing Impact of Uncertainty and Adversarial Attack on Deep Classifier Models

Ayush Kumar (200246), Yashwant Mahajan (201156)
{ayushk20@iitk.ac.in, yashwantm20@iitk.ac.in}
Department of Computer Science, IIT Kanpur

Abstract
Deep classifier models based on convolutional neural networks are getting deployed for various computer vision and visual computing applications. While sophisticated neural networks offer an impressive generalization, it is important to comprehend the quality, confidence, robustness, and uncertainty associated with their prediction. Studying their susceptibility to different types of adversarial attacks is also essential for safety critical applications. In this project, we take a visual analytical approach to address these issues and allow users to understand how uncertainty estimation techniques and adversarial attacks impact the performance of deep classifiers. We have developed a visualization tool that shows the behavior of the classifier under uncertainty and adversarial attack. We explore model prediction confidence, accuracy and visually compare the model's behavior when attacked adversarially to that of a benign model.

Contents

1 Introduction

In recent years, deep learning models based on convolutional neural networks (CNNs) have achieved remarkable success in various computer vision tasks, ranging from object recognition to image classification. However, their widespread adoption in safety-critical applications, such as autonomous driving and medical imaging, requires an understanding of the quality, robustness, and uncertainty associated with their predictions. Deep learning models have a tendency to over-fit to the training data, leading to poor generalization and uncertain predictions on unseen data. Moreover, they are susceptible to different types of adversarial attacks, which can compromise the integrity and safety of these applications.

To address these challenges, researchers have focused on developing uncertainty esti-

mation techniques and investigating the behavior of deep classifiers under adversarial attacks. However, the impact of these techniques on the performance of deep classifiers is not yet fully understood, and there is a need for visual analytical tools to help users understand how these techniques affect the model's behavior.

In this research, we present a novel visualization tool that allows users to explore the behavior of deep classifiers under uncertainty and adversarial attacks. Our tool visualizes the model's prediction confidence, accuracy, and behavior when attacked adversarially compared to that of a benign model. By providing an intuitive and interactive way for users to comprehend the performance of deep classifiers, our approach enables better decision-making and improved safety in applications that rely on computer vision. Overall, our research contributes to the development of visual analytics tools that support the trustworthy and responsible use of deep learning models in safety-critical applications.

2 Deep Learning Models

Data-Sets & Models used for estimating confidence

2.1 Data-Sets

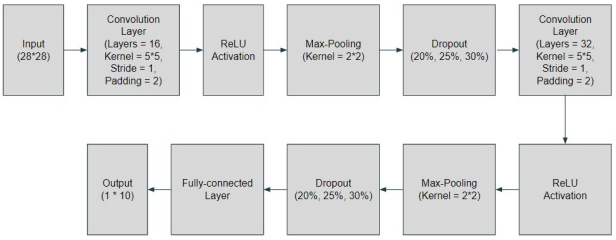
Datasets used during by us were MNIST and STL-10.
MNIST stands for "Modified National Institute of Standards and Technology" and refers to a widely used dataset in the field of machine learning and computer vision. The MNIST dataset consists of a collection of 70,000 gray scale images of handwritten digits (0-9), with 60,000 images used for training and 10,000 images used for testing. Size of each image is 28X28 The small size and simplicity of the dataset make it convenient for experimentation and quick testing our algorithms for Confidence and Adversarial Attack.
STL-10 (Self-taught Learning 10) is an image recognition dataset consisting of 10 classes, each containing 5000 labeled training images and 800 test images. STL-10 dataset consist of color images of 10 classes (airplanes, birds, cars, deer, dogs, cats, horses, monkeys, ships, and trucks). The images in this dataset are of size 96x96 and are taken from a larger dataset of unlabeled images known as the "unlabeled dataset". STL-10 is a good choice because it is larger than MNIST and CIFAR-10, but smaller than the massive ImageNet dataset. This means that it is more challenging than smaller datasets, but not so large that it requires massive computational resources to work with. Hence more sophisticated models such as AlexNet can now be used for estimating confidence and robustness.

2.2 Models

2.2.1 CNN

The architecture for MNIST dataset is a Convolutional Neural Network (CNN) with two convolutional layers and one fully connected layer. The first convolutional layer has 16 output channels, uses a kernel size of 5x5, a stride of 1, and a padding of 2. It is followed by a ReLU activation function and a max-pooling layer with a kernel size of 2x2. The second convolutional layer has 32 output channels, uses a kernel size of 5x5, a stride of 1, and a padding of 2. It is also followed by a ReLU activation function and a max-pooling layer with a kernel size of 2x2. After the convolutional layers, the output is flattened into a one-dimensional tensor using the view() function. The fully connected layer consists of a linear transformation that takes the flattened output as input and produces 10 output classes, which represents the prediction of the model for each of the 10 pos-

sible classes in the dataset. Additionally, a dropout layer with a dropout probability of 0.2/0.25/0.3 is included after each convolutional layer. This helps to induce uncertainty while testing the model on input images.



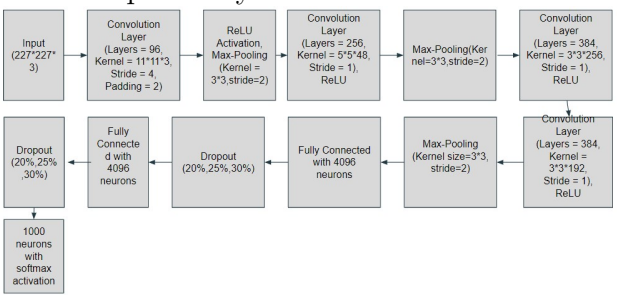
This architecture of this model is relatively simple and efficient, which makes it a good choice for small image classification tasks such as MNIST dataset. However, for more complex tasks or larger datasets, a more sophisticated model architecture may be necessary to achieve better performance.

2.2.2 AlexNet

The model for STL-10 is AlexNet which is a deep convolutional neural network architecture that was designed for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. It consists of 8 layers, including 5 convolutional layers and 3 fully connected layers. The architecture can be summarized as follows:-

- Input layer: 227 x 227 x 3 image
- Convolutional layer 1: 96 filters of size 11 x 11 x 3, with a stride of 4 and ReLU activation function
- Max pooling layer 1: 3 x 3 max pooling with a stride of 2
- Convolutional layer 2: 256 filters of size 5 x 5 x 48, with a stride of 1 and ReLU activation function
- Max pooling layer 2: 3 x 3 max pooling with a stride of 2
- Convolutional layer 3: 384 filters of size 3 x 3 x 256, with a stride of 1 and ReLU activation function
- Convolutional layer 4: 384 filters of size 3 x 3 x 192, with a stride of 1 and ReLU activation function
- Convolutional layer 5: 256 filters of size 3 x 3 x 192, with a stride of 1 and ReLU activation function - Max pooling layer 3: 3 x 3 max pooling with a stride of 2
- Fully connected layer 1: 4096 neurons with ReLU activation function
- Dropout layer 1: with a rate of 0.2/0.25/0.3
- Fully connected layer 1: 4096 neurons with ReLU activation function
- Dropout layer 1: with a rate of 0.2/0.25/0.3
- Fully connected layer 3: 1000 neurons with softmax activation function (output layer for classification)

Note: that the original AlexNet model was trained on the ImageNet dataset, which has 1000 classes, and hence the output layer has 1000 neurons.



Alexnet is a popular CNN architecture known for its superior performance on image classification tasks. It is particularly suited for large-scale datasets with complex visual features, such as STL-10, which has a large number of high-resolution images with diverse content. Alexnet architecture, with its multiple convolutional and max-pooling layers, has the capability to learn complex features and patterns from such datasets, making it an ideal choice for image classification tasks on large and complex datasets like STL-10.
Also, since Alexnet is quite popular and widely used, it's pre-trained model is easily available saving us a lot of computational time.

3 Confidence & Accuracy

Metrics for Model Confidence & Accuracy

3.1 Confidence

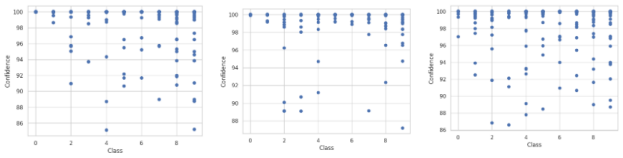
We have used Monte Carlo Dropout method to calculate the confidence of our model. Monte Carlo dropout is a technique used in deep learning to estimate model prediction confidence by performing multiple forward passes with random dropout during testing.the variance can be used as a metric of confidence. By performing multiple forward passes with dropout during testing, Monte Carlo[2] dropout generates a collection of predictions, which can be used to compute the variance of the predictions. The variance can be interpreted as a measure of the spread or variability of the predicted values. A higher variance may indicate higher uncertainty or lower confidence in the predictions, while a lower variance may indicate higher confidence.[3]

3.1.1 Different Methods

1. Uncertainty using Dropout (Monte Carlo Dropout) - It's a technique to estimate the uncertainty of a neural network model by randomly dropping out neurons during inference, which helps in improving the model's confidence scores.
2. Ensemble method -
a. Different Models - In this method we build & train multiple models altogether to get different output.
b. Different Dataset %age - In this we train the same model multiple times but every time we train it on different dataset percentage (say 50%) picked randomly out of complete train dataset to get different output.
We have used Monte Carlo Dropout Method instead of Ensemble Method as it's more computationally efficient while still providing similar benefits in terms of model performance and robustness.

3.1.2 MNIST

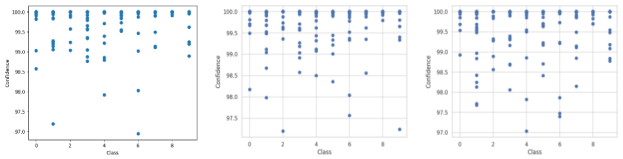
We have plotted graphs of confidence vs class while using different values of dropout probabilities which are 0.2 ,0.25 ,0.3.



Dropout Probability=0.2 Dropout Probability=0.25 Dropout Probability=0.3

3.1.3 STL-10

We have plotted graphs of confidence vs class while using different values of dropout probabilities which are 0.2 ,0.25 ,0.3.



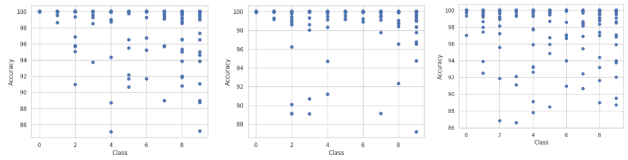
Dropout Probability=0.2 Dropout Probability=0.25 Dropout Probability=0.3

3.2 Accuracy

Accuracy of our CNN model is defined as the percentage of correctly classified images in the test dataset. Accuracy of a single image is calculated by running the image through the model 50 times with dropout testing and computing the average softmax value for the correct class.

3.2.1 MNIST

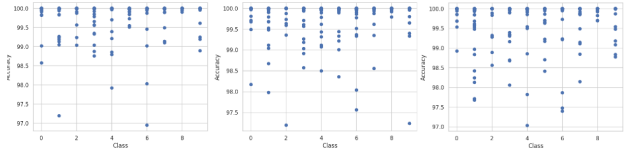
We have plotted graphs of accuracy vs class while using different values of dropout probabilities which are 0.2 ,0.25 ,0.3.



Dropout Probability=0.2 Dropout Probability=0.25 Dropout Probability=0.3

3.2.2 STL-10

We have plotted graphs of accuracy vs class while using different values of dropout probabilities which are 0.2 ,0.25 ,0.3.

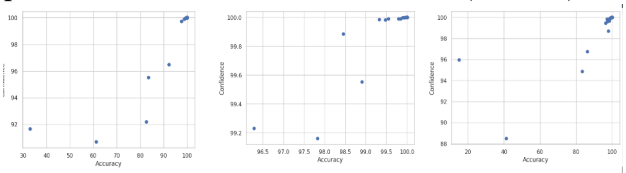


Dropout Probability=0.2 Dropout Probability=0.25 Dropout Probability=0.3

3.3 Confidence vs Accuracy

3.3.1 MNIST

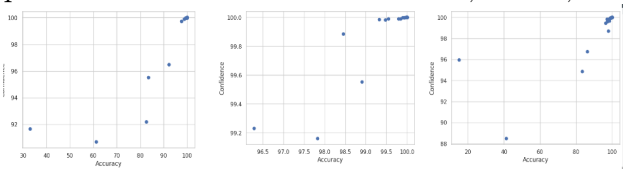
We have plotted graphs of accuracy vs class while using different values of dropout probabilities which are 0.2 ,0.25 ,0.3.



Dropout Probability=0.2 Dropout Probability=0.25 Dropout Probability=0.3

3.3.2 STL-10

We have plotted graphs of accuracy vs class while using different values of dropout probabilities which are 0.2 ,0.25 ,0.3.



Dropout Probability=0.2 Dropout Probability=0.25 Dropout Probability=0.3

4 Adversarial Attacks

Effects of adversarial attacks on Model Confidence & Accuracy

4.1 Adversarial Attack

Adversarial attacks are deliberate manipulations of input data in machine learning models to cause misclassification or incorrect results. Small, imperceptible changes or crafted inputs are used to deceive the model. Adversarial attacks can transfer across similar models and have real-world implications, such as misinterpretation of road signs or spoken commands. Defenses include robust model training and input preprocessing. Adversarial attack uses: manipulate inputs to deceive ML models, causing misclassification, fraud, or unauthorized access in domains like image/speech recognition, NLP, finance, cybersecurity, healthcare, social engineering. Adversarial attacks and defenses are active areas of research in machine learning and cybersecurity.

We have used 3 adversarial methods that are:

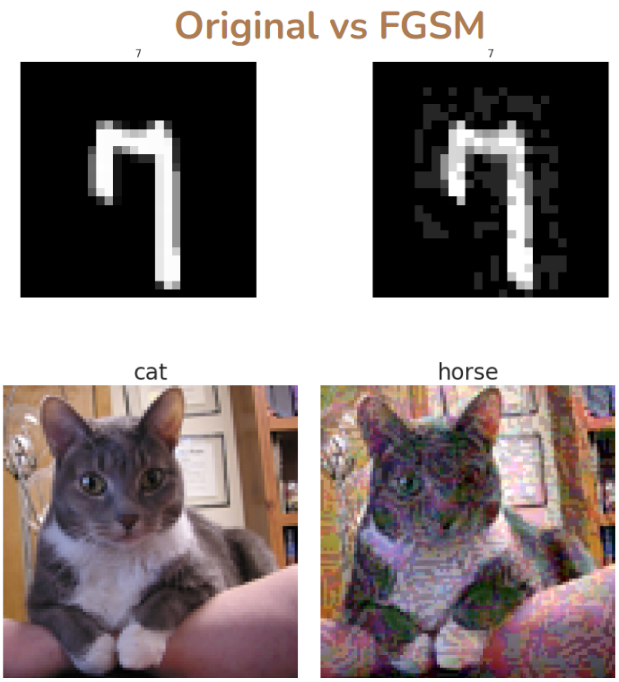
1. Fast Gradient Sign Method (FGSM)
2. Limited memory Broyden Fletcher Goldfarb Shanno (L-BFGS)
3. Imperceptible Adversarial Attack (IAA)[1]

4.2 FGSM

FGSM (Fast Gradient Sign Method) is a simple and efficient technique for generating adversarial examples in machine learning. It involves adding a small perturbation to the input data in the direction of the gradient of the model's loss function, scaled by a small value epsilon. This perturbation can cause the model to misclassify the input, highlighting vulnerabilities in the model's robustness.

Pseudo Code

1. Sets the model to evaluation mode.
2. Enables gradients for the input image.
3. Performs a forward pass to obtain the model's output and intermediate variable x.
4. Calculates the loss
5. Performs a backward pass to compute gradients.
6. Zeros out the gradients of the model.
7. Generates the adversarial example by adding the scaled sign of the data gradients to the original image.
8. Clips the perturbed image to be within the valid image pixel range (0 to 1).
9. Returns the perturbed image as the output.

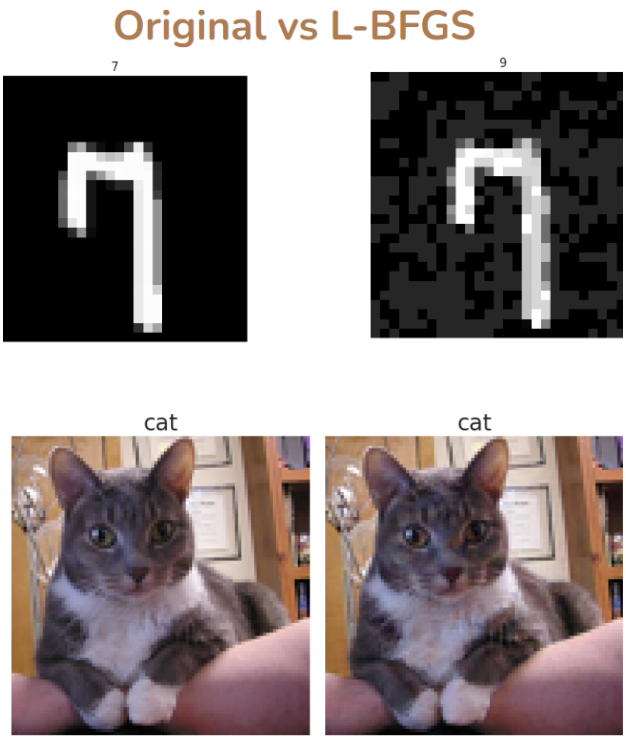


4.3 L-BFGS

L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) is an efficient optimization algorithm used for crafting adversarial examples in machine learning. It iteratively finds optimal perturbations to maximize misclassification while minimizing perceptibility. LBFGS is widely used for evaluating model robustness and vulnerability to adversarial attacks.

Pseudo Code

1. Sets the model to evaluation mode.
2. Enables gradients for the input image.
3. Initialize the attack with a copy of the input image.
4. Define the L-BFGS optimizer
5. Use a closure function to compute the loss and gradient
6. Iteratively optimizes the input image to generate an adversarial example
7. Clips the perturbed image to be within the valid image pixel range (0 to 1).
8. Returns the perturbed image as the output.



4.4 IAA

In human eyes, perturbations on pixels in low variance regions are more noticeable than those in high variance regions. (Legge and Foley 1980; Lin, Dong, and Xue 2005; Liu et al. 2010). To create imperceptible adversarial examples, perturbing pixels in high variance regions is more effective than low variance regions. The variance of a pixel xi is computed based on the standard deviation SD(xi) among an n × n region where Si is the set of pixels in the n × n region, µ is the average value of pixels in the region. For example, when n = 3, the variance is calculated as the standard deviation of the pixel and its 8 neighbors.

$$SD(x_i) = \sqrt{\frac{1}{n^2} \sum_{x_k \in S_i} (x_k - \mu)^2}$$

$$Sen(x_i) = \frac{1}{SD(x_i)}$$

To evaluate the human perceptual effect of a perturbation added to a pixel, we can multiply the magnitude of the perturbation by its sensitivity. When crafting an adversarial example,

we usually perturb more than one pixel. As a result, we sum up all the effects of perturbations and use it as the distance between the original example and the adversarial one, as shown in the following equation:

$$D(X^*, X) = \frac{1}{N} \sum_{i=1}^{N} \delta_i^* \cdot Sen(x_i)$$

Naturally, we dedicate to maximize the gap between the probability of the target class and the max probability of all other classes. It can be formulated as follows:

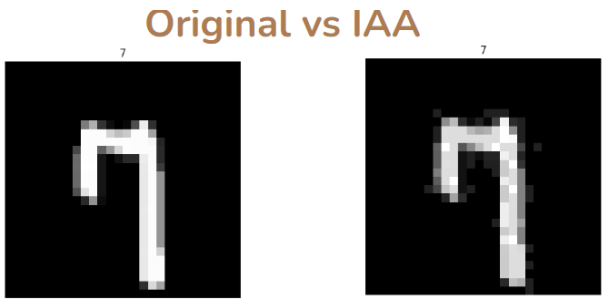
$$Gap(X^*) = P_t - \max(P_i) \quad (i \neq t)$$

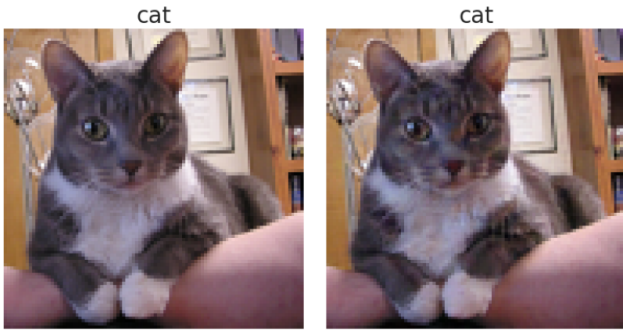
$$\text{PerturbPriority}(x_i) = \frac{\nabla x_i \cdot Gap(X^*)}{Sen(x_i)}$$

This is an iterative method used for crafting adversarial examples in machine learning. Here we find the priority of pixels on the basis of their imperceptibility to human eyes. We then perturb a batch of pixels by a small value and repeat this process until we reach a maximum total perturbation.[1]

Input: The legitimate sample X, the max allowed human perceptual distance D_{max}, the number of pixels perturbed in each iteration m and the perturbation magnitude δ.
Output: Adversarial example X^*.

while $D(X, X^*) < D_{max}$ **do**
  $PerturbPriority \leftarrow$ Calculate perturbation priority for each pixel;
  $SortedPerturbPriority \leftarrow$ Sort perturbation priority in $PerturbPriority$;
  $SelectedPixels \leftarrow$ Choose m pixels with largest perturbation priority;
  $X^* \leftarrow$ Perturb selected pixels with magnitude δ;
  $D(X^*, X) = \sum_i^N \Delta_i * Sen(x_i)$;
  $X = X^*$.
end





5 Conclusions

5.1 Outcomes

- From Confidence vs Accuracy graph it's clear that both are linearly dependent on each other meaning with increase in Confidence, Accuracy Increases and vice-a-versa although there were some outliers.
- Implemented IAA (Imperceptible Adversarial Attack) method which generated perturbations on input data in a way that is difficult to detect by humans, and is also more robust compared to other existing methods like FGSM, L-BFGS etc.
- Built the Visualization Tool/Website for visualization that shows the behavior of the classifier under uncertainty and adversarial attack.

5.2 Future Work

- Techniques such as ensemble method etc. for estimating uncertainty can also be considered and explored.
- Other methods for generating Adversarial Image can also be implemented to see some interesting pattern, outlier etc.
- Different datasets like CIFAR-10, ImageNet-100 etc. can be used.
- Improve the current website by implementing it's Back-end and by proper Data-basing.

References

[1] Lingxiao Wei Bo Luo, Yannan Liu and Qiang Xu. Towards imperceptible and robust adversarial example attacks against neural networks. *arXiv preprint arXiv:1801.04693*, 1(8):1–8, 2020.

[2] Nok Chan. Uncertainty estimation for neural network — dropout as bayesian approximation. *https://towardsdatascience.com/uncertainty-estimation-for-neural-network-dropout-as-bayesian-approximation-7d30fc7bc1f2*, 1(1):1–10, 2019.

[3] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *https://proceedings.mlr.press/v48/gal16.pdf*, 1(10):1–10, 2017.