PES
UNIVERSITY

Dissertation on

"Interpretability in Neural Networks"

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

Submitted by:

Sai Rohit S 01FB15ECS255
Sai Sasank Y 01FB15ECS256

Under the guidance of

Internal Guide
V R Badri Prasad
Assistant Professor,
PES University.

January – May 2019

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Interpretability in Neural Networks'

is a bonafide work carried out by

Sai Rohit S	**01FB15ECS255**
Sai Sasank Y	**01FB15ECS256**

In partial fulfilment for the completion of eighth semester project work in the Program of Study Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2019 – May. 2019. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 8th semester academic requirements in respect of project work.

Signature	Signature	Signature
V R Badri Prasad	Dr. Shylaja S S	Dr. B K Keshavan
Assistant Professor	Chairperson	Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. ______________________	______________________
2. ______________________	______________________

DECLARATION

I hereby declare that the project entitled **Interpretability in Neural Networks** has been carried out by me under the guidance of Prof V R Badri Prasad, Associate Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University,Bengaluru** during the academic semester January - May 2019. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

01FB15ECS255	**Sai Rohit S**
01FB15ECS256	**Sai Sasank Y**

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof V R Badri Prasad, Department of Computer Science, PES University for his continuous guidance, assistance and encouragement throughout the development of this project.

I am grateful to the project coordinators Prof Preet Kanwal and Prof Sangeetha, for organising, managing and helping out with the entire process.

I take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. K.N.B. Murthy, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my parents.

ABSTRACT

When deployed for usage in the industry, machine learning models need to develop trust in the user so that he or she would accept the results of the model and use it. This would enhance faith as now the model would not be a black box. The results of the explanation based system must also be stable in the sense that there should not be a lot of change in the explanations for minor change in the input. Through this project, we wish to develop a self-explanation based architecture where the explanations are learnt through the training process along with the actual task in hand. We demonstrate the same for the task of Image Classification performed on the MNIST dataset. We argue this architecture is extendable to other modalities as well. This model is shown to be better than the posteriori based approaches in terms of the stability and robustness of the explanations. The goal of this project is to develop the same and compare it with the posteriori based explanations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
INTRODUCTION

1.1 Overview

In the trivial task of Image Classification, an image is taken as an input and a probability distribution over the set of output classes is given as an output. This output tells how the probability of the input image belonging to a particular output class. This problem has found application in a variety of scenarios.

For example, in the image below an image classification model takes a single image and assigns probabilities to 4 labels, *{cat, dog, hat, mug}*. As shown in the image, keep in mind that to a computer an image is represented as one large 3-dimensional array of numbers. In this example, the cat image is 248 pixels wide, 400 pixels tall, and has three color channels Red,Green,Blue (or RGB for short). Therefore, the image consists of 248 x 400 x 3 numbers, or a total of 297,600 numbers. Each number is an integer that ranges from 0 (black) to 255 (white). Our task is to turn this quarter of a million numbers into a single label, such as *"cat"*.

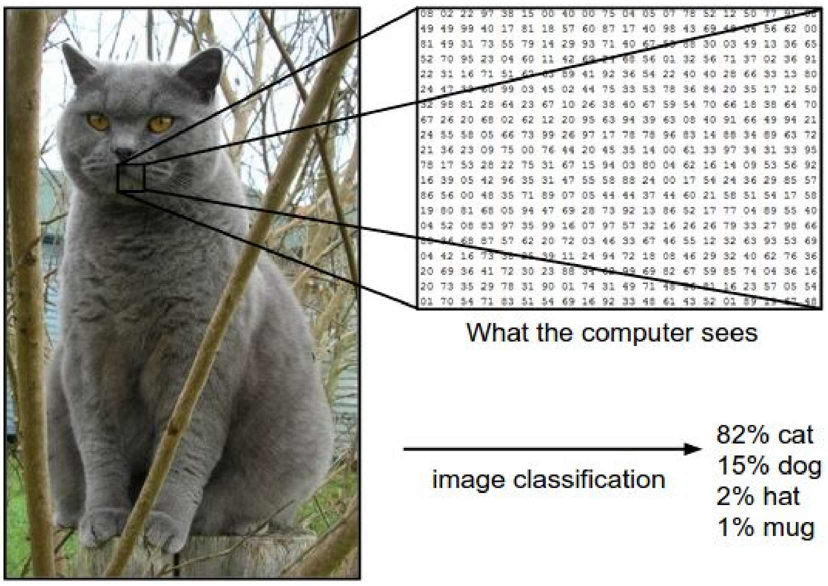
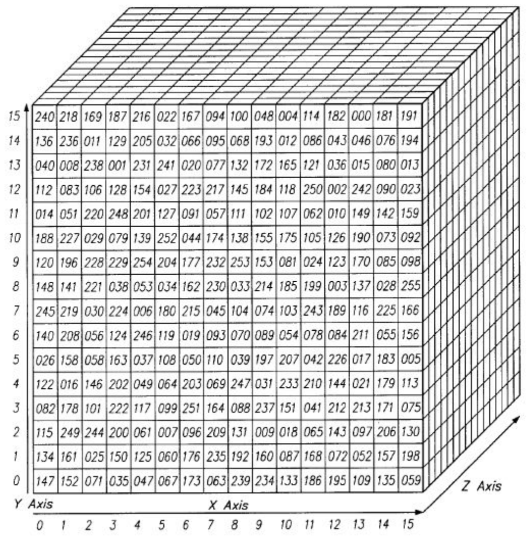


Fig 1.1 : Image Classification Example

The most common deep learning architecture used for this task is the Convolutional Neural Network [1].

Convolutional neural networks are deep artificial neural networks that are used primarily to classify images (e.g. name what they see), cluster them by similarity (photo search), and perform object recognition within scenes.
The results obtained by Convolutional networks in these tasks is one of the key factors that has contributed in the progress of Deep Learning as a field and also in the increase in the number of real life use cases of Deep Learning. Convolutional Neural Networks are also the driving force behind a lot of Computer Vision tasks. This has led way to solve multiple problems like autonomous driving vehicles.

As mentioned previously, the computer sees an image as a tensor of numbers. If we consider an image which has three colour scales, (RGB) it will be represented as a 3D matrix of numbers. A CNN takes this as input. Given below is an example of what the CNN sees as an input when we pass an image.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	240	218	169	187	216	022	167	094	100	048	004	114	182	000	181	191
14	136	236	011	129	205	032	066	095	068	193	012	086	043	046	076	194
13	040	008	238	001	231	241	020	077	132	172	165	121	036	015	080	013
12	112	083	106	128	154	027	223	217	145	184	118	250	002	242	090	023
11	014	051	220	248	201	127	091	057	111	102	107	062	010	149	142	159
10	188	227	029	079	139	252	044	174	138	155	175	105	126	190	073	092
9	120	196	228	229	254	204	177	232	253	153	081	024	123	170	085	098
8	148	141	221	038	053	034	162	230	033	214	185	199	003	137	028	255
7	245	219	030	224	006	180	215	045	104	074	103	243	189	116	225	166
6	140	208	056	124	246	119	019	093	070	089	054	078	084	211	055	156
5	026	158	058	163	037	108	050	110	039	197	207	042	226	017	183	005
4	122	016	146	202	049	064	203	069	247	031	233	210	144	021	179	113
3	082	178	101	222	117	099	251	164	088	237	151	041	212	213	171	075
2	115	249	244	200	061	007	096	209	131	009	018	065	143	097	206	130
1	134	161	025	150	125	060	176	235	192	160	087	168	072	052	157	198
0	147	152	071	035	047	067	173	063	239	234	133	186	195	109	135	059

Fig 1.2 : What a CNN sees as an Input when an Image is passed.

The basic operation performed by a CNN is called Convolution. From the Latin *convolvere*, “to convolve” means to roll together. For mathematical purposes, a convolution is the integral measuring how much two functions overlap as one passes over the other. Convolution can be thought of as a way of mixing two functions by multiplying them. One of the most primary reasons why this is needed is that it is not sufficient if we take each pixel in an image individually and try to perform our tasks. This is because of what we call spatial property in images. In an image a neighbourhood of pixels when combined or seen together depict a particular concept. This concept cannot be identified if we take each pixel separately. This means that each pixel in an image is somewhat related to the pixels in its neighbourhood. To preserve this, a CNN takes a square or rectangle set of pixels at once rather than considering each pixel.

Common Layers in a Convolutional Network : Each segment or a convolutional block usually has 3 layers in it. The first is the convolutional layer that performs the mentioned convolution on the input to the layer. The second one is usually a non linearity layer that adds the nonlinearity component to the function. The final one is usually the Pooling Layer.

There is a reason for adding each of the above mentioned layers after the convolutional layer. The concolutional operation is a linear one. When we try to solve real world problems using neural nets, a significant amount of them are not linear problems and thus cannot be solved by just using a linear function like convolution. For this purpose we introduce a nonlinearity into the network. In the past, nonlinear functions like tanh and sigmoid were used, but researchers found out that ReLU layers work far better because the network is able to train a lot faster (because of the computational efficiency) without making a significant difference to the accuracy. It also helps to alleviate the vanishing gradient problem, which is the issue where the lower layers of the network train very slowly because the gradient decreases exponentially through the layers.

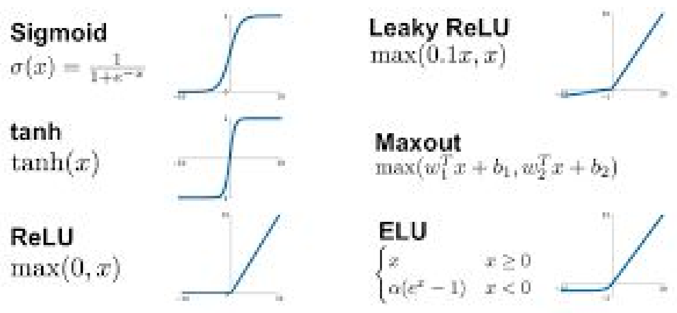


Figure 1.3 : Commonly used Activation Functions

The pooling layer, which is a downsampling technique serves two purposes. Firstly it reduces the number of network parameters that have to be learnt by gradually reducing the size of the input as we progress through the network. This helps us save the computational resources like time and cost. Secondly, it helps us avoid overfitting.

Some commonly used pooling techniques are Max Pooling, Average Pooling and L2-norm Pooling.

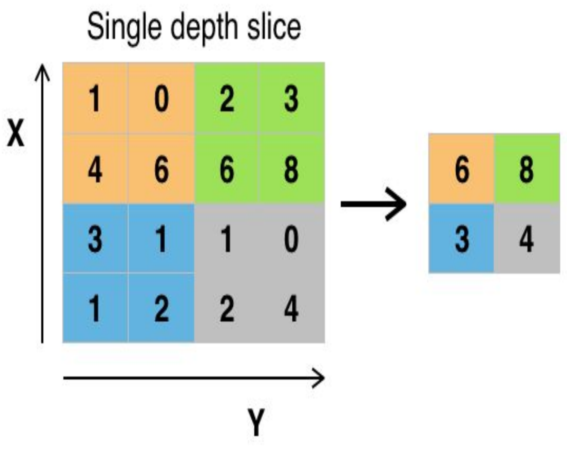


Figure 1.4 : Max Pooling Technique.

A typical convolutional network consists of a set of these blocks followed by a dense layer. The dense layer is used to convert the 2D input to a 1D output. It is also called the Flatten layer.

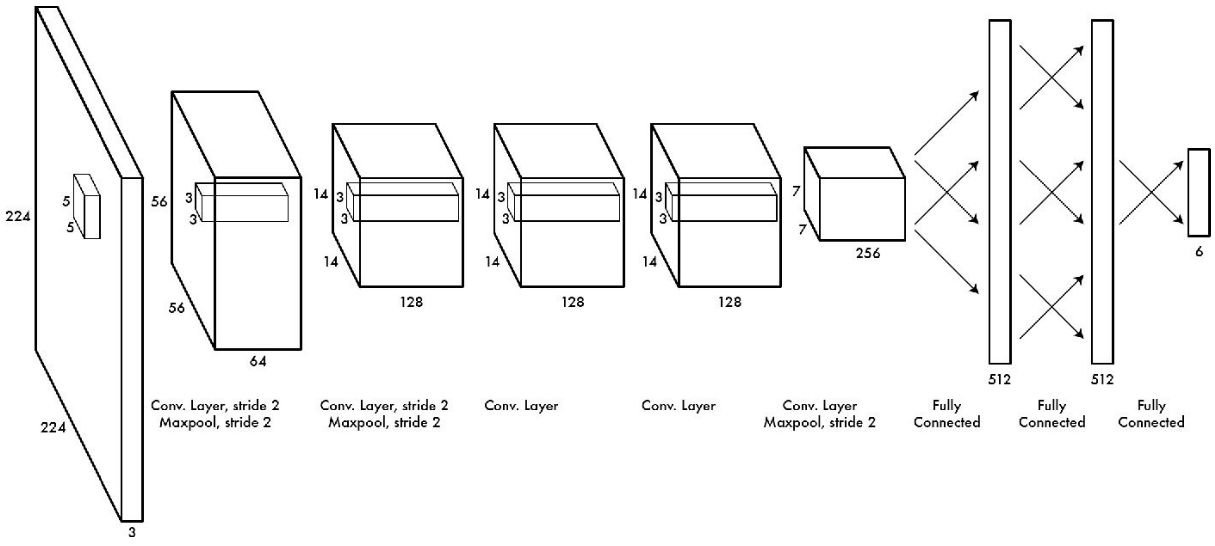


Fig 1.5 : Vanilla CNN

A Vanilla CNN takes an image as an input and generates the corresponding probability distribution over the output classes as output. In this model, we only obtain the probability of the image belonging to each output class and not why the model predicted it that way. That is that, for a person using this model, it looks like a black box which takes an output and gives an output. He or she does not understand the working of the model. This is where explanations or interpretations for the output come into the picture.

The need for an explanation based machine learning system arises from the fact that to solve real world problems, a single metric like performance accuracy is not sufficient. Also in some problems there are no serious consequences of the machine learning algorithm failing or going wrong. These can be algorithms such as a Recommendation System for movies or TV shows. Not knowing why the model works the way it does is not a serious issue in cases like

this. However in cases like medical diagnosis, the knowledge of why the model works the way it does can help in understanding the working of the model.
Also when a machine learning model is deployed into an industry where the professional using it is not aware about the workings of the model or the algorithm, interpreting the results can go a long away in making him or her obtain a better trust in the deployed algorithm. Given that these interpretations can be provided in terms of the domain of the input to the model and the corresponding industry professional has sufficient domain knowledge, it will be beneficial to provide an explanation along with the model output.

Defining the explanations for a machine learning model can be very abstract. The definitions may vary and also nature of the interpretations provided for the output can also be different. One definition which is considered as the base meaning for the interpretability of a model is that **Interpretability is the degree to which a human can understand the cause of a decision** [2]**.** Using this as basis, a lot of different variants of explanations can be provided. Another way to establish the different explanations can be based on the time at which the explanations are learnt. The explanations can be learnt as a part of the training phase or can be independent. By independent we mean that the explanations are learnt after the entire training process is completed. We also need to define metrics to evaluate how good an interpretable model is. Given that there are multiple different ways in which explanations can be provided, there has to be a common metric irrespective of how these explanations are.

As mentioned, on the basis of the time at which the explanations are learnt we can divide the explanation frameworks into two types. The first if Posteriori explanations where the explanation model learning does not interfere with the training process. The model for the task at hand is originally trained as before and then the explanation models are built on top of these models. Examples for this type of frameworks are LIME[3] and SHAP[4]. The second type of models are Self Explanation Models[5] where the training process and the explanation learning happen simultaneously. In these models, the architecture is varied in such a way that training and explanation learning are done in parallel. Because of this, these models are more complex and involve some clear cut tasks to be performed beforehand such as the definition for explanations.

Considering the task of image classification, there can be multiple ways to back a decision made by a machine learning model. Work done in this field till now uses heat maps over the regions of the image which affect the output positively as a method to explain the image. This means that the pixels in an image are divided or scored based on how they affect the output prediction. A lot of methods till date use this as the definition for the explanation for an image. LIME and SHAP based explanations are the most common ones in this category.

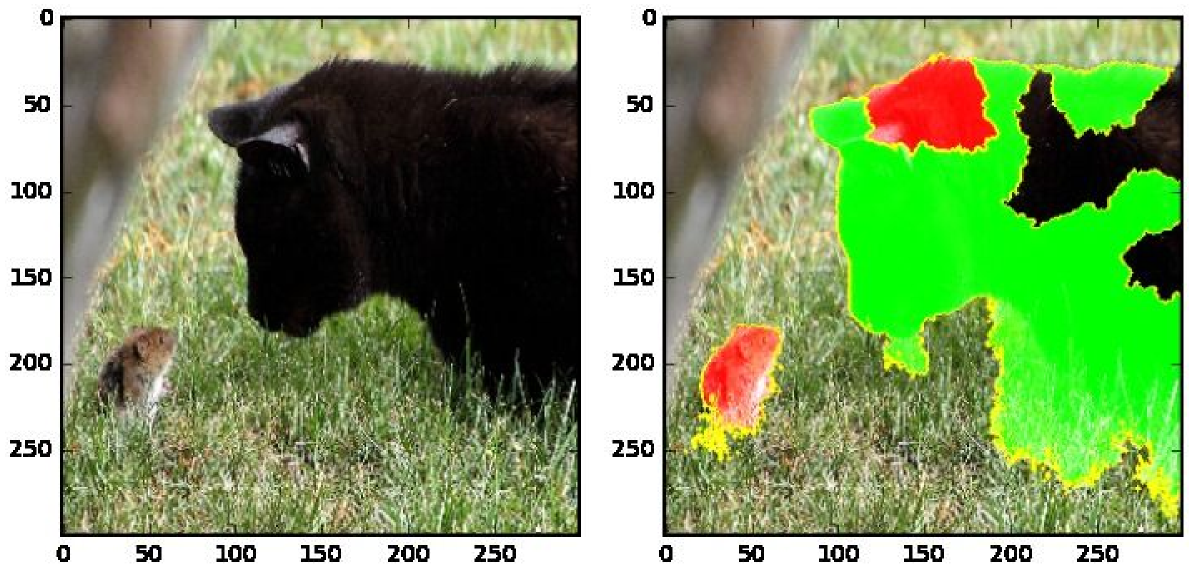


Fig 1.6 : An image of a cat passed to a CNN and the corresponding heat map based explanation. Red indicates negative regions and green the positive ones.

The above figure is an example of the heat map based technique. As mentioned before we need to have a metric to evaluate how good an interpretable model is. The metric defined here is called stability. Stability is a measure for how similar or consistent the explanations are for similar or neighbouring samples. To understand this better, let us consider an example. Say we have an image of a cat which the model classifies correctly and its corresponding explanation. When a tiny amount of noise is added to the image such that the classification output does not change, stability measures how different the explanations are for the original image and the noisy image. In fact we measure stability as the ratio of change in the explanation to the change in the original input image.

The issue with the heat map based approaches such as LIME and SHAP was that the explanations although understandable were not stable. This resulted in a need for a more stable explanation framework. David Alvarez-Melis and Tommi S. Jaakkola, through their paper “Towards Robust Interpretability with Self-Explaining Neural Networks” came up with a self explanation based framework where the explanation model was learnt along with the training model. Their definition of explanation was encoding an image into concepts and indicating how important each concept was in the final classification.

1.2 Scope

This project will pertain to proving that the current heat map based posteriori methods are not robust enough for the task of image classification and then using the concepts of self explanation, we wish to come up with a new definition of explanation and prove that it is robust and at the same time does not lead to a compromise on the classification accuracy.

In doing so, we wish to use the MNIST dataset for the same. MNIST is a baseline dataset for all image processing tasks and has been used as a standard in the papers mentioned also. Thus we will be using the same and testing our architecture.

To calculate the robustness of the interpretability model, we will be using stability as the metric. Stability tells us how different the explanations are for two sample which lie in the same neighbourhood or which are very similar to each other.

1.3 Objective

The aim of this project is to prove that the heat map based explanations are understandable but not stable enough to be used. We wish to come up with a self explanation based framework and a new definition for explanation which is :

1. Understandable and logically accurate to be classified as an explanation.
2. More stable when compared to the heat map based explanations.

We define explanations for an output as the understanding of how well the model learnt to classify that particular category and where it learnt classify as cat as a cat. Out intuition is that, if the model has learnt the right way, it can be trusted with respect to the decision it has made. Checking if the model has learnt the right way can be a complex problem when we consider typical architectures. For this reason follow a self explanation based architecture which is followed by a Unsupervised Clustering based grounding technique.

1.4 Outcomes

Firstly, we wish to prove that the current methods like LIME and SHAP are not robust enough to be used in real time applications. Then we wish to implement the self explanation and prove that the results obtained from these are robust but are not easily understandable by a non machine learning professional. Then we wish to implement our Clustering based self explanation architecture and prove that it is both robust and easily understandable by a lehman.

For each test sample, we will be providing a set of images from which the model learnt to classify this particular output class. These we hope will be similar to the corresponding test image passed and thus will serve as a proof that the model learnt the right way. By doing so we wish to prove that the model can be trusted in the decisions it makes and thus is reliable to be in used in real world problems.

Given that the interpretation of self explanations can be complex, we wish to develop a explanation report for each test sample that starts with the input image and explains the entire setup including the representations of the concepts, their corresponding values for the input image and their influence scores followed by the clustering based grounding results.

Chapter 2

RESEARCH BACKGROUND

2.1 Literature Survey

The problem of obtaining explanations for a model's predictions can be framed in multiple ways. At the most basic level, they can be divided into two categories based on the time at which these explanations are learnt. These are posteriori models and Self Explaining Models. Posteriori Models are the type of models where the explanation model is built on top of already developed machine learning models. In these type of models, the training phase and the explanation learning phase are completely separate.

Commonly used Posteriori Models are LIME [3] and SHAP [4]. The second type of models called self explaining models are novel machine learning architectures where the explanations learning procedure is inherent to the training phase [5].

2.2 Local Interpretable Model Agnostic Explanations - LIME

2.2.1 Introduction

LIME is a result of some of the early efforts made to improve interpretability in black box models similar to neural networks. The key feature of LIME is that it is model agnostic and hence can be used on any model to derive explanations for samples.

The intuition behind LIME is quite straightforward. LIME uses the concept of surrogate models - models that are trained to approximate the predictions of another model, to interpret the model at hand. These surrogate models are trained locally, i.e. on individual samples for which we seek explanations.

For a given black box model and a sample you want an explanation for, LIME trains a local surrogate model in the neighborhood of this sample with an objective of imitating the black box model. We then use the local surrogate model to interpret the black box model. So consequently, the local surrogate model needs to be interpretable by itself. Examples for interpretable models are linear models, decision trees and so on.

An explanation model for an example *x* in LIME is defined as follows:

$$explanation(x) = argmin_{g \in G} L(f,g,\pi_x) + \Omega(g)$$

where *L* is a loss function that measures how similar the predictions of the original model *f* and the local surrogate model *g* are. Π_x is a proximity parameter that defines the size of the neighborhood around *x* to train the local model *g*. $\Omega(g)$ is used to quantify the complexity of the local surrogate model *g*. Finally, we try to pick *g* from a family of local surrogate models *G* such that it minimizes the expression.

1. Choose a sample/example for which you want to generate an explanation.
2. Perturb this sample and get blackbox predictions for all the perturbed samples.
3. Samples in this dataset are weighed against their proximity from the chosen sample.
4. Train a interpretable weighted model on this dataset.
5. Explain the black-box prediction using the trained interpretable model.

Perturbing the input depends on the modality which is concerned with the problem we are trying to solve.

In image modality, we look at different approaches to input perturbation such as perturbing individual pixels, or perturbing superpixels, a set of interconnected pixels.

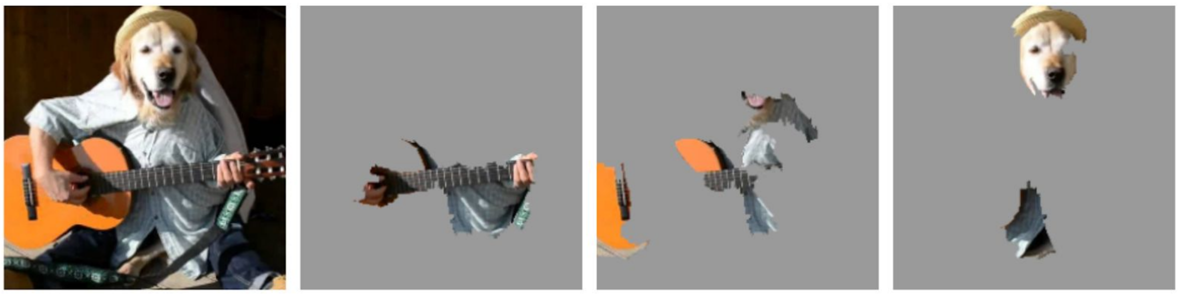


Fig 2.1 Perturbing images to form a dataset to train the local surrogate model using superpixel perturbation.

In other problems like text classification, we are dealing with text modality. We can perturb the input by removing words from the text at random.

2.2.2 Examples

Let us look at explanations generated by LIME for image and text classification problems.

In fig 3.2 we have LIME explanation for cats and dogs classifier, where the explanation is through a heat map indicating the influence of different regions of the image on the model's prediction. Here, green indicates positive influence and red indicates negative influence.

In fig 3.3 we can see LIME explanations for a text classifier with classes “atheism” and “christian”. The explanations basically assigns scores to each word in the text and lists k words with top k scores influencing the prediction.

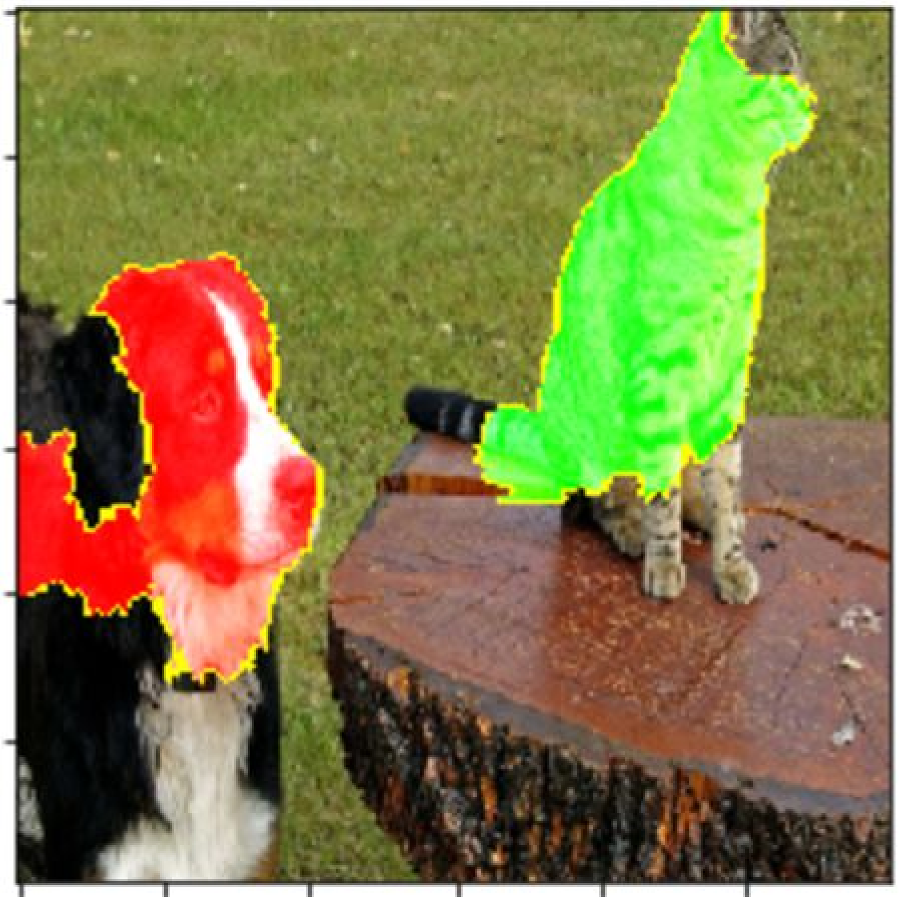
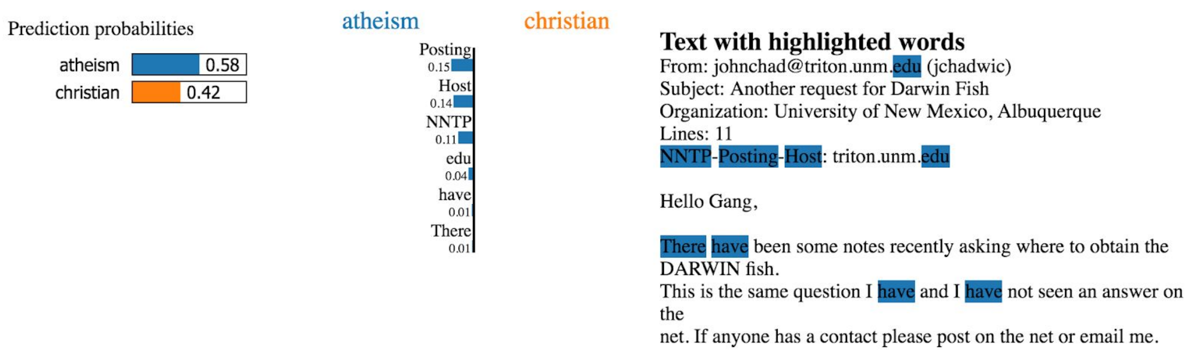


Fig 2.2 Explaining the prediction "cat" with heatmaps, where red indicates negative influence on the prediction and green indicates positive influence on the prediction.



Text with highlighted words
From: johnchad@triton.unm.edu (jchadwic)
Subject: Another request for Darwin Fish
Organization: University of New Mexico, Albuquerque
Lines: 11
NNTP-Posting-Host: triton.unm.edu

Hello Gang,

There have been some notes recently asking where to obtain the
DARWIN fish.
This is the same question I have and I have not seen an answer on
the
net. If anyone has a contact please post on the net or email me.

Fig 2.3 Explaining a text instance using LIME, where words in the text are attached with scores based on their influence on the model's prediction.

2.2.3 Advantages

LIME has several advantages. We have established that it is model agnostic, meaning the underlying model can be anything and its internals are not required by LIME. You can choose to use a linear model or a decision tree as the local surrogate, whichever helps making interpretations easier and underneath that, use complex models such as neural networks to make predictions.

LIME also generates human friendly explanations and is often a strong choice for this reason alone. LIME is adaptable to multiple domains like images, text and tabular data. LIME has interfaces/APIs in Python and R making it popular preference for practitioners.

2.2.4 Disadvantages

In LIME, the way we define neighborhood for an instance is left to the user. It is in fact not well defined and is actually an issue that needs to be handled based on the problem at hand. While sampling around a data point, it is possible to sample unlikely data points as LIME ignores the possible correlations between features in the input. Complexity of local surrogate models is to be defined upfront and is actually a tradeoff between fidelity of the surrogate model to the black-box model and interpretability of the surrogate model. LIME does not generate robust explanations meaning for insignificant changes in the input the explanations differ a lot and consequently we are required to be very critical of the explanations.

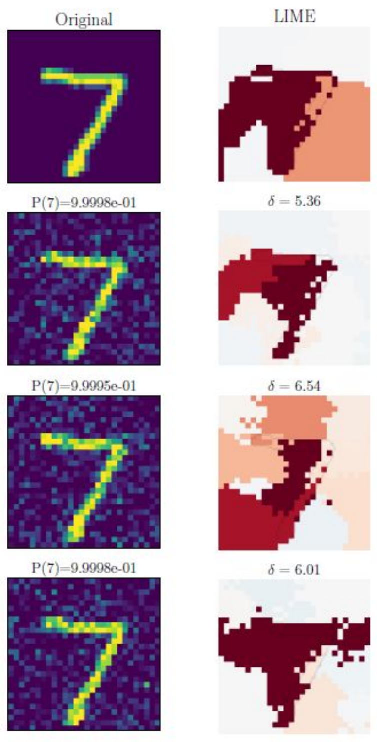


Fig 2.4 : Depicting the non robustness of LIME

2.3 Shapley Values

2.3.1 Introduction

Shapley values have their origin in coalitional game theory. Each feature value of an instance is assumed to be a player in a game where the prediction is pay out. Shapley values dictates the distribution of the payout to the players. In this game, players are assigned payouts which depends on their contributions to the total payout. The "game" here refers to the task of prediction on a single sample/instance of the dataset. "Total payout" is the difference between actual prediction for this instance and average prediction for all the instances. The "players" are the features used to make the prediction.

To calculate Shapley values for a single feature value, we calculate its average marginal contribution towards all possible coalitions of the feature values. It is crucial to realize that the number of coalitions grow exponentially relative to the number of feature values. To avoid exponential complexity, we can take a subset of coalitions and make computations more manageable using Monte Carlo sampling.

The way we interpret a Shapley value for feature value k is as follows. The value of the k-th feature contributed ϕ_k to the particular instance's prediction compared to the average prediction for the dataset.

2.3.2 Examples

Consider a model for prediction of probability of cervical cancer. It contains features such as STDs, Diagnosed STDs, No. of pregnancies etc. Let us look at shapley explanations for an instance with feature values as mentioned in fig 3.3. We see a plot of features vs features' contribution (shapley values) to the actual prediction that the person has cancer with 0.43 probability. We can observe that number of diagnosed STDs has the highest Shapley value meaning it has contributed the most to the actual prediction. Consequently, the other features have respective shapley values indicating their contributions to the actual prediction relative to the average prediction.

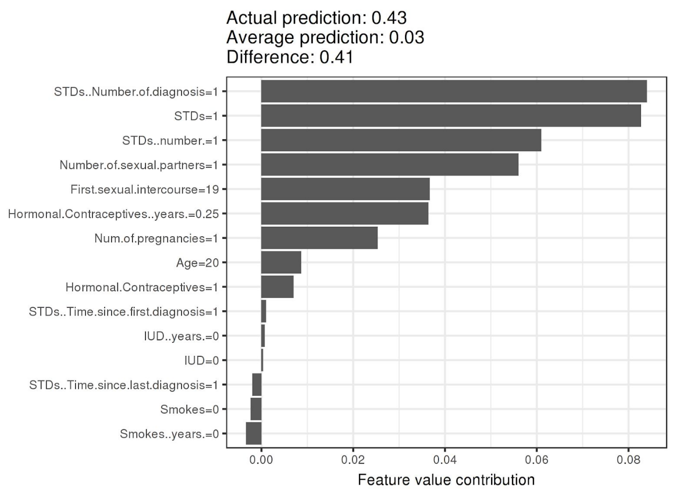


Fig 2.5 Shapley values for features of a cervical cancer dataset.

2.3.3 Advantages

Shapley values are fairly calculated as the total payout or the difference between actual and average prediction is efficiently distributed among the features. LIME does not guarantee such fairness and efficiency. Shapley values are based on a solid mathematical theory giving good foundations to its explanations. This makes it preferable in situations where the laws require explainability. Shapley values allow contrastive explanations unlike LIME. Similar to LIME, Shapley values are model agnostic. The shapley values are easily interpretable and do not require technical rigour. Shapley values is available as a package in R and an alternative SHAP is available in Python.

2.3.4 Disadvantages

Computing shapley values require a lot of time. In most real world situations, only approximate solutions are feasible due to exponential nature of the computation time. The alternatives involve computing on a subset of all possible coalitions, essentially making a trade off between accuracy of shapley values and computation time. Unlike LIME, Shapley assigns values to each feature without any explanation model which restricts us to make abstract understandings of the features' influence. Similar to LIME, while sampling data it is possible to encounter unrealistic data instances ignoring correlations between features.

2.4 Self Explaining Neural Networks

2.4.1 Introduction

The methods discussed till now are the ones which are posteriori. In these type of methods, interpretability does not play a role in the learning process. Self explanations based neural networks are the ones where this is incorporated. The process of learning the interpretations are incorporated into the training process in these type of models.

The other difference between these and the previous models is that the ones like LIME and SHAP are operate at a unit sample level. They are individually learnt for each sample. For example in the case of LIME, the local model is learnt for each sample whenever it is needed. Self explanation based models on the other hand are global in the sense that the same model is used for the interpretations for all the test samples.

Discussing the issues with those models leads us to come up with a metric for interpretability. This metric should be able to define how robust the prediction model is. The metric used for this purpose is called Stability. Stability lets us know how consistent the interpretations of the model are for samples which lie in the same neighbourhood or samples which are semantically similar. Experiments have proven that the posteriori models like LIME and SHAP are not stable in this sense. That is, for a small change in the input, the explanations change by a large margin thus making them non robust. [6]

This leads to the concept of self explanation. Given that we can enforce a training and learning loss, we can make sure that by using the right loss function with a proper stability regularization, the learnt interpretability model can be made robust. And at the same time, given that the same model is used for both the needed function and the interpretability outputs, the processing time is much faster.

The self explanation interpretations are based on depicting the input as a concept vector and a influence matrix and then using the two to depict the importance of each concept in calculating the output. The same theta and encoded vector are used to calculate the output as well. This is done by splitting the model into three networks. The first one learns the encoded vector, the second one learns the influence matrix and the last one uses these two to get the output.

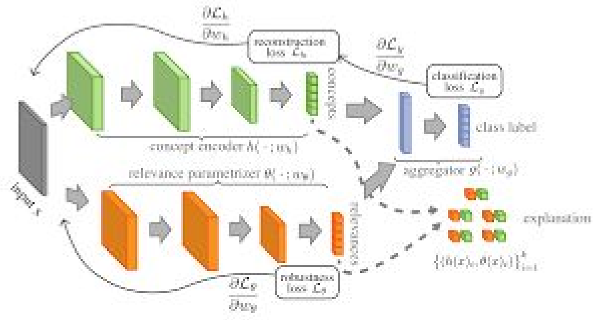

$\frac{\partial \mathcal{L}_h}{\partial w_h}$
reconstruction loss $\mathcal{L}_h$
$\frac{\partial \mathcal{L}_y}{\partial w_y}$
classification loss $\mathcal{L}_y$
concepts
class label
concept encoder $h(\cdot ; w_h)$
relevance parametrizer $\theta(\cdot ; w_\theta)$
aggregator $g(\cdot ; w_g)$
input x
explanation
$\{(h(x)_i, \theta(x)_i)\}_{i=1}^k$
relevances
robustness loss $\mathcal{L}_\theta$
$\frac{\partial \mathcal{L}_\theta}{\partial w_\theta}$

Fig 2.6 : The Self Explanation Architecture

Stability is enforced using the a regularization term which is added to the loss function. The loss functions consists of a term to learn the encoding reconstruction and the loss for the output classification. The experiments performed on these prove that the robustness is enforced and the stability is much better for these than what it was for the previous methods.

2.4.2 Examples

Consider MNIST Digit classification and let us solve it using the Self Explaining Neural Networks architecture. Once the model is trained, we extract the concepts using maximally activated encodings approach. The number of concepts is a hyperparameter and in this example we choose 5 concepts. In fig 3.7, we have two inputs of class “9” and “2”. To obtain predictions for this we just forward propagate it through the model, which is quite the same as in any other neural network. On the other hand in order to obtain explanations, we first extract the 5 concepts using the encoding network. This process of obtaining concepts is also referred to as grounding the model. Grounding is done only once when the model has completed training. The concepts remain the same and need not be computed over and over. To complete the explanation, we compute the influence matrix scores and choose the column according to the class predicted by the model. This column has influence scores corresponding to that particular class.

In fig 3.3, for image of class “9”, we observe that concept 3 has the most positive influence towards the prediction. This is sensible because the concept 3 is “7” which resembles “9”. For image of class “2”, the concepts “7” and “0/2” are influencing the prediction the most and is clearly justified.

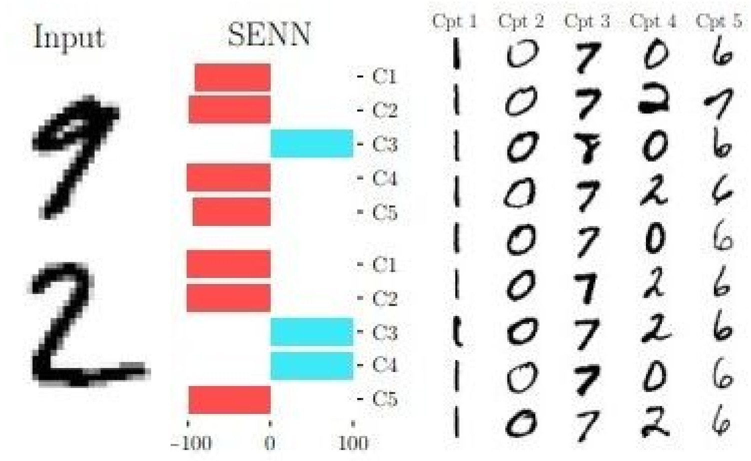


Fig 2.7 Self-Explaining Neural Networks explaining the input images of classes 9 and 2 using concepts and encodings

2.4.3 Advantages

Unlike most methods of interpretability, SENNs don’t employ a posteriori way of explaining. They enforce explainability into the architecture of the model without having to sacrifice performance. SENNs are also more robust and stable to noise in inputs since this class of models are explicitly optimized on robustness loss. The explanations are explicit and human understable. The features computed through encoding are truly relevant to the predictions on the input and the model is therefore faithful. Also, SENNs are trained end-to-end and do not need separate process to compute explanations. SENNs are a class of models rather than one single definite architecture. Model designers can choose whatever fits their needs when it comes to concept encoder, relevance parameterizer etc. This makes SENNs very flexible and more usable to wide variety of problems.

2.4.4 Disadvantages

Using concepts and their maximal images as a method of understanding features is easy for simple datasets. When we consider complex image datasets, the identification of features from images for a human to be easy is yet to be proven. A concept value cannot be used in isolation to understand the explanation. If we consider only a concept value, there could be two images with similar concept scores for a concept but contrasting influence scores thus contributing in different ways. So we need to look at the understanding as a combination of concept scores and the influence matrix. Self explaining neural networks are flexible and consequently there are additional hyperparameters that need to be tuned in order to obtain the model's optimal performance.

CHAPTER 3

METHODOLOGY

3.1 Proposed Approach

The overview of the approach followed for the project is as follows. The first task was to prove that LIME and SHAP based approaches aren't robust enough to be used as interpretable models. We wanted to prove this in the general sense and not confine it to a specific problem type. So, we implemented models to classify MNIST images and to obtain sentiment for IMDB tweets. On top of these we build LIME and SHAP models and proved that they are not robust for both tasks.

The second task was to develop self explanation based models and understand their outputs. To overcome the deficit of not being able to understand self explanation based models easily, we added a clustering based grounding approach to the architecture. We then proved that we could achieve a good stability for these interpretable models by not compromising on the performance. We have implemented these for MNIST based Image Classification and we will argue that they can be extended to other tasks like Sentiment Analysis as well.

The first part of the project was to prove the non robustness of Posteriori LIME and SHAP based Models. For the same we took up two tasks. The first task was Image Classification and the second task was Sentiment Analysis.

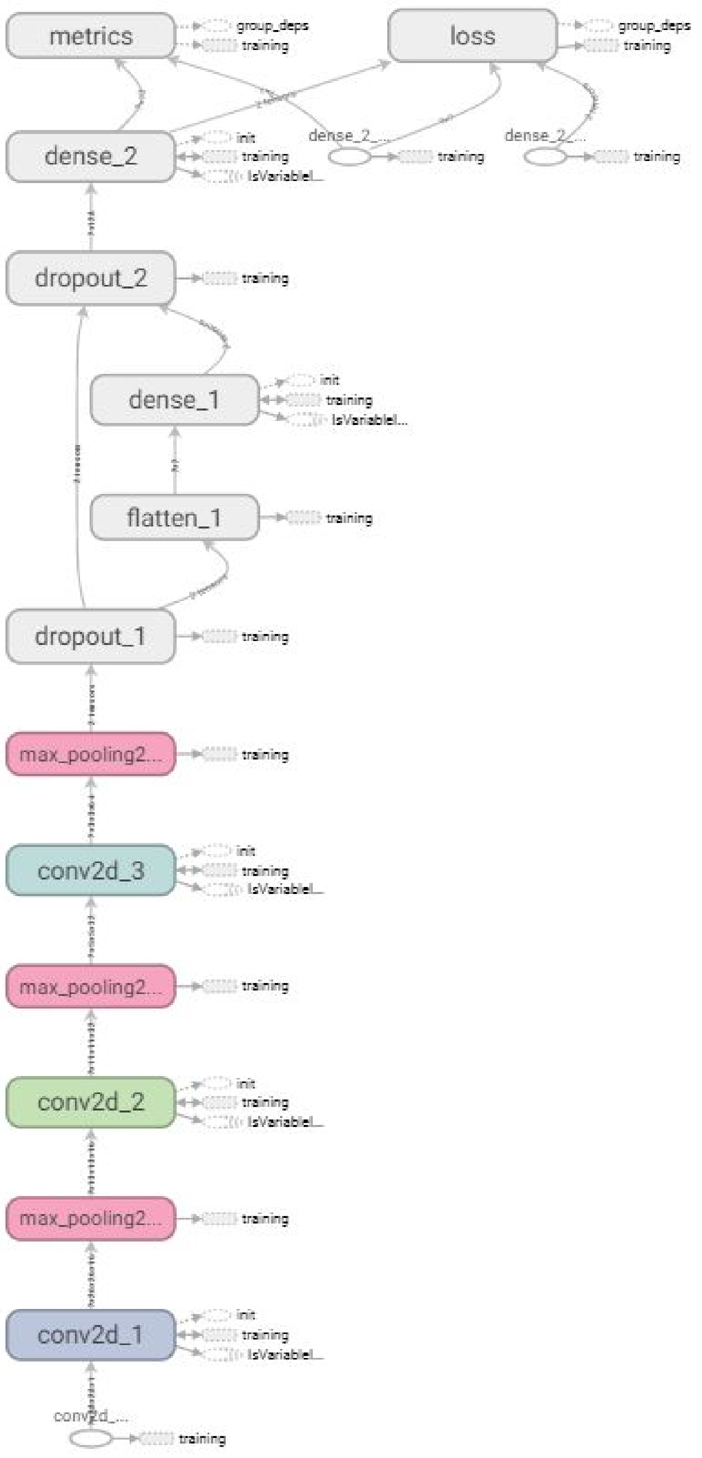


Fig 3.1 : CNN Architecture used for MNIST Classification

For the image classification we took up the MNIST dataset and used a Convolutional Neural Network to train for the same. We then built LIME and SHAP models on top of this trained model separately. The performance of the CNN was tested using the commonly used metrics such as accuracy, precision, recall and f1 score. For the LIME and SHAP models built on top of this model, Stability was calculated. We needed a metric to measure how similar two images were. Hence we used the squared error.

$$\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2.$$

Fig : 3.2 Squared Error Between two images.

In the above equation, n represents the number of pixels in the image and the difference between each corresponding pixel values is calculated which is then summed up and squared to get the similarity score for two images.

To obtain an image in the neighbourhood the original image, noise was added to the original image. It was ensured that the noisy image also belonged to the same output class as the original image. The type of noise added was Salt and Pepper Noise. An example of the original image and the noisy image is given below.

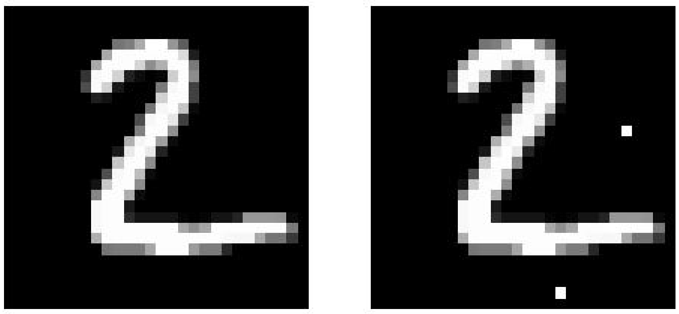


Fig 3.3 : Original Image and Noisy Image

The corresponding explanation images were also generated for both the original image and the noisy image. The similarity score for the explanation images was also calculated for the explanation images using the same metric. Now that we had the similarity scores for the original image and the noisy image, the original image explanation and the noisy image explanation, we had to calculate the stability metric. In order to eliminate the case where the sizes of the original image and its corresponding explanation were different, we divided the similarity score by the product of the dimensions of the image. This ensured that we obtained a normalized similarity metric for images.

Normalized Similarity Score = Squared Error / h X w X d

Here, Squared error was the score calculated using the previously mentioned metric. In the normalization part, **h** represents the height of the image, **w** represents the width of the image and **d** represents the number of dimensions in the image.

Stability was then calculated as the ratio of the normalized similarity of the original image explanation and the noisy image explanation to the normalized similarity of the original image and the noisy image.

$$Stability = \frac{Normalized\ Similarity\ of\ the\ original\ image\ explanation\ and\ noisy\ image\ explanation}{Normalized\ image\ similarity\ of\ the\ original\ image\ and\ the\ noisy\ image}$$

For the task of sentiment classification of imdb reviews, we have designed Recurrent Neural Networks to perform classification. We have trained the model and measured the metrics accuracy, recall, precision and f1-score. Over the RNN model, we have built LIME and Shap models to interpret the examples and predictions on imdb

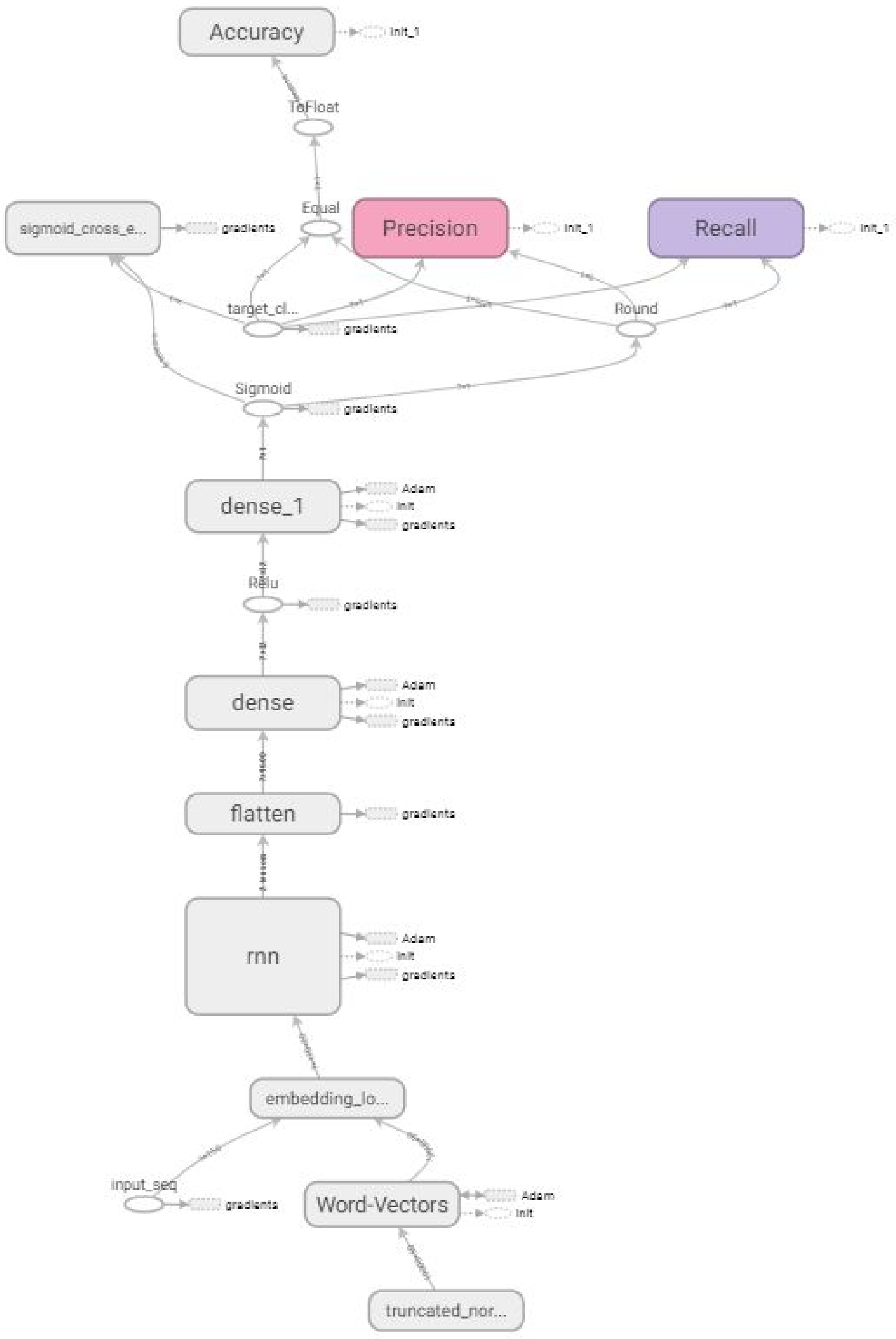


Fig 3.4 RNN Architecture for IMDB Sentiment Analysis

reviews. To evaluate the LIME and Shap models we used the stability metric. Stability is the ratio of change in explanation to change in input. Each explanation is a vector of values. Change in explanation is calculated as the magnitude of the vector obtained by taking the difference between the two explanations.

Change in input is simply,

*(noise / MAX_SEQ_LEN) ** 0.5*

This formalization is general enough that it scales to LIME, SHAP and SENN well to be able to compare their values fairly.

After we obtained the performance metrics and the interpretability metrics for both LIME and SHAP Models, we moved on to the Self Explanation based Models.

There are a couple of major differences between the previous type of models and self explanation models. The first is that in LIME and SHAP based approaches the models were built locally for each test sample. However in self explanation models, the models were global and the same model was used to explain for all the test samples. The second one is the LIME and SHAP based explanation models were built on top of already trained models for the corresponding tasks. However the Self explanation based models included the learning of explanations into the training process.

The self explanation based models divide the architecture into three parts. The first one is used for encoding the input into a vector of concepts. [7] The second part calculates the importance of each concept in the output task and the third part cumulates the two and gets the output for the task of classification or sentiment analysis. The technique used by self explanation models is the concept-relevance scores to explain how important each concept was in the task of classification which serves as an explanation to the corresponding output.

As mentioned in section 2.4.4, this has a couple of drawbacks. It it not easy to understand for a complex dataset just based on the images which maximally activated the concept. To make sure that the understandability of the explanation increased, we had to change the approach for grounding. This is where we added a unsupervised learning block to the model which takes into consideration two separate clusters[8] based on the encoding values and the corresponding relevance scores matrix.

We modified the definition of explanation as depicting the set of images from the training set that had similar encoding scores and concept relevance scores as the test image thereby indicating that the model weight values behaved similarly with respect to the test image. This serves as a condition for identification of images from the training set from which the model learnt to classify this corresponding test image.

Thus our definition of explanation for a test image was the set of images from which the model learnt to classify this corresponding test image. If the model learnt the right way than these set of images must be similar to the test image in terms of class or structure. If it didn't then they won't be similar. This grounding technique is easier to understand for a layman.

To obtain the similar set of training images, we used two clusters. The first one divides the training images into a set of clusters based on the concept scores and the second one divides the image into a set of clusters based on the influence scores for the concepts. For each test image we obtain the cluster of similar concept scores based images and similar influence scores based images. To obtain the set of images which are similar with respect to both the concept scores and the relevance scores, we then take an intersection of the two clusters to obtain the final set of grounding images.

To calculate the stability scores, we used both the concept scores and also the relevance matrix. Given that the clustering is done based on these two, similarity between the encodings and the relevance matrix should indicate the similarity in terms of the grounding images. The squared error metric defined in figure 3.1.1 is used to calculate the similarity between the original encodings and the noisy image encodings and also between the original relevance matrix and the noisy image relevance matrix. A Squared error close to zero for both of them indicate similarity in terms of the concepts and relevance which directly implies similarity in terms of the grounding images and thus a stable interpretable model.

3.2 High Level System Architecture

In this section, we describe the architecture we are proposing to improve interpretability without sacrificing the performance. We talk about the 4 blocks that constitute this architecture and describe on a high level the processes carried out in each of these blocks. Use fig 3.3 as a reference for the rest of the section.

3.2.1 Encoder-Decoder Block

This block contains an autoencoder to encode the input and reconstruct it using the decoder. For a given input, this block produces an encoding and reconstructed input from the encoding. The reconstructed input is used to measure the reconstruction loss and the model is trained to minimize this. The idea is to have meaningful encodings that are faithful to the original input fed into the block. The encoding produced by this block is used by the prediction block to make a prediction on the input and is later used by the explanation block to form clusters.

3.2.2 Relevance Network Block

This network takes the input similar to the encoder-decoder block and produces what we call as the relevance matrix. It is a matrix of scores of shape *k * number_classes.* Here *k* is a hyper-parameter that basically dictates how complex the input space for the problem. *k* as a rule of thumb is proportional to the *number_classes*. For example, in MNIST classification, we chose *k* to be 8. This relevance matrix is used by the prediction block to compute the prediction and also by the explanation block to form clusters. The relevance matrix is also used to stabilize the model by computing robustness loss which helps the model to act robust to noise in the inputs.

3.2.3 Prediction Block

This block takes in the encoding from the encoder-decoder block and the relevance matrix from the relevance network block and multiplies them and consequently applies softmax (designer's choice) to the product. This is the predicted output and is used to compute the classification loss and the model is trained to minimize this.

3.2.4 Explanation Block

In the explanation block, the encodings and the relevance matrices of all training data is collected. Then, two clustering algorithms are used to form clusters, one for encodings and one for relevance matrices. These clusters are used to form an explanation as a set of images that are close to input image both in terms of encoding and relevance matrix.

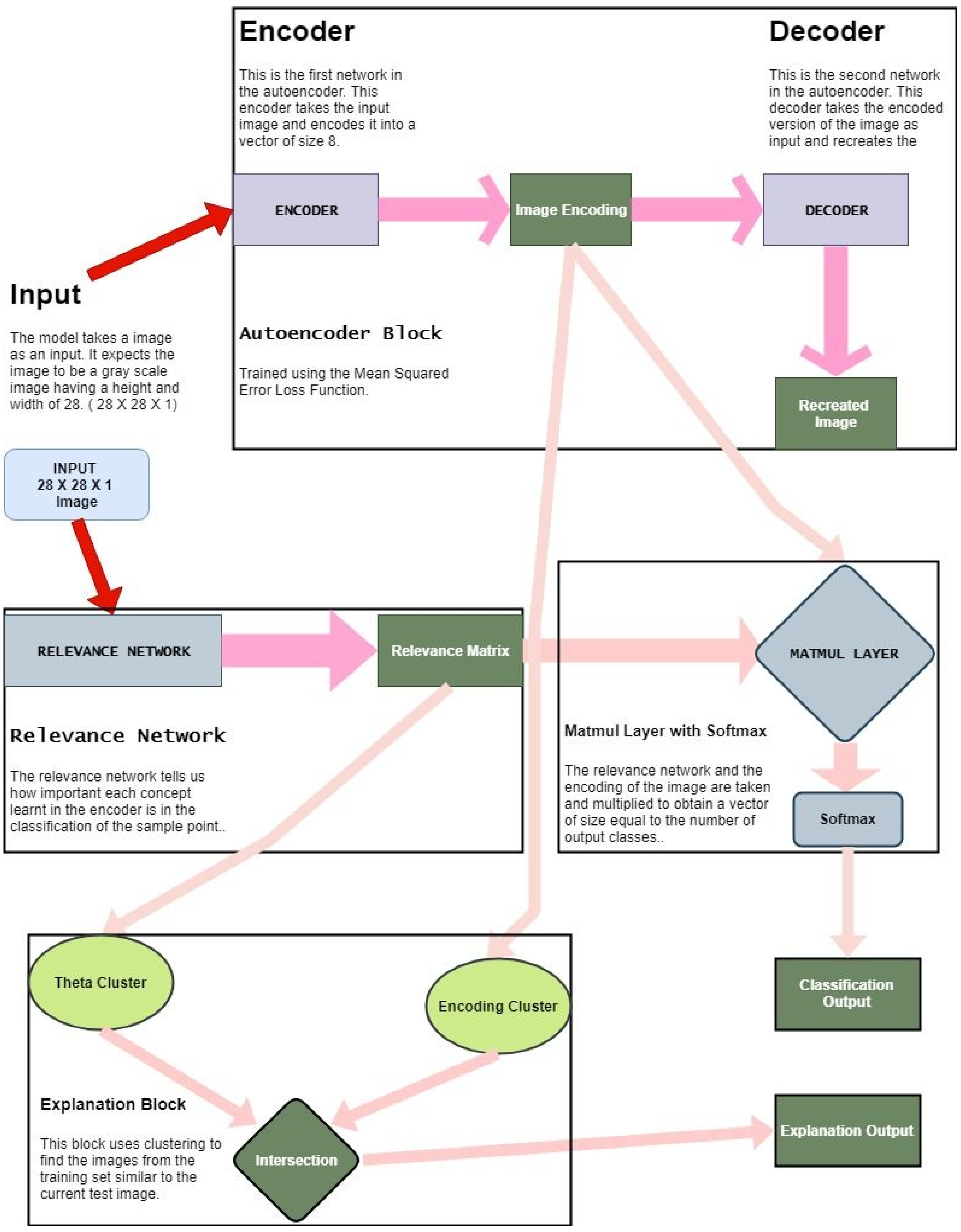

Encoder
This is the first network in the autoencoder. This encoder takes the input image and encodes it into a vector of size 8.
Decoder
This is the second network in the autoencoder. This decoder takes the encoded version of the image as input and recreates the
ENCODER
Image Encoding
DECODER
Recreated Image
Autoencoder Block
Trained using the Mean Squared Error Loss Function.
Input
The model takes a image as an input. It expects the image to be a gray scale image having a height and width of 28. (28 X 28 X 1)
INPUT 28 X 28 X 1 Image
RELEVANCE NETWORK
Relevance Matrix
Relevance Network
The relevance network tells us how important each concept learnt in the encoder is in the classification of the sample point..
MATMUL LAYER
Matmul Layer with Softmax
The relevance network and the encoding of the image are taken and multiplied to obtain a vector of size equal to the number of output classes..
Softmax
Theta Cluster
Encoding Cluster
Classification Output
Explanation Block
This block uses clustering to find the images from the training set similar to the current test image.
Intersection
Explanation Output

Fig 3.5 High Level Architecture of the class of models proposed to improve interpretability without sacrificing performance.

CHAPTER 4
ENVIRONMENT REQUIREMENTS

4.1 Hardware Requirements

For the purpose of training all the models , a GPU was needed to enhance the training process time. The GPU based online development tool called Google Collab Research provided by Google was used for the training process for all the models.

The specs of the same are as follows :

- GPU: 1 x Tesla K80 , having 2496 CUDA cores, compute 3.7, 12GB(11.439GB Usable) GDDR5 VRAM
- CPU: 1 x Single core hyper threaded i.e(1 core, 2 threads) Xeon Processors @2.3Ghz (No Turbo Boost) , 45MB Cache
- RAM: ~12.6 GB Available
- Disk: ~320 GB Available

For the process of deployment we used a flask based localhost server to deploy the webapp that was built.

4.2 Software Requirements

4.2.1 Operating System

- Windows 10
- any stable version

4.2.2 Programming Language

- Python
- 3.5 or above

4.2.3 Libraries / Packages

Package	**Version**
Keras	2.2.4 or above
Tensorflow	1.12.0 or above
Lime	0.1.1.32 or above
Shap	0.28.3 or above

4.3 Data Requirements

4.3.1 MNIST

The database of MNIST contains handwritten digits, which has a training set of 60,000 images and a test set of 10,000 images. It’s a subset of NIST. The images have been size-normalized and are centered in a fixed size image of size 28x28.

It is used widely in the machine learning community to establish benchmarks or evaluate new algorithms and models as it involves almost little to no efforts on preprocessing and formatting.

4.3.2 CIFAR-10

The dataset of CIFAR-10 contains 60000 images of size 32x32x3 across 10 classes. Images are uniformly distributed and therefore contains 6000 images per class. 50000 images are used for training/validation and the rest 10000 are used for testing.

This dataset is also quite popular in the machine learning community to establish benchmarks and evaluate new algorithms/models on the tasks of image classification.

4.3.3 IMDB Reviews

This is a dataset of 25,000 movies reviews collected from IMDB and each review is labeled by a sentiment of positive or negative. All reviews are preprocessed, and each is encoded as a sequence of word indexes of integers. This dataset is commonly used in text classification for the purposes of benchmarking and evaluating new algorithms or models.

Chapter 5
DEMONSTRATION OF OUTCOME

For the purpose of demonstrations of results from our proposed architecture, we will be building a GUI.

The dataset that we have used have been split up into three parts. The first two are the training and the validation set which were used for the processes of training and validation during the learning process.

The last one is the validation which was completely isolated from the learning process and this was the set used as a part of the GUI.

The GUI has an option to select an index for the image in the test set.

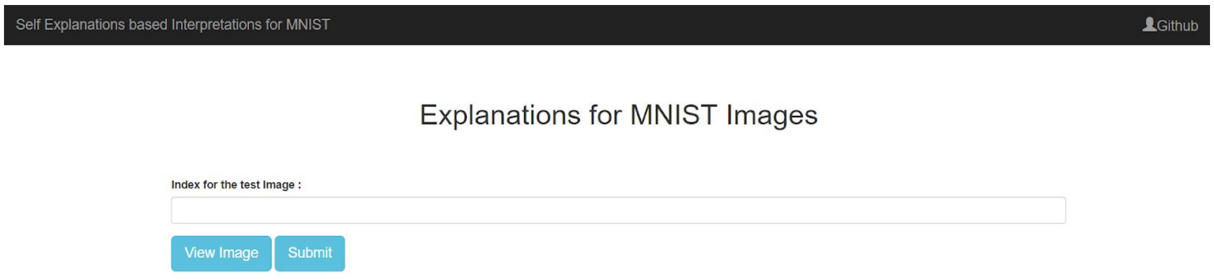

Self Explanations based Interpretations for MNIST
Github
Explanations for MNIST Images
Index for the test Image :
View Image
Submit

Fig 5.1 : GUI with the form to ask for the input.

The GUI has an option to view the image chosen based on the index. This lets the user get an idea of which image he or she has given for testing.

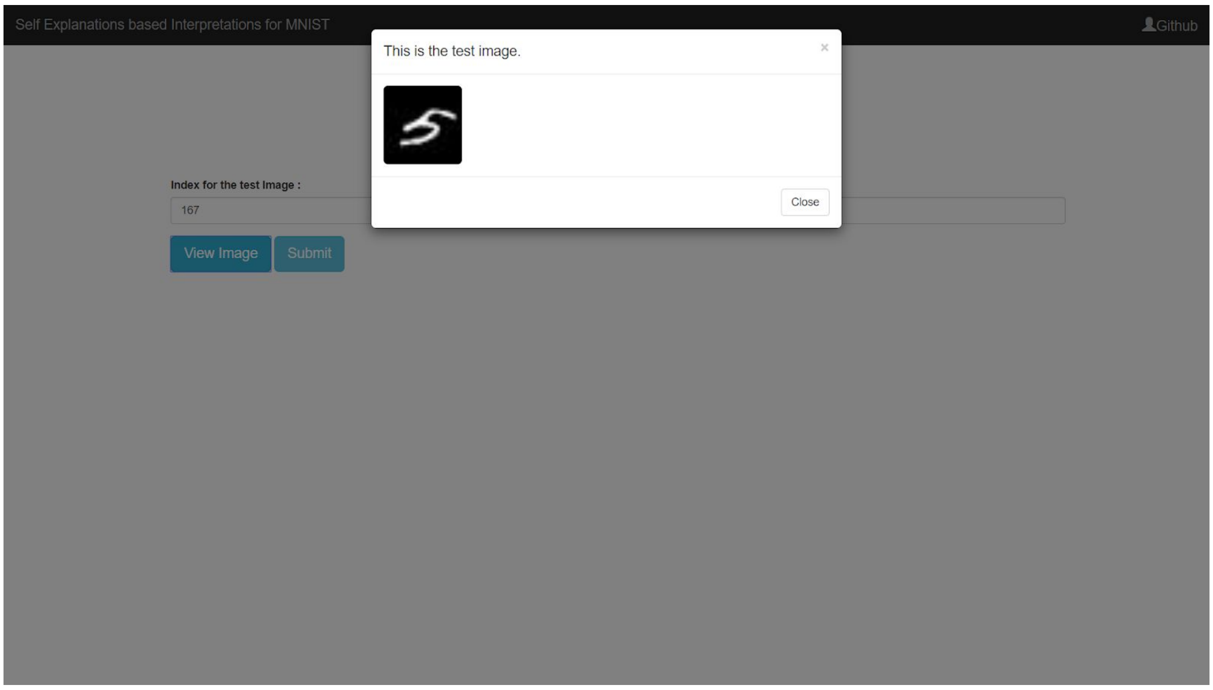
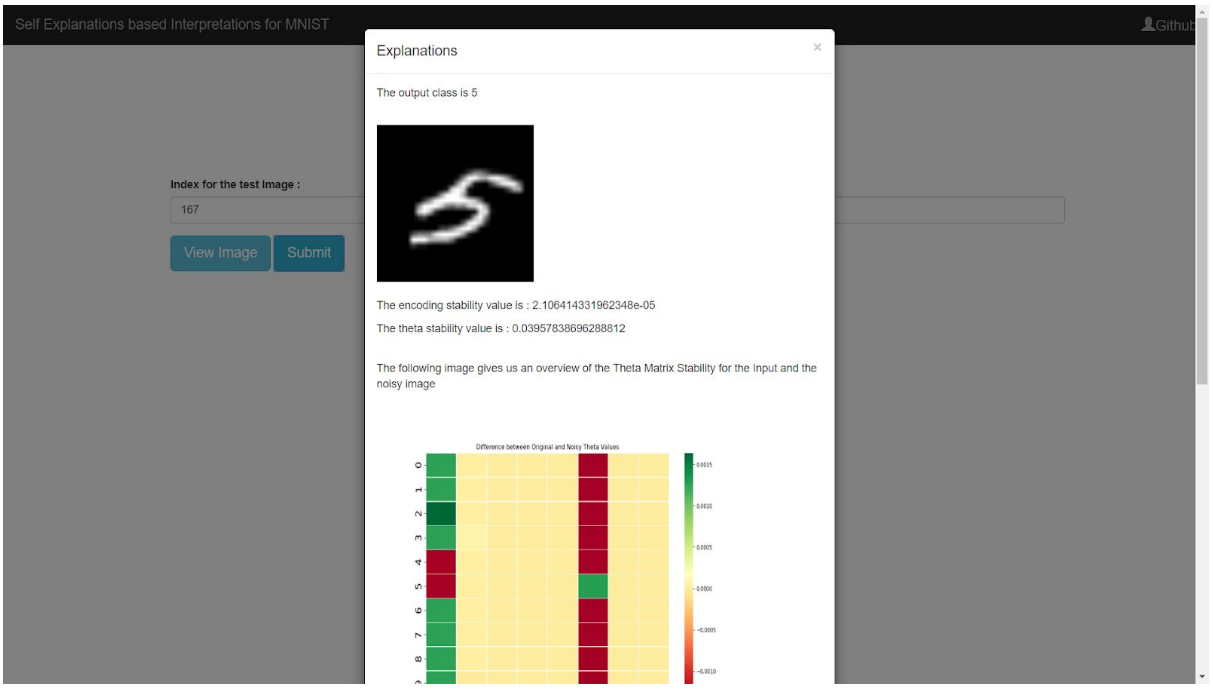

Self Explanations based Interpretations for MNIST
Github
This is the test image.
×
Close
Index for the test Image :
167
View Image
Submit

Fig 5.2 : Viewing the selected Image

Once the viewer has fixed on the image, he or she can submit and get the output. The output on the GUI will showcase the output class, the stability values and the corresponding explanations.

The user has an option to download the entire report which contains all the details regarding the concept values, the relevance matrix, the stability details with the noisy image and the grounding images.


Self Explanations based Interpretations for MNIST
Github
Index for the test Image :
167
View Image
Submit
Difference between Original and Noisy Theta Values

Explanations

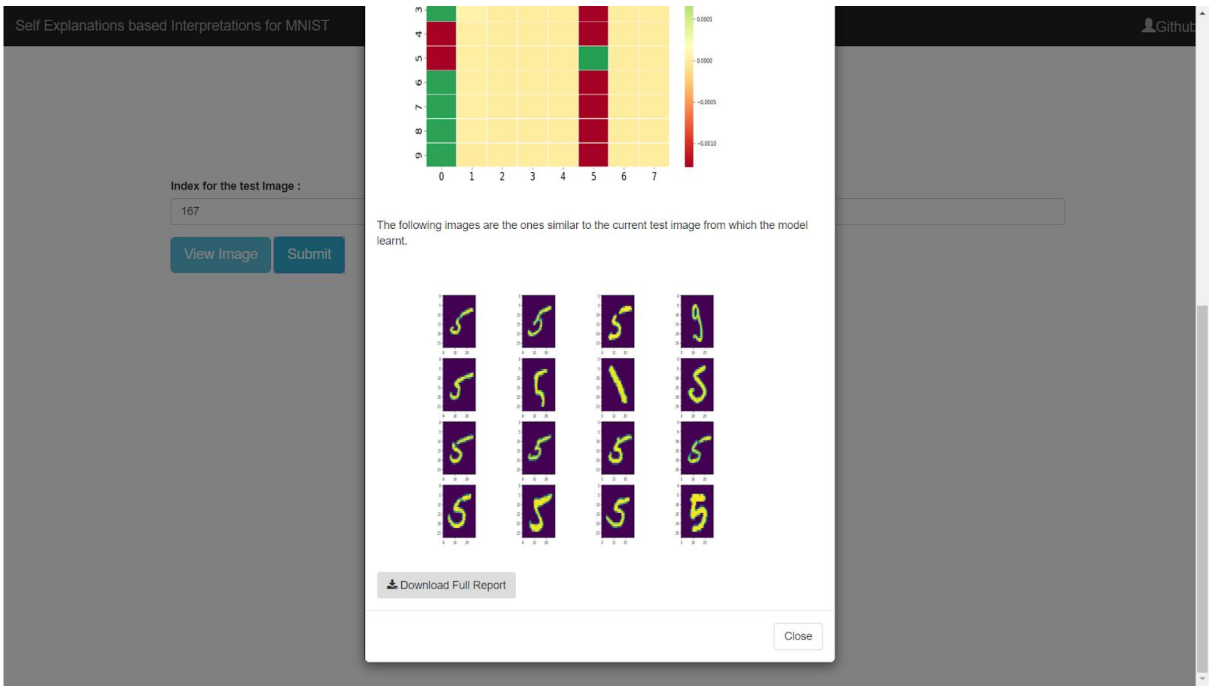
The output class is 5

The encoding stability value is : 2.106414331962348e-05

The theta stability value is : 0.03957838696288812

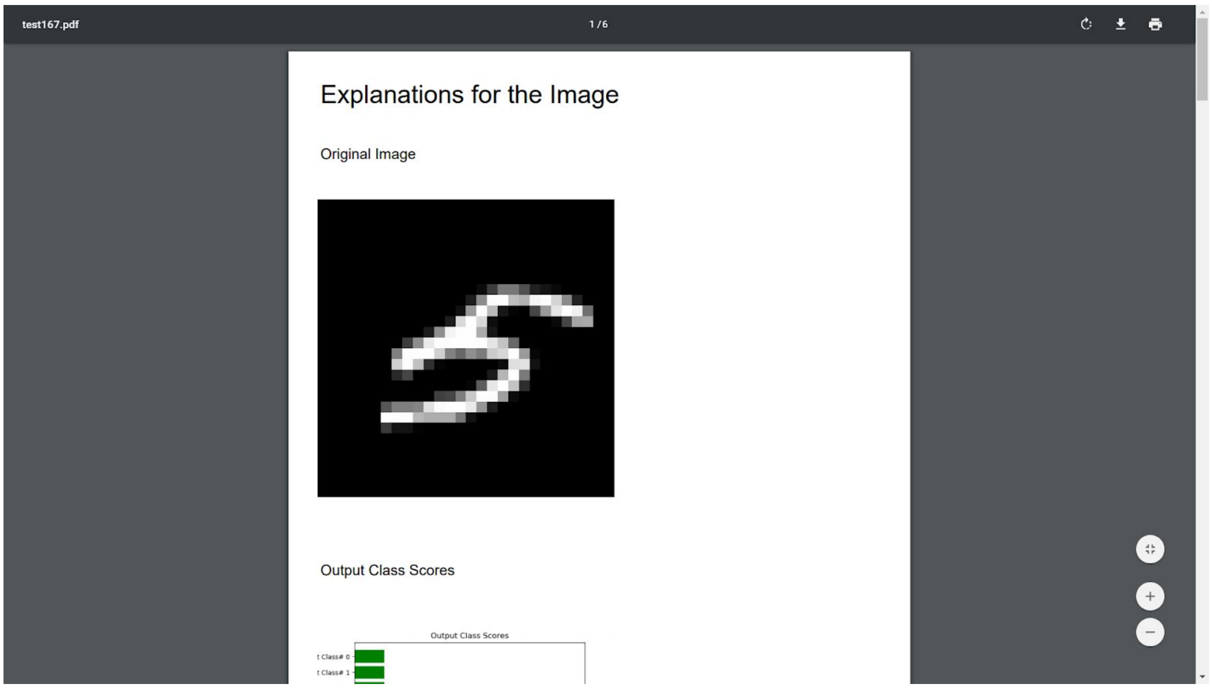
The following image gives us an overview of the Theta Matrix Stability for the Input and the noisy image

Fig 5.3 : Viewing the output on the screen


Self Explanations based Interpretations for MNIST
Github
Index for the test Image :
167
View Image
Submit
Download Full Report
Close

The following images are the ones similar to the current test image from which the model learnt.

Fig 5.4 : Last part of the on screen output with the download button


test167.pdf 1 / 6

Explanations for the Image

Original Image

Output Class Scores

Fig 5.5 : Full report available to the user for download

CHAPTER 6

PROPOSED APPROACH

The first task is to prove that LIME and SHAP based approaches aren't robust enough to be used as interpretable models. We wanted to prove this in the general sense and not confine it to a specific problem type. So, we implemented models to classify MNIST images and to obtain sentiment for IMDB tweets. On top of these we build LIME and SHAP models and proved that they are not robust for both tasks.

The second task is to develop self explanation based models and understand their outputs. To overcome the deficit of not being able to understand self explanation based models easily, we added a clustering based grounding approach to the architecture. We then proved that we could achieve a good stability for these interpretable models by not compromising on the performance. We have implemented these for MNIST based Image Classification and we will argue that they can be extended to other tasks like Sentiment Analysis as well.

The first part of the project was to prove the non robustness of Posteriori LIME and SHAP based Models. For the same we took up two tasks. The first task was Image Classification and the second task was Sentiment Analysis.

For the image classification we took up the MNIST dataset and used a Convolutional Neural Network to train for the same. We then built LIME and SHAP models on top of this trained model separately. The performance of the CNN was tested using the commonly used metrics such as accuracy, precision, recall and f1 score. For the LIME and SHAP models built on top of this model, Stability was calculated. We needed a metric to measure how similar two images were. Hence we used the squared error.

To obtain an image in the neighbourhood the original image, noise was added to the original image. It was ensured that the noisy image also belonged to the same output class as the original image. The type of noise added was Salt and Pepper Noise. An example of the original image and the noisy image is given below.

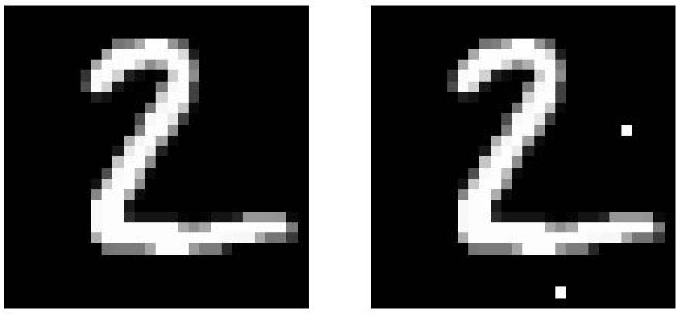


Fig 6.1 : Original Image and Noisy Image

The corresponding explanation images were also generated for both the original image and the noisy image. The similarity score for the explanation images was also calculated for the explanation images using the same metric. Now that we had the similarity scores for the original image and the noisy image, the original image explanation and the noisy image explanation, we had to calculate the stability metric. In order to eliminate the case where the sizes of the original image and its corresponding explanation were different, we divided the similarity score by the product of the dimensions of the image. This ensured that we obtained a normalized similarity metric for images.

Normalized Similarity Score = Squared Error / h X w X d

Here, Squared error was the score calculated using the previously mentioned metric. In the normalization part, **h** represents the height of the image, **w** represents the width of the image and **d** represents the number of dimensions in the image.

Stability was then calculated as the ratio of the normalized similarity of the original image explanation and the noisy image explanation to the normalized similarity of the original image and the noisy image.

$$Stability = \frac{Normalized\ Similarity\ of\ the\ original\ image\ explanation\ and\ noisy\ image\ explanation}{Normalized\ image\ similarity\ of\ the\ original\ image\ and\ the\ noisy\ image}$$

For the task of sentiment classification of imdb reviews, we have designed Recurrent Neural Networks to perform classification. We have trained the model and measured the metrics accuracy, recall, precision and f1-score. Over the RNN model, we have built LIME and Shap models to interpret the examples and predictions on imdb reviews.

To evaluate the LIME and Shap models we used the stability metric. Stability is the ratio of change in explanation to change in input. Each explanation is a vector of values. Change in explanation is calculated as the magnitude of the vector obtained by taking the difference between the two explanations. Change in input is simply,

*(noise / MAX_SEQ_LEN) ** 0.5*

This formalization is general enough that it scales to LIME, SHAP and SENN well to be able to compare their values fairly.

After we obtained the performance metrics and the interpretability metrics for both LIME and SHAP Models, we moved on to the Self Explaining Neural Networks.

There are a couple of major differences between the previous type of models and self explanation models. The first is that in LIME and SHAP based approaches the models were built locally for each test sample. However in self explanation models, the models were global and the same model was used to explain for all the test samples. The second one is the LIME and SHAP based explanation models were built on top of already trained models for the corresponding tasks. However the Self explanation based models included the learning of explanations into the training process.

The self explanation based models divide the architecture into three parts. The first one is used for encoding the input into a vector of concepts. [7] The second part calculates the importance of each concept in the output task and the third part cumulates the two and gets the output for the task of classification or sentiment analysis. The technique used by self explanation models is the concept-relevance scores to explain how important each concept was in the task of classification which serves as an explanation to the corresponding output. This has a couple of drawbacks. It it not easy to understand for a complex dataset just based on the images which maximally activated the concept. To make sure that the understandability of the explanation increased, we had to change the approach for grounding. This is where we added a unsupervised learning block to the model which takes into consideration two separate clusters[8] based on the encoding values and the corresponding relevance scores matrix.

At this point, let us understand the loss functions we need to formalize to achieve the different objectives of self explaining models. The loss functions involved are the classification loss, robustness loss and reconstruction loss.

Below, we see the combined loss function *L*.

$$L = Ly(f(x), y) + \lambda L\theta(f) + \xi Lh(x, \hat{x})$$

Classification loss, *Ly* involves minimizing the cross entropy between the prediction *f(x)* and the target output y for a given input.

Robustness loss, *Lθ* involves encouraging the model *f* to behave locally around the inputs so as to see almost no change in the relevance matrix for very small noise in the input. *Lθ* has robustness coefficient called *λ* that is a trade-off between performance and stability.

Reconstruction loss, *Lh* involves making the autoencoder *h(x,x^)* more faithful by being able to decode the input from the computed encoding. *Lh* has a coefficient *ξ* that is a trade-off between performance and faithfulness. It is sometimes referred to as sparsity strength parameter.

We modified the definition of explanation as depicting the set of images from the training set that had similar encoding scores and concept relevance scores as the test image thereby indicating that the model weight values behaved similarly with respect to the test image. Thus our definition of explanation for a test image was the set of images from which the model learnt to classify this corresponding test image.

To obtain the similar set of training images, we used two clustering models. The first one divides the training images into a set of clusters based on the concept scores and the second one divides the image into a set of clusters based on the influence scores for the concepts. For each test image we obtain the cluster of similar concept scores based images and similar influence scores based images. To obtain the set of images which are similar with respect to both the concept scores and the relevance scores, we then take an intersection of the two clusters to obtain the final set of grounding images.

To calculate the stability scores, we used both the concept scores and also the relevance matrix. Given that the clustering is done based on these two, similarity between the encodings and the relevance matrix should indicate the similarity in terms of the grounding images. The squared error metric is used to calculate the similarity between the original encodings and the noisy image encodings and also between the original relevance matrix and the noisy image relevance matrix. A squared error close to zero for both of them indicate similarity in terms of the concepts and relevance which directly implies similarity in terms of the grounding images and thus a stable interpretable model.

CHAPTER 7

DEVELOPING THE ARCHITECTURE

In this section, the entire process of architecture development and fine tuning is described. It depicts the initial stages of the architecture starting from the most basic one and how it was modified based on the results obtained. The corresponding results and reasons for the change in the architecture are also mentioned here.

The initial implementation of the architecture closely resembled the layout of the Self Explanation Neural Network proposed by David Alvarez and Tommi Jaakkola in their paper Towards Robust Interpretability with Self-Explaining Neural Networks. However the finer details of the network such as the architectures for the encoder block, the relevance parametrizar block, the activation functions used, the size and the number of filters in each layer, the dense layer parameters were not made aware by them in their work. Thus we had to select these based on our ideas and needs.

One important metric that was used to verify the overall goodness of the model was its ability to differentiate concepts based. This was done by representing each concepts by the set of images that maximally activate the concept. It wasn't necessary for all concepts to uniquely represent a different set of images but at least a subset of the concepts had to be discernible. When we traced this back to the encoding we realised that a difference in the encoded vectors resulted in proper grounding. Thus the focus was to ensure that each image activated a different concept. We also noted the performance metrics for the same.

Over the course all the experimentation the following was kept constant :

1. The architecture for the Encoder

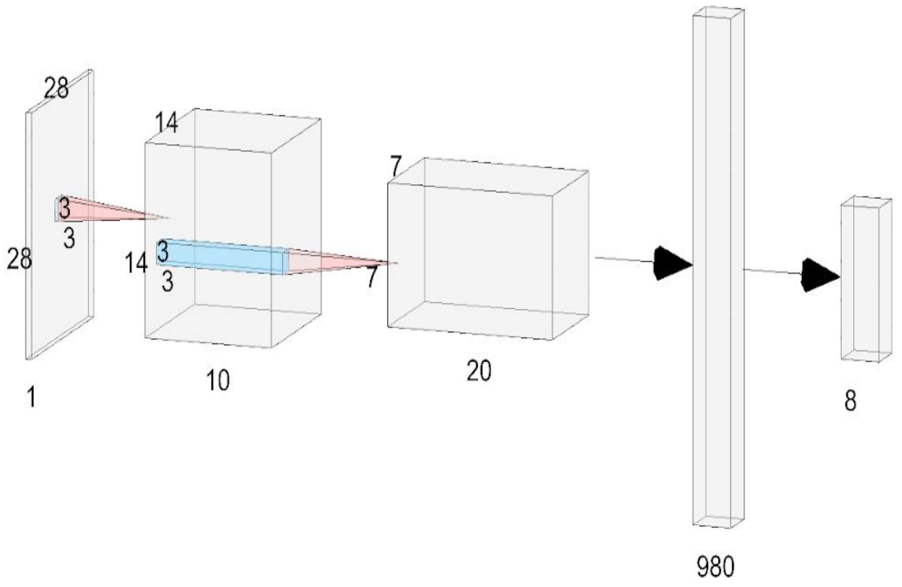


Fig 7.1 : Encoder Architecture

2. The architecture of the Decoder.

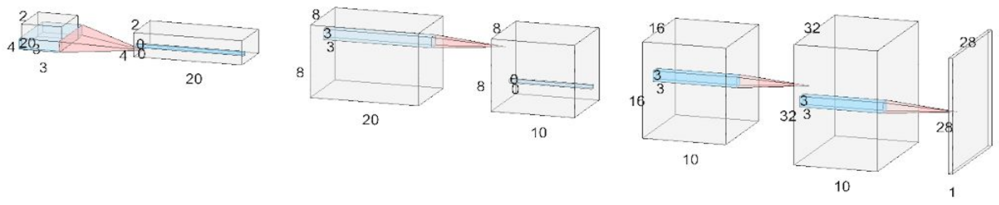


Fig 7.2 : Decoder Architecture

At the end of the encoder, the 8 dimensional vector is reshaped into a matrix of size 2 X 4 before passing to the decoder input.

3. The architecture of the relevance network. At the end of the network, the 80 dimensional vector was reshaped into a matrix of size 8 X 10.

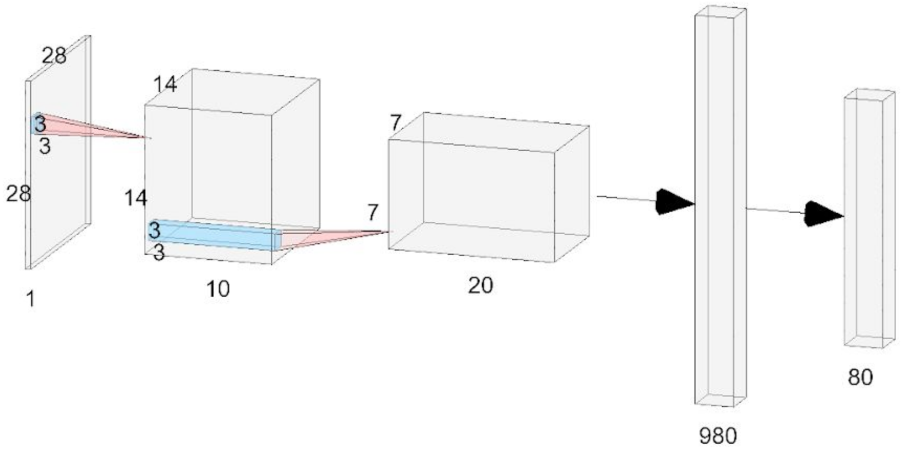


Fig 7.3 : Relevance Network Architecture

Once these were kept constant the other hyper parameters were changed until the right results were obtained.

7.1 Encoder with Sigmoid and Relevance Network with Relu

Initially the activation function at the last layer of the encoder was kept as sigmoid[9] and the one for the last layer of the relevance network was kept as relu[9]. This means that each concept value was between 0 and 1 and the influence scores matrix value was at least 0. In this case the performance metrics on the test set are mentioned in the table 7.1.

The concept images for the same are as follows :

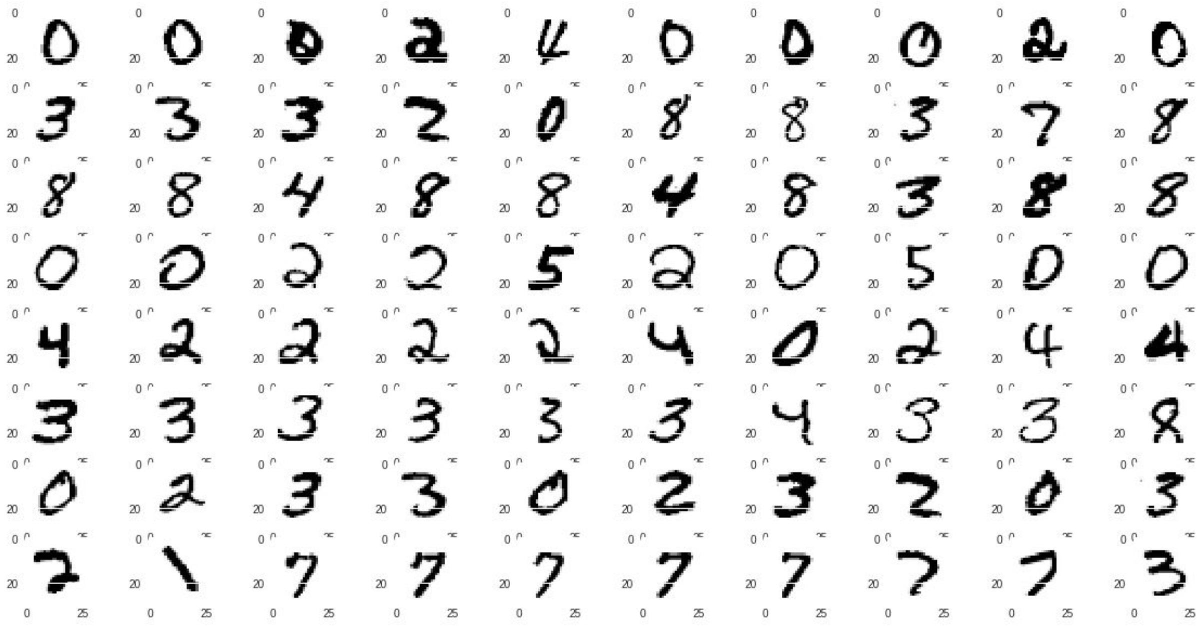


Fig 7.4 : Maximally Activated Concept Images for case 7.1

(Each row is one concept)

As noticed here, there is no discernible pattern in the concept encoding. We attributed this to a couple of things. Firstly all the encoding values were getting saturated and were very close to 1. This caused no perceivable difference in the encodings for each image. The second issue was in the learning, all the features or concepts were forced to have at least a positive influence on the output as the minimum value for the relevance matrix score was zero. This was because of the Relu Activation used at the last layer of the Relevance Network.

7.2 Encoder with Sigmoid and Relevance Network with tanh

Changing the final layer of the relevance network to a tanh[9] activation ensured that the relevance scores were in the range of [-1,+1] thus ensuring that there could be a concept that could influence the output in a negative way. The last layer activation of the encoder was still sigmoid.

The concept images for this case are as follows :

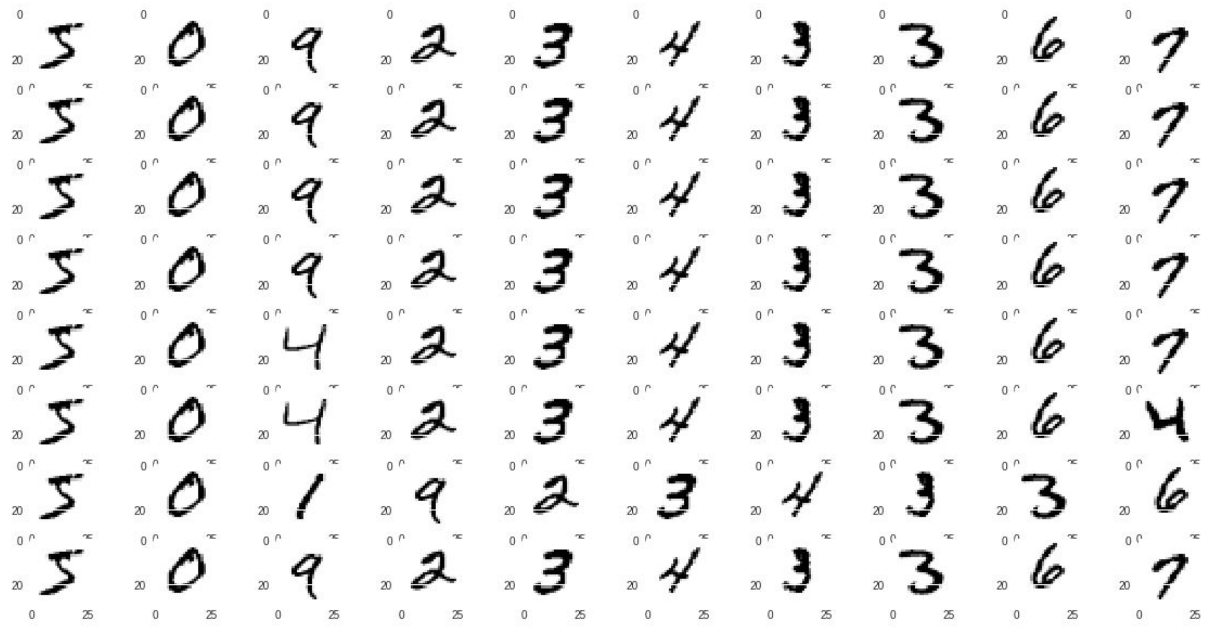


Fig 7.5 : Maximally Activated Concept Images for case 7.2.

(Each row is one concept)

Here we noticed that for each concept the same set of images were getting activated. That is a certain set of images activated all concepts maximally. This again was different from what was needed. We concluded that this was happening because an image was allowed to activate more than once concept.

7.3 Encoder with Softmax and Relevance Network with tanh

In order to ensure that an image maximally activates one concept or in other words, every image had one concept which is activated, the last layer of the encoder was switched to softmax activation.

The concept images for this are as follows :

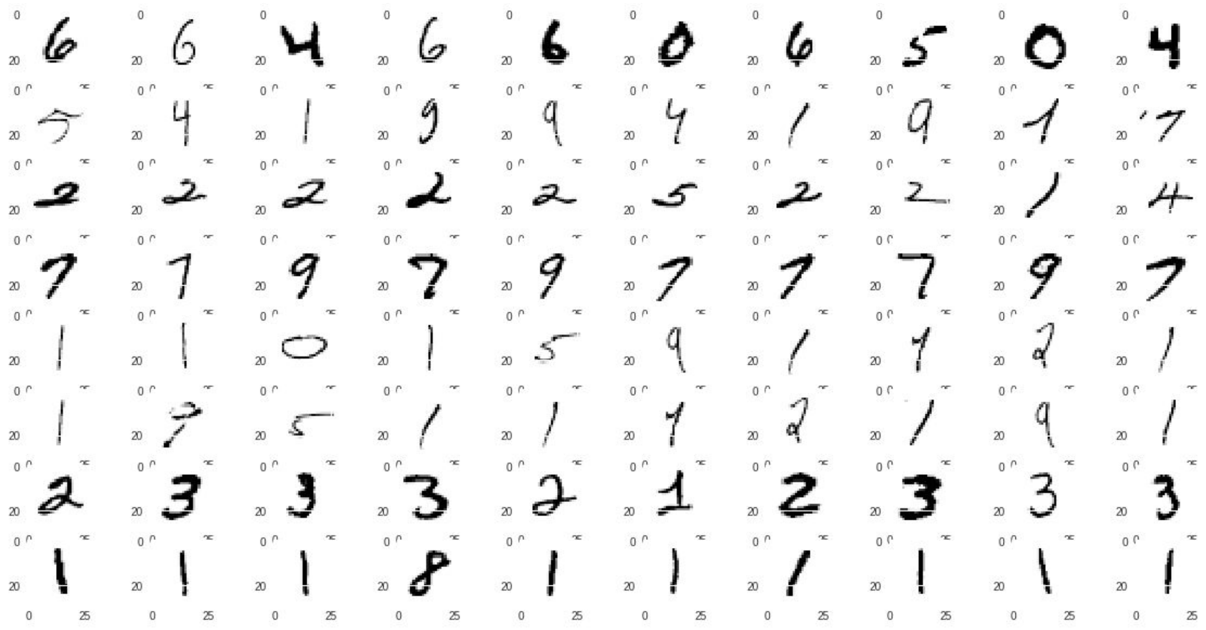


Fig 7.6 : Maximally Activated Concept Images for case 7.3.

(Each row is one concept)

Here, most of the concepts had discernible patterns and thus this case was chosen for further experimentation.

Case	Accuracy	Precision	Recall	F1 Score
7.1	0.9848	0.9849	0.9848	0.9848
7.2	0.9873	0.9874	0.9873	0.9874
7.3	0.9729	0.9730	0.9728	0.9729

Table 7.1 : Performance Metrics for each Case.

CHAPTER 8

RESULTS

8.1 MNIST CNN Classification

The CNN architecture was trained on MNIST. The model was trained for 20 epochs with batch size as 128 and learning rate as 0.001. This was the model that was used to further build LIME and SHAP Models for the interpretability tasks. The optimizer used was Adam Optimizer. The loss function was categorical cross entropy.

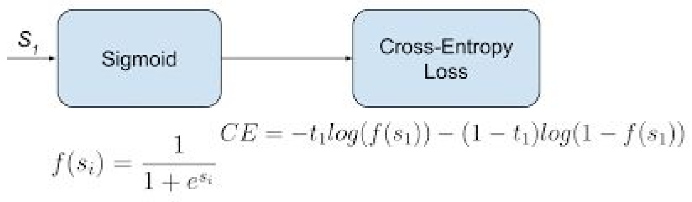

s_1
Sigmoid
Cross-Entropy Loss
$CE = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1))$
$f(s_i) = \frac{1}{1 + e^{s_i}}$

Fig 8.1 : Loss Function Used

The performance graphs for the training and validation process are as follows :

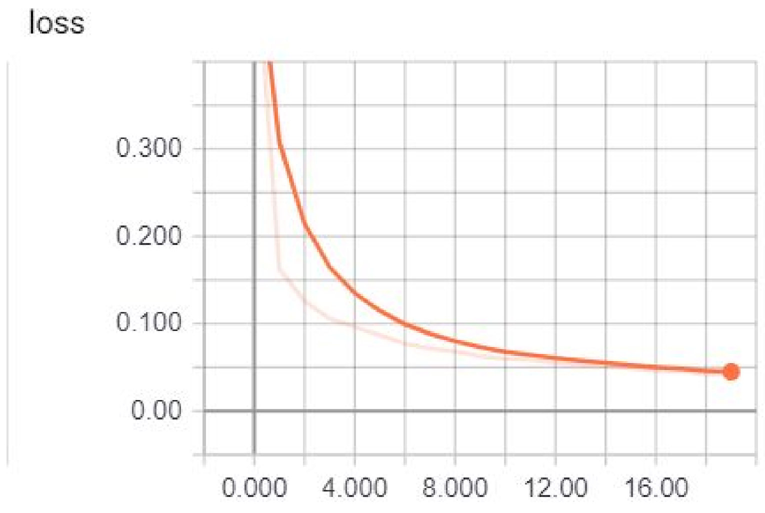


Fig 8.2 Training Loss vs Epochs Graph For MNIST CNN

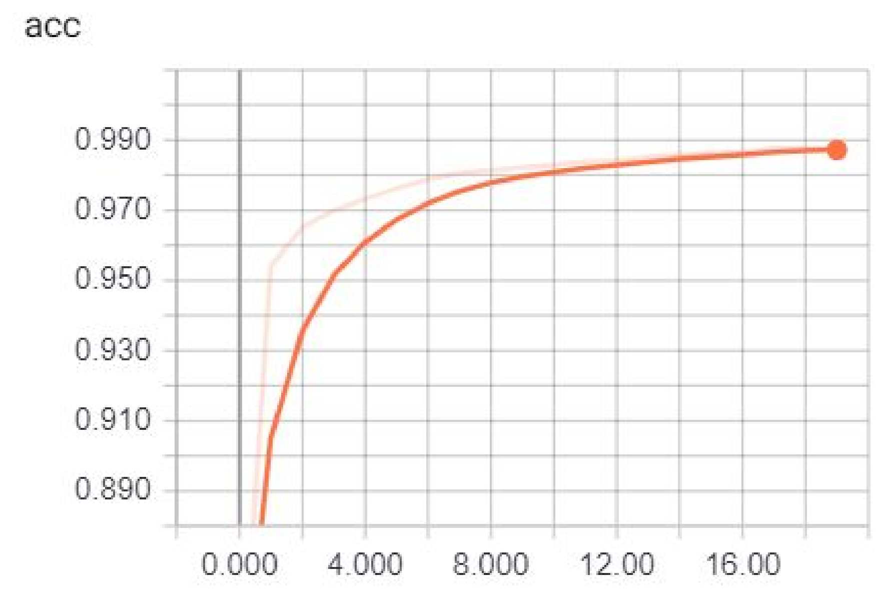


Fig 8.3 : Training Accuracy vs Epochs graph for MNIST CNN

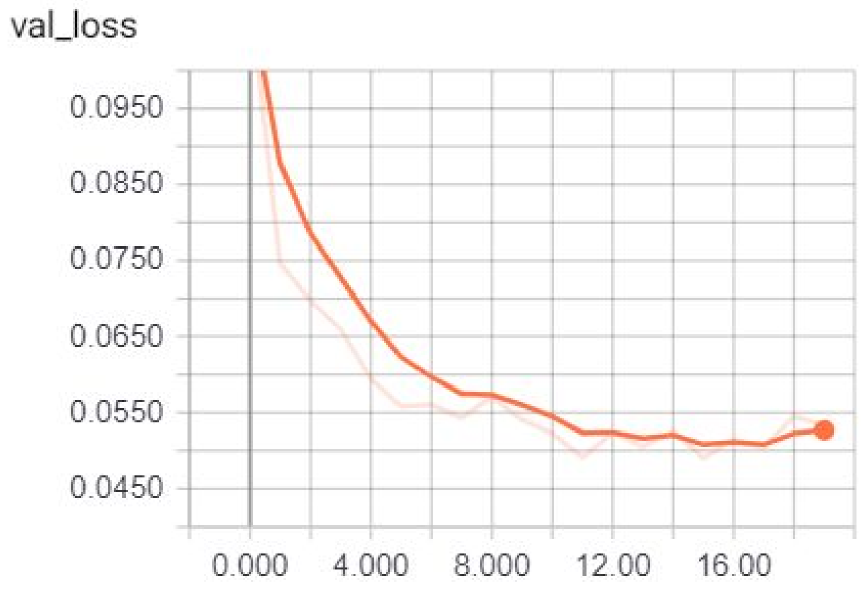


Fig 8.4 : Validation Loss vs Epochs graph for MNIST CNN

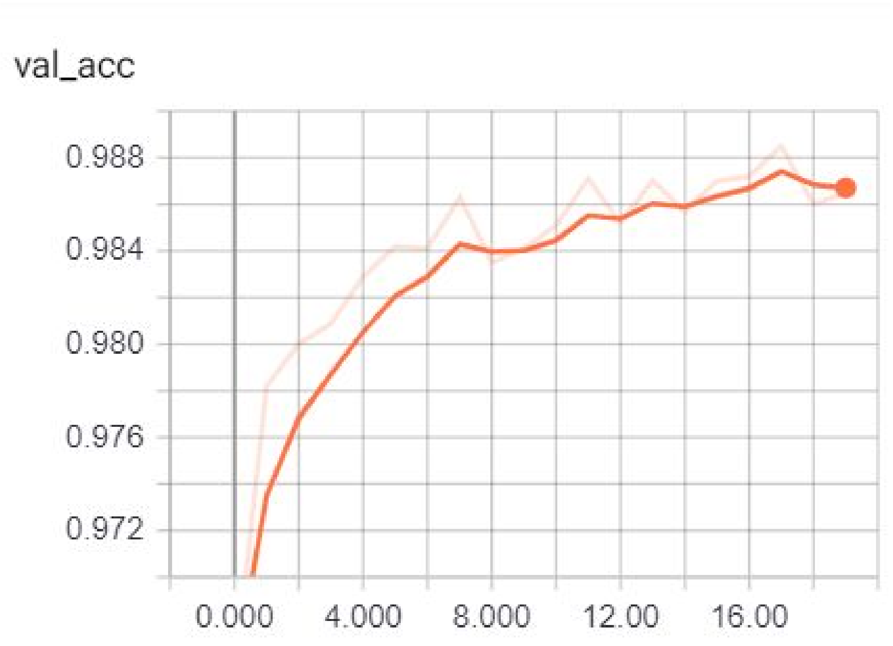


Fig 8.5 : Validation Accuracy vs Epochs graph for MNIST CNN

8.1.1 LIME Explanation with MNIST

On top of this CNN Model, LIME Models were built to get the explanations for each test image.

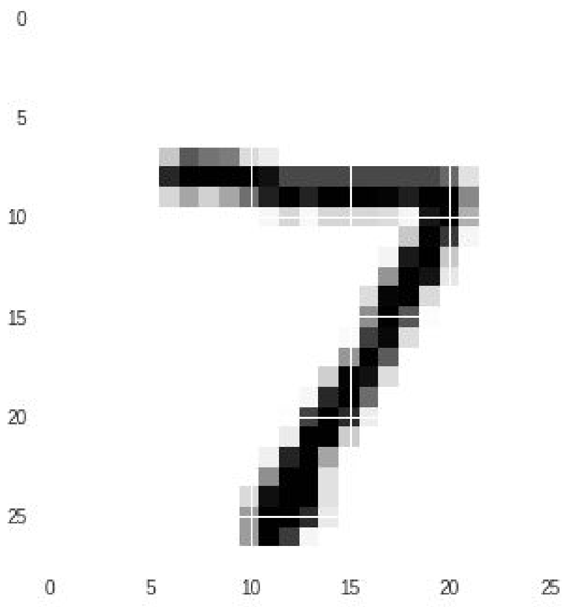


Fig 8.6 : Original Image

Then a certain amount of noise was added to the image to obtain a similar image in the neighbourhood of the original image. This is shown below.

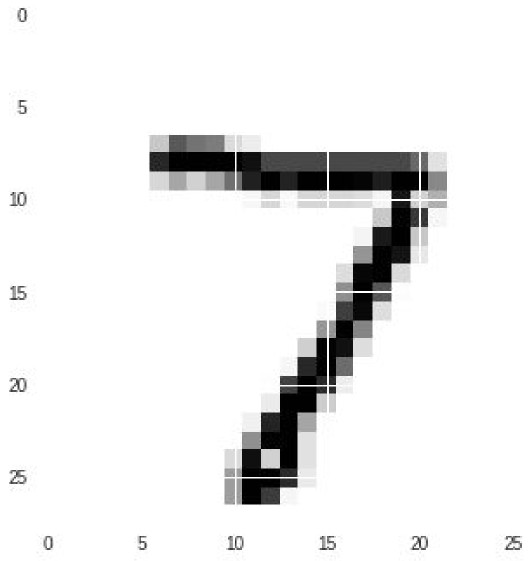


Fig 8.7 : Noise added to the original Image.

Next, the explanations provided by LIME were compared for both the images to check how different they were. Given that the difference between the original image and the noisy image was not a lot, the explanations should not differ by much for the interpretable model to be robust enough.

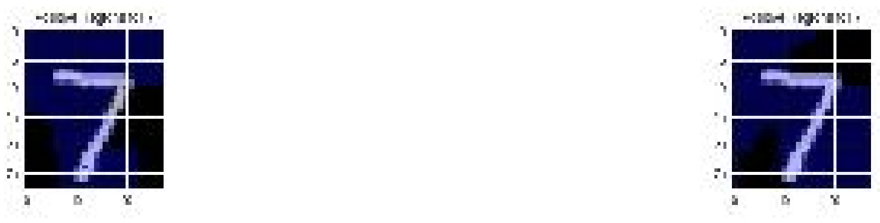


Fig 8.8 : The Explanation for the original image and the noisy image respectively.

It was noticed that the mean squared error between the two images was almost 181.This was very high and thus the LIME Model was not found to be Stable.

8.1.2 SHAP Explanations with MNIST

Again, on top of the same CNN Model, a SHAP model was built for the explanations for MNIST. The results of the same are as follows :

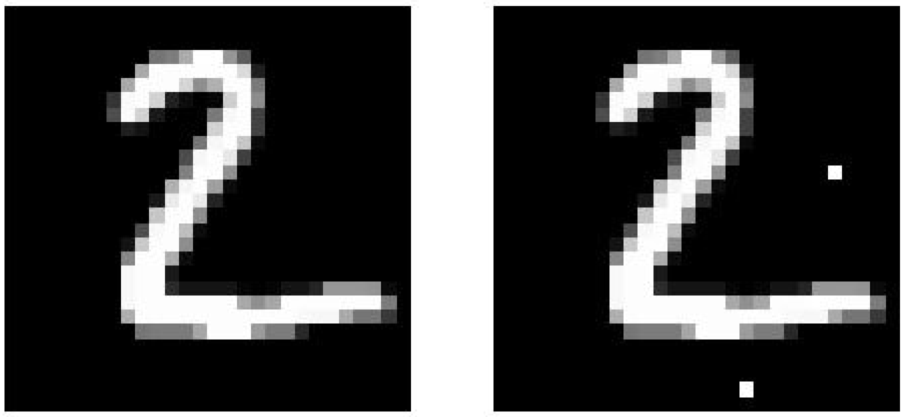


Fig 8.9 Original Image and the corresponding Noisy Image. The squared error between the two images was approximately 0.01.

For these two images, the explanations using SHAP were obtained.

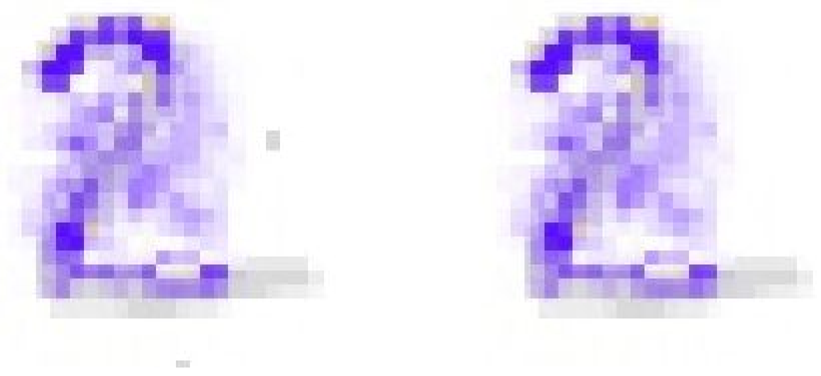


Fig 8.10 : Explanations for the Original Image and the Noisy Image.

It was noted that for two images, whose squared error is just 0.01, the explanations had a squared error of almost 12.5. This proves that the SHAP Explanations obtained are not robust at all.

8.2 IMDB Sentiment Analysis using RNNs

We trained a LSTM RNN model on the imdb dataset. The architecture has an embedding layer followed by a RNN layer with 64 LSTM units further followed by a couple of dense layers with a sigmoid cross entropy loss function.

The model has been trained for 50 epochs and the model's performance over epochs is described in the figures below.

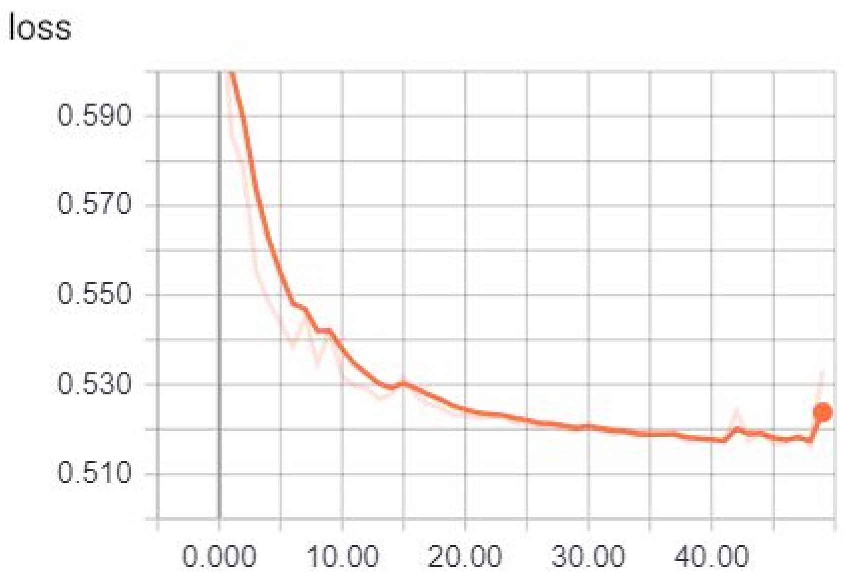


Fig 8.11 Training Loss vs Epochs

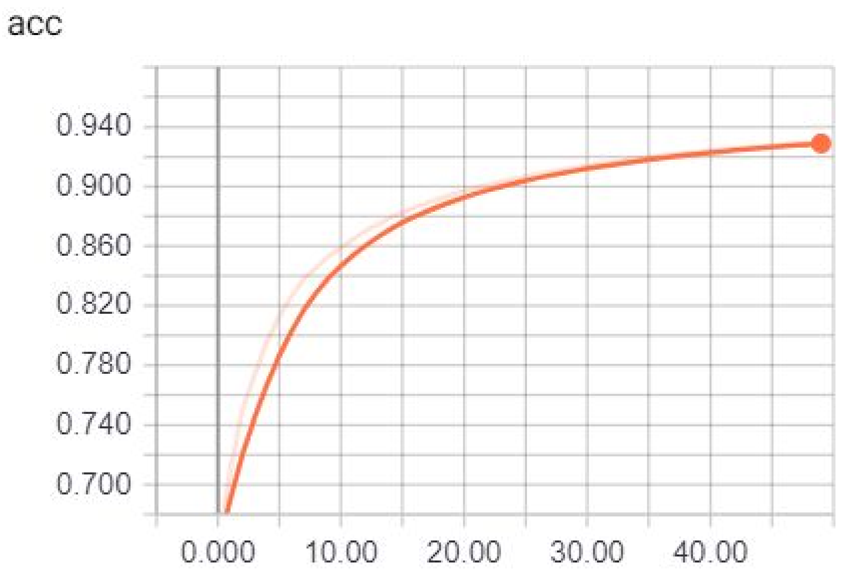


Fig 8.12 Training Accuracy vs Epochs

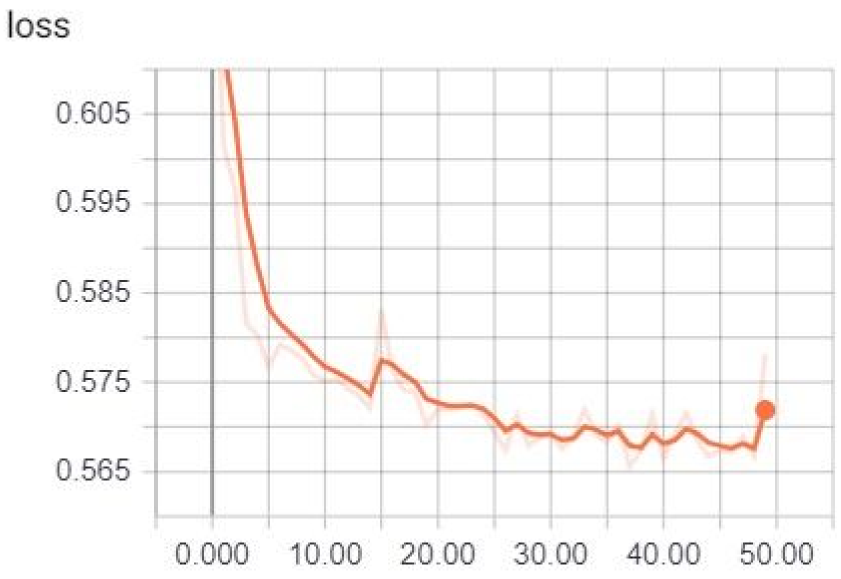


Fig 8.13 Validation Loss vs Epochs

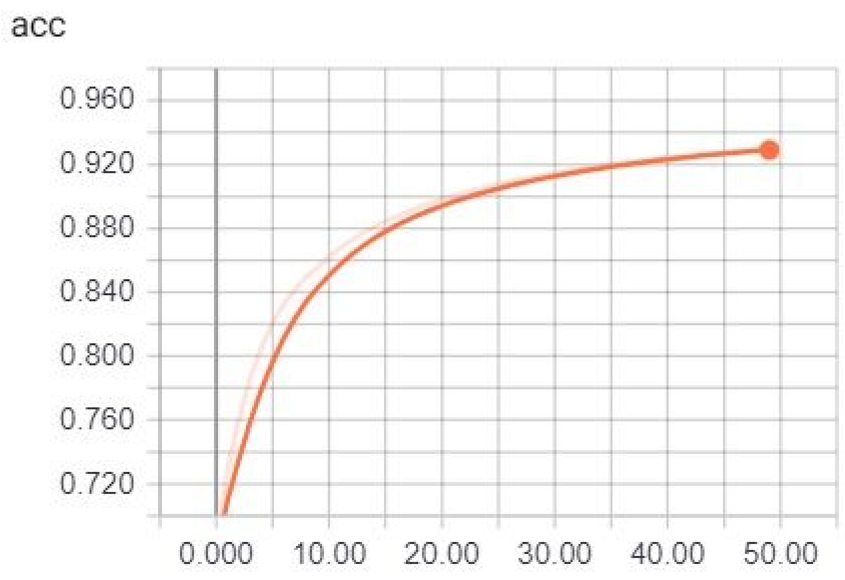
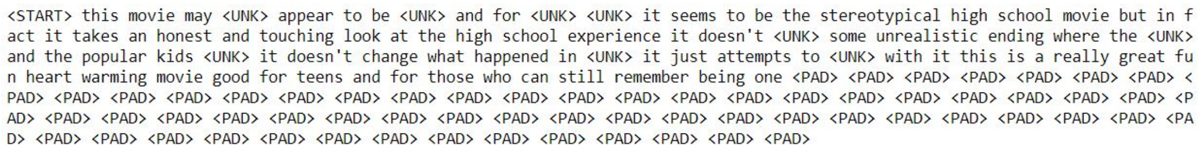


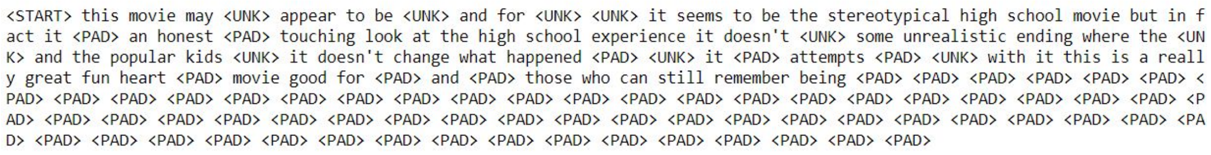
Fig 8.14 Validation Accuracy vs Epochs

On test dataset we achieved 91.7% accuracy, 95.8% precision and 87.3% recall



```
<START> this movie may <UNK> appear to be <UNK> and for <UNK> <UNK> it seems to be the stereotypical high school movie but in f
act it takes an honest and touching look at the high school experience it doesn't <UNK> some unrealistic ending where the <UNK>
and the popular kids <UNK> it doesn't change what happened in <UNK> it just attempts to <UNK> with it this is a really great fu
n heart warming movie good for teens and for those who can still remember being one <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <
PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <P
AD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PA
D> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
```

Fig 8.15 An example review from IMDB Dataset



```
<START> this movie may <UNK> appear to be <UNK> and for <UNK> <UNK> it seems to be the stereotypical high school movie but in f
act it <PAD> an honest <PAD> touching look at the high school experience it doesn't <UNK> some unrealistic ending where the <UN
K> and the popular kids <UNK> it doesn't change what happened <PAD> <UNK> it <PAD> attempts <PAD> <UNK> with it this is a reall
y great fun heart <PAD> movie good for <PAD> and <PAD> those who can still remember being <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <
PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <P
AD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PA
D> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
```

Fig 8.16 Example review that has been perturbed with noise

With the examples from fig 8.15 and fig 8.16 we will look at the LIME explanations obtained by us.

8.2.1 LIME explanations with IMDB

For text in fig 8.15 we can observe its LIME explanation in fig 8.17. We see 5 words are assigned scores based on how they influence the model’s prediction as positive sentiment. We can see that the words “touching”, “great” are influencing the prediction positively.

```
Model's prediction -> Negative: 0.0000, Positive: 1.0000
```

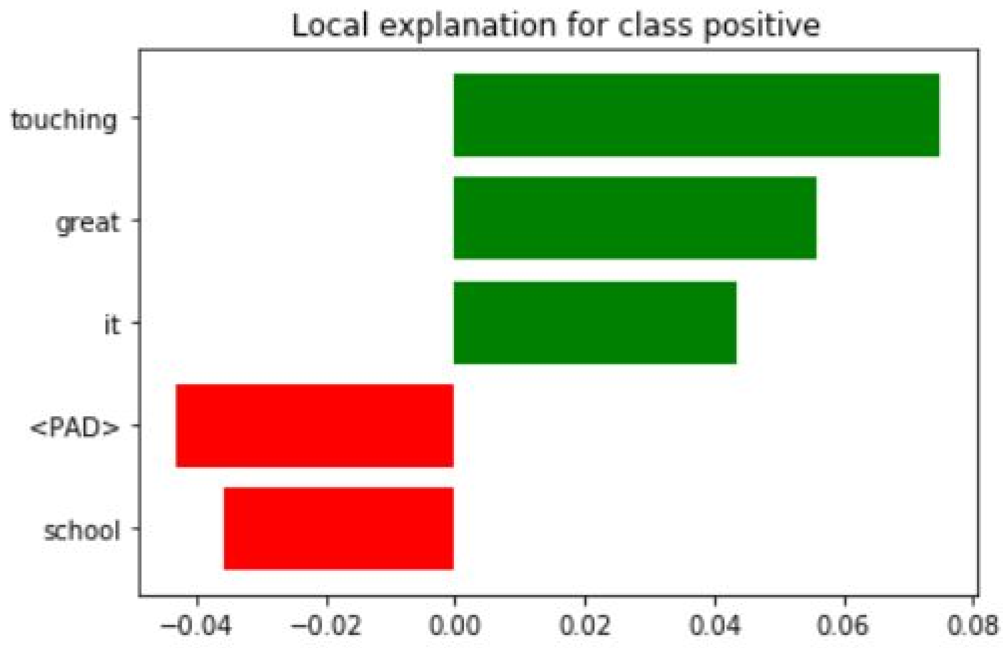


Fig 8.17 LIME explanation for the original text

We added noise to the text in fig 8.15 and obtained the text in fig 8.16. We can observe the LIME explanation for this noisy text in fig 8.18. “Touching” remains the highest influencer of the prediction but “great” is not as big an influencer as it was previously. For a noise of 10 in a text of length 150, i.e. 6.67% noise, we see explanations changing significantly.

```
Model's prediction -> Negative: 0.0000, Positive: 1.0000
```

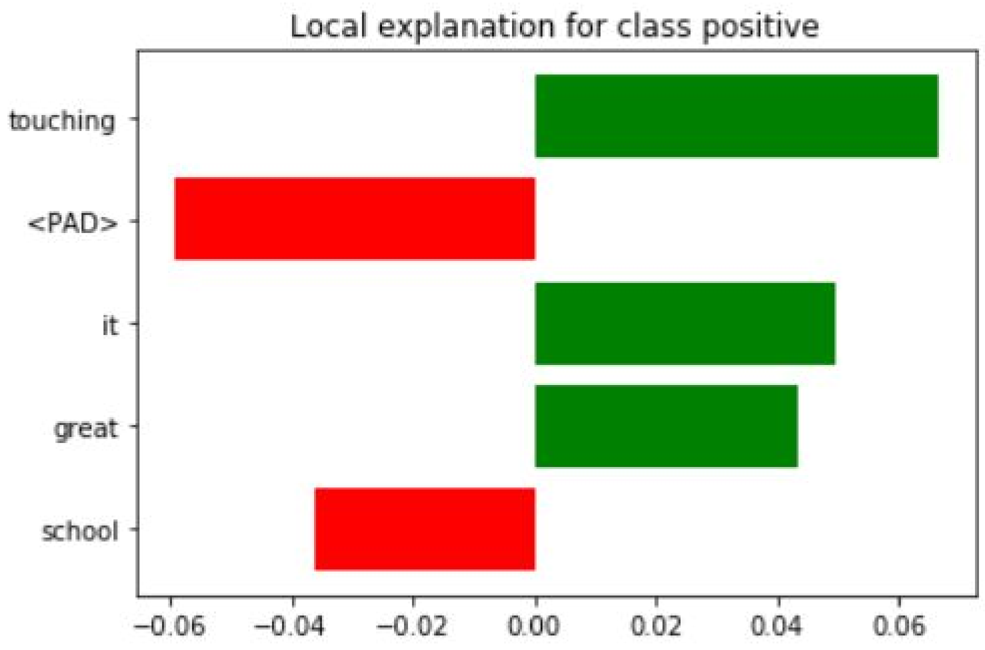


Fig 8.18 LIME explanation for the noisy text

8.2.2 Shap explanations with IMDB

Now, let's look at Shap explanations for the same texts. In fig 8.19, we see shap explanation for the original text. The words "great" and "touching" seem to be the most positive influences on the prediction.

```
Model's prediction -> Negative: 0.0000, Positive: 1.0000
```

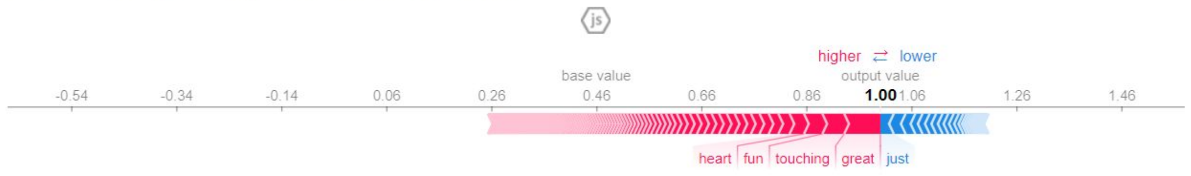


Fig 8.19 Shap explanation for the original text

Below, we can see the explanation for the noisy text. The shapley values seem to have changed for the top influencers indicating sensitivity to noise and therefore lack of robustness.

```
Model's prediction -> Negative: 0.0000, Positive: 1.0000
```

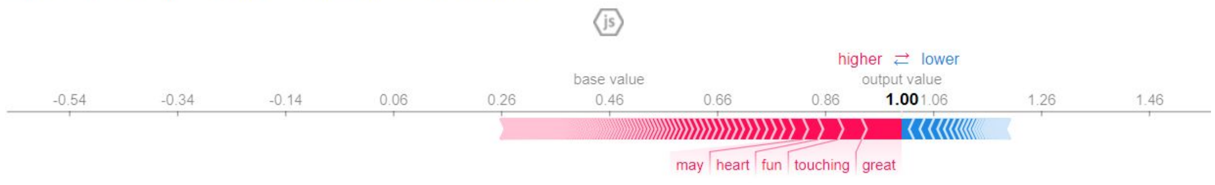


Fig 8.20 Shap explanation for the noisy text

8.3 Clustering Based Grounding Technique for Self Explanations

The entire self explanation model was trained end to end and the following performance metrics were noted :

1. **Training**
    a. Training Accuracy : 0.9797
    b. Training F1 Score : 0.979701
    c. Training Recall : 0.979714
    d. Training Precision : 0.97972
2. **Validation**
    a. Validation Accuracy : 0.9728
    b. Validation F1 Score : 0.97281
    c. Validation Recall : 0.97283
    d. Validation Precision : 0.972839
3. **Testing**
    a. Testing Accuracy : 0.9771
    b. Testing F1 Score : 0.97709
    c. Testing Recall : 0.9797

d. Testing Precision : 0.9772

The outputs were divided into a set of explanations to make the understanding easier. The first one was the set of grounding images which were similar to the current image in terms of the encoding and the relevance matrix. This was obtained by the intersection of the two clusters, the test image's encoding and relevance matrix were assigned to. The second output was with respect to the robustness of the model. The squared error between the encodings of the original image and the noisy image and between the relevance matric of the original image and the noisy image are depicted as a measure of stability. Along with this, the heat map of the change in the relevance matrix and the concept activated in the encoding along with its value is given.

Consider the following test image.

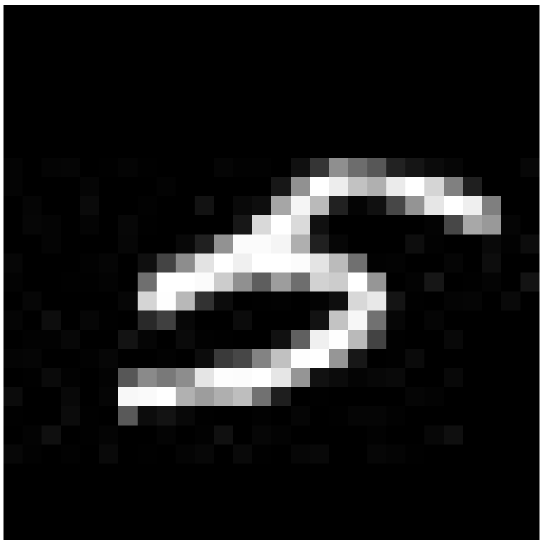


Fig 8.21 Test Image

The output class scores for this image are given by the following graph.

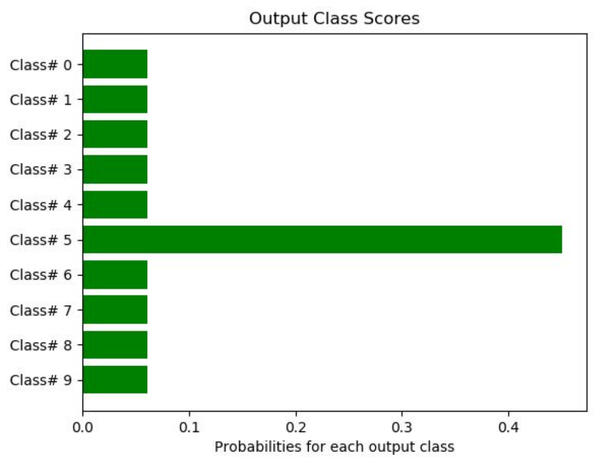


Fig 8.22 : Output class scores for the given test image

In order to measure the stability of the model, we create a noisy image by changing the test image by a minute amount by adding noise. The noisy image is given below.

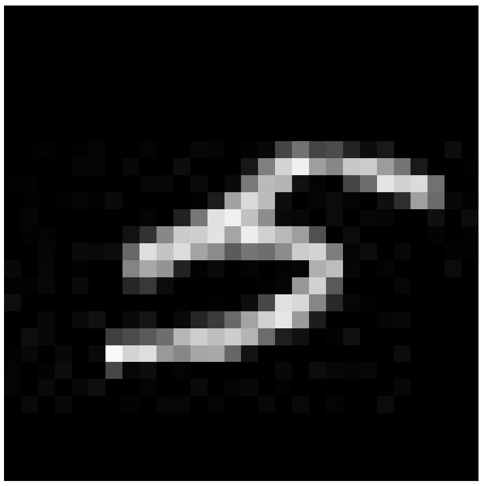


Fig 8.23 : Noisy Image

The next result is the set of images that were similar in terms of the encoding and the relevance matrix to the original image.

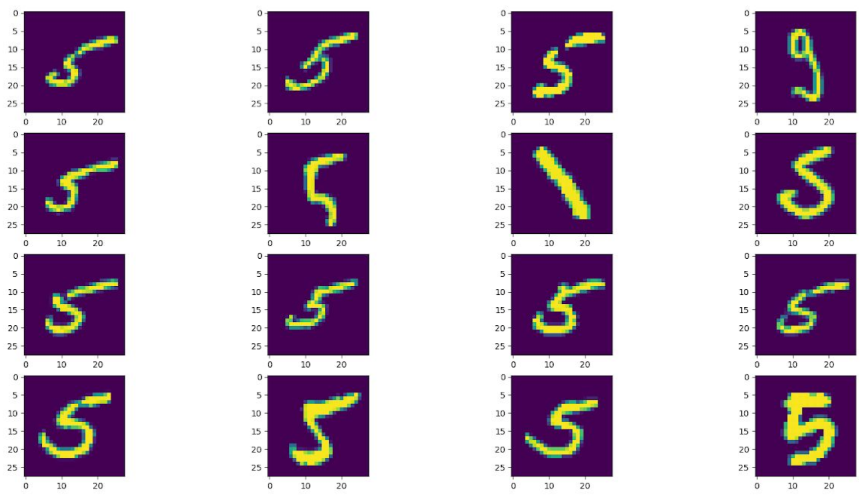


Fig 8.24 : Grounding Elements obtained by the Intersection of the two clusters

As per the model, each image activates one concept maximally. This is based on the encoding that is obtained for the test image. For the explanation to be stable this encoding must be similar for the original image and the test image. The following graph depicts the concept encoding value for the original image and the test image.

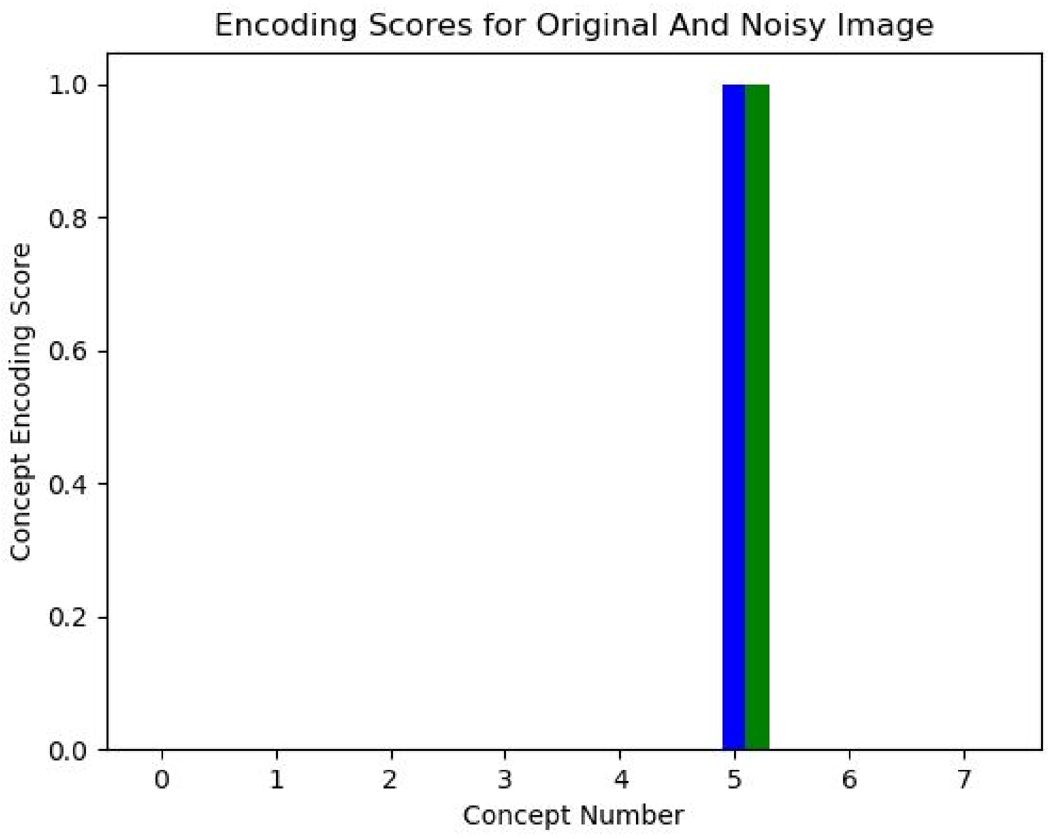


Fig 8.25 : Encoding Concept Scores for Original Image(Blue) and Noisy Image (Green)

We see that both the images activate the 5th Concept. To understand this better, we depict the set of images that maximally activate each concept. This is shown in Fig 8.25 and as we can see, the 5th concept contains images that are of the same class as the given test image.

The squared error between the encoding for the original image and the noisy image is 0.002201667406537006. This is very less and thus given that the error is close to zero, this proves that the encodings are very similar for the two images thus proving its robustness in terms of the encodings.

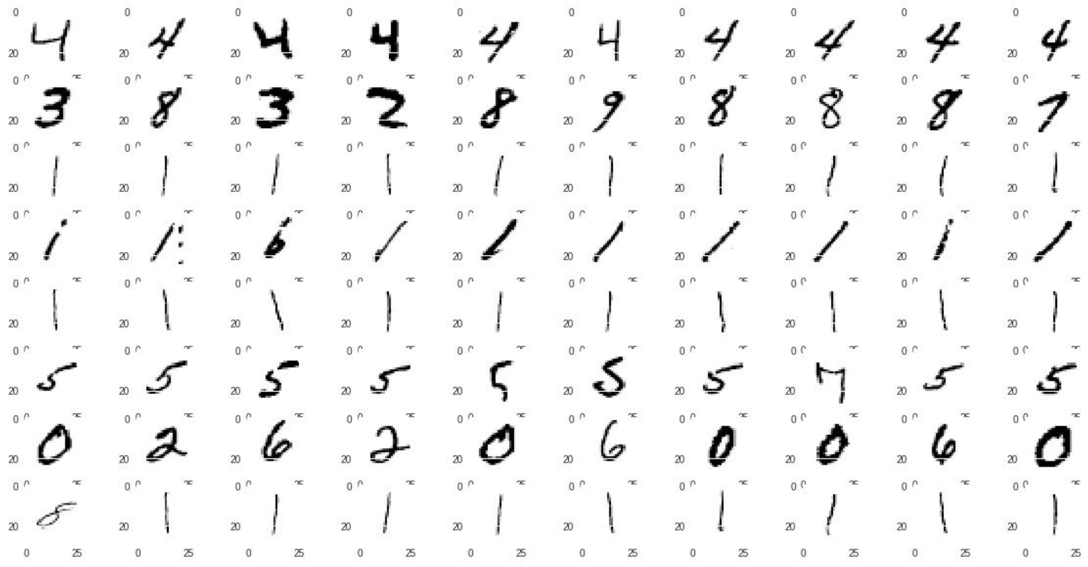


Fig 8.26 : Concept Images for each Concept represented Row Wise

The next step is to prove that even the relevance matrices are robust with respect to slight change in the input. For each output class, there is one column in the relevance matrix that has a high value resulting in the corresponding output. This is like choosing the influence values for each concept in the prediction probability for a particular class. Given that the output class for the test image depicted here is 5, we take the 5th column and see how different each of the relevance scores for the concepts are for the original image and the noisy image.

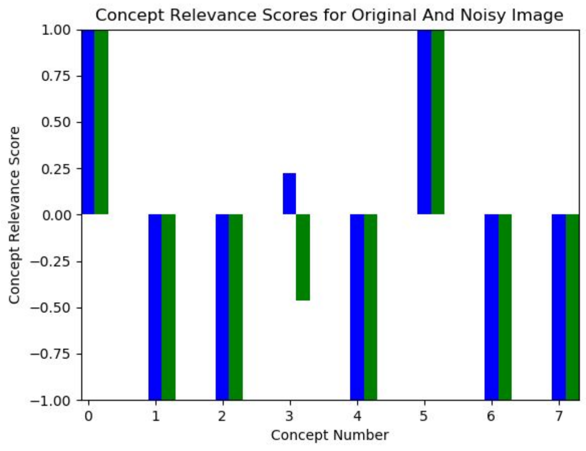


Fig 8.27 : Concept Relevance Scores in the prediction of Output class 5

As we can notice again concept 5 has a high relevance score which is in terms with what was expected. The other thing to notice is that the scores are almost the same for both the original image and the noisy image. To check if this is the case for the entire relevance matrix, we take the difference of the two matrices and represent it as a heat map.

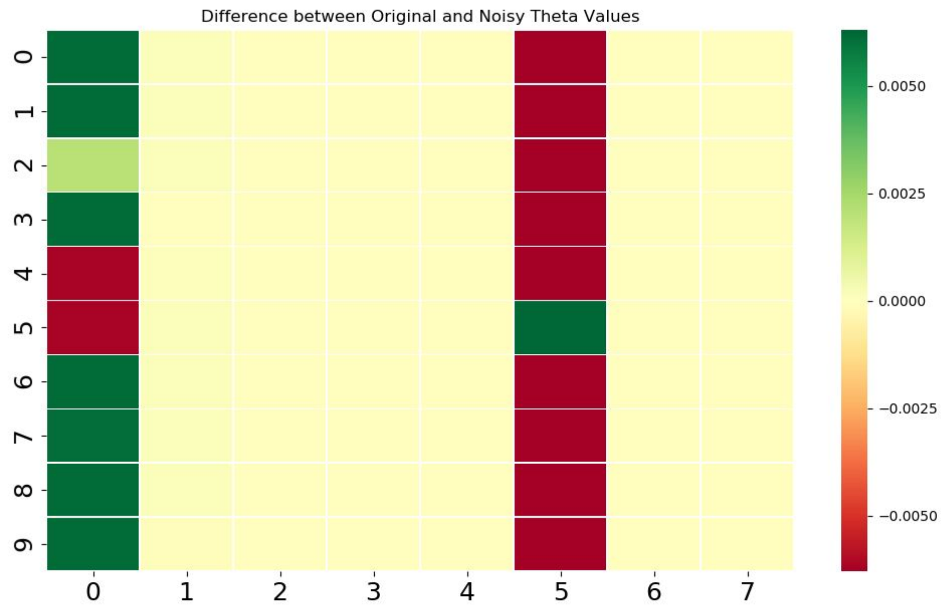


Fig 8.28 : Heat Map of the Difference between the Relevance matrix for the original image and the noisy image.

The relevance matrix difference is also very small in the order of 0.0050. Also the squared error between the two relevance matrices is 0.17096562835822213. This again proves that the relevance matrix is also robust to change in the input by a small margin.

As the original image's cluster is decided based on the encoding and the theta value, proving that the two has very similar encodings and the theta values indirectly proves a similarity in terms of the output set of grounding images. This proves the robustness of the model. These set of grounding results are easy to understand also.

CHAPTER 9

CONCLUSIONS

By performing the task of developing posteriori models like LIME and SHAP on the already trained models for Image Classification and Sentiment Analysis, it was concluded that :

1. These models needed to be built locally for each test sample, which is a tedious and time consuming process.
2. These approaches are not robust with respect to small changes in the input as the stability values indicated that the explanations change by a large margin for a small change in the input.

To overcome these effects, the new approach proposed through this project called Clustering based grounding techniques of self explanations for interpretations performed better and the following was concluded :

1. The training process and the interpretable model learning process happen simultaneously during training. This removes the need for training the interpretable model separately for each test sample and thus saves a lot of time.
2. When compared to the original self explanation models in which the results were harder to interpret for a layman, this clustering based technique provided results which were very simple to understand from a layman's perspective.
3. These explanations provided were on the basis of the encodings and the relevance matrix for each test sample. It was proven that both of these are robust to small change in the input. This concluded that the interpretable model was thus stable and much better than the posteriori based approaches.

CHAPTER 10

FUTURE WORK

The obvious extension to this would be to test the same architecture for problems in a different domain such as Natural language processing tasks such as sentiment analysis[10]. Input in the form of text can also be represented by a vector of concepts and can also have a relevance matrix with respect to a task like sentiment analysis. Thus, given that the concepts used in this architecture are almost directly applicable to text based tasks.

Using different architectures for the encoders, relevance parametrizar and the additive function could enhance the performance and also ensure that the proposed approach could be extended to complex tasks in the same domain. For example, to apply the same approach to a very complex dataset like say Image-Net[11], usage of large size neural networks like ResNet[12], VGG16[14], Inception Network[13] for the sub blocks in the architecture can be a possible extension.

In our proposed approach, clustering was performed using K Means and did not have a separate loss function included. This could be changed by adding a clustering at layer[15] at the end of the encoder layer and the relevance block. This would mean that a clustering loss would be added to the loss function and now the clustering training would also happen end to end in conjunction with the learning of the classification output and the interpretable outputs.

CHAPTER 11

REFERENCES

1. Lecun, Yann & Haffner, Patrick & Bengio, Y. (2000). Object Recognition with Gradient-Based Learning.
2. Miller, Tim. "Explanation in artificial intelligence: Insights from the social sciences." arXiv Preprint arXiv:1706.07269. (2017)
3. Tulio Ribeiro, Marco & Singh, Sameer & Guestrin, Carlos. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. 97-101. 10.18653/v1/N16-3020.
4. Lundberg, Scott & Lee, Su-In. (2017). A Unified Approach to Interpreting Model Predictions.
5. Alvarez-Melis, David & S. Jaakkola, Tommi. (2018). Towards Robust Interpretability with Self-Explaining Neural Networks.
6. D. Alvarez-Melis and T. S. Jaakkola. "On the Robustness of Interpretability Methods". In: Proceedings of the 2018 ICML Workshop in Human Interpretability in Machine Learning. 2018. arXiv: 1806.08049.
7. Baldi, Pierre & Guyon, G & Dror, V & Lemaire, G & Taylor, D & Silver, Daniel. (2019). Autoencoders, Unsupervised Learning, and Deep Architectures.
8. K. Jain, Anil & Murty, M & J. Flynn, Patrick. (1999). Data clustering: a review. ACM Comput Surv. ACM Comput. Surv.. 31. 264-323. 10.1145/331499.331504.
9. Nwankpa, Chigozie & Ijomah, Winifred & Gachagan, Anthony & Marshall, Stephen. (2018). Activation Functions: Comparison of trends in Practice and Research for Deep Learning.
10. Alvarez-Melis, David and Tommi S. Jaakkola. "A causal framework for explaining the predictions of black-box sequence-to-sequence models." *EMNLP* (2017).

11. Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. *IJCV,* 2015.
12. He, Kaiming et al. “Deep Residual Learning for Image Recognition.” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016): 770-778.
13. C. Szegedy *et al*., "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 2015, pp. 1-9.
14. Simonyan, Karen and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” *CoRR* abs/1409.1556 (2015): n. Pag.
15. Guo, Xifeng et al. “Deep Clustering with Convolutional Autoencoders.” *ICONIP* (2017).

CHAPTER 12
APPENDICES

APPENDIX A : CIFAR 100 TESTS FOR LIME AND SHAP

12.1 Experiments on CIFAR100 to prove non-robustness of LIME and SHAP

While checking for the robustness of LIME and SHAP, experiments were performed on the CIFAR 100 dataset to check if the models were robust for complex datasets or not. The following depict the results from the same.

A CNN Model was trained to classify images from the CIFAR 100 dataset and LIME and SHAP models were built on top of these models.

12.1.1 LIME Results from CIFAR 100

The following image represents the test image from the Camel class.

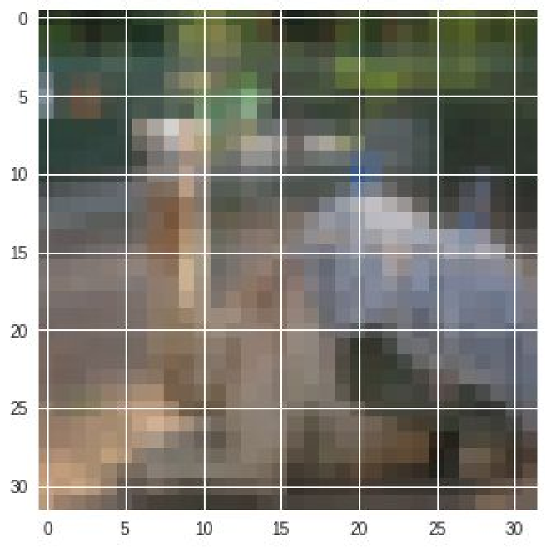


Fig 12.1 Test Image belonging to the Camel Class

The following image represents the corresponding noisy image. That is, a slight amount of noise was added to the test image to obtain the following image.

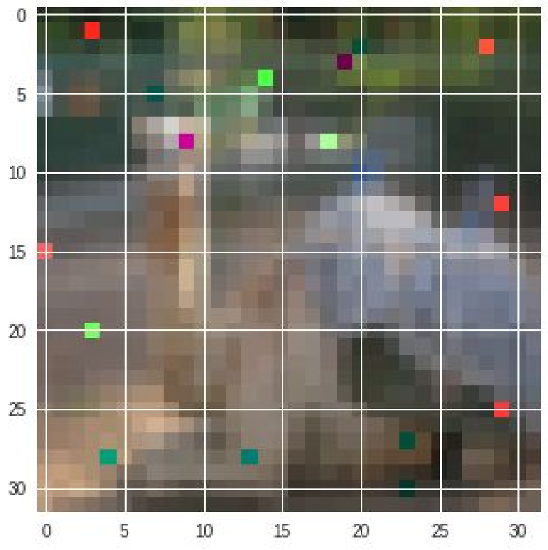


Fig 12.2 Noisy Image

The following figure gives us the original image and its explanation.

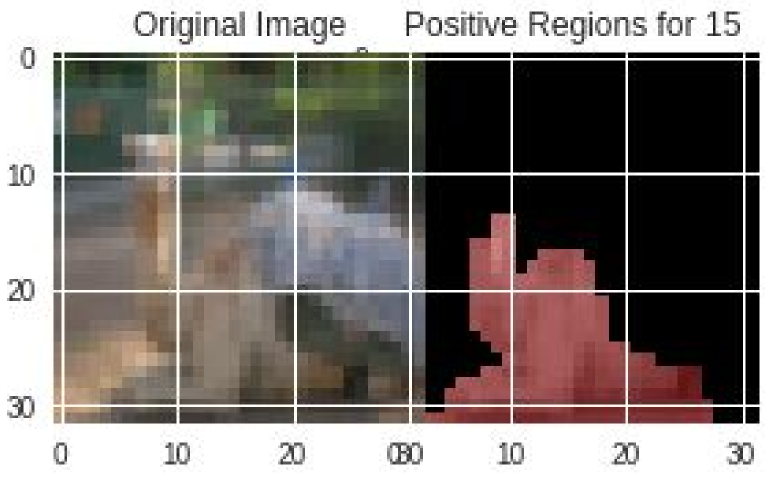


Fig 12.3 Original Image and its explanation

The following figure gives us the noisy image and its explanation.

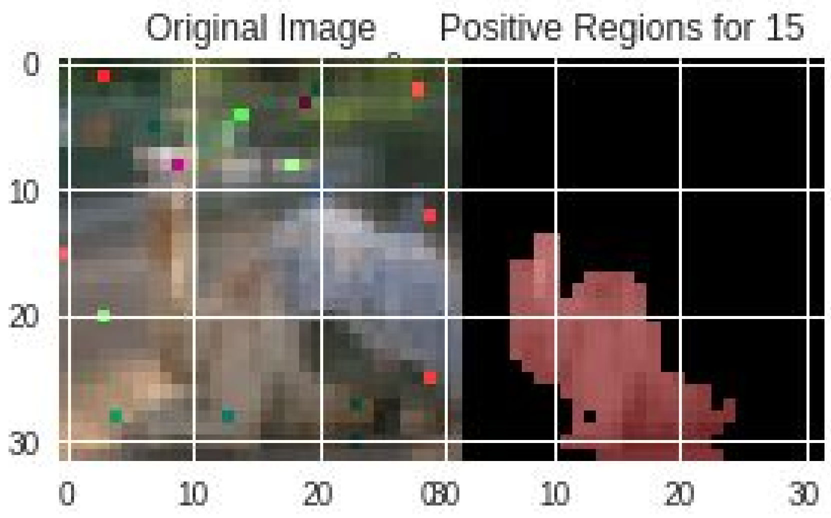


Fig 12.4 Noisy Image and its explanation

Next, we compare the original image and the noisy image, the original image explanation and the noisy image explanation.

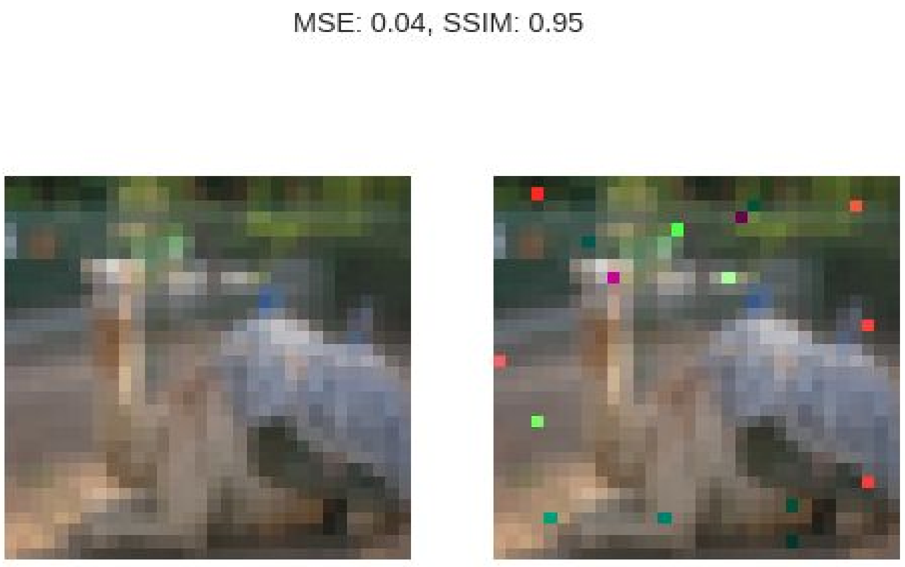


Fig 12.5 Comparison between original image and the noisy image

MSE: 0.76, SSIM: 1.00

Fig 12.6 Comparison between the explanations for the original image and the noisy image.
As we can notice, for a MSE of 0.04 between the original image and the noisy image, the explanations had an MSE of 0.76. This indicates the non robustness of the LIME model.

12.1.2 SHAP Results from CIFAR 100

The following image represents the test image from the Apple class.

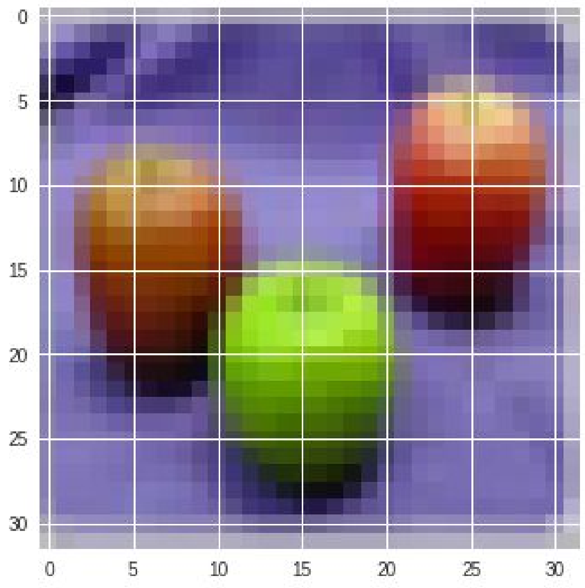


Fig 12.7 Test Image from the Apple Class.

The following image represents the corresponding noisy image. That is, a slight amount of noise was added to the test image to obtain the following image.

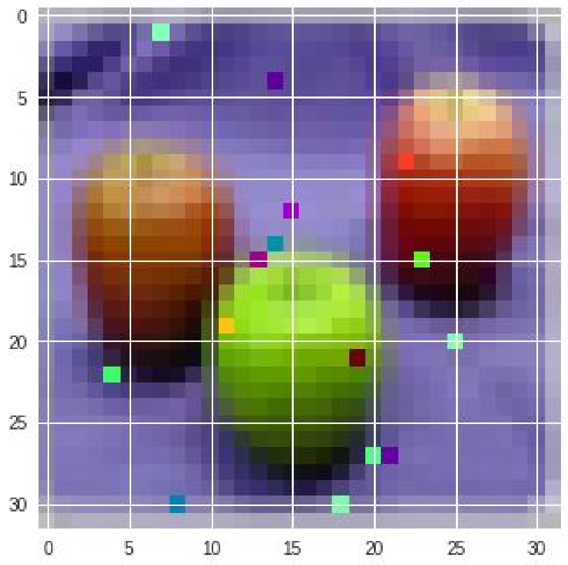


Fig 12.8 Noisy Image

The following figure gives us the original image and its explanation.

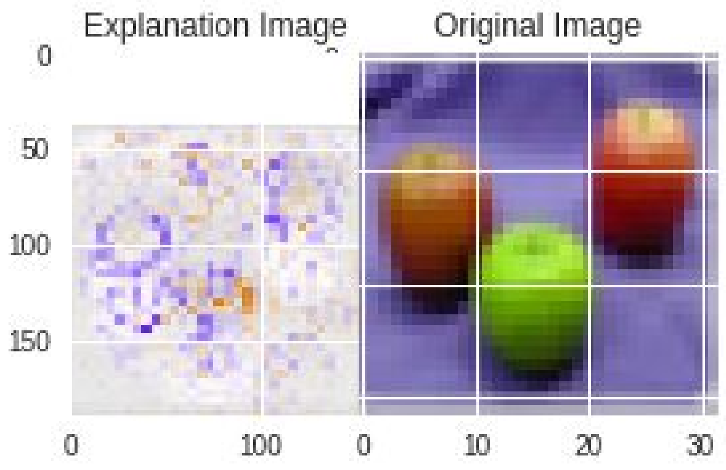


Fig 12.9 Original Image and its explanation

The following figure gives us the noisy image and its explanation.

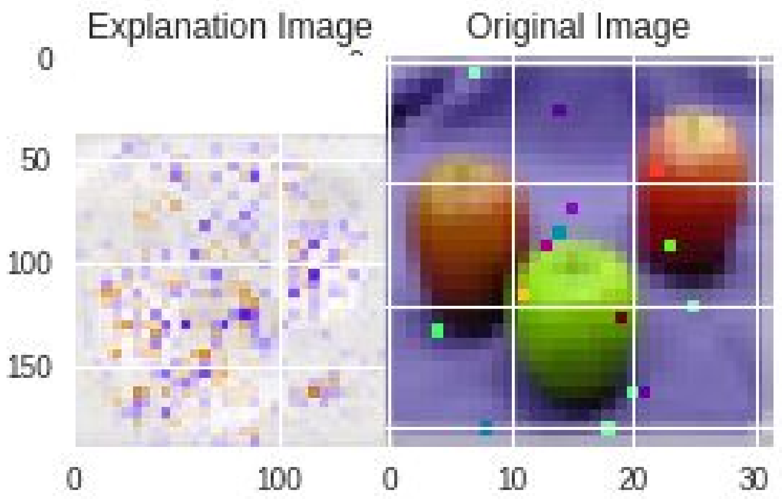


Fig 12.10 Noisy Image and Its Explanation

Next, we compare the original image and the noisy image, the original image explanation and the noisy image explanation.

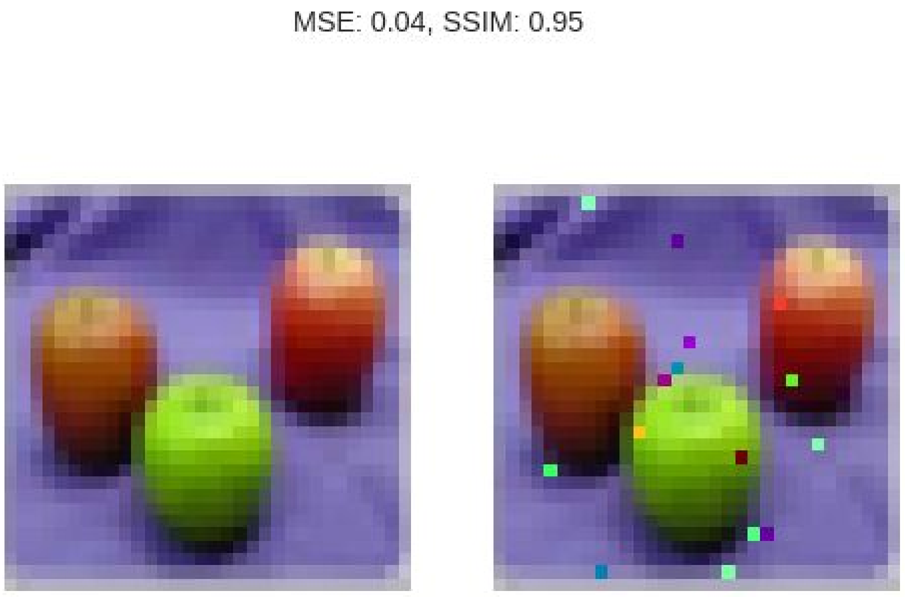


Fig 12.11 Original Image and the noisy image

MSE: 7.57, SSIM: 0.43

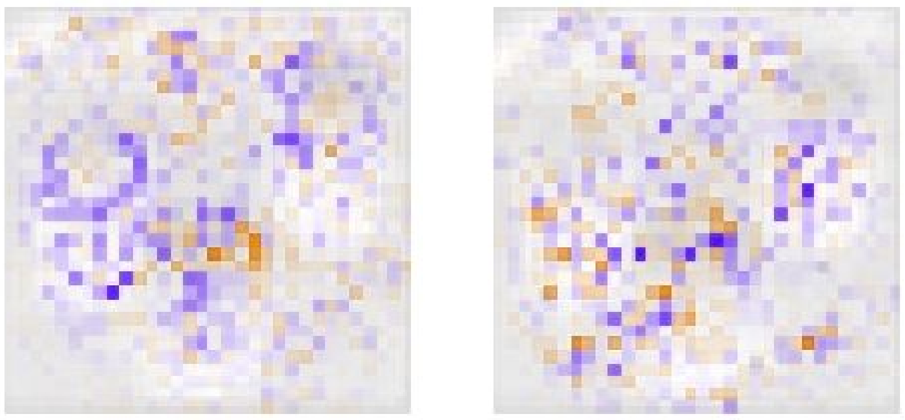


Fig 12.12 Original Image Explanation and Noisy Image Explanation

As we can notice, for a MSE of 0.04 between the original image and the noisy image, the explanations had an MSE of 7.6. This indicates the non robustness of the SHAP model.