Using Weight Matrices to build a Polynomial Representation to assist with Interpretability and Explainability of Deep Learning Models

---

Selwane Nthabiseng

Supervisor(s):
Ms N Ndebele

A research proposal submitted in partial fulfilment of the requirements for the degree of Master of Science in the field of e-Science

in the

School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg

7 August 2020

Declaration

I, Selwane Nthabiseng, announce that this proposition is my claim, unaided work. It is being submitted for the degree of Master of Science in the field of e-Science at the University of the Witwatersrand, Johannesburg. It has not been submitted for any degree or examination at any other university.

Selwane Nthabiseng

7 August 2020

Abstract

Deep Learning (DL) models use representational learning to discover the representation for classification and detection. Learned representations present a problem because the representation is encoded in such a way that cannot be explained using human intuition and known mathematics, thus not allowing us to reason out and interpret results from DL models. Knowledge learned by a neural network is internally stored in the weights. In this research project, we present a novel approach to building a polynomial representation of the learned function using the weight matrices. We were able to represent the learned function and also use convolution to map and to visualise the behaviour of the model. A popular explanation method, Local interpretable model agnostic explanations (LIME) was also used to obtain an understanding of predictions made using the Feedforward neural network (FFNN).

Acknowledgements

My sincere thanks goes "DST-CSIR National e-Science Postgraduate Teaching and Training Platform (NEPTTP)" for the funding. I would like to thank my supervisor Ms N Ndebele and my advisor Mr K Masemola for their contributions, hard-work, encouragement and advice on my research for these masters.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Meaning
NID	Neural Interaction Detection
DL	Deep Learning
DNN	Deep Neural Network
GPU	Graphics Processing Unit
CPU	Central Processing Unit
Deep CNN	Deep Convolutional Neural Network
CNN-INTE	Convolutional Neural Network Interpretation
RETAIN	REverse Time AttentIoN
LIME	Local Interpretable Model-Agnostic Explanations
NN	Neural Network
BDl	Bayesian Deep Learning
LRP	Layer-wise Relevance Propagation
BETA	Black Box Explanation through Transparent
ML	Machine Learning
YOLO	You Only Look Once
FFNN	FeedForward Neural Network
NN	Neural Network
SVD	Singular value decomposition

This is dedicated to my family, My husband Moloko, my son Neo Manthata and my parents.

1 Introduction

Machine Learning (ML) is the technology of having computers to act without having to write specific set of the instructions for the machine [1]. Computers have the ability to learn using ML algorithms or models, ML algorithms can be explained as mathematical expressions used to represent data in a context of a problem and gain insights from the data [2]. Neural Networks (NN) are a class of models within the general machine learning literature.

NN architecture was modelled after the neuronal structure of the animal brain [3]. NN are made up of many straightforward connected processors known as neurons, each producing a sequence of real-valued activations. Input neurons are activated through the settings, while other neurons are activated through weights of the previous layer. There are two types of NN namely, single layer perceptrons and multi-layer perceptrons. A single-layer perceptron is a linear classifier that is used for binary classification, a multi-layer perceptron is used in learning and identifying non-linear patterns [4].

Deep Learning (DL) is an extension of multi-layer perceptrons and it doesn't depend on the hand-crafted representation. DL uses representational learning to discover the representations required for detection or classification [5]. The DL model discovers layers of representation jointly simultaneously instead of one at a time. In joint feature learning the model automatically adjusts all its internal options when one is adjusted while not requiring human intervention [6]. Traditional ML uses hand-crafted features to get the desired output. The learning process in DL is an automated process for finding better representation. One advantage of learned representation is that it has better performance compared to predefined learned features [7].

The adoption of the DL methods was not popular when they were proposed by Frank Rosenblatt in 1957 [8], even though his focus was on the hardware, his work laid a foundation for the development of DL. This changed after the availability of faster Graphics Processing Units (GPUs) and large labelled data sets [9]. Real-world data is often challenging to work with since it is more complex and often can suffer from the corruption that may affect data interpretations, data processing, classifiers and models generated from data as well as decisions made based on data.

DL methods have been successful in image processing and natural language processing, because they do not use hand-crafted features. Despite their impressive accuracy, the DL models are treated as black-box approximators, which fulfill a nonlinear mapping from an

input variable to a corresponding output variable [10]. These methods are now being applied in different fields such as health sciences for medical diagnosis, planning and control, legal and justice for profiling cases (legal documents) and human profiling. This increases the need for human beings to be able to understand the inner workings of the model [11]. European General Data Protection Regulation (EGDPR) specifies that all the decisions made using ML models especially relating or affecting individuals must be explainable, which increased the demand for the interpretability and explainability of the black-box models [12].

Model interpretability is tied in with having the option to make-out the mechanics without fundamentally knowing why. It is also characterised as the measure of reliably foreseeing a model's outcome without endeavouring to know the explanations for the scene. It is simpler to know the purpose of specific choices or forecasts if the interpretability of an ML model is higher. While model explainability provides the information to truly clarify what is going on. In essence, it is the comprehension of the inquiry "Why is this happening ?". Over the years methods such as Local Interpretable Model-Agnostic Explanations (LIME) [13], Reversed Time Attention Model (RETAIN) [14], Layer-wise Relevance Propagation (LRP) [15], Black Box Explanation through Transparent Approximations (BETA) [16], and others have been developed to explain and interpret black box models.

Learned representations do present a problem, because the representation is encoded in such a way that cannot be explained using human intuition and known mathematics, thus not allowing us to reason out and fully interpret results with understanding of how they were obtained. In this project we explore a method to extract the representation into a pure mathematical representation which will afford us the opportunity to analyse the inner workings of the model.

Polynomial interpolation is concerned with reproducing the learned function from the Feed-forward neural network (FFNN) as a numerically analysable function at a lower dimensional space. One might question why don't we attempt to model the problem using these numerical tools from scratch without going through ANNs. The answer is that ANNs are well suited at learning complex functions of unknown origins, then it becomes easier to extract the already learned function and analyse it further.

The aim of this project is to expose the inner working of the learnt function of a deep neural network. A way to map the training process is proposed, which stores weights at each epoch then builds polynomials from the first principal component of each. After a polynomial representation has been built, mathematical tools are available to assess the training process.

We applied basic analysis on the polynomial to try and interpret the deep learning models training process. The one method used was to map out how each epoch's function affected the transformation of the next epoch by taking the convolution of the first epoch with the second epoch which is representative of the transformation that has occurred. This gives an

advantage over the tree surrogate technique because tree surrogates only explain the final black box, without giving explanation to how the black box has evolved from random matrices to the learned function.

Addition of explaination or interpretability of DNN, will not only increase the trust in the model but also adoption of them. The contribution from this project is the proposal of a method that will help in understanding the general behaviour of the model instead of explaining a particular instance. This work will help in improving the overall value of the DNN. This method can be used in different scenarios in different fields.

The thesis is structured as follows: Chapter 2 is separated in two sections: the background which gives an overview of several standard machine learning procedures such as CNNs and DNNs, explanation to interpretability and explainability and related work where different methods are reviewed. Chapter 3 gives detail on the approach and datasets used. Chapter 4 gives a description and analysis of the results and Chapter 5 is the conclusion and future work in the expansion of DNN explanation.

1.1 Research Aim

In this project we looked at the classification problem of 3 datasets using Feedforward neural networks (FFNN). The aim of this project is to translate a deep learning model representation into a pure mathematical representation. This will be used to analyse the internal mechanism of the model and expose the inner working of the learnt function of a deep neural network. A method of mapping the training process is proposed that stores weights at each epoch and thereafter builds polynomials from the first principal component of each. The polynomial function is then used to explain and interpret the results of an ML model using functional analysis.

1.2 Research Questions

- Can the proposed technique be utilised to give the behavioural extrapolation and give usable knowledge into deep network operation enabling system behaviour understanding and improvement?
- Can we explain how the model works, and how it reaches certain conclusions?

Hypothesis: Given a DL model, we can model the change in weights during the training process, by building an Nth degree polynomial and analysing the resultant.

2 Background and Related work

2.1 What is Deep Learning?

Deep learning is a mathematical framework for learning representation from data. It gained popularity from the year 2013. In deep learning feature extraction is included as part of their learning process [7]. DNN works as a multistage information extraction process, where data goes through sequential filter and returns some information regarding the task [17]. The data will be represented in multiple levels where the higher level features will represent an abstract explanation of the data. This abstract representation makes the deep network more robust to intra-class variability [18]. In order to get high-level features, the low-level features are extracted first. For example when using DNN in images classification or detection, local combinations of edges create a certain pattern, which are used to construct parts and parts result in objects [19].

A Feed Forward Neural Network (FFNN) takes an input X and tries to map it to the correct target value Y. These mappings happen through a series of simple data transformation layers that are learned through the exposure to examples or training, this mapping can be defined as "$Y = f(X, \theta)$ and learns the value of the parameters θ that results in the best function" [20]. These networks are called feed forward because information flows in the forward and are such that X is used calculate some intermediate function in the hidden layer which in turn is used to calculate Y [21]. Forward propagation passes input data in the forward direction through the network [22], back propagation allows the information to be passed back into the network to adjust the weights of the network and minimise the distance between output score and desired target score [23].

The process of training optimises the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs. Figure 2.1 shows how the learning process is implemented in DNN. The weights (which are numerical values) on each layer stores specifications of what the layer does to its input data [17]. To correctly map the input to the targeted value, one must determine the right set of values for the weights of all layers in a network because a network may contain many weights. This task might seem a little complicated because changing a single parameter value will affect the whole network. To be able to control the output of a network, we must be able to measure how far this output is from the expected results, this is achieved by using a loss function [17]

Loss functions are used to evaluate a set of parameters also known as a candidate solutions [6]. In DL a loss function has an important job in that it must faithfully distil all aspects of the model down into a single number in such a way that improvements in that number are a sign of a better model [6]. The loss function computes the distance score between the output of a model and the target value, this shows how well the model performed and this score is used as a feedback signal to adjust the weights. Figure 2.1 shows process of weight adjustment. Back-propagation is used to adjust the weights by looking for the minimum error function in the weight space using the gradient descent method. This method looks for a combination of weights that minimises the loss function and uses that as a solution for learning problem [24].

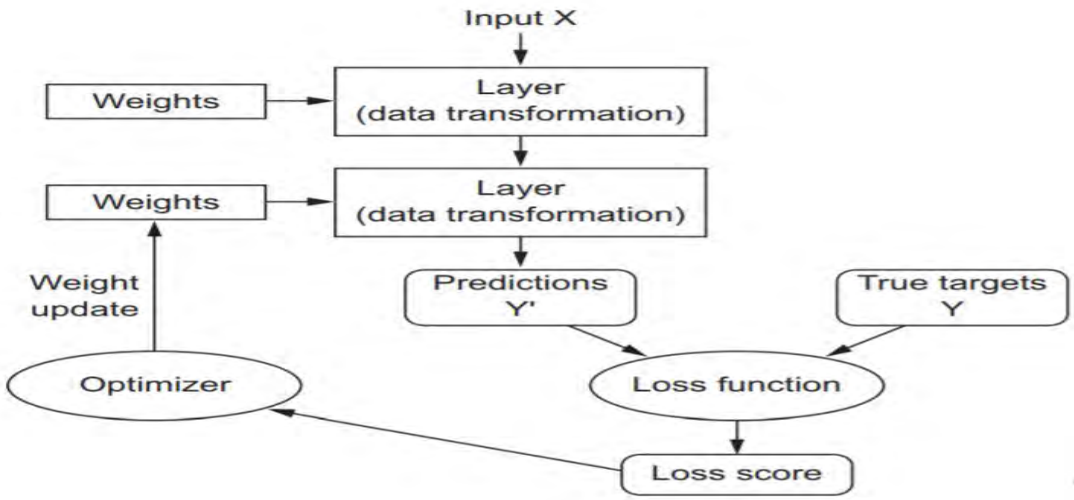


FIGURE 2.1: Training of Feedforward Neural Network: [25]

NN are made up of neurons, these neurons are ordered into layers figure 2.2 shows the structure of NN, the first layer called input layer, hidden layer and output layer. The neurons are connected to at least one neuron from the previous layer and every connection is measured by a real number called the weight coefficient, which shows the degree of importance for a given connection in a NN.

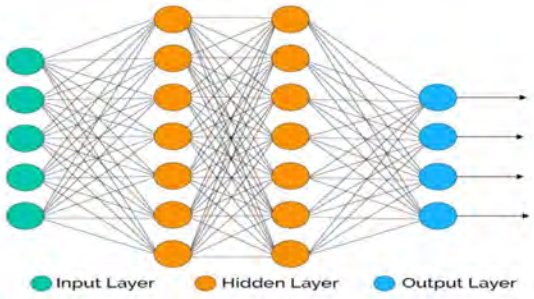


FIGURE 2.2: Architecture of Neural Network: [26]

The input layer collects patterns in the input data, the output layer produces results of a given input and hidden layers can extrapolate salient capabilities inside the input data that have predictive power regarding the outputs [27]. There are different types of NN but they

all have similar principles.

Convolutional Neural Networks (CNNs) have become popular and are considered as powerful tools in computer vision related tasks such as image classification, object detection, face recognition and scene labelling, because of their performance compared to other models. When training a DL model for an image, the image is taken as input and passed to layers of the model for feature extraction. CNNs neurons are structured in a manner similar to a 3 dimensional shape having width, length and depth [28]. CNNs are powerful because of their ability to extract important features at different levels of abstraction[29]. CNN learns higher-order features in the data using convolution by transforming input data from the input layer through all connected layers into a set of the class score given by the output layer. CNN's are organised into three main layers, the input, feature learning and classification layers

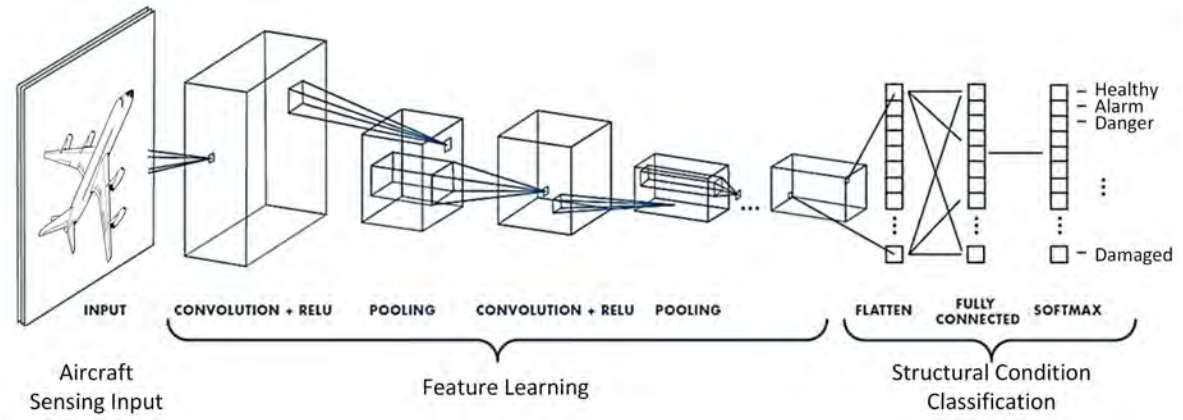


FIGURE 2.3: Covolutional Neural Network: [30]

The input layer takes in input data and stores it before it gets processed. This data is stored in a three dimension width, length and depth. It is easy to scale a CNN compared to the traditional NN because neurons in a layer are not fully connected to the neurons from the previous layer.

The feature learning layer consists of a combination of layers, that is repeated in a sequence of convolutional layer and pooling layer. The convolutional layer extracts features from an input image and maintains dimensional relationship between pixels by learning features of an image using small squares of input data. Given a 5×5 image, to find a feature map of a 3×3 matrix we slide a filter or kernel over the original image using a skipping factor of 1 pixel (skipping factor specifies the number of pixels you want to skip), for every position. The dot product between the two matrices is computed an multiplication outputs added to get integer value which forms a single element of the output matrix. The filter or kernel sees the only part in the current stride and acts as a feature detector from the original image. This convolution gives different feature maps because this operation captures the local dependencies in the original image [31]. The output size of the map is defined by

$$M_x^n = \frac{M_x^{n-1} - K_x^n}{S_x^n + 1} + 1; M_y^n = \frac{M_y^{n-1} - K_y^n}{S_y^n + 1} + 1 \tag{2.1}$$

where M is the feature map and S is the skipping factor with x and y denoting the number of pixels to be skipped in the horizontal direction and vertical direction respectively.

Convolution is a linear operation, so to account for non-linearity we use Rectified Linear Unit (ReLU). Most of real world data is non-linear it is important to use a non-linear operation to introduce non-linearity in the CNN. ReLU is used after every convolution operation [31]. The pooling layer is responsible for merging similar features into one by evaluating the maximum of a local patch of units in one feature map, to reduce computational power [32]. Pooling reduces the dimensionality of each feature but retains the most important information [31].

The classification layer is a fully connected layer, this layer maps the score of each image to the score of the class by connecting neurons on the previous layer to all neurons on the current layer [33].

Deep learning has proven to be highly accurate when used for supervised learning. DL models depend on the amount of available labelled data for training [34]. Using transfer learning is one of the solutions used when there is not enough labelled training data for the classification purpose. These models are already trained on large labelled data sets by learning information about object characteristics that is enough to transfer to the objects that are not in the original data set [35]. These methods are used for unsupervised image classification in CNN. Deep learning models are often referred to as black box approaches as they are able to accurately classify large data sets without showing how they arrive at conclusions. This is a disadvantage especially if you are unable to explain the classified data points.

Before the implementation of Graphics Processing Unit (GPUs), DNN took a long time to train using Central Processing Unit (CPUs), this was one of the challenges during the initial development of DNN. GPUs are trained online and where the weights are updated after every error backpropagation step [36],[37]. The adoption and implementation of the DNN is dependent on fast and powerful hardware.

2.2 Interpretability and explainability

Over the years, DNN models have shown an impressive accuracy in object detection and classification, one major limitation of the DNN is understanding the internal operations and behaviour of the model [38]. Sometimes we look for more than good accuracy in a model, to be able to trust the decision made by a model we should be able to understand how it reaches a certain conclusion and be able to explain its errors [39]. In classification and detection tasks it is important to prove that the high accuracy is the result of using proper problem representation instead of exploitation of artefact's in data. DNN had been shown misclassify inputs without a resemblance to the proper class.

In 2015 Google and Android released an app that can be used to tag and arrange photos automatically. The photo app was meant to store your pictures in albums, this would make the storage and searching for pictures easy [40]. This application was able to identify some pictures correctly and arranged them in right folders while other pictures were not correctly tagged and arranged in wrong folders [40]. In China, the police department uses facial recognition system to detect jaywalkers. The names and pictures of the jaywalkers are exposed and the perpetrators are then warned of the legal implications of jaywalking [41]. The disadvantage of this system is that it could not differentiate adverts and people. This led to a public shaming of a well-known business woman by accusing her of jaywalking after her bus advertisement was misinterpreted by the system.

Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) is used by some of U.S states to predict probability that defendants might commit future crimes. The results of this software were proven to be socially biased and unreliable [42]. This has resulted in the demand for techniques that will interpret and help understand what was learned by the model [43]. The ability to interpret and explain the DL models is important because of their application in the field such as self driving cars, diagnosis in medicine, loan approval in banks where the unseen failures might result in life-threatening actions or enormous economic loss.

Interpretability can be thought of as a science of understanding how the model works and what it does [19], by using the visual or textual presentation of the connection between input features and the output prediction. The objective of interpretability is to depict the internals of a framework in a way that is reasonable to people. The victory of this objective is tied with the cognition, information and inclinations of the client. Is the degree to which human can reliably anticipate the models comes about and it is also the extend which a user can understand the cause of a decision [44].

Model explainability is important for when we detecting weaknesses of a model and to check if there are any biases in the data. It is also used to justify models result and this can be useful when we want to improve the model or try to discover new information on the data. Explainability of the model also increases the end users trust in the model. The importance of this is shown in medical sciences where the doctors can not blindly trust the models without understanding how the models reach those conclusions [12]. To explain an event is to provide some information about its casual history.

2.3 Related Work

In this section we will look at other proposed methods for interpreting the DNN.

2.3.1 Network weights

In this section we will be looking at methods that use network weight to explain the models results. Network weights are described as the strength of the connection between the nodes. Weights suggest the importance of the input value and what sort of impact the input can have on the output [45].

Neural Networks are known for their ability to automatically extract features during learning process and to model complex statistical interactions. These interactions shows significant information about where features often have common impacts on other features on predicting an outcome. The NN's weights are nonzero in which all features interact blindly. Before the final output in (FFNN), any interacting features must follow strongly weighted links to a prevailing hidden unit. That is, in the corresponding directed chart, at least one common descendant will share interacting features[46]. There are generally two approaches used for interaction detection. First approach is to conduct individual tests for every combination of features [47], and the second approach pre-specifies all interaction forms of interest, then uses an regression analysis method to simultaneously determine important features [48].

Neural Interaction Detection (NID) detects the statistical interactions by interpreting the learned weights of an FFNN. In NID the interaction detection is made up of 3 steps which are generating interaction ranking, determining cutoff on interaction ranking and pairwise interaction detection. Training an FFNN with regularisation by applying sparsity regularisation to restrain insignificant interacting paths and push the modelling of major effects toward any univariate networks. We note that this method can additionally generalise beyond sparsity regularisation. In the second step, we interpret the learned weights to find the ranking of interaction candidates [46].

The framework has the practical utility of accurately detecting general types of interactions without searching an exponential solution space of interaction candidates. These methods provide theoretical justification on why interactions between features are created at hidden units and why hidden unit importance approximation satisfies bounds on hidden unit gradients. This approach extracts generalised non-additive interactions between variables from the weights of a neural network [46].

Sensitivity analysis is the examinations of how the shortcoming of the output of the scientific model can be isolated and distributed to different sources of instability in its input [49]. The features that form the definition can be augmented by pertinence scores showing the contribution for each feature. The clarification will be real-valued vector of the same measure as the input, where significant highlights are given positive scores and insignificant highlights are set to zero [43].

Sensitivity analysis may be an essential approach for understanding the relationship and

the impact of each input parameter on the yield of a issue. The key point behind sensitivity analysis is that by somewhat shifting each explicative input parameter and enlisting the reaction within the yield, parameters are explained with the most alevated affectability values picking up the most noteworthy significance. Sensitivity analysis of the foremost critical parameters can be exceptionally valuable for dissecting complex issues. Sensitivity analysis is picking up more significance due to the surprising capacity of NN to clarify the nonlinear connections between the explicative and reaction factors of a issue [50].

A common formulation of sensitivity analysis defines relevance score as:

$$S_k, l = \frac{\partial o_k}{\partial X_i} \tag{2.2}$$

Where S_k, l is the explanation, X_i is the input and o_k is the output. Sensitivity analysis uses a sensitivity matrix calculated based on the partial derivatives of the outputs of the network with respect to each input.

ANN are black box models therefore it is unclear on which features are useful in defining the key properties of the input vector class. This visualisation technique reduces the complexity of the problem by removing less important features without compromising the results of the model. The Garson Algorithm shows which inputs are important in the multiclass output classification problems [51].

.

2.3.2 Proxy models

In this we will be looking at surrogate models. These methods are aimed to estimate the predictions of the hidden black-box approach.

Permutation feature importance is defined as a model inspection technique that is used for any fitter estimator. This approach offers a distinctly compressed global insights into the models behaviour [52]. This method measures the importance of a feature by calculating the increase of a models prediction error after the feature is rearranged. A feature is said to be important if rearranging the values of that feature increases the models error, this proves that the model depends on that feature for the prediction. And a feature is said to be unimportant if rearranging its values does not affect the model's error [53]. This technique takes into account all interactions with other functions, through permuting the characteristic you also break the interaction outcomes with different features. By permuting the feature you furthermore might break the interplay consequences with different capabilities [53].

Local Interpret able Model-Agnostic Explanations (LIME) are another example of methods that can be used to understand how each feature contributes to the prediction. They are used to attempt to learn and understand why the models have made certain conclusions. LIME is one of the few methods that can be applied to images, tabular and text data. This method can be used to point out important regions or features of an input for decision across a variety of models and problem domains [54]. This method assesses what occurs to the

predictions while you deliver variations of your data into the machine learning model. This approach also generates a new dataset such as permuted samples and the corresponding predictions of the black-box [53].

In a tabular dataset LIME's output is a list of features 2.4 and their contributions in prediction of a data sample. This helps in determining which feature changes will have more impact on the prediction.

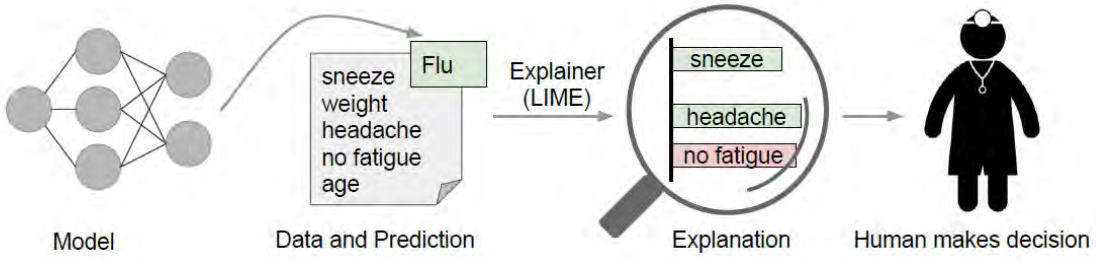


FIGURE 2.4: Tabular LIME:[55]

For images, since the interpretable space is a binary vector demonstrating the existence or absence of a super-pixel (important pixel). The procedure comprises of randomly making a portion of the super-pixels grey. At the end of the day, on the off chance that we need to clarify prediction for a picture, we will disturb the image and get predictions on images with other hidden super-pixels. This procedure is illustrated by the figure 2.4

LIME methods cannot be used for compliance scenarios, where you might be legally required to fully explain a prediction because it is not sufficient for complete attributions. These explanations are unstable, this instability makes it difficult to trust the explanations. The explanations of closely related points may have different explanations, and if the sampling process is repeated the explanation that comes out might be different [53].

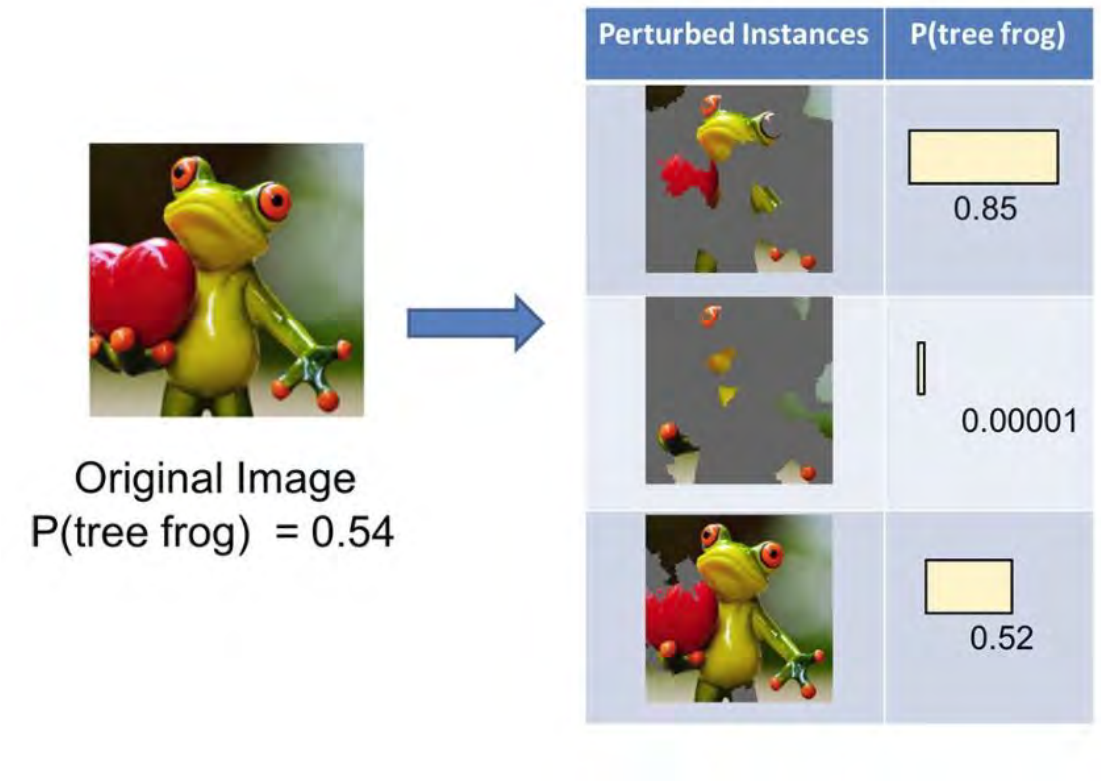


FIGURE 2.5: LIME for Images: [56]

SHapleyAdditive exPlanations (SHAP) is one of the methods that is used to give a value of importance for a feature for every prediction. Unlike LIME methods this method presents results that are consistently and locally accurate. The additive feature attribution method based on expectation. It is also explained as an integreted approach used to explain the output of any given ML model, it gives different combinations of features [57].

Deep Learning Important FeaTures (DeepLIFT) is a back-propagation based approach unlike perturbation methods such as LIME and permutation feature importance, this technique engenders the significance sign from an output neuron in reverse through the layers to the enter in a single by pass, making them productive. This method calculates an importance score for each input for a given output. DeepLIFT clarifies the contrast in output from a few 'reference' output in terms of the distinction of the input from a few 'reference' inputs. The 'reference' input speaks to a few default or 'neutral' input that's chosen concurring to what is suitable for the issue at hand [58]

Layer-wise Relevance Propagation method starts from the output going to the input. It weights the neurons with more contributions. The results of this method are given in the form of a heatmap. The relevance of each neutron can be defined using the following formulae:

$$R_i = \sum \frac{a_i W_i j^+}{\sum a_i W_i j} R_J \tag{2.3}$$

j passes through all the neurons that exist on a path between i and j, W_ij shows the connection weight and a_i's are the results of the data form neuron i, during classification [59]

The Convolutional Neural Network Interpretation (CNN-INTE) method uses Decision Trees (DT) to interpret the inner workings of hidden layers and how the model classifies the test instances to increase trust on a trained DNN. The results of this method are represented in a graph that indicates the consecutive separations of the true category and also the hypotheses [60]. CNN-INTE offers the global interpretation for any check instances on the hidden layers within the whole feature area, it does not compromise the accuracy of the model to be interpretable. It is designed on the Tensorflow platform, this makes the model scalability easy to accommodate larger datasets and it interprets the inner workings of Deep CNN models, this are some of the advantages of the CNN-INTE method.

Automatic rule extraction is another example of method used for understanding the learning process in DL and and how the prediction is made. The rule extraction techniques can be split into decompostional and pedagogical methods [61]. Decompositional algorithms draw out the rules to copy the behaviour of each unit at the neural level. These methods are said to be better compared to pedagogical algorithms with regards to understanding the internal structure of NN. The purpose of pedagogical algorithms is to extract rules, which maps out the inputs to the outputs instead of considering the internal structures of the NN. "they treat the network as a black box, and discovers trends and functions from the inputs to the output" [54].

Polynomial's methods are an example of decompositional approach. The function approximation task is to build a function that reproduces a given relationship between input and output. Using artificial neural networks to estimate functions involves building a neural network that approximates the desired functional mapping from numerical input vectors to numerical output vectors. Polynomials are widely used as the basis of non-linearity modelling [62].

3 Methodology

This part of the project is concerned with reproducing the learnt function (after training the FFNN) as a numerically analysable function at a lower dimensional space. ANNs are well suited at learning complex functions of unknown origins, so it is easier to extract the already learnt function and analyse it further instead of attempting to start with a polynomial model.

3.1 Data set

1. Iris dataset: "The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper [63]. The Data set has the following classes: setosa, versicolor, virginica. Each class has 50 samples.

TABLE 3.1: Iris Data set

	Setosa	Versicolor	Virginica
Samples per class	50	50	50

2. Wine Quality Datasets:" Two datasets are included, related to red (1) and white (0) vinho verde wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests" [64]. Table 3.2 shows the total number of samples available for each class

TABLE 3.2: Wine data set

	Red	White
Samples per class	1599	4898

3. Fashion-MNIST: is an image dataset from Zalando's article [65], it has 60000 training dataset and 10000 testing dataset. Every image is a 28×28grayscale and it has 10 classes. Table 3.3 is a description of the data and number of samples for each class. Table 3.4 gives a

TABLE 3.3: Data description

	Training	Testing
Classes	10	10
Samples per class	6000	1000

description of classes.

TABLE 3.4: Fashion Mnist

Code	0	1	2	3	4	5	6	7	8	9
Name	T-shirt or top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle

3.2 Method

The advantage of deep learning models is the fact that it can detect and successfully map high dimensional abstract features into low some function. The implication is that there is a mathematically sound function that is being developed by the deep learning algorithm. In this research we propose methods to extract such functions. Our methodology is comprised of three main steps: train FFNN, polynomial interpolation and functional analysis. In polynomial interpolation we will use Singular Value decomposition to identify principal directions for each layer's weight at every epoch, and we are going to use Lagrange interpolation to build the polynomial representation of the learnt function.

Given some training data which consists of x as inputs and Y as output, a single layer neural network is the approximation: $Y = Ax$ where A is a weight matrix, the process of deep learning uses two passes (forward and backward) to optimise A to fit Y better. We can increase the number of weight matrix (A) to help in representing more abstract function representations. e.g $Y = f(x)$ where f is a linear transform of x modelled by A.

1: **Train FFNN** Train a FFNN with 3 hidden layers and one output layer. The rectified linear unit (ReLU) was used as the activation function in the hidden layers and softmax for the output layer. The model takes in input features and maps them to output categories, this is illustrated by figure 3.1.

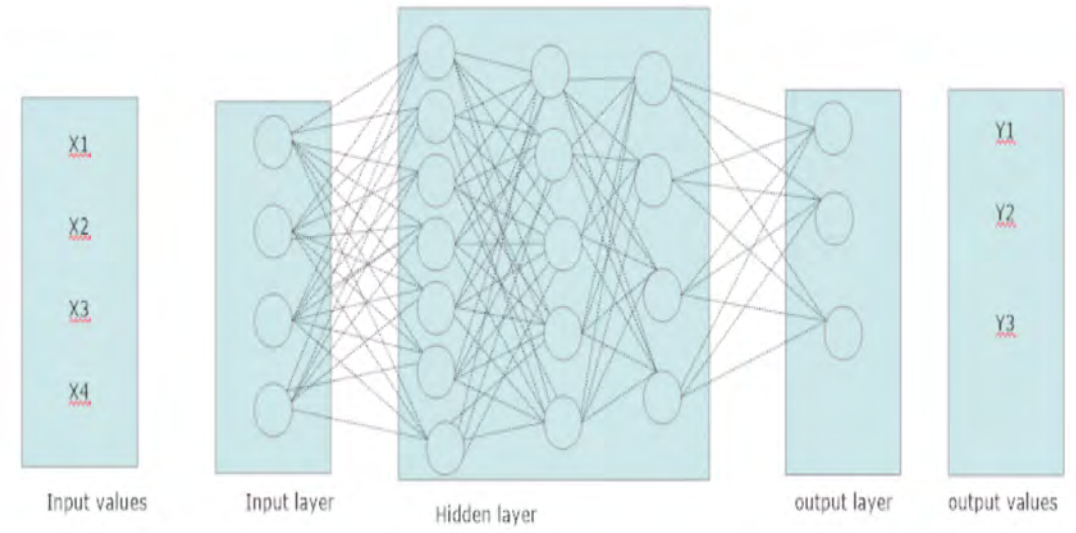


FIGURE 3.1: Feedforward Neural Network

Extract principle directions from the weight matrices in the trained model using singular value decomposistion (SVD). In dimensionality reduction we reduce the high-rank matrix to a low-rank matrix while preserving important information. SVD decomposes matrix A into three product matrices U,D and V, and is represented as :

$$A = UDV^T \tag{3.1}$$

Where U $m \times m$ and V $n \times n$ matrices are orthogonal matrices of A and D $m \times n$ is a diagonal matrix of A. U can also be referred to as left singular vector and V as the right singular vector, D is also known as the singular values of matrix, which represents importance of values of different features in the matrix. The first principal component of V is obtained because V carries most of the variance.
"Example of SVD: Let's look at a classical application of this. Imagine that we have a matrix A whose columns represent movies and the rows different users. The entries of the matrix are numbers 0 to 5 where 0 means a user does not like a certain movie and 5 means they really like a given movie as illustrated below: The U matrix will represent weights that would match each user's preference to movies categories, D represents the weight of the movie category and V represents the degree to which a movie belongs to a category" [66].

A polynomial is fitted on the new weight matrix (after matrice decomposition) using langrage interpolation, which is given by:

$$P(x) = \frac{(x - x_2)(x - x_3)(...)(x - x_n)}{(x_1 - x2)(x_1 - x_3)(...)(x_1 - x_n)} \tag{3.2}$$

This function will show how the weights are adjusted when training. The polynomial

function interprets the learning process in a FFNN, and we are going to use analytical tools to understand the learned functions. The polynomial function are subjected to analytical examination using other tools such as plot training data against new functions on the domain, convergence analysis or run convolutions of the final layers of different epochs.
The following steps were followed to get results for this part of the project:

- In each epoch iterate through the layers
- For each layer send the matrix to the SVD algorithm and take the first principal component
- Use the results from SVD as Y and generate the X from the length of the SVD result
- Use (X, Y) as arguments to the Lagrangian polynomial interpolation algorithm
- Store the polynomial in a dictionary indexed by the layer number
- Repeat this for the next epoch

3: **Functional analysis and model interpretation** We used convolution method to map out how each epoch's function affected the transformation of the next epoch by simply convolving the first epoch with the second epoch the resulting convolution is representative of the transformation that has occurred

3.3 Explanation method

In this section we will be looking at the LIME method. These method looks at important features in the datasets. These methods will be used to answer the questions "Why was this prediction made? and which features or variables were used?". Local Interpretable Model-Agnostic Explanations (LIME) method is used to attempt to learn and understand the models output. The following steps are used for training a LIME method

- After training FFNN model.
- Select the data point you want to explain.
- Shuffle values of one column on the given dataset, use the new dataset to make predictions and see how this has affected the loss function.
- This process is the repeated for all columns in a dataset.
- The decrease of performance shows how important a feature is.

3.4 Performance measures

Accuracy: We used classification accuracy to evaluate how the model performs. Accuracy is another measure used to evaluate models of classification. Accuracy represents a percentage of predictions that the model gets correct, represented by equation 3.3:

$$\text{Accuracy} = \frac{\text{correct Predictions}}{\text{Overall prediction}} \tag{3.3}$$

Confusion matrix or a heatmap gives the description of the prediction outcomes for a classification problem. Total number of right and wrong predictions is presented by counting values and divided per class.

4 Results

Given a DL model, we can model the change in weights during the training process, by building an Nth degree polynomial and analysing the resultant. In this section we looked at the trained 3 layer FFNN, layer0 has 15 neurons, layer1 has 10 and layer2 (output) has different neurons depending on the dataset. We used ReLU as the activation function and softmax function for the output layer.

The aim of this work is to expose the inner working of the learnt function of a deep neural network. We also proposed a way to map the training process by storing weights at each epoch then building polynomials from the first principal component of each layer. We used SVD to extract the first principal component of the V matrices, the output from this is then used to fit a polynomial function. Appendix B shows the results for the three datasets used for this experiment. The one method used was to map out how each epoch's function affected the transformation of the next epoch by obtaining the convolution of the first an second epoch such that it represents the transformations that occurred. Appendix A shows the results for simple convolution.

The LIME method was used to understand which features are important. The LIME method results showed the prediction probability for each class, relative importance and value ranges for each feature. We looked at the explanations for each class for Iris and Wine quality dataset, and we looked at the explanation for class 9 on the fashion Mnist dataset.

4.1 Data Exploration

In this section we look at models performance, correlation matrix, and heat-map for all the datasets.

4.1.1 Iris data set

Table 4.1 is the correlation matrix for the features of the Iris data set which are sepal length, sepal width, petal length and petal width. Sepal length has a high positive correlation (>0.8) with petal length and petal width. However, sepal length similar to petal length and petal width, has a low negative correlation with sepal width.

TABLE 4.1: Correlation matrix

	Sepal length(cm)	Sepal width(cm)	Petal length(cm)	Petal width(cm)	Target
sepal length(cm)	1.000000	-0.109369	0.871754	0.817954	0.782561
sepal width (cm)	-0.109369	1.000000	-0.420516	-0.356544	-0.419446
petal length (cm)	0.871754	-0.420516	1.000000	0.962757	0.949043
petal width (cm)	0.817954	-0.356544	0.962757	1.000000	0.956464
target	0.782561	-0.419446	0.949043	0.956464	1.000000

Figure 4.1 shows the models performance after training the 3 layer FFNN over 1000 epochs, the red lines represent the testing accuracy and the blue lines represents the training accuracy. The distance between the red graph and the blue graph shows that the model is overfitting. Overfitting indicates the the model has modelled data very well including the noise in the training data set. The average performance is 97.63%.

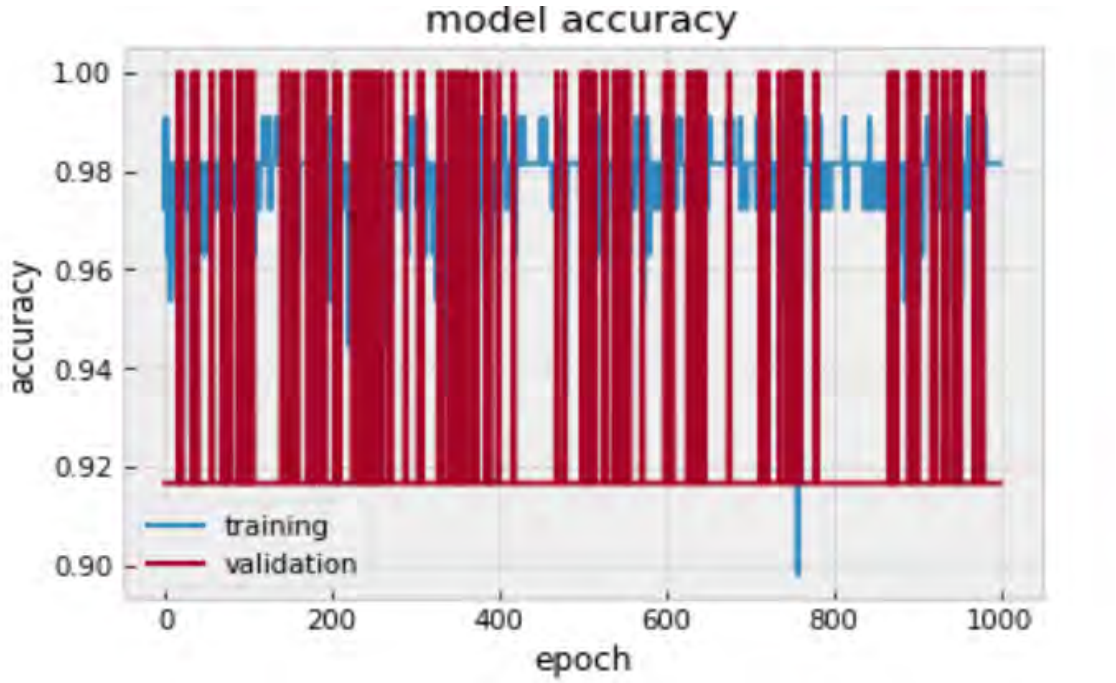


FIGURE 4.1: Iris: Model perfomance

Figure 4.2 shows the heatmap of Iris data set. The Y-axis represents the true label while the X-axis represents the predicted label/ labels predicted by the model (FFNN). The heatmap shows the relationship between two classes. It shows how many data points were not correctly classified. The numbers 0, 1, 2 represents the classes Iris Setosa, Iris Versicolour, Iris Virginica. The heatmap shows that no data point was misclassified.

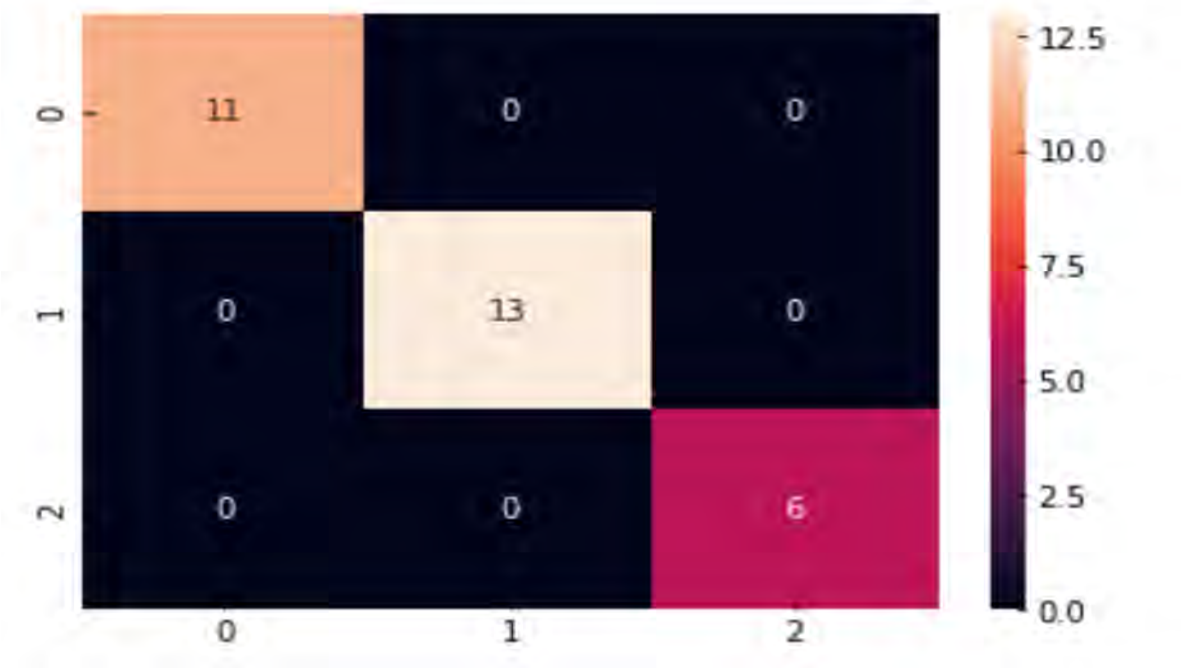


FIGURE 4.2: Iris: Heatmap

4.1.2 Wine quality

Figure 4.3 show the accuracy of the model at 100 epochs. The average model's performance is 97.05%. This means the model is able to accurately classify over 90% of the test data set. The validation accuracy graph has two spikes which might be the results of overfitting, because we have more data points for class 0 compared to class 1. The test data set has 1300 total observations, 989 represents class 0 (red wine) and 311 represents class 1 (white wine). Figure 4.4 shows the heatmap for the Wine quality data set, we have 11 data points that are classified as class 0 while they belong to class 1, and 6 data points classified as class 1 while they belong to class 0. Out of 989 in class 0 983 we correctly classified and in class 1 300 were correctly classified from the 311. that

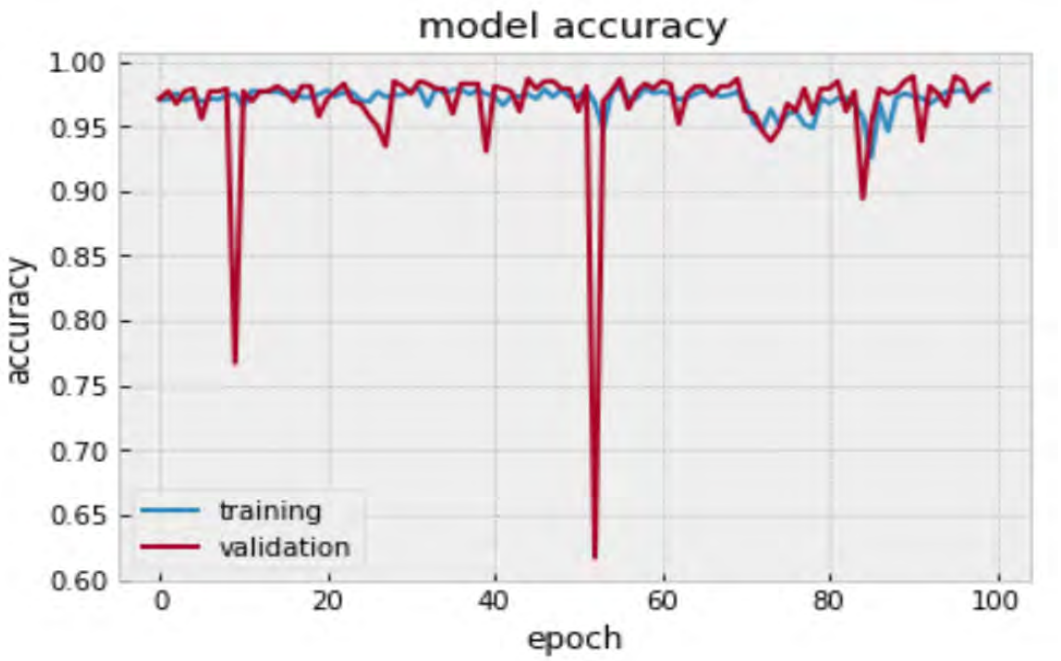


FIGURE 4.3: Wine quality: Model performance

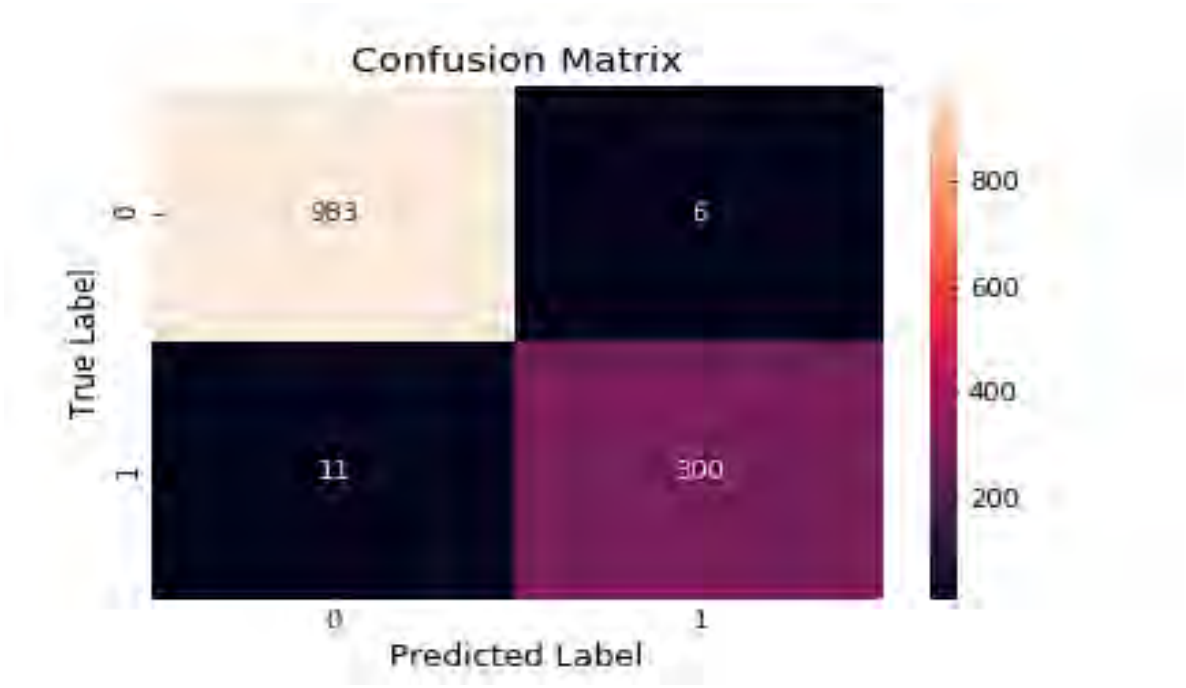


FIGURE 4.4: Wine quality: Heatmap

4.1.3 Fashion Mnist

Average model performance is 79.80%. Figure 4.5 shows the performance of the model at 1000 epochs. The test data set has 10000 samples in total, and 1000 samples from each class. The heatmap 4.6 shows total number of samples that are misclassified for each class. For example we can see that class 6 has the highest number of samples that are misclassified as class 0 or class 4, and class 2 has 158 samples that are misclassified as class 4. This graph also shows that class 9 and 7 had no samples that are misclassified as class 0, 1, 2, 3, 4. Table 4.2 shows that class 6 has the lowest precision, recall and f1-score and class1 has the highest precision, recall and f1-score.

TABLE 4.2: Fashion Mnist: Precision and recall

	precision	recall	f1-score	support
0	0.74	0.75	0.75	1000
1	0.94	0.94	0.94	1000
2	0.67	0.52	0.58	1000
3	0.74	0.84	0.79	1000
4	0.66	0.56	0.61	1000
5	0.95	0.83	0.89	1000
6	0.37	0.47	0.42	1000
7	0.80	0.94	0.87	1000
8	0.92	0.88	0.90	1000
9	0.91	0.88	0.89	1000
AVG/Total	0.77	0.76	0.76	10000

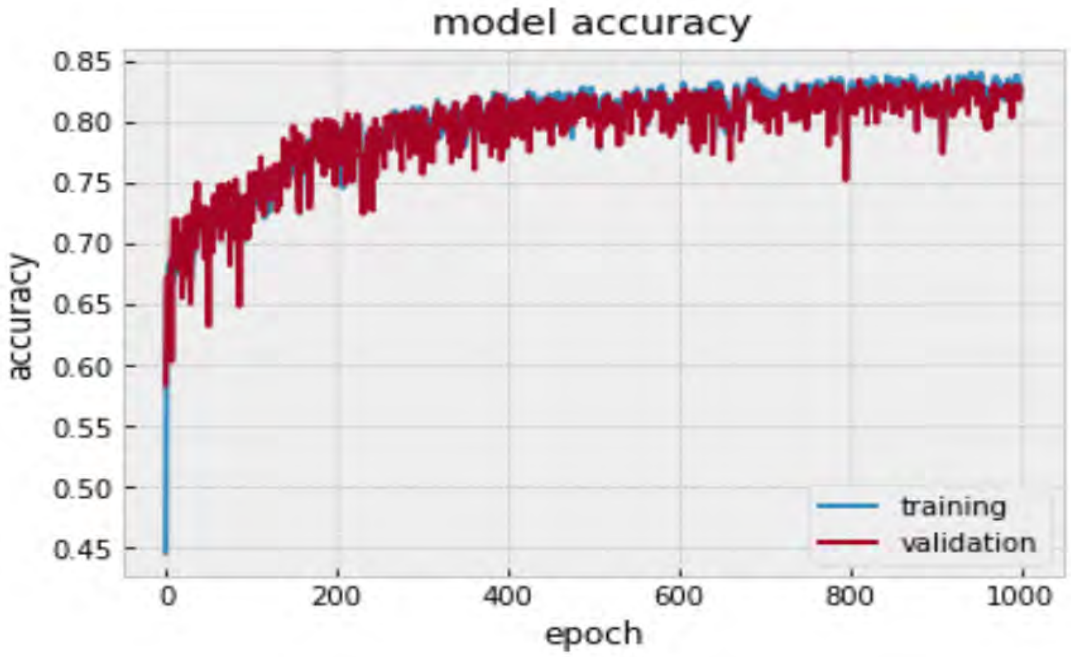


FIGURE 4.5: Fasion Mnist: Performance

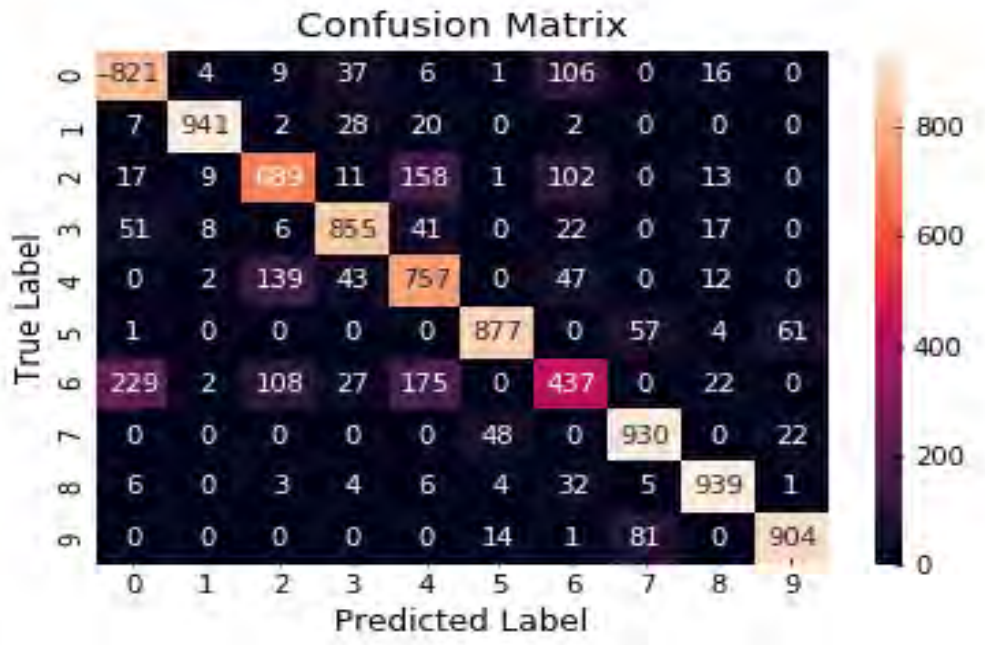


FIGURE 4.6: Heatmap matrix

4.2 LIME Method

4.2.1 Iris Data

Class 1 explanation: for class 1 explanation we looked at instance 1. Instance 1 true positive of class 1, the description of this instance is as follows, sepal length(cm) is 6.00, sepal width(cm) is 2.20, petal width (cm) is 1.00 and petal length (cm) is 4.00 refer to 4.8. Petal length and petal width are positively contributing to the prediction of class 1 while sepal width and sepal length are negatively contributing to the prediction of class 1, this is illustrated by figure 4.7. Class 1 was predicted with a prediction probability of 1.0, figure 4.8 shows the prediction, values for the feature and relative importance for each feature. Petal length has relative importance of 0.23 which is the highest compared to other features, and sepal length has the lowest relative importance.

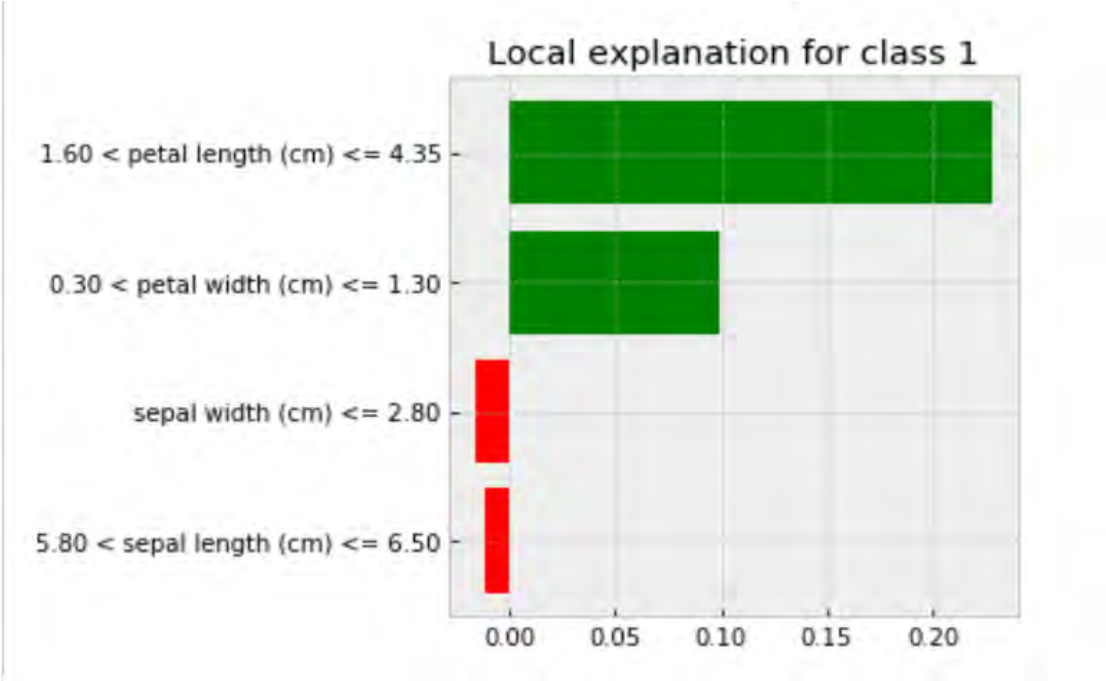


FIGURE 4.7: Class 1 Explanation

FIGURE 4.8: Class 1 Explanation features

Class 2 explanation: for class 2 explanation we used instance 10. Instance 10 has the following description Sepal length(cm) is 6.10, sepal width(cm) is 2.60, petal width (cm) is 1.40 and petal length (cm) is 5.60,figure 4.10 shows the prediction probability of class 2 as 0.94. Petal length, petal width and sepal width are features providing negative valuation to the prediction of class 2, while sepal length is the only feature providing positive valuation to the prediction, this is shown by figure 4.9.

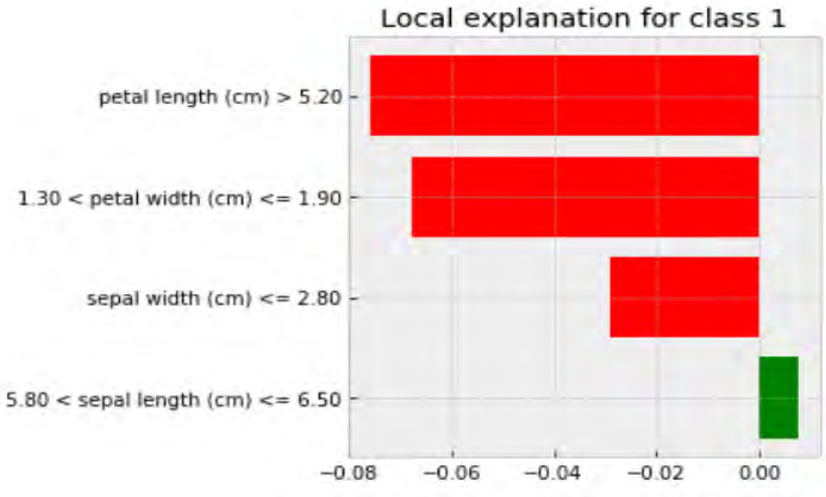


FIGURE 4.9: Class 2 Explanation

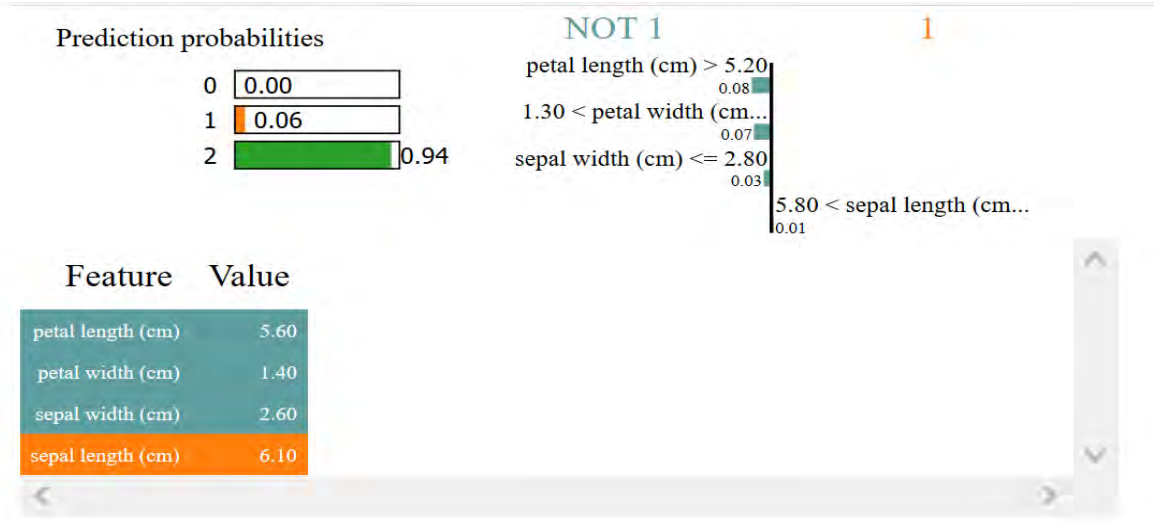


FIGURE 4.10: Class 1 Explanation features

Class 0 Explanation: We looked at instance 15 for class 0 explanation. The instance has the following description petal length (cm) 1.50, petal width (cm) 0.10, sepal length (cm) 4.90 and sepal width (cm) 3.10, figure 4.12 show the prediction probability of class 0 1.00.The graph also shows the relative importance for each feature as follows petal length (cm) 0.36, petal width (cm) 0.08, sepal length (cm) 0.02 and sepal width (cm) 0.01. Petal length and sepal width are features that provide negative valuation and petal width and sepal length are positive valuation for the prediction illustrated by figure 4.11.

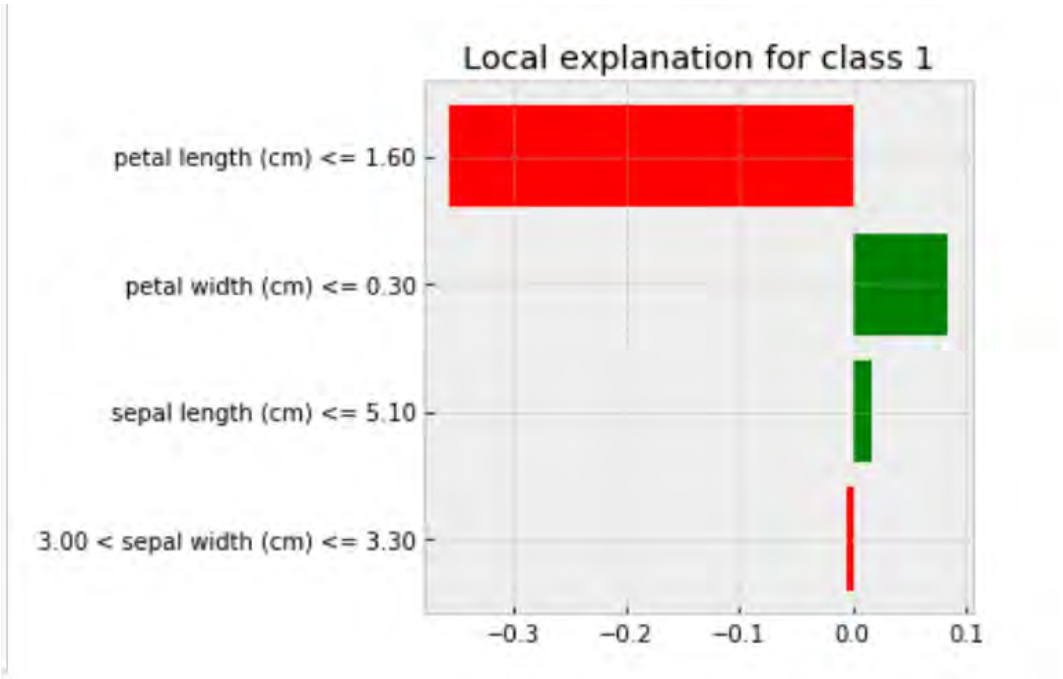


FIGURE 4.11: Class 0 Explanation

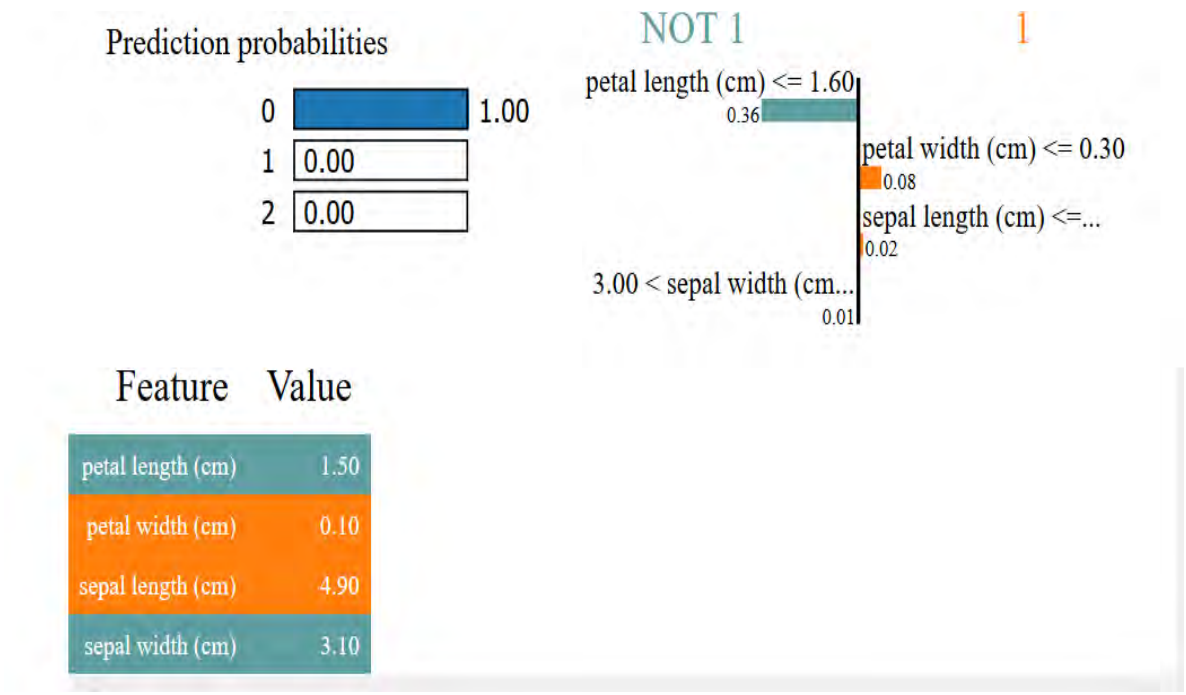


FIGURE 4.12: Class 0 Explanation features

4.2.2 Wine quality

Class 0 explanation: We used instance 1 for class 0 explanation. We looked at the top 5 features. The description of this instance areas follows; volatile acidity 0.14, residual sugar 1.40, citric acid 0.18, chloride 0.05, fixed acidity 6.80. Figure 4.13 shows the prediction probability of class 0 as 0.73. Figure 4.14 shows most important 5 feature for class 0. shows that volatile acidity, residual sugar provide negative valuation to the prediction, while fixed acidity, chlorides and citric acid provide positive valuation to the prediction.

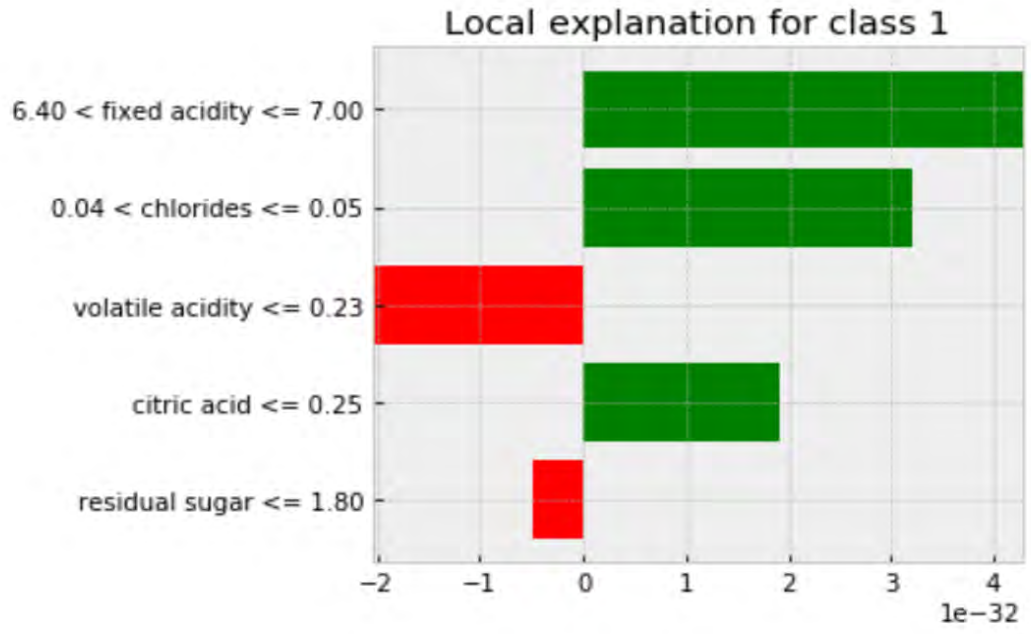


FIGURE 4.13: Class 0 Explanation

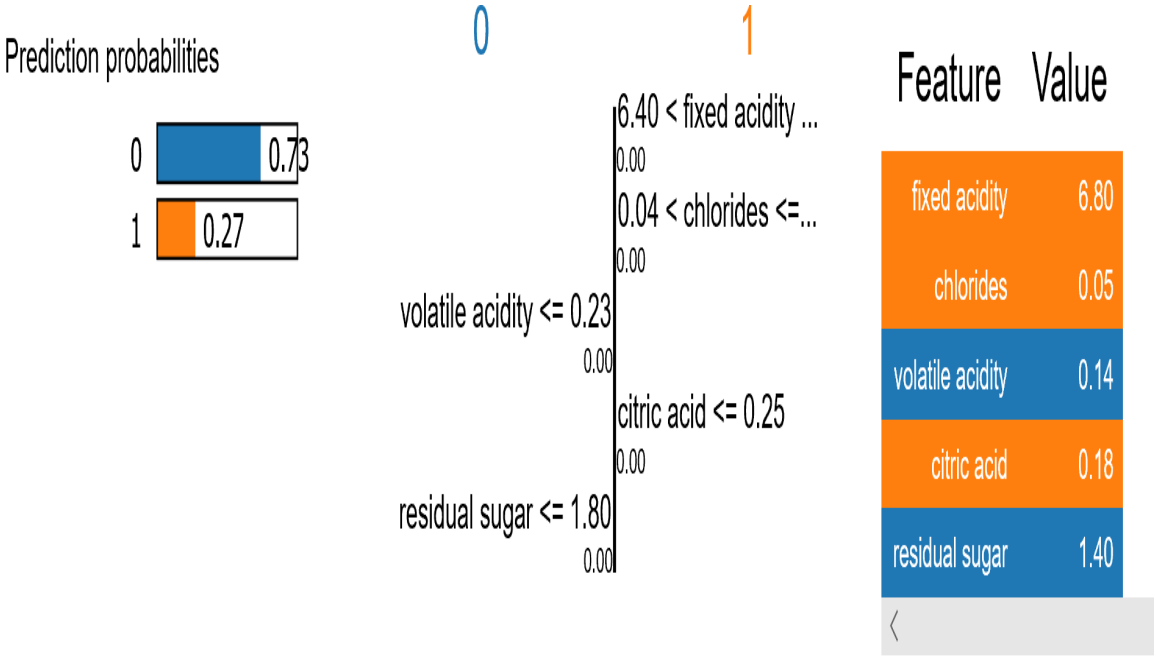


FIGURE 4.14: Class 0 Explanation features

Class 1 explanation: we used instance 100 for class 1 explanation. Top 5 important features for instance are total sulfur dioxide 16.0, fixed acidity 10.70, sulphates 0.65, free sulfur dioxide 5.0, and chlorides 0.07. Figure 4.16 shows the prediction probability of 1.00. Total sulfur dioxide has the highest relative importance of 0.61 while chlorides has the lowest relative importance of 0.05. Total sulfur dioxide, sulphates, fixed acidity and chloride have positive valuation to the prediction while free sulfur dioxide is the only that provides negative valuation, shown on figure 4.15.

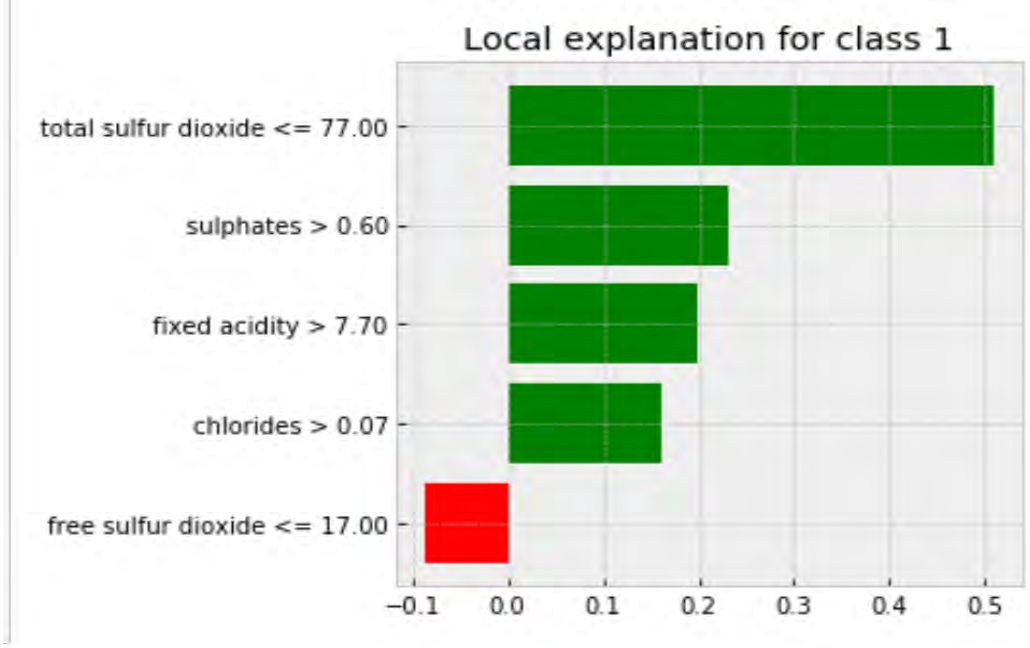


FIGURE 4.15: Class 1 Explanation

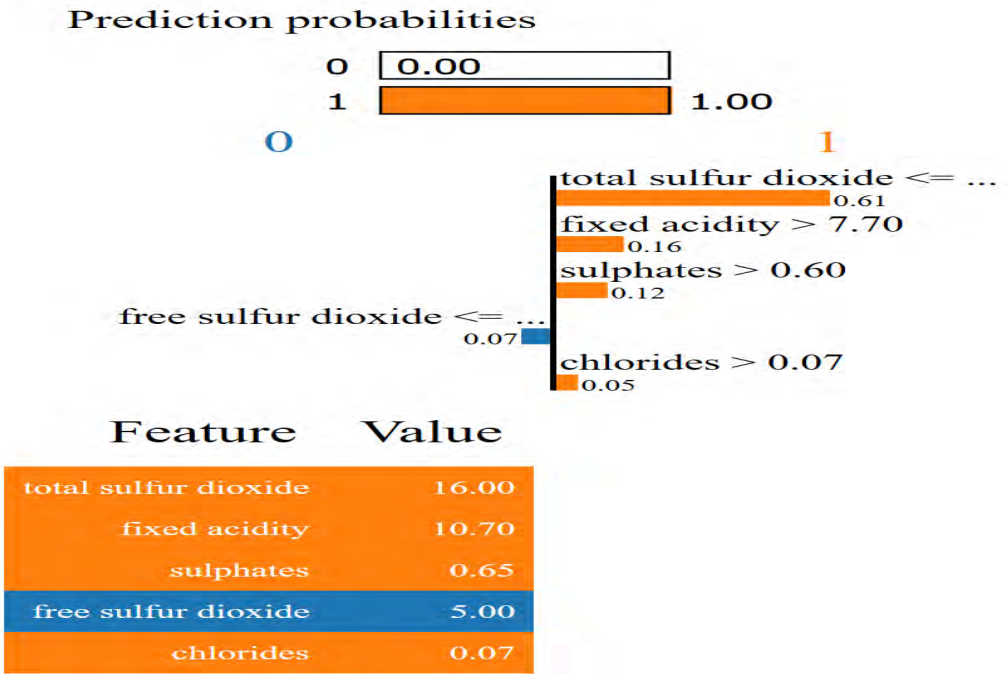


FIGURE 4.16: Class 1 Explanation features

Using 842 where the label was 0 and the model predicted 1 The sulphates value is 0.69 and the value for volatile acidity is 0.20 4.17. Sulphates has a higher relative importance than volatile acidity hence it is classified as class 1 at 83% instead of class 1 4.18.

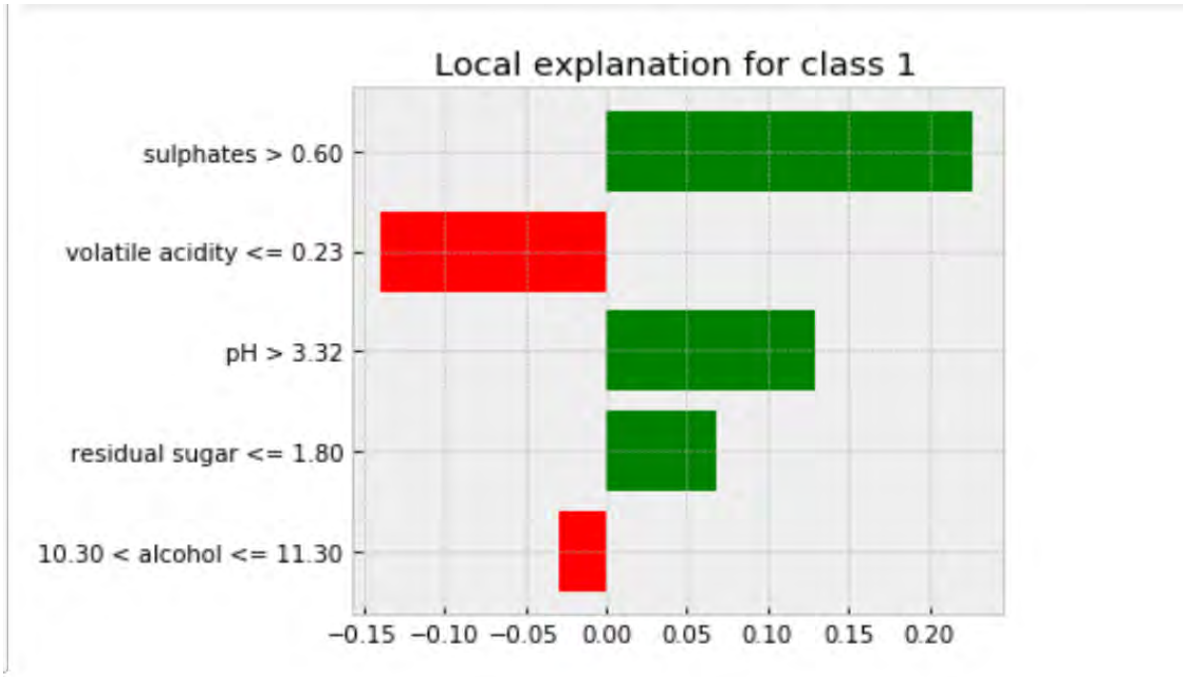


FIGURE 4.17: Local explanation

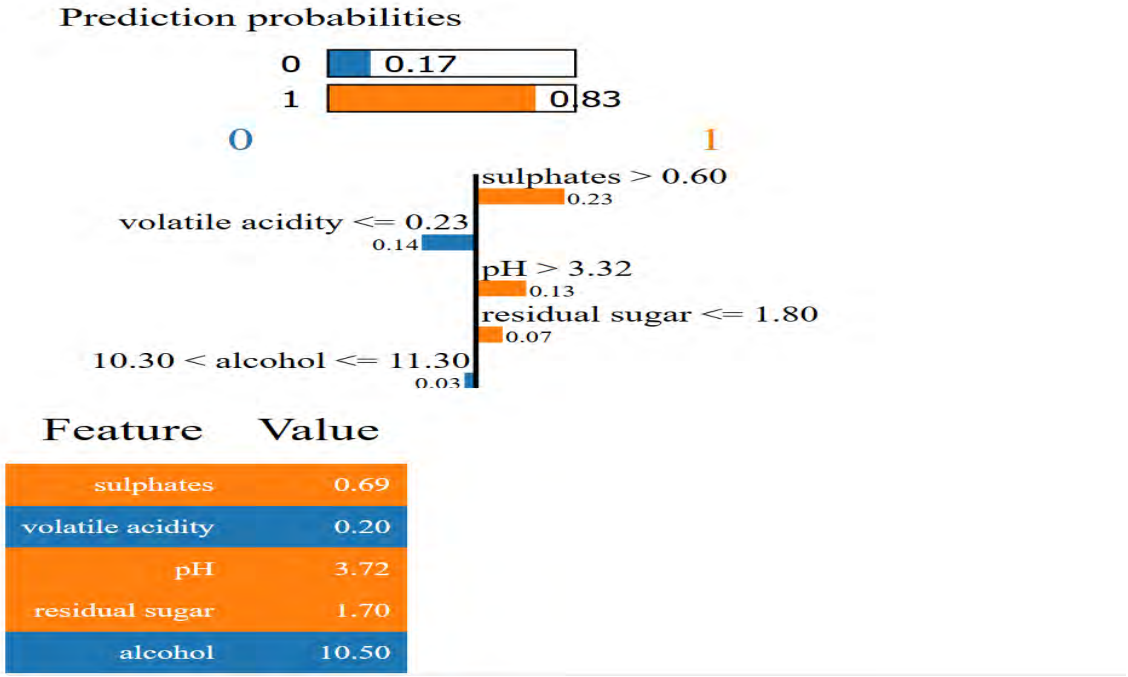


FIGURE 4.18: Local explanation features

4.2.3 Fashion Mnist

Figure 4.19 shows regions that are important for classifying class 9. Figure 4.20 shows positive regions for all classes in comparison with class 9 (Ankle boots). Class 7 (sneakers) and Class 9 have similar activation regions this could help explain why 81 instances of class 7 were classified as class 9 refer to the heatmap 4.6.

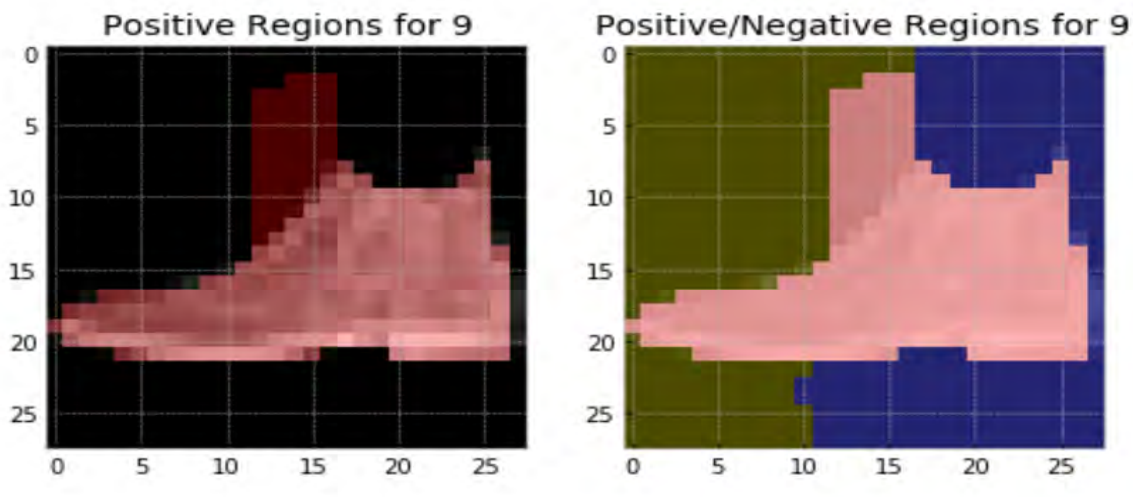


FIGURE 4.19: Class 9 Regions

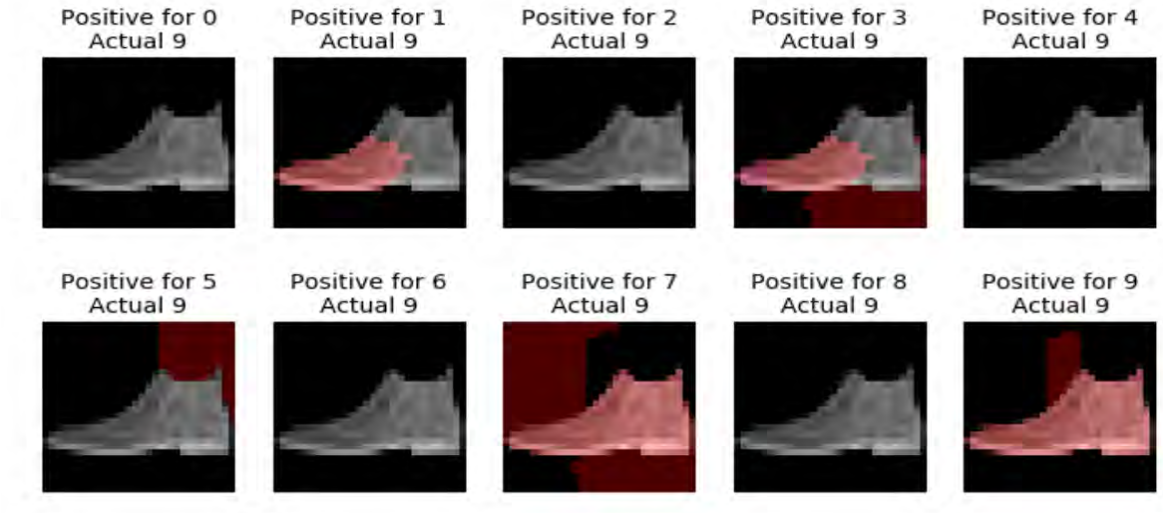


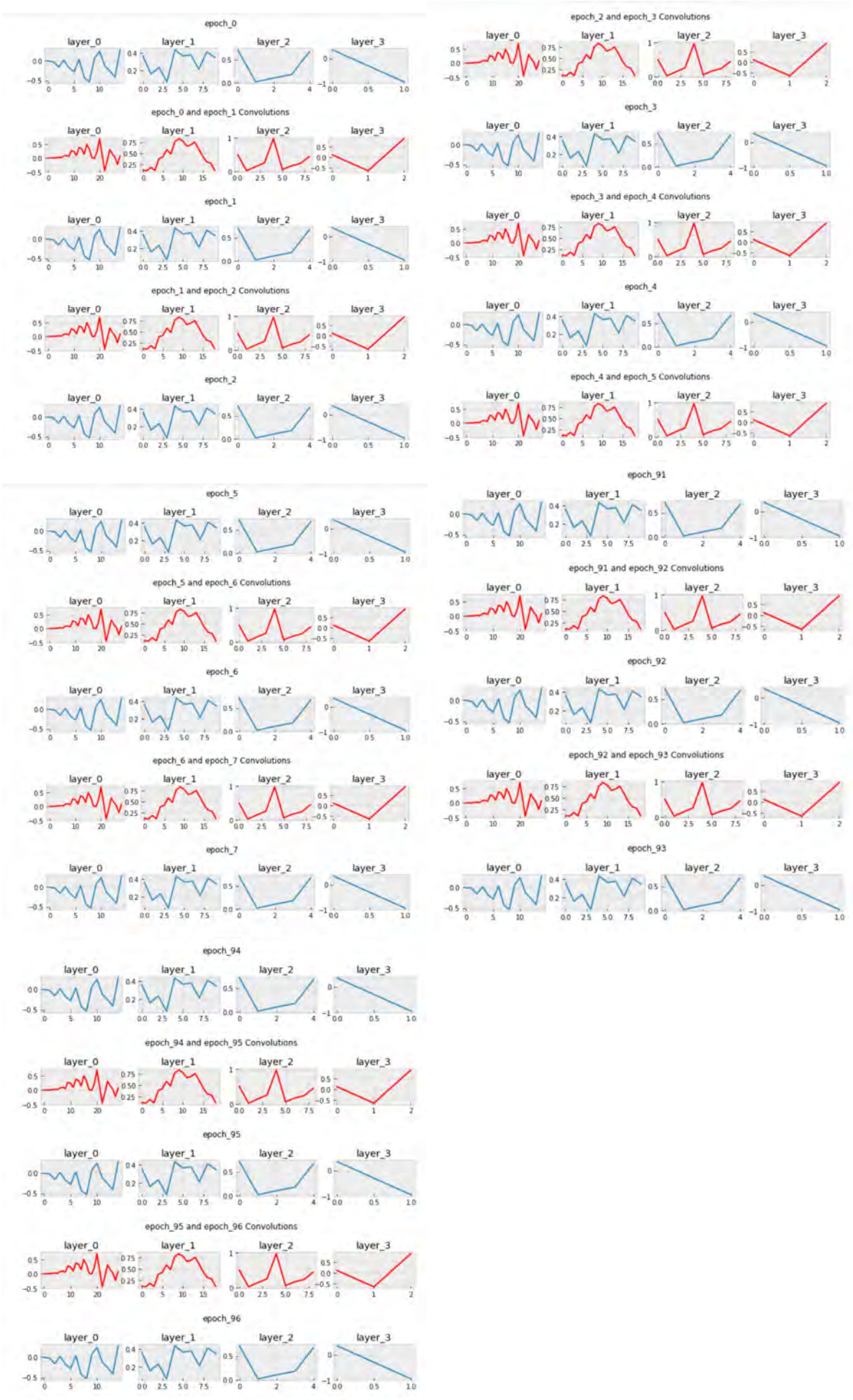
FIGURE 4.20: Positive regions for class 9

4.3 Polynomial interpolation

Table 4.3, shows the results for polynomial interpolation part in the methodology. The results are only representing two epochs (epoch 0 and epoch 1) for each data set, P(x) represents the polynomial function, X shows the number of nodes at that layer and y are SVD results. Figure 4.21a shows the convolution of the polynomial function. Convolution method was used to show the behaviour of the model. In figure 4.21a, shows that there is not a lot of changes in the models behaviour from epoch 1 to the last epoch, because there is no significant change in the graphs. The blue graph represents the training results per epoch and the red graphs shows convolution of two epochs, the graphs shows similar behaviour or pattern from epoch 1 to epoch 90 and the convolution graphs also have similar pattern.

TABLE 4.3: Wine Polynomial interpolation

$epoch_0$				
	$layer_0$	p(x)	x	y
		-3.23862482e-10, 2.06164715e-08, -3.47411891e-07, -7.78649556e-06, 4.72200195e-04, -1.04102940e-02, 1.36920408e-01, -1.19135346e+00, 7.07056301e+00, -2.85593054e+01, 7.63067943e+01, -1.26768356e+02, 1.15621829e+02, -4.25672737e+01, -9.11021754e-02	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	-0.09110218, -0.05122982, 0.33642676, -0.27325457, 0.28258833, -0.01930992, -0.3615444 , -0.2563718 , 0.03383022, 0.12903747, -0.28134817, -0.37012035, 0.31131324, -0.01845747, 0.43972418
	$layer_1$	2.09854878e-05, -9.98133781e-04, 1.94259829e-02, -2.01178852e-01, 1.20330946e+00, -4.20330980e+00, 8.16980976e+00, -7.55285041e+00, 1.86167142e+00, 1.33857489e-01	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0.13385749, -0.57024211, -0.3764759 , -0.04627177, 0.07858355, 0.44721633, 0.36031622, -0.36523172, 0.00297101, 0.20885958
	$layer_2$	0.2073358 , -1.65089554, 4.12884364, -3.33680005, -0.05854975	0, 1, 2, 3, 4	[-0.05854975, -0.7100659 , -0.10656677, -0.68933666, 0.07639947
	$layer_3$	1.22431305, -0.2582294	0, 1	-0.2582294 , 0.96608365
$Epoch_1$				
	$layer_0$	-3.23858952e-10, 2.06161218e-08, -3.47396301e-07, -7.78690895e-06, 4.72207451e-04, -1.04103828e-02, 1.36921185e-01, -1.19135835e+00, 7.07058517e+00, -2.85593761e+01, 7.63069477e+01, -1.26768568e+02, 1.15621996e+02, -4.25673282e+01, -9.11021456e-02	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	0.09110215, -0.05122979, 0.33642685, -0.2732546 , 0.28258833, -0.01930995, -0.36154437, -0.25637183, 0.03383023, 0.12903745, -0.2813482 , -0.37012032, 0.31131324, -0.01845748, 0.43972412
	$layer_1$	2.09855217e-05, -9.98135205e-04, 1.94260081e-02, -2.01179096e-01, 1.20331086e+00, -4.20331469e+00, 8.16981961e+00, -7.55286070e+00, 1.86167550e+00, 1.33857548e-01	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0.13385755, -0.57024211, -0.37647596, -0.04627169, 0.07858351, 0.4472163 , 0.36031622, -0.36523166, 0.00297108, 0.20885953
	$layer_2$	0.20733574, -1.650895 , 4.12884229, -3.33679901, -0.05854993	0, 1, 2, 3,	-0.05854993, -0.7100659 , -0.10656697, -0.6893366 , 0.07639939
	$layer_3$	1.22431293, -0.25822929	0, 1	-0.25822929, 0.96608365



(A) Wine quality Epoch Convolution

4.4 Discussion

In this project we looked at using polynomials functions to expose the inner workings of a deep learning method. After training a FFNN, we then mapped the weights and ran SVD on the extracted weight matrix and then fitted a polynomial function. After these polynomials were generated, mathematical analysis was applied to understand the network. This function was used for further analysis such as understanding how each layer weight matrix changed over training, this can be achieved by using convolution method. If you convolve the first one to the second one to see how they changed, operations ran on the first one to get the second one. This is the method used on this project to get behaviour of a FFNN 6, because the data that was used to test this method was simple and clean it does not show any major changes during the training process.

Polynomial functions can also be used to show which features are important and this can be achieved by using penalising regression along the lines of Lasso to extract those dynamics.

This gives an advantage over the tree surrogate technique because tree surrogates only explain the final black box, without giving explanation to how the black box has evolved from random matrices to the learned function. This also validates the hypothesis that polynomials can be used to extract the learned function of a NN.

5 Conclusion

5.1 Future Work

What makes a good explanation?. With the current revival of explainability and interpretability around DL, most of the work and research in this field tends to follow the researchers' intuitions about what makes a 'good' interpretation [44]. To ensure that the explanation is valid and unbiased, the method used for explanation and interpretation must be clear and understood. In reality, several recent studies have shown that assignment approaches can actually yield inaccurate or deceptive outcomes, given the reality that they are visually attractive to humans [67]. Some literature demonstrates only a limited sample that study in explainable AI usually does not follow or draw on structures of interpretation from social science and this might be a disadvantage. For the future project in the explainable DL, it is important to understand how how humans identify, produce, choose, analyse, and present explanations seems almost necessary. We also need to look at robust quantitative metrics to evaluate explanations generated.

On this project we looked at the interpretation and the explanation of a simple FFNN, where we were able to show that you can use to analyse the learnt function. The method might not be applicable to other DL models because of how the learning process in different models. For example in CNN the nodes are not fully connected which might affect the learnt function.

5.2 Conclusion

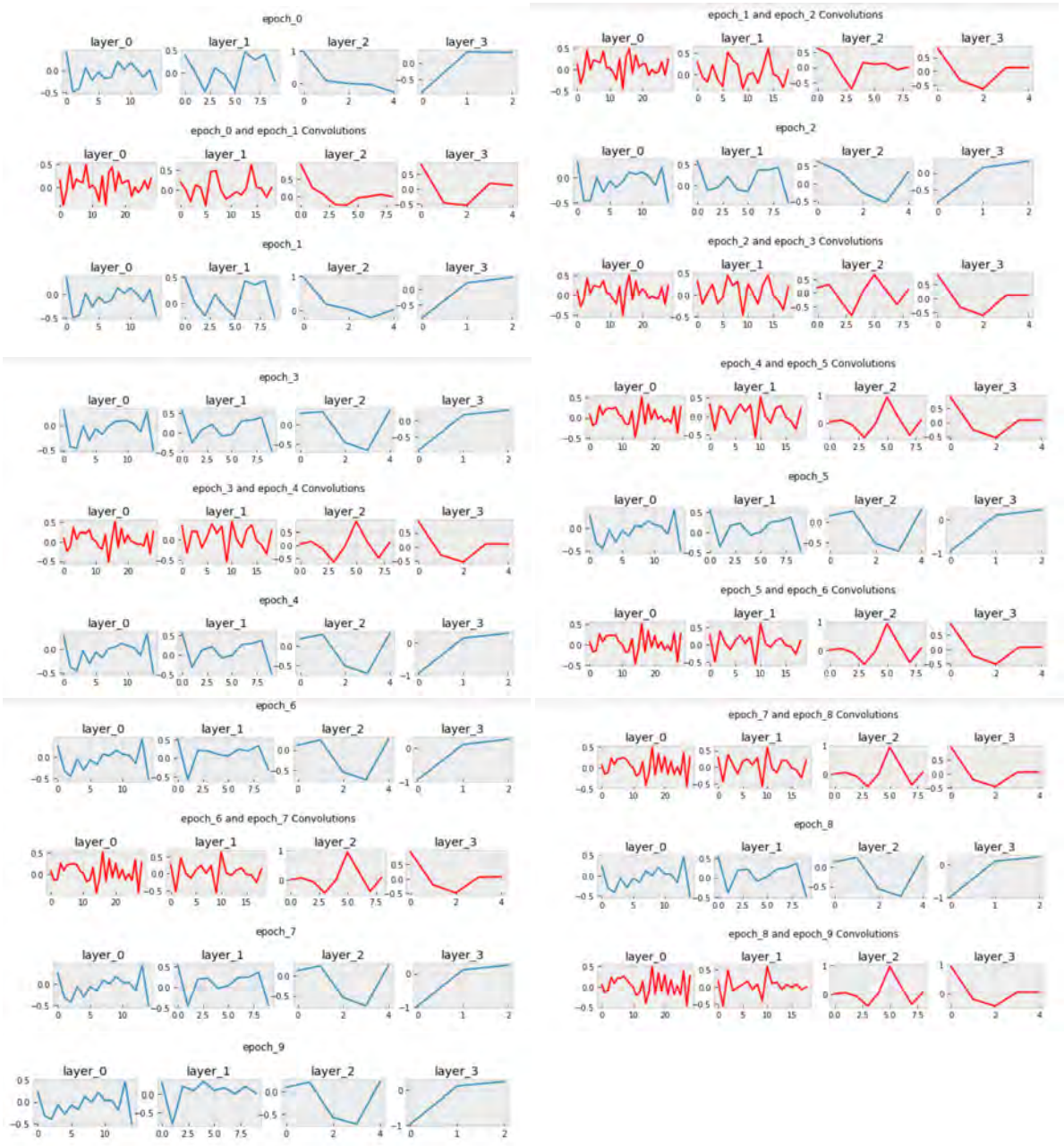
Increasing popularity of DL models in fields such as medicine, legal system, banking and self driving cars, increases the need for explanation and interpretation of black-box models. Explanability and interpretability of DL model are domain-specific approaches, hence it i difficult to have a formal or all-purpose explanation. This implies that model's interpretation or explanation applications must be domain and use-case specific problem. The proposed method can be applied for different problems

LIME methods are known to be unstable, this makes them unreliable because their explanations may change from one prediction to the next.

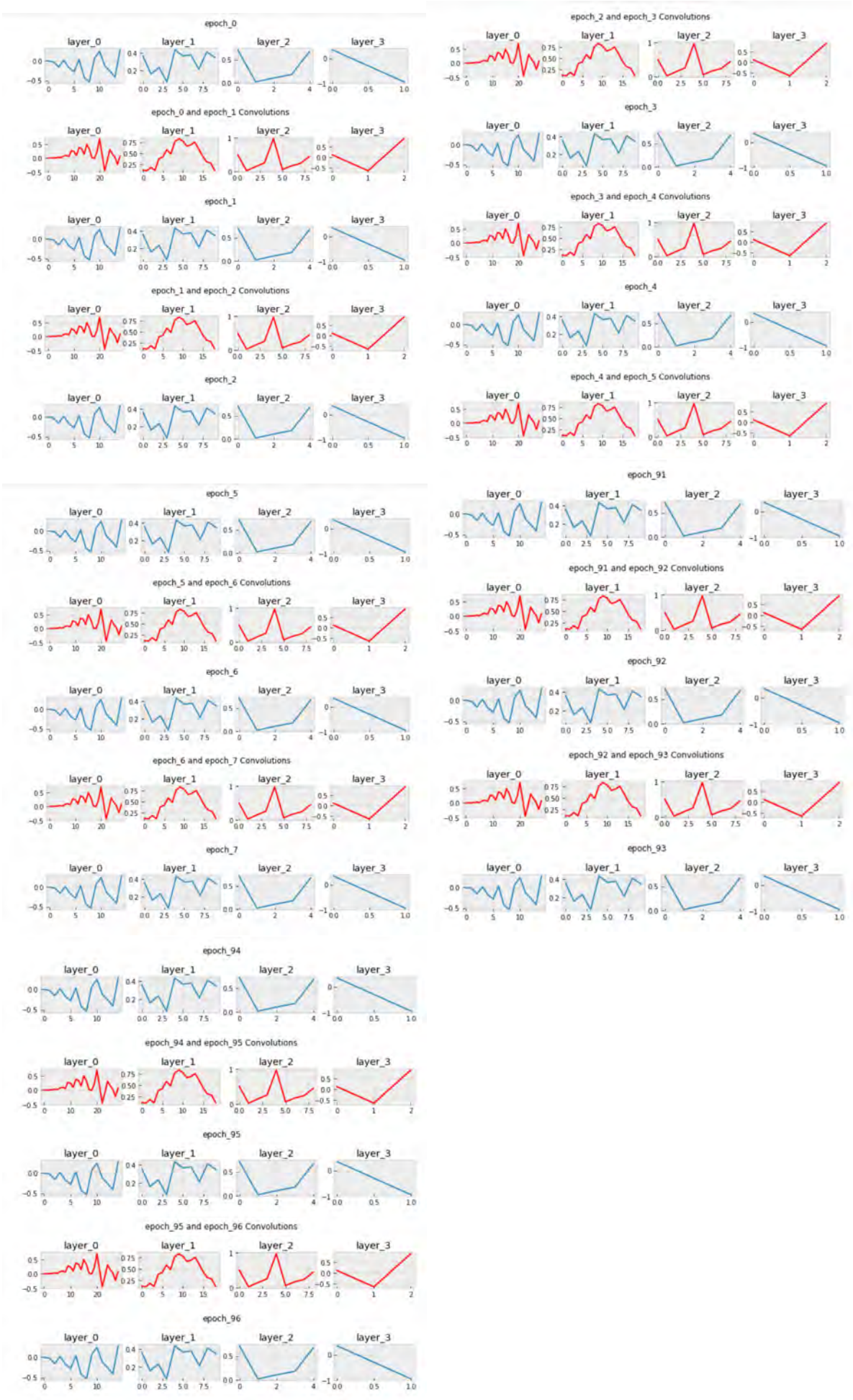
6 Appendix A

These graphs shows how the behaviour of a model, using convolution.

epoch to epoch convolutions:



(A) Iris Epoch Convolution



(A) Wine quality Epoch Convolution

(A) Fashion MNIST Epoch Convolution

7 Appendix B

In this appendix we are looking at the sample results for polynomial interpolation part in the methodology. The results are only representing two epochs (epoch 0 and epoch 1) for each data set, P(x) represents the polynomial function, X shows the number of nodes at that layer and y are SVD results.

$epoch_0$				
	$layer_0$	p(x)	x	y
		-3.23862482e-10, 2.06164715e-08, -3.47411891e-07, -7.78649556e-06, 4.72200195e-04, -1.04102940e-02, 1.36920408e-01, -1.19135346e+00, 7.07056301e+00, -2.85593054e+01, 7.63067943e+01, -1.26768356e+02, 1.15621829e+02, -4.25672737e+01, -9.11021754e-02	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	-0.09110218, -0.05122982, 0.33642676, -0.27325457, 0.28258833, -0.01930992, -0.3615444 , -0.2563718 , 0.03383022, 0.12903747, -0.28134817, -0.37012035, 0.31131324, -0.01845747, 0.43972418
	$layer_1$	2.09854878e-05, -9.98133781e-04, 1.94259829e-02, -2.01178852e-01, 1.20330946e+00, -4.20330980e+00, 8.16980976e+00, -7.55285041e+00, 1.86167142e+00, 1.33857489e-01	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0.13385749, -0.57024211, -0.3764759 , -0.04627177, 0.07858355, 0.44721633, 0.36031622, -0.36523172, 0.00297101, 0.20885958
	$layer_2$	0.2073358 , -1.65089554, 4.12884364, -3.33680005, -0.05854975	0, 1, 2, 3, 4	[-0.05854975, -0.7100659 , -0.10656677, -0.68933666, 0.07639947
	$layer_3$	1.22431305, -0.2582294	0, 1	-0.2582294 , 0.96608365
$Epoch_1$				
	$layer_0$	-3.23858952e-10, 2.06161218e-08, -3.47396301e-07, -7.78690895e-06, 4.72207451e-04, -1.04103828e-02, 1.36921185e-01, -1.19135835e+00, 7.07058517e+00, -2.85593761e+01, 7.63069477e+01, -1.26768568e+02, 1.15621996e+02, -4.25673282e+01, -9.11021456e-02	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	0.09110215, -0.05122979, 0.33642685, -0.2732546 , 0.28258833, -0.01930995, -0.36154437, -0.25637183, 0.03383023, 0.12903745, -0.2813482 , -0.37012032, 0.31131324, -0.01845748, 0.43972412
	$layer_1$	2.09855217e-05, -9.98135205e-04, 1.94260081e-02, -2.01179096e-01, 1.20331086e+00, -4.20331469e+00, 8.16981961e+00, -7.55286070e+00, 1.86167550e+00, 1.33857548e-01	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0.13385755, -0.57024211, -0.37647596, -0.04627169, 0.07858351, 0.4472163 , 0.36031622, -0.36523166, 0.00297108, 0.20885953
	$layer_2$	0.20733574, -1.650895 , 4.12884229, -3.33679901, -0.05854993	0, 1, 2, 3,	-0.05854993, -0.7100659 , -0.10656697, -0.6893366 , 0.07639939
	$layer_3$	1.22431293, -0.25822929	0, 1	-0.25822929, 0.96608365

TABLE 7.1: Wine Polynomial interpolation

epoch_0				
	layer_0	p(x)	x	y
		2.11165326e-09, -2.08923483e-07, 9.30895735e-06, -2.46540329e-04, 4.31324442e-03, -5.23930865e-02, 4.52069099e-01, -2.78706145e+00, 1.21851705e+01, -3.70031899e+01, 7.52093779e+01, -9.61425493e+01, 6.89978348e+01, -2.10335430e+01, -1.83148041e-01	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	-0.18314804, -0.35335666, -0.19154304, 0.13358602, -0.20581752, 0.00296288, 0.33001527, -0.01718213, -0.17726921, 0.21877842, 0.19866423, -0.4226566 , 0.20452492, -0.42926955, 0.33464867
	layer_1	6.45762628e-05, -3.08161981e-03, 6.12844035e-02, -6.60305067e-01, 4.18780645e+00, -1.58630869e+01, 3.46127651e+01, -3.91269869e+01, 1.72929818e+01, -2.86095679e-01	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	[-0.28609568, 0.21534608, -0.27649617, 0.44069314, -0.12008888, 0.01072556, 0.43940082, -0.01699707, 0.48005292, -0.40344214
	layer_2	-0.01803438, 0.0971183 , 0.00648505, -0.40281079, 0.51524073	0, 1, 2, 3, 4	0.51524073, 0.19799891, 0.22395571, 0.52658331, 0.60652888
	layer_3	0.84554413, -1.23403091, 0.01237822	0, 1,2	0.01237822, -0.37610856, 0.92649293
Epoch_1				
	layer_0	7.10034958e-09, -6.99202329e-07, 3.09801650e-05, -8.15726088e-04, 1.41986457e-02, -1.71924242e-01, 1.48374433e+00, -9.19615183e+00, 4.07029566e+01, -1.26237762e+02, 2.64683559e+02, -3.52309661e+02, 2.64306354e+02, -8.34098737e+01, -1.96195215e-01	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	-0.19619521, -0.33154008, -0.25403646, 0.19136032, -0.2110734 , -0.03950047, 0.39795822, -0.02074713, -0.13172838, 0.16857493, 0.12928137, -0.41482899, 0.2130491 , -0.44800684, 0.25466684
	layer_1	2.09855217e-05, -9.98135205e-04, 1.94260081e-02, -2.01179096e-01, 1.20331086e+00, -4.20331469e+00, 8.16981961e+00, -7.55286070e+00, 1.86167550e+00, 1.33857548e-01	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0.13385755, -0.57024211, -0.37647596, -0.04627169, 0.07858351, 0.4472163 , 0.36031622, -0.36523166, 0.00297108, 0.20885953
	layer_2	0.04982924, -0.36004673, 0.68235952, -0.14712773, -0.55205303	0, 1, 2, 3, 4	-0.55205303, -0.32703874, -0.19997643, -0.53729391, -0.50951725
	layer_3	0.56217916, -0.56605118, -0.13576214	0, 1,2	-0.13576214, -0.13963416, 0.98085213

TABLE 7.2: Iris Polynomial interpolation

Epoch_0				
		p(x)	x	y
	layer_0	1.02713374e-08, -1.04067023e-06, 4.75303997e-05, -1.29280004e-03, 2.33016049e-02, -2.92976344e-01, 2.63377619e+00, -1.70635435e+01, 7.92399001e+01, -2.58775794e+02, 5.72929543e+02, -8.05914511e+02, 6.36736316e+02, -2.09219360e+02, 1.40661508e-01	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14	0.14066151, 0.4360671 , 0.24076241, 0.23281349, 0.21986179, -0.5683618 , -0.05118958, -0.09608509, -0.14950033, 0.12232888, 0.02291792, -0.2592896 , 0.02036167, -0.36860955, -0.23087966
	layer_1	1.92844663e-04, -8.01074263e-03, 1.40584236e-01, -1.35575277e+00, 7.81723926e+00, -2.74085052e+01, 5.65317582e+01, -6.18616364e+01, 2.70131392e+01, -3.84061396e-01	0,1,2,3,4,5,6,7,8,9	-0.3840614 , 0.4849473 , -0.6600985 , 0.10955357, -0.07966646, 0.12726319, 0.0113221 , 0.24675584, 0.16655742, -0.24143380.18
	layer_2	-2.07994739e-04, 8.24208785e-03, -1.37146018e-01, 1.24615294e+00, -6.72997002e+00, 2.20037955e+01, -4.22829683e+01, 4.33524484e+01, -1.80105394e+01, 3.84119093e-01	0,1,2,3,4,5,6,7,8,9	0.3841191 , -0.16607362, 0.41335428, 0.29864177, 0.22447771, -0.07972511, 0.3984914 , -0.3090847 , 0.0412868 , -0.50203955
Epoch_1				
	layer_0	1.43892292e-08, -1.44301530e-06, 6.52130059e-05, -1.75467659e-03, 3.12825489e-02, -3.89050489e-01, 3.46006370e+00, -2.21849214e+01, 1.02012273e+02, -3.30125475e+02, 7.24974308e+02, -1.01269781e+03, 7.95593243e+02, -2.60335608e+02, 1.63853228e-01	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14	0.16385323, 0.500466 , 0.25013566, 0.19779144, 0.19193129, -0.5924296 , 0.0011992 , -0.06613187, -0.08952589, 0.05943337, -0.00692113, -0.24093635, 0.05037161, -0.31999648, -0.23287505
	layer_1	2.07868084e-04, -8.60913180e-03, 1.50572889e-01, -1.44646470e+00, 8.30371073e+00, -2.89709744e+01, 5.94312297e+01, -6.46564264e+01, 2.80441049e+01, -3.46218497e-01	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	-0.3462185 , 0.50113297, -0.6486435 , 0.11134157, -0.09344627, 0.15072656, 0.01577904, 0.2853177 , 0.17956963, -0.2247488
	layer_2	-3.02715961e-04, 1.20367557e-02, -2.00799796e-01, 1.82613163e+00, -9.84227549e+00, 3.19640994e+01, -6.05758669e+01, 6.06254627e+01, -2.42383023e+01, 3.50098968e-01	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0.35009897, -0.07971777, 0.33765432, 0.2506581 , 0.2850958 , -0.29880172, 0.34176072, -0.40072697, 0.05368976, -0.49336466

TABLE 7.3: Fashion mnist Polynomial interpolation

8 Appendix C

This show the results from functional analysis using infima and suprema. This was done to see how the model behaves from random step.

Epochs	Layer 0	Layer 1	Layer 2	Layer 3
epoch_0	0.40959036	0.46359032	0.9613653	0.32951745
epoch_1	0.36584845	0.49697563	0.96023697	0.36655423
epoch_2	0.3301933	0.5642091	0.63975215	0.3680669
epoch_3	0.30853456	0.5716847	0.36992815	0.31536865
epoch_4	0.34363574	0.56093687	0.3300561	0.29856607
epoch_5	0.39195576	0.55077136	0.2971081	0.2878856
epoch_6	0.4052696	0.5065049	0.26835456	0.27463388
epoch_7	0.44024593	0.53192383	0.25899914	0.24772054
epoch_8	0.45340723	0.5347331	0.24562982	0.23491965
epoch_9	0.45467672	0.33169183	0.23045433	0.22632663

FIGURE 8.1: Iris Suprema

[h]

Epochs	Layer 0	Layer 1	Layer 2	Layer 3
epoch_0	-0.50132203	-0.42710978	-0.25783554	-0.8902792
epoch_1	-0.50123787	-0.2853163	-0.2134208	-0.9099802
epoch_2	-0.49662372	-0.4066611	-0.5471867	-0.91547984
epoch_3	-0.5049001	-0.4502339	-0.672131	-0.93354356
epoch_4	-0.5148058	-0.46519384	-0.7061618	-0.9430131
epoch_5	-0.52051955	-0.469594	-0.72763395	-0.9491632
epoch_6	-0.52952784	-0.5910669	-0.7427154	-0.9548965
epoch_7	-0.51370156	-0.45367435	-0.75169885	-0.96260494
epoch_8	-0.5120305	-0.47567165	-0.74739176	-0.9664033
epoch_9	-0.5165135	-0.8216771	-0.7381933	-0.96862495

FIGURE 8.2: Iris Infima

Epochs	Layer 0	Layer 1	Layer 2	Layer 3
epoch_0	-0.536786	0.0638624	0.0218493	-0.940158
epoch_1	-0.536786	0.0638625	0.021849	-0.940159
epoch_2	-0.536786	0.0638625	0.021849	-0.940159
epoch_3	-0.536786	0.0638625	0.021849	-0.940159
epoch_4	-0.536786	0.0638625	0.021849	-0.940159
epoch_5	-0.536786	0.0638625	0.021849	-0.940159
epoch_6	-0.536786	0.0638625	0.021849	-0.940159
epoch_7	-0.536786	0.0638625	0.021849	-0.940159
epoch_8	-0.536786	0.0638625	0.021849	-0.940159
epoch_9	-0.536786	0.0638625	0.021849	-0.940159
epoch_10	-0.536786	0.0638625	0.021849	-0.940159
epoch_11	-0.536786	0.0638625	0.021849	-0.940159
epoch_12	-0.536786	0.0638625	0.021849	-0.940159
epoch_13	-0.536786	0.0638625	0.021849	-0.940159
epoch_14	-0.536786	0.0638625	0.021849	-0.940159
epoch_15	-0.536786	0.0638625	0.021849	-0.940159
epoch_16	-0.536786	0.0638625	0.021849	-0.940159
epoch_17	-0.536786	0.0638625	0.021849	-0.940159
epoch_18	-0.536786	0.0638625	0.021849	-0.940159

FIGURE 8.3: Wine quality Infima

Epochs	Layer 0	Layer 1	Layer 2	Layer 3
epoch_0	0.271268	0.433152	0.710852	0.340739
epoch_1	0.271268	0.433152	0.710853	0.340737
epoch_2	0.271268	0.433152	0.710853	0.340737
epoch_3	0.271268	0.433152	0.710853	0.340737
epoch_4	0.271268	0.433152	0.710853	0.340737
epoch_5	0.271268	0.433152	0.710853	0.340737
epoch_6	0.271268	0.433152	0.710853	0.340737
epoch_7	0.271268	0.433152	0.710853	0.340737
epoch_8	0.271268	0.433152	0.710853	0.340737
epoch_9	0.271268	0.433152	0.710853	0.340737
epoch_10	0.271268	0.433152	0.710853	0.340737
epoch_11	0.271268	0.433152	0.710853	0.340737
epoch_12	0.271268	0.433152	0.710853	0.340737
epoch_13	0.271268	0.433152	0.710853	0.340737
epoch_14	0.271268	0.433152	0.710853	0.340737
epoch_15	0.271268	0.433152	0.710853	0.340737
epoch_16	0.271268	0.433152	0.710853	0.340737
epoch_17	0.271268	0.433152	0.710853	0.340737

FIGURE 8.4: Wine quality Suprema

Epochs	Layer 0	Layer 1	Layer 2
epoch_0	0.503539	0.453452	0.349658
epoch_1	0.369773	0.471551	0.281151
epoch_2	0.453436	0.485431	0.29529
epoch_3	0.447737	0.506655	0.312464
epoch_4	0.460251	0.498464	0.325964
epoch_5	0.455158	0.491195	0.329175
epoch_6	0.46454	0.480222	0.34101
epoch_7	0.424705	0.476817	0.341304
epoch_8	0.408564	0.477376	0.346166
epoch_9	0.435352	0.473131	0.343772

FIGURE 8.5: Fashion MNIST Suprema

Epochs	Layer 0	Layer 1	Layer 2
epoch_0	-0.362493	-0.638219	-0.533886
epoch_1	-0.471981	-0.625439	-0.506985
epoch_2	-0.541072	-0.628476	-0.480705
epoch_3	-0.543973	-0.634973	-0.493699
epoch_4	-0.458454	-0.638315	-0.490237
epoch_5	-0.475573	-0.642078	-0.492348
epoch_6	-0.534287	-0.640529	-0.49689
epoch_7	-0.441587	-0.642617	-0.491418
epoch_8	-0.429568	-0.644871	-0.482943
epoch_9	-0.500042	-0.637073	-0.491749

FIGURE 8.6: Fashion MNIST Infima

Bibliography

[1] *Machine Learning*. URL: https://www.coursera.org/learn/machine-learning.

[2] *10 Machine Learning Methods that Every Data Scientist Should Know*. 2019. URL: https://towardsdatascience.com/10-machine-learning-methods-that-every-data-scientist-should-know-3cc96e0eeee9.

[3] Aegeus Zerium. *Artificial Neural Networks Explained*. 2018. URL: https://blog.goodaudience.com/artificial-neural-networks-explained-436fcf36e75.

[4] Nahua Kang and Nahua Kang. *Multi-Layer Neural Networks with Sigmoid Function-Deep Learning for Rookies (2)*. 2017. URL: https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f.

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[6] Jason Brownlee. "Loss and Loss Functions for Training Deep Learning Neural Networks". In: *Machine Learning Mastery* (2018). URL: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/.

[7] Angel Alfonso Cruz-Roa et al. "A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2013, pp. 403–410.

[8] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[10] Bae-Muu Chang. "Design of Solving Similarity Recognition for Cloth Products Based on Fuzzy Logic and Particle Swarm Optimization Algorithm." In: *KSII Transactions on Internet & Information Systems* 11.10 (2017).

[11] S. Chakraborty et al. "Interpretability of deep learning models: A survey of results". In: *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. 2017, pp. 1–6. DOI: 10.1109/UIC-ATC.2017.8397411.

[12] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models". In: *arXiv preprint arXiv:1708.08296* (2017).

[13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2016, pp. 1135–1144.

[14] Edward Choi et al. "Retain: An interpretable predictive model for healthcare using reverse time attention mechanism". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3504–3512.

[15] Sebastian Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140.

[16] Himabindu Lakkaraju et al. "Interpretable & explorable approximations of black box models". In: *arXiv preprint arXiv:1707.01154* (2017).

[17] F. Chollet and others. *Keras: The Python Deep Learning library*. Astrophysics Source Code Library. June 2018. ascl: 1806.022.

[18] Tsung-Han Chan et al. "PCANet: A simple deep learning baseline for image classification?" In: *IEEE transactions on image processing* 24.12 (2015), pp. 5017–5032.

[19] Hayit Greenspan, Bram Van Ginneken, and Ronald M Summers. "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique". In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pp. 1153–1159.

[20] lakshya /@xtaraim. *A Very Basic Introduction to Feed-Forward Neural Networks*. 2018. URL: https://medium.com/@xtaraim/a-very-basic-introduction-to-feed-forward-neural-networks-97a33b34604f.

[21] Yash Upadhyay /@yashup1997. *Introduction to FeedForward Neural Networks*. 2019. URL: https://towardsdatascience.com/feed-forward-neural-networks-c503faa46620.

[22] Vikashraj Luhaniwal. *Forward propagation in neural networks Simplified math and code version*. 2019. URL: https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250.

[23] Imad Dabbura. *Coding Neural Network - Forward Propagation and Backpropagtion*. 2019. URL: https://towardsdatascience.com/coding-neural-network-forward-propagation-and-backpropagtion-ccf8cf369f76.

[24] Raul Rojas. "The backpropagation algorithm". In: *Neural networks*. Springer, 1996, pp. 149–182.

[25] Francois Chollet. *Deep learning with Python*. Manning, 2018.

[26] Rahul Bhatia. *When not to use Neural Networks*. 2018. URL: https://medium.com/datadriveninvestor/when-not-to-use-neural-networks-89fb50622429.

[27] James Chen. *Neural Network Definition*. 2019. URL: https://www.investopedia.com/terms/n/neuralnetwork.asp.

[28] Justin Johnson Andrej Karpathy. URL: http://cs231n.github.io/convolutional-networks/.

[29] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[30] Iuliana Tabian, Hailing Fu, and Zahra Sharif Khodaei. "A convolutional neural network for impact detection and characterization of complex composite structures". In: *Sensors* 19.22 (2019), p. 4933.

[31] Ujjwalkarn. *A Quick Introduction to Neural Networks*. 2016. URL: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/.

[32] Sumit Saha and Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks-the ELI5 way*. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[33] Ashwin Bhandare et al. "Applications of convolutional neural networks". In: *International Journal of Computer Science and Information Technologies* 7.5 (2016), pp. 2206–2215.

[34] Joris Guérin et al. "Cnn features are also great at unsupervised classification". In: *arXiv preprint arXiv:1707.01700* (2017).

[35] Ali Sharif Razavian et al. "CNN features off-the-shelf: an astounding baseline for recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2014, pp. 806–813.

[36] Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification". In: *arXiv preprint arXiv:1202.2745* (2012).

[37] Dan Claudiu Ciresan et al. "Convolutional neural network committees for handwritten character classification". In: *2011 International Conference on Document Analysis and Recognition*. IEEE. 2011, pp. 1135–1139.

[38] Matthew D Zeiler, Graham W Taylor, Rob Fergus, et al. "Adaptive deconvolutional networks for mid and high level feature learning." In: *ICCV*. Vol. 1. 2. 2011, p. 6.

[39] Aravindh Mahendran and Andrea Vedaldi. "Understanding deep image representations by inverting them". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 5188–5196.

[40] Loren Grush. "Google engineer apologizes after Photos app tags two black people as gorillas". In: *The Verge* (2015).

[41] Laurence Dodds. "Chinese businesswoman accused of jaywalking after ai camera spots her face on an advert". In: *The Telegraph* (2018).

[42] Julia Angwin et al. *Machine Bias*. 2019. URL: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[43] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. "Methods for interpreting and understanding deep neural networks". In: *Digital Signal Processing* 73 (2018), pp. 1–15.

[44] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial Intelligence* 267 (2019), pp. 1–38.

[45] *Everything you need to know about Neural Networks*. URL: https://hackernoon.com/everything-you-need-to-know-about-neural-networks-8988c3ee4491.

[46] Michael Tsang, Dehua Cheng, and Yan Liu. "Detecting statistical interactions from neural network weights". In: *arXiv preprint arXiv:1705.04977* (2017).

[47] Yin Lou et al. "Accurate intelligible models with pairwise interactions". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2013, pp. 623–631.

[48] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[49] *Sensitivity analysis*. 2019. URL: https://en.wikipedia.org/wiki/Sensitivity_analysis#cite_note-Risk_Analysis-1.

[50] Maosen Cao et al. "Advanced Methods in Neural Networks-Based Sensitivity Analysis with their Applications in Civil Engineering". In: *Artificial Neural Networks: Models and Applications* (2016), p. 335.

[51] Jamal Alsakran et al. "Visualization analysis of feed forward neural network input contribution". In: *Scientific research and essays* 9.14 (2014), pp. 645–651.

[52] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[53] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book/. 2019.

[54] Leilani H Gilpin et al. "Explaining Explanations: An Overview of Interpretability of Machine Learning". In: *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2018, pp. 80–89.

[55] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. arXiv: 1602.04938 [cs.LG].

[56] Sameer Singh Marco Tulio Ribeiro. *Local Interpretable Model-Agnostic Explanations (LIME): An Introduction*. 2016. URL: https://www.oreilly.com/content/introduction-to-local-interpretable-model-agnostic-explanations-lime/.

[57] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4765–4774.

[58] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3145–3153.

[59] In: ().

[60] Xuan Liu, Xiaoguang Wang, and Stan Matwin. "Interpretable deep convolutional neural networks via meta-learning". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2018, pp. 1–9.

[61] Hiroshi Tsukimoto. "Extracting rules from trained neural networks". In: *IEEE Transactions on Neural networks* 11.2 (2000), pp. 377–389.

[62] Jin-Song Pei, Joseph P Wright, and Andrew W Smyth. "Mapping polynomial fitting into feedforward neural networks for modeling nonlinear dynamic systems and beyond". In: *Computer Methods in Applied Mechanics and Engineering* 194.42-44 (2005), pp. 4481–4505.

[63] Ronald A Fisher. "The use of multiple measurements in taxonomic problems". In: *Annals of eugenics* 7.2 (1936), pp. 179–188.

[64] P. Cortez et al. "Modeling wine preferences by data mining from physicochemical properties". In: *Decision Support Systems* 47.4 (1998), pp. 547–553.

[65] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].

[66] Patrick Luboobi. *Foundations of Machine Learning : Singular Value Decomposition (SVD)*. 2018. URL: https://medium.com/the-andela-way/foundations-of-machine-learning-singular-value-decomposition-svd-162ac796c27d.

[67] Marco Ancona, Cengiz Öztireli, and Markus Gross. "Explaining deep neural networks with a polynomial time algorithm for shapley values approximation". In: *arXiv preprint arXiv:1903.10992* (2019).