Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Stella Student

Software Processes for Dummies:
Re-inventing the Wheel

Master's Thesis
Espoo, June 18, 2011

DRAFT - typed [May 16, 2018]

Supervisors: Professor Antti Ylä-Jääski, Aalto University
Professor Pekka Perustieteilijä, University of Helsinki
Advisors: Olli Ohjaaja M.Sc. (Tech.), Aalto University
Elli Opas M.Sc. (Tech), Aalto SCI

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Aalto University
School of Science

ABSTRACT OF
MASTER'S THESIS

Author:	Stella Student	
Title:	Software Processes for Dummies: Re-inventing the Wheel	
Date:	June 18, 2011	**Pages:** vii + 58
Major:	Data Communication Software	**Code:** T-110
Supervisors:	Professor Antti Ylä-Jääski Professor Pekka Perustieteilijä	
Advisors:	Olli Ohjaaja M.Sc. (Tech.) Elli Opas M.Sc. (Tech)	

The thesis example file (`thesis-example.tex`), all the chapter content files (`1introduction.tex` and so on), and the Aalto style file (`aalto-thesis.sty`) are commented with explanations on how the Aalto thesis works. The files also contain some examples on how to customize various details of the thesis layout, and of course the example text works as an example in itself. Please read the comments and the example text; that should get you well on your way!

Keywords:	ocean, sea, marine, ocean mammal, marine mammal, whales, cetaceans, dolphins, porpoises
Language:	English

Acknowledgements

I wish to thank all students who use LaTeX for formatting their theses, because theses formatted with LaTeX are just so nice.

Thank you, and keep up the good work!

Espoo, June 18, 2011

Stella Student

Contents

List of Tables

List of Figures

This page was intentionally left blank.

Chapter 1

Introduction

Motivation

In past decades deep learning has garnered a lot of attention. Primarily due to its success at discovering complex structures in high-dimensional data. This has lead to the application of deep learning to many domains of science, business, and government. Compared to conventional machine learning method 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 algorithm along with hardware has resolved majority of bottlenecks in applications of deep learning and made deep learning 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 (if manually sorted) 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 learn-

ing 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 increase 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 organic material. These optical fingerprints of chemical components obtained from spectrometer can be used to identify various materials. To correctly identify material spectrometer relies on deep learning model which have 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 regions. For such materials 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 can streamline the process of material sensing solution and also open up new business opportunity which would have been not possible due to limited availability of annotated samples.

Therefore, by exploring the effectiveness of available data augmentation technique to improve 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 application of data augmentation in various other domains.

Research Problem

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

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

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, 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 fairly new technique and it is difficult to translate its theoretical framework into real application.

Therefore, 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 do so, 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 in exploration and implementation of data augmentation to solve the problem posed by limited training data.

Structure of the Thesis

The thesis starts off by laying out the motivation for research. Then provides an overview of relevant literature. In Chapter 3 A brief overview of the process of conducting 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. Chapter 6 is concluding chapters where the insight gained from the evaluation is discussed and concluded by laying out the limitation and contribution of this paper.

Table 1.1: Thesis structure

Chapter 1: Introduction
Chapter 2: Literature review
- Machine Learning
- Deep Learning
- Data Augmentation
- Model Evaluation
Chapter 3: Methods
- Crisp DM process
Chapter 4: Implementation
- Business understanding
- Data understanding
- Data preparation
- Data modeling
Chapter 5: Evaluation
- Base Model
- Augmentation using data warping
- Augmentation using SMOTE
- Augmentation using GANs
Chapter 6: Discussion and Conclusion
- Research Finding
- Limitations and future work

Chapter 2

Literature Review

In this chapter, the major groundwork and preliminaries related to the subject of the study, is going to be reviewed. We find it necessary to provide a general overview of machine learning, deep learning, and data augmentation as these key concepts are the foundational stone of the thesis.

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 and then the machine produces the rules that match given input to its expected output (Chollet, 2017).

At the heart of the machine learning lies input data, expected output and a way to measure whether the rule learned by the machine is reliable or not. The job of the machine learning is to transform input data into meaningful output. 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 as representation. To summarize, machine learning is an iterative process of finding the appropriate representation of input based on the expected output and feedback from the metrics that determined

the reliability or accuracy of current representation.

Deep Learning

Deep learning 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 a representation at a higher, more abstract level (LeCun et al., 2015). The deep in deep learning comes from the sequence of layers/representation used in a learning process. Depending upon the task number of layers in representation varies but the modern state of the art deep learning system can have hundreds of successive layers of representation.

In deep learning, layered representation is learned 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 network so that the final output most accurately estimates the actual output. In order 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. This whole process is done through optimizer which uses the backpropagation algorithm. During the first iteration weights of the network are assigned random values which mean 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 a close as they can be to actual output. The figure 2.1 summarizes the basic building block of a deep learning process.

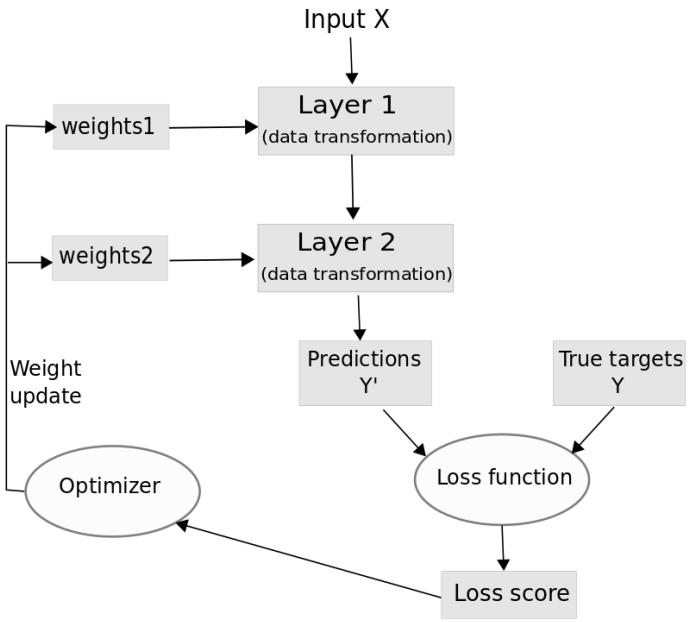

Figure 2.1: Building block of deep learning process

Why deep learning?

Deep learning requires relatively large amount of data and significant computing power to analyze the data. Furthermore, it also requires an efficient algorithm that can calculate hundreds of weights. These requirement posed by the deep learning were the major bottleneck for application of deep learning in early 90's. However, nowadays there has been a substantial improvement in terms of Central Processing Unit (CPU) and Graphical Processing Unit (GPU). Similarly, digitalization and increasing reach of Internet has generated a significant amount of data. Furthermore, open source projects contributed by both corporation and researcher in the field of deep learning has streamlined the implementation and deployment of deep learning. As a result of all this, deep 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 result (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) and better quality of generated samples by generative probabilistic models (Najafabadi et al., 2015). Compared to conventional machine learning technique 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 project 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 allowing it to be trained on any size of data. Furthermore, deep learning models are modular allowing the layers from model to be reused to solve similar problems. These interesting properties make deep learning intriguing and useful for researcher and companies alike.

Supervised learning

Machine learning problems can be broadly classified into Supervised and Unsupervised learning. The most common form of machine learning problem is supervised learning. Like the majority of machine learning problem, 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.

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 overfitting 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 method namely data warping, synthetic minority over-sampling technique, and generative adversarial nets.

Data Warping

The most conventional application of data augmentation has been in image classification, and it has proven to be fairly 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 augmentation, 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 individual write letter slight differently. Augmenting data through this the technique of data warping depends on the problem at hand and sometimes are heuristic in nature which is not guaranteed to be optimal in other situations.

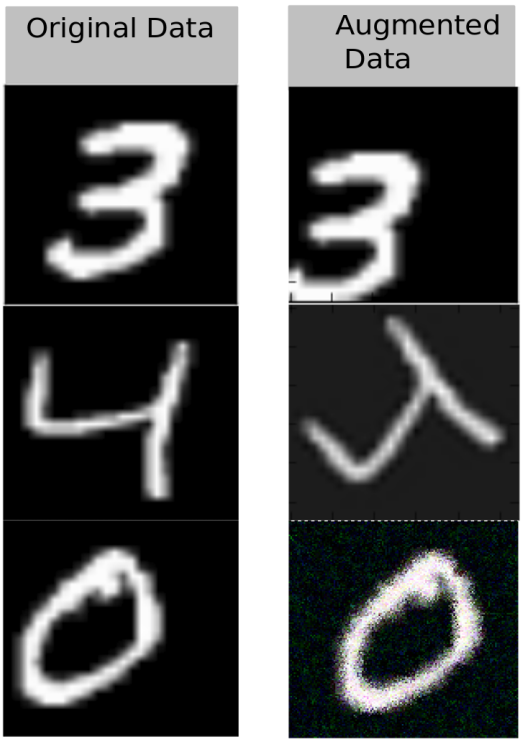

Figure 2.2: Augmentation via data warping

The figure 2.2 shows the data augmentation applied to image data using various data warping technique. Top image is warped using simple random shift of pixels, where as the bottom image of zero is warped by adding random noise to pixels, lastly the middle image of four is warped using simple rotation of pixels. This simple technique can make the model more robust as in real life application such as hand written recognition people write the same digit with different style such as slightly shifted or rotated left or right. This analogy can

be extended to the spectral data also as they can shift slight up or down due to the variation in concentration or due environmental factors compared to the sample prepared in lab.

SMOTE

Synthetic Minority Over-Sampling Technique (SMOTE) is data augmentation technique which is inspired by data warping, particularly its ability to reduce the class imbalance in hand-written digit problem (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 datasets 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, 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 example, let x be a sample from minority class. For each sample x, the other samples from the same minority class with the smallest Euclidean distance from the x are identified, these samples are also called as nearest neighbor. One of the nearest neighbors is randomly picked x^R. The new synthetic sample is defined in the equation below.

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

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

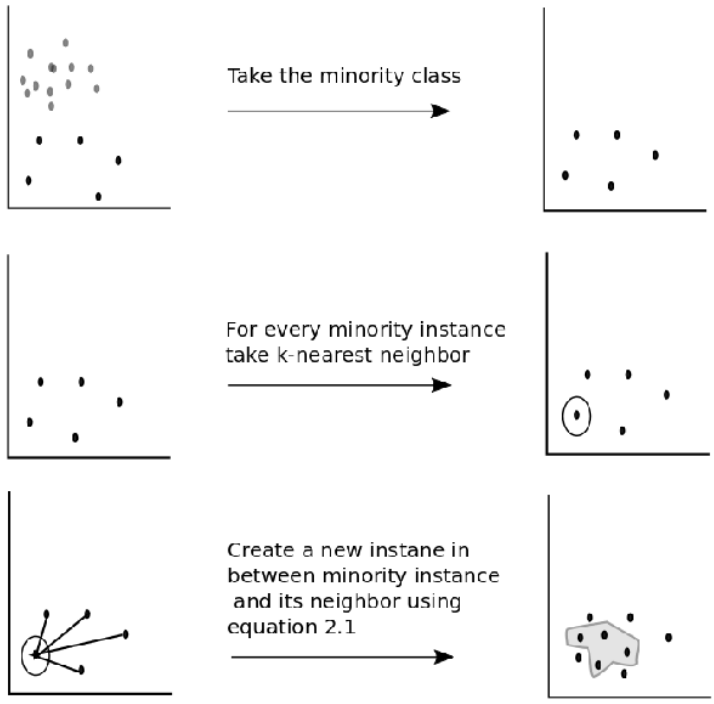

Figure 2.3: Synthetic data generation using SMOTE

One alternative to SMOTE is to oversample the minority class with replacement. However research has shown that this practice can increase the number of samples in training set but 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 causes model to over-fit the data. Ideally we would want to increase the decision boundary of minority class so that it generalizes well for unseen data. In the same paper application of SMOTE on various datasets showed that SMOTE approach can improve the accuracy of classifier for a minority class.

GAN

One of the promising technique 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 GAN: Generator and Discriminator. The role of generator network is to take a random input and generate a sample data, whereas the role of the discriminator is to take input from the real data or 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 minimax 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 $Pd(x)$
$Pd(z)$	The generator's distribution
Z	Sample form $P(z)$
$G(z)$	Generator Network
$D(x)$	Discriminator Network

Here, generator $G(z)$ takes an input from $P(z)$. The generated data is supplied to discriminator network along with the actual data. The task of discriminator network is to take an input from real and generated data and try to predict whether the input is real or generated. It takes input x from $Pd(x)$ where $Pd(x)$ is the real distribution, $D(x)$ then solves a binary classification

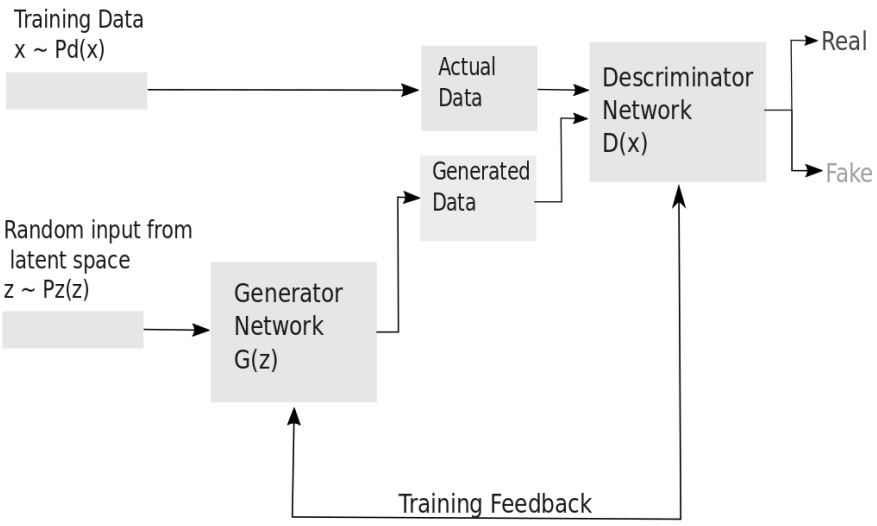

Figure 2.4: GANs architecture

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:

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

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 application such as computer vision and image recognition. GANs have been effective 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 in the rain using only data collected on a sunny day (Gurumurthy et al., 2017). Recently there has been many studies in application of GANs in medical diagnosis particularly to augment the medical imaging data. Medical industry also faces the similar problem as data is heavily regulated and costly data acquisition. In one of the recent study conducted by Frid-Adar et al. (2018) authors successfully used GANs to augment the limited medical image data to improve deep learning model. Using GANs as an augmentation authors were able to gain 7 percent improvement in accuracy over the traditional augmentation like data warping.

GANs offer a unique proposition in data augmentation. As very little human interaction is needed. Unlike data warping, instead of figuring out augmentation that is label invariant we let the deep learning model to find the label invariant augmentation. Similarly the expectation from the GANs model is that the data is generated from the actual data distribution so that the new generated data has the core characteristics of real data.

Model Evaluation

Machine learning divides classification 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). Since the thesis tries to evaluate the performance of various deep learning models, metrics for model assessments needs to be clearly laid out.

Empirical evaluation remains the most used approach for the model assessment. The empirical comparison is done through applying algorithms from the

model on various data sets and then evaluating the performance of the classifiers that the algorithms have produced. Often the performance of the classifiers is evaluated through accuracy. The performance of a classification algorithm from given model can also be evaluated by 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 below.

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 simply the proportion of the total number of prediction that was correct. Precision and recall provided more insight than accuracy. To take the analogy of cancer diagnosis problem, precision tells what proportion of patients diagnosed by 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 it breaks 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. Furthermore, often precision and recall are expressed in one number which is its weighted average known as the F1 score. The F1 score is simply 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. 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, each of which can be useful in various scenarios. One way to averaging binary metric is macro averaging. Macro averaging simply calculates the mean of the binary metrics, giving equal weight to each class. In problems where infrequent classes are nonetheless, 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, model metrics are merely the extension from binary classification metrics. It is often the case that no singular metrics can comprehensively measure and evaluate the quality of 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 model based not only on the accuracy of majority class but also based on infrequent minority classes.

Chapter 3

Methods

Cross Industry Standard of Data Mining

Data mining is a process of knowledge extraction or information discovery from large amount 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 process including machine learning. For data mining to work several technologies and technique needs to work together. Data mining incorporates various techniques and technologies like database management, computing infrastructure, statical methods, data visualization and communication. In order to successfully carry out knowledge discover process one needs effective standards for various aspects of data mining.

Conceived by the consortium of multinational companies, Cross-Industry Standard of 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 fairly old process conceived in 1996, the iterative process outlined in a CRISP-DM applies well in contemporary data science project. Apart from being iterative process CRISP-DM also links the business objectives and understanding to data science project, providing more nuanced business perspective to data science project. CRISP-DM is one of the most popular and broadly adopted data mining model 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.

Since CRISP-DM is a comprehensive process and contains various sub-steps

Figure 3.1: CRISP-DM process

within its main six phases. This thesis does not strictly follow the entire process primarily because the main objective of thesis is not necessarily to obtain and deploy production level model but to find way to improve the existing model with the helps of data augmentation. Therefore, CRISP-DM is used as reference model for developing and evaluating various machine learning algorithm built during in the thesis.

Business understanding

All data science project is a means to an end, the end usually being the business objective. To successfully carry out the data science project, the clear understanding of 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).

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.

Data preparation

Since data comes various shapes and forms rarely it is not possible to directly use data in its raw from. Therefore this phase includes various activities undertaken to construct final dataset which will be fed into machine learning models. The first step will be to clean the data remove the data that is corrupted or missing. Then perform transformation such as normalization. This is also a phase where various augmentation technique is applied to get the final data that is fed to the deep learning model.

Modeling

This phase involve selecting the appropriate machine learning model which can achieve the goal outlined in the initial phase. Usually there are various method can be used to achieve the same goal so different method can be tested to determine the optimal method. This phase also include designing the testing set and assessing the model.

Evaluation

Once the modeling is completed models need to be evaluated. Before proceeding to deployment model needs to be thoroughly evaluated to make sure that model achieves the business objective. Model evaluation can take several form from analyzing the complexities of algorithm to the speed at which model takes time to execute. Although small issues but the can affect the business goal.

In this thesis, an entire chapter has been devoted for model evaluation as one of the important task of the thesis is to evaluate various machine learning

model obtained through various augmentation technique.

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 as complex as deploying machine learning model in distributed cloud environment to as simple as generating simple report for communicating the result. 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 programing 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).

Business understanding

As outlined in a 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 nano meters. 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, device variations, etc. spectra from two same compounds might not necessarily identify by making a simple comparison. That is why 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 and commodities especially in

the grain, cereal products, and oilseed 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 is preserved after the examination. Therefore NIRs solution provides various advantages over the other type material sensing solution. One of the major hindrance for the application of NIRs has been the size and cost. Conventional spectrometers are relatively big and are also costly. However, the spectrometers produce by the case company are portable and economical. As discussed above portability add some complexities however the advantages overweights the drawbacks. The portability offers novel application of NIRs solution in various mass market not just limited to traditional agricultural and pharmaceutical industries.

Overall innovation NIRs solution offered by the case company has big potential and offers quite many advantage over traditional sensor. Despite its advantage there are also several challenges. Traditionally the measurement with the spectrometer is carried out in a controlled environment in the laboratory setting where the environmental variables such as temperature, lights and moisture level can be controlled which enables consistent measurements. Due to 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 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 tries to capture all the environmental as well as sample variation in a laboratory to produce training samples. However, there are several many variations, and this takes a considerable amount of time to produce data to train a deep learning model.

Similarly, due to regulatory restriction pharmaceuticals material are not readily available restricting the pool of available samples to train a deep learning model to identify the pharmaceutical material. Furthermore, due to the nature of manufacturing process no two scanner are the same there are slight variation from one scanner to another. So this means there will be slight variation in a measurement of same sample from two different scanner.

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 is deemed to be the most suitable tool as augmentation can both save time and money in data collection and also improve generalization ability of 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.

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 wavenumber. 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 wavenumber.

As discussed above NIRs is used extensively in pharmaceutical industry to detect raw materials. Incidentally, pharmaceutical raw materials are also among the heavily regulated materials, so there is limited sample available for

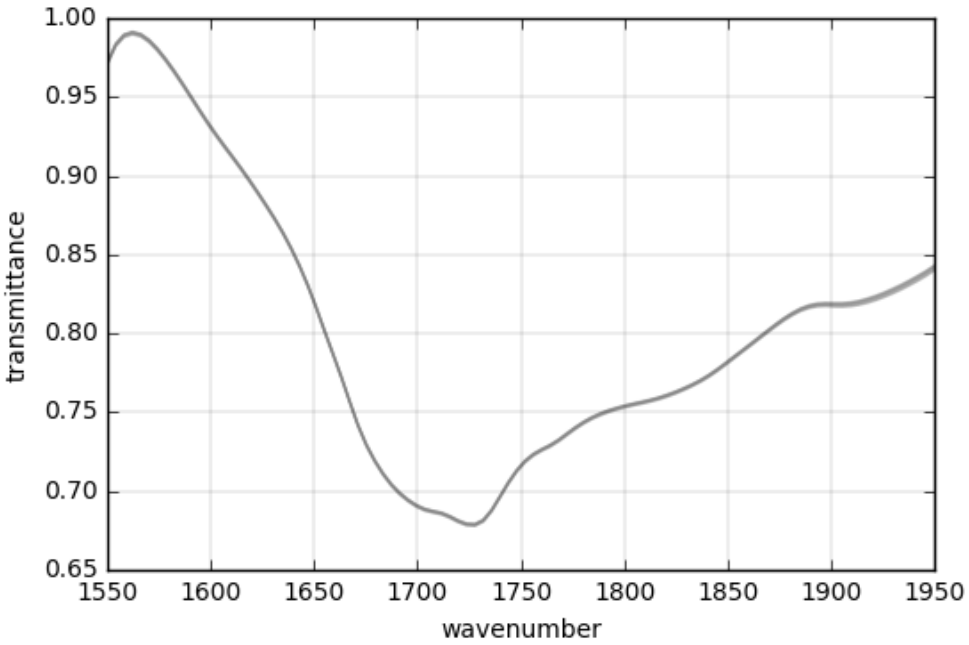

Figure 4.1: Spectral data

training a neural network. This makes the data ideal for testing effectiveness of data augmentation. The model built in the thesis will try to detect four different pharmaceutical raw material. These four different pharmaceutical compound will be called as variables '0',1','2','3' and '4' respectively. This variables are coded as a categorical variable with label '0','1','2','3' and '4'.

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' contain several different compounds that do not belong to any '0','1','2' and '3'. Essentially these are the unknown compounds representing various pharmaceutical materials. The target variable 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.

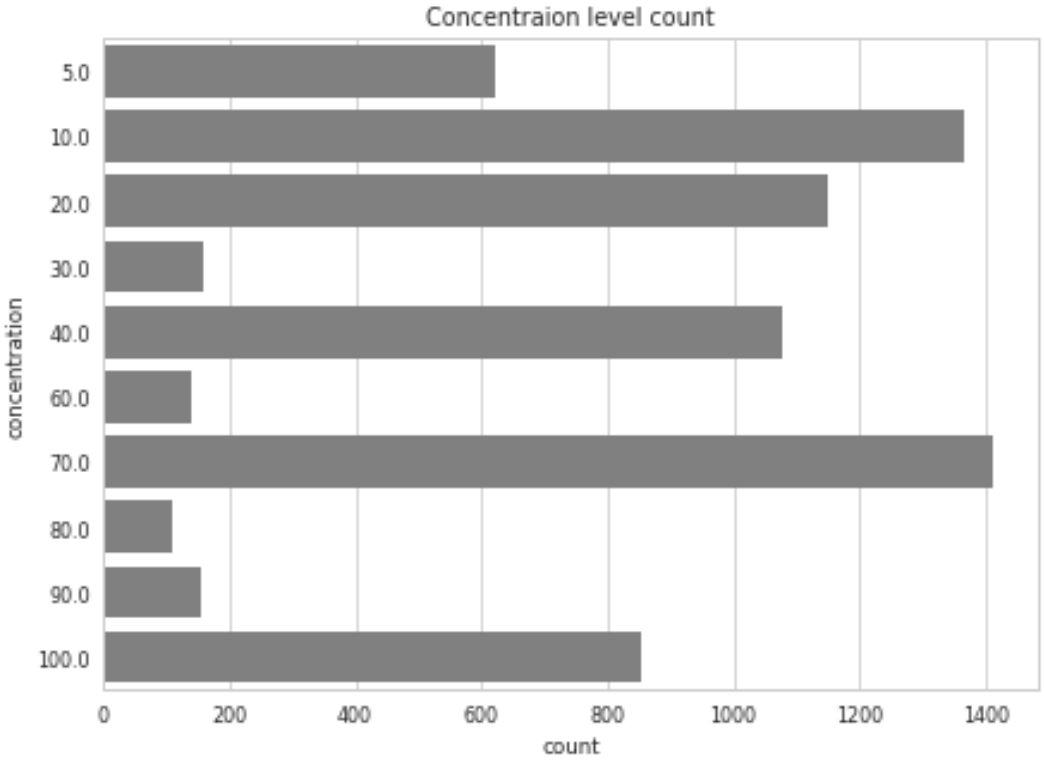

Figure 4.2: Concentration level of compounds

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

The compound '0','1','2' and '3' are hard to acquire therefore the data is very imbalanced. Label '4' which represent the samples that do not belong to '0,'1','2' and '3' are readily available and therefore this label is the majority label.

Data preparation

Spectral data are collected from the spectrometer and stored in a central database. The data do not need extensive data preparation as data are reasonably clean with some exception of missing labels. To prepare the data samples with missing labels are removed, and the data is transformed into a couple of domain-specific data transformation 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 dataset that is fed to the deep learning model. Therefore it is logical that data augmentation is performed in this phase. that various augmentation is performed on data. Above mentioned augmentation namely data warping, SMOTE, and GANs will be implemented in the following subsection.

Data augmentation using data warping

Data warping is one the simplest and easiest way to apply data augmentation. In a popular neural network implementation library Keras, data augmentation for the image such as transformation, scaling, etc. 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 transformation preserve the image label. For example, rotated image data of 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 the 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 the different condition like 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 1.

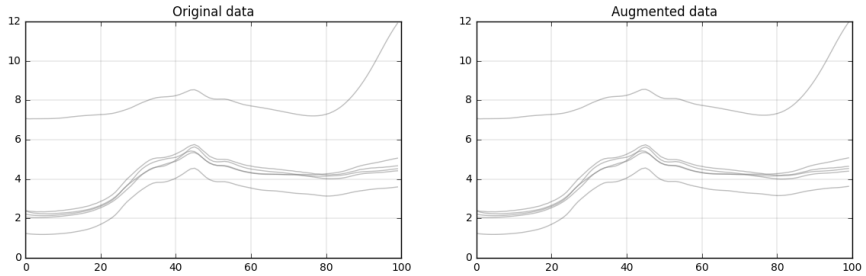

Figure 4.3: Augmentation using data warping

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 with randomly generated data. Depending upon the augmentation the value of noise can be increased or decreased by adjusting the parameters from where the random noise is generated.

Data augmentation using SMOTE

SMOTE was conceived to better handle the imbalance datasets 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 class so that all the class label are represented equally in the final

data set. The detail 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). Figure 4.4 shows the generated synthetic samples from SMOTE.

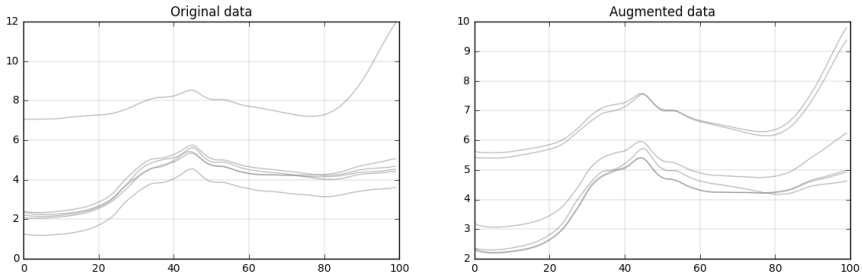

Figure 4.4: Augmentation using data warping

In the figure 4.4 we can see that augmented data have bit more variation than the data augmented through data warping. The new generated spectral data is just an interpolation of new data between existing data. As we can see that the 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 field setting the spectrometer measurement can fluctuates within the range of measurement done in sample. The one drawback here is that unlike image augmentation the label invariance can be detected from simple observation, in this case we cannot easily find out that whether the label has been preserved.

The overall idea of augmentation in SMOTE is quite similar to data warping. The augmented data is simply 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 sample compared to data warping.

Data augmentation using GAN

To implement the GAN laid out in the theoretical review we make two deep learning model. One generator and another discriminator. Then we stack together two model and start the model training process. The architecture of the GAN 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: Augmentation using data warping

Here 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 above 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 discriminatory are just two neural networks.

The first step of the training process is to draw random points from the normal distribution 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 GAN using these random vectors, with the all the label as “real ”. This 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 closely replicated the actual shape of original data.

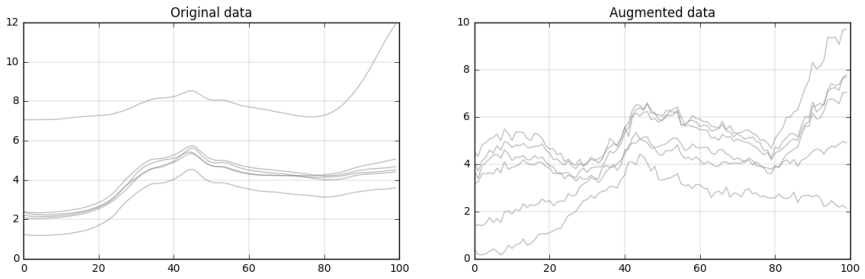

Figure 4.6: Augmentation using data warping

Modeling

In this phase, we build a deep learning model where we test data prepared through different augmentation method for its accuracy, precision, and recall. 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 is various type of neural networks, and in this thesis, we will be working with the kind of network known as the convolutional neural network.

Convolution neural network was chosen primarily because unlike the typical dense layer convolution layers learns local pattern in their input feature space whereas dense layer learns global pattern (Chollet, 2017). Furthermore,

convolution net can learn the spatial hierarchies of pattern these critical characteristics of convolution net makes it translation invariant meaning pattern no matter where it is, convolution net can recognize it (Chollet, 2017). For example, convolution network is quite successful in image classification because it can detect the local pattern along with its spatial hierarchies such as eye, nose, mouth no matter where they are located in the image. Since we are treating the data obtained from spectrometer as a fingerprint of particular chemical compound, identifying local pattern 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 simply means 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 we divide data into a number of small batches. On completion of every batch, the loss is computed, and weights of networks get updated.

Picking the right network architecture is more of an art than a science– (Chollet, 2017). Having said that there are some best practices and general agreed principles. Based on the best practices and generally agreed principles the architecture of network was constructed. For instance the objective function of the model categorical crossentropy as this generally used for multi-class classification problem. Similarly, convolution layer is followed by max-pooling layer which is also common practice in designing constitutional network.

Layer (type)	Output Shape	Param #
reshape_1 (Reshape)	(None, 50, 1)	0
conv1d_1 (Conv1D)	(None, 48, 42)	168
max_pooling1d_1 (MaxPooling1	(None, 24, 42)	0
conv1d_2 (Conv1D)	(None, 22, 55)	6985
max_pooling1d_2 (MaxPooling1	(None, 11, 55)	0
conv1d_3 (Conv1D)	(None, 9, 65)	10790
max_pooling1d_3 (MaxPooling1	(None, 4, 65)	0
flatten_1 (Flatten)	(None, 260)	0
dense_1 (Dense)	(None, 36)	9396
dense_2 (Dense)	(None, 30)	1110
dropout_1 (Dropout)	(None, 30)	0
dense_3 (Dense)	(None, 12)	372
dropout_2 (Dropout)	(None, 12)	0
dense_4 (Dense)	(None, 4)	52

Total params: 28,873
Trainable params: 28,873
Non-trainable params: 0

Figure 4.7: Model Architecture

Chapter 5

Evaluation

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

Base model without augmentation

The first model was built using data without augmenting for the purpose of benchmarking. 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 accuracy of the model is relatively high. However model do not correctly classify minority labels evidently reflected by low F1-score.

Augmentation using data warping

The second model is built using the same network but with augmentation. In this case, we have used simple data warping technique defined in the implementation section. It has the same number of epochs and batch size. In this model trainind 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 decrease in accuracy.

Augmentation using SMOTE

The third model is again built using the same network using SMOTE as augmentation method. SMOTE has been throughly discussed in the literature review and as mentioned previously SMOTE was implimented here using the pythons third party module named 'imblean'. It also has the same number of epochs with same batch size. Here augmentated data is more than 10-times the actual data and using agumented data, training sets labels are balanced. Table 5.3 summarizes the results obtained from the model.

Augmentation using data warping also shows improvement over base model as F1-score reaches 0.80. This model significantly improved the classification

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

performance of both minority and majority class compared to previous model. This is evident by the increase in both F1-score and accuracy.

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 approximated 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 the considerable increase in both precision and recall.

Model Comparison

From the evaluation of the model, we can infer that classification model performed much better when using augmented data. There was significant im-

provement over the base model without using augmentation when compared to model created through augmentated data. If we observe the accuracy alone of all the model relative increase in accuracy is not significant. Under the hodd if we observe precision, recall and F1-score there seem to be considerable improvement. This improvement in F1-score improves the quality of prediction reducing the misclassification rate of infrequent but nevertheless important minority class.

Augmentation using data warping was an improvement over base model even though over-all accuracy declined by three percentage points. There is an improvement of more than 10 points in F1 score compared to the base model. This means that model built using data warping as augmentation strategy is better able to detect minority class than 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 base model and model with data warping augmentation. With SMOTE we can see improvement in accuracy as well as in F1-score. Unlike data augmentation using warping, this model does not sacrifice performance on correctly classifying minority class at the cost of correctly classifying majority class. Overall accuracy is quite high however there is still space for improvement in correctly classifying minority class as F1-score is relatively low compared to accuracy.

The fourth model built using augmented data from GANs generator performed better than all the previous model. There is an improvement in both accuracy and F1-score compared to all the previous model. There is a quite

significant improvement over the base model as there is improvement over the base model. F1-score is around 30 points. Although GANs provide a significant improvement over the base model, we can still see that there is a room for some more improvement. Since GANs itself is a neural network, there is still some possibility to fine tune the network.

To conclude we can see that GANs had a significant improvement in the model especially in detecting minority classes compared to any other model.

Chapter 6

Discussion

We have built and compared various data augmentation strategy using deep learning model. We went through several iteration before coming up with the final deep learning architecture. This chapter discuss possible short coming of the thesis and how this study can be extended in future.

Limitations

One of the critical assumptions in augmentation is that the augmentation process retains the label. Meaning if the spectral data from the certain material is augmented then the augmented data still represent 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 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 not straight way of intuitively check whether augmentation preserves the key characteristics of spectral data.

Similarly, at the implementation phase, we realized that 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. This means that we are unable to use data augmentation for one of the popular 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 it will be quite over egregious assumption to assume that augmentation will preserve the label for a

particular point instead for a class.

Furthermore this study should not be taken as definitive proof that augmentation technique 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. Furthermore, augmentation can be domain specific so 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 test set. These has some limitation. Due to several iteration even though the test was untouched during training period it might have been slightly biased. Similarly, in real life application just the accuracy on test might not be sufficient to evaluate the models factors such as model complexity and time take to train the model needs to be considered too.

Future work

The future work could focus extending the current work by tackling above mention limitation and also experimenting with more augmentation technique that were not part of this study.

There are other various augmentation technique 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 auto encoder and other different variation of GANs itself. These generative models could provide further improvement over the existing model. Similarly there are also different variation of SMOTE which could be evaluated against generative models. This would also be interesting to test with spectral data. We can also create new augmented data by combining existing augmentation method and create 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 by 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 limitation. This opens up new avenue for future work that could improve and extend this studies. Data augmentation offers 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 therefore several augmentation approach needs to be tested in order to come up with the augmentation approach that fits need.

Chapter 7

Conclusions

In this thesis we have brought forward various augmentation framework from simple augmentation that shifts the randomly shifts the data to the one where deep learning model generates the new data. The key findings and the contribution of this paper is discussed below.

Key Findings

Thesis was set out to improve the classification model using data augmentation. After comparing the model 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 bee the collection of data. As demonstrated in the thesis augmentation can help mitigate this issue. From the evaluation of various augmentation we observed that augmentation using the GANs were superior than the augmentation using data warping and SMOTE.

Contributions

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

deep learning model.

Theoretical Contribution

Data augmentation is a contemporary issue in the field of data science. It has been extensively used in multidimensional data, for example, in image detection and computer vision problem. 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, methods like GANs has been recently developed generative models. This is active research topic in the field of deep learning. By applying this new concept into the business case we hope the further the understanding of the generative models and its implication in businesses.

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, nevertheless it provides a sound proof of concept that augmentation can improve the model performance. Thesis offers a way to tackle the problem faced by 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 is 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 researcher or companies interested in data augmentation can also benefit from this as frameworks for augmentation is clearly laid out in the thesis.

Finally

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

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 C.1: Implimentation of GANs

Appendix D

Data generation through GANs

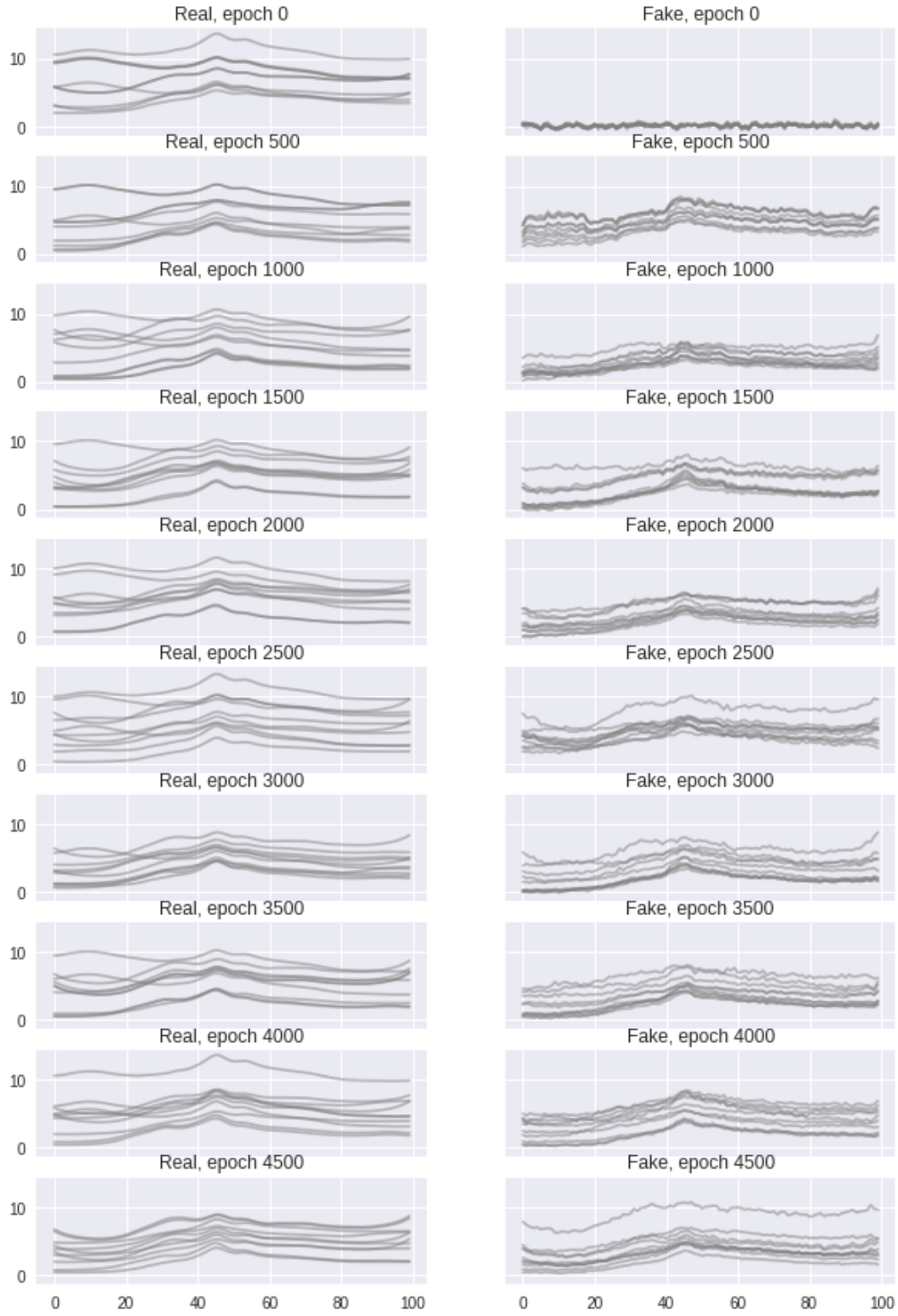

Figure D.1: Data generation process using GANs

Appendix E

Deep learning model without augmentation

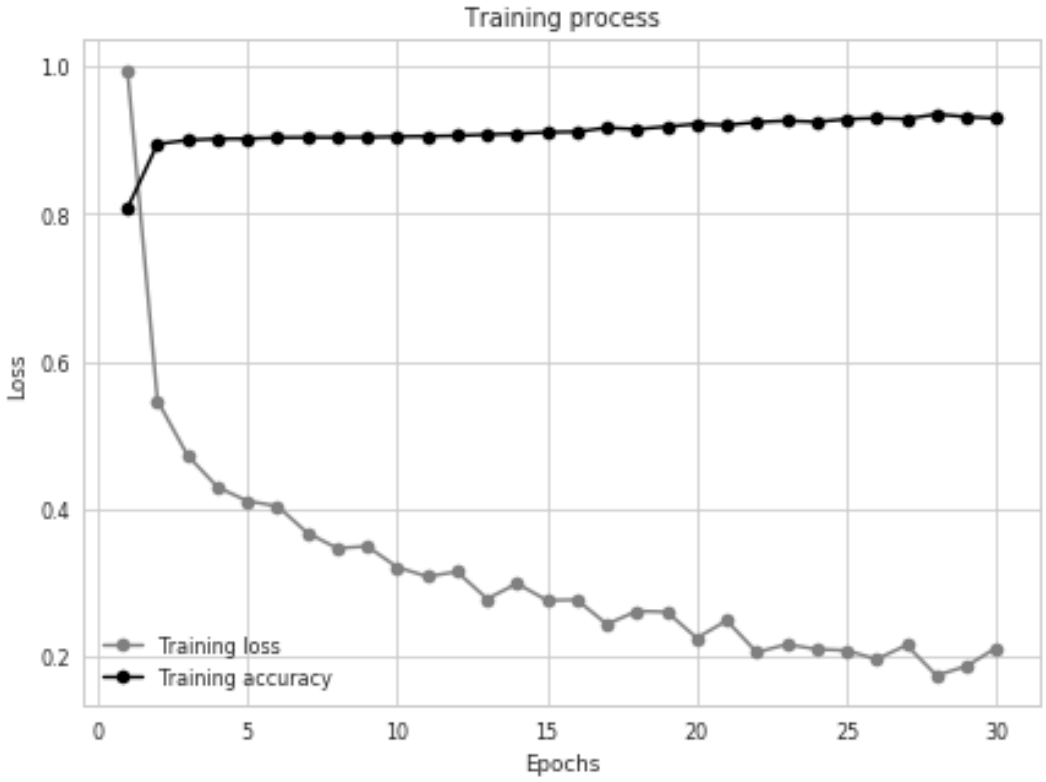

Figure E.1: Deep learning model without augmentation training process

Appendix F

Deep learning model training

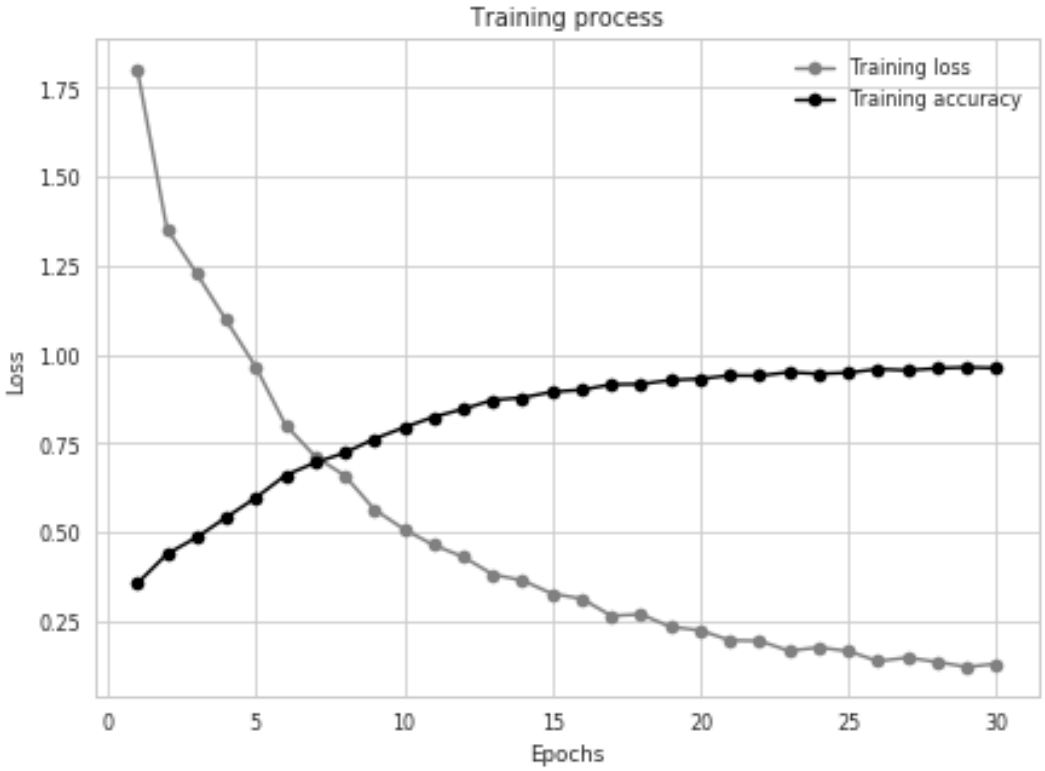

Figure F.1: Training loss and accuracy for model with augmentation usign data-warping

Appendix G

Deep learning model training

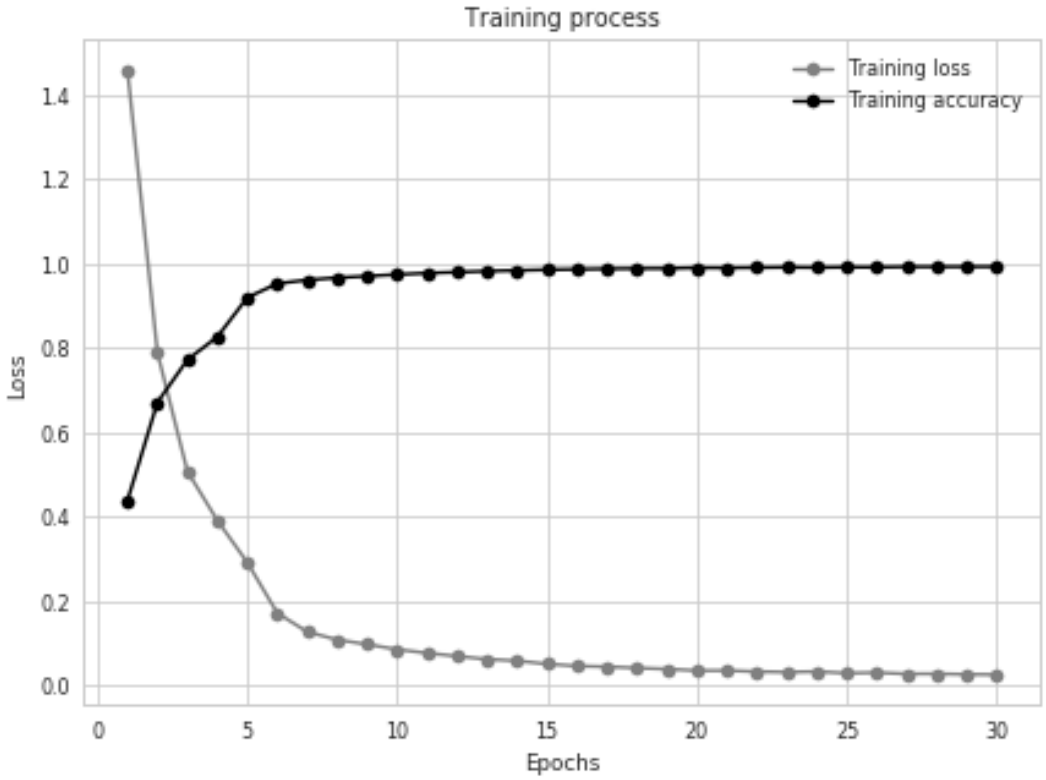

Figure G.1: Training loss and accuracy for model with augmentation usign data-warping

Appendix H

Model misclassification

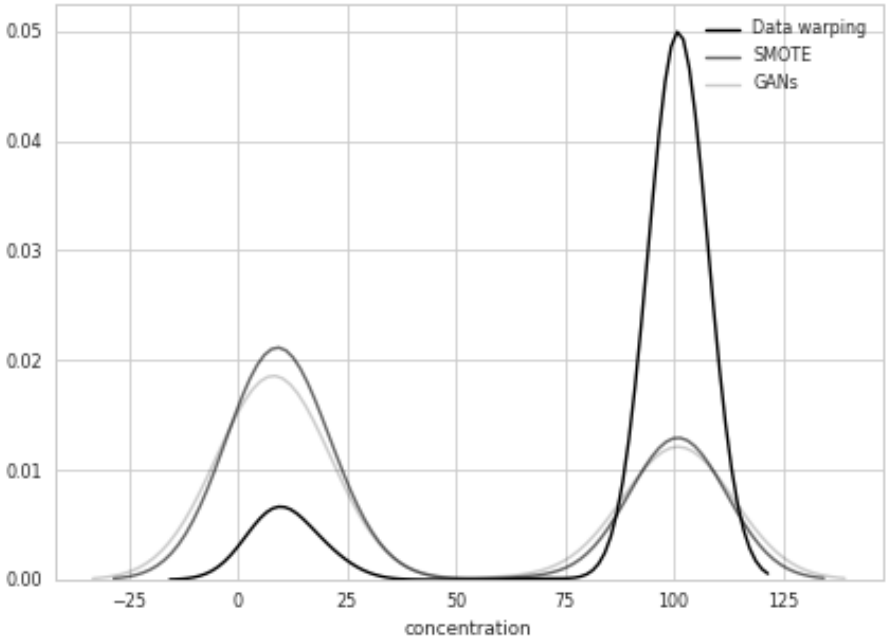

Figure H.1: Density plot of model missclassification at various concentration level

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.