Aalto University
School of Business

Improving Material Classification Using Data Augmentation

Case Study

Master's Thesis
Biranjan Raut
Aalto University School of Business
Information and Service Management
Fall 2018

Aalto University
School of Business
Master of Science in Economics and Business Administration

ABSTRACT OF MASTER'S THESIS

Author:	Biranjan Raut		
Title:	Improving Material Classification Using Data Augmentation Case Study		
Date:	June 18, 2018	**Pages:**	vi + 58
Major:	Business Analytics	**Code:**	T-110
Advisor(s):	Pekka Malo, Aalto University Juuso Liesiö, Aalto University Pekka Röyttä, Spectral Engines Oy		

One of the major barrier for the application of machine learning in both academic and commercial field has been the limited availability of data. Over the past decades majority of the problem associated with the application of machine learning has been more or less solved with high speed computing processor and efficient algorithm. However, data is still illusive as substantial time and money is spent in collection and preparation of data. This thesis tries to solve this problem through artificially generating data; process also known as data augmentation.

In the thesis various augmentation technique namely data-warping,synthetic minority over sampling and generative adversarial nets are explored as a possible augmentation solution. Models with augmentation and without augmentation are compared and evaluated using harmonic mean of precision and recall.This evaluation metric enables model evaluation based not just on majority class but also based on less frequent minority class. Deep learning model are designed to classify organic compound using the spectral data acquired through Near infrared sensor.

From evaluation of the deep learning model we could observe noticeable improvement in the model with augmentation. Almost all the model performed better over the model without any augmentation. Augmented model did well not only with overall accuracy but also with precision and recall. This thesis tries to tackle contemporary problem using both old and new techniques from very domain specific simple algorithm to the state of the art generative models.

Keywords: machine learning,deep learning, spectroscopy, data augmentations, SMOTE, GANs, data-warping,neural network

Language: English

Contents

Abstract **ii**

1 Introduction **1**
1.1 Motivation 1
1.2 Research Problem 3
1.3 Aim of the Study 3
1.4 Structure of the Thesis 4

2 Literature Review **5**
2.1 Machine Learning 5
2.2 Deep Learning 6
2.2.1 Why data augmentation? 7
2.3 Data Augmentation in Deep Learning 9
2.3.1 Data Warping 9
2.3.2 SMOTE 11
2.3.3 Generative Adversarial Nets 13
2.4 Model Evaluation 16

3 Methods **19**
3.1 Cross Industry Standard of Data Mining 19
3.1.1 Business understanding 20
3.1.2 Data understanding 21
3.1.3 Data preparation 21
3.1.4 Modeling 21
3.1.5 Evaluation 22
3.1.6 Deployment 22

4 Implementation **23**
4.1 Business understanding 23
4.2 Data understanding 25
4.2.1 Exploratory Analysis 26
4.3 Data preparation 28
4.3.1 Data augmentation using data warping 28
4.3.2 Data augmentation using SMOTE 29
4.3.3 Data augmentation using GANs 31
4.4 Modeling 32

5 Evaluation **35**
5.1 Base model without augmentation 35
5.2 Augmentation using data warping 36
5.3 Augmentation using SMOTE 36

5.4 Augmentation using GANs 37
5.5 Model Comparison 38

6 Discussion **40**
6.1 Limitations 40
6.2 Future work 41

7 Conclusions **43**
7.1 Key Findings 43
7.2 Contributions 43
7.2.1 Theoretical Contribution 44
7.2.2 Practical Contribution 44

A Augmentation using data warping **45**

B Augmentation using SMOTE **46**

C Neural network **47**

D Augmentation using GANs **48**

E Data generation through GANs **52**

F Deep learning model without augmentation **53**

G Deep learning model training **54**

H Deep learning model training **55**

I Model misclassification **56**

Bibliography **56**

List of Tables

2.1 Confusion matrix for binary classification 16

5.1 Evaluation metrics of first model without augmentation 35
5.2 Evaluation metrics of second model with augmentation 36
5.3 Evaluation metrics of third model with augmentation 37
5.4 Evaluation metrics of fourth model with augmentation 37
5.5 Model comparison based on F1-score 38

List of Figures

2.1 Building block of deep learning process 8
2.2 Augmentation via data warping on hand written digits 10
2.3 Synthetic data generation using SMOTE 12
2.4 GANs architecture 14

3.1 CRISP-DM process 20

4.1 Spectral data 26
4.2 Concentration level of compounds 27
4.3 Augmentation using data warping 29
4.4 Augmentation using SMOTE 30
4.5 The implementation of GAN in keras 31
4.6 Augmentation using GANs 32
4.7 Model Architecture 33

C.1 Detail overview of the network 47

E.1 Data generation process using GANs 52

F.1 Deep learning model without augmentation training process . . 53

G.1 Training loss and accuracy for model with augmentation usign data-warping 54

H.1 Training loss and accuracy for model with augmentation usign data-warping 55

I.1 Density plot of model missclassification at various concentration level 56

Chapter 1

Introduction

1.1 Motivation

In past decades deep learning has garnered plenty of attention, primarily due to its success at discovering complex structures in high-dimensional data. The success of deep learning has lead to the application of deep learning to many domains of science, business, and government. Compared to conventional machine learning methods, deep learning methods require little feature engineering, and it performs better with raw data in its natural form (LeCun et al., 2015). Deep-learning methods are a set of representation learning methods that allows a machine to be fed with raw data and to discover the representations needed for detection or classification (LeCun et al., 2015).

Improvement in the hardware, software, and availability of data has made the application of deep learning possible in various tasks. Although advancement in algorithms along with hardware has resolved the majority of bottlenecks in applications of deep learning and made it accessible to everyone, deep learning still requires a large number of annotated training samples for better performance. One way to increase the annotated training samples is by collecting more unstructured data and building a manual or automated system of data annotations. However, this method can be expensive, error-prone and in some domains simply impossible due to a limited source of data. Another way to overcome this problem is by artificially generating more training samples from existing data. This process of creating artificial data from existing data is also known as Data Augmentation. Data augmentation has potential to increase the accuracy of the existing deep learning models and at the same time reduce the overall cost associated with data acquisition and manipulation. It also can create an entirely novel application of deep learning in a domain where it was previously impossible to apply deep learning due to a limited number of data. Therefore, this thesis will explore existing data augmentation techniques to improve deep learning model by artificially generating the training samples.

The primary motivation for taking on this contemporary problem in the field of deep learning comes from a challenge faced by a case company. Case company produces portable material sensors based on Near Infrared (NIR) spectroscopy. NIR spectrometers are widely used in various industries to measure material content, such as moisture, fat, protein, hydrocarbons, textiles, polymers and pharmaceutical ingredients. Portable material sensors designed by the company can capture the unique properties of chemical components inside the organic material. These unique properties act as optical fingerprints of chemical components. The optical fingerprint obtained from spectrometer can be used to identify various materials. To correctly identify the material, spectrometer relies on a deep learning model which has been trained on thousands of annotated samples. The annotated samples are obtained in a laboratory setting which is both time and labor-intensive process as deep learning model requires thousands of samples to produce a reliable and consistent model. Similarly, certain materials such as pharmaceutical ingredients are difficult to obtain due to regulatory reasons. For such materials, a wide range of samples cannot be collected which adversely affects the performance of deep learning model. Availability of annotated training samples is a significant limiting factor in application of deep learning for a case company. Data augmentation has potential to streamline the process of data acquisition. In addition, this could lead to new business opportunities which would not have been possible due to limited availability of annotated samples. Therefore, by exploring the effectiveness of available data augmentation techniques for improving deep learning model, this thesis not only tries to solve the problem faced by the case company but also provides a platform for further research in the application of data augmentation in various other domains.

1.2 Research Problem

This thesis will mainly try to address the specific case where the company cannot collect enough samples due to regulatory reasons. Thus, Data augmentation will be explored as a possible solution to come up with a robust deep learning model. The primary research question will be:

How to apply existing data augmentation methods to increase the classification accuracy of a deep learning model that classifies the organic materials using limited training data obtained from the Near-Infrared spectrometer?

1.3 Aim of the Study

Although data augmentation techniques are not new in the field of deep learning, some of the techniques are domain-specific and others still at experimental phase with varying degree of success. For example, the technique of data-warping is not grounded in a sound theoretical background and thus produces inconsistent result across different types of applications. Similarly, GANs is a relatively new technique, and it is difficult to translate its theoretical framework into a real application.

Consequently, this thesis has a two-fold objective. The primary objective will be to use data augmentation to improve the accuracy of existing object classification model. To achieve the primary goal, various contemporary data augmentation techniques will be explored. The secondary objective of the thesis is to bridge the gap between existing data augmentation literature and its applications. In this way, the thesis can also provide an avenue for further research, especially in exploration and implementation of data augmentation to solve the problem posed by the limited availability of training data.

1.4 Structure of the Thesis

The thesis starts off by laying out the motivation for research. Then it provides an overview of relevant literature review in Chapter 2. In Chapter 3, a brief overview of the process of conducting a machine learning project is described. In Chapter 4, the process described in Chapter 3 is implemented. In Chapter 5, models obtained from data mining process is evaluated using the framework discussed in literature review. Chapters 6 and 7 are concluding chapters where the insight gained from the constructing and evaluating deep learning models using various data augmentation techniques is discussed and concluded by laying out the limitations, future work, and contributions of this paper.

Chapter 2

Literature Review

In this chapter, the theoretical concept related to the subject of the study will be reviewed. We find it necessary to provide a general overview of machine learning, deep learning, and data augmentation as these concepts are the foundation stone of the thesis.

2.1 Machine Learning

Often machine learning and deep learning are used together to mean the same thing. However, there is a distinction between machine learning and deep learning. Machine learning is a broader set of automated learning tools which include deep learning as sub-field (Chollet, 2017). Machine learning powers many aspects of modern life from simple web search to complex language translation. In a simple language, machine learning can be understood as a process where humans input data, as well as expected output from the data, then the machine produces the rules that match given input to its expected output (Chollet, 2017). Expected output is optional as some machine learning problem do not have any expected output and the job of machine is to transform the input into a meaningful output. At the heart of the machine learning lies input data and a way to measure whether the rule learned by the machine is reliable or not (Chollet, 2017). In this sense, the central problem of machine learning is to figure out the best possible transformation that provides the most accurate result. Sometimes this transformation is also referred to as representation.

Machine learning task can be broadly classified into Supervised and Unsupervised learning. Supervised learning is the most common form of machine learning tasks. Like the majority of machine learning problems, the problem this thesis is trying to tackle also falls under supervised learning problem. Supervised learning problem is about learning to map input data to known targets also known as labels or annotations, given a set of examples (Chollet, 2017). Most common application of deep learning such as image classification, speech recognition, and language translation falls under supervised learning. In this thesis, the problem is to map the spectral data in the form of arrays to its corresponding label of an organic compound. Unsupervised machine learning is the branch of machine learning that consist of finding the best representation or transformation of the input data without the help of any explicit expected output. Unsupervised machine learning are used for the purpose of data visualization, data compression, data denoising, or to better understand the correlations present in the data at hand (Chollet et al., 2015).

To summarize, machine learning is an iterative process of finding the appropriate representation of input. It uses a feedback loop from the metric that determines the quality of representation and keeps iterating until it reaches the point where no further improvement of the metric is possible.

2.2 Deep Learning

Deep learning is a subfield of machine learning. "It is a representation learning method with multiple levels of representation, obtained by various non-linear models that each transforms the representation at one level into representation at a higher, more abstract level" – LeCun et al. (2015). The deep in deep learning comes from the sequence of layers/representations used in a learning process. Depending upon the task, the number of layers in representation varies. A modern state-of-the-art deep learning system can have hundreds or thousands of successive layers of representation.

Deep learning learns the layered representation through models called neural network; structured in layers stacked on top of each other (Chollet, 2017).

The neural networks transform the data from one representation to another. The way the networks transform the data is stored in layer's weight. In essence deep learning is finding the appropriate weights of all layers in the network so that the final output most accurately estimates the actual output. To find most suitable weights, we need to have metrics that measure how far the present output of deep learning model is from the actual output. The process of optimizing the weights in relation to loss-measure is done through optimizer which uses the backpropagation algorithm. Due to the limited scope and time we won't go deep into backprpogation but simply put it is a method to update weights in neural network. During the first iteration, weights of the network are assigned with random values which means the model will have a high loss value. With every iteration, weights are adjusted to minimize the loss value. These iterations are also known as training loop or training feedback. With sufficient iteration, the model will be able to determine the weights that yield minimal loss value, which means the model output is as close as they can be to actual output. The figure 2.1 summarizes the basic building block of a deep learning process.

2.2.1 Why data augmentation?

Deep learning requires a relatively large amount of data and significant computing power to analyze the data. Furthermore, it requires an efficient algorithm that can calculate hundreds of weights. These requirements posed by deep learning were the significant barriers to the application of deep learning in early 90's. However, nowadays there has been a substantial improvement concerning Central Processing Unit (CPU) and Graphical Processing Unit (GPU). Similarly, digitalization and increasing reach of the internet has generated a significant amount of data. In addition, open source projects contributed by both corporations and researchers in the field of deep learning has streamlined the implementation and deployment of deep learning. As a result of all this, deep

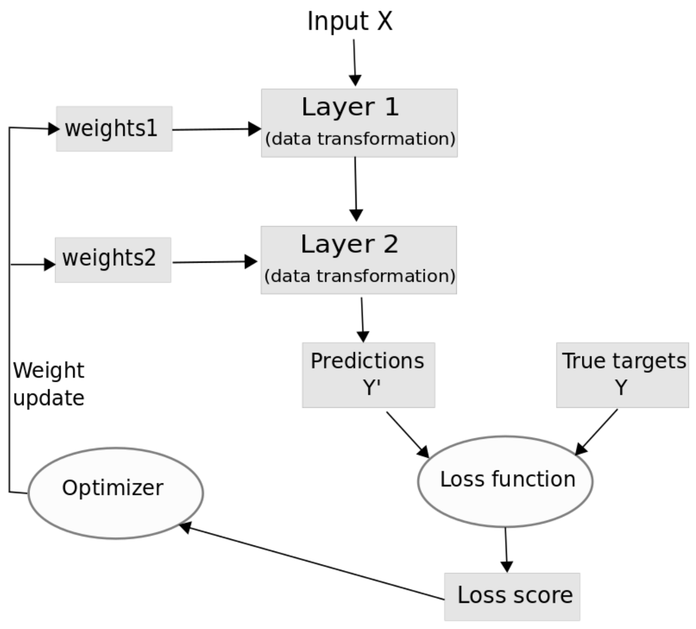


Figure 2.1: Building block of deep learning process

learning has been applied to solve various complex problems like image and speech recognition, analyzing particle accelerator data, DNA mutations and autonomous driving with successful results (LeCun et al., 2015).

Empirical studies have shown that data representation obtained from deep learning often yield better machine learning results in terms of improved classification modeling (Larochelle et al., 2009). Similarly, data generated by a deep learning model is also deemed to be of a better quality than the data generated by generative probabilistic models (Najafabadi et al., 2015). Compared to conventional machine learning techniques, deep learning technique offers better performance and makes problem-solving simpler by requiring less data engineering and domain expertise. Deep learning is able to capture relevant features of complex problems with the help of multiple representations. Deep learning is simple, scalable, versatile and reusable. It is simple in a sense that it does not require feature engineering which means it removes complex machine learning pipelines with a simple end-to-end trainable models. Thanks to various open source projects, deep learning computation can be parallelized on GPU's or in a distributed system. Deep learning models are trained by iterating over a small batch of data. This batch processing adds the flexibility to train on any size of data. Furthermore, deep learning models are modular allowing the layers from model to be reused to solve similar problems. These pragmatic properties make deep learning functional and useful for researchers and companies alike.

2.3 Data Augmentation in Deep Learning

Data augmentation has been applied in deep learning with varying degrees of success. The most prominent application of data augmentation has been in image classification, and it has proven to be a successful technique to improve the accuracy of a model (Krizhevsky et al., 2012). Data augmentation can reduce over-fitting as well as make model more robust by introducing artificially generated variables that make the training sample diverse. For example, a recent study on the application of data augmentation in facial recognition showed a significant improvement in the model as a result of data augmentation (Kortylewski et al., 2018). This thesis will primarily go through three data augmentation methods, namely data warping, synthetic minority over-sampling technique, and generative adversarial nets.

2.3.1 Data Warping

The most conventional application of data augmentation has been in image classification, and it has proven to be reasonably successful to improve the accuracy of a model. One of the widely used and accepted practices for augmenting image data is to perform geometric and color transformation, such as reflecting an image, cropping and translating the image. Through the application of the various transformation to one image, multiple images can be generated with different perspective and shades. These newly generated images can be used as new training samples. This approach of data augmentation is known as data warping. The term data warping was first used in the context of producing random variations in hand-printed English text data so that new characters are generated mimicking the stylistic variation within and between the writers (Yaeger et al., 1997).

From its initial application in late 1990 this technique was further extended to improve the performance of a neural network and achieve record performance of the time on MNIST handwritten digit database using a neural network in 2003 (Wong et al., 2016). Training data was created by applying simple distortion such as translations, rotation, and skewing. These various transformations helped to create naturally occurring variation as different individuals write letters slightly differently. Augmenting data through the techniques of data warping depends on the problem at hand. Furthermore, these techniques are heuristic in nature which is not guaranteed to be optimal in all situations.

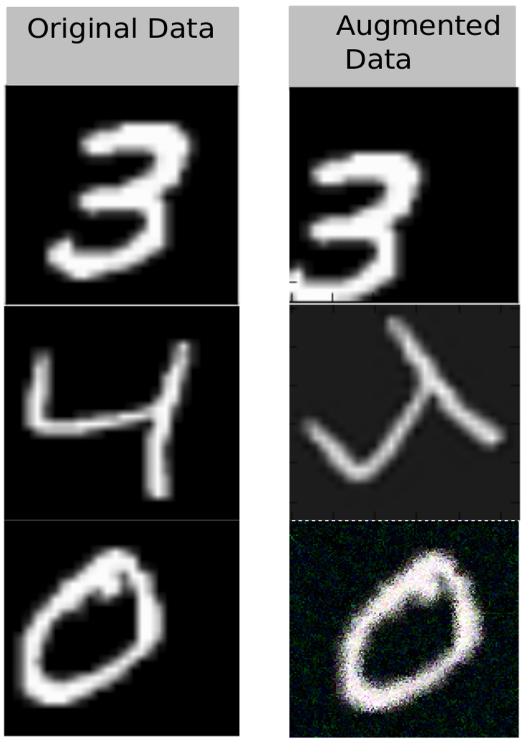


Figure 2.2: Augmentation via data warping on hand written digits

The figure 2.2 shows the data augmentation applied to image data using various data warping techniques. The top image is warped using a simple random shift of pixels, whereas the bottom image of zero is warped by adding random noise to pixels, and the middle image of four is warped using just a rotation of pixels. This simple technique can make the model more robust. For example by artificially generating several many variation of hand written digit, model can capture naturally occurring variation in hand-writings. This analogy can be extended to the spectral data also, as they can shift slightly up or down due to the variation in concentration or due environmental factors compared to the sample prepared in the lab. Thus, by training model with augmented data model is likely to generalize well in previously unseen measurement data.

2.3.2 SMOTE

Synthetic Minority Over-Sampling Technique (SMOTE) is a data augmentation technique which is inspired by data warping, particularly its ability to reduce the class imbalance in hand-written digit problems (Chawla et al., 2002). It is a data augmentation approach in which minority label is over-sampled by creating new synthetic data. This method has been used mainly to address the problem of class imbalance, where real-world data-sets often contain a small percentage of target class examples. The advantage of SMOTE compared to data warping is that synthetic samples are generated in feature space, and this property of SMOTE algorithm makes it application-independent (Chawla et al., 2002).

New synthetic samples are obtained from the minority class using the information available in the data. Synthetic samples are generated by taking the difference between feature vector under consideration and its nearest neighbor, then multiplying this difference by a random number between 0 and 1 and finally adding it to the feature vector under consideration (Chawla et al., 2002). For each sample x, the other samples from the same minority class with the smallest Euclidean distance from the x are identified. These other samples are also called as nearest neighbor. One of the nearest neighbors is randomly picked x^R. The new synthetic sample S is defined in the equation below.

$$S = x + u(x^R - x), where\ u \sim U(0, 1) \tag{2.1}$$

(Chawla et al., 2002)

Figure 2.3 succinctly summarizes the synthetic data generation process using SMOTE.

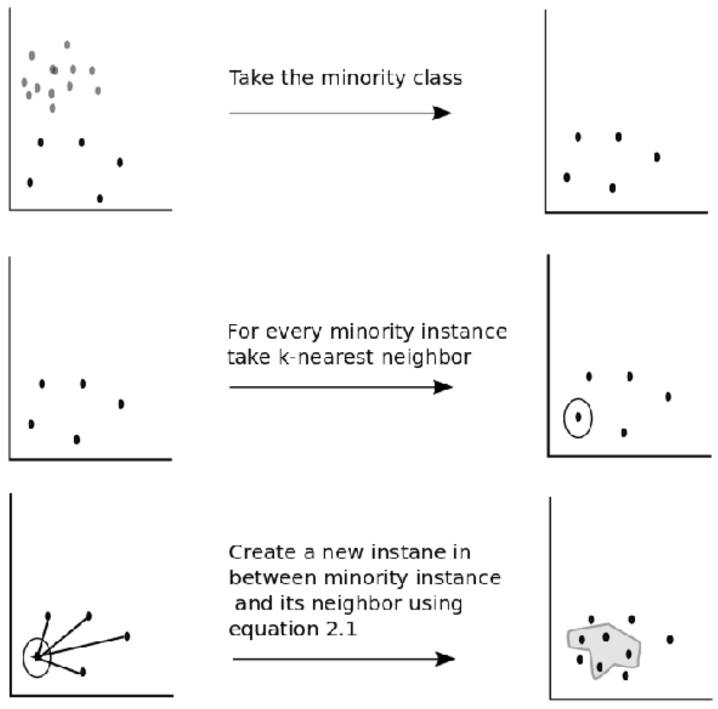
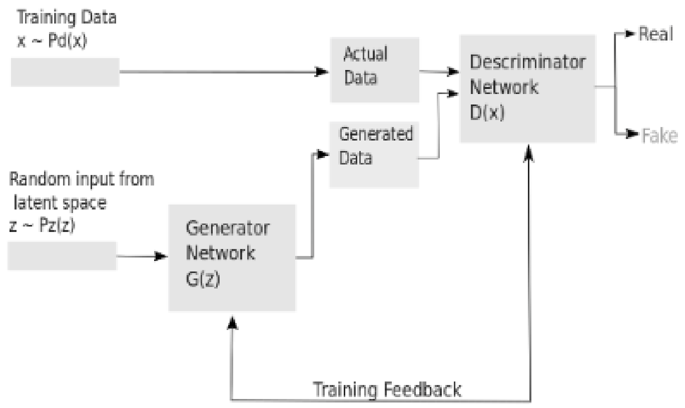


Figure 2.3: Synthetic data generation using SMOTE

One alternative to SMOTE is to over-sample the minority class with replacement. Some previous study has shown that while this practice can increase the number of samples in training set, it does not significantly increase the class recognition of minority class (Chawla et al., 2002). In the experiment conducted by Chawla et al. (2002); when the minority class was over-sampled by increasing amounts, the decision tree tried to identify similar but more specific regions in the feature space of minority class. In other words, the replication of same data causes the model to focus on those repeated samples and also to over-fit the data. Over-sampling the minority class with replacement can produce an undesired result of shrinking the decision boundary for minority class. Ideally, we would want to increase the decision boundary of minority class so that it generalizes well for unseen data. In the same paper, Chawla et al. (2002) demonstrated that SMOTE approach could improve the accuracy of the classifier for a minority class by applying it on various imbalanced data-sets.

2.3.3 Generative Adversarial Nets

One of the promising techniques of data augmentation in the field of deep learning is Generative Adversarial Nets (GANs). It is a powerful technique to generate data. GANs uses a min-max strategy where one neural net successively generates counterfeit samples from the original data distribution in order to fool the other net, and the other net is then trained to better distinguish the counterfeits (Goodfellow et al., 2014). There are two main elements of a GANs: Generator and Discriminator. The objective of the generator is to take a random input and generate a sample data, whereas the objective of the discriminator is to take input from the original data or data generated from generator and to predict whether the input is real or generated. Discriminator aims to maximize the probability of assigning the correct label to both original sample and the counterfeit, whereas generator simultaneously aims to minimize the difference between original and counterfeit samples (Goodfellow et al., 2014). In this sense, the discriminator and generator play a two-player mini-max game. The generator obtained from this process can be used to generate synthetic samples of data which closely represents the actual data. The figure below illustrates the GANs architecture.



Symbol	Meaning
$Pd(x)$	Distribution
X	Sample from $Pd(x)$
$P(z)$	The generator's distribution
Z	Sample form $P(z)$
$G(z)$	Generator Network
$D(x)$	Discriminator Network

Figure 2.4: GANs architecture

Here, generator $G(z)$ takes an input from $P(z)$. The generated data is supplied to discriminator network along with the actual data. The discriminator takes input x from $Pd(x)$ where $Pd(x)$ is the real distribution, $D(x)$ then solves a binary classification problem. Two models compete against each other which ideally leads to the generator being able to produce fairly real looking samples from $P(z)$. This process can be mathematically represented as:

$$min_G max_D L(D,G) = E_{x \sim Pdata(x)}[logD(x)] + E_{z \sim P_z(z)}[log(1 - D(G(z)))] \quad (2.2)$$

(Goodfellow et al., 2014)

In equation 2.2 first term is entropy that the data from real distribution P_d passes through the discriminator. We want to maximize discriminator given real data $P_d(x)$ by maximizing $E_{x_d(x)}[logD(x)]$. Meanwhile given fake data $P_{(z)}$ the discriminator is expected to output a probability of $D(G(z))$ close to zero by maximizing second part of equation $E_{z\ P_z(z)}[log(1-D(G(z)))$. Whereas, the generator is trained to increase the chances of $D(x)$ producing a high probability for a fake example, thus to minimize $E_{z\ P_z(z)}[log(1-D(G(z)))$.

GANs has been extensively used in machine learning applications such as computer vision and image recognition. GANs have been useful even with relatively small sets of data as one can use a transfer learning technique. GANs have been used for example to train a self-driving car to drive in the night or rain using only data collected on a sunny day (Gurumurthy et al., 2017). Recently there have been many studies in the application of GANs in medical diagnosis mainly to augment the medical imaging data. Medical industry also faces the problem of limited data, as data is heavily regulated and costly to acquire. In one recent study conducted by Frid-Adar et al. (2018), authors successfully used GANs to augment the limited medical image data to improve a deep learning model. Using GANs as augmentation method authors were able to gain 7 percent improvement in accuracy over the traditional augmentation methods like data warping.

GANs offer a unique proposition in data augmentation. Unlike data warping, instead of figuring out augmentation that is label invariant we let the deep learning model to find the label invariant augmentation. Thus, the expectation from the GANs model is that the data is generated from the actual data distribution so that the newly generated data has the core characteristics of real data.

2.4 Model Evaluation

Since the thesis tries to evaluate the performance of various deep learning models, metrics for model assessments needs to be clearly laid out. Model evaluation depends on the type of problem model is trying to solve. Machine learning divides classification problem into binary, multi-class, multi-labeled and hierarchical tasks (Sokolova and Lapalme, 2009). The classification problem tackled in this thesis falls under the multi-class classification problem. Formally multi-class problem is defined as a classification problem where the input is to be classified into one and only one, of 'l' non-overlapping classes (Sokolova and Lapalme, 2009).

Empirical evaluation remains the most used approach for the model assessment (Sokolova and Lapalme, 2009). The empirical comparison is made through applying algorithms from the model on data-sets that are not used for model training and then evaluating the performance of the classifiers that the algorithms have produced. Often accuracy remains the most used evaluation measure for evaluating models (Sokolova and Lapalme, 2009). However, accuracy is not the only measure for performance evaluation. The performance of a classification algorithm from given model can also be evaluated by dividing the accuracy into smaller components. For instance, computing the number of correctly recognized class examples (true positives), the number of correctly recognized examples that do not belong to the class (true negatives), and examples that either were incorrectly assigned to the class (false positives) or that were not recognized as class examples (false negatives) (Sokolova and Lapalme, 2009). These four values constitute a confusion matrix shown in the table 2.1.

Table 2.1: Confusion matrix for binary classification

Data class	**Classified as pos**	**Classified as neg**
pos	true positive (tp)	false negative (fn)
neg	false positive (fp)	true negative (tn)

Accuracy is just a proportion of the total number of prediction that was correct. Precision and recall provided more insight than just the overall accuracy. To use the analogy of cancer diagnosis problem: precision tells what proportion of patients diagnosed by the model as having cancer had cancer whereas recall tells what proportion of patient that actually had cancer was diagnosed by the model as having cancer. What precision and recall do is they break the accuracy into small metrics like true positive, false positive and false negative. Precision is defined as the number of true positives over the number of true positives plus the number of false positives. Recall is defined as the number of true positives over the number of true positives plus the number of false negatives (Pedregosa et al., 2011). Furthermore, often precision and recall are expressed in one number which is its weighted average known as the F1 score. In other words F1 score is just the harmonic mean of precision and recall.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.5}$$

$$F1 = \frac{2(recall \times precision)}{recall + precision} \tag{2.6}$$

The above metrics are expressly designed for the binary classification task, where the input is to be classified into one, and only one, of two non-overlapping classes (Sokolova and Lapalme, 2009). However, these binary metrics can be easily extended to the multi-class problem. There is a number of ways to average binary metric calculations across the set of classes. One way to averaging binary metric is macro averaging. Macro averaging calculates the mean of the binary metrics, giving equal weight to each class. In problems where infrequent classes are important, macro-averaging can be a means of highlighting their performance (Pedregosa et al., 2011).

$$Precision_M = \frac{\sum_i^l \frac{tp_i}{tp_i + fp_i}}{l} \tag{2.7}$$

$$Recall_M = \frac{\sum_i^l \frac{tp_i}{tp_i + fn_i}}{l} \tag{2.8}$$

$$Fscore_M = \frac{2(Precision_M \times Recall_M)}{Precision_M + Recall_M} \tag{2.9}$$

To summarize, there are various metrics for evaluation of classification models. For a multi-class classification problem, model metrics are merely the extension from binary classification metrics. It is often the case that no single metrics can comprehensively measure and evaluate the quality of the model. Therefore depending on the application of model, relevant application metrics needs to be used. In this thesis, macro-averaging is used to evaluate the models, as this approach will emphasize the quality of the model based not only on the accuracy of majority class but also based on infrequent minority classes.

Chapter 3

Methods

3.1 Cross Industry Standard of Data Mining

Data mining is a process of knowledge extraction or information discovery from a large number of data (Kurgan and Musilek, 2006). In this sense, one can argue that data mining is a blanket term for all kind of knowledge discovery processes including machine learning. For data mining to work, several technologies and techniques need to work together. Data mining incorporates various techniques and technologies like database management, computing infrastructure, statistical methods, data visualization, and communication. In order to successfully carry out knowledge discovery process one needs effective standards for various aspects of data mining.

Conceived by the consortium of multinational companies, Cross-Industry Standard of Data Mining (CRISP-DM) encapsulates a tried and tested method for data mining that builds upon previous attempts to define knowledge discovery methodologies. Although it is a somewhat old process conceived in 1996, the iterative process outlined in a CRISP-DM is still pragmatic in contemporary data science projects. Apart from being an iterative data-driven process, CRISP-DM also links the business understanding and objectives to data science projects. The inclusion of a business aspect to data mining adds a more nuanced business perspective to data science projects. CRISP-DM is one of the most popular and broadly adopted data mining models which has been acknowledged and relatively widely used in both research and industrial communities (Kurgan and Musilek, 2006). The figure 3.1 below summarized the iterative process of CRISP-DM.

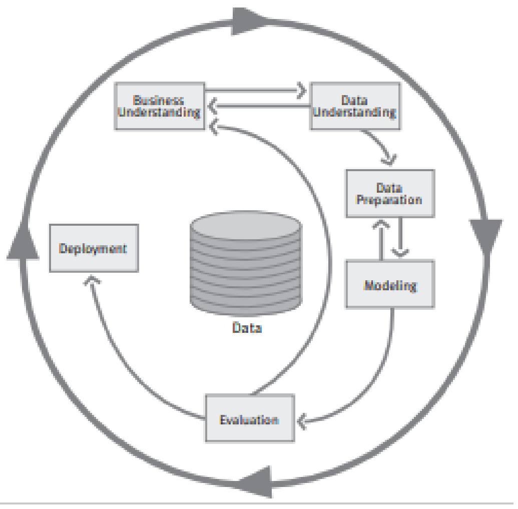


Figure 3.1: CRISP-DM process

CRISP-DM is a comprehensive process that starts with business understanding and ends with the final deployment of a data mining solution. Therefore it has various sub-steps within its main six phases. This thesis does not strictly follow the entire process because the primary objective of the thesis is not necessarily to obtain and deploy a production-level model but to find a way to improve the existing model with the help of data augmentation. Therefore, CRISP-DM is used as the reference model for developing and evaluating various machine learning algorithms built in the thesis.

3.1.1 Business understanding

All data science projects are a means to an end, the end usually being the business objective. To successfully carry out the data science project, a clear understanding of the underlying business objective is vital. So in this phase, the prime focus is on understanding the project objectives and requirements from a business perspective, then converting this knowledge into a data mining problem definition and data mining objective (Chapman et al., 2000).

3.1.2 Data understanding

After understanding the business goal and setting the data mining objective next logical step is to collect and analyze the relevant data. This stage primarily deals with querying data from the database and checking whether the data is fit to carry out data mining process. Similarly, the first general insights are also discovered during this phase which helps to enhance the understanding of data at hand.

3.1.3 Data preparation

Data in its raw-form cannot directly be used for modeling. Data needs to go through some preparatory steps. Therefore this phase includes various activities undertaken to construct final data-set which will be fed into machine learning models. The first step often will be to clean the data, meaning remove the data that is corrupted, missing or unnecessary. Then if needed perform transformation such as normalization. This is also a phase where various augmentation techniques are applied to get the final data that is fed to the deep learning model.

3.1.4 Modeling

This phase involves selecting the appropriate machine learning model which can achieve the goal outlined in the initial phase. Usually, various methods can be used to achieve the same goal. For example, one can use various machine learning strategies for classification task from simple logistic regression to convolutional neural network. Thus, different methods can be tested to determine the optimal method. This phase also includes designing testing set and some initial assessment of the model.

3.1.5 Evaluation

Once the modeling is completed models need to be evaluated. Before proceeding to deployment, the model needs to be thoroughly evaluated to make sure that it achieves the business objective and needs. Model evaluation can take several forms such as simple analysis of the accuracy of model's prediction on evaluation data, analyzing the complexities of the algorithm to the speed at which model takes time to execute. There is no one silver bullet for model evaluation as there are a variety of evaluation measures depending on the business objective and need. Thus, a model needs to evaluated based on the measure that most accurately reflects the business needs.

In this thesis, an entire chapter has been devoted to model evaluation as one of the crucial tasks of the thesis is to evaluate various machine learning models obtained through various augmentation techniques.

3.1.6 Deployment

Once the suitable model meeting business needs is created, it needs to be deployed in the field. Depending on the objective of business it could be from as complicated as deploying machine learning model in a distributed cloud environment to as simple as generating a simple report for communicating and implementing the insights gathered from data mining process. Since model deployment is out of the scope of the thesis, it will not be discussed in subsequent chapters.

Chapter 4

Implementation

This section outlines the application of data mining process. In this thesis python programming language is used to implement various data augmentation as well as to build deep learning model. Deep learning is implemented using Keras. “Keras is a high-level neural network API written in Python and is capable of running on top of TensorFlow, CNTK or Theano”– (Chollet et al., 2015).

4.1 Business understanding

As outlined in the motivation section, case company manufactures the portable near-infrared spectrometer. Near-infrared spectroscopy(NIRs) can measure the chemical composition of biological materials by analyzing the diffuse reflectance or transmittance of the samples at several wavelengths from 700 to 2500 nanometers. One of the unique properties of infrared spectrum is that no two organic compounds have the same infrared spectrum (Royal Society of Chemistry, 2017). Pure compounds can be identified by examination of their spectra provided that a chemist has a copy of the spectrum so any unknown pure compound can be identified by making a comparison (Royal Society of Chemistry, 2017). This process is straightforward in theory, however in practice due to sample variation, environmental factors and device variations, spectra from two same compounds might not necessarily be identified by making a simple comparison. That is why a machine learning model is needed which takes into consideration several factors and can identify compound based on previously trained data.

NIRs has been widely used in various food commodities especially in the grain, cereal products, and oil-seed processing industries (Elmessery and Abdallah, 2014). It is also extensively used in chemical and pharmaceutical industries for classification of raw materials (Elmessery and Abdallah, 2014). NIRs technology is gaining popularity for material detection as it is fast, reliable, non-destructive, and relatively cheaper (Evans et al., 1999). Furthermore, there is very minimal sample preparation, or pretreatment and material are preserved after the examination. Therefore NIRs solution provides various advantages over the other types of material sensing solutions.

One of the major hindrances for the application of NIRs has been the size and cost. Conventional spectrometers are relatively big and are also costly. However, the spectrometers produced by the case company are portable and economical. As discussed above, portability adds some complexities, however the advantages over-weight the drawbacks. The portability offers a novel application of NIRs solution in various mass markets, not just limited to traditional agricultural and pharmaceutical industries.

Overall innovative NIRs solution offered by the case company has significant potential and offers quite many advantages over a traditional sensor. Despite its advantage, there are also several challenges. Traditionally the measurement with the spectrometer is carried out in a controlled environment in a laboratory setting where the environmental variables such as temperature, lights and moisture level can be controlled, enabling consistent measurements. Due to the portable nature of the product, above mentioned environmental variables cannot be controlled. The fluctuation in environmental variables introduces fluctuation in a measurement. The fluctuation of measurement means that the classifier needs to be trained in a diverse sample of data, mimicking the actual use condition. Similarly, samples in the real world come from various sources with a various level of concentration. At present companies try to capture all the environmental as well as sample variation in a laboratory to produce training samples. However, there are several variations, and this takes a considerable amount of time to produce data to train a deep learning model. Similarly, due to regulatory restriction pharmaceutical materials are not readily available, restricting the pool of available samples to train a deep learning model. Furthermore, due to the nature of the manufacturing process, no two scanners are the same. So this means there will be slight variation in a measurement of the same sample from two different scanners. Therefore there is a need for robust model that can take various factors in consideration and correctly classify materials.

The primary business objective is to produce a robust model that can classify material from the measurement data produced from the portable spectrometer with reasonable accuracy despite all the challenges discussed above. The aim is also to reduce the data acquisition cost and time. To achieve this business goal, data augmentation with deep learning is deemed to be the suitable tool, as data augmentation has the potential to save both time and money in data collection and also improve the generalization ability of the model by increasing the diversity of samples. As mentioned above, there is a precedence of data augmentation being used in a similar situation with a successful result.

4.2 Data understanding

The primary source of data analyzed in this thesis comes from the NIR spectrometer. A spectrometer analyses a compound by passing near-infrared radiation over a range of different frequencies and measuring the absorption made by each type of bond in a compound. This produces a spectrum, usually a plot of percent transmittance against wave-number. In simple words, transmittance describes how much light passes through a sample unchanged and is measured as a percentage. The image illustrated in figure 4.1 below illustrates the typical data obtained from spectrometer which is a plot of percent transmittance against wave-number.

As discussed above NIRs is used extensively in pharmaceutical industry

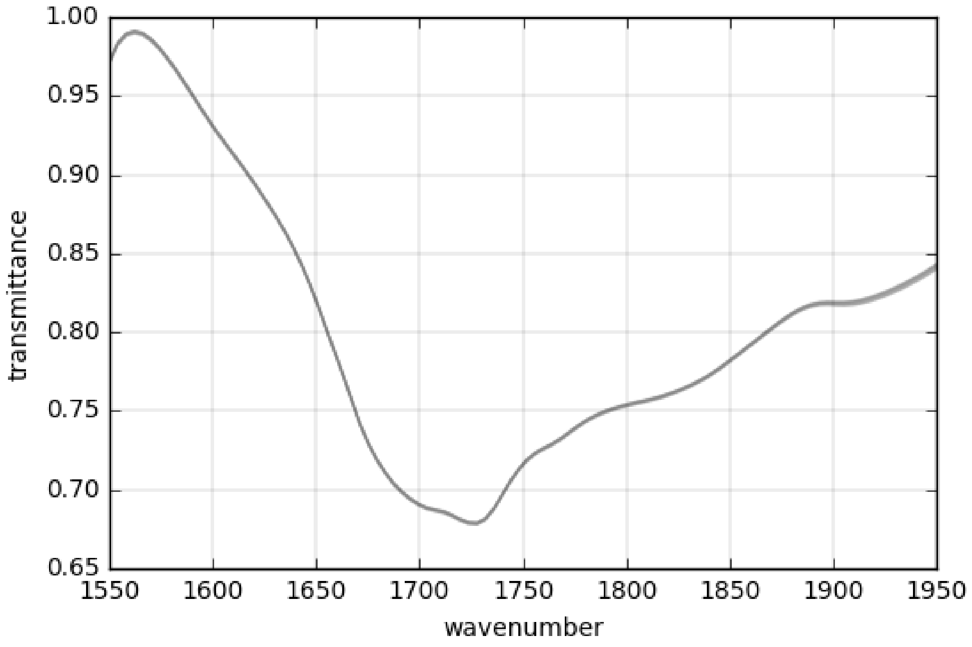


Figure 4.1: Spectral data

to detect raw materials. Incidentally, pharmaceutical raw materials are also among the most heavily regulated materials, so there is limited sample available for training a neural network. As a result, pharmaceutical raw materials are ideal for testing effectiveness of data augmentation. The model built in the thesis will try to detect four different pharmaceutical raw materials. These four different pharmaceutical compounds will be called as variables '0',1','2','3' and '4' respectively. These variables are coded as a categorical variable with label '0','1','2','3' and '4'. The compound '0','1','2' and '3' are hard to acquire. Label '4' constitute the compounds that are readily available, and therefore this label is the majority class.

4.2.1 Exploratory Analysis

Pharmaceutical compounds '0', '1','2' and '3' are closely related to each other but are not the same compound. Whereas the compound '4' contains several different compounds that do not belong to any '0','1','2' and '3'. The target variables are not 100 percent pure compound. The concentration of the target varies anywhere from 5 to 100 percent. Figure 4.2 shows the concentration count for all the compound.

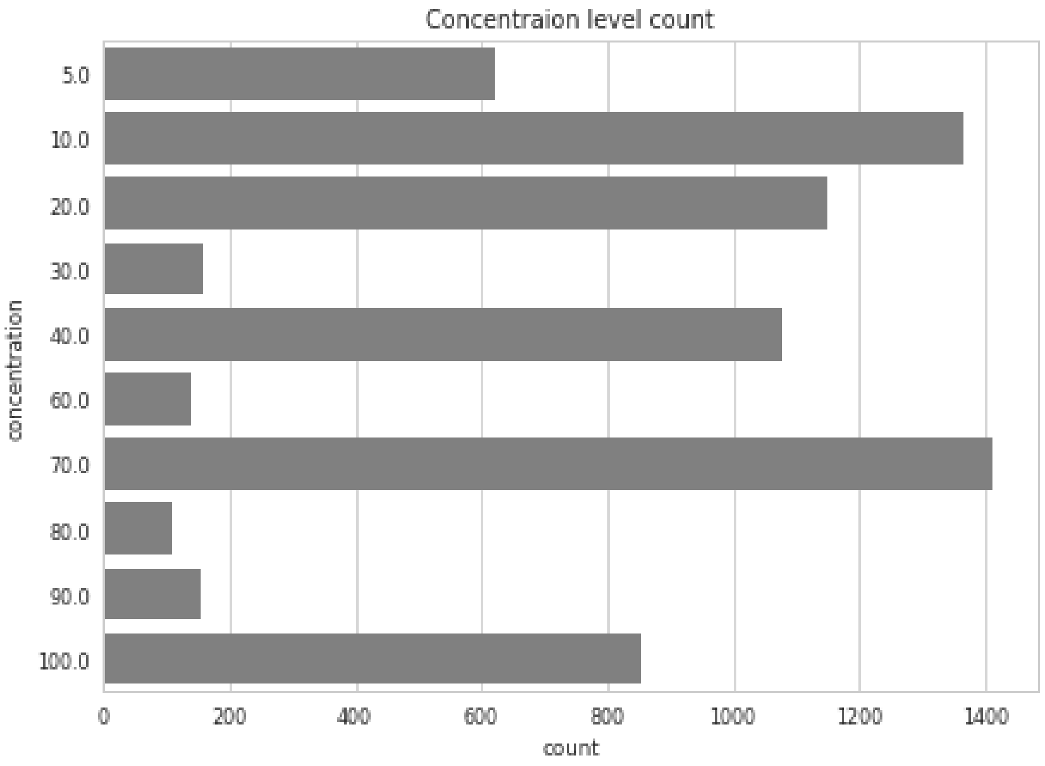


Figure 4.2: Concentration level of compounds

As we can see from the figure 4.2 concentration varies quite a lot. Most compounds have the concentration of 70 percent followed by the 10 percent. Not many compounds have the concentration level of 80 percent. The concentrations of unknown samples are unknown and therefore are not shown in the figure 4.2. The task of the classifier is to detect the compound regardless of the concentration. Given the spectral data, the classifier has to detect whether there is a presence of target compound or not irrespective of its concentration level.

4.3 Data preparation

Spectral data is collected from the spectrometer and stored in a central database. The data does not need extensive data preparation as it is reasonably clean with some exception of missing labels. To prepare the data, samples with missing labels are removed, and the data is transformed using a couple of domain-specific data transformations such as Fourier transformation. Fourier transformation takes a signal and expresses it in terms of the frequencies of the waves that make up that signal. These transformations enable us to compress the data to 50 points from 100 points of spectra.

As discussed earlier, data preparation phase included all the steps undertaken to create a final data-set that is fed to the deep learning model. Consequently, data augmentation is a part of this phase. Above mentioned augmentation techniques, namely data warping, SMOTE, and GANs, will be implemented in the following sub-section.

4.3.1 Data augmentation using data warping

Data warping is one of the simplest and easiest ways to apply data augmentation. In a prominent neural network implementation library Keras, data augmentation for images such as transformation, scaling, rotation and other similar transformations are natively supported. As described earlier, data warping creates new data primarily using a transformation that rotates, shifts or adds distortion which still preserves the label. This type of augmentation for image data is quite straightforward as these types of transformations preserve the image label. For example, rotated image data of a cat still retains its label as cat image upside down is still a cat image. However, for spectral data, augmenting the data by rotation no longer preserves the label. In this sense augmenting data using the data warping technique is highly domain specific and requires a thorough understanding of data at hand.

For spectral data, the appropriate augmentation was to add a small amount of randomly distributed slope so that the points are slightly shifted up or down from the original data. By adding a small amount of slope, we assumed that it could mimic the variation caused by measuring the same sample under different conditions such as humidity, temperature and sample concentration. The figure below illustrates the augmentation on the spectral data along with the similar augmentation on image data. The simple algorithm used to augment data is presented in appendix A.

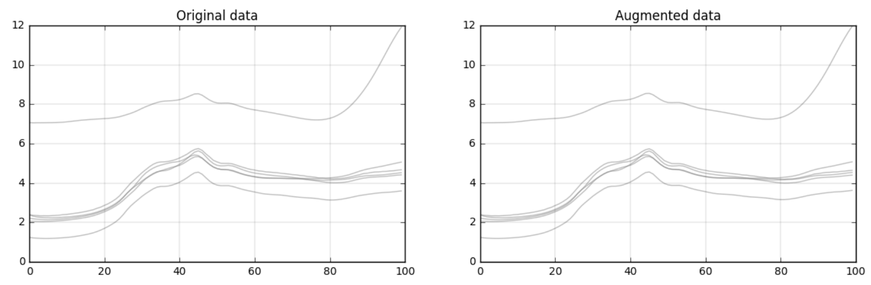


Figure 4.3: Augmentation using data warping

The random points added to the original point is minimal, therefore augmented data is not easily differentiable from the original. However, on close observation, we can observe that augmented data have points slightly up or down compared to the original figure. This type of data augmentation is akin to distorting the image by adding random noise. Depending upon the augmentation the value of noise can be increased or decreased by adjusting the parameters from where the random noise is generated.

4.3.2 Data augmentation using SMOTE

SMOTE was conceived to better handle the imbalance data-sets by oversampling the minority class. What SMOTE does is it leaves the majority class intact, in this case unknown samples, and it creates synthetic samples for all other minority classes so that all the class labels are represented equally in the final data set. The details of how SMOTE is implemented is already discussed in the theoretical review section. The augmentation using SMOTE was implemented via python module called 'imblearn' (Lemaître et al., 2017). This implementation is displayed in appendix B. Figure 4.4 shows the generated synthetic samples from SMOTE.

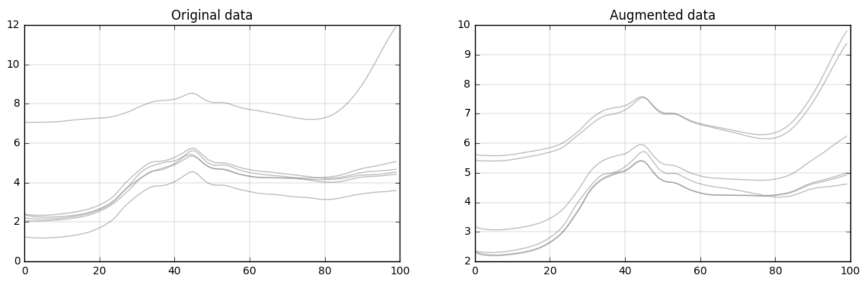


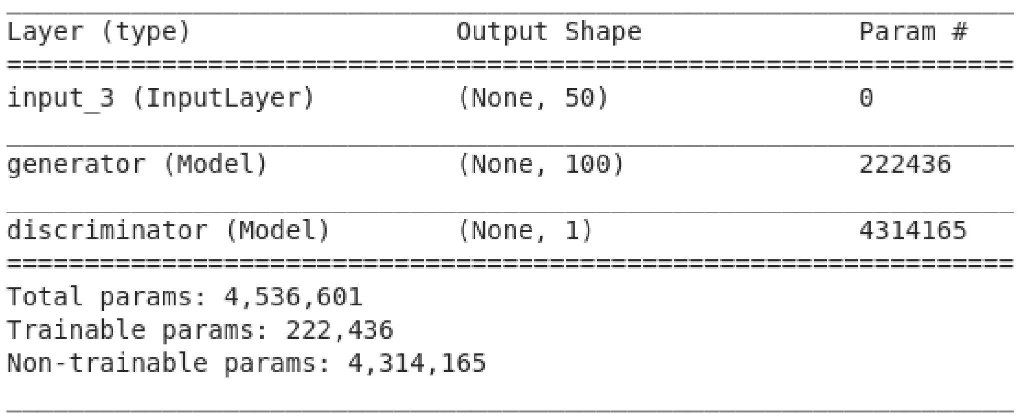
Figure 4.4: Augmentation using SMOTE

In the figure 4.4 we can see that augmented data have a bit more variation than the augmented data through data warping. The new generated spectral data is just an interpolation of new data between existing data. As we can observe from the figure 4.4 original y value is within the range of 1 to 12, and the augmented data y value is also within that same range. One can argue that this behavior is ideal because in a field setting the spectrometer measurement can fluctuate within the range of training samples.

The overall idea of augmentation in SMOTE is quite similar to data warping. The augmented data is merely shifted up or down. However, in data warping the data was created simply by adding random points to one spectral data. Here the data is interpolated between a spectral data and its k-nearest neighbor. This technique increases the variation of the sample compared to data warping.

4.3.3 Data augmentation using GANs

To implement the GANs laid out in the literature review we make two deep learning models: one generator and another discriminator. Then we stack together the two models and start a model training process. The architecture of the GANs model is illustrated in the figure below.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         (None, 50)                0
_________________________________________________________________
generator (Model)            (None, 100)               222436
_________________________________________________________________
discriminator (Model)        (None, 1)                 4314165
=================================================================
Total params: 4,536,601
Trainable params: 222,436
Non-trainable params: 4,314,165
_________________________________________________________________
```

Figure 4.5: The implementation of GAN in keras

Here the generator model turns a vector that is randomly sampled from normal distribution into a data mimicking the actual data generated from the sensor. One common problem with GANs is sometimes generator gets stuck with generating data that is just noise. A possible solution is to use a dropout layer on both the discriminator and the generator. Here the discriminator model takes as input a data that is generated or actual and classifies it into one of two classes: “generated data” or “real data”.

The figure 4.5 shows a GANs network architecture in Keras interface. In GANs architecture two models, discriminator and generator, are chained together into GANs network. When trained, the model will move the generator in a direction that improves its ability to fool the discriminator. Generator and discriminator are just two neural networks.

The first step of the training process is to draw random points from the normal distribution, and then generate a data from those random points. The second step is to mix the generated data with actual data, then label it either "real" or "generated" and feed it into the discriminator. The third step is to draw new random points from a normal distribution and train GANs using these random vectors, which are all labelled as "real". This third step effectively updates the weights of the generator to move them toward getting the discriminator to predict "real" for generated data (Chollet, 2017). The figure 4.6 shows the generated images from the GANs. From the figure 4.6 we can see that augmentation from GANs is unlike any other before. The other augmentation strategies closely replicated the actual shape of original data, whereas generated data from GANs do not have smooth slopes like original data. The full implementation using python and Keras is illustrated in appendix D.

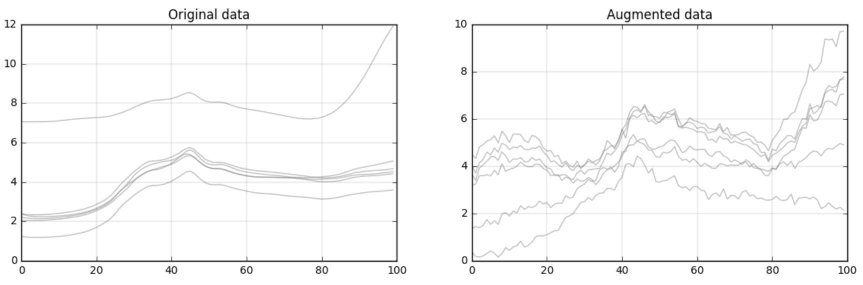


Figure 4.6: Augmentation using GANs

4.4 Modeling

In this phase, we build deep learning models using data prepared through different augmentation methods. A neural network was chosen primarily because of its ability to handle high dimensional data, the requirement of little feature engineering and the availability of open source libraries that can leverage graphical processor for a speedy result. There are various types of neural networks, and in this thesis, we will be working with the kind of network known as the convolutional neural network.

Convolutional neural network was chosen primarily because unlike the typical dense layers, convolution layers learn local patterns in their input feature space whereas dense layers learn global patterns (Chollet, 2017). Furthermore, convolution net can learn the spatial hierarchies of patterns. These critical characteristics of convolution net makes it translation invariant, meaning that the convolution net can recognize the patterns, no matter where they are. (Chollet, 2017). For example, convolution network is quite successful in image classification because it can detect the local patterns along with its spatial hierarchies such as eye, nose and mouth, no matter where they are located in the image. Since we are treating the data obtained from spectrometer as a fingerprint of the particular chemical compound, identifying local patterns and hierarchies in spectral data can help in accurately classifying one fingerprint from another. The figure 4.7 illustrates the architecture of the convolution neural network used for modeling.

The model was trained for 30 epochs in mini-batches of 2000 samples. Epoch means merely the number of iteration over all the samples in training data, whereas batch is the number of training data that is used to update the weights. We do not pass the entire data set into the neural network at once but instead divide the data into a number of small batches. On completion of every batch, the loss is computed, and weights of networks get updated.

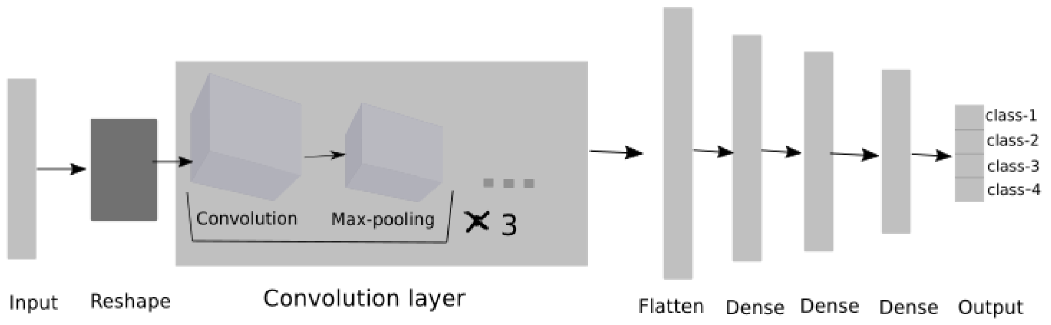


Figure 4.7: Model Architecture

Picking the right network architecture is more of an art than a science–(Chollet, 2017). Having said that, there are some best practices and generally accepted conventions. The architecture of network proposed in the thesis is based on the best practices and generally agreed principles. For instance, categorical cross-entropy is used as the objective function of the model, as it is generally used for multi-class classification problems. Similarly, convolutional layers are followed by a max-pooling layer which is also a common practice in designing convolutional network.

Figure 4.7 gives the graphical overview of the network architecture, for full detail overview see ”Appendix C”. The first layer simply reshapes the input data into a shape that is compatible with one dimensional convolution layer. Then three subsequent layers are convolution layers. In between each convolution layer there is a max-pooling layer. Max-pooling layer will perform down-sampling operation reducing the number of parameters and computation in the network. This helps to reduce over-fitting. Then the following fully connected layer will combines high level feature learned by convolution network and passes it to dense layer. The last two dense layer are connected through a dropout layer which randomly droops out (setting to zero) a number of output features of the layer during training. This operation is also done to avoid over-fitting. The last layer is a Dense layer that has a softmax activation function with 4 units, which is needed for this multi-class classification problem. This means that the resulting output will be array of length four for each input vector, the array contains the likelihood of the input vector belonging to each class.

In summary, in the implementation chapter we have implemented various data augmentation techniques and generated artificial spectral data. Similarly we have also implemented convolutional neural network to classify pharmaceutical material.

Chapter 5

Evaluation

In this section, we will dissect the various models built in modeling phase using the evaluation framework established in the literature review. Data were partitioned approximately fifty-fifty into training and testing set. The training set was used both to train the model and also to generate augmented data. Finally, the test set was used to evaluate the model. As discussed in literature review, F1-score is chosen as the primary measuring rod for the model performance and benchmarking.

5.1 Base model without augmentation

The first model was built using training data without augmentation. This model primarily serves as a benchmark model against which various other models with augmentation will be compared. Table 5.1 below summarizes the result from the first model without using any augmented data.

Table 5.1: Evaluation metrics of first model without augmentation

Metrics	**Value**
Accuracy	0.97
Precision	0.69
Recall	0.59
$F_1 - score$	0.63

The overall accuracy of the first model is 0.96, and the F1-score which is the harmonic mean of macro-precision and macro-recall is 0.63. Since this is a multi-class classification with highly imbalanced data, accuracy alone does not provide the full picture. As we can see, the accuracy of the model is relatively high. However, the model does not correctly classify minority labels reflected by low F1-score.

5.2 Augmentation using data warping

The second model is built using the same network but with augmentation. In this case, we have used a simple data warping technique defined in the implementation section. It has the same number of epochs and batch size. In this model trained data is balanced using augmentation. Through augmentation, generated data is more than ten times the original data. Table 5.2 summarizes the results obtained from the model.

Table 5.2: Evaluation metrics of second model with augmentation

Measures	**Value**
Accuracy	0.95
Precision	0.76
recall	0.88
$F_1 - score$	0.77

Augmentation using data warping shows some immediate improvement over F1-score as it climbs up to 0.77 with the considerable increase in both precision and recall. However, this model increases F1-score at the expense of some misclassification in majority class reflected by a decrease in accuracy.

5.3 Augmentation using SMOTE

The third model is built using the same network using SMOTE as augmentation method. SMOTE has been thoroughly discussed in the literature review and as mentioned previously, SMOTE was implemented here using the python's third-party module named 'imblean'. It also has the same number of epochs with same batch size. Here augmented data is more than 10-times the actual data and augmented data is used to balance training sets labels. Table 5.3 summarizes the results obtained from the model.

Table 5.3: Evaluation metrics of third model with augmentation

Measures	**Value**
Accuracy	0.98
Precision	0.84
recall	0.81
$F_1 - score$	0.80

Augmentation using data warping also shows improvement over base model as F1-score reaches 0.80. This model significantly improved the classification performance of both minority and majority class compared to the previous models. The improvement in performance is evident by the increase in both F1-score and accuracy.

5.4 Augmentation using GANs

The fourth and final model is built using the augmented data from GANs network. We have created a generator for each class except the majority class. Using the generator for each class, approximately thirty thousand new samples of minority classes were created. The classification model uses the same network with the same number of epochs and batch size as previous models. Table 5.4 summarizes the results obtained from the model.

Table 5.4: Evaluation metrics of fourth model with augmentation

Measures	**Value**
Accuracy	0.99
Precision	0.92
recall	0.89
$F_1 - score$	0.91

Augmentation using data generated from GANs also shows improvement over base model as F1-score reaches 0.91, with a considerable increase in both precision and recall.

5.5 Model Comparison

From the evaluation of the model, we can infer that classification models performed much better when using augmented data. There was a significant improvement over the base model without using augmentation when compared to a model created through augmented data. Relative increase in accuracy of the model with and without augmentation is not significant. However, if we observe precision, recall and F1-score there seem to be a considerable improvement. This improvement in F1-score improves the quality of prediction reducing the misclassification rate of an infrequent nevertheless important minority class.

Augmentation using data warping was an improvement over base model even though overall accuracy declined by three percentage points. There is an improvement of more than 10 points in F1 score compared to the base model. An apparent increase in F1 score and a small decline in accuracy means that the model built using data warping as augmentation strategy is better able to detect minority classes than the base model but at the cost of some misclassification of majority class.

Table 5.5: Model comparison based on F1-score

Models	**F1-score**
Base model	0.63
Data warping	0.77
GANs	0.91
SMOTE	0.80

Classification model built using SMOTE technique seems to outperform both the base model and the model with data warping augmentation. With SMOTE we can see improvement in accuracy as well as in F1-score. Unlike data augmentation using warping, the model built using SMOTE does equally well in classifying both majority and minority class. Overall accuracy is quite high, however there is still space for improvement in classifying minority class as the F1-score is relatively low compared to accuracy.

The fourth model built using augmented data from GANs generator performed better than all the previous models. There is an improvement in both accuracy and the F1-score compared to all the previous models. There is a quite significant improvement over the base model reflected by an increase of 30 points in F1-score. Although GANs provide a significant improvement over the base model, we can still see that there is room for some more improvement in the F1 score.

To conclude, model with augmentation provide significant improvement in material classification. Using F1-score as the measure we can rank the performance of GANs model as first, SMOTE as second and data warping as third respectively. Among all the models GANs had a significant improvement in the model as it improved performance on detecting majority and minority classes.

Chapter 6

Discussion

We have built and compared various data augmentation strategies using deep learning models. Although thesis has followed best practices and generally accepted conventions from designing the deep learning model to formal data research methods like CRISP-DM, there are inevitably some limitations. Furthermore, this thesis with its limited scope and time could not go through all the possible augmentation techniques. This chapter discusses the possible shortcomings of the thesis and how this study can be extended in the future.

6.1 Limitations

One of the critical assumptions in augmentation is that the augmentation process retains the label. This means that if the spectral data from the particular material is augmented then the augmented data still represents the same material even though it is slightly different from the original spectral data. This assumption mostly holds true for image augmentation as an image of blurred or rotated cat image is nevertheless a cat image. However, for spectral data, it is not apparent at what point changing the data also changes the underlying representation of its label. Therefore there is no straight way to intuitively check whether augmentation preserves the essential characteristics of spectral data.

Similarly, at the implementation phase, we realized that due to the way the augmentation method is implemented and also due to the assumption of label invariance, we were unable to use augmentation for a problem other than classification problem. Thus we are unable to use data augmentation for one of the prominent areas of NIRs application which is detecting concentration level of particular raw material in a compound. Augmentation methods discussed here only support the class data, and also assuming that augmentation will preserve the label for a particular point instead of a class will be a very broad assumption.

Furthermore, this study should not be taken as definitive proof that augmentation techniques discussed here will always perform better than the model without augmentation. For instance, if there is plenty of available data, further augmentation might not result in better performance. In addition, augmentation can be domain specific, so the type of augmentation that works in one domain might not necessarily work in other cases.

Lastly, the evaluation metrics exclusively evaluated models based on the classification result on the test set. This approach has some limitations. Due to several iterations, even though the test was untouched during the training period, the model might have been slightly biased. Similarly, in real life application, just the accuracy on the test might not be sufficient to evaluate the models. Factors such as model complexity and time taken to train the model needs to be considered too.

6.2 Future work

The future work could focus on extending the current work by tackling above mentioned limitations and also experimenting with more augmentation techniques that were not part of this study.

There are other various augmentation techniques that this research could not cover due to limited time and scope. For example, there are other promising generative models which were not tested in this research like variational autoencoder and other different variations of GANs itself. These generative models could provide a further improvement over the existing model. Similarly, there are also different variations of SMOTE which could be evaluated against generative models. We can also create new augmented data by combining existing augmentation methods and creating entirely new augmentation scheme.

We could extend the present model to solve regression problem by redefining the regression problem as a classification problem where possible. For instance, regression problem can be reformatted by binning the points and classifying the data in bins. This approach could broaden the application of data augmentation in NIRs.

All in all, like every study, this study too has its own limitations. This opens up new avenues for future work that could improve and extend this study. Data augmentation offers a big opportunity and there are growing works done in this field. It is unlikely that we can find one augmentation approach that works in all the cases, and therefore several augmentation approaches need to be tested in order to come up with the augmentation approach that fits individual need.

Chapter 7

Conclusions

In this thesis, we have brought forward various augmentation frameworks, from simple augmentation using data warping to generative adversarial network. The key findings and the contribution of this paper are discussed below.

7.1 Key Findings

The thesis was set out to improve the classification model using data augmentation. After comparing the models with and without augmentation, we can infer that in the case of NIRs data, augmentation can help improve classification accuracy of model especially in the case of imbalanced data sets. In all the models built for this research, augmentation was used to balance the data set which enhanced the mode performance. In the field of NIR spectroscopy, one of the principal limitations of its application has been the collection of data. As demonstrated in the thesis, augmentation can help mitigate this issue. From the evaluation of various augmentations, we observed that augmentation using the GANs was superior to the augmentation using data warping and SMOTE.

7.2 Contributions

As mentioned in the introduction section, there was a primarily two-fold objective of the thesis. One was to provide a theoretical contribution in the field of deep learning and data augmentation, and the other was to propose concrete augmentation framework for the case company to improve the performance of a deep learning model.

7.2.1 Theoretical Contribution

Data augmentation is a contemporary issue in the field of data science. It has been extensively used in multidimensional data, for example, image detection and computer vision problems. However, data augmentation has not been widely used in one-dimensional data like spectral data or time series data. So through the application in one-dimensional data, this thesis hopes to contribute in growing literature of application of data augmentation.

Similarly, GANs is a concept that was developed only recently, making its application a contemporary research topic in the field of deep learning. By applying this new concept into the business case, we hope to further the understanding of the generative models and their possible implications in businesses.

7.2.2 Practical Contribution

This thesis has applied theoretically grounded methods to tackle a business problem faced by a case company. One of the primary objectives was to produce methods to improve the classification accuracy of existing model. Although the model presented in the thesis requires further evaluation and test, it provides a working framework to improve the model performance using data augmentation. The thesis offers a way to tackle the problem faced by the industry in general. The application if NIRs is restricted due to high data acquisition cost. Data augmentation can save data acquisition cost and extend the application of NIRs in other fields.

Furthermore, all the experiments are done using open source technology and framework. Therefore, findings of the thesis can easily be implemented for product development in the case company. Similarly, other researchers or companies interested in data augmentation can also benefit from this as frameworks for augmentation is laid out in the thesis.

Appendix A

Augmentation using data warping

```
def augmenting_f(X):
        """Augment data by adding random slope background"""
        total_samples = X.shape[0]
        n_features = X.shape[1]

        xs = np.linspace(0, 1, n_features)
        xs = np.tile(xs, (total_samples, 1))

        # expand and transpose so that numpy broacasting
   multiplies each
        # row in the features by one slope
        slopes = np.random.normal(0, 0.03/2.355, total_samples)
        slopes = np.expand_dims(slopes, axis=0).T
        X += xs*slopes

        return X
```

Listing A.1: Function to create data warping

Appendix B

Augmentation using SMOTE

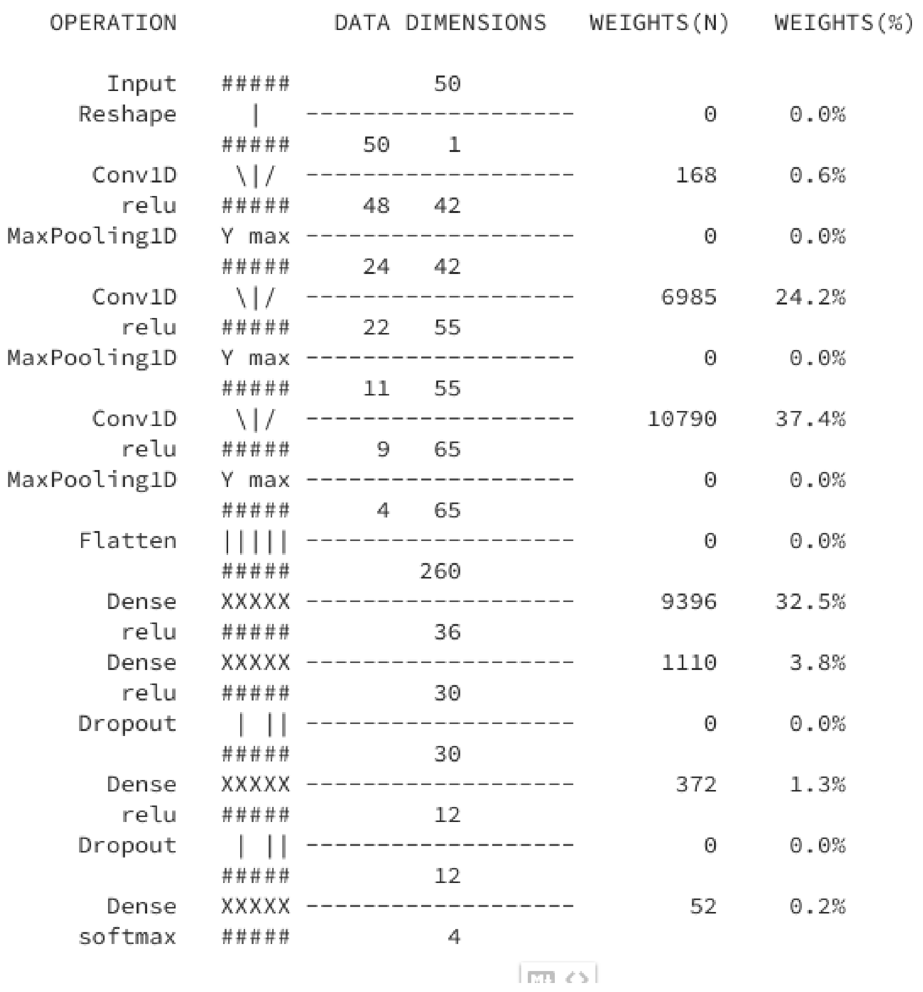
```
from imblearn.over_sampling import SMOTE, ADASYN

augmented_x, augmented_y =
    SMOTE(random_state=123).fit_sample(origina_x, original_y)
```

Listing B.1: Implimentation of SMOTE

Appendix C

Neural network



```
    OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

        Input   #####         50
      Reshape     |   -------------------         0     0.0%
                #####     50    1
       Conv1D    \|/  -------------------       168     0.6%
         relu   #####     48   42
 MaxPooling1D   Y max -------------------         0     0.0%
                #####     24   42
       Conv1D    \|/  -------------------      6985    24.2%
         relu   #####     22   55
 MaxPooling1D   Y max -------------------         0     0.0%
                #####     11   55
       Conv1D    \|/  -------------------     10790    37.4%
         relu   #####      9   65
 MaxPooling1D   Y max -------------------         0     0.0%
                #####      4   65
      Flatten   ||||| -------------------         0     0.0%
                #####        260
        Dense   XXXXX -------------------      9396    32.5%
         relu   #####         36
        Dense   XXXXX -------------------      1110     3.8%
         relu   #####         30
      Dropout    | || -------------------         0     0.0%
                #####         30
        Dense   XXXXX -------------------       372     1.3%
         relu   #####         12
      Dropout    | || -------------------         0     0.0%
                #####         12
        Dense   XXXXX -------------------        52     0.2%
      softmax   #####          4
```

Figure C.1: Detail overview of the network

Appendix D

Augmentation using GANs

```
class GAN():
        def
   __init__(self,train_data,rand_dim,dense_unit,batch_size,
        log_interval=500):

        self.train = train_data
        self.rand_dim = rand_dim
        self.data_dim = train_data.shape[1]
        self.dense_unit = dense_unit
        self.batch_size = batch_size
        #self.nb_steps = nb_steps
        self.log_interval=log_interval
        self.disc_loss_real = []
        self.disc_loss_generated = []
        self.combined_loss = []



        optimizer = keras.optimizers.Adam(lr= 5e-4,
   beta_1=0.5, beta_2=0.9)

        # Build and compile the discriminator
        self.discriminator = self.build_discriminator()
        self.discriminator.compile(optimizer=optimizer,
   loss='binary_crossentropy')

        # Build and compile the generator
        self.generator = self.build_generator()
        self.generator.compile(optimizer=optimizer,
   loss='binary_crossentropy')

        # The generator takes noise and input
        z = keras.Input(shape=(rand_dim,))
        #gen = self.generator(z)

        # For the combined model we will only train generator
        self.discriminator.trainable = False

        # Discriminator takes generated image as input and
   checks its validity
        dis = self.discriminator(self.generator(z))

        # The combined model = generator + discriminator tkaes
        # noise as input => generated images => determines
   validity
```

```
        self.combined = keras.models.Model(z,dis)
        self.combined.compile(optimizer=optimizer,
loss='binary_crossentropy')

        def build_generator(self):
        generator_input = layers.Input(shape=(self.rand_dim, ))
        x = layers.Dense(self.dense_unit,
activation='relu')(generator_input)
        x = layers.Dense(self.dense_unit*2,
activation='relu')(x)
        x = layers.Dense(self.dense_unit*4,
activation='relu')(x)
        x = layers.Dense(self.data_dim)(x)
        generator_model =
models.Model(inputs=[generator_input],
        outputs=[x],name='generator')
        return generator_model

        def build_discriminator(self):
        discriminator_input =
layers.Input(shape=(self.data_dim,))
        x = layers.Reshape((-1, 1))(discriminator_input)
        x = layers.Conv1D(100, 20, activation='relu')(x)
        x = layers.Dropout(0.2)(x)
        x = layers.Flatten()(x)

        x = layers.Dense(self.dense_unit*4,
activation='relu')(x)
        # x = layers.Dropout(0.1)(x)
        x = layers.Dense(self.dense_unit*2,
activation='relu')(x)
        # x = layers.Dropout(0.1)(x)
        x = layers.Dense(self.dense_unit, activation='relu')(x)
        x = layers.Dense(1, activation='sigmoid')(x)
        # x = layers.Dense(1)(x)

        generator_model=
models.Model(inputs=[discriminator_input],outputs=[x],
        name='discriminator')
        return generator_model


        def get_data_batch(self,seed):

        # iterate through shuffled indices, so every sample
gets covered evenly
        start_i = (self.batch_size * seed) %
(self.train.shape[0])
        stop_i = start_i + self.batch_size
        shuffle_seed = (self.batch_size * seed) //
self.train.shape[0]
        np.random.seed(shuffle_seed)
        # wasteful to shuffle every time
```

```
     train_ix = np.random.choice(
list(range(0,self.train.shape[0])), replace=False,
size=(self.train.shape[0]) )
     # duplicate to cover ranges past the end of the set
     train_ix = list(train_ix) + list(train_ix)
     x = self.train[ train_ix[ start_i: stop_i ] ]

     return np.reshape(x, (self.batch_size, -1) )


     def train_mod(self,nb_steps=100):

     #---------------------
     # Train Discriminator
     #---------------------
     count = 0
     fig, axes = plt.subplots(10,
2,figsize=(10,15),sharex=True,sharey=True)
     #subplot_kw=dict(facecolor='white')
     for i in range(nb_steps):
     K.set_learning_phase(1) # 1 = train

     # train the discriminator
     for j in range(1):
     np.random.seed(i+j)
     z = np.random.normal(size=(self.batch_size,
self.rand_dim))
     x = self.get_data_batch(seed=i+j)

     g_z = self.generator.predict(z)
     #              x = np.vstack([x,g_z]) # code to train
the discriminator on real and generated data at the same
time, but you have to run network again for separate losses
     #              classes =
np.hstack([np.zeros(batch_size),np.ones(batch_size)])
     #              d_l_r =
discriminator_model.train_on_batch(x, classes)

     d_l_r = self.discriminator.train_on_batch(x,
np.random.uniform(low=0.999, high=1.0,
size=self.batch_size)) # 0.7, 1.2 # GANs need noise to
prevent loss going to zero
     d_l_g = self.discriminator.train_on_batch(g_z,
np.random.uniform(low=0.0, high=0.001,
size=self.batch_size)) # 0.0, 0.3 # GANs need noise to
prevent loss going to zero
     # d_l_r = discriminator_model.train_on_batch(x,
np.ones(batch_size)) # without noise
     # d_l_g = discriminator_model.train_on_batch(g_z,
np.zeros(batch_size)) # without noise
     self.disc_loss_real.append(d_l_r)
     self.disc_loss_generated.append(d_l_g)
```

```
        #--------------------
        # train the generator
        #--------------------

        for j in range(1):
        np.random.seed(i+j)
        z = np.random.normal(size=(self.batch_size,
    self.rand_dim))
        # loss = combined_model.train_on_batch(z,
    np.ones(batch_size)) # without noise
        loss = self.combined.train_on_batch(z,
    np.random.uniform(low=0.999, high=1.0,
    size=self.batch_size)) # 0.7, 1.2 # GANs need noise to
    prevent loss going to zero
        self.combined_loss.append(loss)
        plt.ioff()

        # Saving weights and plotting images
        if i % (nb_steps/10)/(nb_steps/100) == 0:
        real = self.get_data_batch(seed=i)
        fake = self.generator.predict(z)


        #ax1 = add_subplot(221,facecolor='white')
        #ax2 = subplot(222,facecolor='white')
        #self.f.title('Real')

        axes[count,0].title.set_text('Real,␣epoch␣%s' %i)
        axes[count,1].title.set_text('Fake,␣epoch␣%s' %i)
        for plots in range(self.batch_size):

        axes[count,0].plot(real[plots],alpha=0.5,color='grey')
        axes[count,1].plot(fake[plots],alpha=0.5,color='grey')

        #self.f.title('Fake')

        count += 1
        #   print('Step: {} of {}.'.format(i, starting_step +
    nb_steps))
        #  K.set_learning_phase(0) # 0 = test

        # loss summaries
        print('Generator␣model␣loss:␣
    {}.'.format(self.combined_loss[-1]))
        print('Discriminator␣model␣loss␣gen:␣
    {}.'.format(self.disc_loss_generated[-1]))
        print('Discriminator␣model␣loss␣real:␣
    {}.'.format(self.disc_loss_real[-1]))

        return self.generator, fig
```

Listing D.1: Implimentation of GANs

Appendix E

Data generation through GANs

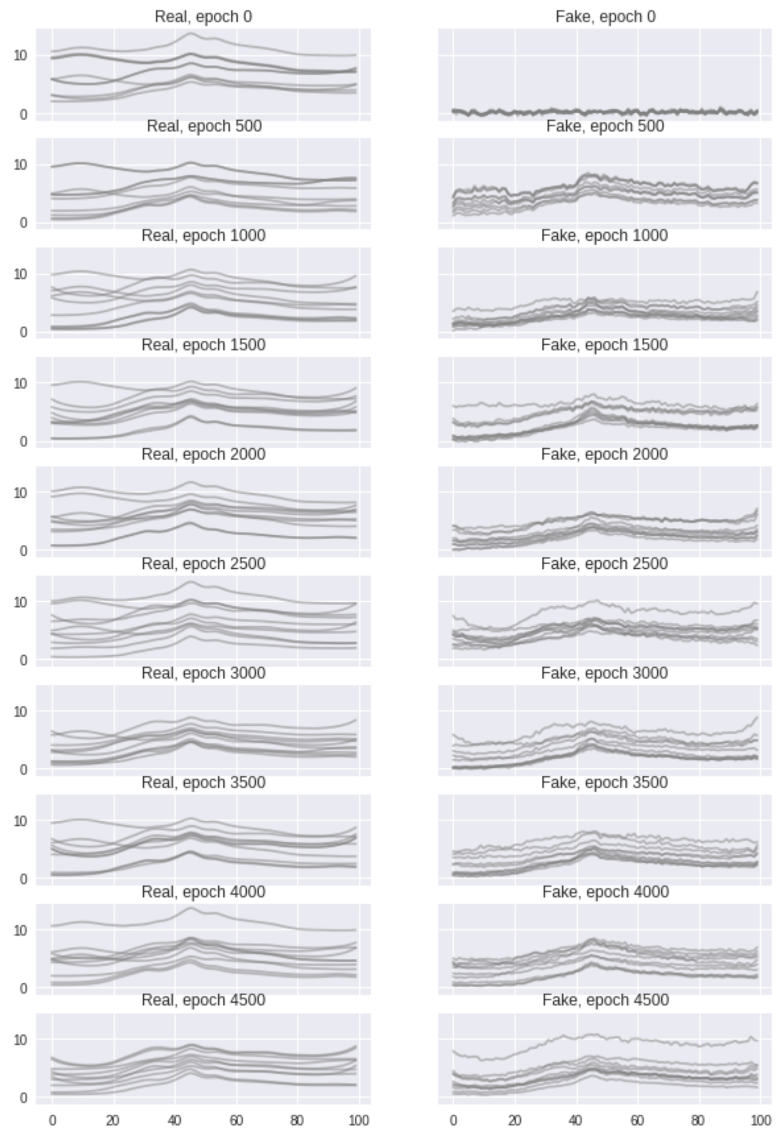


Figure E.1: Data generation process using GANs

Appendix F

Deep learning model without augmentation

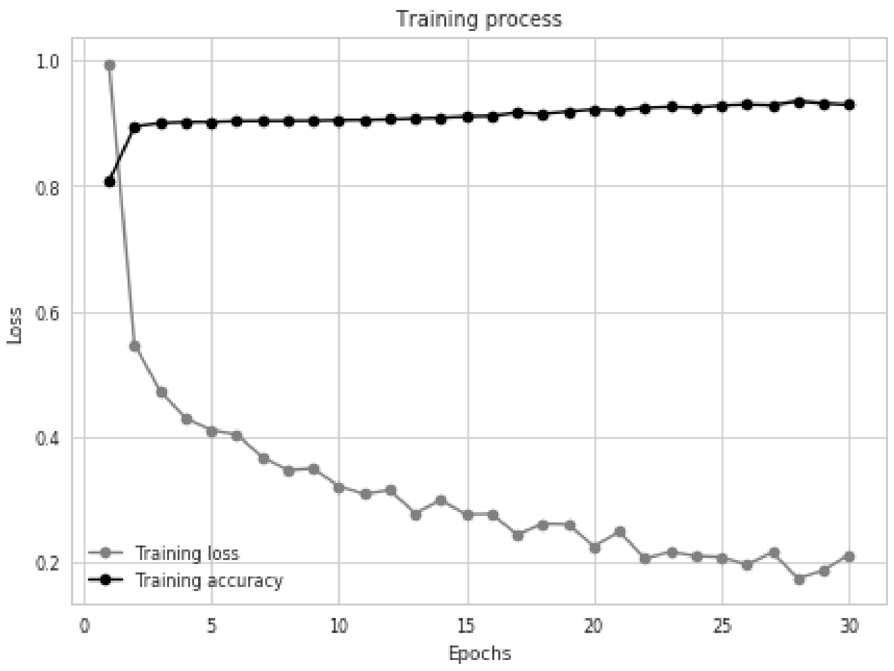


Figure F.1: Deep learning model without augmentation training process

Appendix G

Deep learning model training

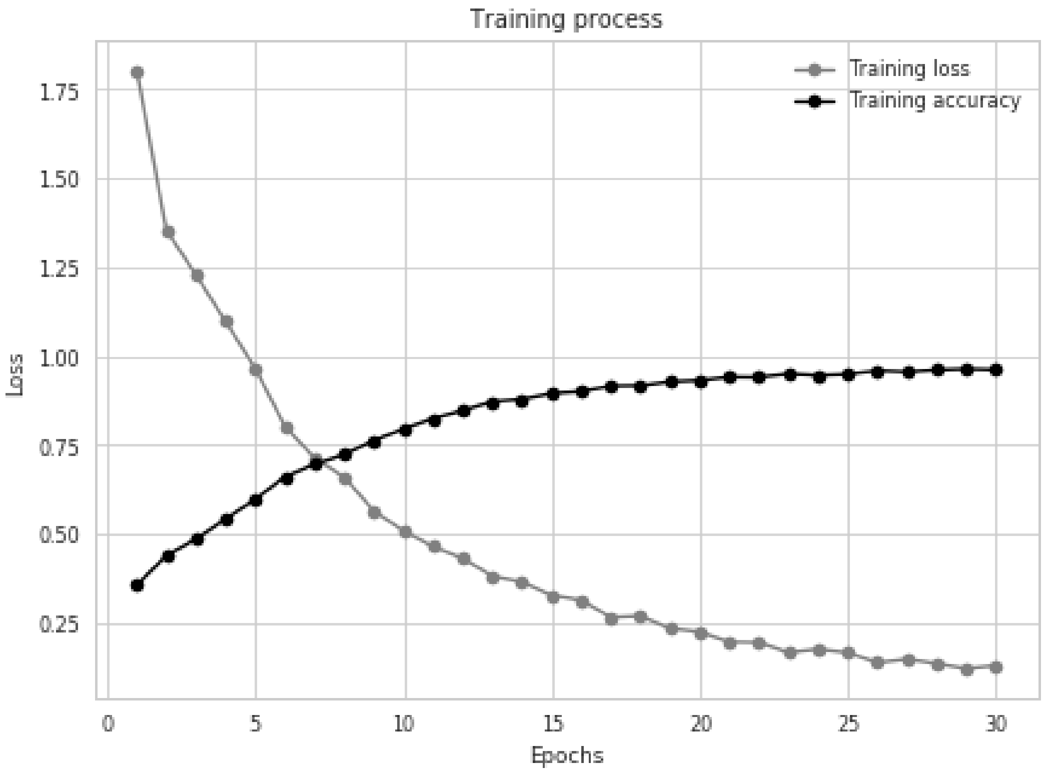


Figure G.1: Training loss and accuracy for model with augmentation usign data-warping

Appendix H

Deep learning model training

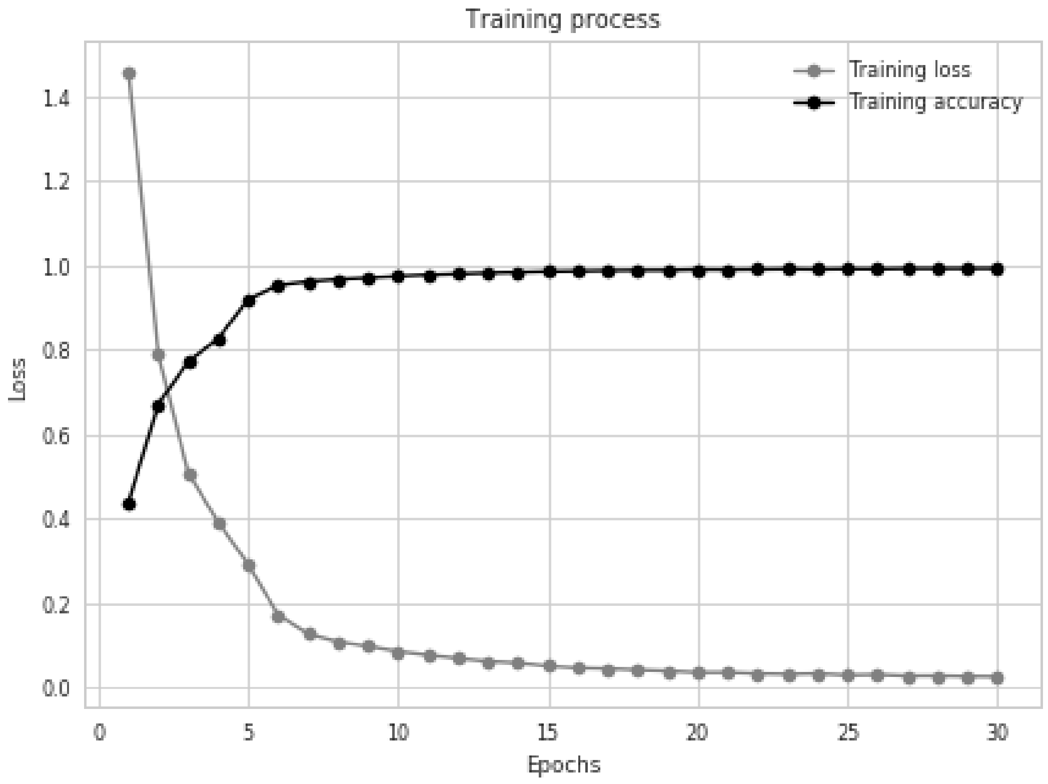


Figure H.1: Training loss and accuracy for model with augmentation usign data-warping

Appendix I

Model misclassification

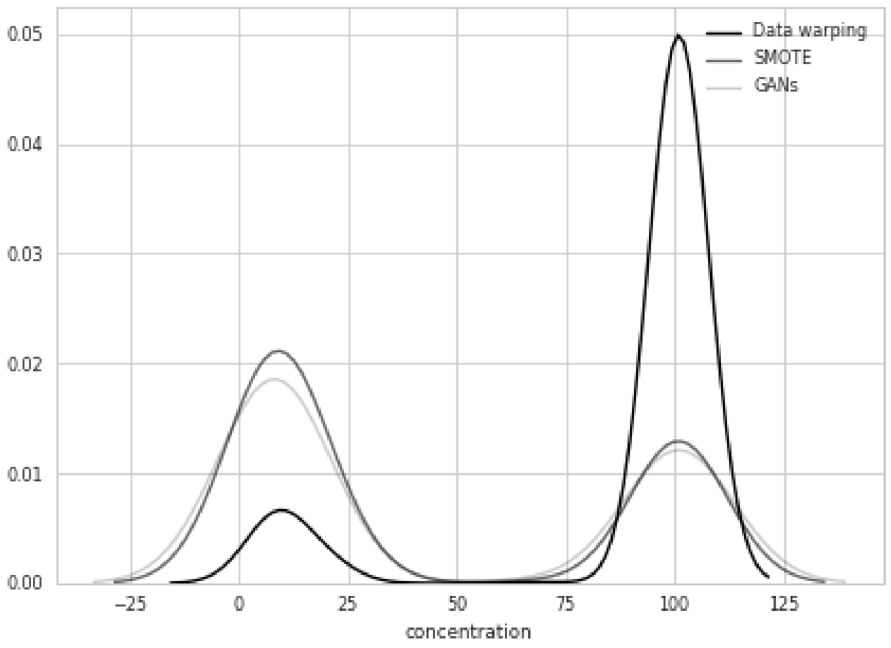


Figure I.1: Density plot of model missclassification at various concentration level

Bibliography

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R. (2000). Crisp-dm 1.0 step-by-step data mining guide.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Chollet, F. (2017). *Deep Learning with Python*. Manning Publications Company.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Elmessery, W. M. and Abdallah, S. E. (2014). Manufacture evolution of a microbial contamination detection unit for processed tomatoes inside food factories. *AMA-AGRICULTURAL MECHANIZATION IN ASIA AFRICA AND LATIN AMERICA*, 45(4):32–38.

Evans, A., Huang, S., Osborne, B., Kotwal, Z., and Wesley, I. (1999). Near infrared on-line measurement of degree of cook in extrusion processing of wheat flour. *Journal of Near Infrared Spectroscopy*, 7(2):77–84.

Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. (2018). Synthetic data augmentation using gan for improved liver lesion classification. *arXiv preprint arXiv:1801.02385*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Gurumurthy, S., Sarvadevabhatla, R. K., and Radhakrishnan, V. B. (2017). Deligan: Generative adversarial networks for diverse and limited data. *arXiv preprint arXiv:1706.02071*.

Kortylewski, A., Schneider, A., Gerig, T., Egger, B., Morel-Forster, A., and Vetter, T. (2018). Training deep face recognition systems with synthetic data. *arXiv preprint arXiv:1802.05891*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kurgan, L. A. and Musilek, P. (2006). A survey of knowledge discovery and data mining process models. *The Knowledge Engineering Review*, 21(1):1–24.

Larochelle, H., Bengio, Y., Louradour, J., and Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(Jan):1–40.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lemaître, G., Nogueira, F., and Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., and Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1.

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

Royal Society of Chemistry (2017). Infrared spectroscopy (ir). `http://www.rsc.org/learn-chemistry/collections/spectroscopy/introduction#IRSpectroscopy`. Accessed: 2018-05-08.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.

Wong, S. C., Gatt, A., Stamatescu, V., and McDonnell, M. D. (2016). Understanding data augmentation for classification: when to warp? In *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on*, pages 1–6. IEEE.

Yaeger, L. S., Lyon, R. F., and Webb, B. J. (1997). Effective training of a neural network character classifier for word recognition. In *Advances in neural information processing systems*, pages 807–816.