Convolutional Neural Networks for Raman Spectrum Classification

Raghav Patne

Kongens Lyngby 2019

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Abstract

Raman Spectroscopy is a widely used technique for the identification of chemical compounds. It has also been applied in the diagnosis of deadly diseases like Diabetes or Cancer in early stages, as it can be used to detect biomarkers, based on the spectral patterns of certain cells and tissues, that would be otherwise difficult to detect for the doctor. A Machine Learning algorithm, is used to match these spectral patterns, which are otherwise difficult to identify for the untrained eye. Support Vector Machines and Artificial Neural Networks have been popular choices for classifying Raman Spectra for medical diagnostics, but have always required one or more preprocessing steps, like Baseline Correction and Principal Component Analysis(PCA). This Thesis aims to eliminate these steps by investigating the use of a Convolutional Neural Networks(CNN), as a true end-to-end learning tool for classification of raw Raman Spectra to detect Diabetes.

Preface

This thesis was prepared at DTU Compute in fulfilment of the requirements for acquiring an M.Sc. in Engineering.

Lyngby, 16-August-2019

Raghav Patne

Acknowledgements

First of all, I would like to give the biggest thanks to my supervisor Tommy Sonne Alstrøm for the unending guidance, support and encouragement throughout the course of the project.

Also, I would like to express my gratitude to my co-supervisor Juan José Rubio Guillamón, for taking the extra time out for me, whenever possible.

Thank you to my dearest friend Shivam, for always being there and for the moral support.

Lastly, I would like to thank my brother Omkar and my Parents for all the love and support, through the ups and downs.

Contents

Chapter 1

Introduction

Spectroscopy is the study of interaction of light with matter. Light interacts with different kinds of matter in different ways, depending upon the chemical and physical properties of the substance. This interaction can have encoded in it a lot of information about the chemical makeup of the substance. One such important spectroscopic technique, commonly used in the identification and analysis of unknown chemical compounds is **Raman Spectroscopy**. Raman Spectroscopy, is a type of **Vibrational Spectroscopy** [27], in that it exploits the molecular vibrations that result in a substance, after interaction with light, to deduce information about the chemical makeup of the substance. The applications of this phenomenon are wide ranging, across various different industries, like pharmaceuticals, airport drug control, and even in Medical diagnostics [10][25]. Although invented in 1928, and named after its discoverer, `Dr.C.V Raman`, the popularity of this technique has only risen recently, due to the advancements in the instrumentation and techniques required to detect and enhance the weak Raman signal [12][31].

Fig. 1.2 shows a typical setup for collecting a Raman Spectrum. The specifications and the working of these instruments are outside the domain of this work, but it is worthy to note that the excitation laser frequency used in Raman spectroscopy is in the visible or near visible range. Also, the instruments like the optical filter, which is used to remove the Rayleigh scattered light and the detectors which will generate the actual spectrum, may be prone to errors and

be a source for noise in our spectrum measurements. In Raman Spectroscopy, a monochromatic source of light, the frequency of which is known, is shone upon a sample substance. Most of the light will undergo no change (according to a phenomenon called *Rayleigh Scattering*), which means that the wavelength of the light collected is the same as that of incident light, but a very small portion of the light will interact with the molecules to give out light consisting of a variety of frequencies at different intensities (this phenomenon is called *Raman Scattering*). The spectrum of these different frequencies, is very unique to a chemical compound, and thus acts as a molecular *fingerprint* [27]. The Figure 1.1 shows the difference between the Raman spectrum for the compounds ethanol and methanol, which are chemically very similar but their Raman Spectra are very unique. Although it is important to note how both these spectrum's have a peak in the same region of around $3000cm^{-1}$ Raman Shift. This is indicative of the presence of the common alcohol functional group which is present in both these chemicals. For historical reasons, and ones which are not relevant here, the Raman Spectrum is represented as a plot of *Intensity vs the Raman shift* (see Fig. 1.1). The Raman shift is a measure of change of wavelength of the light, and is measured in wavenumbers. The Raman Shift $\Delta\nu = 1/\lambda_1 - 1/\lambda_2$ where λ_1 is the wavelength of the incident light and λ_2 is the wavelength of the scattered light. Since the energy of a photon is related to the wavelength as $E = h\frac{c}{\lambda}$[27], where c is the speed of light and λ is the wavelength of the light, the Raman shift is a measure of energy of the scattered photon.

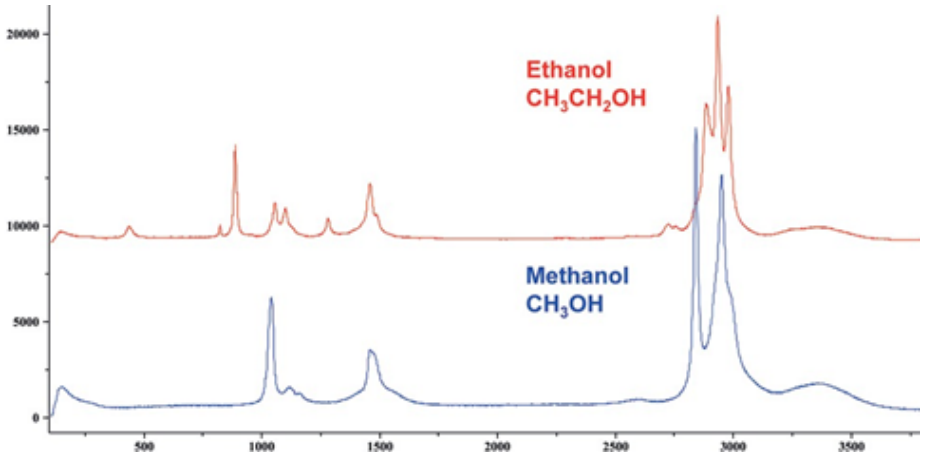


Figure 1.1: Figure showing difference between two chemically similar compounds[1]

[1]This image was taken from `http://www.horiba.com/en_en/raman-imaging-and-spectroscopy/`

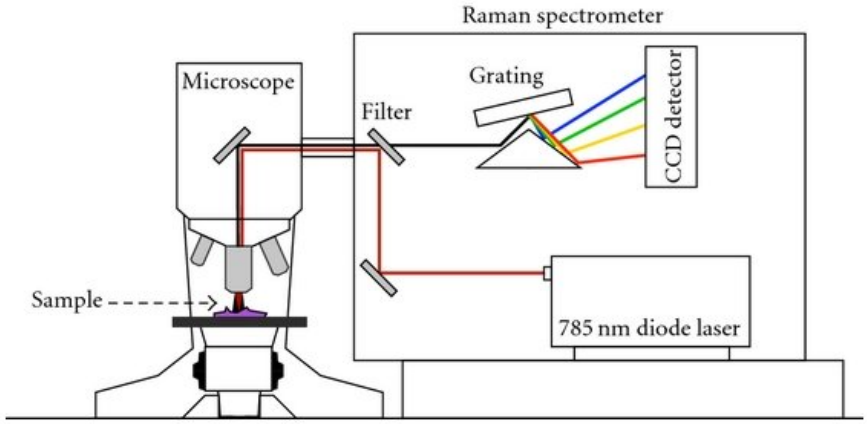


Figure 1.2: Typical setup of a Raman experiment. This image was taken from-[4], courtesy of B. Fenn et al.

A variety of other information can be inferred from a Raman Spectrum, which can help us infer a lot of information regarding the chemical analyte in question. Some of them are listed below [27]:-

1. The peaks obtained at specific frequencies (or Raman shifts) are indicative of the structural make up of the molecule of a substance.
2. Width of the peak can determine the quality of the compound.
3. The intensity at the peak determines the amount of substance contained in a particular analyte.
4. Since Raman Spectroscopy exploits the vibrations of the molecule, it can also be used for analysing chemicals undergoing a reaction as well.

1.1 Diabetes Diagnosis using Raman Spectroscopy

In simpler words, by matching a few Raman spectral patterns, one can perform a qualitative analysis about a chemical substance. For identification of chemical make up of a substance, we are mostly concerned with the peaks present in the spectra. A peak present in the fingerprint region for a certain chemical, as was in the case of the alcohol group in Figure 1.1, is indicative of the molecules being present in the substance.

These phenomena described above can also be extended to the analysis of human tissues and cells, because human tissues and cells are themselves made up of chemical compounds like lipids and proteins, and Raman Spectroscopy can enable us to infer whether these cells and tissues have undergone disease transformation, by checking for disease specific biomarkers[2], thereby enabling the early diagnosis of deadly diseases like Cancer and Diabetes[10]. The highly specific nature of Raman Spectroscopy helps us identifying diseases because the spectrum of a healthy subject is different from that of a diseased patient[32][14]. Furthermore, due to recent advancements in the instruments for clinical diagnostics using Raman Spectroscopy, which allow *in-Vivo*[3] and *non-invasive*[4] measurement of the Raman Spectra [31], has resulted in reducing the costs of these spectral acquisition processes and also reduces the risk of infection to the patient[26]. Another advantage of Raman Spectroscopy is that it is not affected by water, which is a major component of the human body, but it does not interfere with Raman Spectra, unlike other spectroscopy techniques like Near-Infrared(NIR) Spectroscopy[18].

Although, earlier we might have trivialised the process of diagnosis using Raman Spectroscopy, by comparing it with the Ethanol example, where one can differentiate between the two spectra, just by looking at one spectral region(called a *univariate* approach), but in practise, diagnosis using biological spectra is very different from this approach and involves its own set of challenges. Firstly, in many cases, to differentiate between healthy and diseased spectra, we have to investigate the spectra at more than one spectral regions, as can be seen from the figure 1.3 and 1.4, and there is no one specific region. Also, we are never certain of exactly how the spectrum of a diseased patient is going to be different from that of a healthy patient, a very small difference in one particular spectral region can sometimes carry more importance in identifying healthy samples

[2]Biomarkers, as the name suggest, means the development of a phenomenon, or a chemical that is used to infer the presence of a certain condition. For example, presence of certain chemical A may imply the patient is Diabetic. Thus, A is a biomarker for Diabetes.

[3]In-Vivo means the measurement are directly taken from the organisms

[4]Non-Invasive simply means that no incision or puncture was made to the human body to collect the spectra

from diseased samples[14][32]. To answer all these questions, we need to take a multivariate data analysis approach, and make use of a machine learning model to classify raman spectrum, in a higher dimensional space, and thus infer about the differences between the spectrum of a diseased patients and healthy subjects. Although as we will see later, that this approach has its own set of challenges.

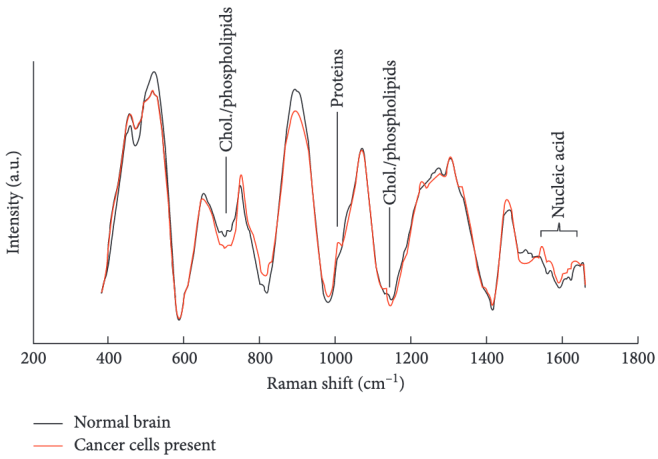


Figure 1.3: Figure showing difference between Raman Spectra of Cancer Cells vs. that of normal cells. Figure taken from [10]

In this Thesis, we are going to investigate the use of Convolutional Neural Networks (see chapter 2) to classify Raman Spectral data of one such biomedical problem - that of classifying Diabetes Mellitus 2 (DM2) cases. Diabetes Mellitus 2, has no cure, but early diagnosis and care can help prolong the further complications that follow[18][7]. The data was obtained from the study carried out by Guevara et al. [18], which was a proof of concept study to classify Raman Spectra of the skin of Diabetic and Non-Diabetic individuals, collected at various skin sites, and to investigate if chemicals called *Advanced Glycation End Products* (AGEs) were responsible for this classification. The study used various statistical tests to prove the same. The study also found that different skin sites contained different sets of AGEs, but also found certain AGE that was common to all skin sites. Previous studies, using Raman Spectroscopy for analysis of Diabetic Patients has focused on monitoring different biomarkers like blood Glucose or Hemoglobin levels, but have required some sort of invasive procedures or restrictions on sampling sites of the patients to collect the spectra[26][17]. This could expose patients to the risk of infections, and thus the study by Guevara et al. proposes a completely non-invasive, optical solution

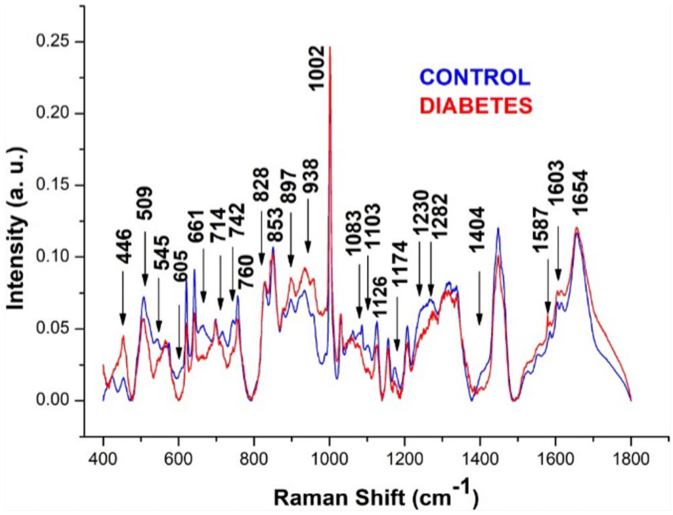


Figure 1.4: Figure showing difference between Raman Spectra of Diabetic and control blood serum sample. Figure taken from [17]

to screen the condition of Diabetes Mellitus Type 2 (DM2). If the Classification model used to classify these spectra can be made robust, and susceptible to noise, then this method can be used easily in a clinical setting to screen the DM2 condition.

1.2 Previously used Methods

In classification of Raman Spectra, each spectrum is sampled at equal interval of the Raman Shifts, and thus each spectrum $\mathbf{x} \in \mathbf{R}^{(1\times N)}$ (the set of real row vectors). Thus the entire dataset is a matrix $\mathbf{X} \in \mathbf{R}^{(M\times N)}$ (the set of real $m \times n$ matrices), where M is the number of different samples and N is the number of equidistant Raman Shifts, the spectrum is sampled at. Thus essentially, each spectrum can be represented as a point in coordinate space $\mathbf{R}^N$, or in other words, in a N-dimensional vector space. The task is then to find a function mapping the set $\mathbf{x}$ to $\mathbf{y}$, $f\colon x \to y$, where $\mathbf{y} \in [1, c]$ and c is the number of different classes the classification model is trying to predict. This mapping, also known as our model, is parameterised by $\mathbf{W}$, called the set of the *model's parameters*. A model is said to be linear, when the output of the model is expressed by some linear combination of the set of model parameters $\mathbf{w}$ and the features

of the vector **x**. In our case the features of every vector is the intensities of the spectrum sampled at fixed intervals of the Raman Shift. The model's parameters and thus the mapping function is estimated using an iterative supervised training algorithm. A subset of the dataset is used for training and the rest of the data is used for testing the model. The goal then is to maximise the performance of the model in classifying the unseen test data correctly. This will be discussed in more detail in chapter 2.

The following is a subjective discussion of some of the most commonly used Machine Learning models to classify Raman Spectral Data. Although this list is not exhaustive, the aim is to discuss the commonly used technologies, and their limitations, to highlight the problem statement.

1.2.1 Support Vector Machine(SVM)

Support Vector Machines(SVM) are a very popular alternative to Neural Networks for classification tasks[25]. SVMs are inherently linear classifiers, but can also be applied very efficiently to non-linear classification tasks, by using kernel functions, which transforms data to a higher dimensional space, where it is easier to find a hyperplane to classify the data.

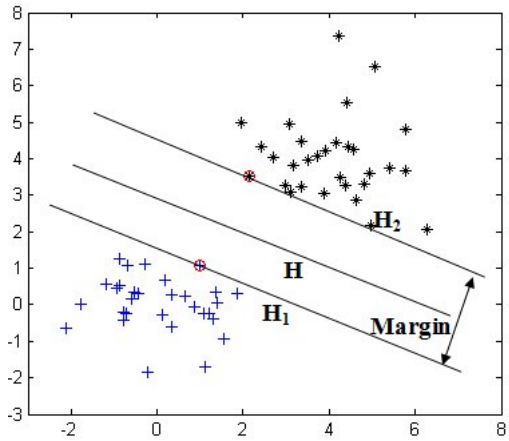


Figure 1.5: Figure showing a Linear Support Vector Machine. Figure taken from [39], courtesy of Zhang et al.

The basic principle for the algorithm in a Support vector machine, is to find a hyperplane that maximizes the distance between the hyperplane and the training samples of two classes. Although there are several variations to this algorithm, but the basic principle is the same. SVM's have been used for classification of Raman spectra and have shown very high performance accuracy for binary classification tasks [37][33]. But to attain a good performance, it usually requires non trivial pre-processing and dimensionality reduction step like Principal Component Analysis(PCA, see chapter 2). Also, multiclass classification becomes computationally very expensive for non-linear tasks, using SVMs[25], as we essentially have to train the n classifiers for separating n classes, and each of these require a transformation of all the dataset to a higher dimensional space.

1.2.2 Random Forests

Random Forests is an ensemble method, that makes use of decision trees to classify new samples. An Ensemble method is one in which, several classifiers are trained on different subsets of the same data, and the final output of the classifier is obtained by a voting scheme, like majority voting. It is a optimisation of decision trees, which usually suffers from the problem of **overfitting** - a common Machine Learning phenomenon, where a model becomes very well tuned to the training data, and does not generalise well, thereby performing very bad on unseen test samples. In a Random Forests algorithm, Multiple Decision trees are created to classify data. The underlying principle of the decision tree is still the same, which is to maximise the *purity gain metric* at each branch of the tree. A purity gain metric is a measure of how well a branch of the tree performs in separating the classes. In Random Forests, a randomly selected subset of the dataset is chosen out of the original dataset, taking only a subset of the features at a time. Several Decision Trees are then created using these subsets.

When a new sample is to be classified, it is classified individually by all trees, and the final class of the sample is computed by a voting scheme, for example, aggregate voting, where the class with the highest number of all the individual tree decisions is chosen as the output of the random forests. Optimisations can then be made to choose a different setting of subset of variables to create the decision trees. With the creation of large number of decision trees, the processing time for a new sample using this algorithm is high. Random Forests algorithm has been used in classification of biomedical raman spectra with a good performance[25][35].

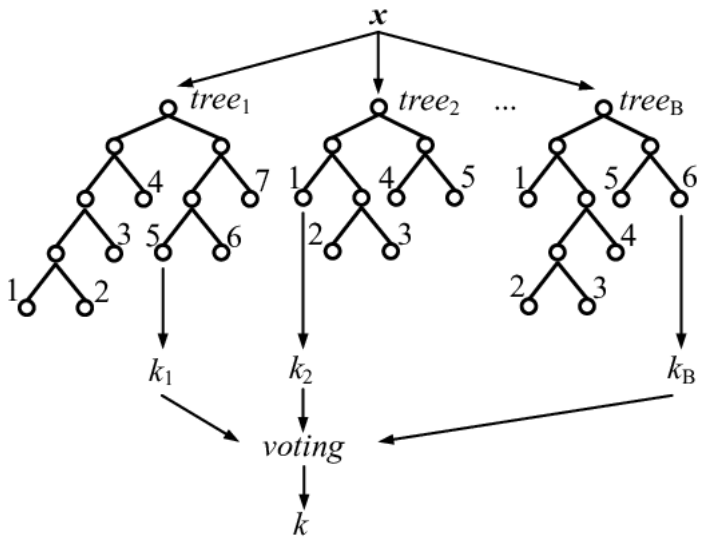
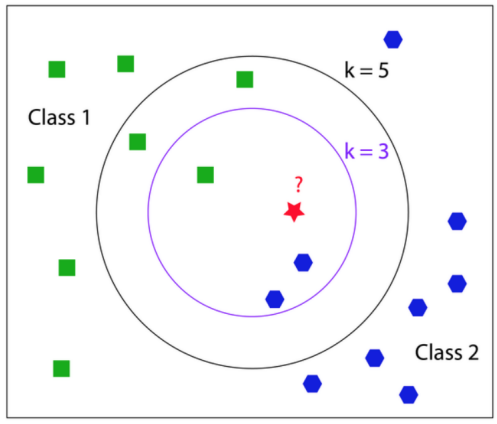


Figure 1.6: Figure showing a general Random Forest Architecture. Figure taken from [15], Courtesy of Gelzinis et al.

1.2.3 Nearest Neighbor Classifier

Nearest Neighbor Classifiers are a class of Non-Parametric models in Machine Learning. In a very high dimensional space, distances can be used as a measure of similarity, in that higher the distance between two data objects, the less similar they are[8], and this is the principle behind Nearest Neighbor Classifiers. As the name suggests, Nearest Neighbor Classifiers, use the neighbors nearest to the point in the high dimensional space to classify it as the same class as this neighbor. There are many variants of this algorithm, but the most prominent one is the K-Nearest Neighbor(KNN) Classifier. In a KNN classifier, K Nearest Neighbor's of a point are computed using a distance metric. The point is then assigned the class which is in majority amongst the K nearest neighbors, as can be seen in Figure 1.7. Although, there are no learnable parameters in this model, there are two Hyper-parameters to choose here, the number K, and the choice of the distance metric. These can be found, using Cross Validation(see chapter 2). There are other variation to the KNN classifier, like the weighted neighbor classifier, where the nearest neighbors are assigned weights according to a certain scheme, like distance from the point in question, and then carry out the majority voting, where each point's vote is weighted by the weights assigned to them. KNN classifier's have been used in the past for classification of biological raman spectra, after preprocessing and a dimensionality reduction[11][5]. A

major drawback to KNN is that we cannot infer much about the relationship between the features, or the contribution of the features, which help in classifying the spectra, as there are no learnable model parameters. Also, classification requires all the dataset, which makes increases the space complexity when the dataset is very large. KNN will be discussed further in chapter 2.



1-NN rule : it will classified as blue hexagon.
3-NN rule : it will classified as blue hexagon.
5-NN rule : it will classified as green square.

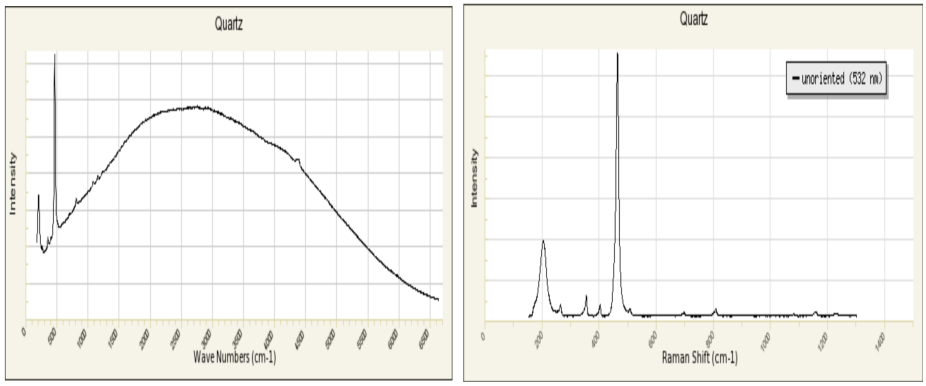
Figure 1.7: Figure explaining the K-Nearest Neighbor Classification Algorithm. Figure taken from [9], Courtesy of Cover et al.

1.3 Problem Description

Thus, we saw earlier that to successfully classify healthy and diseased raman spectra correctly, we need to model the relationship between the spectra at various different spectral regions. Thus essentially, we want a classifier that performs well in a very high dimensional space, because each region is a feature in our dataset. But usually, in vibrational spectroscopy datasets, the number of features is very high, as compared to the number of samples[1]. Thus, as we traverse in higher dimensions of space, the space becomes very sparse, and we are able to model the classifier very easily on the training data, but the classifier does not generalise well on new test data[2], thereby overfitting to the training data. This phenomenon is known as the *curse of dimensionality*, and the models discussed above suffer heavily from it. On the other, if we use less number of features, we are making the model very trivial, and will not be able to model the complex relationships required to classify the healthy spectrum and the dis-

eased spectrum. Thus there exists an optimal number of features to gain the maximum performance from a model. Dimensionality reduction steps like PCA can help, but we are not guaranteed that PCA will capture the information that helps separate the two classes.

As mentioned earlier, all the models mentioned above require pre-processing steps. This hinders the possibility of making a fully automated setup for classification of raman spectra based on these models. Standard pre-processing steps for raman spectra include, signal smoothing, baseline correction, and a dimensionality reduction step, like Principal Component Analysis(PCA) or Linear Discriminant Analysis(LDA). Signal smoothing, as the name suggests is the process of removal of high frequency noise, which may be caused due to the instruments used in the acquisition of the spectra. Baselines in raman spectrum is usually caused by a phenomenon called fluorescence[25]. Figure 1.8, shows a spectrum before and after baseline correction. This is done, by modelling a polynomial to estimate the baseline, after which the baseline is subtracted from the spectrum to give the corrected spectra, as in Figure 1.9. Although there are many robust algorithms to model a baseline, and they also combine the process of signal smoothing, like - the Vancouver Raman Algorithm[40], which is very commonly used in biomedical applications[18], and the Assymetric Least Squares Smoothing[13], they still require some parameter setting which need to be set[25].



(a) Spectrum with Baseline **(b)** Spectrum after Baseline Correction

Figure 1.8: Figure showing the spectrum of the mineral Quartz, before and after baseline correction.[5]

[5]This figure is taken from `http://rruff.info/quartz/display=default/`, Courtesy of Lafuente et al.

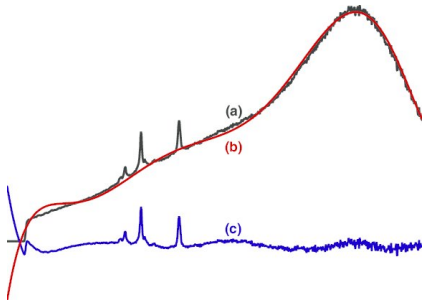


Figure 1.9: Figure explaining the polynomial fitting for baseline. The red line is the polynomial approximated as baseline for the raw spectrum in Black, to give the blue baseline corrected spectrum. Figure taken from [24], Courtesy of Lasch

Pre-processing is a non trivial task, and the wrong kind of pre-processing can lead to a degradation in performance of the classifier, and thus there is not "one standard method" for pre-processing for raman spectra, as we might lose certain information in the preprocessing[1]. Acquarelli et al. showed this same principle in their study, and also found that the best pre-processing strategy can be found through a search procedure, measured using Cross-Validation. The study also found that a shallow Convolutional Neural Network, when applied to raw data, performed better than many of commonly used spectrum classification methods, applied on processed data. Liu et al. applied a deep convolutional neural network, for the classification of the raman spectra of minerals from the RRUFF[23] database[6], which is an open database. They proposed their Deep Convolutional Neural Network(CNN) as an unified solution for the classification of Raman Spectra. Also, their CNN was able to classify raw raman spectra (spectra with all noise artifacts and a baseline) with an accuracy greater than that to classify Baseline Corrected Data, and even better when compared with the performance of models like Support Vector Machines(SVM), and KNN, when applied to baseline corrected data. Thus Neural Networks are able to provide us end-to-end learning, without the need for several non trivial pre-processing steps. Although, there has not been many applications of convolutional neural networks to classify biomedical raman spectra, and thus we are going to investigate the use on Convolutional Neural Networks, as an end-to-end learning tool in the classification of raman spectra of Diabetic patients.

[6] `http://rruff.info/`

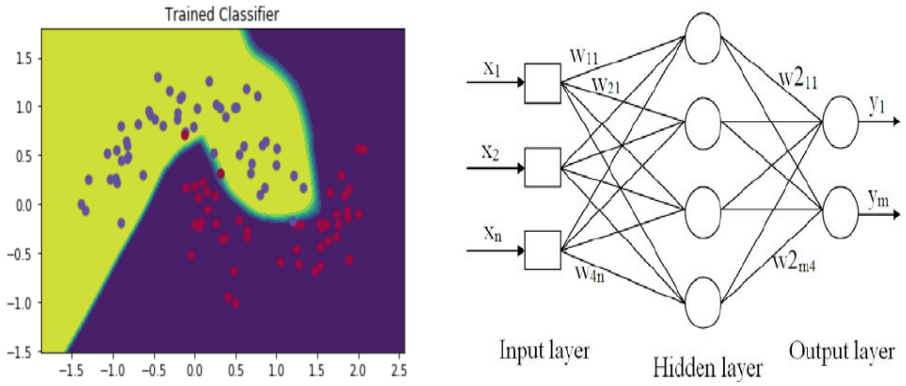
Chapter 2

Methods

2.1 Neural Networks

Neural Networks are a class of highly non-linear classifiers. Neural Networks are highly efficient at modelling data in a very high dimensional space to perform classification tasks. Figure 2.1b represents the basic architecture of a basic **Feed Forward Neural Network**, while Figure 2.1a represents the contour plot of the decision boundary made by a Feed Forward Neural Network. The image Figure 2.1a was made using software packages which are discussed in **??**. This Network can be thought of as a computational device, that takes in an input vector at the **Input Layer**, and produces an output at the **output layer**. For Classification, the output nodes are modelled such that for a n-class classification problem, we will have n nodes, and each node represents the probability of the n-th class. The network also has a hidden layer of neurons. Each layer of neurons is connected to the neurons in the following layer, using connections, which have a certain weight($\mathbf{w}$) attached to it. Please note that the connections only propagate forward and thus, there are no backward connections, or connections between the nodes within the same layer. Thus the connection between the first node of the input layer and the first layer of the hidden layer is $\mathbf{w_{11}}$, as shown in Figure 2.1b. In the discussion henceforth, we are going to use a convention, where the layer number or the node numbering will start from zero, hence this weight will be $\mathbf{w_{00}^{(1)}}$. The superscript 1 signifies the layer which

these weights connect to. The task of the neuron's in each inner layers is to compute a weighted sum of the neurons from the previous layer, and to apply a non-linearity function to it.



(a) **(b)**

Figure 2.1: Figure showing the architecture of a Feed Forward Neural Network, and its decision boundary made for a toy example dataset. The figure Figure 2.1b is taken from [38], courtesy of Zainal Mokhtar and Mohamad-Saleh

Thus, if $\mathbf{a^{(1)}}$ represents the vector of the values of the neurons in the first hidden layer at a given instance, matrix $\mathbf{W^{(1)}}$ represents the matrix of the weights between the first hidden layer and its previous layer, and $\mathbf{b^{(1)}}$ represents the bias associated with each neuron in that layer, σ represents the non-linear activation function applied to this weighted sum, then the relationship is as given in Equation 2.1. The matrix $\mathbf{W^{(1)}}$ is a $(m \times n)$ matrix, where m in the number of neurons in the layer represented by $\mathbf{a^{(1)}}$, and n in the number of neurons in the previous layer. Thus the element in the i-th row and the j-th column of this matrix, $\mathbf{w_{ij}}$ represents the weight of the connection between the i-th node of the previous layer to the j-th node in the current layer.

$$\mathbf{a^{(1)}} = \sigma(\mathbf{W^{(1)}a^{(0)}} + \mathbf{b}) \tag{2.1}$$

The Equation 2.1 is known as computing the **forward pass** on the neural network. Thus our output vector $\mathbf{y}$ can modelled using a proper non-linear activation like a sigmoid function(see Equation 2.2)or the softmax function to

output probabilities for the particular classes. It can be shown that these are both equivalent, when the number of classes is 2, as is in our case.

$$\sigma(\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{1} + \mathbf{e}^{-\mathbf{x}}} \tag{2.2}$$

$$\sigma(\mathbf{x_i}) = \frac{\mathbf{ex_i}}{\sum_{\mathbf{j=1}}^{\mathbf{j=K}} \mathbf{e}^{\mathbf{x_j}}} \tag{2.3}$$

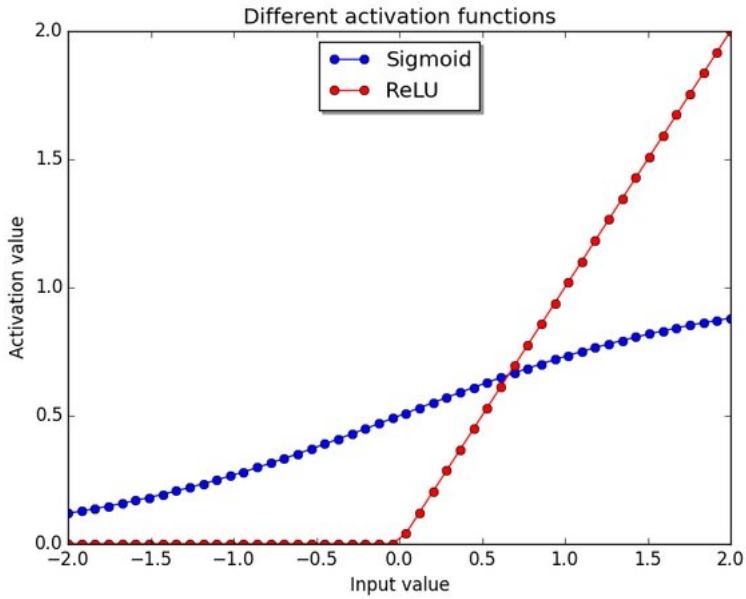


Figure 2.2: Figure showing the Sigmoid Activation and the ReLu activation function. This figure is taken from [16],courtesy of Glauner

For the inner layers as well, sigmoid activation function can be used, but its gradients (slope of the function) towards the edges is very small and tends to zero, which can make the training very slow. A better alternative, and one that is widely used is the Rectified Linear Unit function, also called the ReLu function as shown below.

$$\mathbf{ReLu}(\mathbf{x}) = \mathbf{max}(\mathbf{0}, \mathbf{x}) \tag{2.4}$$

2.2 Training Neural Networks

If you recall in section 1.2, we saw that every model is parameterised, by a set of parameters $\mathbf{W}$. Neural Networks are no different in that sense. The set of parameters in the case of Neural Networks, are all the weights between all the connections and the biases associated with all the nodes. The parameters in a neural network, are trained by finding the maximum likelihood estimation of a distribution of the outputs y of the neural network $p(y|x, \theta)$, where θ is the set of all of our models parameters. This can be done by modelling a cost function which is the negative of the log of this likelihood function, $\mathbf{C} = \mathbb{E}[-\log(\mathbf{p}(\mathbf{y}|\mathbf{x}, \theta)]$. Thus, the task of estimation of our model's parameters θ is reduced to that of an finding the values of the parameters, for the minima for the cost function. Thus we want to find the parameters where the derivative of the cost function $\frac{\partial C}{\partial \theta} = 0$. This means, that finding the optimal parameters is a search problem in a high dimensional space, to find the region for zero gradient. **Backpropagation** is algorithm to find the gradient of each parameter with respect to the cost function, $\mathbf{g_k}$.

Gradient Descent is the algorithm to traverse in this high dimensional space. The basic underlying principle is that we want to traverse in the opposite direction of the gradient $\mathbf{g_k}$. Thus at the (k)-th iteration of the algorithm our parameters change as- $\theta_{(\mathbf{k+1})} = \theta_\mathbf{k} - \eta \mathbf{g_k}$, where η is a parameter called step size. There are several optimisations made to this algorithm, in the Adam's Optimisation Algorithm[22], and speeds the process of convergence.

2.3 Convolutional Neural Networks

Convolutional Neural Networks(CNN) are a step up on the Feed Forward Neural Network. In Convolutional Neural Networks, for a forward pass, we use sliding convolutional kernels to perform the convolution operation to give the feature map in the next layer. The advantages of Convolutional Neural Networks are that there is **parameter sharing**, as the same convolutional network slides

over the entire layer, and **translational invariance**, which means that a spatial shift in a pattern is still detected in the subsequent feature maps formed. Parameter sharing reduces the total number of parameters to train, and thus helps avoid the problem of. We use 1-d convolutional filters in our model as our raman data is 1-dimensional itself.

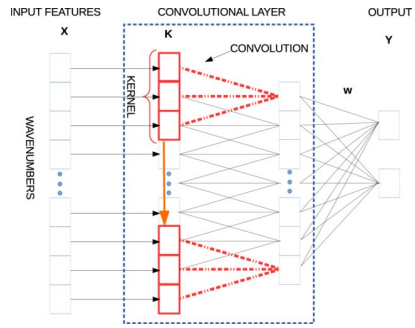


Figure 2.3: Figure showing the convolution operation. This image is taken from [1], courtesy of Acquarelli et al.

The convolution output for the layer a^1, with a convolutional kernel $\mathbf{k}$ of size m can be defined as follows.

$$\mathbf{a_j^{(1)}} = \sigma(\sum_{\mathbf{l=0}}^{\mathbf{l=m}} \mathbf{k[l]a[j+l]} + \mathbf{b_j})$$

CNNs have been very successful in large scale computer vision applications, and are a great tool to provide end-to-end learning.Acquarelli et al. had proposed, in their paper-[1], that CNN when applied to raw spectral data, the learned kernels is able to perform smoothing and baseline correction on the signal. Thus, we would like to investigate, whether a Deep CNN, which is trained on many more samples, than the shallow CNN, is able to learn better features than the shallow CNN.

2.3.1 Network Architecture Used

For the set of experiments, two different types of convolutional Neural Network architectures were trained. We shall call them, Shallow CNN and a Deep CNN. A variant of the convolutional neural network proposed by Acquarelli et al., was

used, as our shallow CNN. The architecture of the neural network is as shown in Figure 2.4. The Deep CNN that was used, was that proposed by Liu et al. in their paper - [25]. The shallow CNN was trained on both Baseline Corrected Data, and Raw Data. The **hyperparameters** of the model, were tuned using a randomised grid search algorithm, which will be described in section 2.4. The Deep CNN was trained on the RRUFF[23] dataset, and then used for the diabetes dataset after Transfer Learning. The architecture of the Deep CNN is shown in Figure 2.5.The maxpool layer is used, as a dimensionality reduction. The maxpool layer slides a window over the feature maps, and outputs the maximum over a non overlapping region thereby reducing the dimensions. The **Batch Normalisation** was used in both the CNN's to normalise the activation's of the inner layers, to solve the problem of *internal covariate shift*, as suggested in - [20]. In Batch Normalisation, each node in every layer of the neural network is normalised by subtracting mean and dividing by the standard deviation, and then expressed . Thus if z_i^1 is the value in the i-th node of the first layer, we first normalise them as $z_{inormalised}^1 = \frac{z_i^1 - \mu}{\sqrt{\sigma^2 + \epsilon}}$, where μ is the mean, σ is the standard deviation, and then each node is expressed as $z_i^1 = \gamma z_{inormalised}^1 + \beta$. $\gamma and \beta$ are learnable parameters and are therefore learned during training. ϵ is used for numerical stability.

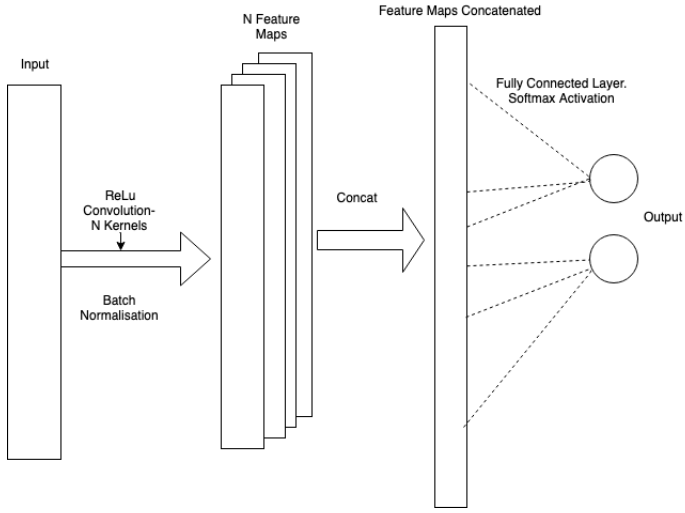


Figure 2.4: Figure showing the architecture of shallow CNN implemented. The number of convolution kernels N was tuned using Random Grid Search

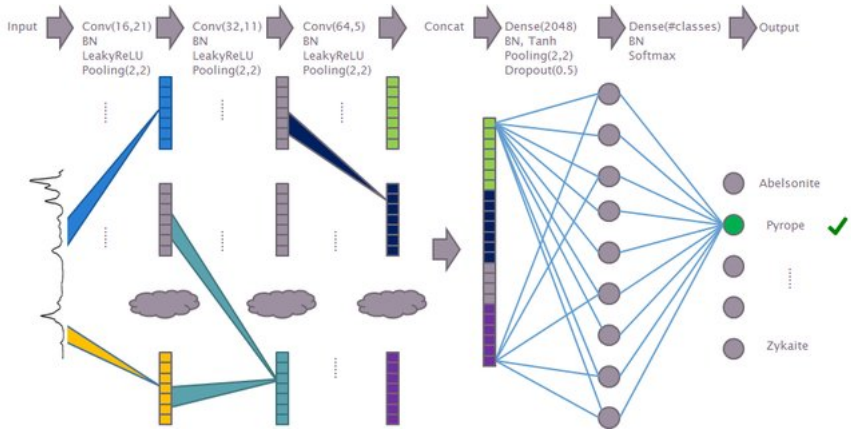


Figure 2.5: Figure showing the architecture of Deep CNN.

2.3.2 Loss Function

The loss function that has been used for the shallow CNN is the Cross Entropy Loss, defined as,

$$C = -\sum_{(n=1)}^{n=N}\sum_{k=1}^{k=K} y_{nk}\log y_{nk}^{output}$$

, where N is the total samples, and K is the total number of classes. Thus, y is a vector the outputs, and y_k is the k-th element in that vector. The loss function used for the Deep CNN is the weighted cross entropy loss, defined as follows,

$$C = -\sum_{(n=1)}^{n=N}\alpha_n\sum_{k=1}^{k=K} y_{nk}\log y_{nk}^{output}$$

where α_n is the inverse of the number of samples belonging to the class of the sample n.

2.3.3 Regularisation

For the shallow CNN architectures, L2 regularisation was used. In L2 regularisation, the cost function is added the term $\lambda||w||_2^2$ to penalise the weights to overfitting the data. The hyperparameter λ was tuned using the grid search(see section 2.4).

Dropout was used in Deep CNN as a regularisation technique. In dropout regularisation, some of the nodes are pinned to zero, during training, as suggested by [34].

2.3.4 Data Augmentation

The distribution of data per class for the data obtained from the RRUFF[23] database was not equal and the number of samples per class were also less, hence data augmentation was done to increase the number of training spectra. The data augmentation was done in an offline manner, as in the training data was augmented first, and then training was done. The Data Augmentation scheme suggested by Liu et al. was used, and the following steps were taken.

1. Each Spectrum was shifted left and right randomly, by a a few wavenumbers(Raman Shift), chosen randomly.

2. Random Noise proportional to the magnitude at each wave number was added. This was modelled using a Gaussian distribution.

2.3.5 Transfer Learning

To investigate if the deep CNN has learned features, that can be used, for the diabetes dataset, the Deep CNN with its learned weights and biases, were stored, and the weights of the first convolutional layer were loaded in the architecture as shown in Figure 2.6. The layer was frozen, as in the layer is only used in computing the forward pass on the network, but during training, the weights of the layer are not updated. Only the Last fully connected layer, was trained during the training. This is because we want to use the first layer of the Deep CNN as a feature extractor.

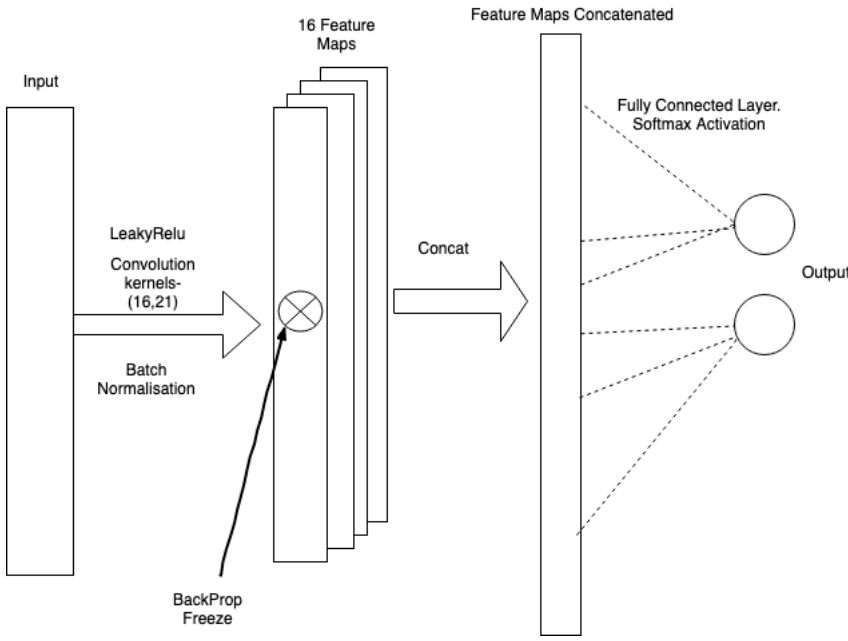


Figure 2.6: Figure showing the transfer learning architecture.

2.4 Hyperparameter Tuning

The various hyperparameters for the Deep CNN are reported in the paper [25], and they are as follows

1. Number of kernels in the respective layers are - 16, 32 , 64
2. Size of the kernels in the respective layers are - 21, 11, 5
3. Dropout Rate - p = 0.5
4. Number of Epochs - 50
5. Adams algorithm hyperparameters - Learning Rate $= 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-8}$

The hyper parameters of the shallow CNN was tuned, using a search procedure, called the Randomised Grid Search [6]. In randomised grid search, all hyperparameter's to be searched are arranged in a grid, and the best hyperparameter setting is selected by trying out a random subset of the points in the grid, which

is different from a full grid search where all the points in the grid search are evaluated, and thus it is a trade off between computational time and quality of the result. Thus at every node of the grid, the neural network is trained using the hyperparameter setting the node represents, and the performance of the network is evaluated using K-fold cross validation. The range of values of the hyperparameters for the training of the shallow CNN, were taken from [1]. They are as follows:-

1. kernel size $\in [2, 20]$
2. stride of convolution $\in [2, 20]$
3. Number of kernels $\in \{1, 2, 3, 4\}$
4. λ for L2 Regularisation : $\in \{0.001, 0.01, 0.1, 0\}$

2.5 Principle Component Analysis(PCA)

Principle Component Analysis is a dimensionality reduction step, which projects the data onto a new set of axes. Principle Component analysis, can help us visualise some of the discriminative features in the data. The new axes are chosen, such that they account for maximum variance in the data. PCA was applied to the dataset to visualise the data on a new set of axes, and also as a dimensionality reduction step for the KNN classifier that was implemented, to see if it helped improve the performance of the classifier. In PCA, the dataset was mean subtracted, and the principle components were obtained from singular value decomposition. In singular value decomposition, a data matrix $\mathbf{X}$ is represented as a product as

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}}$$

, where $\mathbf{U}$and $\mathbf{V}$ are orthonormal matrices and Σ is a diagonal matrix, the squares of diagonal elements of which explain the fraction of variance explained by the principle components in their order. V is the orthonormal matrix, the columns of which are the vectors which represent the principle components.

2.6 Cross Validation

Cross Validation is a technique to get an estimate of the generalisation error of a model. The generalisation error is defined as the *expected value* of the test error, or in layman terms can be called as the test error computed on

infinitesimal amount of data. Although when we dont have that amount of data, we have to make approximations. Thus the data is split into training sets and test sets. The training set is used to train the model, and the test data is used to compute the test error on the trained model. There are 3 major variants of doing cross validation, the holdout method, the K-Fold Cross Validation method and the Leave One Out Method. Cross validation was used to compute all our metrics. When tuning a hyperparameter, as in the number of neighbors K, in KNN, 2- layer Cross Validation was used. This was done to ensure that we get a true unbiased estimate of the model's performance. For the 2-layer cross validation, we did 5-fold cross validation on the outer loop and Leave-One-Out Cross Validation on the inner loop. The scores for all the nodes in the grid search to find the hyperparameters for the neural network was also done using Cross Validation.

2.7 Model Comparison

To compare the CNN's performance with a baseline model, which was a KNN classifier, a number of metrics were measured. The accuracy of the classifiers, the sensitivity and the specificity, the AUC-ROC (Area under curve of the receiver operating characteristic). The Receiver Operating Characteristic is a plot which plots the False Positive Rate(FPR) against the **True Positive Rate(TPR)**, under various threshold values of dividing the model's prediction as positive or negative. . The 95 % credibility interval for the difference in generalisation error of the two models was found using the student's t-distribution, to see if the performance of one classifier is statistically significantly better than the other. For binary classifiers, and in many biomedical studies, it is sometimes beneficial to report the **sensitivity** and the **specificity** of the model to judge the model performance. The sensitivity is also called **Recall** or the **True Positive Rate(TPR)**. The specificity is also called the **True Negative Rate(TNR)**. These terms can be expressed by the phrase, sensitivity or recall will tell us the fraction of diabetic patients that are classified as diabetic, by the model, while specificity is the fraction of all the healthy patients that were classified as healthy. The Term **True Positive(TP)** is used for the cases where a positive case is identified as the correct class, and **False Negative(FN)** for the cases where a positive case is identified as negative. The definitions of **False Positive(FP)** and **True Negative(TN)**, can be thus analogously inferred. All these 4 values can be visualised in a graphical manner by plotting the **Confusion Matrix**. The sensitivity and the specificity can be defined as follows:-

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

$$FalsePositiveRate(FPR) = \frac{FP}{FP + TN}$$

To see whether the difference in performance of the CNN and the KNN are statistically significant or just a result of random variations in the dataset caused by Cross Validation, we used the students t-distribution test to compare both the models. The model's were trained and evaluated for test errors on the same splits of a 10-fold Cross Validation. The true difference of the test error is then assumed to be the mean of a Normal distribution, where these 10 estimates of the difference of test errors are sampled from the same normal distribution. We then use the student's t-distribution test to compute the 95% credibility interval for the true generalisation to lie in it. If the interval contains zero, then it cannot be said that either of the model is better than the other.

2.8 Software Used

All Code was written in Python version 3.7[1]. The following is a list of packages that were used, with a list of how they were used.

1. For the Neural Network training, model saving and loading, Pytorch [29] was used, which is a python library, and abstracts many standard Deep Learning tasks like back propagation, gradient descent, regularisation.

2. For data manipulation and data visualisation tasks, Scipy[21] packages Numpy[36], and Pandas[28], and Matplotlib [19] were used. All Plots and images were prepared using [19].

3. For implementation of K-Nearest Neighbor, and cross validation, Scikit-Learn[30] was used

4. For the grid search skorch[2] along with Scikit-Learn[30] was used.

5. PseudoVoigt package written by Alstrøm et al. in Matlab[3]

6. Baseline Correction and Normalisation of spectra was done using the pre-processing kernel written in Matlab, provided by Guevara et al.

[1] `https://docs.python.org/3/`
[2] `https://skorch.readthedocs.io/en/latest/?badge=latest`
[3] `https://se.mathworks.com/products/matlab.html`

2.9 Hardware Specifications

1. Training of the Shallow Networks, and the K-Nearest Neighbors, and computation of all metrics was done, on my local machine, which has a 2,7 GHz Intel Core i5 processor, with 8GB of Memory.

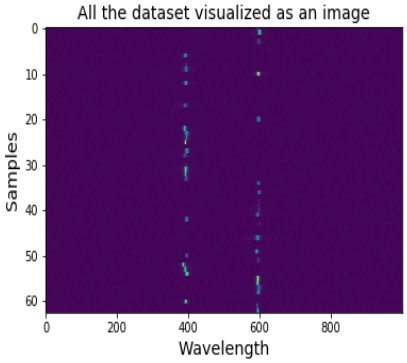
2. The Training of the Deep CNN and the hyperparameter search was carried out on the DTU High Performance Computing(HPC) Clusters on 4 Intel Xeon Gold 6126 (12 core, 2.60GHz) processors, with a maximum memory requirement set to 10GB .

Chapter 3

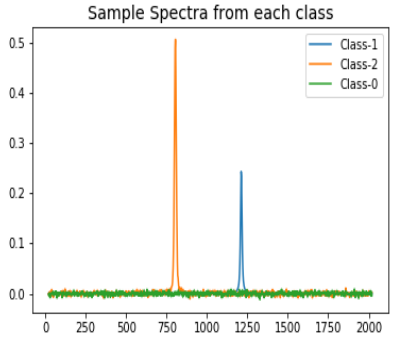
Data,Experiments, and Results

3.1 Simulated Data Experiments

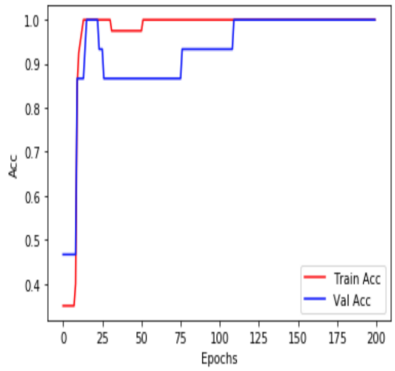
The figure 3.1 shows the simulated dataset that was used to do a sanity check on the CNN classifier. The data was generated from the software package provided by Alstrøm et al.. The dataset was created using the *pseudovoigt* package, to create 60 samples in total, with 20 per class, to create peaks at $1200cm^{-1}$ for class label 2 and at $800cm^{-1}$ for class label 1. Class zero was modelled by random Gaussian noise. 40 of the samples were used for training, and 20 were used for testing. The spectrum intensities for class 1 and 2 were picked randomly between the range $[0, 1]$. The shallow CNN described earlier, was used, random hyper parameter setting. The Network performed very well, as can be seen from Figure 3.1c. Figure 3.1d shows that the network has identified the important spectral regions very well.



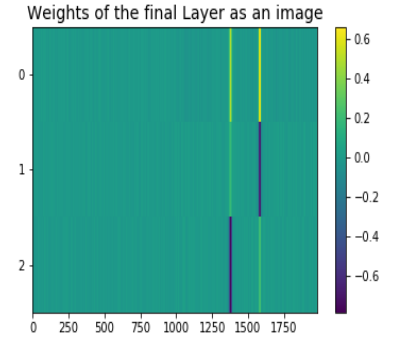
(a) The entire dataset visualised as an image



(b)



(c) The Performance of the Network



(d) The weights of the final layer

Figure 3.1: Sanity Check with Simulated Dataset

3.2 Baseline Corrected Data Experiments

The dataset was obtained from Guevara et al.'s study-[18] through their kaggle[1] website page. The dataset consists of 4 files containing the raw raman spectra from 20 patients, each file containing spectra sampled at different locations of the body for every patient. 11 of the patients are Diabetic and 9 are Healthy. As suggested by the authors of the paper, Guevara et al., all spectra were cropped to the region $800cm^{-1}$ to $1800cm^{-1}$. Thus, each of the files is 20×1000 spectrum, with 11 of class 1 and 9 of class 0, with the entire dataset being a 80×1000 large dataset, with 44 positive class and 36 of negative class. The kernel for the baseline for preprocessing of the raman spectra was also provided by the authors. The Vancouver Raman Algorithm(VRA) was used for Baseline Correction[40], and the data was normalised using the standard normal variate. The Figure 3.2 shows the spectrum, after baseline correction and normalisation. The first image, within every color is that of a diabetic patient, while the second one is that of a Healthy patient. No clear deviation is observed between the two spectra. Hence, indeed we do require a multivariate approach to this problem.

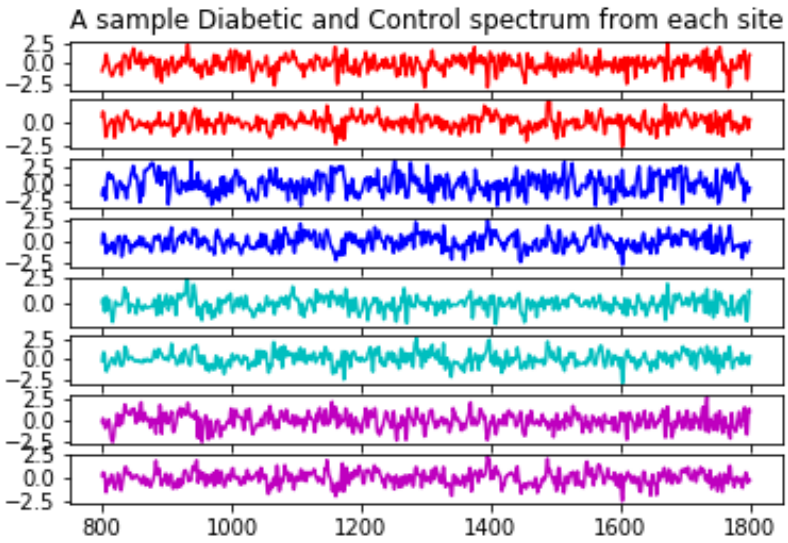


Figure 3.2: Sample Baseline Corrected and Normalised Spectra from each sampling site. The 4 colors each represent a sampling site.

[1] https://www.kaggle.com/codina/raman-spectroscopy-of-diabetes/version/8

3.2.1 PCA Analysis

The data is plotted onto the first 2 principle components that account for maximum variance. This is done to check for if this is a trivial problem, as sometimes PCA may lead to some discriminative features within the dataset to be explored.

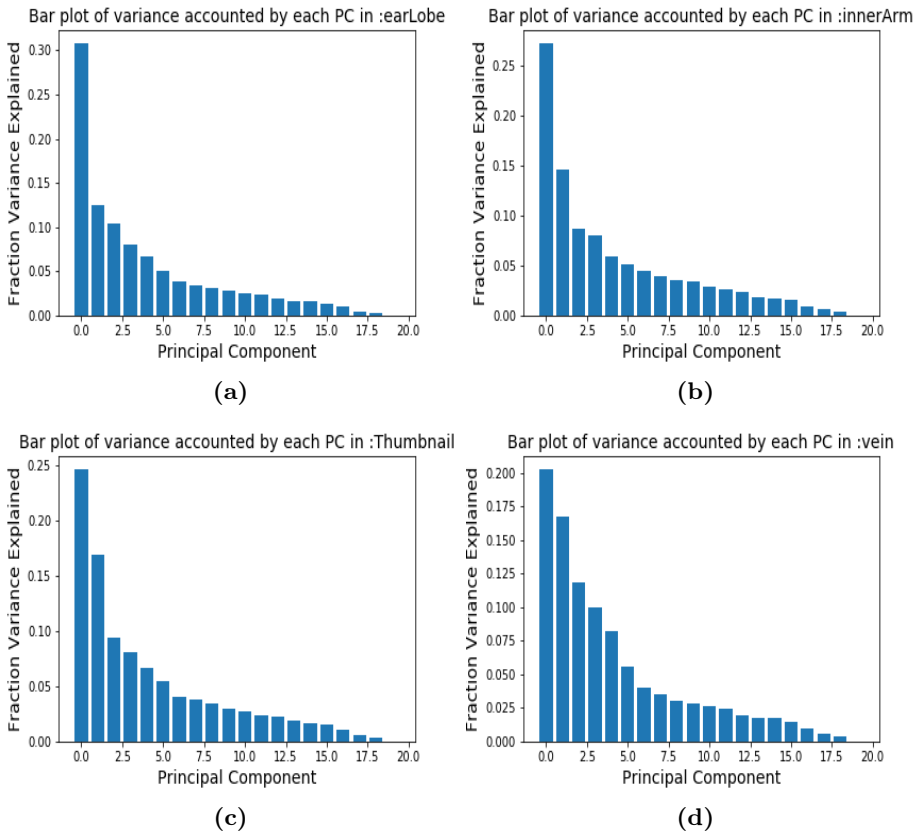


Figure 3.3: Bar Plot showing the fraction of variance explained by each Principal Component

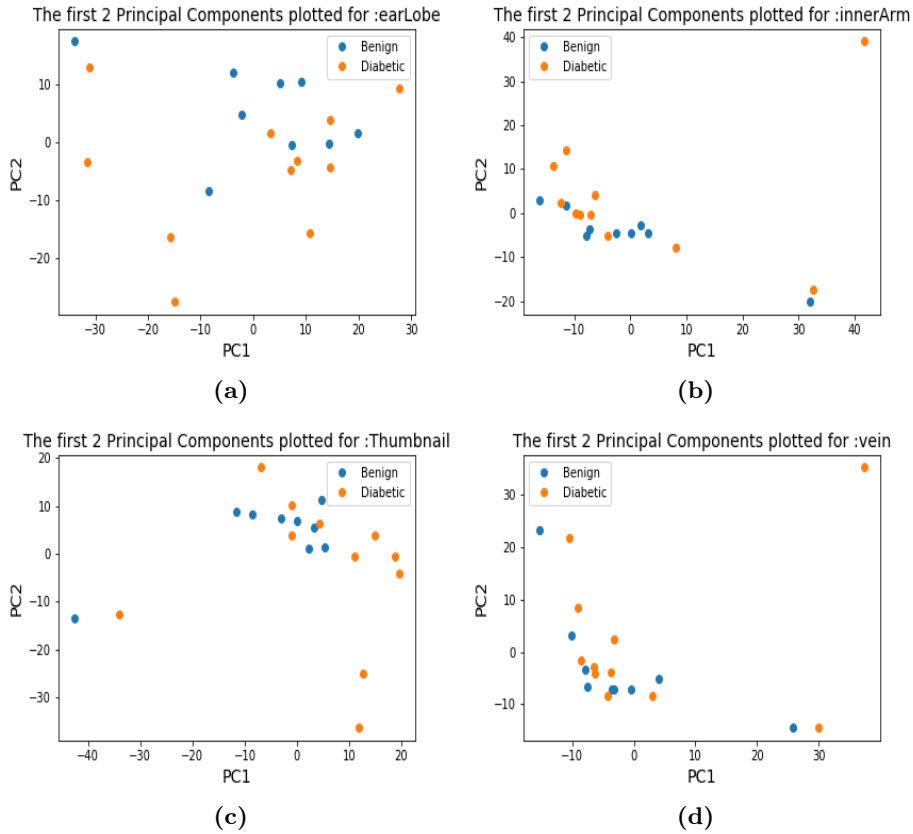


Figure 3.4: Projection of Data onto first 2 Principal components for each sampling site

The Data does not show any clear separation when projected onto the first 2 principle components. Also the first two principle components only account for less than 45% of the variance in the data, and hence, it could be a reason why we could not see the separation.

3.2.2 K-Nearest Neighbors to classify each file

K-nearest neighbor classifier was applied to each dataset, again to check for triviality of the problem. The dataset was projected onto the principle components, before classification. Only training data was used to find out the principle components. The test data was projected onto these components to ensure no amount of test data is used for training. 2-fold cross validation was done to get the unbiased estimate of the model's generalisation error. 5-fold cross validation was used on the outer loop to compute the models performance, and a Leave-One-out Cross validation scheme was used to tune the value of K. The **Euclidean Distance** was used in all the KNN classifiers implemented henceforth.

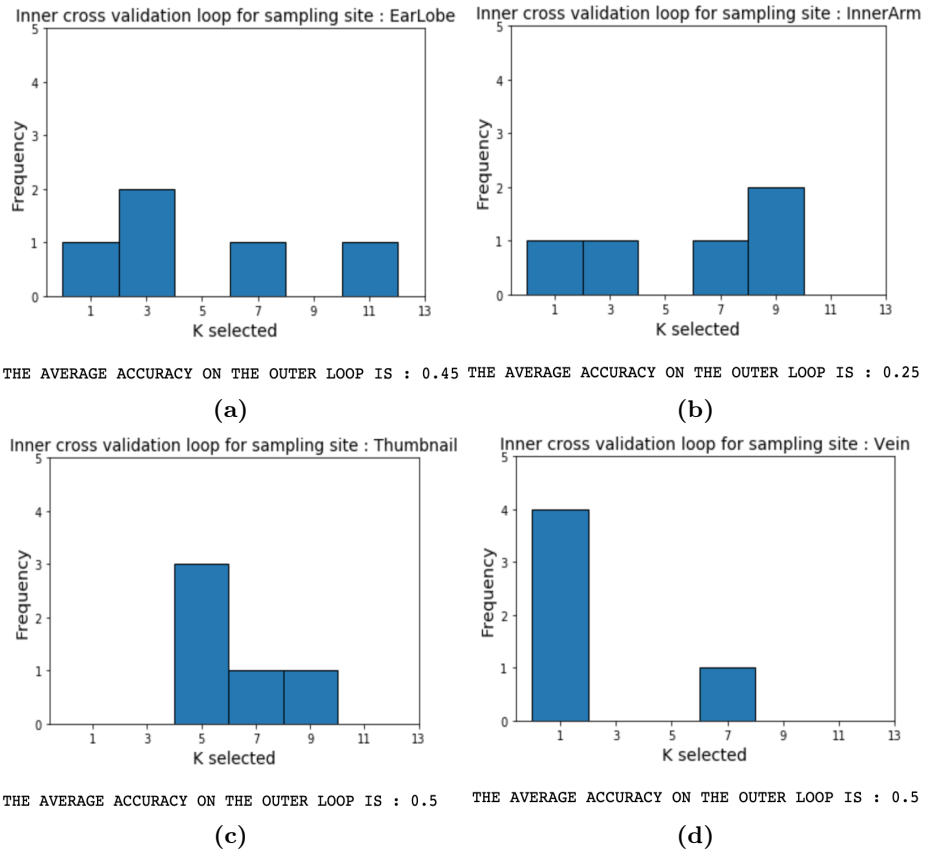


Figure 3.5: Standard Output of the 2-layer Cross validation on each sampling site

As we can see, the KNN model performs worse than random guessing, and has a high variance, and there was not a single sampling site, which performed better than the random guessing. The random guessing accuracy is 55 %, because we have 11 diabetic cases and 9 healthy cases. Thus randomly predicting any one of this class will give us an accuracy of 55%.

3.2.3 PCA-KNN Classifier on entire dataset

The entire dataset was then used for classification. Again PCA-KNN classifier was applied, as before. PCA was done only on the training data, so as not to get a unbiased estimate for the performance on the test set. 2-fold cross validation was used to compute the true unbiased estimate of the model's generalisation error. 5-Fold Cross validation was used on the outer loop to compute the metrics, while Leave-One-Out was used in the inner loop to tune the hyperparameter. The Hyperparameter K was found using an inner Cross validation loop, and found to be k=1. The Figure 3.6 shows the performance of the model. The **sensitivity** of the model was found to be **84.09%** and the **specificity** was **88.88%**.

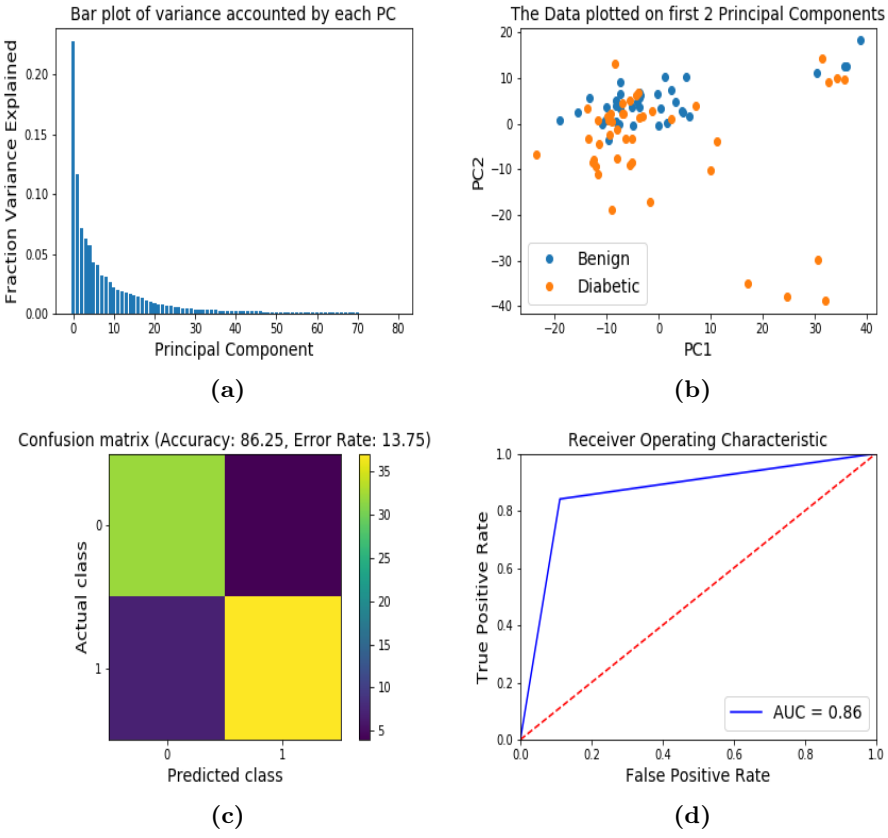
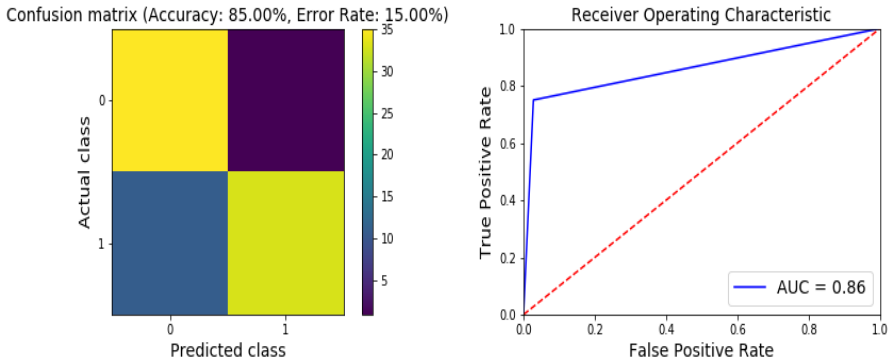


Figure 3.6: Outputs of the PCA-KNN Classifier(K=1)

3.2.4 Applying the KNN classifier to the entire dataset

The entire dataset was used, without any dimensionality reduction. 2-fold cross validation was again used to compute all metrics and the inner loop was used to tune K, which was found to be 1 again. The **sensitivity** and **specificity** of this model was **75% and 97.22%** respectively.



(a) **(b)**

Figure 3.7: Confusion Matrix and ROC(Receiver operating Characteristic) plot of the KNN classifier(K=1)

3.2.5 Convolutional Neural Network

The hyper parameters of the shallow CNN were searched using the random grid search method discussed earlier. The search was run on the High Performance Computing Clusters over 1000 randomly selected combinations out of 5000 total possible combinations. The score for each grid point was computed using 5-fold cross validation. The values were as follows:- Number of kernels : 2, kernel size:12, stride = 17, λ for l2 regularisation:0.01. The shallow CNN was trained using the Baseline Corrected and Normalised data, and the following metrics were computed using K fold Cross validation. The **sensitivity** and the **specificity** of the model is **84.09%** and **77.78%**

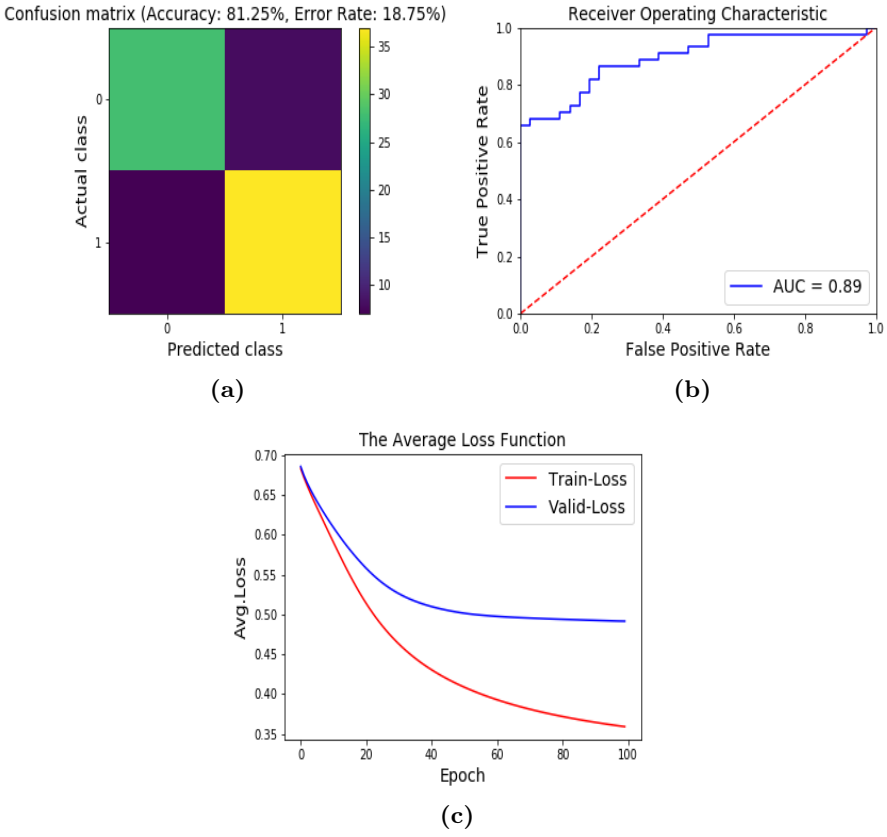


Figure 3.8: Cross Validation of the Convolutional Neural Network

3.2.6 Comparison of Both Classifiers

The 95 % credibility interval (thus $\alpha = 0.05$) for the true difference in both the model's generalisation error was found using the student t-distribution. Using 10-fold Cross validation, the difference in the test errors of both the model's, computed on the same data split, is found for the 10 folds, and these 10 difference is approximated to be a normal distribution. Since zero is contained in the credibility interval,$[-5.50, 15.50]$, it cannot be concluded that one classifier is significantly better than the other.

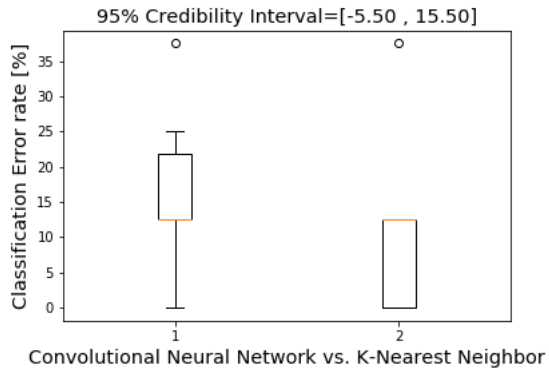


Figure 3.9: Box Plot of the Test Error Rate in ever cross validation Loop

3.3 Raw Data Experiments

The raw data was collected from the kaggle page, and cropped to the region $800cm^{-1}$ to $1800cm^{-1}$, as explained in section 3.2. The images on the top indicate Diabetic spectra and the ones in the bottom indicate the Healthy Spectrum. Clearly the spectrum has a baseline, and shows no clear deviation between the healthy spectrum and the diabetic spectrum.

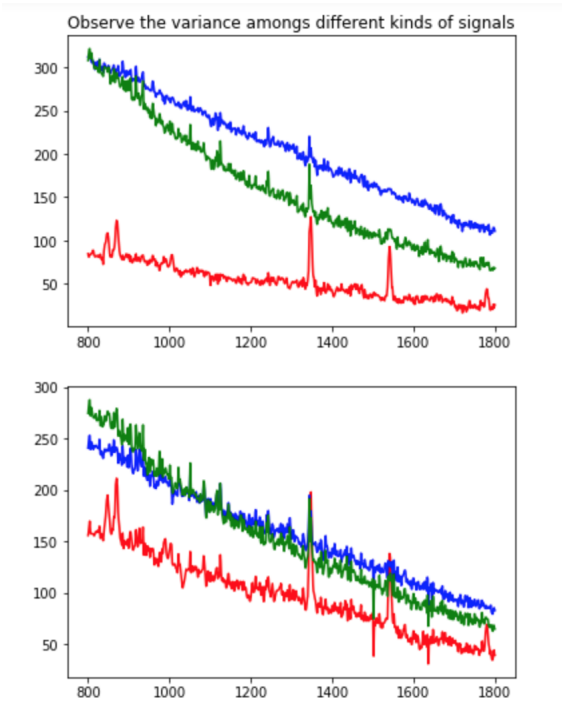


Figure 3.10: Few of the samples of the raw data. The figure on the top is diabetic spectra, of the same patient at 3 different sites.

3.3.1 Convolutional Neural Network

As discussed in subsection 3.2.5 earlier, again the hyperparameters of the shallow convolutional neural network were tuned, using the randomised grid search. The hyperparameters found were :- λ:0.01, stride:7, number of convolutional kernels:4, size of each kernel:8. These Hyperparameters were then used for the CNN, and by 5-fold cross validation, the following metrics were computed. The **sensitivity and specificity** were **95.45%** and **91.66%** respectively.

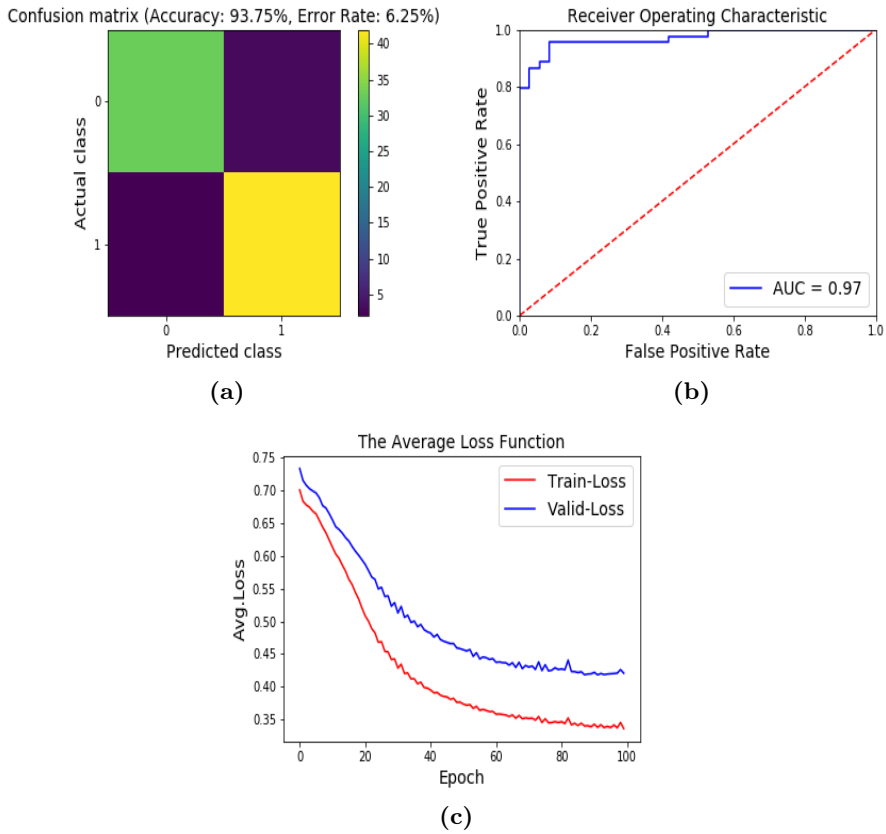
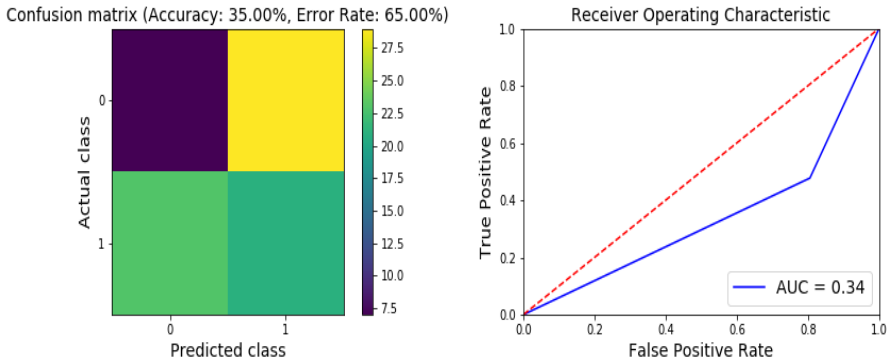


Figure 3.11: Cross Validation of the Convolutional Neural Network

3.3.2 K-Nearest Neighbor Classifier

The K-Nearest Neighbor Classifier has an accuracy of **35%**, which is worse than random guessing(**55%**) as can be seen from the plots below. The **sensitivity** and the **specificity** of the model are **47.77%** and **19.44%**



(a) **(b)**

Figure 3.12: Performance of raw Data with KNN

3.4 Deep Neural Network Experiments

The Deep Neural Network was trained. The architecture is the one proposed by Liu et al.. The dataset for the training was taken from the RRUFF database[23]. The raw spectra that was available was of low resolution(Intensity sampled at $1.9285cm^{-1}$). Each spectra was cropped upto the region of $1500cm^{-1}$, as was done in [25], and starting from $150cm^{-1}$. Thus each vector was of the size 1×701. Minerals with only 1 or 2 spectra were removed, and thus, the original dataset contained 3763 spectrum, of 653 types of minerals. A Leave one out scheme as done in [25], where a sample from each class was selected randomly for testing was used, to make the training and the testing data split. Data Augmentation was carried out as explained earlier. Each spectra was normalised to the range of $[0, 1]$ Figure 3.13 shows the histogram of the number of samples per class, and the number of such classes. The Network was trained, on the High Performance Computing Clusters. The Accuracy of the network was **47.47%**. The trained model was saved, with its learned weights and biases.

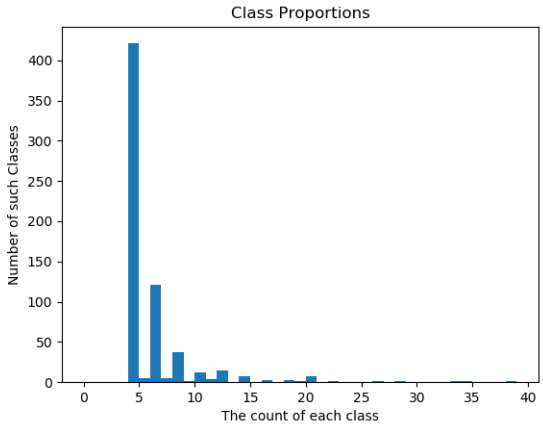


Figure 3.13: Class Proportions

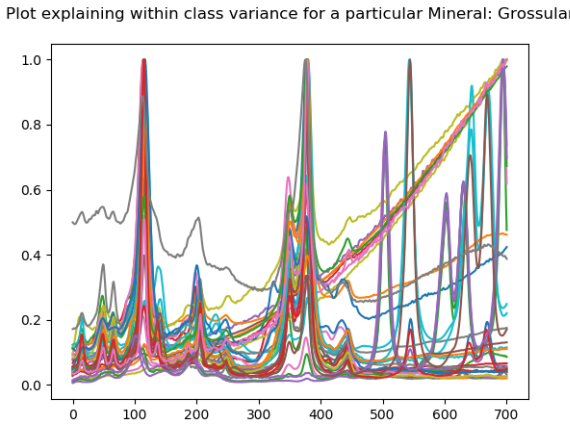
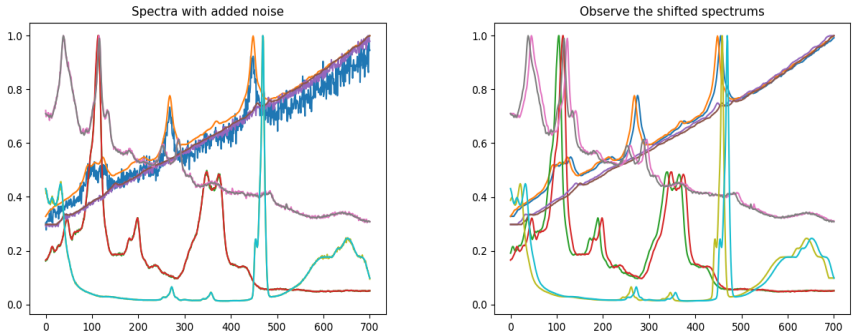
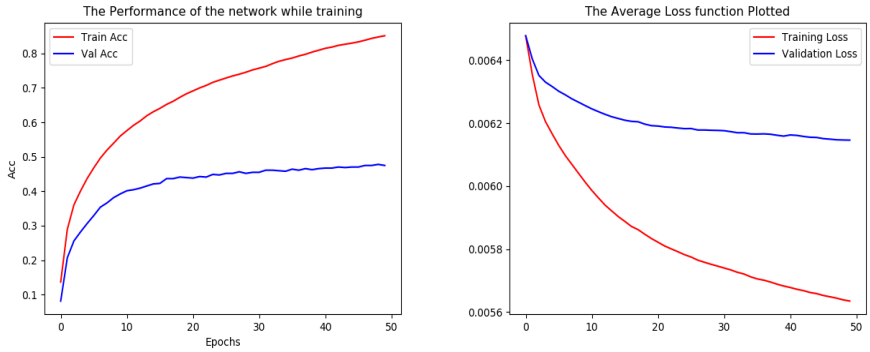


Figure 3.14: Variance within a class with highest number of spectra. This Mineral Name was Grossular.



(a) **(b)**

Figure 3.15: Images of some of the spectra, with their augmented data.



(a) **(b)**

Figure 3.16: Network Performance after training on the RRUFF database

3.4.1 Transfer Learning

The architecture as described earlier, was used for the model. The weights of only the first layer of convolutional filters from the Deep CNN, were transferred to the model. Since the RRUFF database that was used, contained low resolution dataset, while the Diabetes Dataset [18] had intensity sampled at interval of $1cm^{-1}$, the Diabetes Dataset was interpolated and cropped to get the input vector size same as that of the Minerals Dataset, and transfer learning was applied, using the architecture discussed earlier. The Network Performs almost equal to the random guessing, with an accuracy of **56.44%**. The Sensitivity of the model is **70.45%**, while the specificity is **47.22%**.

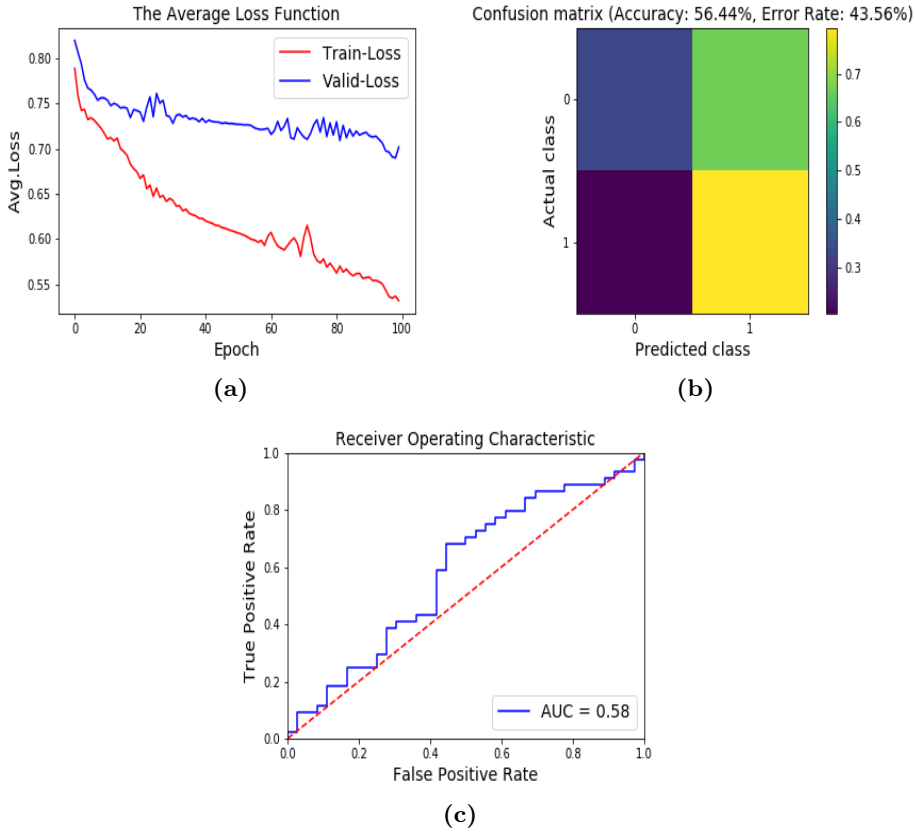


Figure 3.17: Network Performance after transfer Learning

Chapter 4

Discussion and Conclusion

Thus, we investigated the use of Convolutional Neural Networks as an end-to-end learning tool for classification of raw raman spectra. The shallow CNN which was trained on raw Data, performed really well, while the transfer learned model, performed at par with random guessing. KNN performed better when applied to baseline corrected data, than CNN. Although, it is likely that this might have been a result of randomness in the test/training splits in the data, as was shown by the student t-distribution test. KNN performed worse than random guessing on raw data, which was expected, as the baseline is accounting for most of the distribution of the data, and thus a Euclidean Distance, cannot help us determine the different classes.

- The CNN shallow network performed exceptionally well as compared to the KNN model and the transfer learned model, when applied to raw spectral data. Although, that estimate of the models performance is likely to have been slightly biased. This is because we used the same dataset, for hyperparameter optimisation and then evaluated the metrics on the same dataset. Ideally there should have been a 2 fold cross validation to compute the metrics on the models performance on a different subset of data, and the hyperparameters tuned on the inner cross validation loop.
- The student's t-distribution test assumes that the test errors follow a normal distribution. But there is no evidence to suggest the same. The

box plots of the comparison shown in subsection 3.2.6, clearly show that the distribution is not a normal distribution.

- The Deep CNN's low performance on the validation set, and the fact that the network does not show signs of overfitting from the loss functions that were plotted in Figure 3.16b, could also be indicative that the test set is not a representative of the training set, and that they are inherently different. This can be checked for by plotting some of the failed cases of classification by the network. As we can see, that some of these spectra

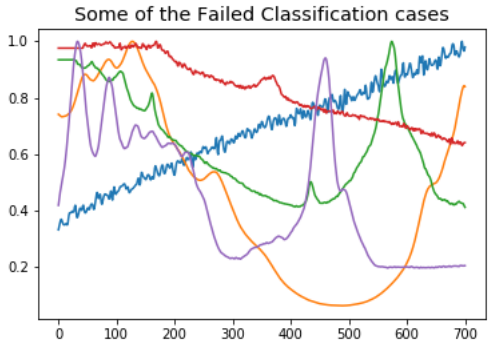


Figure 4.1: Some of the failed Classification Cases

are of very bad quality, and just contains noise and baseline. For example, let us visualise the spectrum in blue for its training counterpart spectra.

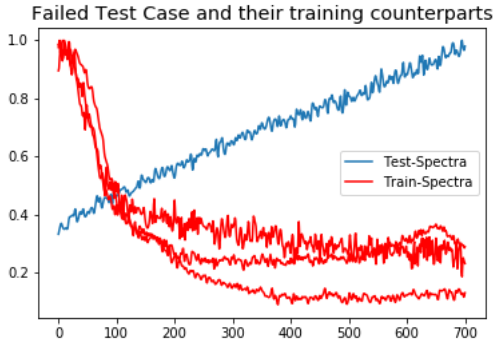


Figure 4.2: Figure showing the test spectrum that was failed in classification, and training spectra from the same class

The same was the case as for the other failed spectrum presented in Figure 4.1, as can be seen in the figure below This is indeed the case. This

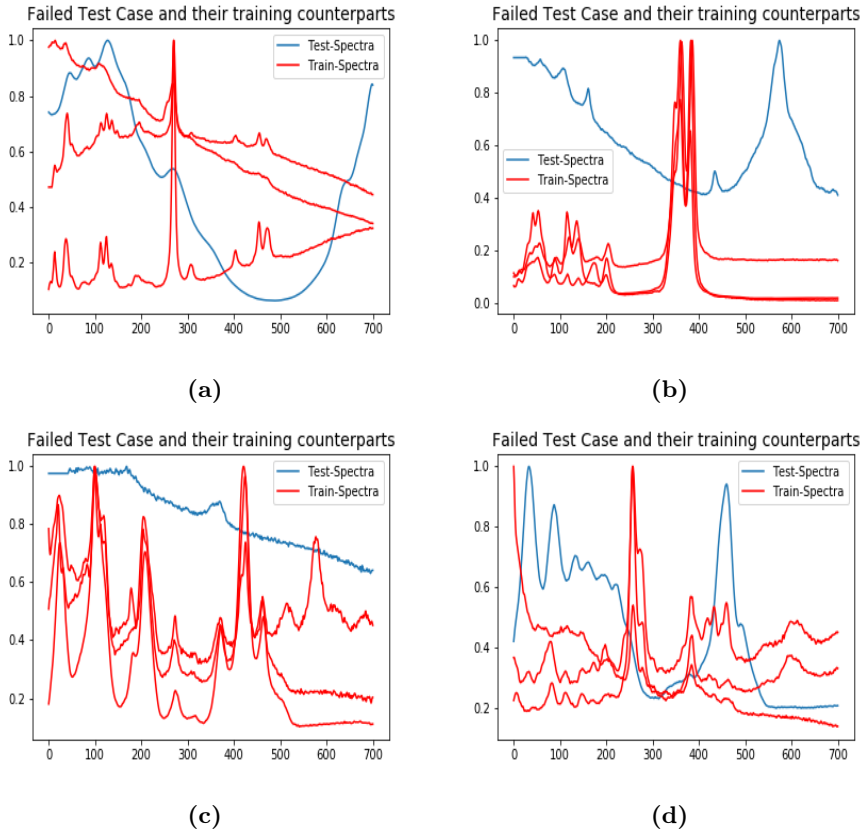


Figure 4.3: Failed Test cases and their training samples

could be the way spectral acquisition works, with RRUFF[23], as the spectrums are acquired by chemists all over the world, and are submitted to RRUFF, which verify them using certain diffraction techniques[23].

- The Figure 3.16b also indicates a slight underfitting phenomenon to the network, as the validation loss has not stabilised yet, and thus could decrease even more.

- For the Transfer Learning, we tried to use the Deep Neural Network's Convolutional Layer, as a feature extractor, directly. In hindsight, this was not the optimal way to carry out the transfer learning. A more efficient way would have been to, transfer the weights of the Deep neural network

and then carry out the training, with the Convolutional layer also being trained, along with the fully connected final layers. It likely that the convolutional kernels are doing something very simple, as peak detection, because the spectra of the minerals have very clear definitive, and distinct peak, which are a fingerprint of the chemicals present in the substance, and this is too trivial for processing a biological spectrum, which has way too many peaks.

- The Distance metric used for all KNN models was the Euclidean Distance. This was another Hyperparameter. Ideally, this should also have been tuned in the inner cross validation loop of the 2-layer cross validation.
- For the Hyperparameter Tuning of the CNN, there were some parameters which should have been also tuned as part of the random grid search algorithm. Maximum epoch's, to be used for training, is one of them. Also the Learning Rate is one of them.

4.1 Future Work

Thus, we saw that convolutional neural network did give us end-to-end Learning for raw spectral data. Acquarelli et al., had proposed that this could be because the CNN is learning filters that provide signal smoothing of the raman spectrum. It will be interesting to investigate, what kind of filters are being learned in the case of the shallow CNN and that of the DEEP CNN. In the previous section, I had speculated that maybe the weights learned in the Deep CNN are doing something trivial as peak detection, and that is not very helpful for processing the biological Raman Spectra, which has way too many peaks, and peak detection does not really help. It would be also interesting to investigate this claim, and in general to investigate the learned features of the network.

At the same time, the weights of the fully connected layer, they can tell us about the important spectral regions, that the CNN is looking at to do the actual classification, as was in the case of Figure 3.1d. These should be plotted for the shallow CNN, and using a forward or backward selection algorithm, on the feature maps produced by the convolutional layer, similar to as proposed by [1], we can finally find out the important spectral regions for Diabetes Detection based on Raman spectroscopy.

Appendix A

Appendix

The code for the all the experiments can be found at the following link under the **DM2_Diagnosis** folder, in the form of Jupyter Notebooks. All other details about the code are explained in the readme file. All the images from all experiments were generated through the code, and are also present there under **Images** folder. `https://lab.compute.dtu.dk/s161227/cnn_raman_classification_git_repo`

Bibliography

[1] Jacopo Acquarelli, Twan van Laarhoven, Jan Gerretzen, Thanh N. Tran, Lutgarde Buydens, and Elena Marchiori. Convolutional neural networks for vibrational spectroscopic data analysis. *Analytica Chimica Acta*, 954, 12 2016. doi: 10.1016/j.aca.2016.12.010.

[2] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space. Technical report. URL `https://scibib.dbvis.de/uploadedFiles/155.pdf`.

[3] Tommy S Alstrøm, Mikkel N Schmidt, Tomas Rindzevicius, Anja Boisen, and Jan Larsen. A PSEUDO-VOIGT COMPONENT MODEL FOR HIGH-RESOLUTION RECOVERY OF CONSTITUENT SPECTRA IN RAMAN SPECTROSCOPY. Technical report. URL `http://www2.imm.dtu.dk/pubdb/views/edoc{_}download.php/6972/pdf/imm6972.pdf`.

[4] Michael B. Fenn, Petros Xanthopoulos, Georgios Pyrgiotakis, Stephen Grobmyer, P Pardalos, and Larry L. Hench. Raman spectroscopy for clinical oncology. *Advances in Optical Technologies*, 2011, 08 2011. doi: 10.1155/2011/213783.

[5] Seong-Joon Baek, Aaron Park, Jin Young Kim, Seung You Na, Yonggwan Won, and Jaebum Choo. Detection of basal cell carcinoma by automatic classification of confocal raman spectra. pages 402–411, 08 2006. doi: 10.1007/11816102_44.

[6] James Bergstra, James Bergstra@umontreal Ca, and Yoshua Bengio@umontreal Ca. Random Search for Hyper-Parameter Optimization Yoshua Bengio. Technical report, 2012. URL `http://scikit-learn.sourceforge.net`.

[7] Zephania Birech, Peter Waweru Mwangi, Fredrick Bukachi, and Keith Makori Mandela. Application of Raman spectroscopy in type 2 diabetes screening in blood using leucine and isoleucine amino-acids as biomarkers and in comparative anti-diabetic drugs efficacy studies. *PLOS ONE*, 12(9):e0185130, sep 2017. ISSN 1932-6203. doi: 10.1371/journal.pone.0185130. URL `http://dx.plos.org/10.1371/journal.pone.0185130`.

[8] Shihyen Chen, Bin Ma, and Kaizhong Zhang. On the similarity metric and the distance metric. *Theoretical Computer Science*, 410(24):2365 – 2376, 2009. ISSN 0304-3975. doi: https://doi.org/10.1016/j.tcs.2009.02.023. URL `http://www.sciencedirect.com/science/article/pii/S0304397509001819`. Formal Languages and Applications: A Collection of Papers in Honor of Sheng Yu.

[9] Thomas M Cover, Peter Hart, et al. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[10] Sishan Cui, Shuo Zhang, and Shuhua Yue. Raman Spectroscopy and Imaging for Cancer Diagnosis, 2018. ISSN 20402309.

[11] Xiaoyu Cui, Zeyin Zhao, Gejun Zhang, Shuo Chen, Yue Zhao, and Jiao Lu. Analysis and classification of kidney stones based on raman spectroscopy. *Biomedical Optics Express*, 9:4175, 09 2018. doi: 10.1364/BOE.9.004175.

[12] Ruchita S. Das and Y.K. Agrawal. Raman spectroscopy: Recent advancements, techniques and applications. *Vibrational Spectroscopy*, 57(2):163–176, nov 2011. ISSN 09242031. doi: 10.1016/j.vibspec.2011.08.003. URL `https://linkinghub.elsevier.com/retrieve/pii/S0924203111001111`.

[13] Paul H C Eilers and Hans F M Boelens. Baseline Correction with Asymmetric Least Squares Smoothing. Technical report, 2005. URL `https://zanran{_}storage.s3.amazonaws.com/www.science.uva.nl/ContentPages/443199618.pdf`.

[14] Rekha Gautam, Sandeep Vanga, Freek Ariese, and Siva Umapathy. Review of multidimensional data processing approaches for Raman and infrared spectroscopy. *EPJ Techniques and Instrumentation*, 2(1):8, 2015. ISSN 2195-7045. doi: 10.1140/epjti/s40485-015-0018-6. URL `https://doi.org/10.1140/epjti/s40485-015-0018-6`.

[15] Adas Gelzinis, Antanas Verikas, Evaldas Vaiciukynas, Marija Bacauskiene, Jonas Minelga, Magnus Hallander, Virgilijus Uloza, and Evaldas Padervinskis. Exploring sustained phonation recorded with acoustic and contact microphones to screen for laryngeal disorders. pages 125–132, 12 2014. doi: 10.1109/CICARE.2014.7007844.

[16] Patrick Glauner. *Deep Convolutional Neural Networks for Smile Recognition*. PhD thesis, 08 2015.

[17] J. L. González-Solís, J. R. Villafan-Bernal, B. E. Martínez-Zérega, and S. Sánchez-Enríquez. Type 2 diabetes detection based on serum sample Raman spectroscopy. *Lasers in Medical Science*, 33(8):1791–1797, nov 2018. ISSN 0268-8921. doi: 10.1007/s10103-018-2543-4. URL `http://link.springer.com/10.1007/s10103-018-2543-4`.

[18] Edgar Guevara, Juan Carlos Torres-Galván, Miguel G. Ramírez-Elías, Claudia Luevano-Contreras, and Francisco Javier González. Use of Raman spectroscopy to screen diabetes mellitus with machine learning tools. *Biomedical Optics Express*, 9(10):4998, oct 2018. ISSN 2156-7085. doi: 10.1364/BOE.9.004998. URL `https://www.osapublishing.org/abstract.cfm?URI=boe-9-10-4998`.

[19] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL `http://arxiv.org/abs/1502.03167`.

[21] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed <today>].

[22] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. dec 2014. URL `http://arxiv.org/abs/1412.6980`.

[23] Barbara Lafuente, R T Downs, H Yang, and N Stone. The power of databases: The RRUFF project. Technical report. URL `http://rruff.info/about/downloads/HMC1-30.pdf`.

[24] Peter Lasch. Spectral pre-processing for biomedical vibrational spectroscopy and microspectroscopic imaging. *Chemometrics and Intelligent Laboratory Systems*, 117:100–114, 08 2012. doi: 10.1016/j.chemolab.2012.03.011.

[25] Jinchao Liu, Margarita Osadchy, Lorna Ashton, Michael Foster, Christopher J Solomon, and Stuart J Gibson. Deep Convolutional Neural Networks for Raman Spectrum Recognition : A Unified Solution. Technical report. URL `https://pdfs.semanticscholar.org/9a37/ad485a197e2727be3c94f35f0d0f3d6b44fe.pdf`.

[26] Signe M. Lundsgaard-Nielsen, Anders Pors, Stefan O. Banke, Jan E. Henriksen, Dietrich K. Hepp, and Anders Weber. Critical-depth Raman spectroscopy enables home-use non-invasive glucose monitoring. *PLOS ONE*,

13(5):e0197134, may 2018. ISSN 1932-6203. doi: 10.1371/journal.pone.0197134. URL http://dx.plos.org/10.1371/journal.pone.0197134.

[27] Richard L McCreery. *Raman spectroscopy for chemical analysis*, volume 225. John Wiley & Sons, 2005.

[28] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

[31] Isaac Pence and Anita Mahadevan-Jansen. Clinical instrumentation and applications of Raman spectroscopy. *Chemical Society reviews*, 45(7):1958–79, apr 2016. ISSN 1460-4744. doi: 10.1039/c5cs00581g. URL http://www.ncbi.nlm.nih.gov/pubmed/26999370http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4854574.

[32] Miguel Ghebré Ramírez-Elías and Francisco Javier González. Raman Spectroscopy for In Vivo Medical Diagnosis. In *Raman Spectroscopy*. InTech, apr 2018. doi: 10.5772/intechopen.72933. URL http://www.intechopen.com/books/raman-spectroscopy/raman-spectroscopy-for-in-vivo-medical-diagnosis.

[33] Martina Sattlecker, Conrad Bessant, Jennifer Smith, and Nick Stone. Investigation of support vector machines and Raman spectroscopy for lymph node diagnostics. *The Analyst*, 135(5):895, apr 2010. ISSN 0003-2654. doi: 10.1039/b920229c. URL http://xlink.rsc.org/?DOI=b920229c.

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[35] Seng Khoon Teh, Wei Zheng, David P. Lau, and Zhiwei Huang. Spectroscopic diagnosis of laryngeal carcinoma using near-infrared Raman spectroscopy and random recursive partitioning ensemble techniques. *The Analyst*, 134(6):1232, jun 2009. ISSN 0003-2654. doi: 10.1039/b811008e. URL http://xlink.rsc.org/?DOI=b811008e.

[36] S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30, March 2011. ISSN 1521-9615. doi: 10.1109/MCSE.2011.37.

[37] Effendi Widjaja, Wei Zheng, and Zhiwei Huang. Classification of colonic tissues using near-infrared Raman spectroscopy and support vector machines. *International Journal of Oncology*, 32(3):653–662, mar 2008. ISSN 1019-6439. doi: 10.3892/ijo.32.3.653. URL `http://www.spandidos-publications.com/10.3892/ijo.32.3.653`.

[38] Khursiah Zainal Mokhtar and Junita Mohamad-Saleh. An oil fraction neural sensor developed using electrical capacitance tomography sensor data. *Sensors (Basel, Switzerland)*, 13:11385–406, 09 2013. doi: 10.3390/s130911385.

[39] Xiaoliang Zhang, Jiali Li, Yugang Liu, Zutao Zhang, Zhuojun Wang, Dianyuan Luo, Xiang Zhou, Miankuan Zhu, Waleed Mohammed, Guangdi Hu, and Chunbai Wang. Design of a fatigue detection system for high-speed trains based on driver vigilance using a wireless wearable eeg. *Sensors*, 17:486, 03 2017. doi: 10.3390/s17030486.

[40] Jianhua Zhao, Harvey Lui, David I Mclean, and Haishan Zeng. Automated Autofluorescence Background Subtraction Algorithm for Biomedical Raman Spectroscopy. Technical report. URL `https://journals.sagepub.com/doi/pdf/10.1366/000370207782597003`.