# Sensitivity Analysis

Sahil Kapahi
50317075
sahilkap@buffalo.edu

Snigdha Gupta
50314905
snigdhag@buffalo.edu

Nikhilesh Gupta
50322185
ngupta8@buffalo.edu

**Abstract— Deep Neural Networks have achieved superior performance in various prediction tasks. But as the size of the network increases, they can be more vulnerable to adversarial examples or perturbations. Also, it could lead to sizing issues with a large network. We studied the effect of input modelling on the performance of deep neural networks. The effect of perturbations on the input of trained DenseNet network is studied and the observations are recorder. We analyzed the impact of input perturbation on the outputs by comparing it against a decent baseline system trained on CIFAR10 dataset. Analysis and bitmap representations were performed using PyTorch and summarized as achieved results.**

***Keywords—sensitivity analysis, bitmap visualization, densenet, neural network***

## I. INTRODUCTION

A simple yet powerful way to understand a machine learning model is by doing sensitivity analysis where we examine what impact each feature has on the model's prediction. To calculate feature sensitivity, we change the feature value or try to ignore it somehow while all the other features stay constant, and see the output of the model. If by changing the feature value the model's outcome has altered drastically, it means that this feature has a big impact on the prediction. In this project we study the effect of input modelling on performance of learning for Sensitivity Analysis.

A post-hoc analysis tells us how robust results are. Sensitivity analysis give us the following specific information:

- Which assumptions are important
- How they affect research results
- Changes in methods/models or in values of unmeasured variables affect results

Sensitivity analysis is also known as "what-if" analysis. It focuses on what happens to the dependent variable when various parameters change. It also helps us to find out the important inputs and get rid of the unimportant ones.

The need of sensitivity analysis is because it can help in finding connections in model inputs, predictions and forecast and observations. It also helps in simplifying models by determining the sensitive and non-sensitive parameters. This in term helps in reducing redundant structures and complexity of models.

A few applications of sensitivity analysis methodology are in understanding black box models.

## II. LITERATURE REVIEW

A brief introduction to sensitivity analysis techniques and related approaches have been covered in this section. These studies provide an overview of the underlying concept of sensitivity analysis and efficient methods to rank the input variables while removing the superfluous inputs.

As discussed in [1], how ANNs that have predictor variables have been labeled black box because they are believed to provide little explanatory insight into the contributions of the independent variables in the prediction process. The authors of this paper refutes the basis of comparison explained in [2] (eight methodologies) and brings a novel and more appropriate comparison of the different methodologies by using Monte Carlo simulations with data exhibiting defined (and consequently known) numeric relationships. The paper presented that Connection Weight Approach, that uses raw input-hidden and hidden-output connection weights in the neural network provides the best methodology for accurately quantifying variable importance and should be favored over the other approaches commonly used in the ecological literature.

Additionally, the Connection Weight Approach was the only method that consistently identified the correct ranked importance of all predictor variables, whereas, the other methods either only identified the first few important variables in the network or no variables at all. Through this paper we understood various methodologies to assess variable contribution for ANNs and that using simulated data helps in determining true differences.

The authors of [3] investigate the sensitivity of a general feedforward neural network by performing the study on multi-layer perceptron. The paper proposes a bottom-up approach to study the sensitivity of feed-forward neural network (multi-layer perceptron) to input and weight perturbations. They start by discussing the sensitivity of a single neuron and derive an analytical expression that expresses the relationship between a neuron's output error and its input and weight errors. The authors also use this expression to form an algorithm that allows to quantify the sensitivity of an MLP and analyses the relationship between the MLPs output error and input error as well as its structural configuration. The results obtained by the expression formulated in this paper treats both input and weights as statistical variables.

As presented in [4], a new tool that addresses the issue of Global Sensitivity Analysis of Model Output (SAMO) to decompose the output variance with respect to each input, as sensitivity indexes can be applied when the training stage has finished. Though the local sensitivity approach calculates derivative for one variable at a time close to a point, in comparison global approach relies on whole parameter space. It considers the interaction effects of an input with all other available variables. The authors put forward a method for variance-like measure, Fourier Amplitude Sensitivity Test (FAST) to perform a variance decomposition of multivariate input space into one-dependent variable s. All the inputs can thus vary simultaneously with their own frequency. The output can thus be thought of as a sum of these input signals oscillating with different frequencies. The authors even extended the FAST algorithm to compute the total sensitivity index and make choice of frequencies less sensitive. The searched variance of the higher spectrum can be calculated as a difference of the total variance and the low spectrum variance. The algorithm can be applied to any form of function to find the total sensitivity index. Variables having very less impact on the output variance can then be removed and re-training can be done to find the effect of pruning the less important variables. Finally, they evaluate EFAST against the well-defined algorithms to validate the results and concluded with EFAST providing much better results for global sensitivity analysis.

Another approach to determine sensitivity of the input variables as in [5] focuses on a new algorithm called Sensitivity-Based Linear Learning Method (SBLLM), which is an algorithm that uses sensitivity formulas to calculate weights of each neuron in a two-layer feedforward neural network. The five-step algorithm includes learning weights of layer 1 and 2 and the associated sensitivities, evaluating the sum of squared errors, checking for convergence, checking improvement of cost function and finally updating the outputs. If convergence hasn't been reached, the improvement of the Q function compared to the last Q function is checked and then weights get updated accordingly with repetition of this process. This new algorithm is straightforward and has O(n^2) runtime.

## III. DATASET

The dataset used for this project is CIFAR10 image dataset which consists of 60,000 images classified exclusively into ten classes. A total of 50,000 images have been apportioned for training purpose while the remaining 10,000 for testing the trained network. Image classification is a vital part of digital image analysis, primarily applied in facial/image recognition, autonomous driving, and image stitching.

Since CIFAR10 is a well-understood dataset and widely used for benchmarking computer vision algorithm in the field of machine learning, conducting sensitivity trials on it was deemed useful for the purpose of this research.

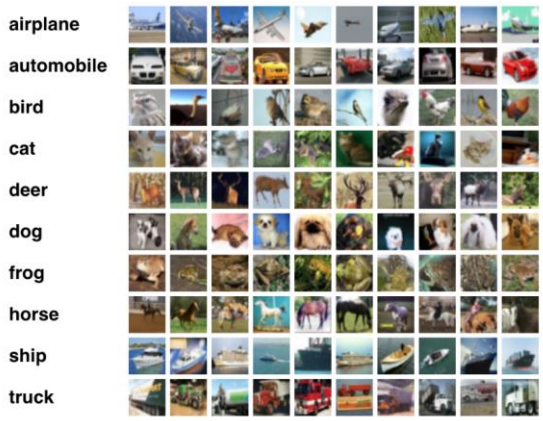


Fig.1. CIFAR10 dataset

## IV. IMPLEMENTATION

The baseline model for CIFAR10 dataset was trained on a DenseNet 121 network up to a decent accuracy of 92.18% with simple pre-processing applied to training data. The pre-processing step on the original data included normalization of the three RGB channels, random cropping and random horizontal flipping of images. The model was trained for 200 epochs with variable learning rate which ranged from 0.01 – 0.0001. Cross Entropy Loss criteria was used on training data fed to an SGD optimizer in batch sizes of 64.

TABLE I. DENSENET MODEL'S HYPERPARAMETERS

| Batch Size | Epochs | Loss | Optimizer |
|---|---|---|---|
| 64 | 200 | Cross Entropy | SGD Optimizer |

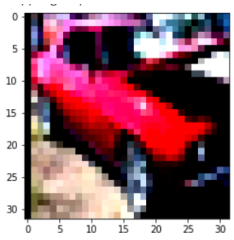


Fig.2. Sample training image with basic pre-processing applied

Densely connected networks simplify connectivity pattern between layers introduced in other architectures. Research in the field of Convolutional Networks has shown that for substantially deeper networks, more accurate, and efficient training can be achieved if the network consists of shorter connections between layers close to the input and those close to the output. DenseNet can alleviate the vanishing -gradient problem, strengthen feature propagation, encourage feature reuse and substantially reduced the number of parameters. DenseNet 121 used in our analysis achieves the best results with comparatively lesser number of

parameters. Dense nets do not sum the output feature maps of the layer with the incoming feature maps but rather concatenates them. The feature map is the information of the system. Additionally, the compression allows model compactness which reduces the number of feature maps at transition layer. DenseNet scale naturally to hundreds of layers, while exhibiting no optimization difficulties and hence was our go to approach in selecting this model architecture.

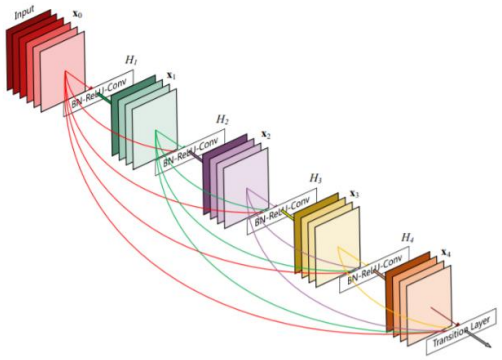


Fig.3. Densely Connected Convolution Networks.[7]

## V. RESULT FOR VARIOUS INPUT PERTURBATIONS

The perturbations were made to the input of a trained neural network to see how it impacts the output accuracy and the trend along the confusion matrix. These perturbations were applied as certain preprocessing steps to the input data in order to study the impact of each variation individually. This could be helpful to analyze how different perturbations can impact the output of an otherwise trained neural network under different real-world circumstances. This could include noise in the signal which could result in the distortion of the pixels in certain way or different lighting issues which could result in different brightness and contrast problems.

### *A. Salt and Pepper noise*

Salt and pepper noise are generated when there are defects in the CCD elements, flecks of dust inside the camera, and during image transmission. Impulse noise appears when some of the pixels in the image are replaced by outliers while the rest remain unchanged. Outliers can have a fixed minimum or maximum gray-scale value or may vary within that range. In order to test our model's sensitivity, we inculcated salt and pepper noise using skikit-image library. We randomly replaced pixels with 0.009 proportion as can be seen in Fig.4. The accuracy dropped to 79.46% resulting from few pixels being replaced by noise in the input data.

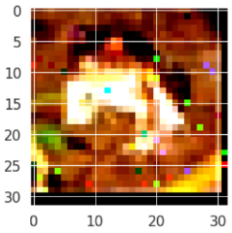

Fig.4. Salt and Pepper noise in input data

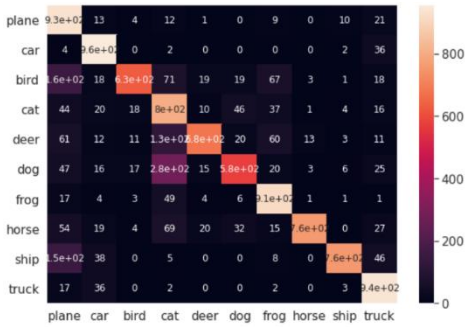


Fig.5. Confusion matrix for Salt and Pepper noise in input data

### *B. Contrast*

In visual perception of the real world, contrast is determined by the difference in the color and brightness of the object and other objects within the same field of view. We multiplied the intensity of each image with a constant number (0.65) to scale down the intensity and then added 25 to shift the intensity up on the scale and recover any lost information. We do this to reduce the contrast values of the pixels. We reduced the contrast of the images to test the response of our model but the decrease in accuracy was minor as can be seen from Table III.

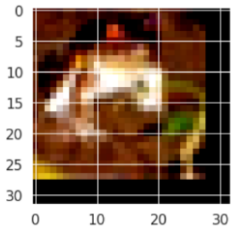

Fig.6. Reduced Contrast in input data

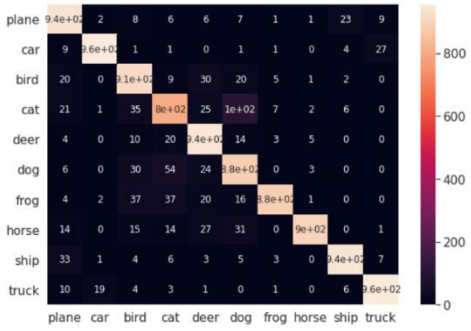


Fig.7. Confusion matrix for reduced contrast in input data

### C. Sharpeness

Image sharpening refers to any enhancement technique that highlights edges and fine details in an image. We convoluted all the pixels in the image with a specific filter to enhance the edges and find details of the image. The network responded with a substantial decrease in accuracy as in Table III as we tried to suppress the surrounding information and increased the weights on the center pixel. We used the below mentioned kernel to sharpen the image:

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix}$$

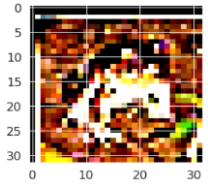

Fig.8. Image sharpening

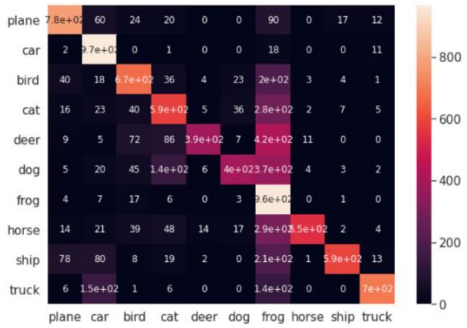


Fig.9. Confusion matrix for image sharpening

### D. Color Jitter

The Hue, Saturation, and Brightness values are the aspect of color in RGB channels. These terms are most often used in reference to the color of each pixel in a cathode ray tube ( CRT ) display. Saturation is an expression for the relative bandwidth of the visible output from a light source. Brightness is a relative expression of the intensity of the energy output of a visible light source. Hue is the pigment of the color. We change the values of hue, saturation, brightness and contrast to specific ranges and observe the differences in the predictions made by the model.

TABLE II. HSB AND CONTRAST RANGES FOR EXPERIMENTS

| Name | Brightness | Contrast | Saturation | Hue |
|---|---|---|---|---|
| Color Jitter (I) | 0.6,1.4 | 0.6,1.4 | 0.6,1.4 | 0.1 |
| Color Jitter (II) | 0,0.3 | 0,0.3 | 0,0.3 | 0.1 |
| Color Jitter (II) | 1.4,2 | 1.4,2 | 1.4,2 | 0.1 |

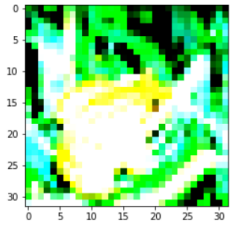

Fig.10. Changed HSB values in input data


Fig.11. Confusion matrix for HSB variation

### E. Gaussian Blur

In image processing, a Gaussian blur (also known as Gaussian smoothing) is the result of blurring an image by a Gaussian function. It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, distinctly different from the bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination. We used a kernel size of 3x3 with default gaussian scale. Blurring the image considerably reduced the accuracy to 43.58% as most of the edge information gets removed.

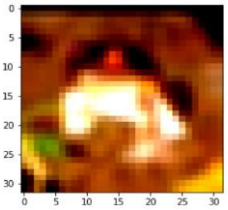

Fig.12. Gaussian blurring in input data


Fig.13. Confusion matrix gaussian blurring in input data

### F. *Sobel Filter*

The Sobel operator is used in image processing and computer vision, particularly within edge detection algorithms where it creates an image emphasizing edges. We used a 3x3 kernel size to extract only the edge information and were still able to get 56.05% accuracy with most of the background and texture information already lost as a part of the process.

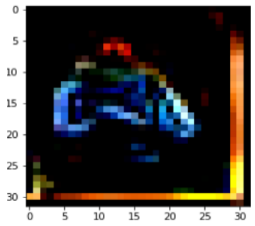

Fig.14. Sobel operator on input data


Fig.15. Confusion matrix for sobel operation on input data

The accuracies of all our experiments are summarized in the below mentioned table:

TABLE III. INPUT PERTURBATION AND OBSERVED ACCURACIES

| Input Perturbation | Accuracy |
|---|---|
| Baseline | 92.18% |
| Salt and Pepper | 79.46% |
| Contrast | 91.08% |
| Sharpening | 65.92% |
| Color Jitter (I) | 88.64% |
| Color Jitter (II) | 9.66% |
| Color Jitter (III) | 58.38% |
| Gaussian Blur (I) | 48.96% |
| Gaussian Blur (II) | 43.58% |
| Sobel Filter | 56.05% |

### G. *Occlusion*

Occlusion can further lead to change in output prediction value but how the output gets affected remains unknown. Different portions of the image when occluded can change the predicted outcome depending on which portion of the image gets occluded. We performed an experiment by occluding different portions with a window size of 8x8 on a 32x32 image and shifted the window by 8 pixels successively to see how different portions can change the predictions in the form of confusion matrix. For different classes of the dataset, the output gets affected differently.

Fig.16. Occlusion of the objects in images

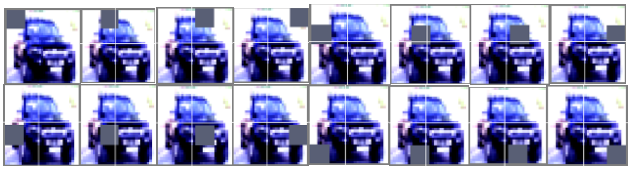

Fig.17. Sequence of occlusions on the image dataset

As can be seen from Fig.17., we moved the occlusion window with change in prediction values. Since background in the images do not produce any useful information regarding the object, occluding the background gave no change in the prediction values. However, if the occluded portions lie adjacent to an important portion of the image such as the face of dog, it can produce different results with some probability depending on the pose of the object. We recorded that the network gets confused between dogs and cats on occluding the face information, deer and horse depending on if we cover the antlers of a deer, truck and a car if we occlude the front of the car, plane and a ship when the wing of the plane gets occluded. From the observations, it was evident that certain object parts contributed more towards the output prediction.

## VI. DISCUSSION

From different perturbations and their results, it is evident that the network is more sensitive to certain information and can behaves indifferently to rest of it. Information like contrast and color jitter (I) have a minor impact on the output prediction whereas features like edge information or changing brightness, saturation and hue - Color Jitter (II) greatly affect our model's predictions making these features more sensitive for our model. Interpreting the gaussian blurring and sobel filter for edge extraction clearly reveal that these form complementary pair.

Extracting only edge information, while suppressing the rest of it including texture, background and surface gave an accuracy of more than 50% while taking away the edge information as in blurring gave less than 50% accuracy. This seems logical from Laplacian recovery concept where the original image can be thought of as a sum of its gaussian scaled image and the Laplacian of the image. Though we used sobel for edge extraction as compared to Laplacian operator, the effect was somewhat similar. On the other hand, sharpening of an image, images with certain noise (salt and pepper) have a moderate effect on the accuracies of our model. We also noticed that occluding a certain portion of image can lead to more incorrect predictions as compared to other portions. This again highlights the fact that certain features are more important than others. Occluding the background rarely changed the prediction unless in some cases the occluding portions served as an extension to some important set of pixels which captured useful information. The study gives useful insight on what features are more sensitive and should be taken care of while training a network. These features like edge information, surface and texture information and color jitter effect can play major role in deciding the accuracy on test case scenarios of the real world when we try to train the network on digital images. The feature space can thus be reduced to the more sensitive information. For example, we can retrain the network with even less background information and even reduce the intensity resolution of the image as this information proved less sensitive to the output predictions.

## VII. Bitmap Visualization

The weights learnt by a convolutional neural network are in the form of a convolutional kernel and neural networks can have a large number of such kernels depending on the number of color channels in the data. We trained the network on all the three color channels and the weights are learnt for each of the channel independently. The bitmap visualization of the convolutional layers can reveal if a layer is learning anything useful from the input data. Usually, first layers provide most of the relevant and interpretable information compared to the layers deeper inside the network. A well-trained network usually has a smooth filter patterns without any noise as described in [6]. Noisy patterns indicate that the network hasn't been trained for long enough, or possibly a very low regularization strength that may have led to overfitting.

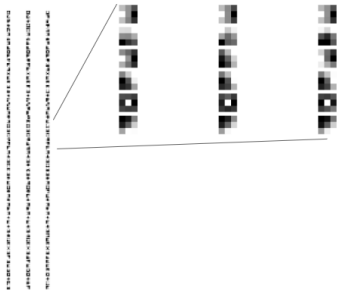

Fig.18. Bitmap visualization of the first convolutional layer

As can be observed from Fig.18., all the three channels learnt nearly similar patterns which clearly shows that one can expect to train the network on a single channel with nearly similar results as with the three channels.

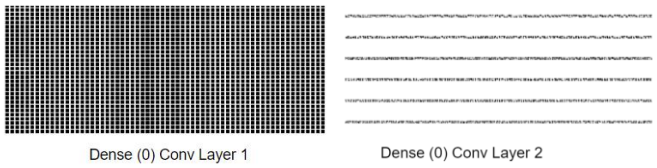


Fig.19. Bitmap visualization of first two layers in Dense 1(0).

From Fig.19. it is clear that the first layer is not learning anything from the input data whereas the second layer is learning something useful from the training data. As we go deeper into the dense layers, it is possible that we can extract more useful patterns and information that we can relate to different features of the input data.

## VIII. Conclusion

The implementation of the baseline model and developing different pre-processing perturbations on the test data helped us to complete a detailed sensitivity analysis on CIFAR10 dataset. We found that features like contrast do not have a very huge impact and can be considered as un-important inputs. However, features like edge information, occlusion of certain portions and sharpening greatly impact the accuracies. The analysis presented in our work can be extended to other datasets with similar visualization to extract the useful features.

## IX. Future Improvements

Keeping in mind the time allotted for this task and the strength of the team, we believe that a decent result was obtained and the assigned tasks were completed in time. For this study we chose CIFAR 10 dataset, however, we believe that our approach can be experimented on larger and more complex datasets in order to get more insights on the impact of inputs over the outputs of models. The bitmap visualization was done only for the initial layer but in future we would like to interpret and draw results even from the deeper layers to get a more in-depth understanding of the learning process. This could certainly help prune the less relevant features and thus input modelling can be done at various levels of fidelity in simulation environments.

## Acknowledgment

We would like to extend our gratitude to Professor Karthik Dantu for giving us an opportunity to work on this project. We would also like to thank our Teaching Assistant Charuvahan Adhivarahan for helping us in completing this project and guiding us throughout the term. Through this project we were able to learn concepts of Robot Learning and implement them practically allowing us to learn and continue our interests in the field.

## REFERENCES


[1] Julian D. Olden, Michael K. Joy, Russell G. Death, "An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data," 2004 Elsevier B.V

[2] Muriel Gevrey, Ioannis Dimopoulos, Sovan Lek, " Review and comparison of methods to study the contribution of variables in artificial neural network models," 2002 Published by Elsevier Science B.V.

[3] Xiaoqin Zeng and D. S. Yeung, "Sensitivity analysis of multilayer perceptron to input and weight perturbations," in *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1358-1366, Nov. 2001, doi: 10.1109/72.963772.

[4] E. Fock, "Global Sensitivity Analysis Approach for Input Selection and System Identification Purposes—A New Framework for Feedforward Neural Networks," in IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 8, pp. 1484-1495, Aug. 2014, doi: 10.1109/TNNLS.2013.2294437.

[5] Enrique Castillo, Bertha Guijarro-Berdiñas, Oscar Fontenla-Romero, Amparo Alonso-Betanzos, "A Very Fast Learning Method for Neural Networks Based on Sensitivity Analysis" Journal of Machine Learning Research 7 (2006) 1159–1182

[6] CS231 n: Convolutional Neural Networks for Visual Recognition. https://cs231n.github.io/

[7] Densely Connected Convolutional Networks, Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger


## INDIVIDUAL CONTRIBUTION

| **Team Member** | **Task** |
|---|---|
| Sahil Kapahi | Research, code implementation, analysis, reports and presentation |
| Snigdha Gupta | Research, code implementation, analysis, reports and presentation |
| Nikhilesh Gupta | Research, presentation |

Github link to the code:
https://github.com/sahilkap07