

An Investigation Into Machine Learning Solutions Involving Time Series Across Different Problem Domains

Owen Corrigan, BSc (Hons)

A thesis presented for the degree of PhD

Supervised by:
Prof. Alan F. Smeaton

School of Computing



Dublin City University
January 2018

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Owen Corrigan

Owen Corrigan

59316418

January 2018

Acknowledgements

I would like to thank my supervisor Prof Alan Smeaton and the other member of my supervisory panel Dr. Margaret Farren.

I would also like to thank Dr. Sinéad Smyth from the School of Nursing and Health Science in DCU for her collaboration on the PredictED system.

I would like to acknowledge support from Science Foundation Ireland under grant number 12/RC/2289 (Insight Centre).

Table of Contents

Acknowledgements	i
List of Figures	vi
List of Tables	ix
Abstract	x
1 Problem Statement	1
1.1 Machine Learning Background	4
1.2 Applications of Supervised Machine Learning	6
1.2.1 Email Spam Detection	7
1.2.2 Optical Character Recognition	7
1.2.3 Image classification	8
1.2.4 Medical Diagnostics	9
1.2.5 Image captioning	9
1.2.6 Machine Translation	10
1.2.7 Applications of Machine Learning — Summary	10
1.3 An Overview of Machine Learning Models	11
1.3.1 Linear models	11
1.3.2 Support Vector Machines	13
1.3.3 Trees and Forests	14
1.3.4 Neural Networks	15
1.3.5 Summary of Machine Learning Models	15
1.4 An Overview of Deep Learning	15
1.5 Neural Network models	16
1.5.1 Convolutional Neural Network (CNN)	18
1.5.2 Recurrent Neural Network (RNN)	19
1.5.3 Neural Network Models Summary	19
1.6 Hidden Layers in Deep Neural Networks	19
1.6.1 Dropout Layers	20

1.6.2	Pooling Layers	20
1.6.3	Binarising Layers.	21
1.6.4	Sigmoid Function Activation	21
1.6.5	Rectified Linear Units	22
1.6.6	Scaled Exponential Linear Units	23
1.6.7	Hidden Layers Summary	23
1.7	Case Studies	24
1.8	Methodology	25
1.9	Problem Statement Summary	27
2	Literature Review	29
2.1	Applications of Machine Learning to Learning	29
2.1.1	Student Grade Prediction	32
2.1.2	Student Retention	37
2.2	Applications of Machine Learning in Agriculture	40
2.3	Application of Machine Learning to Environmental Science	45
3	Case Study: Predicting Student Examination Outcomes	49
3.1	Ethical and Privacy Considerations	52
3.2	Remodelling Student Data into Weeks	52
3.3	Feature Extraction — Regular Features	56
3.4	Training Models	59
3.5	The PredictED System	59
3.5.1	Choosing Which Modules to Predict Student Outcomes	60
3.5.2	Year 1 PredictED Implementation	64
3.5.3	Year 2 PredictED Implementation Description	64
3.6	Performance of Predictions	66
3.7	PredictED Challenges	70
3.8	Intervention Evaluation	71
3.9	Deep Learning Approach	73
3.10	Conclusion	75
4	Case Study: Application of Machine Learning to an Agriculture Problem	77
4.1	Problem Description	77
4.2	Description of Data and Target	79
4.3	Data Preparation and Feature Extraction	80
4.3.1	Saturation of Sensors	81
4.3.2	Re-Sampling the Data	81
4.3.3	Feature Extraction	81

4.3.4	Classifier Selection	84
4.3.5	Cross Validation	85
4.3.6	Feature Selection	86
4.3.7	Time to Detection	87
4.3.8	FFT vs Non-FFT features	88
4.3.9	Wide Range Accelerometer vs Low Noise Accelerometer vs. Gyroscope	89
4.3.10	Resampling the data to lower frequencies	91
4.4	Why a normal approach won't work	91
4.5	Time Series Analysis	93
4.6	Conclusion	94
5	Case Study: Predicting Air Pollution Levels from Images using Wear- able Cameras	95
5.1	Problem Statement	95
5.2	Data Preparation and Cleaning	96
5.2.1	Participant Activity	97
5.2.2	Images	100
5.2.3	Properties of Datasets	100
5.2.4	Pollution Levels by participant	103
5.2.5	Pollution Levels by village	107
5.3	Experimental Method	107
5.3.1	Standard Machine Learning Pipeline	109
5.3.2	Training Air Pollution Levels on Images	111
5.3.3	Data Augmentation	115
5.4	Deep Learning Regression	117
5.4.1	Regression using a Shallow Network	117
5.4.2	Regression Using Orginal VGG Model with All Layers Fixed .	118
5.4.3	Regression using a VGG model with the convolutional layers fixed	121
5.4.4	Regression using an end-to-end trained VGG model	122
5.4.5	Regression using ResNet	122
5.4.6	Effectiveness of Data Augmentation	123
5.4.7	Combined Image and Location Model	123
5.4.8	Evaluation	125
5.5	What is the Neural Network Learning?	127
5.6	Conclusion	127
6	Conclusions	128

6.1	Contributions to Case Study Domains	128
6.2	Comparison of Case Studies	130
6.2.1	Target Variable	130
6.2.2	Importance of Precision vs. Recall	131
6.2.3	Real Time vs. Analysis	132
6.2.4	Deployment	133
6.2.5	Case Study Summary	133
6.3	Hypothesis Evaluation	133
6.4	Future Work	134
Appendix 1: Publications		136
Appendix 2: Ethical Considerations for the PredictED Project		137
6.5	Research Ethics Committee	137
6.6	Anonymisation	138
6.7	Student Awareness of Monitoring and Opting into Project	138
6.8	Data Protection	139
6.9	Plain Language Statement	139
Bibliography		142

List of Figures

1.1	Illustration of Least Squares. Image downloaded from https://upload.wikimedia.org/wikipedia/commons/thumb/b/b0/Linear_least_squares_example2.svg/220px-Linear_least_squares_example2.svg.png	12
1.2	Illustration of Support Vector Machine. Image downloaded from https://upload.wikimedia.org/wikipedia/commons/thumb/2/2a/Svm_max_sep_hyperplane_with_margin.png/220px-Svm_max_sep_hyperplane_with_margin.png	13
1.3	Illustration of Convolutional Neural Network. Image downloaded from https://ujwlkarn.files.wordpress.com/2016/08/screen-shot-2016-08-07-at-4-59-29-pm.png	17
1.4	Illustration of a Recurrent Neural Network. Image downloaded from http://d3kbpzbmcyntmx.cloudfront.net/wp-content/uploads/2015/09/rnn.jpg	18
1.5	Illustration of Max Pooling. Image downloaded from https://upload.wikimedia.org/wikipedia/commons/e/e9/Max_pooling.png	21
1.6	Illustration of ReLu. Image downloaded from https://i.stack.imgur.com/8CGIM.png	22
1.7	Illustration of SeLu. Image downloaded from https://cdn-images-1.medium.com/max/800/1*m0e8lZU_Zrkh4ESfQkY2Pw.png	23
3.1	Activity levels for module CA103 (Computer Systems Hardware) over a 5-year period.	51
3.2	Activity levels for module HR101 (Psychology in Organisations) over a 5-year period.	51
3.3	Test / train split for CA103 in week 6.	54
3.4	Test / train split for CA103 in week 11.	54
3.5	Classifier Scores	60
3.6	Example email sent to students.	65
3.7	Example email sent to students.	66

3.8	Students' view of web application.	67
3.9	Prediction accuracy for SS103 (Physiology for Health Sciences) by week.	68
3.10	Prediction accuracy for MS136 (Mathematics for Economics and Business) by week.	69
3.11	F1 scores by week. Each coloured line represents a different module.	70
3.12	CAO Points for participants and non-participants	72
3.13	Leaving Certificate Mathematics for participants and non-participants.	72
3.14	Illustration of architecture of RNN applied to predicting students grade.	74
4.1	Example of Spectrogram from a suckling calf	82
4.2	Example of Spectrogram from non-suckling calf	83
4.3	Suckling Model Comparison	85
4.4	ROC feature selection	86
4.5	thresholds	87
4.6	time to detection	88
4.7	FFT comparison	89
4.8	Sensor Combinations for calf suckling	90
4.9	Downsampling Results	92
5.1	Participants Per Day	98
5.2	Participant Count by Week	99
5.3	Participant Count by Month	99
5.4	Example image taken from wearable camera	100
5.5	Histogram of Particle Counts	101
5.6	Normalized Histogram of Lower 95th Percentile of Particles	102
5.7	Histogram of Particles distributed into 1,000 bins	102
5.8	Mean Particle Counts per Participant	104
5.9	Particle Counts per Participant Boxplot	105
5.10	Particles per Participant Boxplot – Averaged Over Session	106
5.11	Distribution of Air Particle Values per Village	108
5.12	Standard Machine Learning Results Boxplot	112
5.13	Hidden Layers in a Neural Network	113
5.14	Example of a Convolutional Neural Network	114
5.15	Architecture of Shallow Network	118
5.16	Shallow Neural Network Learning Curve	119
5.17	VGG architecture	119

5.18 VGG with All Layers Fixed	120
5.19 VGG Fixed Convolutions Learning Curve	121
5.20 VGG Adaptable Model Learning Curve	122
5.21 Resnet Architecture	124
5.22 ResNet Model Learning Curve	125
5.23 Combined Network Architecture	126

List of Tables

2.1	Summary of applications of Machine Learning to Learning.	39
2.2	Summary of applications of Machine Learning to agriculture.	44
2.3	Summary of applications of Machine Learning to Environmental Sensing	48
3.1	Sample table of features with 10 students. The table contains two weeks of features (weeks 7 and 8), and four features from each week.	55
3.2	Sample of 10 log entries from Moodle (IP address has been redacted).	58
3.3	Performance of Classifiers on Final Week of Semester	59
3.4	Performance in end-of-module examination showing number of eligible students for module (Regs), and comparing average non-participants' exam mark (Opt-out) vs. participants' average exam mark (Opt-in) and significance of difference between these.	63
3.5	Regression Results	75
4.1	Performance of Machine Learning algorithm on suckling detection task.	84
5.1	Distribution of pollution particle counts	103
5.2	AQI categories and health impacts	109
5.3	Results when using the Standard Machine Learning Pipeline	110
5.4	Data Augmentation Examples	116
5.5	Image Pollution Prediction Results	126

Abstract

An Investigation Into Machine Learning Solutions Involving Time Series Across Different Problem Domains

Owen Corrigan

In this thesis we will examine architectures and models for machine learning in three problem domains each of which are based around the use of time series data in time series applications. We set out to examine whether the architecture and model solutions in different problem domains will converge when optimised towards a similar solution or not.

Stated clearly, our central research question is ***“That problem-solving in diverse problem domains using Machine Learning applied to time series data requires diverse models in order to achieve the best performance”***.

To investigate this research hypothesis we use a case study methodology. We will investigate three separate and diverse problem domains, and compare their results and best solutions.

The first problem domain is in the field of educational analytics, the second is in the field of agri-analytics and the third is in the field of environmental science.

Chapter 1

Problem Statement

Supervised Machine Learning is the study of automating the creation of computer programs by developing a program which in turn learns how to process new information by learning from past examples. This is a broad goal, and there is an incredibly large array of problem domains that this can be applied to.

Of particular importance and interest to the work in this thesis are those problems which can be solved using Supervised Machine Learning and which are based on the use of time series data. In a typical Machine Learning architecture each data point is considered to be independent of all the other data points. However, this is an invalid assumption when dealing with time series data, as each data point has a temporal dependence on points which came before it, and even after it.

Previously, the state of the art in each individual problem domain was achieved using a similarly broad array of techniques. However, in recent years we have seen the solutions to the vast majority of problem domains have converged to using a Deep Neural Network (DNN). For an example of some of these domains, see Section 1.2. Not only has this DNN replaced the models which were previously used, but has even automated much of what is easily the most cumbersome task in supervised Machine Learning, that of feature engineering.

Feature engineering is an important process in Machine Learning and involves converting raw data into representations which are more conducive to getting an algorithm to correctly predict an output. For example, if our goal was to predict whether a given image was of a house or not, we might develop features which can estimate the number of windows, detect if a door is present, if a roof is visible

or not, and so on. We could then train the Machine Learning pipeline directly on these features. This is likely to give better results than training a classifier on the raw pixels of the image directly. The feature engineering process can be time-consuming and requires expert knowledge, and so automating this process is a major advantage of the deep learning approach.

One might wonder whether the trend of the best approach to solving problems trending towards a small set of deep learning solutions will continue. To support this view, Neural Networks have begun to emerge as the “winners” in many benchmark tasks which lie outside the field of Supervised Machine Learning where they originated, such as Reinforcement Learning (Alpha go) and Unsupervised Learning.

In this thesis, we argue that the opposite is true, that after a period of convergence, the best-performing models for solving problems are actually starting to diverge into different solutions for each problem domain. In particular, a diverse variety of Neural Network models have emerged, and we argue that this will continue. Some of these models will be discussed in Section 1.5. In addition to this, we argue that those methods not based on non-deep learning still have niche problems for which they offer the best solutions, given certain constraints. In particular, this is the case in Chapter 4 which looks at problems in the field of agri-analytics.

Stated clearly, our hypothesis is this: ***“That problem-solving in diverse problem domains using Machine Learning applied to time series data requires diverse models in order to achieve the best performance”***. In essence, we are arguing that the choice of model is important, and where neural networks are used the network design is important. As much as possible we should use prior information about the model to influence the design of the model (LeCun et al., 1989). This thesis will verify this in the context of three novel case studies. In each of these we have tried a variety of solutions, outlined later in Section 1.7 and described more fully in the rest of the thesis.

We have found that the best performing solutions for each of these case studies require different Machine learning approaches, and that the best solutions vary depending on the properties of the problem domain. This has allowed us to build up an intuition of which models to use in various situations, which we have included in our thesis Conclusions chapter, which is Chapter 6.

In each of these case studies we have identified several contributions to the state

of the art of their domain. These are stated in Section 3.10, 4.6 and 5.6 summarised in Section 6.1.

We argue that new and emerging problem domains in future will continue to require novel models and that through this, human supervision and guidance will remain important. In other words, the goal of a truly human-independent, plug-and-play style model for Machine learning is intractable, and that human intuition will remain essential, albeit at a higher level than before the advent of automated feature extraction through deep learning.

In this hypothesis, we have used of the word “diverse” in two different contexts, and we will give further consideration to these meanings. The first is the diversity of problem domains. We will examine a number of example problem domains in Section 1.2. A problem domain can be categorised along many axes. Here we have identified 4 of the main ones, including.

- Target Variable. Whether the target variable is continuous (e.g in the case of regression) or class labels (classification).
- Importance of Precision vs. Recall. In some problems there are high costs associated with missing a positive case, for example in the detection of diseases. In other problems there are high costs associated with incorrectly identifying a false positive, for example finger print matching. Depending on the whether high recall or high precision is important, the user of the classification may choose a different algorithm.
- Real Time vs. Analysis. Whether inference must be done in real time, or is there sufficient time between predictions to allow more processing of complex features.
- Deployment. What hardware will the resulting algorithm be deployed on. Is it possible for the parameters to be updated on this hardware. Will this be real-time?

The second use of diversity has to do with diverse solutions.

- Algorithm Choice. Which class of algorithm is used. For example random forests, neural networks, linear models or support vector machines.
- Inductive bias. What set of assumptions about the data is taken by the algorithm.

- Hyper parameters. What hyper parameters are used in the
- Architecture of Neurons (in the case of Neural Networks). What is the composition of neural networks. For example, are convolutions used? Recurrent Layers? Attention mechanisms? (See Section 1.3.4).

We argue that the solutions to problem domains which have similar characteristics will potentially also share characteristics. Conversely, the solutions to problem domains which have different characteristics will also have different characteristics.

This hypothesis can be compared to the famous “No Free Lunch” theorem. According to this theorem, Wolpert and Macready (1997) state that any algorithm which has a better performance over one class of algorithms will be offset by performance over another set of problems. Hence there is no universal ‘best classifier’ algorithm, and that good performance of a classifier is restricted to a number of domains. This hypothesis sets out to validate this theorem across a number of problem domains, and to establish that the chosen problem domains might be defined as ‘a set of problems’ under the above definition.

We will begin our discussion with some background into Machine Learning.

1.1 Machine Learning Background

We will now give a broad overview of the field of Machine Learning, with a particular focus on the topics that are related to the hypothesis question posed above. As a full overview of this topic is not within the scope of this thesis, the reader is advised to take a look at any one of the following books, each of which contains a broad coverage survey of Machine Learning throughout the years (Friedman et al., 2001; Bishop, 2006; Kelleher et al., 2015; Goodfellow et al., 2016).

As was mentioned previously, Supervised Machine Learning is the study of automating the creation of computer programs, by developing a program which in turn learns the desired result or outcome from a set of examples. In order to understand what this means, we will first examine the process of creating a computer program without using such learning techniques. Many tasks in the real world can be solved using explicit rules which have been programmed in, for example a calculator program. However there are still many other tasks which are difficult or impossible to do with this stepwise approach, particularly those for

which it is difficult to create clear rules or where rules are vaguely specified. For example, think of the difficulty of writing a program to tell what number a handwritten digit is as can be seen in the MNIST dataset LeCun et al. (1995). We could develop some rules for this based on the number of pixels switched on, or we could search for edges and curves corresponding to the different shapes of the digits. However this would be both difficult and time consuming. Furthermore, we could not guarantee that such a program consisting of rules for different digits would run correctly for new inputs, for example it might confuse the digits “5” and “6”. Instead, we can come up with a large dataset of examples of hand written numbers, and then design a program to discover these rules for determining which digit is handwritten, automatically.

It has only been possible to do this in recent years with the advent of the following:

- More powerful computer processing capability;
- Advances in algorithm development;
- Larger data sets which can be used for training models.

There are many different ways that these learnable parameters can be determined. The large choice of configurations available makes choosing the optimal combination a difficult task, and we will explore this in a lot of the detail in this thesis.

There is also a wide variety of problems that we can solve with these methods, which we will discuss later in Section 1.2. For the moment, we will discuss the evaluation of such applications. The most common way of evaluating what is the state-of-the-art for a given task is to release or use a public data set of examples to learn from. Often, there will be competitions or a public benchmarking to see who and which technique can get the best results on an unseen data set, using this released training set. Well known examples of this include MNIST (LeCun et al., 1998) for handwritten digits, ImageNet (Deng et al., 2009) for image classification, COCO (Lin et al., 2014) for object segmentation, and AudioSet (Gemmeke et al., 2017) for audio classification. This approach is considered to be the gold standard for assessing progress in a particular task based on Machine Learning.

The technique of having separate data sets on which to test (called *the test set*) and to train on (called *the training set*) mentioned above, is called cross-validation. This is important because we want to develop a program which will not just be

able to perform well on examples used in the specific data set, but one which will generalise well to any potential examples which are shown to it. Often we will keep a third data set for evaluating hyper-parameters (called *the validation set*). An algorithm which learns rules which work well on a testing data set, but will not perform well on unseen examples is said to be *Over Fitting* to the data. over-fitting can creep into a system in subtle ways, and much of the individual chapters in this thesis are discussions of how we implemented our cross-validations schemes to avoid this.

Hyper parameters are parameters which affect the the learning process, but unlike learnable parameters, do not adapt to any particular set of data. For example, one commonly used hyper parameter is the learning rate. A high learning rate will result in a faster convergence towards an optimal configuration, however the trade-off is that there is a greater risk of not finding an optimal training error. Hyper parameter tuning is currently an area that requires much experimentation and intuition in order to set correctly, and there is a lot of work currently on automating the discovery of the best values for hyper parameters.

So far we have discussed the thesis question of different types of Machine Learning models, without much discussion about the models themselves. In Section 1.3 we will discuss a number of models in greater detail. We will now discuss in more detail several notable applications of Machine Learning in order to understand why their best solutions might require different models.

1.2 Applications of Supervised Machine Learning

As we can see from the definition of supervised Machine Learning earlier, it is tool with a broad range of applications. The question that this thesis poses is to examine the relationship between problems in different fields and their best performing solutions. For context, here we present an overview of a selection of applications of Machine Learning. We have selected these for their historical importance, but also to emphasise the broad nature of the problems that Machine Learning can be applied to.

1.2.1 Email Spam Detection

One of the earliest success stories with Machine Learning is the application to finding spam emails. In the early days of email, unsolicited emails were a staple of owning an email address. However, by using Supervised Machine Learning to detect emails with certain characteristics and signatures, spam emails could be filtered out with a high degree of confidence. The process of detecting spam emails first involves extracting features from their text. Some of these methods include

- Extracting Term Frequency-Inverse Document Frequency vectors. The insight here is that any text document can be defined by its most important words. Words that are common in the document, and uncommon in all other documents in a collection of texts are likely to be good words for defining the content in the document in question.
- Frequencies of bi-grams and n-grams. This is a simple count of how often certain letter pairs (bi-grams) or longer sequences (n-grams) of letters appear together.

A training set for the task of automatic spam detection can be found in the Spam Base dataset (Hopkins et al., 1999; Lichman, 2013). Features were extracted and then used as inputs to a model such as a Naive Bayes classifier, which then returned an estimate of how likely a given email was to be spam. This process is typical of a Machine Learning pipeline involving text.

However, recently, Machine Learning algorithms using a Recurrent Neural Network (RNN) (see later in Section 1.5) have become the state of the art in spam email detection. RNNs also have the advantage that they require less feature engineering than older natural language processing pipelines. For example, they can be trained character by character or word by word, without extracting counts of words.

1.2.2 Optical Character Recognition

Another early application of Machine Learning is in the task of Optical Character Recognition, automatically recognising text from scanned images of documents. This had a very obvious initial practical application for the automated sorting of

post or mail, where the front of printed letter envelopes could be read and digitised to improve sorting of envelopes. The goal of this task is to be shown an image of some text (in the case of the post office application, an address) and to convert it to machine readable text. As we mentioned earlier, a famous dataset for this task is the MNIST dataset. In this task, we take an image and attempt to read the characters in it. This is a constrained version of the image classification task, as the images will belong to a much smaller subset of images.

The state of the art solution to this problem used to be using a rather manual programming process of finding edges on individual letters and characterising letters based on shapes and edges. Now that has changed completely and Machine Learning based approaches far out-perform rule-based. This was one of the first problems that neural networks were shown to have state-of-the-art performance in.

1.2.3 Image classification

In this task, an image is known to belong to one of several pre-defined classes usually based on image content, so example classes would be those images that contain cats, or dogs, or cars, or whatever else. Our goal is to determine which class or classes, an input image belongs to. We train our model with hundreds or thousands of examples of each class, both positive and negative examples. The output of our model is a vector with a probability of the given image being a member of each class. As this is quite a complex task, we must train our models with a huge corpus of images, for example those provided by ImageNet (Deng et al., 2009).

Again, the state of the art in this task until a few years ago used to involve image pre-processing and a certain level of manually-programmed extraction of low level features like colours and shapes and lines, followed by a classifier, commonly a Support Vector Machine (SVM) being used to tie it all together. However, Neural Networks have now been firmly established as leaders in this task and in the ImageNet task, we can get error rates down to 3% or even lower.

More recent developments in Machine Learning have seen an emergence of new neural networks models. Some recent successful techniques include inception modules (Szegedy et al., 2015), residual blocks (He et al., 2016) and separable convolutions (Chollet, 2016). These developments have shown that to get state-of-the-art results in tasks, some engineering of the neural network architec-

ture is necessary.

1.2.4 Medical Diagnostics

This application area covers a broad range of problems, where the goal is to diagnose a patient with some illness or condition, given some evidence. For example, given an MMR scan of a patient who is suspected to have cancer, we could diagnose the malignancy of the cancer. Doctors must do this manually now, however this is an expensive service and so is a natural area for supervised Machine Learning where there are lots of example data sets on which Machine Learning approaches can be trained. For a broad overview of this topic, see Litjens et al. (2017).

In Vansteenkiste (2017) they use a three dimensional Convolutional Neural Network to analyse the voxels of a cancer cell. This problem is similar to the image classification example mentioned earlier, except it takes into account multiple inputs depending on the problem, and the output is often a measurement of how serious a disease is present rather than a yes-no decision on whether an image belongs to a class or not.

A non-technical difference between these two topics is that the cost of a single wrong mis-classification can be extremely high, and so even a small increase in accuracy can be meaningful.

1.2.5 Image captioning

In this problem area we are given an image, and our goal is to output a description of the image as a plain English sentence (s), not just as a set of words or tags. We mention this problem, as it is an interesting combination of the text and image classification problems that we have seen above. The COCO dataset (Lin et al., 2014) mentioned above can be used to benchmark progress in this task. One issue with this task is that there can be many correct answers or captions/sentences describing what is in a photograph or video clip, and so it can be difficult to evaluate whether a generated solution is correct or not.

The COCO dataset above evaluates solutions using the BLEU metric, which is a measure of the distance between a correct sentence and sentence output by an algorithm (Papineni et al., 2002). The current state-of-the-art solution in this prob-

lem is to use a combination of RNNs and CNNs (Mao et al., 2014).

1.2.6 Machine Translation

The goal of Machine Translation is to take an input sentence from one natural language and to output it into another natural language, like from English to French, for example. To do this, Machine Learning models are trained on a corpus of translated sentences, such as that available in the Europarl data set (Koehn, 2005).

The first attempts at Machine Translation involved linguists creating rule based systems. Starting in the 1980's, statistical attempts outperformed these linguistic approaches, leading to the famous quote "every time I fire a linguist, my accuracy goes up" attributed to Frederick Jelinek. Many of these statistical techniques were based on Hidden Markov Models (HMM's), for example see Vogel et al. (1996). Recently, sequence-to-sequence models based on a combination of RNNs and Encodings have out-performed even these HMM based models (Cho et al., 2014).

The development of the field of Machine Translation follows closely to our conceptual model. First, statistical models outperformed human attempts to create rule based systems. These models themselves were then out-performed by deep learning models with models adapted for the specific problem domain.

1.2.7 Applications of Machine Learning — Summary

We have included several examples of applications of Machine Learning in the sub-sections above. We have seen how up until recently many of the solutions to these problems are vastly different in approach, how recently deep learning approaches have unified many of the approaches, and finally how divergent solutions have begun to emerge once again. This narrative appears to back up our research question. However, this is simply a post-hoc analysis, and it would be a mistake simply take this at face value without asking for new evidence, particularly given the nature of the topic. In this thesis we provide further evidence for this hypothesis by examining three new case studies, and examine whether their solutions diverge from each other or are similar. These projects have been chosen in order to be as diverse from each other as possible, although each has a time-series element to the data used. In the following chapters we will examine

the resulting Machine Learning models which gave the best results and examine the relationship between them.

This section has examined several applications of Machine Learning, and given us more context into what problem domains it can be applied to. Now that we have discussed the basics of Machine Learning and some applications, we can now begin to discuss some factors that go into the design of a Machine Learning pipeline.

1.3 An Overview of Machine Learning Models

The goal of Machine Learning can largely be viewed as inferring a function given some examples of inputs and outputs. All models in Machine Learning are an attempt to learn this function, under different constraints. As this thesis is about the relationship between problem domains in Machine Learning and the resulting models, it is necessary to discuss some of these models. We will restrict ourselves here to a short informal discussion of their properties. We will also limit our discussion to the models which were relevant to the work done in this thesis.

1.3.1 Linear models

Linear models are the simplest and oldest of all the models in Machine Learning. Estimation using a linear model works by simply multiplying each input feature by a weight and summing the results together. During training, these weights can be found by iteratively applying the gradient descent algorithm to try different weights until a sufficiently small error is found. Gradient descent requires an objective function to minimise, the most common of which is the least squares objective, which minimises the difference between the square of the estimated value and the true value. This strongly penalises estimations which are wrong by a large amount. It is illustrated in Figure 1.1.

Linear models can over-fit, especially when we do not have enough data or when we have a lot of correlated features. In order to combat this we can add a term to the objective function to penalise large weights. Some options for this penalty term include the sum of all the weights (such as Lasso Regression by Tibshirani, 1996), the sum of all the weights squared (Ridge regression by Hoerl and Kennard, 1970) or some combination of the two (Elasticnet Regression by Zou and Hastie,

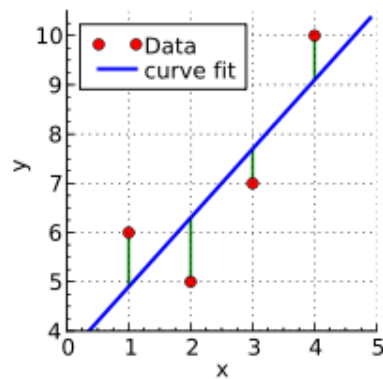


Figure 1.1: Illustration of Least Squares. Image downloaded from https://upload.wikimedia.org/wikipedia/commons/thumb/b/b0/Linear_least_squares_example2.svg/220px-Linear_least_squares_example2.svg.png

2005).

One interesting aspect of linear models is that if we assume that the noise is distributed according to a normal curve, we can prove that a linear model with a least squares objective function is the optimal classifier. However, we found in the applications that we study in this theses that the noise is far from distributed normally.

Linear models tend to perform well on problems where the data is made up of a small number of features, where the features are not correlated with each other, and are each correlated with the target variable. As the model is quick to train, we often use this as a first pass or as a baseline model, against which we evaluate other, more complex models.

So far we have discussed linear models in the context of regression, where the target variable is a number. Such models are useful for problems where we want to predict a number, for example in Chapter 5 we want to predict how many particles of pollution are in the air. The other type of target variable is a categorical variable, where the target is a 0 or a 1. For example, in Chapter 4 our output is categorical, either the calf is suckling from its mother, or it is not. Our ideal output in this case is a variable whose value is a probability which is close to 0 if the outcome is unlikely, and close to 1 if the outcome is likely. We can adapt a linear model to handle this case by putting the output of the function into a sigmoid function, which converts large negative values to be close to 0, large positive values to be close to 1, and values which are close to zero to be close to 0.5. This function is defined as follows

$$S(x) = \frac{1}{1 + e^{-x}}$$

The resulting algorithm is called a logistic regression (even though it is not really a regression, but a classification). We can extend this to multiple categories by using a softmax function which is an extension of the sigmoid to multiple variables. When making a prediction, we choose the class with the highest probability.

1.3.2 Support Vector Machines

In contrast to Linear Models, where it makes most sense to talk about them in the context of regression, Support Vector Machines (Cortes and Vapnik, 1995) make most sense to talk about in the context of Classification. Support Vector Machines work by creating a line or boundary to divide the feature space into positive examples and negative examples. It chooses this line by maximising the “margin” between the two spaces. What this means is that examples which are correctly classified do not count towards the penalty function, however those that do are penalised by the distance to the line. This is illustrated in Figure 1.2.

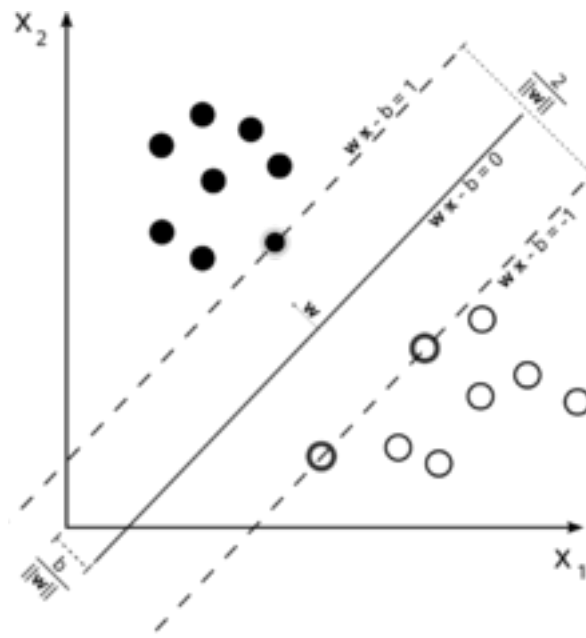


Figure 1.2: Illustration of Support Vector Machine. Image downloaded from https://upload.wikimedia.org/wikipedia/commons/thumb/2/2a/Svm_max_sep_hyperplane_with_margin.png/220px-Svm_max_sep_hyperplane_with_margin.png

The dividing line is linear in this example, however we can adapt this to be a

non-linear function. Because of the way the objective function for an SVM is formulated (using the kernel trick) this can still be learned quickly if we choose a non-linear function with certain properties (preserves the dot product). Examples of these non-linear functions include linear functions, radial functions and Gaussian functions. This is as opposed to a linear model, where to expand it into a non-linear model would involve expanding the entire set of features using one of these functions. Just as Linear Models can be adapted to classification tasks, Support Vector Machines can be adapted to regression tasks.

We find that Support Vector Machines can give good results, and were often state-of-the-art at many of the tasks discussed in Section 1.2 before they were overtaken by the performance of deep learning methods. However, they do not scale well to very large data sets, and can take a prohibitively long time to train, particularly when using certain choices of kernel.

1.3.3 Trees and Forests

Tree models work by directly creating rules, similar to a tree of if-else statements. It does this by examining each of the variables, choosing the one which has the highest *entropy* relative to the target variable, and splitting the data set into two groups based on this. It continues to split these groups into smaller and smaller units until each example has been classified correctly, or until a maximum depth has been reached. This maximum depth is set as a hyper-parameter of the model. The resulting model is a tree of if-else statements which when followed down to a leaf node, will tell which class the input data should belong to.

One downside of tree models is that they do not rely on simple linear algebra operations, unlike the previous two models, and so they cannot speed up using optimised linear algebra libraries. Another downside is that they have a tendency to over-fit to their training data, due to the large number of effective parameters. Finally, they are very unstable, where small changes in the dataset can have a large change in the resulting tree that is created. This is also known as having a high *variance*. However if many trees are aggregated into a forest model they can perform very well.

By combining tree models with bootstrap aggregation we obtain what is called a forest model, which can give good results and is robust to noise. What this means is that we train hundreds of individual decision trees, each tree then “votes” on which class the example should belong to. We will argue in Chapter 4 that random

forests can out-perform deep neural networks under certain conditions.

1.3.4 Neural Networks

Neural networks are models composed of layers of Linear Models. They are inspired by, and loosely resemble, the connections between neurons of the human brain. Despite their recent resurgence, they are among the oldest proposed methods of Machine Learning. In each layer, we train a Linear Model where the inputs to the model come from the previous layer, and the outputs will be fed to the next layer. The layers which are between the input layer and the output layer are referred to as hidden layers. In between each layer is a non-linear activation function, which increases the flexibility of the functions that can be learned. An example of such a non-linear activation function is the sigmoid function, discussed above. When training a neural network, errors are propagated through the network using the back propagation algorithm.

Neural networks are interesting because they represent a model which can learn a non-linear function in a reasonable amount of time. The complexity of the function learned is restrained only by the capability of the architecture. Hence by adding more layers, we can approximate an arbitrarily complex function.

1.3.5 Summary of Machine Learning Models

Despite the growth of deep models, non-deep-learning methods still remain relevant as we will discuss in Chapter 6. In fact, we find that under certain constraints, random forests out-perform deep learning methods (See Chapter 4). We will next discuss deep learning methods in more detail, and give some intuition as to why they have been so successful in comparison to other forms of Machine Learning.

1.4 An Overview of Deep Learning

Deep learning is an approach to Machine Learning which involves using neural networks with many layers as the architecture of the resulting classifier. As neural networks are very configurable in comparison to other models, and we can adapt them in different ways to the task they are trying to perform. Recently, deep learning approaches have dominated benchmarks across a wide variety of

problem domains. There are a number of factors which explain why deep neural networks have had such an improvement in performance.

- Neural network models can be trained quickly using the hardware capabilities of Graphics Processing Units (GPU's). The optimization techniques that deep learning models use to train can be implemented using the linear algebra primitives that are implemented directly in hardware on GPU's. GPU's are devices designed for graphics-intensive applications like computer gaming but have recently been used specifically to implement these instructions, which means they can parallelise their operations many times faster than a general purpose CPU can. This can speed up training times by a factor of up to 50 times, which in turn has allowed experimentation with larger models and larger data sets.
- Models developed using deep learning are trained end-to-end. Previously it was thought it would only be computationally feasible to train models layer by layer, individually. Using back propagation, it is possible to update the weights for any or all layers of the neural network in one pass.
- Careful random initialisation of weights. It is very important that the weights are initialised to be a random orthogonal matrix, as otherwise the weights may lead the training algorithm to get stuck in local minima (Saxe et al., 2013).

In the following two sections we will examine in more detail, the choices available to configure neural networks. We will see that there is a large amount of ways that neural networks can be configured into different models. We argue that the choice makes our hypothesis, that models will diverge to be different depending on the problem domain, stronger.

1.5 Neural Network models

As we have hinted in the previous section, Neural Networks are an incredibly versatile model for Machine Learning. Since they are layered, it is easy to build complex models out of them by combining their layers in novel ways. It is also easy to plug in novel hidden layers and activation functions, many of which we will discuss in Section 1.6. For example, by limiting the connections between layers to be those which are in a *receptive field* of the previous layer, we come up with a network

which can handle image classification accurately. This called a Convolutional Neural Network (CNN) and is illustrated in Figure 1.3.

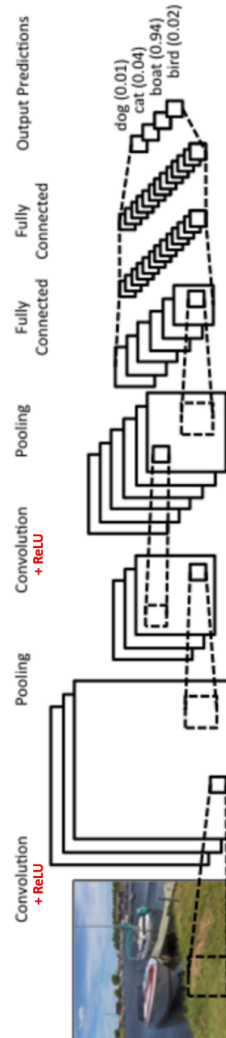


Figure 1.3: Illustration of Convolutional Neural Network. Image downloaded from <https://ujwlkarn.files.wordpress.com/2016/08/screen-shot-2016-08-07-at-4-59-29-pm.png>

Another example of the flexibility of neural network models is that by allowing the output of one layer to be fed back into itself, we invent an architecture which excels at handling sequences of data, called a recurrent neural network (RNN). This is illustrated in Figure 1.4.

There are many such models that can be devised as a result of the flexibility in neural network architecture design, and each will have different properties which make them suitable for different kinds of problems in different domains. We will now examine these two configurations of Deep Learning models, respectively.

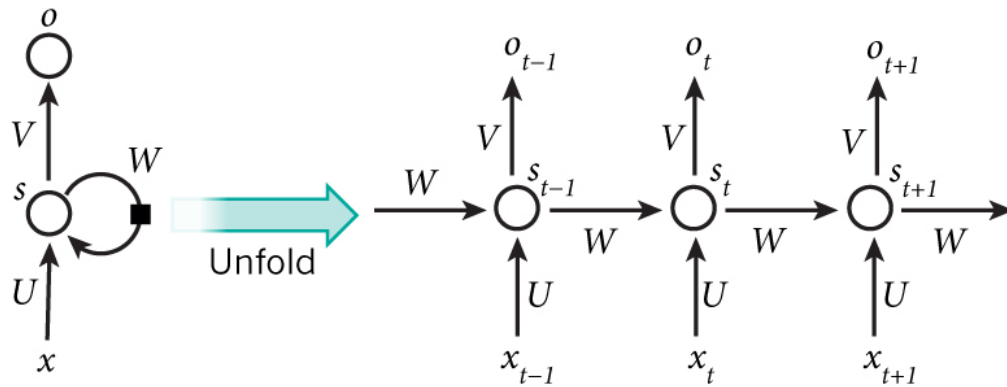


Figure 1.4: Illustration of a Recurrent Neural Network. Image downloaded from [http : / / d3kbpzbmcyymx.cloudfront.net/wp-content/uploads/2015/09/rnn.jpg](http://d3kbpzbmcyymx.cloudfront.net/wp-content/uploads/2015/09/rnn.jpg)

Again, we will restrict ourselves to a short, informal discussion of the topics and refer the reader to the text books mentioned in Section 1.1 to learn more.

1.5.1 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are networks designed to preserve spatial features in images. In theory a neural network could be designed which connects each input pixel to every neuron in the next layer. In practice this would lead to a large number of connections, which would lead to long training times. This would also effectively reduce how deep we could make such a network.

One obvious feature of images that we can take advantage of are their spatial coherence. In an image, pixels near each other will be similar, generally, and different images of similar objects like the cats, dog and cars mentioned earlier, will often contain similar arrangements of pixels. To take advantage of this, we only connect pixels to a receptor field in the layer in front of them. This drastically reduces the number of connections and weights to be trained.

What we end up with in this way is a model which learns low-level features at earlier layers (for example identifying lines and edges), while at the later levels recognising higher level concepts (noses, eyes and faces) and then at the final layer(s) recognising objects like cats, dogs and cars.

CNNs are a biologically-inspired network configuration, where the idea of receptor fields comes from the study of the brain. Experimentation has shown that cat brains have receptors which pick up lines in certain directions, similar to how the

earlier layers of a CNN respond to inputs (Hubel and Wiesel, 1963).

1.5.2 Recurrent Neural Network (RNN)

Just as CNNs take advantage of spatial coherence within images, Recurrent Neural Networks take advantage of temporal coherence in their training data. The trick with using an RNN is to feed the output of a neural network back into itself. This allows it to build up a sense of state. For example, we can envisage a situation where we are translating a sentence. Certain word orders are different between English and French, for example nouns and adjectives. In French “*The United Nations*” becomes “*Les Nations Unies*” Because of this, when learning to translate sentences we must remember the context of the current word, as well as it’s direct translation. Because an RNN can keep track of states, it will reposition the word in the output sequence as well as translating it. This is not just a useful property for machine translation from natural language to natural language, but also for speech synthesis, speech transcribing, a wide variety of text tasks, and stock price prediction.

1.5.3 Neural Network Models Summary

We can see that the building blocks of neural networks are simple perceptrons which can be stacked and combined together in many ways, to create a large assortment of different models. Each of these models gives a different model with different properties suitable for different problem domains. As an example of the flexibility of these models, we train a combined CNN and demographic features into a single deep model in Chapter 5. In the context of our hypothesis, we argue that each problem domain in which we attempt to solve problems with deep learning approaches will require a different model, and that there is no one single “Deep Neural Network” which will perform well for all problem domains.

1.6 Hidden Layers in Deep Neural Networks

In the previous section we discussed some of the ways that neural networks can be connected and configured. Another way neural networks can be customised is by composing them with “special” hidden layers. These units can change the

properties of the whole network, including by reducing their potential to over-fit (dropout layers), making them less sensitive to location shifts (pooling layers) and making models more amenable to be implemented on an embedded device (binarising layers). The combination of ways we can combine hidden layers with models amounts to a large amount of possible configurations. We will now examine some of these special hidden layers in more detail.

1.6.1 Dropout Layers

Dropout is technique used to reduce over-fitting in deep neural networks. It sets the weights of a certain number of randomly chosen neurons in a layer to zero, which means they will have no contribution to the output target (Srivastava et al., 2014). The ratio of layers is chosen as a hyper-parameter and typical values range between 5-15%. In this way, the network is encouraged to contain redundant features, which reduces the ability of the network to fit to noise. This is useful as larger neural networks have a lower chance of getting stuck in a local minimum, but are more likely to over-fit due to the large number of trainable parameters. By using dropout layers we get the benefits large networks, without the over-fitting.

1.6.2 Pooling Layers

Pooling layers are most commonly associated with CNN's. A pooling layer lowers the dimension of a layer by taking a moving window grid over a layer and combining this into a single number using an operation such as average, max or min. The operation used gives us the name of the pooling layer, for example a max operation will be called a max pooling layer (See Figure 1.5).

The result of this is to lower the total number of trainable parameters in a neural network, and hence speed up training and also lower the potential of a CNN to over-fit on an image. Another effect of this layer is that the model will be more invariant to vertical and horizontal shifts. A pooling layer can be thought of as a blurring operation on the middle layers of a CNN.

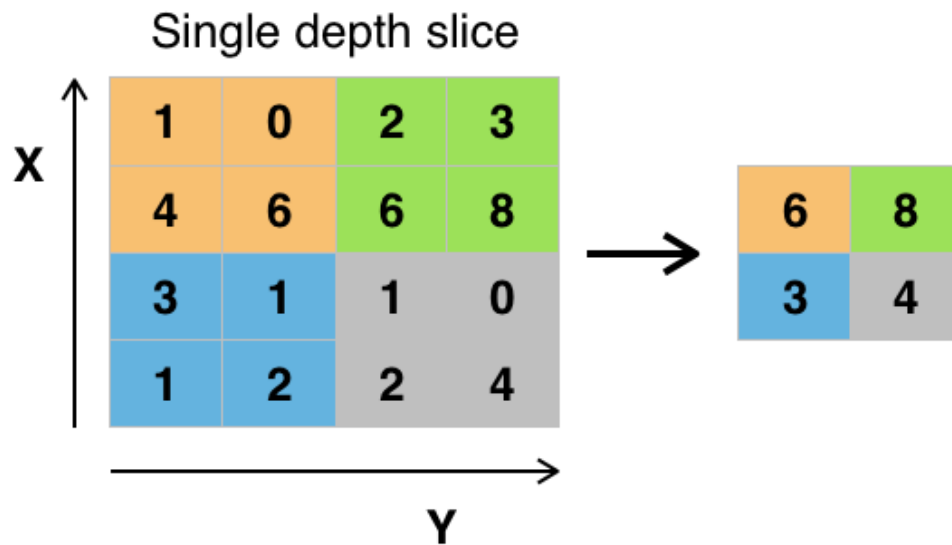


Figure 1.5: Illustration of Max Pooling. Image downloaded from https://upload.wikimedia.org/wikipedia/commons/e/e9/Max_pooling.png

1.6.3 Binarising Layers.

Similar to a dropout, these layers are inserted in-between hidden layers and they change the value of neurons. However, instead of setting the layer to zero, they convert it to binary weight — either a 0 or a 1 (Hubara et al., 2016). The resulting network can then be expressed simply in terms of gated operations such as AND, NOT and XOR. The performance of a classifier built using this architecture will be reduced due to the reduced flexibility of the model. However the resulting neural network can be implemented easily on an embedded device, where power and memory constraints could prevent a full neural network from being possible.

1.6.4 Sigmoid Function Activation

A sigmoid operation constricts the output of a neuron to be between 0 and 1, and was discussed in Section 1.3.1. It is the conventional activation function of a neural network and it used to be used between every layer of a network. However, by using this activation function in the middle layers, the weights of neurons can frequently be set to 0 permanently. Once this happens, they will not contribute the the final classification. This is known as the vanishing gradient problem. Hence the sigmoid function is typically not used in the middle layers of a neural network,

but used in the final layer in a single class classification problem, or its multivariate equivalent (the softmax function) for multi-class classification.

1.6.5 Rectified Linear Units

The Rectified Linear Unit (ReLU) is an activation function which addresses the imploding and exploding gradient problem. If the output of the hidden layer is positive, it remains the same, however if it is negative it is set to zero. This function is illustrated in Figure 1.6.

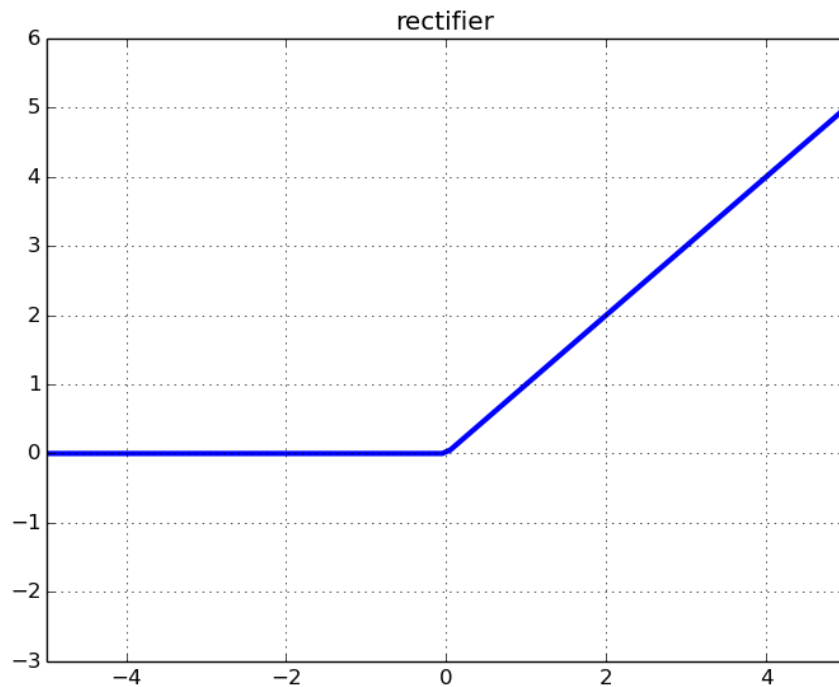


Figure 1.6: Illustration of ReLU. Image downloaded from <https://i.stack.imgur.com/8CGIM.png>

This keeps the output of the activation function large enough so as to not tend towards 0, while still maintaining the non-linearity required to allow the neural network to learn non-linear functions.

1.6.6 Scaled Exponential Linear Units

The Scaled Exponential Linear Unit (SeLu) is an activation function which has been shown to self-normalise a neural network (Klambauer et al., 2017). The function has the following formula

$$selu(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leq 0 \end{cases}$$

This function is plotted in Figure 1.7.

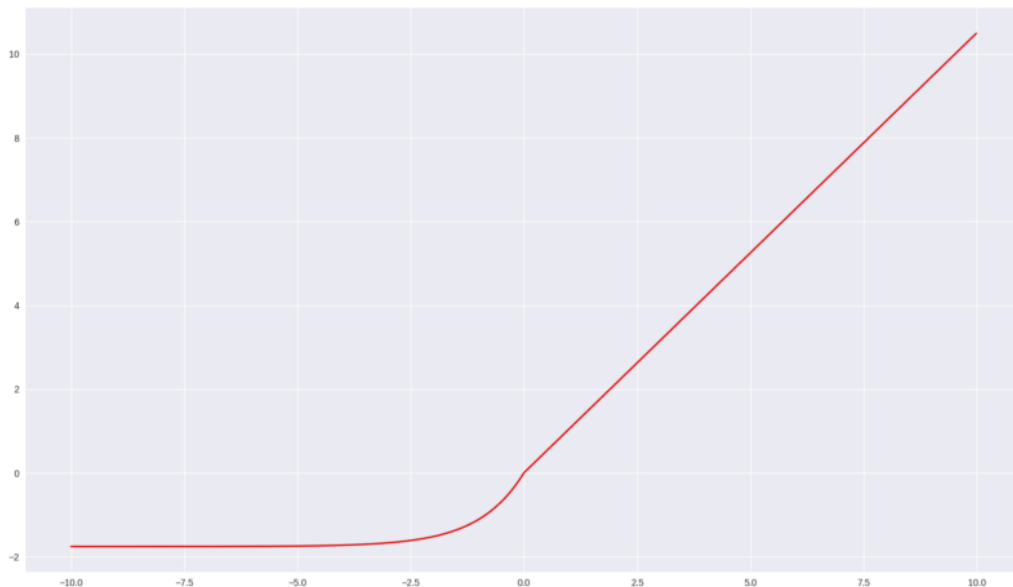


Figure 1.7: Illustration of SeLu. Image downloaded from https://cdn-images-1.medium.com/max/800/1*m0e8lZU_Zrkh4ESfQkY2Pw.png

The use of this activation function is to improve the performance of simple Feed Forward Networks.

1.6.7 Hidden Layers Summary

Here we have examined just some of the known ways of configuring hidden layers in a neural network. In Section 1.5 we examined some of the ways neural networks can be configured into different models. By combining these two aspects of deep neural networks, we create an even larger space of potential neural network configurations. The large space of configurations which can be chosen is

evidence that there is enough room for certain problem domains to favour some solutions and other problem domains to favour others. It remains to be seen whether this is actually the case, and we will investigate this hypothesis by examining solutions to three diverse problem domains. We will now summarise these problems which we expand on in more detail in later chapters.

1.7 Case Studies

In Section 1 we defined our hypothesis as ***“That problem-solving in diverse problem domains using Machine Learning applied to time series data requires diverse models in order to achieve the best performance”***. The alternative to this hypothesis is that there is a single model and architecture which is suitable for all problem domains. We have in the previous sections discussed some aspects of Machine Learning which have fleshed this question out in more detail. Our aim is to evaluate three case studies of practical importance in different problem domains, and evaluate the resulting models which deliver best performance. We will now briefly introduce each of these case studies. More detail can be found for each case study in their respective chapters.

The first case study is an application to predict what grade a student will receive in an undergraduate University course, and to build a system to intervene for students who are at risk of failing (see Chapter 3). The data in this case study are clickthrough logs from students from a Virtual Learning Environment (VLE). Every interaction a student makes is recorded with a timestamp and an ID referring to which resource was used. We extract features for each student and match these features with the results that the students received at the end of the semester. This gave us a large dataset from which we could model which students would be successful and which would not be. One requirement of this case study was that it was important to make accurate predictions as early as possible for students, as otherwise we could not intervene effectively. For example, an email intervention in the first week of the semester would give the student more time to change their behaviour than an email in the days before an exam.

The second case study is an application to detect whether a calf is suckling from its mother cow or not (see Chapter 4) using the movement data captured by movement sensors attached to the calves’ neck. The application of this model is to assist farmers to wean calves off their mothers’ milk so the cow can go back to

commercial milk production as soon as possible. Calves wore accelerometer and gyroscope sensors around their neck, and we are given sensor readings of the movement of the head of the calf. Similar to the previous case study, time is important as the sooner we can detect whether the calf is suckling or not, the earlier an intervention can take place. One requirement of this case study was that the algorithms, once trained, had to be able to run in real time.

The third case study is an application to estimate air pollution levels from images taken on lifelogging cameras worn around the necks of participants (see Chapter 5). In this case study we attempt to correlate image data collected from a wearable camera worn by a life logger with local air pollution levels collected from a portable PM2.5 sensor. We are also provided with demographic data about the subjects, and we wish to include this in the model. Potential applications of this project include a smartphone application to inform users of potential risks of pollution. Time is an important aspect of this problem because the measurements cannot be considered independent. If there is a high level of pollution at one moment, there is likely to be a high level at the next moment. We can take advantage of this temporal locality to improve the performance of our regressor.

Our aim in choosing these case studies was to find practical problems with as much diversity as possible. However, all of these problem domains have one thing in common: there is a time element to data gathered in each of these 3 case studies. In this thesis we examine the solutions to each of these problem domains with this in mind.

1.8 Methodology

As mentioned earlier in Section 1.7, this thesis will use a case study methodology to carry out research. We will now describe the case study methodology in more detail and justify our use of it in this thesis.

Case study methodology is often used in multidisciplinary sciences to bring multiple different methods together. In the field of Machine Learning, case studies are often used to teach concepts using an application such as one of those mentioned in Section 1.2 (Segaran, 2007; Kelleher et al., 2015). In this thesis we will use a case study methodology, as we are applying machine learning to a variety of different tasks to investigate our hypothesis.

Case study methodology is a formal research method which involves examining the subject of the study in depth. Case studies can be useful in trying to analyse a difficult to define research question, leading to Yin (2013)'s definition of the case study research method as "an empirical inquiry that investigates a contemporary phenomenon within its real-life context; when the boundaries between phenomenon and context are not clearly evident; and in which multiple sources of evidence are used". An alternative definition is given by Stake (1995) "As a form of research, case study is defined by interest in individual cases, not by the methods of inquiry used". Other definitions can be found in Merriam (1988); Stake (1995); Miles et al. (2013); Gillham (2000). A more detailed overview of case study methodology can be found in Zainal (2017) and Johansson (2003).

Case study methodologies were originally introduced in work by Frederic Le Play on the topic of household budgets (Sister Mary Edward Healy, 1947) and are primarily used in social sciences, in application in areas such as education (Gulsecen and Kubat, 2006), sociology (Grassel and Schirmer, 2006), community-based problems (Johnson, 2006), law (Lovell, 2006) and medicine (Taylor and Berridge, 2006). They are also used in "practice-oriented" fields such as environmental studies, social work, education, and business studies.

Case study methodology can be considered a combination of quantitative and qualitative research as expressed by Lamnek (2005) who says that "the case study is a research approach, situated between concrete data taking techniques and methodological paradigms". It is important not to confuse case study methodology with qualitative research. For an example of research projects which use a case study methodology involving a mixture of quantitative and qualitative aspects, see Block (1986); Hosenfeld (1984).

Within this paradigm, the method of choosing case studies is important. Stake (1995); Patton (1990) argue that choosing a typical or average case will not provide the richest information. Instead, it is important to choose abnormal case studies which can provide insight into the research question. Some properties which are useful for a case study to have is to be information-rich, critical, revelatory, unique, or extreme. Choosing case studies in this way is known as "information orientated sampling", in comparison to the statistical concept of random sampling. In this thesis each case study will involve extensive quantitative evaluation within each of the case studies.

Other research methods besides the case study methodology include Simulation, Experimental, Correlational, Qualitative, Interpretive Historical and Argumenta-

tive. Using case studies can be considered a “bridge” between these different research methodologies, as each case study can be composed of one or more of these other methodologies. In this way it can be considered a “meta-method”. The process of investigating a research question through the lens of many different research methods is known as “Triangulation” (Denzin, 1973).

One important aspect of the case study methodology is the concept of generalisation (Winston, 1997). One case study may give misleading information if it happens to be an atypical case. By replicating the case study multiple times, we can verify or falsify some theoretical proposition (Campbell, 1975). Multiple cases enhance and support the previous results and this helps to raise the level of confidence in the robustness of the method (Glaser, 2017). In this thesis we have chosen three case studies, as this will allow us to do a comparative analysis between the resulting solutions.

In Section 6.2 we will compare each of the case studies against each other. By doing this we can compare the results of our case studies and attempt to validate our hypothesis.

1.9 Problem Statement Summary

In summary, we have defined our research hypothesis as ***“That problem-solving in diverse problem domains using Machine Learning applied to time series data requires diverse models and models in order to achieve the best performance”***. We will investigate this question by implementing solutions to three case studies in different problem domains and examining their resulting best models.

We have discussed some of the many applications of Machine Learning and shown that there is a trend of applications across many problem domains having state-of-the-art solutions which were vastly different up until recently. In many cases these best-performing solutions were overtaken in terms of performance by deep learning models, and even more recently were then overtaken by deep learning models which were customised to the problem domain. We questioned whether in the future this will lead to multiple models converging into a single model, in the context of time series problems. This is the inspiration for the hypothesis question in this thesis.

We have discussed deep learning and the large number of models and configu-

rations available. This shows that there is enough space so that it is feasible for different problem domains to have different optimal solutions. Finally we have introduced our three case studies, and shown that if they are diverse enough so that if our hypothesis is true, they should result in different models for best performance.

Chapter 2

Literature Review

In the previous chapter we have stated our hypothesis question and we have introduced our research methodology. This methodology involves investigating the optimal solution to applications or problems in three separate problem domains, and examining whether those solutions are distinct. In this chapter we will review the state-of-the-art for each these problem domains. This will help us develop solutions to them which we will draw on heavily in the following chapters. We will not discuss a background of Machine Learning itself, as we have covered much of it in Chapter 1.

We also note that there is a large difference between the literature review of the first topic (applications of Machine Learning to Educational Analytics) and the other two (agriculture and environmental science). This is because the problem we have chosen in the education space is well studied, and so we can compare our solution with other approaches. However, as an investigation into the effectiveness of deep learning, this problem domain is novel, as these approaches have only recently begun to be applied (Li et al., 2017; Okubo et al., 2017). The problems we have chosen in the other two case studies are, as far we have been able to determine, unique. Hence the literature review for these sections will involve a more broad review of the problem domains.

2.1 Applications of Machine Learning to Learning

In this section we will investigate previous work on the application of Machine Learning to (human) learning. In particular, we will focus on the problem of esti-

making student grades from the web logs of a Virtual Learning Environment (VLE). As we mentioned above, there is a lot of previous work on this topic against which we can compare our own work. Across the literature in this problem domain there is a wide variety of source data, target variables and evaluation metrics. This makes the comparison between them difficult, however it does establish a performance baseline against which we can compare.

Source data used in the literature is usually one of the following:

- VLE data.
- Massive Online Open Courses (MOOC's) data.
- In-class examinations.
- Demographic information.

Similarly, the target variable for a Machine Learning classifier in the learning analytics area is usually one or more of

- Whether the student passed or failed.
- The grade a student achieved.
- Student retention.

The metric used to evaluate the success or failure of a classification is one or more of the following

Accuracy The percentage of correctly classified examples. This can give misleading results if the data is biased. For example, imagine a dataset of which 95% of the target variables belong to a single class. If a classifier achieved an accuracy of 90%, this actually performs worse than simply always predicting the most common class. Without knowing the ratio of target variables, this figure means little. It ranges from 0 (worst) to 1 (best).

Recall Recall is the number of true positives divided by the total number of positives. For example, in the case of predicting if a student will fail or not, this number tells us how many students who failed did we miss. We can increase this by lowering the cutoff threshold. It ranges from 0 (worst) to 1 (best).

Precision Precision is the number of true positives divided by the number of true positives plus the number of false positives. For example, this number

tells us of the students we predicted will fail, how many will actually fail. It ranges from 0 (worst) to 1 (best). We can increase this by increasing the cut-off threshold. Note that this means that precision and recall are trade-offs of each other.

F-Score Arithmetic mean of precision and recall. This is a measure which is more robust to imbalanced classes than accuracy is. Again it ranges between 0 (worst) and 1 (best).

Receiver Operating Characteristics Area Under Curve (ROC AUC) A Receiver Operating Characteristic plots the values of precision and recall against each other for various values of a threshold. By taking the area under this curve we derive another metric which is robust to class imbalances. This metric ranges between 0 (worst), and 1 (best). One important thing to note is that a ROC AUC score of 0.5 or less means that the classifier behaves no better than assigning a class randomly.

Kappa This is another metric which is robust to class imbalance. It is based on inter-rater agreement.

Similarly, the metric used to evaluate the success of a regression is one or more of the following

Mean Squared Error (MSE) The sum of the square of the differences between the predicted value and the true value. This is useful for giving a direct indication of how wrong our predictions are. However, a single large missprediction will result in Mean Squared Error which has been heavily penalised, and this metric is therefore not suitable for datasets with large outliers.

Mean Absolute Error (MAE) The sum of the absolute differences between the predicted value and the true value. It is more robust to outliers than MSE.

Regression Coefficients The extent to which an input variable is changed with a target variable. For example, if a regression coefficient between number of clicks of a resource and grade of 0.5 is recorded, then for every resource clicked, their grade increases by 0.5. This is only applicable to linear models. It is also important to note that this metric only works well if the input variables are uncorrelated. If they are correlated then we can get large positive and negative regression coefficients without any meaning. Because of this it is important to report this metric with another metric such as p-value or MSE.

p-value The probability that given the null hypothesis is true that an effect of the measured magnitude or larger would have been observed. A commonly used cut-off point is to say that if the p-value is below 0.05 then the result is not significant.

Finally, a small amount of the literature used the resulting Machine Learning algorithms to perform an intervention on students that it detected were likely to perform poorly. In these papers the effectiveness of the intervention was sometimes assessed, typically by the change in the number of students who dropped out (if their target is to increase retention) or a measure of the impact on grades (if their target was to measure performance in exams). We will divide this section into two sub sections, one focusing on papers which attempted to predict student grades, and another which focuses on those which predict student retention.

2.1.1 Student Grade Prediction

The objective of the majority of papers in this field was to predict the grade of students in their end of semester exams for one or more courses. Here we will examine some literature with this objective, in a mostly chronological order. In some of the following papers the target was a numerical grade (e.g. 65%), in others a banded grade (A+, A-, B+, etc. ...) and in others it is simply a binary pass or fail classifier. The metric used to assess the performance of the algorithm will depend on whether a classification or regression is used.

1. In a landmark paper by Calvo-Flores et al. (2006), the authors predicted student exam performance based on VLE log data. Their aim was to find students who are in need of assistance. This work established that features derived from the VLE access logs were enough to predict success with a high degree of confidence. Some of the features that they used include the ratio of resources viewed and total resource views. In their case they tested it on a course with over 240 students and had a prediction accuracy above 80% on a pass or fail binary target.

To achieve this they used a Radial Basis Function Neural Network (RBF NN) with a single hidden layer (Chen et al., 1991). This is a neural network, typically with a single hidden layer, where the outputs of the hidden layer are combined using a radial basis function. The important characteristic of a

Radial Basis Function is that it stores examples of certain classes, and computes a similarity score between the example and the new sample given.

2. In an another influential early paper by Romero et al. (2008b), they introduce a four-step framework for analysing VLE access data. These steps are:
 - (a) Collect data.
 - (b) Process data.
 - (c) Perform data mining/Machine Learning steps.
 - (d) Deploy results.

This paper also described how this data can be processed to be used in a Machine Learning task to predict which students will pass and which will fail. It also described how this data can be visualised. In a follow up paper (Romero et al., 2008a) they described the results obtained using a random forest (Ho, 1998). In this paper they were able to achieve a prediction accuracy of 65%. We note that this result is worse than that achieved in Calvo-Flores et al. (2006), however this is probably due to them using a different data set, rather than any modelling issues. This is a problem across the field where because of privacy issues it is often not possible to create a benchmark dataset, particularly for VLE data. Hence it can be difficult to directly compare results and establish what the state-of-the-art is.

3. Macfadyen and Dawson (2010) again used VLE access features to predict student grades. In this paper they used features such as messages read, time spent online and number of discussion posts that students participated in. Using a linear model they were able to achieve a 73.7% accuracy on a data set composed of 118 students, which is in line with previous studies. Once this had been done, they created a dashboard for lecturers, so that they could see who potentially needed help in their courses.
4. Affendey et al. (2010) used in-class performance to predict end of semester results, unlike the previous papers which relied on VLE data. One might expect that these features would be more correlated with final results than features extracted from VLE access logs because they are a more direct measure of student performance, and this is borne out in the results in which they obtained an accuracy of 95.29% on end of year results on a class of 2,427 students. This was achieved using an RBF Neural Network and a Naive Bayes classifier Zhang (2004).

5. In Casey and Gibson (2010) the authors performed a statistical analysis on the access logs of a VLE for three courses. They examined many features extracted from these access logs, checking if they were correlated with the final exam mark the students achieved. These features included VLE activity, unique page views, number of log ins, whether they were on or off campus, the coverage of resources the students accessed, and the effect of accessing Moodle (the VLE that they used) on different days of the week. This work influenced our own work in deciding which features would be useful to extract from the logs.
6. Work by Baradwaj and Pal (2012) classified students into one of four groups (first, second, third or fail) using an ID3 tree (Quinlan, 1986), as well as a variety of other methods. Again this was based on several in-class features such as previous semester marks, assignment performance and lab work proficiency. They then described some decision rules generated from this tree. This work did not include any empirical results.
7. One question that Molina et al. (2012) addressed is the problem of parameter tuning. In this paper they used 14 different datasets collected from the Moodle VLE. Using this, they created a classifier for each of a set of parameter values. They noted that the accuracy of these classifiers varied between 50-70%. They then went a step further and took a meta-learning approach, using the parameters of the models used (e.g. decision tree confidence factor for pruning, and minimum number of objects per node) and properties of each of the data sets (number of attributes, number of instances and number of classes) as inputs to another model which attempted to estimate what the prediction accuracy of that model would be.
8. In a follow up paper by Romero et al. (2013) they took a similar approach but the meta-learning target changed to determining which model should be used on an unseen data set. This is useful for the “cold start problem”, where we wish to make predictions for a course which is in its first year of running and we do not have any previous data to learn from. They expanded the number of datasets to 32, and for each tested 19 separate models available from the WEKA Machine Learning toolkit (Holmes et al., 1994). For each one they evaluated the performance of the classifier across a variety of metrics. They then came up with a measure to determine which of the 32 datasets was most similar to an unseen dataset.

This is useful as there is a class of modules for which we cannot make any

reasonable predictions: those which are in their first year. Because we do not have any training data to work with, we cannot make predictions for student success. This can be used to choose an appropriate model for such a course.

9. Wolff et al. (2013) used features from both VLE sources and in class examinations to predict student grades. Some features extracted in this paper include log-in frequencies and in-class performance assessments. They used a large cohort of students, with the largest of three courses examined containing 4,397 students. In this paper they reported a precision, recall and f-metric score of 0.8, 0.3 and 0.4 respectively using a linear model. Note that in these data sets the classes are often imbalanced because there are a lot more students passing than failing students. Hence metrics such as these are more useful than accuracy, as they are less susceptible to giving misleading results when we have imbalanced data.
10. In another paper the following year Wolff et al. (2014) described using a similar classifier in a dashboard to help lecturers and administrators prevent students from dropping out of a course. In this paper they described an ensemble of K Nearest Neighbours classifier on weekly aggregate data, K Nearest Neighbours classifier on demographic data, a decision tree trained on VLE data and a Bayes Network trained on both demographic and VLE data. Importantly, they described a method of predicting student grades on a weekly basis. This is important, as if we wish to intervene to help students we must do so as early as possible.
11. Jayaprakash et al. (2014) used an SVM to predict a student's grade using both VLE and demographic data. Importantly, they used this model in a student intervention system across 451 students. The performance of this intervention system was monitored using an experimental design in which some students were part of a control group, others were given "awareness messaging" treatment and a third group were given an Online Academic Support Environment (OASE) treatment. Interestingly, the intervention system caused some poorly performing students to drop out of the course early in order to avoid academic and financial penalties. An analysis showed that there was no significant difference between the dropout rate of the two treatment groups, however it did show a significant difference between both of the treatment groups and the control group.
12. In Dodge et al. (2015) they used features generated from student surveys

and student demographics to predict student grades for 882 students. In this study they created features called “trigger events”, such as the number of log-ins to their VLE. These were manually created with the assistance of faculty. They then counted the number of “trigger events” for each student and correlated with their grade. In this paper, they showed that there was a significant correlation between the number of trigger events and the grade a student achieved using a linear model. They then used this to perform an intervention with students whom they predicted were performing poorly. They found that there was no significant difference between those who received interventions and those who were in the control group.

13. In a very recent paper Okubo et al. (2017) evaluated the performance of a Recurrent Neural Network on VLE data to predicting a student’s grade. Similarly to Wolff et al. (2014), they made predictions on a weekly basis. In this paper they were able to achieve a R^2 score of 0.99 by the final week between the predicted student grade and the grade the student received. They also found that the performance of the RNN outperformed that of a linear regression consistently and hence they were able to make confident predictions earlier on in the year.
14. Interestingly they predicted numerical grade, rather than predicting binary grade (pass or fail) or a grouped grade (A+, A-, B+, etc...). This is a more difficult task to achieve, however the increased granularity of this prediction would be useful for deciding what type of intervention a student should receive. As a counter point, Li et al. (2017) also used a Recurrent Neural Network, however they found that the performance of the classifier actually got worse each week. We will discuss this paper more in the next section.

In general, in the above literature we can notice few common themes. The first is that if a model is built on VLE access data alone, we can expect an accuracy of between 50-80%, depending on many factors including choice of model and parameters, and the properties of the dataset. However, accuracy is a poor metric to evaluate performance on these data sets as they are frequently imbalanced. Secondly the performance of interventions when measured with a control group, varies between either effective or ineffective, and a lot of thought must go into the interventions in order for them to be effective.

In our own work (Corrigan et al., 2015; Corrigan and Smeaton, 2017) we expand upon the literature mentioned above, and these papers will form the basis of the work reported in Chapter 3.

2.1.2 Student Retention

We will now describe work whose target variable is student retention rather than predicting student grades.

1. Singell and Waddell (2010) used demographic information to build a model of which students are more likely to drop out of a course. Some of the features that they used include residential status, gender, race and Grade Point Average (GPA). Their target was to predict which students were likely to drop out, and to examine the relative importance of each of these features. To do this they used a linear model.
2. In a seminal paper Arnold and Pistilli (2012) built a feedback system known as the Purdue Signals project. In this project they predicted student retention using a mixture of VLE data, in-class examinations and demographic data. Some of the features they used include in-class exam performance, VLE usage compared to their peers and GPA. Using this model they created a dashboard in which students were ranked according to a “traffic light” system - green for good, amber for average and red for at-risk. Students could log into this system and view their “traffic light” grade. This data was also given to tutors who could schedule an appropriate intervention. Using this system they were able to achieve a 24 percentage point increase in student retention.
3. Agnihotri and Ott (2014) built a model to predict student retention using a mix of several models, including a neural network model and an ensemble model. They had a large cohort of students and student data to train on, with a total of 1,453 students. They reported precision and recall figures of 74% and 55% respectively.
4. In Cambruzzi et al. (2015), they built a model using a neural network to predict student dropouts. They reported achieving a precision of 87%. They used this model to implement “pedagogical actions” to prevent students from dropping out. They reported an 11% decrease in dropout rates as a result of using this system.
5. Finally, in Li et al. (2017), they used a Recurrent Neural Network to predict a student’s likelihood of dropping out of a MOOC. The best performing algorithm they reported in this paper had an AUC ROC of 0.924. This was evaluated on a massive sample size of 79,168 learners. One interesting aspect of

this paper was that they also take a week by week approach to predictions, similar to Okubo et al. (2017) and Wolff et al. (2014). However, this was the performance at the beginning of the semester. They found that the performance of their classifier actually decreased when more data was received from later weeks.

Year	Source Data	Target	Models	Evaluation metric	Score
2006	VLE	Grade	Neural Network	accuracy	0.802
2008	VLE	Grade	Forest	accuracy	0.650
2008	VLE	Grade	Decision Tree	N/A	N/A
2010	VLE	Grade	Linear	accuracy	0.737
2010	Demographic	Retention	Linear	N/A	N/A
2010	In Class	Grade	Neural Network	accuracy	0.953
2010	VLE	Grade	Linear	N/A	N/A
2012	In Class	Grade	Decision Tree	N/A	N/A
2012	VLE	Grade	N/A	N/A	N/A
2012	VLE	Retention	N/A	N/A	N/A
2013	VLE	Grade	Naive Bayes	Sensitivity	0.850
				Precision	0.860
				Recall	0.860
				Kappa	0.700
				AUC	0.890
2013	VLE	Grade	Linear	Precision	0.800
				Recall	0.300
2014	N/A	Retention	Neural Network	Recall	0.740
				Precision	0.550
2014	VLE	Grade	Decision Tree	N/A	N/A
2014	VLE	Grade	SVM	accuracy	0.670
				false positive	0.350
				precision	0.610

Year	Source Data	Target	Models	Evaluation metric	Score
2015	N/A	Retention	Neural Network	recall	0.700
				Precision	0.870
2015	Student surveys	Grade	Linear	p value	0.000
2015	VLE	Grade	SVM	AUC	0.600
2017	VLE	Grade	Recurrent Neural Network	accuracy	0.990
2017	VLE	Grade	Recurrent Neural Network	MSE	201.070
2017	MOOC	Retention	Recurrent Neural Network	R2	0.130
				AUC	0.924

Table 2.1: Summary of applications of Machine Learning to Learning.

In Table 2.1 we have summarised the publications assessed in the application of Machine Learning to educational analytics, sorted chronologically. We can see from this table that the most common metric to report is accuracy, and this has in general been increasing over time. We can also see that more recently, regression has started to become more represented as regression measures such as MSE and R2 are more commonly reported. This may be because it is harder to get a good result with a regression compared to a two class classification task, and that recent advances in algorithms has made regression more feasible. It should be noted that it is difficult to compare papers directly, as each of these studies was conducted on a different dataset.

Another aspect that the work we have surveyed all have in common, either implicitly or explicitly, is the element of time in their input data. In many of the early projects, time was ignored, and the resulting prediction could not have been used to make an intervention mid-semester. Later on, many of these projects, including our own work, created new features for each week and used these to make a rolling prediction. Recently we have seen some papers take a more sophisticated approach to time series prediction using Recurrent Neural Networks. In general

using time series methods is an under explored topic which we will examine in Chapter 3.

2.2 Applications of Machine Learning in Agriculture

Here we will review the literature on the applications of Machine Learning in Agriculture. The case study we have chosen in this problem domain is to detect whether a calf is suckling from the mother cow or not based on movement sensors placed in a device on its neck. As we have mentioned earlier, we have not been able to find any comparable previous research for this exact problem. Hence, this literature review will be more broad than the previous section. Again we will list the related work in a chronological order.

1. In McQueen et al. (1995) the authors presented two early case studies of Machine Learning applied to an agricultural context. The first was an application of disease prediction in plants. In this application, the author reviewed a project which used the AQ11 decision tree algorithm (Michalski and Larson, 1978) to determine the diseased status of a given plant. To collect this data, the plants were manually assigned to one of 17 disease categories by an expert collaborator. The features were collected from a survey of the plants and included details such as environmental descriptors, condition of seed, condition of fruit pods, condition of leaves and condition of stem. This algorithm was found to have an accuracy of 92%. A second case study presented in this paper was to assist with the culling of cows. In a farmer's stock of cows about 20% are culled each year to preserve feed reserves. The target variable in this case was whether a farmer chose to cull a cow or not. The features used included features such as location, age, trait survey and production details. This application was found to have an accuracy of 95%.
2. In "Niche Modeling and Geographic Range Predictions in the Marine Environment Using a Machine-learning Algorithm", Wiley et al. (2003) developed a way of determining the presence of certain types of fish in locations. In this project the aim was to predict whether a certain fish species would be present or absent in a location given Bathymetry (seafloor depth) and Physio-chemical parameters (temperature, salinity, oxygen levels) of that space of water. They evaluated their algorithm using the AUC ROC

measure, and achieved a scores between 0.613-0.991, depending on the fish species.

3. In “Applying Machine Learning to Extract New Knowledge in Precision Agriculture Applications” Dimitriadis and Goumopoulos (2008) recorded light, temperature and moisture from sensors embedded in a field. Using this, they built a model of whether a field was healthy, under heat stress or under draught stress using a decision tree. They reported an 89.13% accuracy rate, and a 5% false positive rate. This model was then used to assist the operation of an irrigation system.
4. In a project to identify land cover usage, Rogan et al. (2008) classified satellite imagery into one of several categories (forest, agricultural, urban, grassland). They compared the performance of a neural network and two decision tree algorithms, finding that the neural network produced the best results, with an 84% accuracy. This speeds up what was previously a laborious manual process. Some applications of land cover include measuring forest pest infestation, logging, wildfires and sub-urbanization.
5. In “Cow behaviour pattern recognition using a three-dimensional accelerometer and support vector machines” Martiskainen et al. (2009) use an SVM to determine the behavior of a cow. This is very relevant to our case study on this topic. In this study, they extracted 9 features from accelerometer data gathered from cows. They used this to determine whether a cow was standing, walking, lying, ruminating, feeding or walking lamely. They were able to achieve a 78% precision, with a kappa value of 0.69.
6. In “Visual detection of blemishes in potatoes using minimalist boosted classifiers” Barnes et al. (2010) used images of potatoes to detect the presence of blemishes. They used the AdaBoost ensemble classifier (Freund et al., 1996) and achieved an accuracy of 86.9%.
7. Hayashi et al. (2010) developed a strawberry harvesting robot. As part of this work they needed to detect whether a given strawberry should be harvested or not. Using machine vision techniques they determined correctly whether it was a peduncle 60% of the time. They were also able to estimate the maturity of the fruit to a Mean Absolute Error of 7.4. With these algorithms, they were able to harvest the strawberries successfully at a rate of 41.3% using a suction device.

8. In "Development of an autonomous early warning system for *Bactrocera dorsalis* (Hendel) outbreaks in remote fruit orchards" Liao et al. (2012) developed a system to detect if there is currently a pest outbreak. The inputs to this system included fly population density and meteorological data (temperature, wind speed, solar radiation), and they used an SVM to train the model. They recorded a sensitivity, specificity, accuracy and precision of 98%, 100%, 100% and 100% respectively.
9. Faïçal et al. (2014) used sensors on the ground to optimise the spread of pesticides on a field by Unmanned Aerial Vehicles (UAV's). In this project there were chemical sensors in a field which detect the levels of pesticides which have been sprayed. These were used to guide the UAV to areas which it may have missed, perhaps because of strong winds. In this project they used a Nearest Neighbours approach to interpolate the level of pesticides in areas between sensors.
10. In "A novel behavioral model of the pasture-based dairy cow from GPS data using data mining and machine learning techniques" Williams et al. (2016) attempt to determine whether a cow is resting, walking or grazing. To do this they extract 43 features from a GPS sensor attached to the cow. They found that using the JRIP (Cohen, 1995) algorithm was the most effective way of doing this. They achieved a 85% accuracy, with an ROC AUC score of 0.87 and a F-metric of 0.76.
11. In "A Machine Learning Approach to Forecasting Remotely Sensed Vegetation Health" Nay et al. (2016) used multi-spectral imaging of an area from above to estimate the enhanced vegetation index (EVI) (Huete et al., 2002) using a Gradient Boosting model. To evaluate this model, they compared a model which used only lagged EVI index, and compared this to a model which used both lagged EVI index and multi-spectral EVI. They found that this combined model improves the observed Mean Squared Error by 40-50%.
12. In "Data Mining and NIR Spectroscopy in Viticulture: Applications for Plant Phenotyping under Field Conditions" Gutiérrez et al. (2016) used Near Infra Red (NIR) spectroscopy to determine a grapevine variety. They found that using a SVM, they were able to create a model which was 88.7% accurate.
13. Multi-spectral Imaging is again used in "Deep Gaussian Process for Crop Yield Prediction Based on Remote Sensing Data"(You et al., 2017) to deter-

mine crop yields. Using a combination of a CNN, RNN and Gaussian Process, they were able to obtain a Mean Squared Error of 5.55 on this task.

14. In "Machine Learning Based Computational Analysis Method for Cattle Lameness Prediction", Liakos et al. (2017) use sensors on cows to detect whether they are lame or not. These sensors extract four features - steps per day, meters walked per day, lying per day, and eating per day. In their paper they tested a Neural Network, a Random Forest and an SVM. They found that the Random Forest performed the best. They were able to achieve both an accuracy and ROC AUC score of 1.0.
15. Finally, similar to (Dimitriadis and Goumopoulos, 2008), Goldstein et al. (2017) returned to the problem of optimising an irrigation scheme. In this paper they attempted to predict to the millimetre level an irrigation plan that would have been devised by an agronomist. They used a Gradient Boosting Algorithm using features such as temperature, radiation levels, humidity and rain forecasts. In this paper they were able to predict the prescribed irrigation plan to within 0.113 Root Mean Square Error (RMSE).

Year	Source Data	Target	Models	Evaluation Metric	Score
1995	Survey	Plant Disease Prediction	N/A	Accuracy	0.920
1995	Survey	Culling Cows	N/A	Accuracy	0.950
1999	Images	Mushroom Grading	Decision Tree	N/A	N/A
2003	Bathymetry	Fish presence	GARP	AUC max	0.991
2008	Sensors	Draught Stress	Decision Tree	AUC min	0.613
				Accuracy	0.891
				False Positive	0.050
2008	Satellite Imagery	Land Cover	Neural Network	Accuracy	0.790
2009	Accelerometer	Cow Behavior	SVM	Accuracy	0.850

Year	Source Data	Target	Models	Evaluation Metric	Score
				Kappa	0.690
2010	Images	Blemish	AdaBoost	accuracy	0.896
2010	Image of fruit	Maturity Estimation	N/A	Absolute Error	7.400
2012	meteorological data	Pest Outbreak	SVM	Sensitivity	0.980
				Specificity	1.000
				Accuracy	1.000
				Precision	1.000
2014	Chemical Sensors	Interpolation	Nearest Neighbours	N/A	N/A
2016	GPS data	Cow behavior	JRIP	Accuracy	0.850
				ROC	0.870
				F-metric	0.760
2016	Multispectral Image	Enhanced Vegetation Index	Gradient Boosting	MSE Reduction	40.000
2016	NIR spectrometry	grapevine variety	SVM	Accuracy	0.887
2017	Pedometer	Cow lameness	Random Forest	Accuracy	1.000
				ROC AUC	1.000
2017	Multispectral Image	Crop Yields	RNN + CNN	RMSE	5.550
2017	Soil moisture	Irrigation Plan	Gradient Boosting	RSME (mm)	0.113

Table 2.2: Summary of applications of Machine Learning to agriculture.

In this section we have examined a broad range of applications of Machine Learning to Agriculture. We have summarised these papers in Table 2.2. From the target column we can see that there is a huge variety of projects attempted in this field. We also note that certain algorithms have come into and out of style during this period. In the 1990's, many papers used relatively simple tree methods

and shallow neural networks. Support Vector Machines become more and more popular towards the beginning of the 2000's. Recently we have begun to see Gradient Boosting methods and methods associated with deep learning such as Convolutional Neural Networks become more popular. We observe that results have been consistently high since the earliest papers, indicating that there are a lot of "easy wins" in this area. We again note that it is difficult to establish a "baseline" score, as these papers all use very different methods and datasets.

2.3 Application of Machine Learning to Environmental Science

In the final section of this chapter, we will examine some of the literature in the problem domain of environmental science. The case study that we have chosen in this domain is to determine the air pollution level from an image taken from a wearable camera. Similarly to the previous section, we have not been able to find an exact match to this problem in the literature and so this section will cover a broad range of topics. Once again, we present this related work chronologically

1. In "A Machine-Learning Approach to Automated Knowledge-Base Building for Remote Sensing Image Analysis with GIS Data", Huang and Jensen (1997) developed a method of using multi-spectral imaging to determine if an area is a wetland. Using a decision tree they were able to achieve an accuracy of 74.16%.
2. In a paper with a potentially large environmental impact, Kubat et al. (1998) built a model to detect oil spills from radar imagery. They tested a Neural Network and a decision tree and found that they were able to achieve an ROC score of 81.1.
3. In "Environmental data mining and modelling based on Machine Learning algorithms and geostatistics" Kanevski et al. (2004) used Machine Learning to interpolate soil pollution levels. They used sensors to detect soil pollution in a given area, and they predicted what the level of pollution will be in geographically close unmeasured soil. To do this they used kriging (Oliver and Webster, 1990), Support Vector Machines and Neural Networks.
4. In "Machine-learning paradigms for selecting ecologically significant input variables" Muttill and Chau (2007) investigated algae bloom. They used sen-

sors to detect the levels of several factors including nitrogen levels, previous levels of chlorophyll, temperature rainfall and radiation levels. From this they estimated the current levels of chlorophyll, which is an indication of algae bloom. One significant result from this study was that the most significant factor for predicting algae bloom was previous levels of chlorophyll.

5. In “An information technology enabled sustainability test-bed (ITEST) for occupancy detection through an environmental sensing network” Dong et al. (2010) attempted to determining the occupancy levels of buildings using various sensors placed inside such as gas detectors, CO₂ levels and Wireless Sensor Networks. Using various Machine Learning approaches including an SVM, a Hidden Markov Model (HMM) (Baum and Petrie, 1966) and a Neural Network (NN) they were able to achieve an accuracy level of 75%.
6. In “QELAR: A Machine-Learning-Based Adaptive Routing Protocol for Energy-Efficient and Lifetime-Extended Underwater Sensor Networks” Hu and Fei (2010) were able to extend the lifetime of underwater sensors by optimising the residual energy of the network. They reported that they are able to improve the life span of the network by 20%, using Q-learning, which is a technique used from reinforcement learning, a field which shares similar goals to Machine Learning (Sutton and Barto, 1998).
7. One previous work that is quite similar to the case study that we examine in this thesis is work in “A Machine Learning model of Manhattan air pollution at high spatial resolution” by Keeler (2014). In this paper they attempted to combine previous PM_{2.5} pollution levels in New York with taxi data from New York. In this paper were are able to correlate their observed features to the actual pollution level with a R^2 score of 0.62.
8. In an even more similar paper to our research Liu et al. (2015) in “PM_{2.5} monitoring using images from smart phones in participatory sensing” attempted to detect the level of pollution from an image in shanghai. The pollution levels were so intense there that a haze is visible at certain times, and this is what this paper extracted from the images to detect the pollution level. Using this technique with a linear model gave a Mean Absolute Error of 27 and an R^2 score of 0.084.
9. Finally, in a very recent paper Stingone et al. (2017) used satellite imagery which contains measurements of toxins in the air to correlate exposure to toxins in a given school district, and the grade that young children achieved

in maths exams. Overall they found that exposure to high levels isophrophone dropped grades by 1.19 points.

Year	Source Data	Target	Models	Evaluation metric	Score
1995	environmental variables	vegetation composition	N/A	N/A	N/A
1997	Multispectral Imaging	wetland classification	Descision tree	N/A	0.742
1998	radar satellite	oil spill	Neural Network	ROC	0.811
2004	soil pollution	soil pollution	SVM & Kriging	accuracy	N/A
2005	soil characteristics	pesticide concentration	N/A	N/A	N/A
2007	chemical sensors	algal bloom	nerual networks	N/A	N/A
2010	sensors	building occupancy	Hidden Markov Model	accuracy	0.750
2010	sensors	residual energy	Q-learning	lifetime extended (months)	20.000
2011	satellite	N/A	SVM	N/A	N/A
2014	taxi data	PM25	neural network	r2	0.620
2016	monitoring station	pm25	N/A	N/A	N/A
2017	images	pollution level	linear	MAE	0.270
2017	satellite measurements	children cognitive skills	math scores	r2 performance	0.084 - 1.190

Year	Source Data	Target	Models	Evaluation metric	Score
2017	monitoring station	pm25	N/A	N/A	N/A

Table 2.3: Summary of applications of Machine Learning to Environmental Sensing

Similar to the work reported in Section 2.2 on applications in the Agriculture area, we can see that there is a broad range of applications of Machine Learning to this problem domain. We summarise these papers in Table 2.3. One theme that is common across many of these papers is the use of GIS data of some kind as either a feature or a target or both. Even though the work is quite broad we were able to find an example of a paper which is similar to our case study (Liu et al., 2015) and we will use the results obtained there as a baseline against which we can evaluate our own solution.

Chapter 3

Case Study: Predicting Student Examination Outcomes

In this chapter we investigate the problem domain of Machine Learning applied to education. We introduce the problem of predicting student grades from Virtual Learning Environment (VLE) logs. In each course in a university students access resources such as presentation slides, lecture notes and quizzes. At the end of the course they take an exam which determines their final grade. Our aim is to correlate the final grade that the student achieves with usage patterns of student interactions with their VLE.

In the university which we conducted our study on, Dublin City University, the VLE which was used is Moodle. Moodle is a popular online learning platform which is used in tens of thousands of educational institutions worldwide. According to its own website “Moodle’s worldwide numbers of more than 90 million users across both academic and enterprise level usage makes it the world’s most widely used learning platform”¹.

By default, Moodle records every instance of a student accessing a Moodle page of any kind and records the page, date and time, the student identifier, and the IP address of the device used to access, to a log file.

In Dublin City University, Moodle is the default platform used throughout the institution and is maintained and run by a University-based unit, Information and Systems Services, who provided us with access to Moodle access logs for students from previous years. The University’s Exams office also provided us with student

¹https://docs.moodle.org/32/en/About_Moodle

registrations for modules, past examination performances on a per-student and per-module basis as well as weekly updates from the Moodle access logs during the period of our work, allowing us to work with past student records (Moodle access and exam performances) as training data, and with current students for whom we could make predictions.

In order to predict student performance in end-of-module examinations, we needed to be able to train a classifier on past student performance where the outcome is known (exam results) and hence we needed to work with modules where every year the log files follow the same repeating annual patterns of access. Each module taught in the University, by its very nature, will have different access frequency patterns which are influenced by scheduling of lectures, lab sessions, group sessions, mid-semester tests, assessment deadlines and final exams. Plotting the overall student activity for each module by simply counting the number of student accesses allows us to determine visually if the module displays an annual periodicity. This is useful for us in determining the set of modules for which there is enough training material from past years which can be used and thus helps us define the subset of modules from the thousands taught in the University, on which we can operate. For instance, if a module drastically changed in content or in delivery within the last year or two because a new lecturer took over and removed class tests, this would show up in the activity levels and would mean that because the training material has changed, that module would not be suitable for weekly predictions. Figure 3.1 and Figure 3.2 show the aggregate activity levels for all students over the past five years for 2 of the modules — CA103 (Computer Systems Hardware) and HR101 (Psychology in Organizations) respectively — and each demonstrates a regular annual periodicity meaning that student accesses from previous years, coupled with the performance of those students in those previous years, provides suitable training material for classifying this year's cohort of students.

We initially receive Moodle logs in a format which contains the user id, module id, timestamp of access, IP address of device used and the Moodle resource being accessed. To this we added the result that each student obtained in the written end-of-semester exam, which for most modules constitutes the largest part of how the overall grade for the module is computed. However we then had to transform this raw data by extracting a set of features associated with each student and module which, in particular, focuses on temporal information associated with the logs.

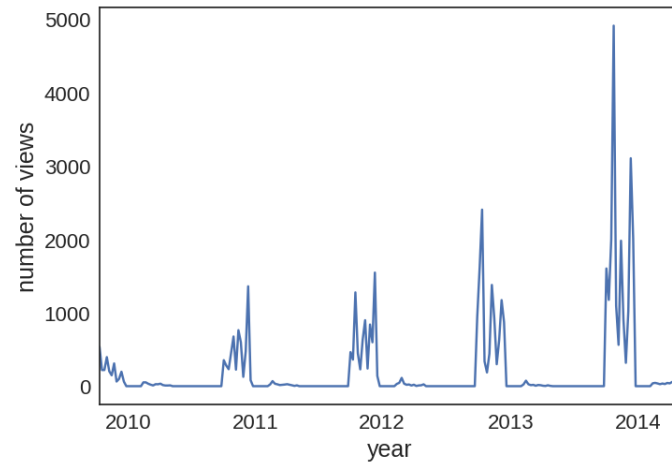


Figure 3.1: Activity levels for module CA103 (Computer Systems Hardware) over a 5-year period.

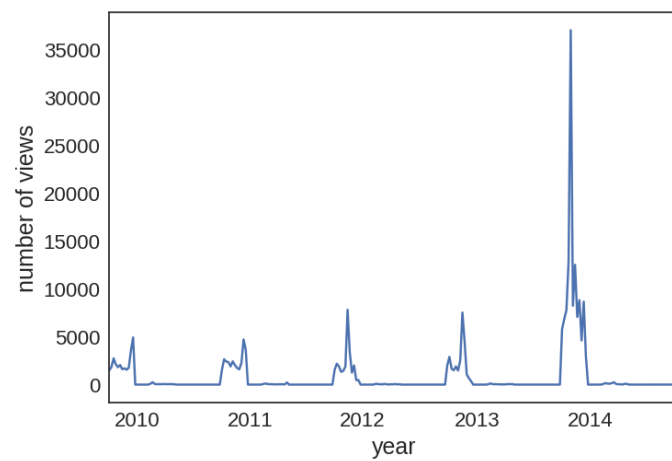


Figure 3.2: Activity levels for module HR101 (Psychology in Organisations) over a 5-year period.

3.1 Ethical and Privacy Considerations

For each of the two years of PredictED we had formal approval from the University's Research Ethics Committee², the University's Data Protection Officer, the Head of student support services, the Registrar and Deputy Registrar, the Dean of Faculty, the University Teaching Committee, and the Module Co-ordinators for each of the 10 modules. In addition, students were presented with a plain language statement of what data was being gathered and what it was being used for when they first logged onto Moodle, and they were given a video tutorial or a visit to a lecture by one of the project coordinators during their first week of semester. When first logging onto Moodle, students were presented with the choice to opt-in or opt-out of receiving the weekly alerts, and they could change their preference at any time during semester.

For more information on ethical and privacy concerns for this case study, see Appendix 2.

3.2 Remodelling Student Data into Weeks

When dealing with a "live" system which is real time, we have the possibility to generate predictions, i.e. to make classifications into predicted outcomes, on a frequent basis. At the extreme level, we could generate a prediction (classification) on demand, i.e. whenever a student would request, which would require access to Moodle access log files in real time. Because this was not possible as the log files were captured by our Information and Systems Services Unit, we had to compromise. Rather than give students the impression they were dealing with a live system which dynamically updated their predicted outcomes based on their VLE activities which would have led to students naturally "gaming" the system in order to improve the estimate of their outcome, we still wanted a level of dynamicity which reflected genuine Moodle engagement activities but prevented "gaming". Thus we decided on a weekly update of outcome predictions and each week we were given a copy of the Moodle access logs for the week previously.

We generated sets of multiple classification features for each student-week combination for the modules we operated on which are described later in Section 3.5, and saved them back into a new data structure using the following Algorithm.

²Dublin City University Research Ethics Committee approvals REC/2014/195, REC/2015/008.

Algorithm 1 Estimating Features by Week

```
1: DECLARE semester_features[num_academic_weeks]
2: for i = 1 to num_academic_weeks do
3:   Declare feature[num_students]
4:   for k = 1 to num_students do
5:     logs  $\leftarrow$  filter_logs(student=k, week=i)
6:   end for
7: end for
```

We found the DataFrame library in the Python package “Pandas” to be a useful data structure for this work (McKinney, 2010). Instead of a multi-dimensional array, the features were stored in a DataFrame, where each row is a feature for a student, and the feature and week are stored as column headers as illustrated in Table 3.1.

This allowed us to generate a new prediction for each student for each week, using all the features available to us up until that week. So for example, for week 3 we took features from weeks 1, 2 and 3 and trained a classifier using training data from previous years, with the results achieved as the predictor variable. We cross-validated this score and took the mean to get the prediction accuracy for that week. Then, once the semester had started in the third week we would use the week 3 classifier we had generated to make predictions for that week. In general as the weeks went on and our system was able to learn more information about students and their levels of engagement with Moodle, the accuracy of the classifiers tended to improve. This is illustrated in Figure 3.3 and Figure 3.4. In Figure 3.3 the first 6 weeks of the semester for years 2011, 2012 and 2013 are used as training set for the year 2014. In Figure 3.4 the first 11 weeks are used as a training set for the same period. Note that in this figure the same patterns exist, however more students were added each year and so the peaks are larger each year.

In the next Section we describe how we turned raw Moodle access logs into features for the classification.

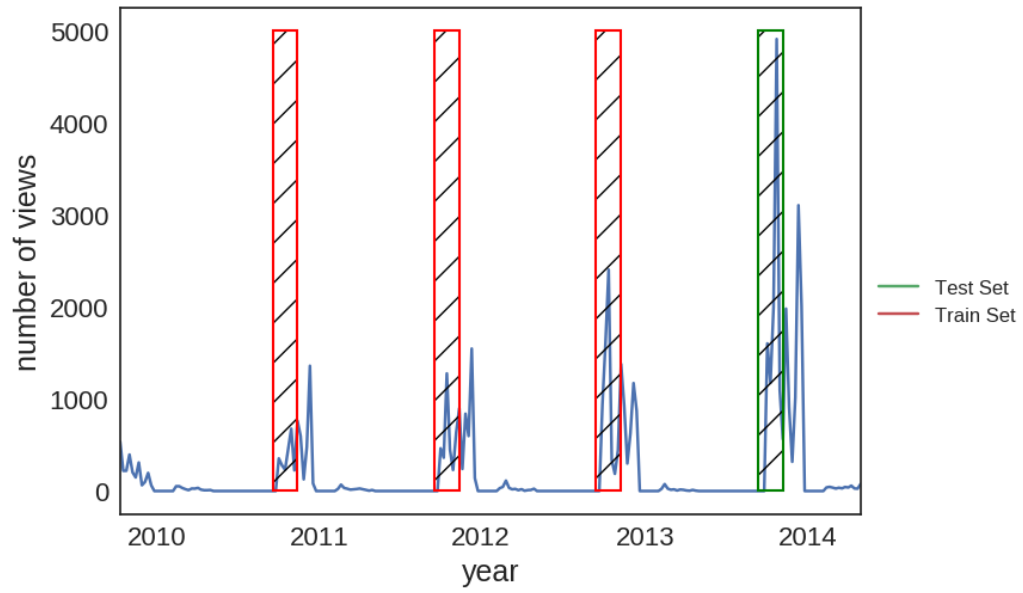


Figure 3.3: Test / train split for CA103 in week 6.

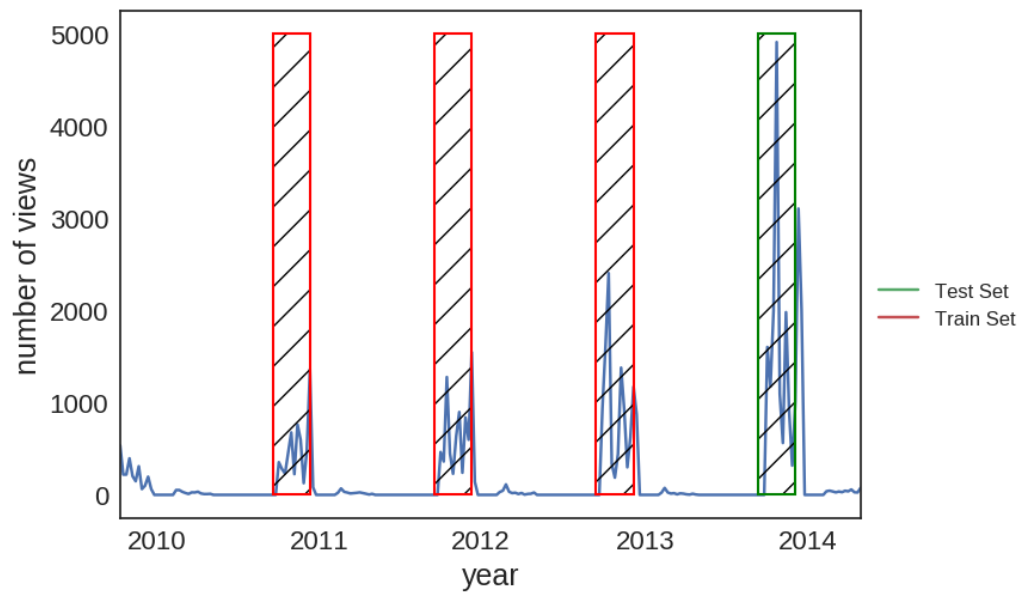


Figure 3.4: Test / train split for CA103 in week 11.

Table 3.1: Sample table of features with 10 students. The table contains two weeks of features (weeks 7 and 8), and four features from each week.

userid	week							
	7				8			
	avg_time	count	in_college	weekend_count	avg_time	count	in_college	weekend_count
33556	19.27	11	1.00	4	16.00	10	1.00	0
33687	18.33	21	0.24	0	12.43	7	1.00	0
33691	16.00	2	1.00	0	16.50	14	1.00	0
33710	10.20	10	1.00	0	18.20	10	1.00	0
33729	10.22	23	0.83	0	9.00	13	1.00	0
33965	19.00	1	1.00	0	9.38	8	1.00	0
33974	23.00	6	1.00	6	22.00	10	1.00	0
33988	10.36	11	1.00	11	21.38	16	0.38	10
34123	9.16	32	0.16	27	21.33	6	1.00	4
34328	13.52	21	1.00	0	14.47	17	0.29	12

3.3 Feature Extraction — Regular Features

Moodle logs were processed by extracting a variety of features for each students' access to each module, in each year for each week. An example of a feature used is a simple count of how many logs each student accessed in total. We also counted in which of the 24 hourly windows students accessed resources, and averaged that across all of the logs as well as the ratio of clicks on week-ends vs. weekdays and the IP address to determine if accesses were from on- or off-campus and used the ratio between these.

To cater for weekly updates to our log files and allow us to generate a new classifier for each week, an additional column was added to the log data, which was the academic week number for that the log. We then divided the log data into 12 separate columns, depending on which week it took place in and filtering out any accesses which took place after week 12. We then trained 12 separate classifiers, the first including only the first week of data, the second including the first two weeks of data (to represent the situation in week two), and so on. Each week, 4 new features were added to the total number of features, so that by the end of week 12 we were making predictions based on 48 features in total. Corresponding to each feature, we extracted a student result which was a simple binary variable, either a pass or a fail.

The features used to train and run the predictor classifier were engineered from the access logs for the University's VLE. Table 3.2 shows a sample of some of these raw access logs.

The column headers in Table 3.2 correspond to the following:

date and time the date and time the Moodle resource was accessed

id a unique identifier for each log entry

userid the id of the student accessing the Moodle resource

ip the ip address from which the resource was accessed

course the id of the course module the student is registered for

module the section of Moodle that was being accessed

cmid content module identifier, a unique identifier for Moodle content

action the action that the student has taken

url the URL of the resource that was accessed

info copy of cmid in most cases

From these access logs we extracted a range of different features. In Section 3.6 we go into more detail about how the logs were filtered and processed into features but some of the features which we extracted include the following:

- How many times a student accessed a particular Moodle resource.
- The ratio of how many times the student accessed a Moodle resource, compared to the rest of the class accessing the same resource.
- Average time of day (in 24 hour clock) a student accessed Moodle.
- How many times was the resource accessed outside the University campus.
- The ratio of times the student accessed a resource during a weekend vs. weekday.
- What ratio of resources the student has accessed.
- How many resources a student accessed, relative to the rest of the class.
- How quickly a student was to access a Moodle resource once it had been posted, relative to the rest of his or her class.

Some of these classifier features have been used in previous studies in particular in work by (Casey and Gibson, 2010), by Bovo (Bovo et al., 2013), and by (Ruipérez-Valiente et al., 2014). Of the above features, the second, third and fourth were novel, in that we had not seen similar features in the literature.

We then used feature selection to determine how many features to use. We tested this across a number of different courses. Of the above 8 features, we found that only including the first four gave us a similar performance to using all of them. This also reduced the time needed to train the model.

Table 3.2: Sample of 10 log entries from Moodle (IP address has been redacted).

date and time	id	userid	ip	course	module	cmid	action	URL	info
2014-09-17 20:27:41	23500016	69291	***	5753	course	0	view	view.php?id=5753	5753
2014-09-17 20:27:36	23500015	69291	***	5753	forum	6136	view forum	view.php?id=6136	5929
2014-09-17 20:27:33	23500014	69291	***	5753	course	0	view	view.php?id=5753	5753
2014-09-11 12:56:08	23490902	61587	***	5753	course	0	view	view.php?id=5753	5753
2014-09-10 06:14:18	23481024	51645	***	5753	course	0	view section	view.php?id=5753	46157
2014-09-10 06:14:13	23481023	51645	***	5753	course	0	view	view.php?id=5753	5753
2014-09-08 14:01:08	23466465	4569	***	5753	course	0	view	view.php?id=5753	5753
2014-09-08 14:00:51	23466459	4569	***	5753	course	0	view	view.php?id=5753	5753
2014-09-08 13:53:44	23466393	4569	***	5753	course	0	view	view.php?id=5753	5753
2014-09-04 17:59:52	23447380	3070	***	5753	url	56893	view	view.php?id=56893	9084

3.4 Training Models

As the next step in applying machine learning to student data, we used a several machine learning algorithms, and selected the one which performed best across all of the modules. To evaluate which one to select, we only considered the performance of the classifier during the final week of features. The results of this experiment can be seen in Table 3.3. In this table we have taken the mean score across the 10 modules selected which we have selected as the modules on which to perform the experiment. From this table we can see that the highest performing classifier was the Support Vector Machine (SVM) (Platt et al., 1999) with a linear kernel.

We can see in Figure 3.5 a boxplot of this information. Here, each algorithm is represented as a box, with the appropriate box being labelled on the x-axis. The red lines indicate the mean of the AUC ROC values, and the height of the boxes indicate the inter-quartile range of the spread of scores. Again, we can see here that the SVM with a linear kernel performs the best.

Table 3.3: Performance of Classifiers on Final Week of Semester

Model	ROC AUC
linear_svm	0.549818
linear model	0.539488
radial_svm	0.537919
knn	0.529566
random forest	0.510561
decision tree	0.508150

Here both SVMs have a 'C' hyper parameter value of 1.0 The KNN has uses 5 nearest neighbours and a Minkowski metric of 2. The Decision Tree uses a gini criterion, with no pruning or maximum tree depth. The Random Forest uses an average of 10 decision trees, each using a gini criterion with no pruning or maximum tree depth.

3.5 The PredictED System

We implemented our system for classification of likely student outcomes for students in our University. In this section we describe the modules we used for this,

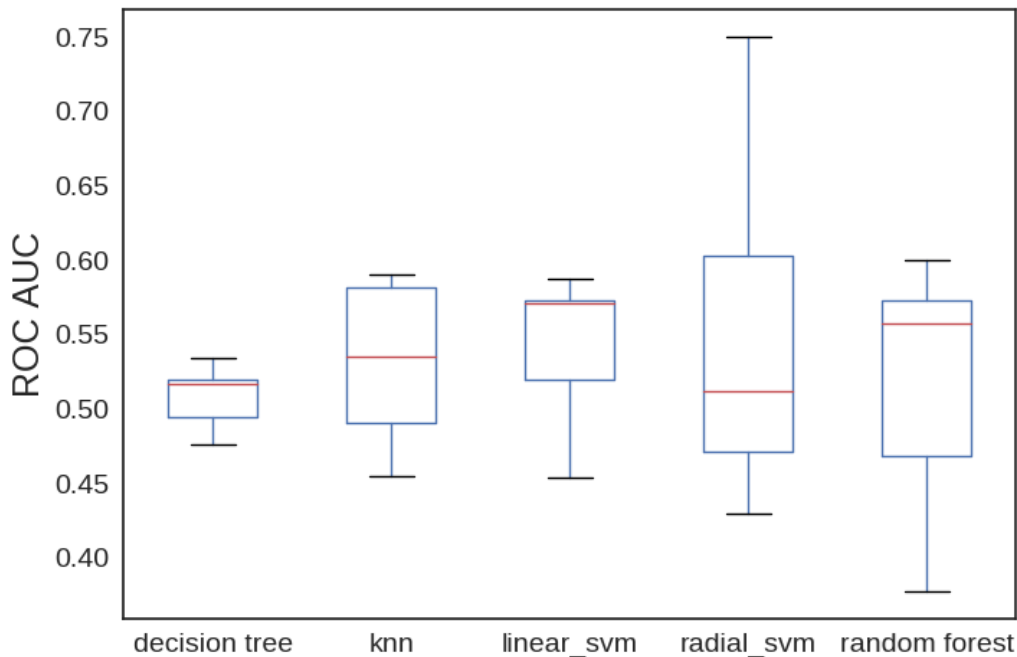


Figure 3.5: Classifier Scores

how we incorporated recommendation of Moodle content and some of the features in our implementation of the PredictED system for two consecutive years.

3.5.1 Choosing Which Modules to Predict Student Outcomes

We were given access to all the VLE access logs for the previous 8 years in which Moodle had been in use in our University. We used this data as part of a system called “PredictED” which would use the weekly classifiers to alert students individually and personally, as to their relative performance in given modules and their likely outcome in the final end-of-semester module examination. Because of ethical considerations (see later) we allowed students to opt-in or opt-out to receiving these weekly alerts.

Having access to past VLE logs and examination outcomes allowed us to experiment with choosing which modules we could run our examination outcome predictions. It was important for us to choose modules for which the size of the class was large enough, the performance of the classifier was good enough and for which the failure rate meant the class imbalance between pass and fail was not too severe. We created a set of heuristics to decide whether the performance of a classifier on a module is good enough on which to make outcome predictions.

The heuristics are as follows:

- A course must have enough previous data, i.e. student completions in previous years, on which to train models. We decided to only take courses with a minimum average of 80 students per year.
- We calculated and graphed the ROC AUC by week (as described later in Section 3.6). Using this metric, a result of 0.5 is a result which is no better than random guessing. So if the graph dips below 0.5 during the semester we do not make interventions in this course for these weeks.
- For some courses we found that the ROC may go up and down during the semester. However, we would expect that as we get more information throughout the semester, the predictive power would improve. We decided that the ROC AUC graph values should be increasing on average in order to use it as a classifier.
- We decided to only focus on first year undergraduate modules, as we felt that these were the modules where the most impact could be made. These modules also tend to be the modules with the greatest number of registered students and the highest failure rates, and thus a lesser class imbalance.

After filtering the modules using these criteria, we found that only 10 modules were suitable candidates from a set of several hundred for the first semester. We can see these courses and the number of students who opted into receiving weekly alerts from the PredictED system in the first three columns in Table 3.4. The final column in this table is an average across all modules. This average is weighed by the number of students in each course.

Even for those modules where the predictions are good enough by the end of the semester, the predictions at the beginning of the semester may not be as accurate. We also set a threshold for the kickoff week for each of the modules where we would not start alerting students until that threshold has been reached. We decided to set that threshold to 0.55 ROC AUC. This meant that in some modules we could start emailing students as early as week 2, whereas in others we did not start emailing them until week 5.

In order to make dynamic predictions based on recent student VLE access behaviour each week, we used a regression to predict the grade that each student in the class would get. We then ranked students by their predicted grade, and then grouped them into deciles. The rationale behind this was that the students could

see how they were doing relative the rest of their class. We used this data to update a web application containing, for each student, the history of their week-on-week class predictions.

Table 3.4: Performance in end-of-module examination showing number of eligible students for module (Regs), and comparing average non-participants' exam mark (Opt-out) vs. participants' average exam mark (Opt-in) and significance of difference between these.

Code	Module Title	Regs	Opt-out	Opt-in	p<0.05	p<0.01
BE101	Intro. to Cell Biology & Biochemistry	284	58.9	62.1	✓	✓
CA103	Computer Systems Hardware	122	70.3	71.3	✓	-
CA168	Digital World	78	63.8	65.3	✓	-
ES125	Social and Personal Development	47	67.0	66.5	-	-
HR101	Psychology in Organisations	152	59.4	63.3	✓	✓
LG101	Introduction to Law	172	53.3	54.9	✓	-
LG116	Introduction to Politics	154	45.7	44.9	-	-
LG127	Business Law	288	60.6	61.8	✓	-
MS136	Mathematics for Economics & Business	157	60.8	69.4	✓	✓
SS103	Physiology for Health Sciences	104	55.3	57.0	✓	-
Across all modules			58.58	61.25	✓	✓

3.5.2 Year 1 PredictED Implementation

For each of the 10 modules in Table 3.4 and for each week after our predictions reach a usable accuracy, our classifier outputs a pass/fail prediction for each student taking the module, plus a confidence value for each. We used the output from this to rank students from least likely to fail to most likely to fail and we also have a tipping point or threshold pointing to the likely overall pass rate for the module. To use this information to feed back directly to students we tested two separate strategies in the first year of implementing PredictED.

For students in some of the modules we sent weekly emails based on our prediction of whether they would pass or fail. For each of these two groups we divided each into two more groups resulting in four groups - “bad” and “poor” for those predicted to fail, “good”, and “great” for those predicted to pass. The rationale behind this is that the best way to divide the groups may be based what we think will happen with their results. Each group is sent a weekly email with “bad” for instance receiving an email saying how students need to work harder, while those in the “great” group were told they are working well on the module. Each email contains pointers to resources the student can use, such as contact details for student support services, contact details for their lecturer.

For students on each of the other set of modules, they were broken into 10 equally sized groups based on the overall prediction ranking and the emails informed each student of what percentile group among the class they fall into. The bottom 50% of students are given a more encouraging email than the top 50%, the idea being that students get an idea of how they are doing relative to their peers in the class. This might encourage some competition among students, incentivising them and hopefully they would see the results of working harder sooner in the feedback loop.

Each week this process was repeated based on the most recent log data downloaded from the Moodle log files. Predictions were generated for each module in the manner described above. For each week of semester, each student was sent a new email (see Figure 3.6).

3.5.3 Year 2 PredictED Implementation Description

In the second year of the running the experiment we gave the students an actionable piece of advice in the form of links to the most important module content

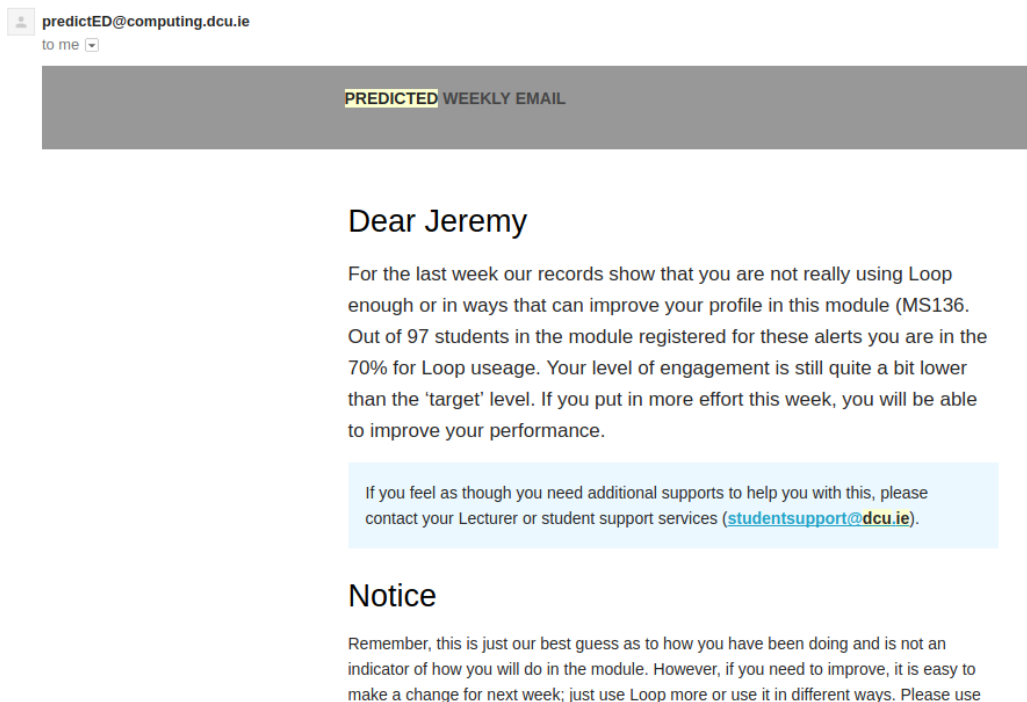


Figure 3.6: Example email sent to students.

which they had not yet viewed. We ranked that course content to determine what to recommend as the most important for each student by counting which content had been viewed the greatest number of times by other students, weighting highly those students who were performing well in terms of class ranking. We took the three most important VLE resources for each student and added them in as links to their feedback email (see Figure 3.7). The students could then click the resources to download and view.

We also developed a web application where students could view further information about their predicted place in class, in order to enable them to take appropriate action to improve their grade. A sample of the personalised graph for just one module is shown in Figure 3.8. The top graph in the figure illustrates that in week 12 of the semester (x-axis), this particular student in module LG127 was predicted to come only within the top 90% of the class (y-axis) and their whole semester shows a gradual decline in relative placing throughout the 12 weeks. The middle graph shows a student in module HR101 was fairly mid-range in class placing, consistently throughout the semester. The bottom graph shows a student who was in the lower part of the class in terms of class placing and then at around week 9 seemed to get a wake-up call and jumped up the class ranking for module SS103, staying in the top 20% until the end of semester. This web appli-

PredictED - Joe Bloggs

Have you reviewed the following links?

- [LEC Morris Chap 7.ppt](#)
- [BE101.docx](#)
- [Lec Morris Chap 6.ppt](#)

Figure 3.7: Example email sent to students.

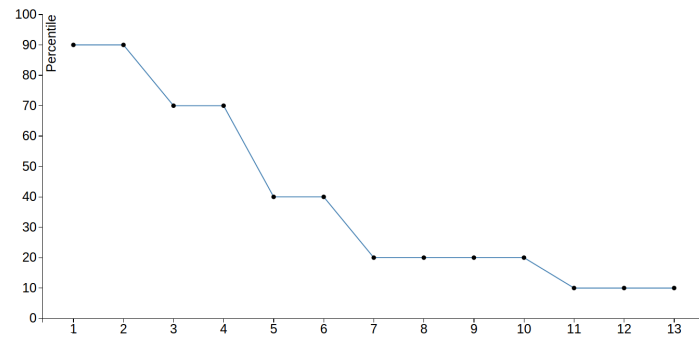
cation also showed a list of the most important resources that the student should view.

3.6 Performance of Predictions

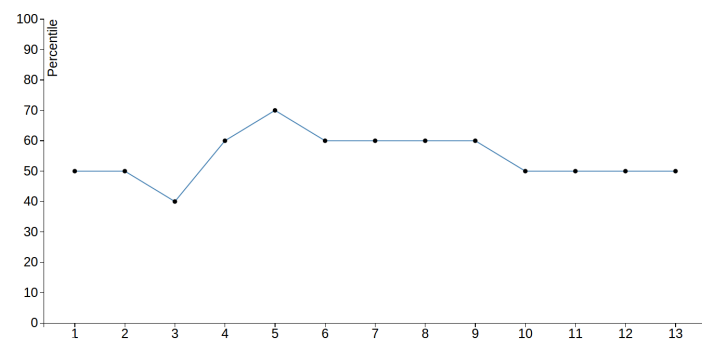
We now look at how we calculate the accuracy of student examination outcome predictions. Once the set of features were extracted for previous deliveries of our target modules, we built a set of classifiers, one for each week of semester and right up to the week of the final examination, which predict students' likely pass/fail in the end-of-semester final examination. In most of the University's modules the examination pass rate is between 70% and 95% and so an issue with the data is that because passes are a lot more common than fails, the data needed to be re-sampled so that the prediction result is not biased towards the most common outcome, i.e. a pass in the module. Another issue is that there can be a lot of features and not a huge amount of instances (students) and so it is necessary to be wary of over-fitting and to choose modules which have large numbers of students.

To determine the accuracy of our predictions we used the Receiver Operator Characteristic Area Under Curve (ROC AUC) relevance measure (Hanley and McNeil, 1982) because it is more resilient to imbalanced classes than other measures, such as simple accuracy. An ROC AUC score greater than 0.5 is better than random. In evaluating the usefulness of different feature combinations we use cross-validation and because the number of samples can be small we use 5-fold valida-

Your predicted place in class for LG127



Your predicted place in class for HR101



Your predicted place in class for SS103

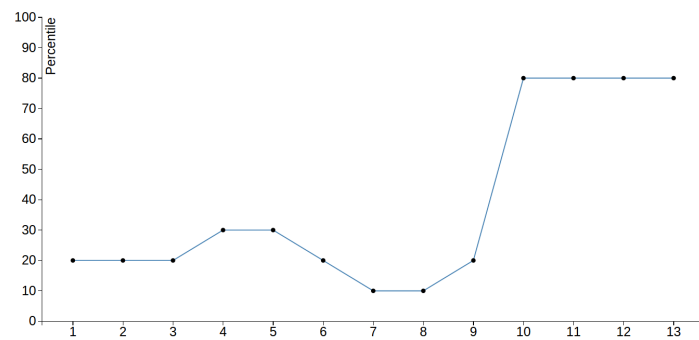


Figure 3.8: Students' view of web application.

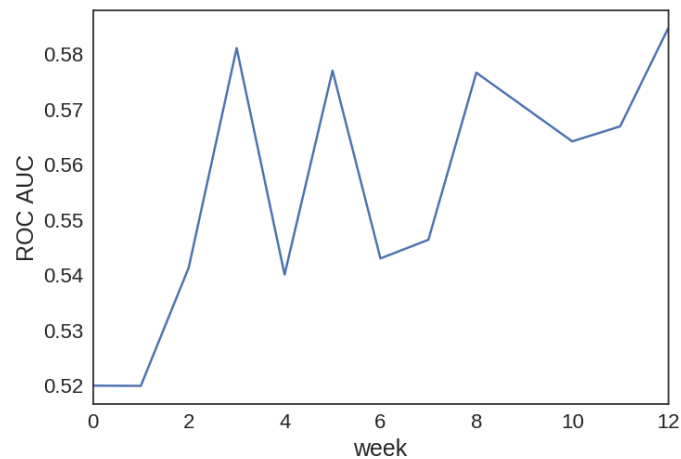


Figure 3.9: Prediction accuracy for SS103 (Physiology for Health Sciences) by week.

tion which reduces the variability of the obtained prediction accuracy score. To determine how effective a predictive classifier is over the semester, and to gauge at which point during the semester it is reliable enough for us to use and actually alert students, we calculate this measure for classifiers built for each of the 12 weeks of log data for which we have extracted features. The features are then scaled using the standard score, and the transformation is saved so that it can be applied to the live data later.

As a rough heuristic, if the resulting ROC AUC value is at or below 0.5, we judge that the classifier is not performing better than random. Once this line is above 0.5 consistently, we regard the classifier as working and usable. In our case, the best modules typically have a value between 0.6 to 0.7 for the weeks following week 3 or 4 of our 12-week teaching semester. This helps us to determine what week to start sending the predictions to students, and also to compare the performance of different classification algorithms on the same data. For example, see Figure 3.9 and Figure 3.10. For the SS103 module in Figure 3.9, we decided that the ROC AUC score was consistently above 0.5 from week 3 onwards, so we started the email students with interventions then. For the MS136 module in Figure 3.10, the predictions were better earlier on (as seen in Figure 3.10), and so we started sending these predictions to students in week 2.

As we saw in Section 3.4, a variety of classifier algorithms and parameter combinations were tried, and we found that a SVM with a linear kernel performs best.

SVMs with more complex kernels and more complex algorithms tend to classify all the examples as the most common case, in this case students passing the mod-

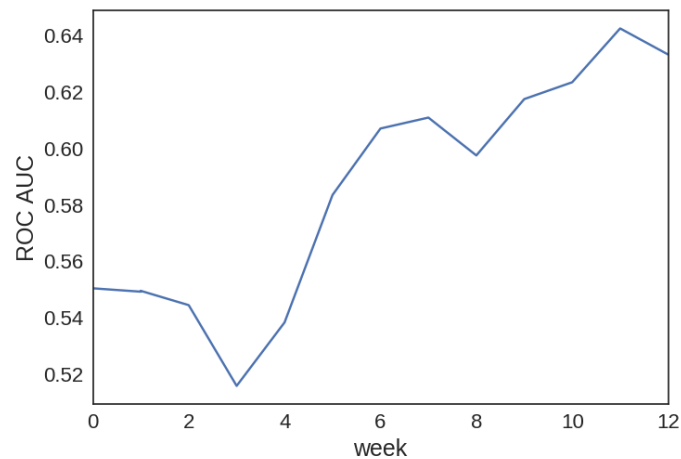


Figure 3.10: Prediction accuracy for MS136 (Mathematics for Economics and Business) by week.

ule. SVMs with a simple linear kernel do not tend to over-fit as much. It was also necessary to rank the students in order of our predictions as to their likelihood of pass/fail. For this to work we modified the SVM implementation to output a confidence score as well as a prediction.

In our results we found that some modules worked better than others. Modules that worked better typically had a high failure rate, such as Mathematics courses. One reason why this might occur is that modules with high failure rates have smaller class imbalances, and so are easier to classify.

Just as it is important to compute the accuracy of how well our predictions would have done in previous years, it is equally important to verify that the predictions that we made were also accurate compared to the actual results that the students obtained at the end of the semester. We did this by making a prediction of whether the student would pass or fail for each week. Once the semester was finished, we compared this to whether the student did indeed pass or fail the examination for that module. From the resulting confusion matrix we calculated the F1-score for each module on a per-week basis, which is shown in Figure 3.11.

We can see from that that most modules received an F1-score between 0.8 and 1 for all weeks. This demonstrates that the predictions that we made were actually accurate. The worst performing module was LG116, which by the 10 had dropped to 0.65 F1 score. This module is “Introduction to Politics” and has a large pass rate, which may explain why it did not perform as well as other modules, given our discussion of class imbalance in these datasets in Section 3.6. The best performing module was BE101 (“Introduction to Cell Biology and Biochemistry”), which

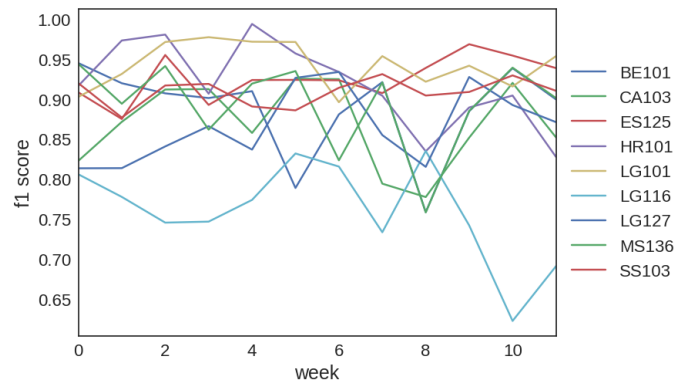


Figure 3.11: F1 scores by week. Each coloured line represents a different module.

reaches an F1 score of 1 by week six. This module has a higher failure rate, and similar to the previous course, this may explain why the performance of this classifier remains so high. One surprising aspect of this graph is that in general, the prediction performance stays high and remains high, with some variation. We would have expected that as more information from students becomes available, that the prediction performance would improve. One thing that may work against this idea is that as more data becomes available for the models to train becomes available, the more opportunity there is for the models to overfit. Each course already only has a couple of hundred data points to learn from at most, and adding extra columns of data adds the potential for more sources of noise.

In terms of performance students who opted into the prediction across modules, in Table 3.4 we can see that in the majority of the courses where the students opted into receiving prediction alerts performed better than those who did not. These are the results from the first year of the study. We can also see across all modules, weighted by the number of students in each, there is an average 3% increase in actual final exam mark between those who opted in and those who opted out.

3.7 PredictED Challenges

One issue with any of the implementation approaches that we could take is that all the algorithms we could use assume that the classification features all come from the same distribution. However, in order to learn from past student behaviour and eventual exam outcomes and to build classifiers, we build models on one year and test on a subsequent year. It is difficult to know how much of this

effect of year-on-year change is present though we have concentrated on course modules where course content is stable and the Lecturer has not changed. However we can at least combat this using cross-validation. By training on some subset of the data and testing on another subset, we can at least get an accurate score of how well the classifier performs. We will however not know how much of the loss of performance is due to the year-on-year changes to and how much is due to the variability of the data.

A more difficult effect to account for is the so called “Hawthorne effect” (McCarney et al., 2007), where students who realise that they are being monitored change their behaviour because of that fact. For example, a student may work harder if they are under the impression that their Lecturer is monitoring their usage of Moodle closely. They may receive a better grade because of this factor, and have nothing to do with the fact that they are receiving weekly results. Alternatively a student may try to game the system by repeatedly refreshing pages to receive good feedback. Because it is hard to account for this effect, particularly when we are providing feedback to students on a weekly basis, we have so far not controlled for this effect. The result is that the effect of the interventions on students may be over-stated.

3.8 Intervention Evaluation

Once the first semester exams had been graded and we received access to the actual exam results, this allowed us to analyse the impact that PredictED had on the first round of students this system was tested on. We first examined the relationship between the scores that students obtained in their Leaving Certificate exams, and whether they opted-into or out of the alerting service. For both conditions the University admission points (known as CAO points) that the student obtained in the Leaving Certificate are plotted. We also plot their grades in the Leaving Certificate exam in Mathematics as that subject is regarded as a strong indicator of academic ability. We can see from Figure 3.13 and Figure 3.12 that there is little difference between those who opted-in to the alerting and those who opted out. This indicates no difference in the entry profiles of participants vs.non-participants overall and therefore there is no spurious relationship between the academic ability of the student and whether they opted-in to PredictED or not.

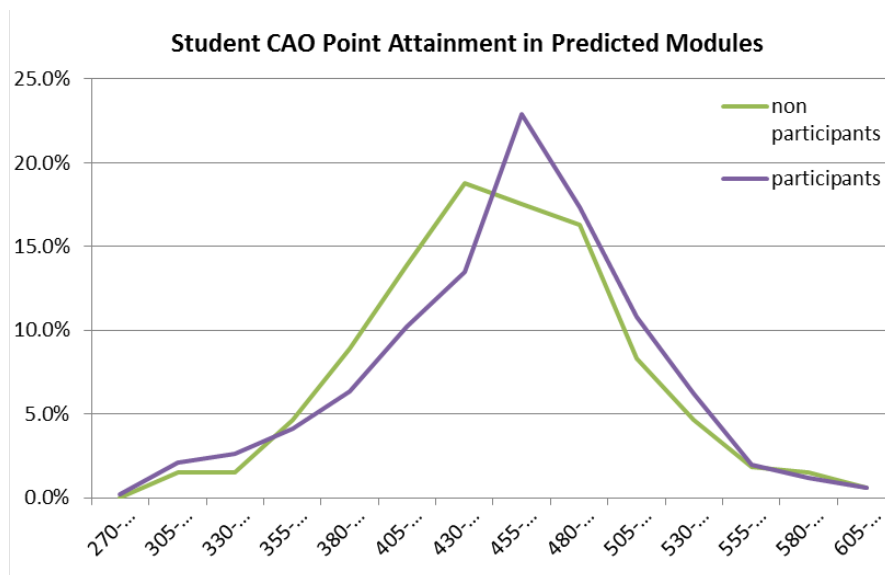


Figure 3.12: CAO Points for participants and non-participants

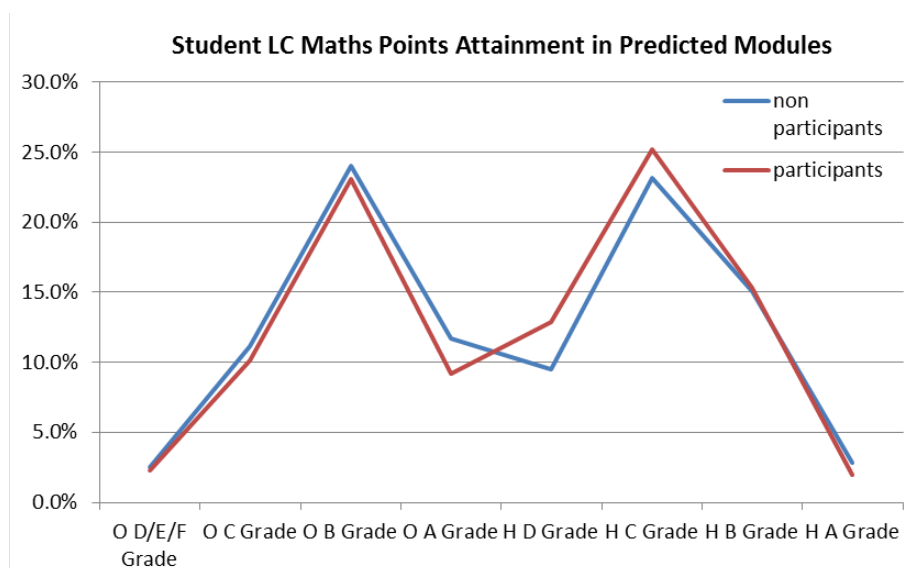


Figure 3.13: Leaving Certificate Mathematics for participants and non-participants.

We also examined the relationship between the predictions that we made, and the actual results that the students ended up achieving overall. We did this by generating a binary pass or fail prediction for each student every week throughout the semester. Once the exams were over and the results available, we compared the predictions against the actual results that students achieved. We then counted the number of true positives, false positives, true negatives and false negatives and calculated the F1-score from this matrix for each week. The F1-score was used due to the highly imbalanced classes present in this results matrix and the F1-score results can be seen in Figure 3.11. We can see from this figure that the F1-score remains consistently high, especially over the latter weeks of the semester.

The most important result that was obtained was the difference in exam performance between those who opted-in to receiving emails each week, compared to those who opted-out. Table 3.4 shows this comparison for 10 of the modules in the first semester of the first year and the last row indicates that for those who opt-in they can expect to see an average increase of +2.67% in their actual exam marks, all other things being equal. The final column of Table 3.4 shows the impact of PredictED on a per-module basis with two checkmarks indicating significantly higher performance for participants (students) and a single checkmark indicating higher performance. This table shows that for 3 modules, BE101, HR101 and MS136, there was a significant improvement in students' exam performance while for 5 other modules there was improved performance. Only 47 students (of a possible 67) registered to take part in PredictED as part of ES125. This low number of registrations for the module as a whole was unexpected but we decided to continue and make the weekly alerts available nonetheless. The average decrease of 0.5% in exam performance could be discounted based on the low N for this group.

3.9 Deep Learning Approach

With the advances in deep learning frameworks and hardware to implement deep learning training directly (Chollet et al., 2015; Abadi et al., 2016) we were able to attempt deep learning solutions to the data and applications described earlier. Two years after we completed the above experiment, for the sake of completion we used a deep learning approach.

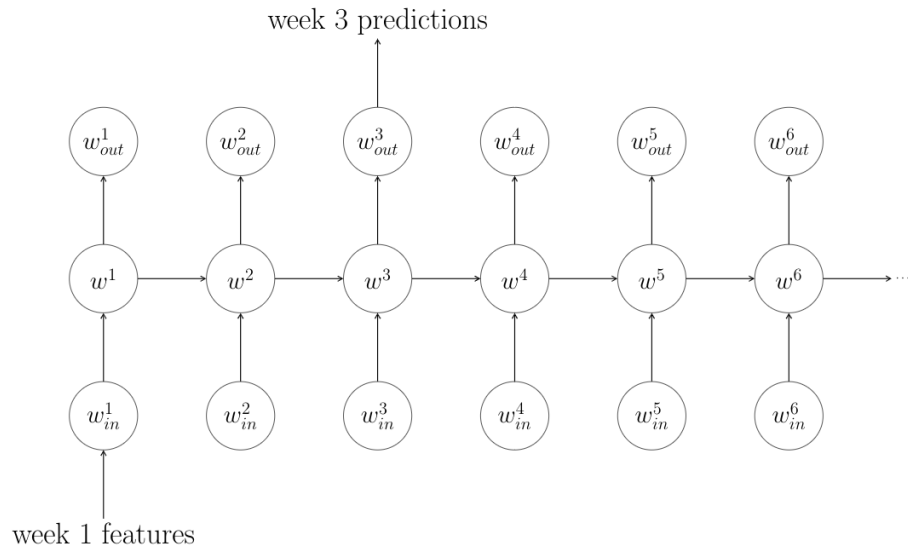


Figure 3.14: Illustration of architecture of RNN applied to predicting students grade.

In our solution we extract very simple features from our data namely the number of times a user accessed Moodle in a given week. We use an LSTM, with the features being the weekly count of times a person accessed their Moodle page. Because our problem involves predictions based on time series and making predictions at multiple steps in time, we used RNN's as we believed they would be a natural fit. An illustration of this architecture can be seen in Figure 3.14. In this diagram we can see that predictions are generated each week, as well as features being entered each week. The prediction from the current week is used as a feature for the next week.

We also changed the task from being a classification of pass and fail to a regression task. The reason for this was that this would give us a more fine grained prediction which we could use to build a more effective intervention system. In Section 3.4 we saw that a SVM performed the best. Here we will again use a SVM, adapted for this new task, to give a baseline against which we can compare our LSTM technique.

In Table 3.5 we see the results of running our regressor across several regression algorithms and parameters. We use the R^2 metric to evaluate the performance of the algorithm. This metric can be interpreted as "the percentage of variance explained by the model".

The models we ran in our experiments were:

- A "Dummy" Regressor which always returns the mean exam results. Our

Table 3.5: Regression Results

Classifier	R^2	p-value
LSTM - Dropout 0.2	0.13547	6.81e-244
LSTM	0.13382	5.36e-238
LSTM - Dropout 0.5	0.13360	3.08e-237
SVM	0.08072	2.30e-87
Dummy Regressor	0.00000	0.99999

classifiers should be at least as good as the Dummy Regressor;

- A Support Vector Machine. This has a ‘C’ hyper parameter of 1.0. with a linear kernel
- A simple LSTM. All of the LSTM’s were run for 300 epochs over the whole data set. This LSTM contained a single recurrent layer, followed by a dense layer with a linear activation function. It was trained using a Mean Squared Loss error, using the ‘Adam’ Optimization algorithm.
- 2 Versions of an LSTM with different values for “Dropout”. This is a technique to reduce over-fitting. During training, this will set a random set of hidden nodes to be ignored. This forces the system to build in redundancy, which reduces the ability of the network to learn features based on noise. The architecture for these models are the same as the above architecture, with the exception that the recurrent layer has a dropout built into it.

From the table we can see that LSTM far outperforms the SVM Regression, explaining 13.3% of the variance of the model, as opposed to 8.1%. We can also see that setting dropout to 0.2 improves the performance of the LSTM marginally. For future work we would like to include more features than simple usage counts, using for example more of the features we extracted in Section 3.3. We describe this in more detail in Section 6.4.

3.10 Conclusion

In this chapter we have introduced the problem of modelling student success. We began by introducing the data available to make predictions and listing some features we extracted from then. We then emphasised the temporal aspect of the

data and developed a method to make predictions on a rolling basis. We demonstrated the intervention system that we implemented for a year and improvements made to it the second year. We showed that this intervention system had a positive impact on student performance.

We then built a new model using deep learning methods based on a Recurrent Neural Network and demonstrated these methods outperformed our previous best methods. As far as we have been able to find, this is the first time that a deep learning model has been applied to VLE student data.

In this chapter we have also conducted our first case study into the the models required for best performance for a time series Machine Learning problem. We have found that in this problem domain, the solution which gives best performance is to use a Recurrent Neural Network with minimal feature engineering. This enabled us to provide students and administrators with predictions on a continuous (weekly) basis with a reasonable performance. In Section 6.2 we will examine this case study in the context of the other two case studies to compare the resulting solutions and attempt to verify our hypothesis.

Chapter 4

Case Study: Application of Machine Learning to an Agriculture Problem

4.1 Problem Description

Our hypothesis is to compare the solutions of multiple applications of Machine Learning to various problem domains on time series. To verify our hypothesis, we require data from a broad range of problem domains. For our next case study, we partnered with farmers in the agricultural space who were able to provide us with a unique source of data. This is a problem which is in a different domain from the other case studies. It also has a time series aspect, because the data comes in as a stream of values. Hence, this is a suitable case study to validate our thesis with.

To give some context to this problem, we will briefly describe the traditional method of weaning calves from their mothers. In a beef farm, the goal is to produce as many calves as possible. When a calf is born, it will naturally suckle from its mother. However, this can be stressful on the mother cow. By preventing the calves from suckling and weaning them from their mother earlier, the mother is put under less pressure, and can more quickly regain her condition. Calves are typically weaned at 8 months old, depending on the condition of the cow. If the calves are weaned too early, then they may become sick and require extra feed to stay healthy.

There are two main methods available to farmers to wean calves:

- Fitting nose spikes to the calf.
- Keep the calves in a separate field from the mother.

Nose spikes worn by the calf work by making it painful for the mother when the calf suckles, hence she will prevent this from happening by kicking the calf and pushing her away. This is seen as cruel, and is banned in many regions.

Keeping the calf separated from the mother has been found to be stressful on both the mother and the calf. In a study done by Arthington and Kalmbacher (2003), it was found that keeping the mother and the calf together reduced stress for the calf and the mother. As a financial incentive, the farmer also benefited, as the grown calves had an average of 20Kg extra weight each. However while it has its appeal, this method of weaning is not scalable, as keeping the calf and the mother together, while preventing the calf from suckling requires a lot of manual intervention.

In this work we try to automatically analyse whether a calf is suckling from its mother. Here we are not forecasting to see if the calves will be suckling in the future, but 'nowcasting' to detect if the calves are suckling at the present moment. This is of value to farmers because it can assist them develop a more humane and cost effective weaning method than the current industry practices.

The set-up for this work was that sensors were worn by some calves from the time that they were a few months old. The sensors included two types of accelerometers (wide range and low noise), as well as a gyroscope. We also arranged to film the calves via CCTV, so that we could get the ground truth as to whether the calves were suckling or not. Finally, the CCTV videos were manually annotated so that every time the calf was suckling it was marked as such.

Working on this case study with domain experts was beneficial as they were able to assist in the labelling process and to explain various patterns for feeding and issues we had with the data.

Our goal was to use Machine Learning to detect if the calf was suckling or not based on the accelerometer and gyroscope data streams and to do this in real time or as close to real time as possible.

Other than the data analytics overlap, there are two key differences between this case study on calf suckling detection and the previous one on educational

analytics:

1. In the current case study we have labels for each time step (the calf is suckling or not), but in the previous case study, the educational analytics application, we only had labels (exam passed or failed), once per semester.
2. The scale of the time measurement is changed from weeks to microseconds. If we can show that this method works across these varied time measurements, given the varied nature of the dataset, then the hypothesis that the methods we described in Section 1 work well, is strengthened.

Given the the first point above we must make a note of how the process will be changed. The Machine Learning classifier will be provided with data for the previous 'k' steps, and the goal will be to predict the likelihood that the calf 0is currently suckling. Here we pose the question of how confident we can be that the cow is suckling.

It is important to the farmer to know in advance when they can expect the calf to start suckling. The less time it takes to alert the farmer that the calf is currently suckling, the more options they have to make some intervention. Hence, the higher the likelihood of an accurate prediction of suckling, the more value they will be able to derive from an automatic classifier. This will allow the end user of this classifier to choose parameters to decide on an appropriate action to take based on them. These will vary depending on the use case. For example, if there is a high accuracy only after the calf has been suckling for a long time, the farmer will not have much time to intervene.

4.2 Description of Data and Target

The data used in this use case was taken from three sources:

1. A low noise accelerometer. The low noise accelerometer had a range of $\pm 2G$, (gravitational constant) and a noise density of $125\mu g/\sqrt{Hz}$ ¹. We found it to be useful for detecting precise movements such as heartbeats.
2. A wide range accelerometer. This sensor had a comparatively wide range

¹<http://kionixfs.kionix.com/en/datasheet/KXTC9-2050SpecificationsRev2.pdf>

of $\pm 16G$, but this was traded off for a higher noise density². This was useful for examining larger movements, such as the movement a calf's head.

3. A gyroscope. The gyroscope was useful for detecting changes in orientation.

For each of these three sensors we had a value for the x-axis, y-axis and z-axis. We also added a magnitude value for each of these values, computed using the standard formula.

$$|v| = \sqrt{x^2 + y^2 + z^2}$$

These three sensors provided by Shimmer were bundled together into a single device and placed around the neck of several calves for periods of up to a few hours. The data was collected over a period of a few months. This dataset was drawn from 28 individual calves, with a total of 3,128 minutes of data which was then annotated. The average number of seconds from each sensor was taken. Later on we measure the impact of leaving out one or more of these features. We did this analysis in order to measure for the company which sensors would be worth it to include. The sensors sampled data at a rate of 256Hz. We had to resample this to a 128Hz frequency in order to align this data with the manual labels for the calf sensors.

The ground truth for when an individual calf was or was not suckling was established by video recording of the calves as they were wearing the sensors. The videos were then annotated manually to establish when the calves were suckling. In addition, some sensors were put on calves who were then just placed in a field with no opportunity to suckle. This was done in order to get a baseline for what a calf's data would look like without suckling.

4.3 Data Preparation and Feature Extraction

Before we started developing the predictions for likelihood of calf suckling, we first had to pre-process the raw data so that it was in a format suitable for analysis. We transformed the data in several ways.

²<https://www.digikey.com/product-detail/en/stmicroelectronics/LSM303AHTR/497-17053-1-ND/6579103>

4.3.1 Saturation of Sensors

When we were setting up the sensors, we ensured that they were not saturated. We analysed the data in order to test for calibration mistakes. We found that some of the data was saturated. This is where there is a large bias in the sensor, i.e. even when it is resting it has a high value. We had to correct for this and remeasure some of the data.

4.3.2 Re-Sampling the Data

Another issue that we dealt with was that the timestamps associated with the different sensors and the ground truth did not match up exactly. The sensors were taking recordings every few microseconds, whereas the ground truth was only manually labelled each second. We solved this by grouping the sensor data into windows of 3 seconds each, with a 1 second step between each group.

4.3.3 Feature Extraction

For each of the windows of sensor data from calf activity, we summarised the data in different ways. Each of the following was extracted for each of the three sensors, and for every dimension (x, y, z, magnitude).

- Minimum, maximum, mean and standard deviation
- Zero Crossing Rate
- Skewness
- Kurtosis

We then took the first difference of the raw values in each of these windows and then extracted the same set of standard features as above from the windows with the first difference window applied.

We also wanted to examine the feasibility of extracting spectral features from the data. Spectral features are features which are derived from the frequency domain, instead of the normal time domain. The Fast Fourier Transformation (FFT) algorithm (Brigham, 1988) can be used to convert between the time domain and the frequency domain and is a well established technique in signal processing. An

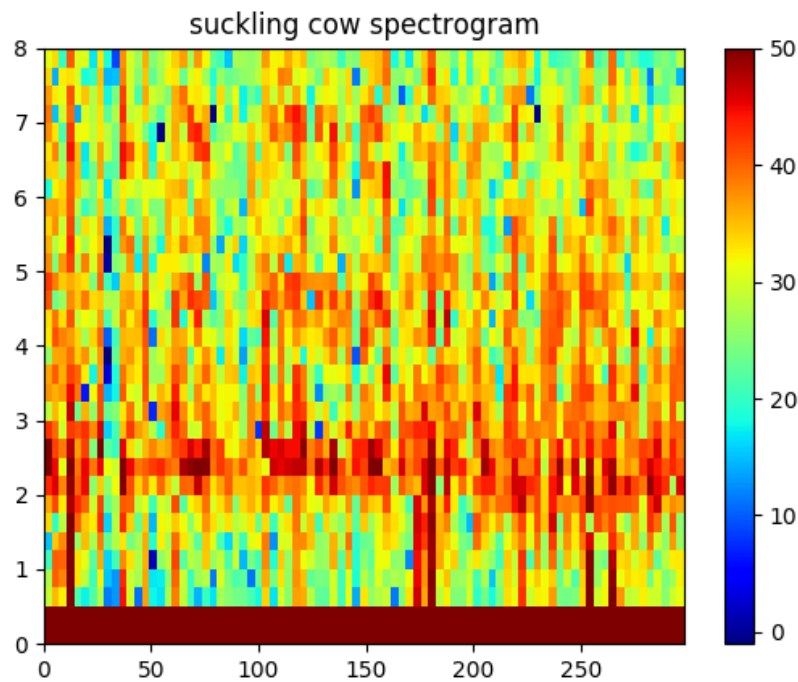


Figure 4.1: Example of Spectrogram from a suckling calf

FFT allows us to pick out features related to the frequencies of signals at a certain point in time, which could give a good insight into differentiating between suckling and non-suckling behaviour.

As a first pass to see if the data would be a good fit to extract features, we plotted the spectrogram of the raw data against the suckling for various values. This allowed us to check if there was some periodic motion that would help when detecting the target variable, i.e. a suckling action.

From Figure 4.1 and Figure 4.2 we can see that there is indeed a pattern. We can see that there is a strong signal in the 2Hz range of motion. This signal is stronger in calves who are suckling than those who are not. This is a good indicator that frequency domain features can be used to detect whether the calf is suckling. In order to capture these features we extracted several FFT features.

Given the results shown in Figure 4.1 and Figure 4.2 we extract the minimum, maximum, mean and standard deviations from the following ranges in the frequency domain and use these as features:

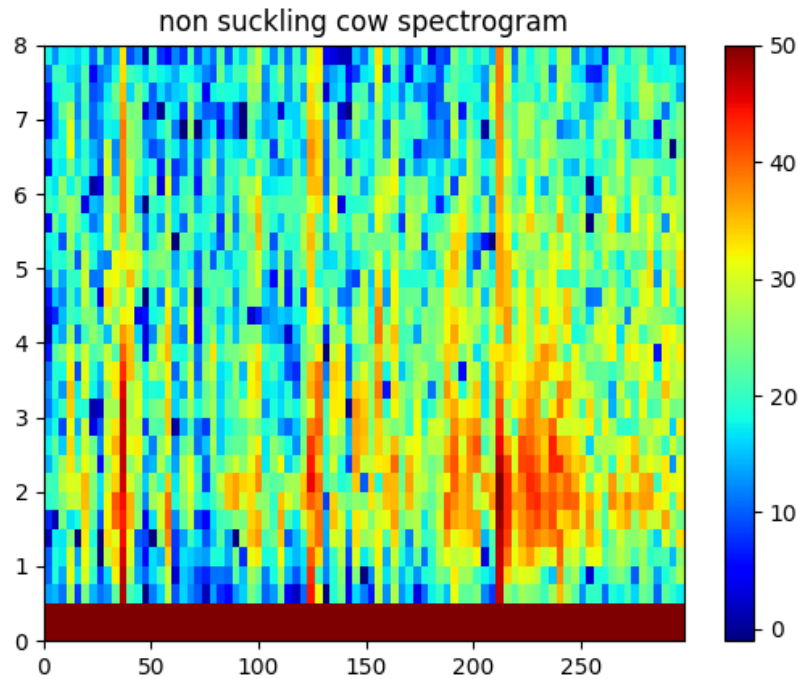


Figure 4.2: Example of Spectrogram from non-suckling calf

- 1-3Hz
- 0-2Hz
- 2-4Hz
- 3-5Hz

Finally, we take the ratio of means of different bands.

Later in this Chapter we compare the performance of the classifier for suckling / non-suckling activities when using the FFT features as opposed to not using them. This is important because it is expensive to compute FFT values on embedded hardware because it is a computationally expensive task. Hence we had to evaluate whether the gain in accuracy when using FFTs was worth making these calculations in terms of the cost of using extra hardware. The results of this analysis can be seen in Figure 4.7.

Once all these steps were taken, we are ready to begin the real analysis of the data.

4.3.4 Classifier Selection

We evaluated a selection of Machine Learning algorithms, the results of which are shown in Table 4.1. We can see from this table that a Random Forest performs the best out of all the classifiers. We can also see from this table that the results of all of the classifiers are quite good. A boxplot of these results can be seen in Figure 4.3. We can see from this graph that the Random Forest has a high average performance, and also has a smaller inter quartile range of ROC scores than competing algorithms.

Table 4.1: Performance of Machine Learning algorithm on suckling detection task.

Model	ROC score
random forest	0.983510
linear model	0.938568
knn	0.913643
linear_svm	0.895173
decision_tree	0.887789

Here the SVMs have a 'C' hyper parameter value of 1.0. We did not use a radial kernel with the SVM as the training time was unrealistically long. The KNN has uses 5 nearest neighbours and a Minkowski metric of 2. The Decision Tree uses a gini criterion, with no pruning or maximum tree depth. The Random Forest uses an average of 10 decision trees, each using a gini criterion with no pruning or maximum tree depth. These hyper parameters are the same as those used in Section 3.4.

We did not explore a deep learning approach in this case study for two reasons

1. The performance of the classifier was already sufficiently high that further improvements through model selection was not necessary.
2. Hardware constraints prevented us from feasibly using a deep learning application in the field. It should be noted that there has recently been some work done on applying deep learning solutions to constrained hardware (Courbariaux and Bengio, 2016), (Han et al., 2015), as well as a number of companies offering deep learning micro controllers (Ionica and Gregg, 2015). However, this was not available at the time we were doing this work.

However, we would like to explore the possibility of applying a deep learning so-

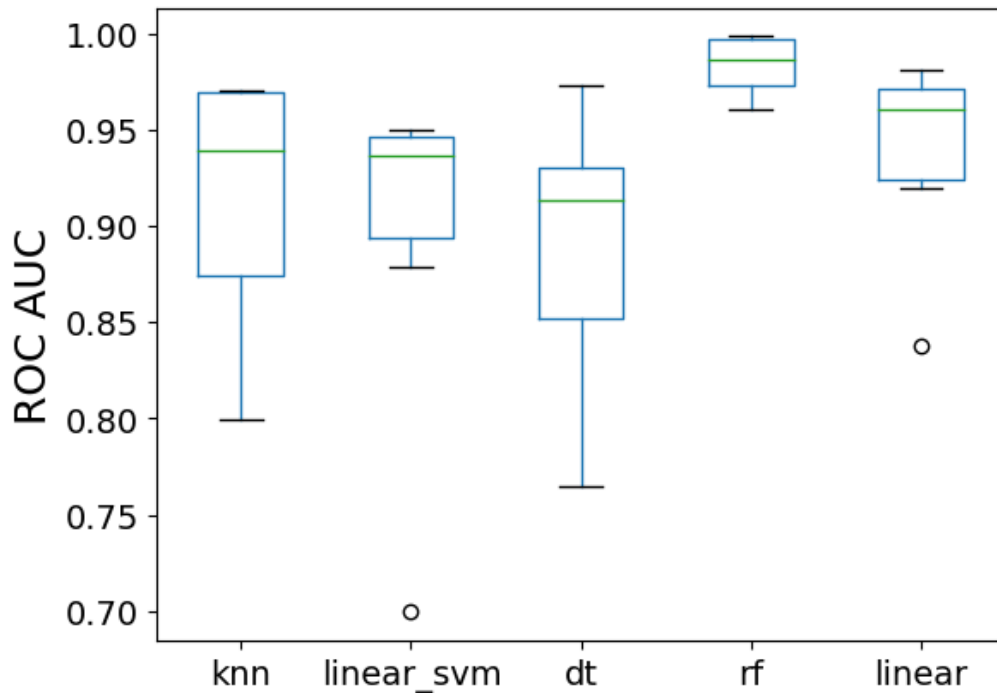


Figure 4.3: Suckling Model Comparison

lution to this topic, and this is described in Section 6.4 on future work.

4.3.5 Cross Validation

The data was gathered from different calves at different points in time. This was because we wanted to ensure that the classifier will generalize across different calves, rather than just learning the features of one individual calf. The way that we did this was to split the calves into a training and testing group, where the same calf would not be present in both groups. We opted for this Leave-One-Calf-Out cross-validation strategy where we would train on all data excluding a calf, and then test on that calf. We repeated this for each calf, which gave us stable accuracy results.

As in the previous chapter, the data was imbalanced because there were far more examples of non-suckling data than there was of suckling data. We dealt with this by re-sampling the data so that positive examples make up a similar proportion of the training data.

As the data was imbalanced, again we use the ROC AUC score as the primary clas-

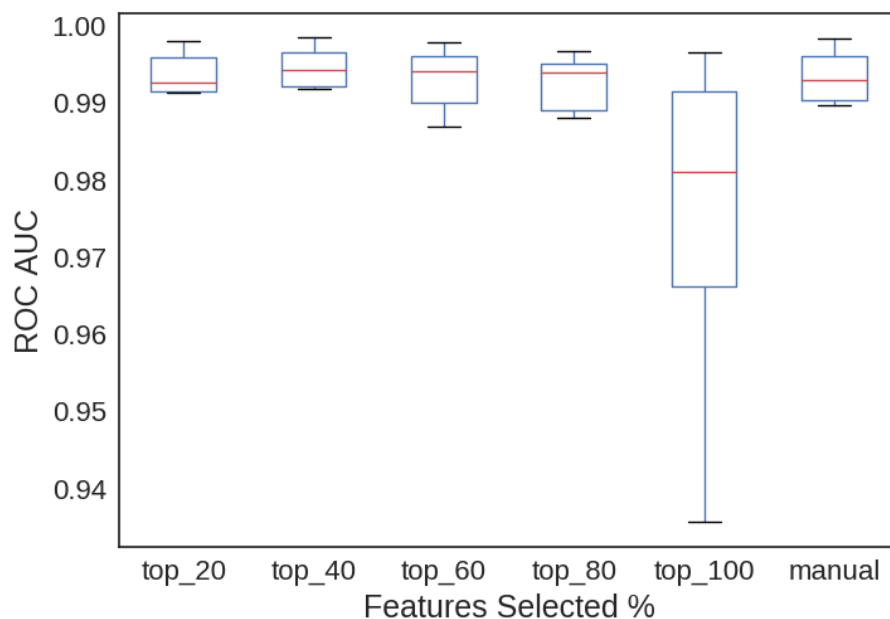


Figure 4.4: ROC feature selection

sifier metric. This is because ROC is a better assessor of classifier performance than accuracy when the data is imbalanced.

4.3.6 Feature Selection

As a first pass, we evaluated the overall accuracy of the generated classifier and evaluated a range of feature selection strategies. We decided to test percentiles ranging from 20% of all generated features, to 100% of the available features. These features were ranked using the ANOVA F-value metric. In Figure 4.4 we can see the effect of trying different numbers of features on the ROC score. From this figure we can see that including all of the features has a negative effect, and that keeping the top 40% of features is the optimal amount. We also compare this to a method of hand-selecting the features based on a manual analysis of the data, and show that it does not perform as well as simply taking the top 40% of features. The manual feature selection process involved manually inspecting graphs of the features and their correlations with the final target variable. We thought that this could outperform the automatic feature selection process, but in all cases taking 40% of the features outperformed it.

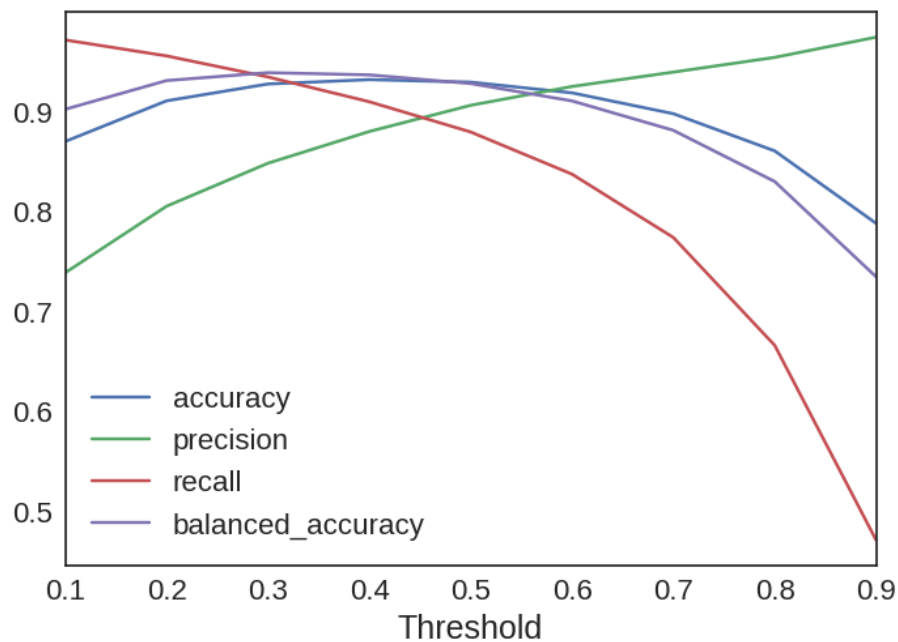


Figure 4.5: thresholds

4.3.7 Time to Detection

One of the important criteria we were given from end users in this use case was having a short time elapse between a calf starting to suckle, and detecting that this is happening. This is a statistic that was emphasised to be important by the end users, namely the time that is taken from the calf beginning to suckle to the first moment of detection. Figure 4.6 plots this against different threshold values to estimate the effect that this has on the detection time. Lowering the threshold of detection will decrease the amount of time taken to detect the first suckling event. However, this will also increase the number of false positives detected in the dataset.

We demonstrate this variable in Figure 4.5, where we plot the accuracy, recall, precision and ROC score as a function of the chosen threshold. This x-axis ranges over the same period as Figure 4.6. In these two diagrams decreasing the threshold below 0.5 gives us shorter times to detection, but also gives us a corresponding decrease in accuracy metrics.

The end user can then use these metrics to choose an appropriate value for the threshold in their application. For example, we can see from Figure 4.6 that if the user wishes to set a hard limit of 5 seconds before an intervention occurs, the

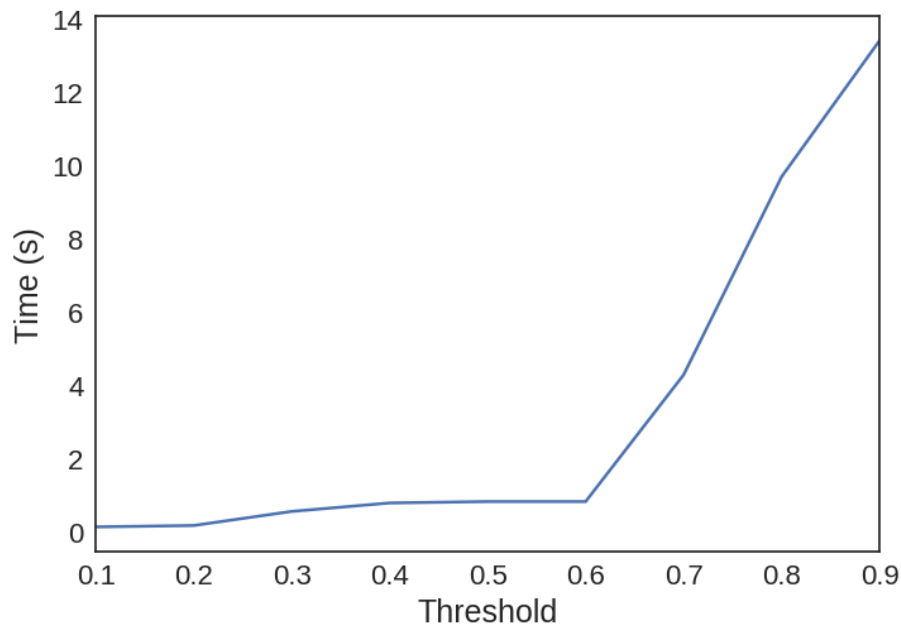


Figure 4.6: time to detection

resulting ROC AUC score is ~ 0.7 . If this can be extended to 10 seconds, then the resulting ROC AUC will be ~ 0.8 . In this case, the slower time to detection may be offset by the improved classifier performance, depending on the context of the application and costs involved in a misclassification.

4.3.8 FFT vs Non-FFT features

Ultimately, it is envisioned that this algorithm will be run on an embedded hardware device. This presents a challenge from an engineering perspective, as the computational capabilities of this device are limited. The improvement in accuracy of features which were expensive to generate must then be weighed against the cost of generating these features. In particular, FFT features are expensive so we must evaluate the performance of our classifier with these features and without these features. We do this by running a cross validation evaluation in two steps, first excluding FFT features and then including them. We then observe the resulting ROC scores. The results of this can be seen in Figure 4.7.

From this diagram we can see that including FFT features greatly improves the performance of the classifier.

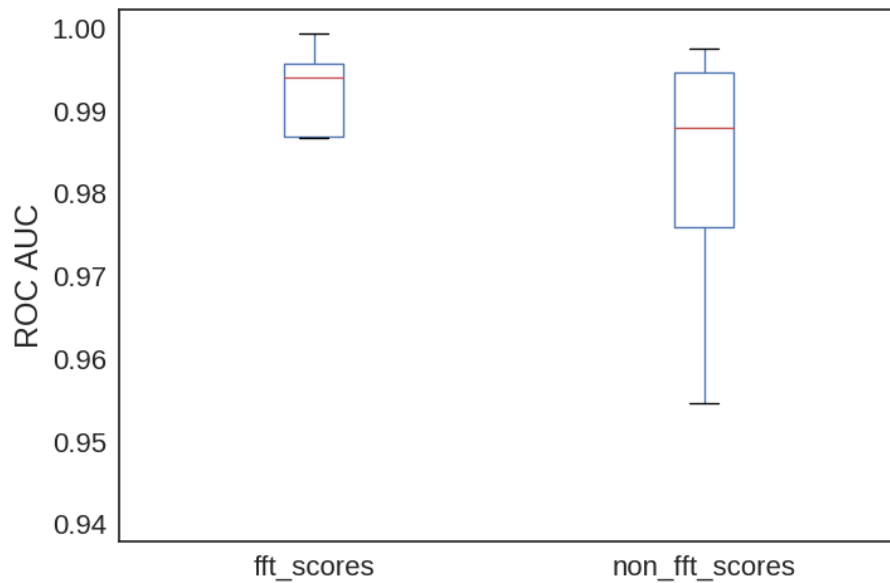


Figure 4.7: FFT comparison

4.3.9 Wide Range Accelerometer vs Low Noise Accelerometer vs. Gyroscope

We have data from three types of sensors available to us.

- Wide Range accelerometer
- Low Noise accelerometer
- Gyroscope

The wide range accelerometer is useful to for capturing large variations in acceleration. This could perhaps be able to detect the motion of a calf's neck going up and down. The low noise accelerometer is useful for micro motion detection. This could for example detect smaller motions on the surface of the calf's skin. The gyroscope is useful for capturing angular velocity. This might be useful for detecting changes in the tilt of the calves head for example.

We compare all combinations of these sensors to get an idea of what configurations work best, and what the various trade offs of not using certain sensors are. (Figure 4.8).

In general, using all the sensors combined together gives the best results. However most combinations work to an acceptable level, except for the gyroscope on it's own.

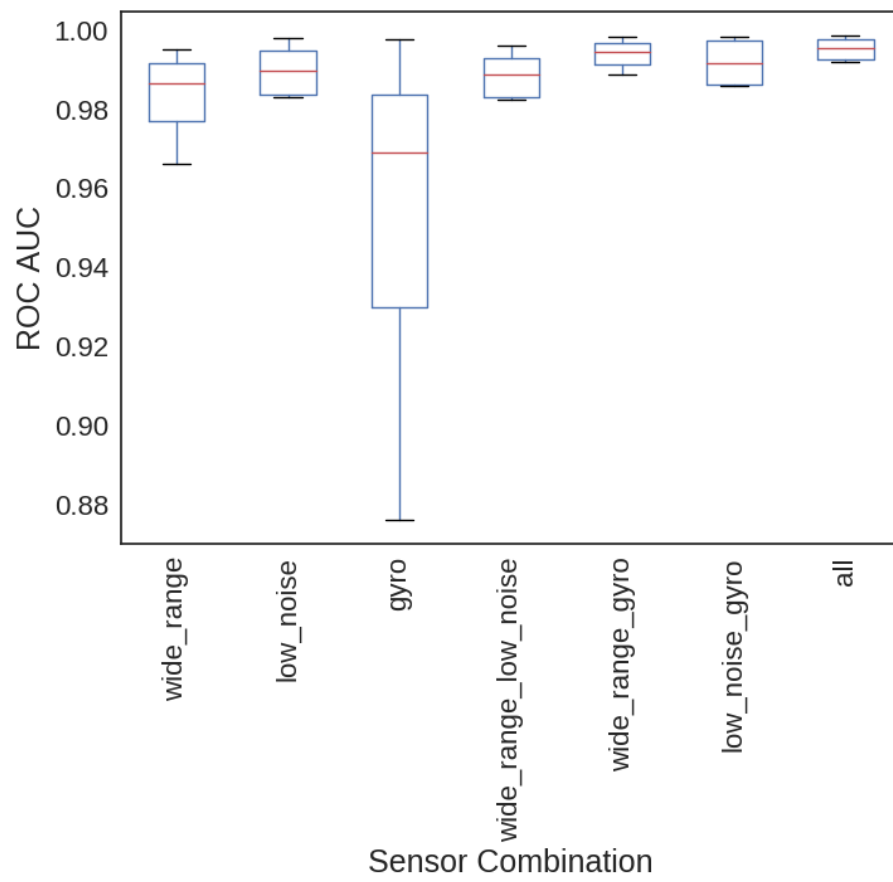


Figure 4.8: Sensor Combinations for calf suckling

4.3.10 Resampling the data to lower frequencies

Recording and processing sensor data at 128Hz is a difficult job for a microcontroller. Extracting features at this speed on limited hardware may prove to be difficult. The microcontroller developer may wish to take recordings at a lower frequency to make the job more feasible. To demonstrate the tradeoffs involved in this decision, we resampled the raw frequency data to a range of lower frequencies (64Hz, 32Hz, 16Hz and 8Hz). We ran the entire pipeline again against this new data, including re-extracting the features from each of these resampled sensor sources. The window parameters (3-second windows with 1 second strides) remained the same. In Figure 4.9 we can see the results of this analysis.

From this diagram we can see that downsampling the data has a negative effect on the resulting accuracy. Any change in the sampling rate must be a tradeoff against the cost of using a higher-spec microcontroller vs the operational costs of using a classifier with worse performance.

4.4 Why a normal approach won't work

The “normal approach” is defined here as a normal Machine Learning pipeline which excludes a time series analysis of the data will not give the best results. This is because it ignores the time series aspect of the data. For example, if we shuffle the rows randomly, as is what happens in cross validation in a typical Machine Learning pipeline, we will lose information. Hence the classifier will only be able to learn an instantaneous representation of whether a calf is suckling or not. It will not be able to learn a model based on what the calf was doing the moment before. We expect that by modelling the time series aspect of the data directly, we can get better results.

We can however, ignore the time series information to get a baseline accuracy that we can compare the time series model to. For example, in a normal Machine Learning pipeline we can train a “dummy classifier” which outputs the mean value of the target variable. We can then use the accuracy that this classifier achieves on a cross validation set as a baseline against which we can compare classifiers. Similarly, we can use a Machine Learning pipeline which ignores the time aspect to give us an indication of what kind of accuracy we should expect to at least beat.

A “normal approach” will also not give us an idea of what the future performance

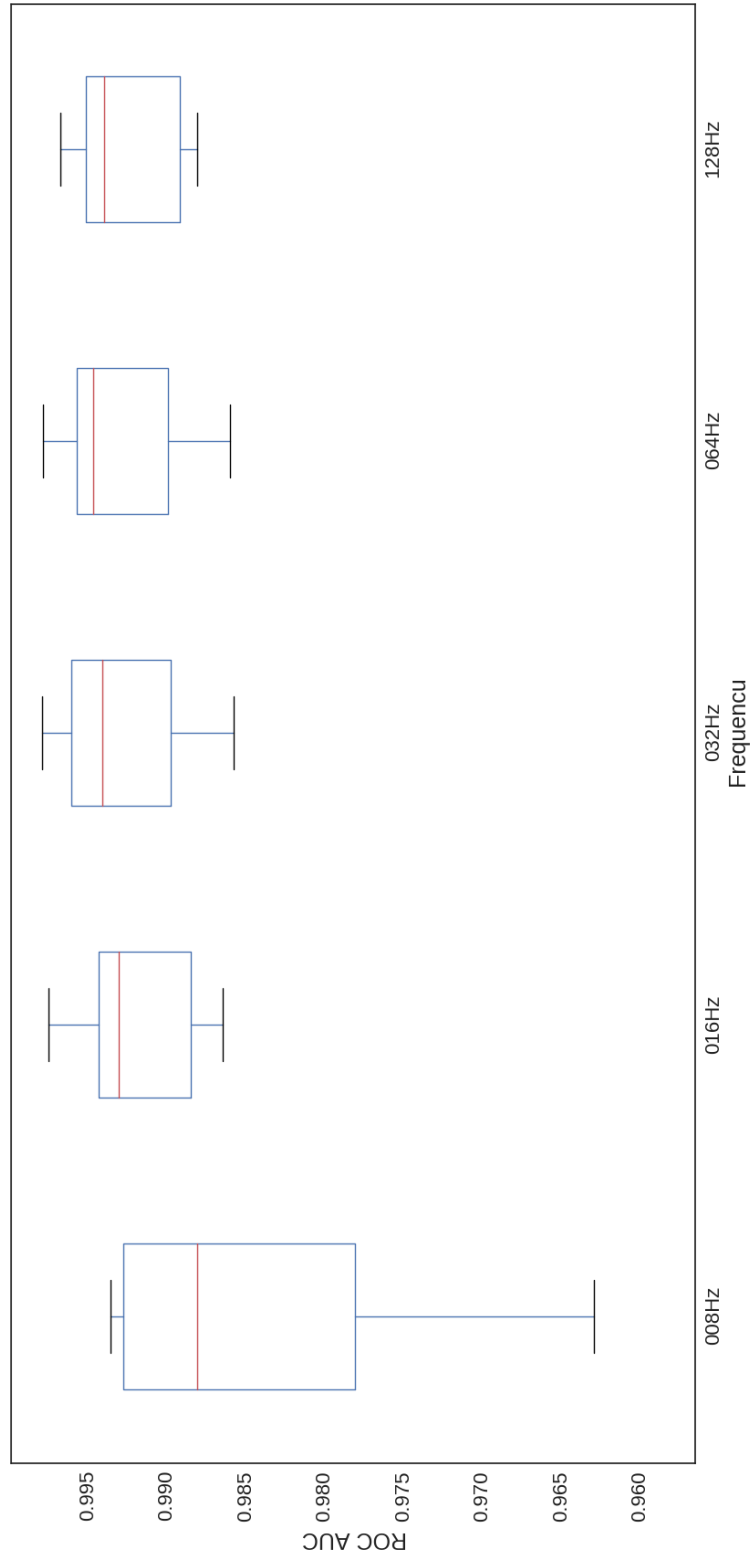


Figure 4.9: Downsampling Results

of the classifier will be. Perhaps the classifier will be able to predict the future accurately, but we will not be able to estimate what the accuracy will be 6 seconds into the future for example. Then end user needs to understand what kind of time frame during which they will have to make a decision. If that time is smaller then they may not be able to take advantage of the extra information delivered by the classifier. This fact must be taken into account when evaluating the overall system. It is important then to estimate the future performance of the classifier.

4.5 Time Series Analysis

To recap, at this stage in our analysis, our pipeline consists of the following steps:

1. Resampling the data down by some factor, depending on the computational capabilities of our hardware e.g. 128Hz, 64Hz, etc.
2. Taking 3 second windows of this data, in strides of 1 second
3. Extracting simple features from each of these windows, e.g. min, max, mean
4. Taking the FFT transform of this window to convert it from the time domain to the frequency domain. We then extract features from this domain.
5. Using cross validation to evaluate the performance. The train-test splits are split by cow. At this step the data is shuffled, so we have lost the time-series aspect of the data.
6. Evaluating the performance of the classifier. The ROC score for the classifier is typically around 0.9, depending on the various parameters decided on at each stage of the pipeline.

This has been useful, as it has demonstrated the following:

- That it is possible to classify whether a calf is suckling or not using accelerometer data.
- Given us a baseline against which we can evaluate our time series analysis techniques.
- Transformed and pre processed the data in a manner which will be useful later.

4.6 Conclusion

In this chapter we examined the question of whether it is possible to determine whether a calf is suckling or not using data from self-worn accelerometer, in real time. We demonstrated the difficulty of this particular environment, and highlighted some of the challenges that exist. We have shown that it is feasible to do this sort of classification, despite these challenges.

We have shown that a non-deep learning approach has performed admirably well for this task, achieving an AUC ROC score of 0.9835 on a cross validation set of a large number of training examples. We examined this performance across a range of parameters that are important for a real world implementation. We showed for example that performance does not degrade significantly when use a single accelerometer, instead of the two that were used in the experiments. We also showed that we can reduce the frequency of sampling from 128Hz to about 16Hz, without suffering a large hit to performance. This allows us to make processing and energy savings in the implementation of the device which will perform the inference. We also examined a method of averaging the signal across a range of time to reduce the number of false positives. We examined the changing the size of the window in this averaging process, and examined the resulting trade-off in time to detection vs. false positive rate.

In this chapter we also evaluated the performance of a number of models (see Section 4.3.4). This is similar to the non deep learning models we evaluated in the previous chapter (Section 3.4). In the previous chapter, the best performing non-deep learning model was a Support Vector Machine, whereas in this case study the best performing model is a Random Forest. This is further evidence that different problem domains require different models, as stated in the hypothesis.

So far we have examined two problem domains, and have determined that the best solution to these two problems have different solutions, supporting our hypothesis. In the next chapter, which will conclude the case studies section of the thesis, we will examine the hypothesis question in context of a combination of GPS and image data.

Chapter 5

Case Study: Predicting Air Pollution Levels from Images using Wearable Cameras

5.1 Problem Statement

In the previous chapters we have demonstrated that Machine Learning can be used for real life problems including helping students improve their grades and assisting dairy farmers to intervene in events related to feeding of young calves. Each of these used Machine Learning but in very different ways, and our next application of Machine Learning will be in another very different domain. What we do in this chapter is describe the application of detecting the quantity of particulates in the air, i.e. air pollution. Particulates are microscopic liquid or solid matter in the earth's atmosphere. In our application, we examine the quantity of Particulate Matter known as the PM2.5 level, in the atmosphere around the level where people live. These particles have a diameter of approximately 2.5 μm , which is about 3% the diameter of a human hair. Increased levels of particulates originating from human activity are associated with higher levels of lung cancer and so knowing human exposure to air pollution like this is an important health concern (Raaschou-Nielsen et al., 2013).

Air quality monitors to measure PM2.5 quantities range in price from €100 upwards. However they are fixed devices and thus measure air quality at a given location only and they are not popular or widespread as portable devices where

their impact in measuring air quality could be greater.

This chapter investigates the potential of using images taken from wearable cameras as a substitute or proxy to estimate the level of PM2.5 particles in the air.

The data used in the experiments in this Chapter were collected by recording air pollution levels in areas around human participants using a wearable PM2.5 sensor. The participants also wore a life-logging camera, specifically the Autographer from OMG¹. This camera is worn on the chest, is front-facing, has a wide-angle lens and captures thousands of pictures per day of wearing. The recordings of images taken with the wearable camera and PM2.5 air quality readings were made available to us by the Barcelona Institute for Health (ISGlobal)² as part of a collaborative project.

We applied Machine Learning techniques to this data to build a model of air pollution levels, using images and GPS coordinates as input. An eventual use case for could be a smart phone application to estimate air quality from phone images and use this to reduce peoples' exposure to particulate air pollution. This could replace a relatively expensive PM2.5 personal sensor device.

5.2 Data Preparation and Cleaning

We now examine what information we obtained from the sensors. This allows us to propose inputs into a series of models developed for pollution detection based on the PM25 sensor, the wearable Autographer camera, and participants in this data-gathering also carried portable GPS devices in addition to the PM25 sensor and wearable camera. Data was gathered by ISGlobal in parts of rural India as part of the CHAI project (Ochieng et al., 2017; Tonne et al., 2017) where people's exposure to air pollution is high because of open air fires used for cooking, diesel generators used for generating electricity, and vehicles used for transport.

The PM25 sensor used for us records the following information:

- date and time
- participantID
- particles in the air (pm25)

¹<http://www.autographer.com>

²<http://www.isglobal.org/en>

- temperature

The GPS data set contains the following data

- datetime
- latitude
- longitude
- elevation

The life camera data set contains the following:

- datetime
- an image
- various setting for the camera

The overall data also contains an identification number for the session the participant was recording (participantID) and the village where the participant is from (villageID). This allows us to merge all these data sets together using the above common data values.

The data was recorded on the three devices at different sampling intervals which caused problems for processing. For example, the lifelogger recorded one image every 30 seconds, and the PM25 sensor created one record every 10 seconds. We take one sample from each of these data sets every 30 seconds, and merge them together. If there is more than one recording in this interval, then the first example is taken. GPS, image and pollution data are joined on participantID, session columns and on the nearest 30 second interval. After merging across the data sources we are left with 241,837 data points.

5.2.1 Participant Activity

For the entire dataset there were 56 individual participants who made recordings across a period of 9 months. This gives us a lot of data to learn from. It also leads to a natural way of performing cross validation where we divide the data set up into a training set, a test set and a validation set such that no individual participant is shared between two sets. This makes it impossible to overfit any

training to a single participant based on his/her lifestyle. The number of participants making recordings each day varies. As seen in Figure 5.1, there are between 0 participants making recordings per day (for example, in the period November, December) up to a maximum of 4 different participants, and between 20 and 30 individual participants making recordings in a given month.

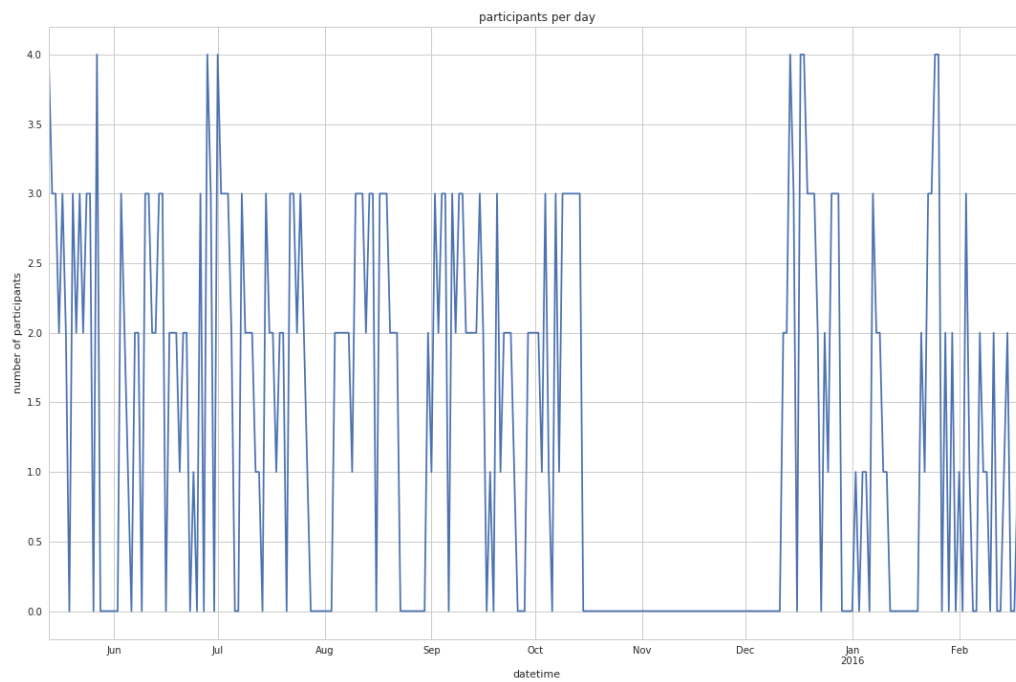


Figure 5.1: Participants Per Day

In Figure 5.2 we can see in which weeks participants were most active in terms of recording. For example in week 38 of 2015, participant 62118 made over 2,000 data recordings. Given that each of the aggregated data points were 30 seconds apart, this means that s/he recorded for approximately 16 hours that week. A similar chart is shown in Figure 5.3, but broken down by month instead.

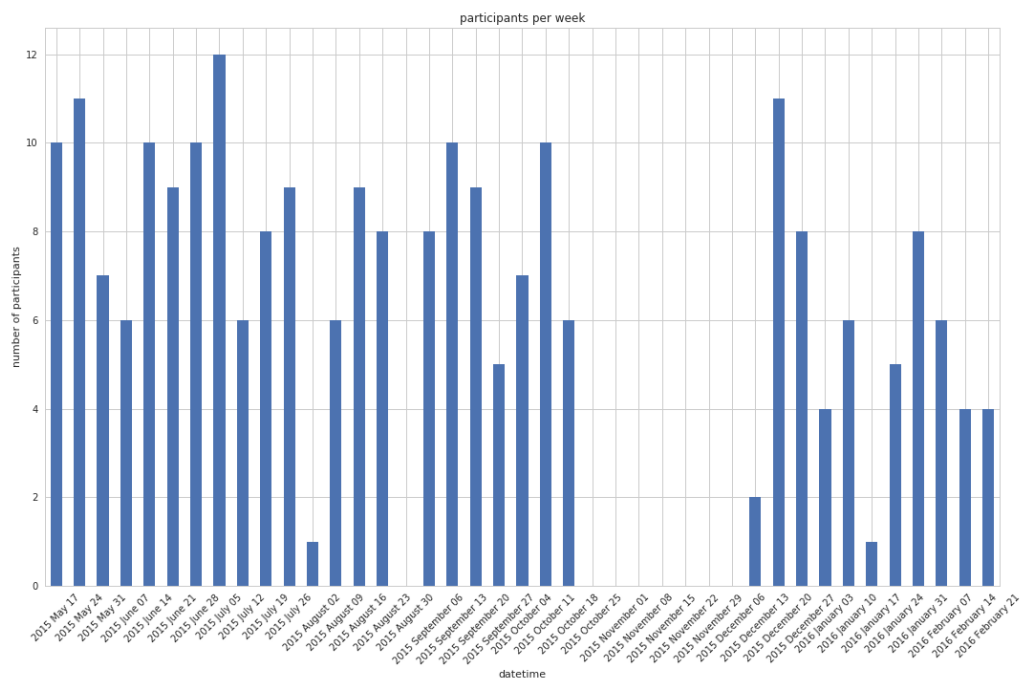


Figure 5.2: Participant Count by Week

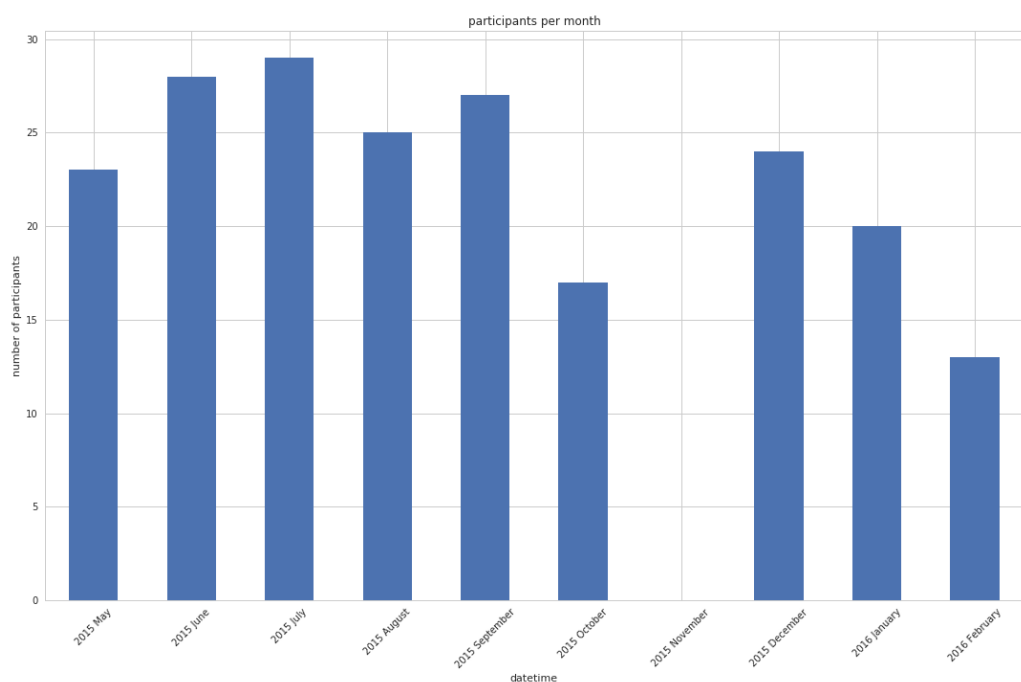


Figure 5.3: Participant Count by Month

5.2.2 Images



Figure 5.4: Example image taken from wearable camera

An example image taken from a participant's wearable camera is shown in Figure 5.4. The image resolution at time of capture is 1936×2592 , with 3 channels (RGB) used. We reduce this down to 224×224 for input into deep learning models later because the fine-grained resolution of the image is not needed. This processing turns landscape ratio images into square which warps the presentation of the image when viewed but these images are not meant to be viewed by the human eye and so long as the squaring of image ratios is used for both training and testing images, that is acceptable.

5.2.3 Properties of Datasets

We now examine some of the characteristics of this dataset. The first thing we look at is the distribution of the air particle values which can be seen below in Figure 5.5. Note that this is a log scale on the y-axis. We can see from this that the vast majority of readings are of the order 10-100, with a few very large outliers. From this diagram we can see that approximately 100 of the 241,837 data points have readings which are larger than 5,000. These outliers will cause problems when fitting the Machine Learning model later, and we will see later how carefully evaluating different objective functions will alleviate this issue. In particular, a

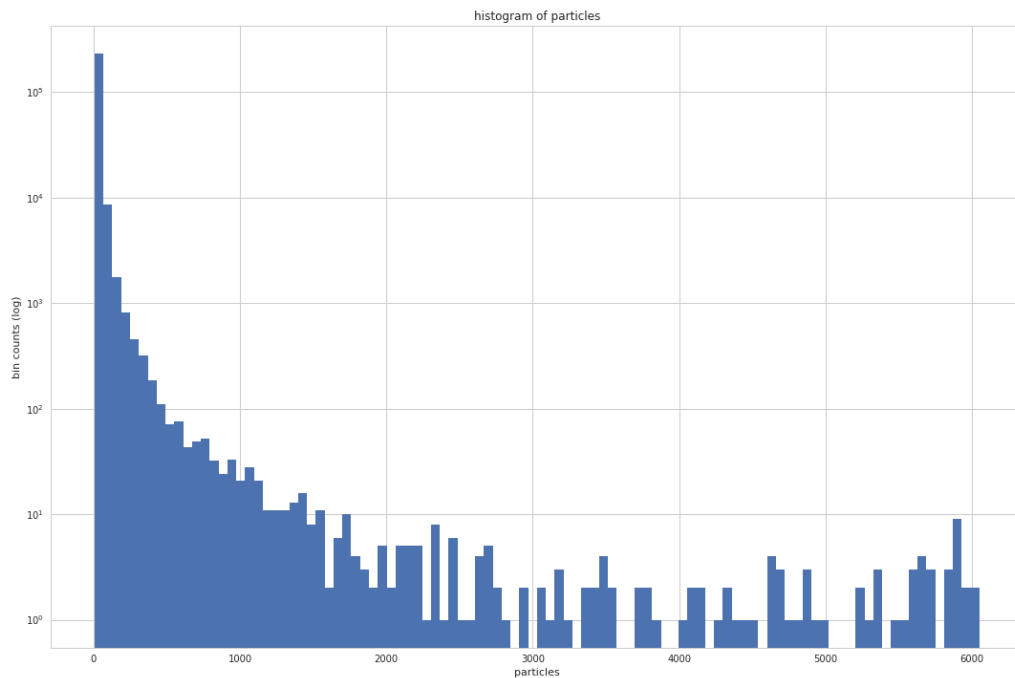


Figure 5.5: Histogram of Particle Counts

regression may not work very well on this data, due to the number of, and size of the outliers.

We next examine the distribution of particle values which fall into a range smaller than the 95th percentile, shown in Figure 5.6. The 95th percentile in this case is 67. Note that the y-axis here is not on a log scale which allows us to examine the shape of the distribution without it being skewed by extremely large values. Also, overlaid in red in Figure 5.6 is a Kernel Density Estimate of the distribution. We can see that the majority of points are between 20 and 40.

The final view of the air particle data values we look at is where we plot a histogram of the distribution of values with many small bins as shown in Figure 5.7. Again we have removed values over the 95% percentile so we can view the distribution without it being skewed. We can see from this image that most values take on a discrete set of values. This indicates that the air quality sensor only reports values to a certain level of accuracy and from Figure 5.7 we can see that this is approximately 0.1. This defines a lower bound on accuracy when we are evaluating the performance of the Machine Learning algorithm.

The key statistics in this distribution are shown in Table 5.1

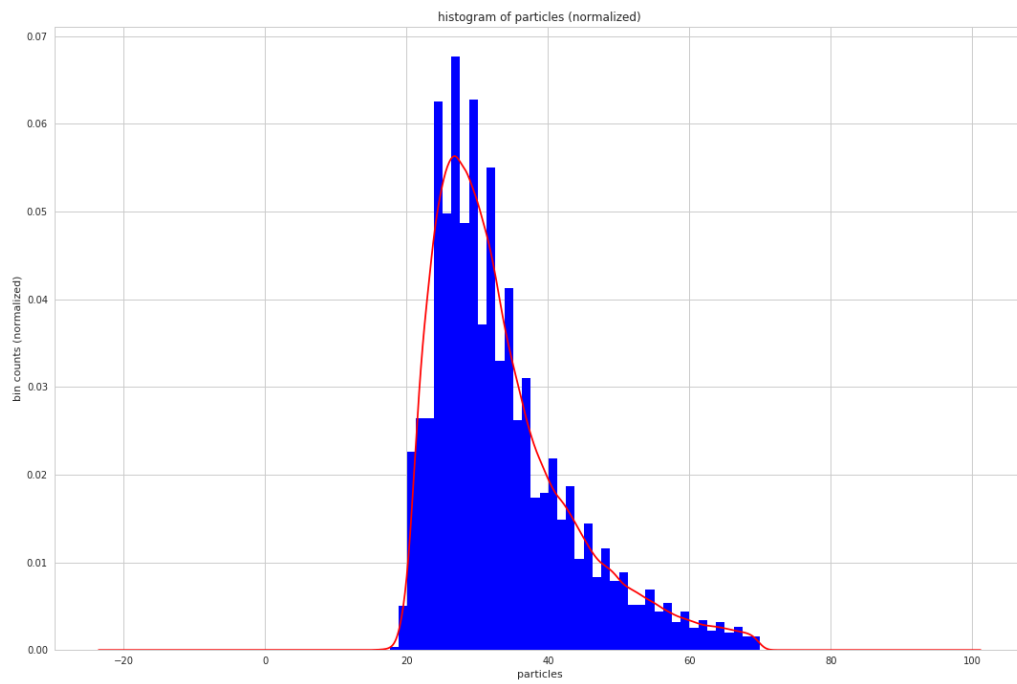


Figure 5.6: Normalized Histogram of Lower 95th Percentile of Particles

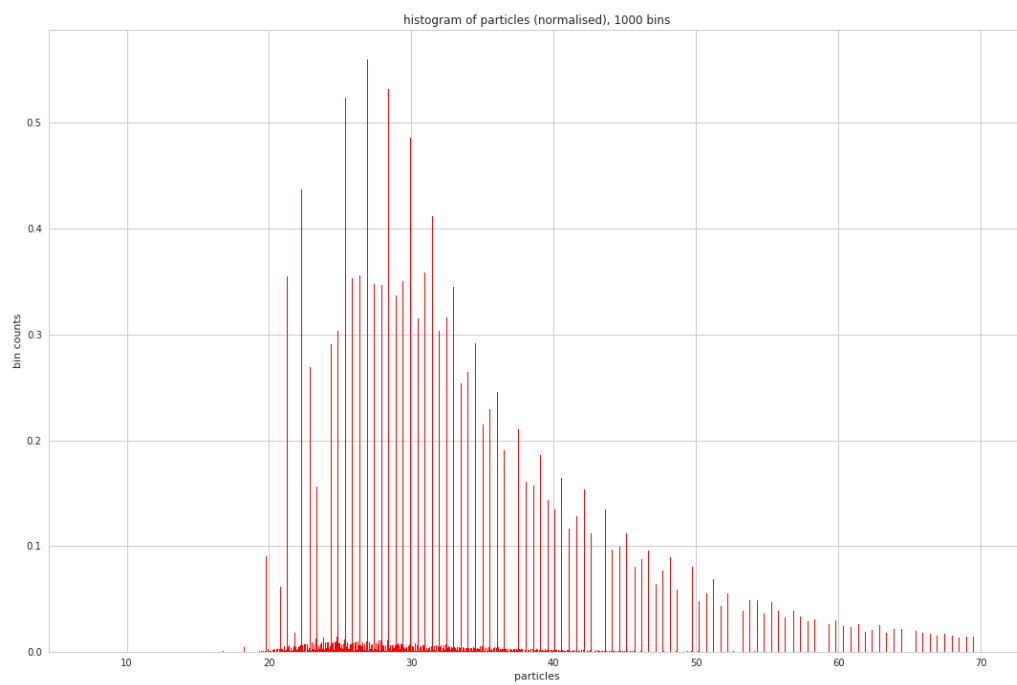


Figure 5.7: Histogram of Particles distributed into 1,000 bins

Table 5.1: Distribution of pollution particle counts

Statistic	Number of Particles
Count	241,837
Mean	42.16
Std	107.79
Min	7.61
25%	26.39
50%	31.46
75%	40.09
max	6,051.43

5.2.4 Pollution Levels by participant

In Figure 5.8 we plot the mean number of particles observed per participant. We can see there is a large variation of particles observed per participant meaning that some get quite little and some others quite a lot of exposure to poor air quality. In Figure 5.9 we plot the mean pollution levels across participants. From this graph we can see that participants 36715, 50860 and 61977 each registered very large outlier values for pollution levels. When we compare the variance with participants to the variance of the whole data set using a one-way ANOVA, we get an F-statistic value of 54.45 and a p-value of 0.00. This means it is unlikely that the means of each of the participant are the same, and the variance is not due to sampling error.

However, this interpretation could be overstating the difference between participants, because each measurement is not independent. For example, if pollution is high at one minute, it will probably be high the next minute too. One way of dealing with this is to take the mean of each recorded session before calculating the F-statistic. This gives us an F-statistic value of 1.24 with a p-value of 0.15. This shows that the variance within participants is still greater than the variance of the whole dataset, but our confidence of that being due to sampling error is reduced. This is apparent when we view the boxplot of pollution per participant which is averaged across sessions as shown in Figure 5.10. Here we can see that there are similar levels of pollution, but with greater variance of values.

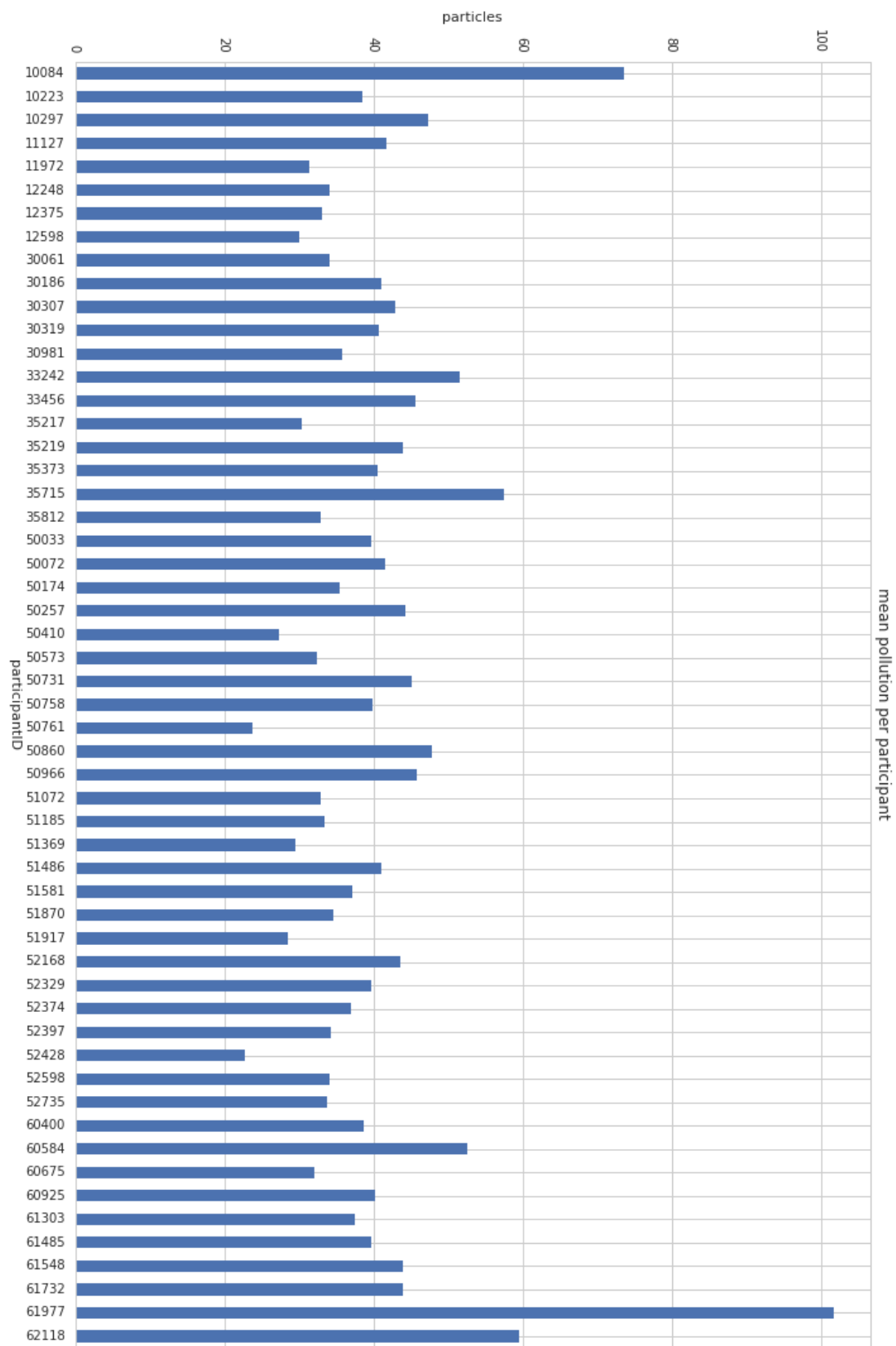


Figure 5.8: Mean Particle Counts per Participant

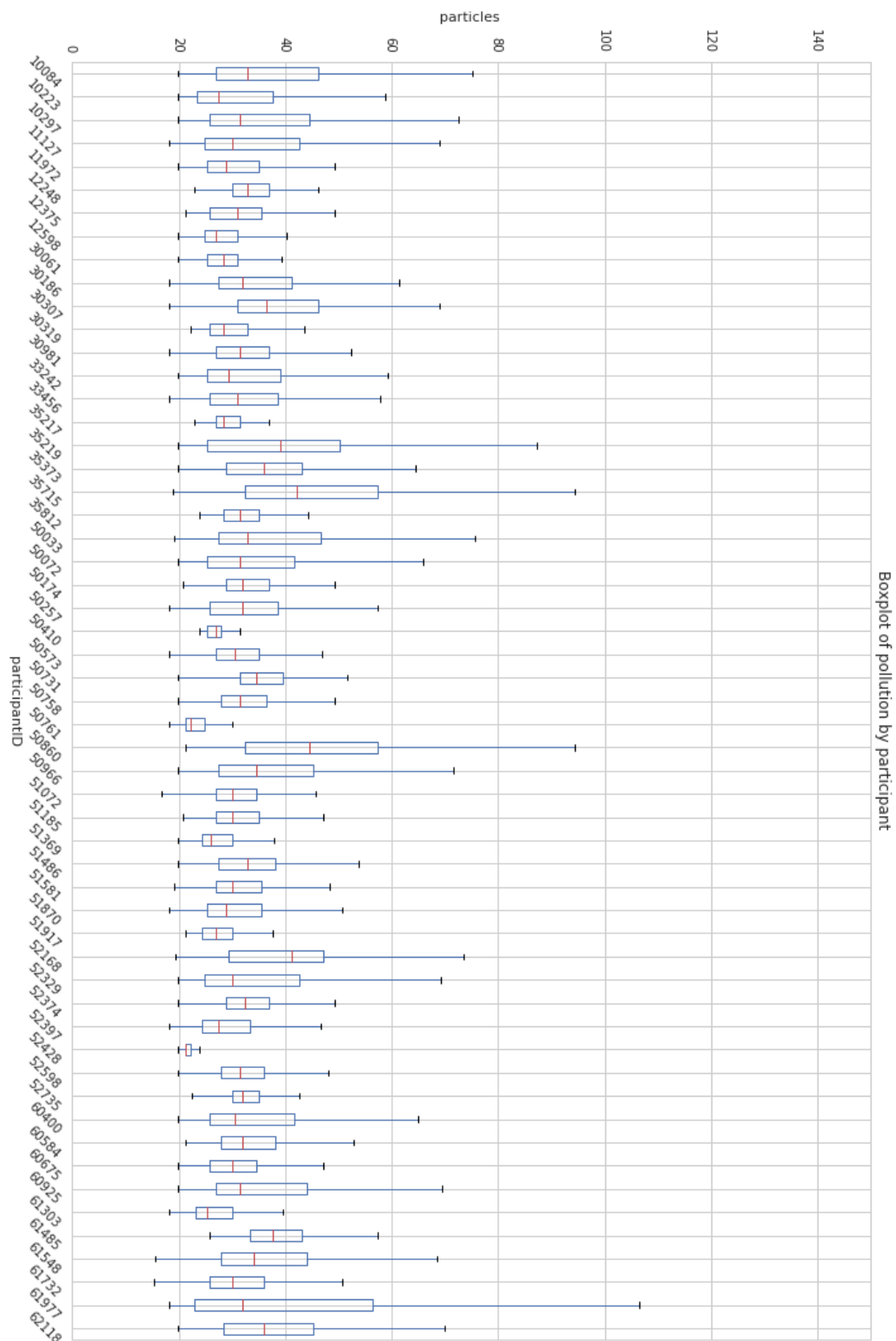


Figure 5.9: Particle Counts per Participant Boxplot



Figure 5.10: Particles per Participant Boxplot – Averaged Over Session

5.2.5 Pollution Levels by village

The data we use in this Chapter is drawn from participants drawn, in turn, from a number of villages in rural India as part of the CHAI project. We calculated the pollution levels across each of the 34 villages where recordings were made and this is shown in Figure 5.11. A lot of the variance of pollution levels seems to depend on which village the recording is made in.

Results of the one-way Anova similar to that carried out when examining pollution levels by participant, are that the F-statistic value is 33.93 and the p-value is 2.92×10^{-175} . So this result confirms a lot of variance is explained by which village a recording is made in, but less than by which participant is making the recording. Again we remind the reader that the results are over-exaggerated because measurements are not independent. If we consider that these observations are not independent, and average across sessions we get the following results for an Anova, with the F-statistic=0.58 and the p-value=0.94. This actually contradicts the previous results. So, overall, it is unlikely that there is a large difference in pollution level caused by being in a different village. However, to verify this we would need more independent observations in different villages, but this is outside the scope of the data collection.

An air quality index (AQI) is a number on an international scale which is used to describe how polluted the air is. There are six AQI categories in decreasing order of quality and these are *good*, *satisfactory*, *moderately polluted*, *poor*, *very poor*, and *severe* and these combine different aspects of air pollution including particulates as well as chemical pollutants like carbon dioxide and others. The values of PM_{2.5} which correspond to the different categories are described in Table 5.2.³

The actual ranges of values of these categories indicate that our participants are experiencing air pollution which is occasionally very poor but which for the most part is satisfactory, bordering on moderately polluted.

5.3 Experimental Method

The ultimate aim of the work in this Chapter is to use Machine Learning to predict air pollution levels from imagery taken from a wearable camera, and possibly including GPS. Below we document the Machine Learning pipeline that we devel-

³This is taken from https://en.wikipedia.org/wiki/Air_quality_index

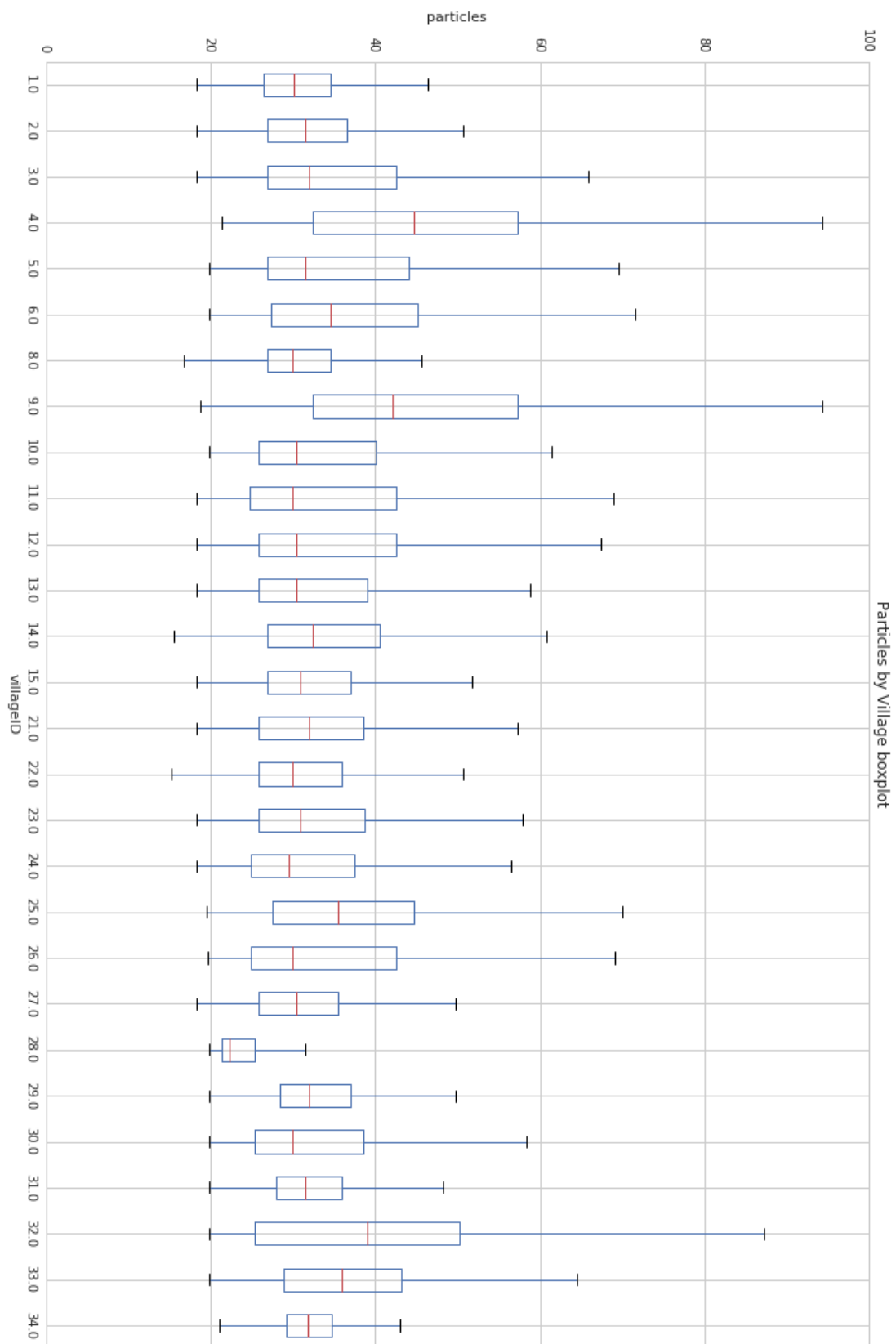


Figure 5.11: Distribution of Air Particle Values per Village

AQI Category	PM.25 (24 hour)	Associated Health Impact
Good	0-30	Minimal
Satisfactory	31-60	May cause minor breathing discomfort to sensitive people
Moderately polluted	61-90	May cause breathing discomfort to people with lung disease such as asthma, and discomfort to people with heart disease, children and older adults
Poor	91-120	May cause breathing discomfort to people on prolonged exposure, and discomfort to people with heart disease
Very poor	121-250	May cause respiratory illness to the people on prolonged exposure. Effect may be more pronounced in people with lung and heart diseases
Severe	250+	May cause respiratory impact even on healthy people, and serious health impacts on people with lung/heart disease. The health impacts may be experienced even during light physical activity

Table 5.2: AQI categories and health impacts

oped for this task, and we present the results achieved.

5.3.1 Standard Machine Learning Pipeline

We start by developing an approach which does not use images to predict air pollution levels, but simply uses the location information. This will give us a baseline score, against which we can evaluate our deep learning models for image analysis, in the next section. We use the following features:

- Longitude
- Latitude
- Elevation
- Temperature

A standard scaling is applied to each of these features. In addition, we use the villageID value, with a one-hot encoding. There are 28 unique villageIDs that we use and this leads to 32 features in total, with 241,837 total measurements.

We applied a variety of standard Machine Learning algorithms including the following

- Linear regression
- lasso and ridge regression
- Random Forest
- Extreme Gradient Boosting
- Linear regression with different objective functions (e.g. Poisson and mean absolute error)
- And finally, a dummy regressor, which always returns the mean of the dataset. This can be used as a baseline against which the other regressors should perform at least as well.

Here the Ridge and Lasso regressions both uses an alpha value of 1.0. The Decision Tree uses a gini criterion, with no pruning or maximum tree depth. The Random Forest uses an average of 10 decision trees, each using a gini criterion with no pruning or maximum tree depth. The extreme gradient boosting algorithm uses a step size shrinkage of 0.3, a maximum depth of 6, an alpha value of 0 and an L2 regularization term of 1.

We used a variety of metrics to evaluate the performance of the predictions, the main one being Mean Absolute Error. We also examine values of Mean Squared Error and R2 score, however both of these give poor scores, as there are large outliers in the data (up to 6,000 in total), and so any predictions are heavily penalised for getting outliers wrong. The fourth metric we looked at as Median Absolute Value, which also penalises outliers though less than Mean Absolute Error does.

We split the data into a training set, a validation set for choosing hyperparameters and a testing set for evaluating the results. We split the data based on participantID to avoid over-fitting. Because the results can change depending on which participants were assigned to which group we shuffled the data 10 times and reported the average score, as well as using these to plot bar plots. The results of this are displayed in Table 5.3.

Table 5.3: Results when using the Standard Machine Learning Pipeline

Regressor	Objective	Mean Absolute Error
Linear Model (keras)	Mean Absolute Error	15.229376
Lasso	Mean Absolute Error	18.150236
Dummy	Mean Absolute Error	18.299901
Gradient Boosting (sklearn)	Mean Absolute Error	18.774428
ElasticNet	Mean Absolute Error	19.615230
Gradient Boosting (XGBoost)	Mean Absolute Error	23.788019

Regressor	Objective	Mean Absolute Error
Linear Model	Huber	38.512339
Linear Model	Poisson	1075.813931

One interesting result from this table is that 5 of the models under-performed a dummy model. This indicates that there is large potential for overfitting. This is due to the fact that a small amount of the data have comparatively large values. Our solution to this was to not penalize by the square of the deviation, as is the most common method (Mean Squared Error). Instead we trained and evaluated our methods using the Mean Absolute Error. An alternative solution is to predict the log of the pollution levels. The most appropriate choice of error depends on the application of this model, and whether we care more about getting more predictions right (MAE), or about predicting extreme outliers correctly (log predictions).

Because there is so much variance in the results we created 10 different training-test-validation splits with participants randomly assigned to a different set in each of these iterations. We plot a boxplot of results for each classifier used which is shown in Figure 5.12. We also plot the results across 4 different evaluation metrics (Mean Absolute Error, Mean Square Error, Median Absolute Error, R2). From now on we will only be interested in Mean Absolute Error (MAE).

From these results we can see that the best-performing classifier is a Linear Model using a Mean Absolute Error objective, which achieves a MAE of **15.22**. This is our baseline that we want to beat using the image models to verify that they work. It is also important to note that the dummy regressor scored a MAE of **18.29**, and that five of the regressors did not outperform this.

5.3.2 Training Air Pollution Levels on Images

In this sub-section we will describe the methods we used to estimate the level of air pollution from images taken from wearable cameras. We will compare various Machine Learning models and their relative performances. However, first we will first describe the various methods we used to estimate pollution levels from images, so we can better understand the trade-offs and results.

As we saw in Section 1.3.4, Neural Networks are a modelling technique which involve stacking many layers of simpler models into a network. For example, the

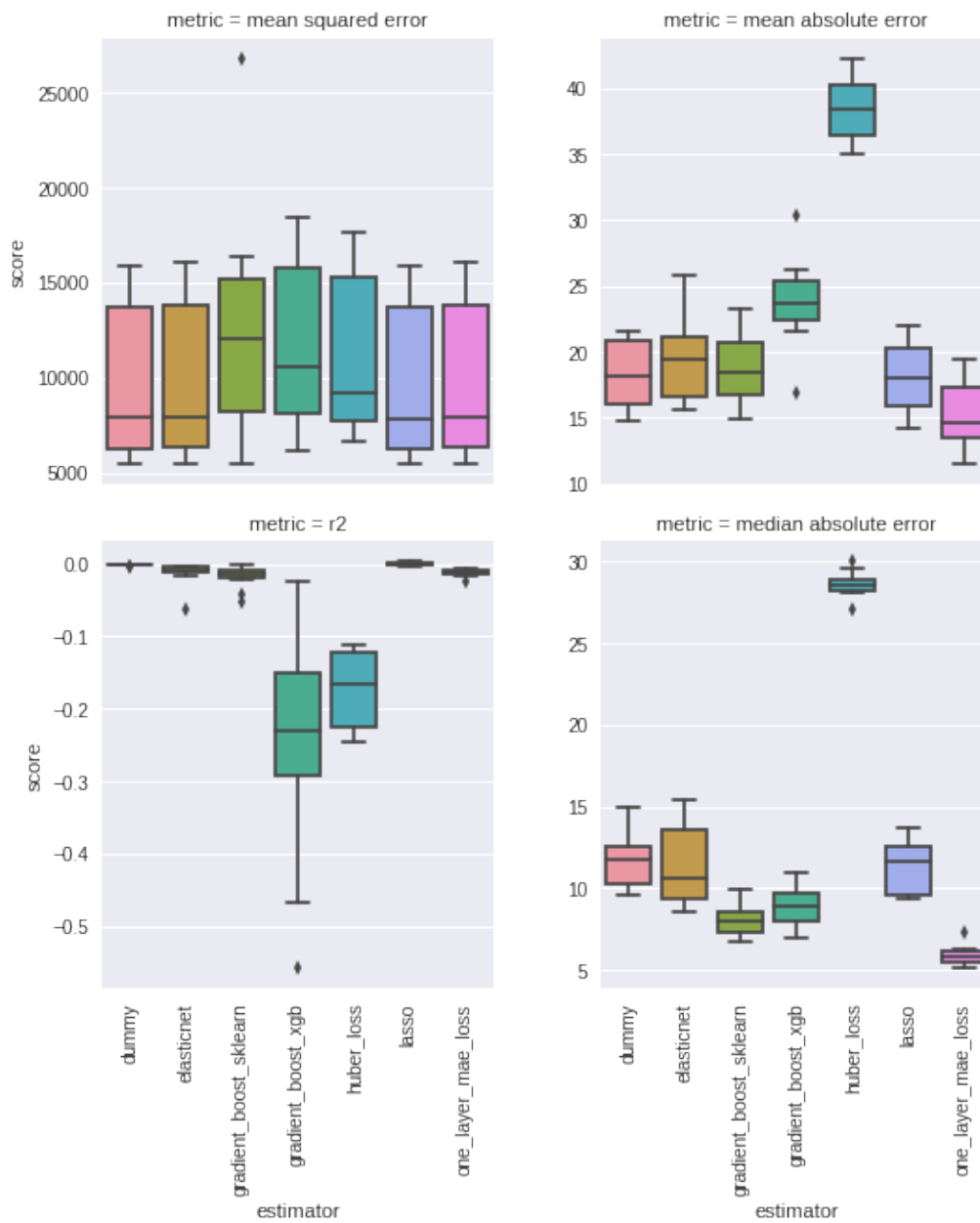


Figure 5.12: Standard Machine Learning Results Boxplot

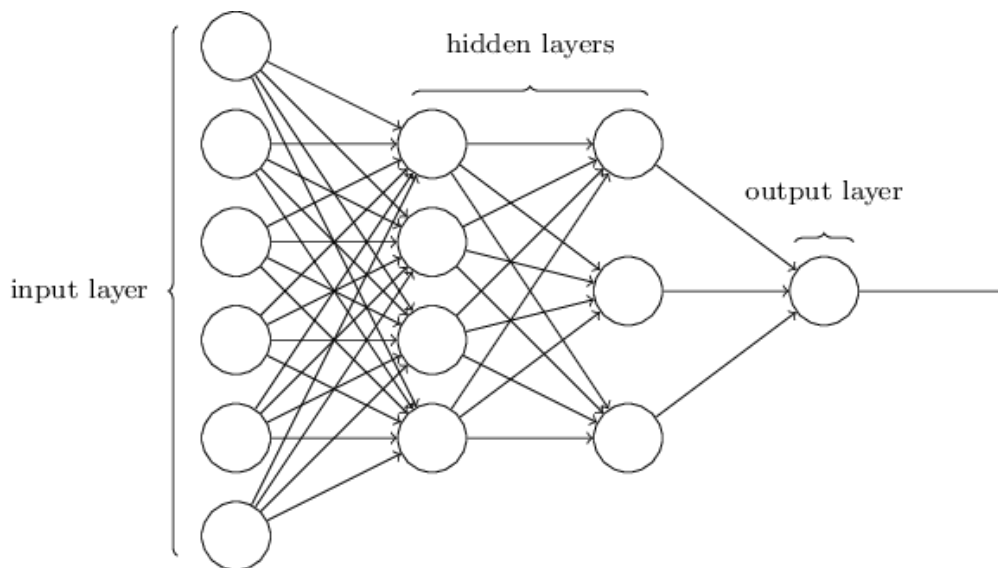


Figure 5.13: Hidden Layers in a Neural Network

inputs to the model could be passed to a logistic regression, the outputs of these can then be passed to the inputs of another logistic regression which finally outputs a value for the target variable. The network then learns a non-linear function that would be impossible to learn if we had only used a single logistic regression. An example of a simple neural network is shown in Figure 5.13⁴. Pooling and dropout layers can be used to reduce overfitting, similar to using a regularisation term in a least squares regression.

Convolutional Neural Networks (CNN) are an extension of the simple neural network and have been shown to do well in classification tasks on images (Ciregan et al., 2012). Figure 5.14⁵ illustrates how deep networks work in image classification. They work by taking small squares over an image, and using these as inputs to a neural network. The layers in the neural network then learn hierarchical concepts. For example lower layers learn to identify edges, and higher levels identify concepts like faces.

One advantage that neural networks have over other Machine Learning approaches is that they can be retrained to do a variety of similar tasks, a process which is called transfer learning. It can take weeks using specialised hardware like Graphics Processing Units to train simple classifiers to perform to the level of the state of the art using large image sets for training. However, once the

⁴Taken from <http://neuralnetworksanddeeplearning.com/chap1.html>

⁵taken from [http : / / www . wildml . com / 2015 / 11 / understanding-convolutional-neural-networks-for-nlp/](http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/)

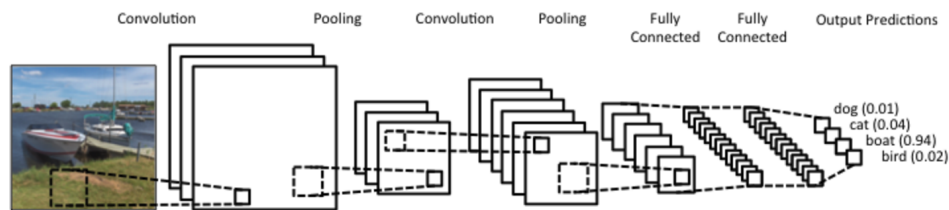


Figure 5.14: Example of a Convolutional Neural Network

network model is learned, we can adapt it to perform similar tasks, by re-training on our task-specific dataset. The intuition behind this is that the lower levels of the network correspond to low-level concepts such as edges and shapes and these features will not need to be re-learned if the new classifier is similar to the original one.

The most common task that Convolutional Neural Networks are used for in image processing is classification. For example, given an image, compute a probability score that the image contains a dog or a cat, or in other words, classify the likelihood of it containing a dog or cat. This is done by setting the final layer of the network model to output the probability of the image containing a range of objects (typically known as softmax). However in the work we are reporting here, our task is regression — to predict or estimate the level of air pollution given an image. If we are using a pre-trained model, we must ensure that the last layer of the model is a single output, the estimation of air pollution value.

One of the first questions we are faced with is how large and how complex and with how many layers should the network be? We measure this by evaluating a variety of different networks. Shallow networks have few layers, whereas deep networks have many layers. Increasing the number of layers has the effect of increasing the complexity of functions that the network can learn, at the expense of increasing the threat of the network overfitting the data. Smaller networks are therefore less prone to overfitting. This is also known as the “entropic capacity” of the model. Larger models, however, are also slower to train. To reduce overfitting we can also use data augmentation, which we will describe in more detail in the next section.

To measure the state of the art for classifying images, a large dataset of publicly available labelled images has been created called ImageNet (Russakovsky et al., 2015). New models can be evaluated by how accurate is their identification of a

variety of image classes based on images from ImageNet. The most popular models for classification are VGG16 (simple), Resnet (extremely deep) and Inception (wide and deep). We will evaluate the accuracy of some of these models on our wearable camera image dataset. However, as discussed above, we will have to adapt the architecture of these models a little in order to change their outputs from classification tasks to a regression task.

When training a Neural Network, it is possible to get better results by repeatedly training over the entire data set. Each one of these passes is called an epoch. Our training data set contains roughly 120,000 samples to train on, which represents about one third of the whole data set. Models with high capacity will tend to improve after several epochs. We plot the training history for both a test set and a validation set to demonstrate whether training for a longer number of epochs will improve results.

It can take up to one hour on a standard desktop computer to train a model for a single epoch in an image classification task. Because this takes so much time, we only train on a single test, train and validation set instead of performing an n-fold cross-validation and averaging over 10 of these sets. Once we have evaluated several models we run the best of them over the 10 datasets in an n-fold manner, to compare performance to the earlier classifiers, which did not use images.

To recap section Section 5.3.1 we evaluated some simple Machine Learning pipelines with out using the image data to form a baseline. These results can be seen in Table 5.3. The table is sorted by the performance in the test set. We can see that the linear model with a mean absolute error objective performed the best, with a score of **15.22** on the test set. We want to beat this using our deep learning models to show that they perform better. Also note that the dummy classifier, which only returns the average of the results scores **18.29** on the test set, so if the deep learning approach does not beat this, then it is not learning anything useful.

5.3.3 Data Augmentation






Data augmentation is the process of increasing the amount of data available for training a classifier in order to improve its performance. In image classification, it takes the form of adding random changes to images to avoid overfitting because often a model can overfit on its training data. For example, if many of the images with high pollution levels have a road on the left side of the image, it may

associate roads on the left side of the image with high levels of pollution. We can generalise this, and also double our training size by taking each image, and flipping it in various ways. That way, when we are training our model, we do not overfit the position of objects within an image. We can do this for a variety of transformations and in our application we use the following transformations illustrated in the Table below.

These transformations are applied randomly, so the regressor will not be trained on the same image twice. So, if we perform 10 epochs for example, we will augment each of the 120,000 images with 10 random variations. We also shuffle the data randomly to train in batches and ensure that for each epoch, each image will only be seen once, in one of its permutations. Table 5.4 shows two example images in the first row, and then each subsequent row shows the variety of image transformations we applied.

Table 5.4: Data Augmentation Examples

Transform	Example 1	Example 2
Original Image		
Width Shifting		
Height Shifting		

Transform	Example 1	Example 2
Zoom		
Horizontal Flipping		
Combined		

In Section 5.4.6 we examine the effectiveness of using data augmentation which is used in all of the following experiments.

5.4 Deep Learning Regression

5.4.1 Regression using a Shallow Network

In the first of our experiments using deep learning for predicting air pollution levels we experiment with a relatively straightforward three-layer neural network. This will serve as a baseline for our deep learning results. The motivation behind using such small network is to reduce potential overfitting. In Figure 5.15 we can see the architecture of the model used. The inputs are the raw pixels from a

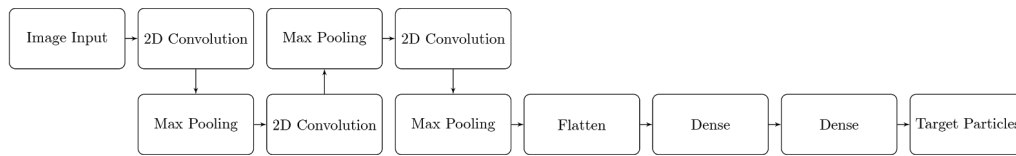


Figure 5.15: Architecture of Shallow Network

224×224 image over 3 channels.

In interpreting the results we note that the training error is the error on the training dataset. The validation error is the loss on the validation set, which was not used during training. A falling training error with a constant or rising validation error means that the regressor is overfitting while a falling training error and validation error means that the regressor is learning generalizable features. A stagnant training error indicates that the regressor is underfitting, and that a more complex architecture may be appropriate.

The result shown in Figure 5.16 show that the validation network reaches a plateau almost immediately. This indicates that the model capacity is too low, and that a more complex network needs to be constructed so that the model can learn more complex features.

The test set performance of this classifier is **40.47**. This result is quite poor compared to the dummy score of **18.29** using non-image features.

5.4.2 Regression Using Original VGG Model with All Layers Fixed

Now we will compare the performance of regression when simply using a state of the art model, almost out-of-the-box and unmodified. For this we use the VGG model (Simonyan and Zisserman, 2014), which contains 16 layers and is a seminal model in image classification. We can see the architecture of this model in Figure 5.17⁶. We will have to adapt the model by replacing the final layer with a single number as an estimation of the pollution level, instead of a softmax classifier which had been used when the task was image classification. This changes the output from being probabilities of one of 1,000 classes being present in the image, to a single number predicting what the level of air pollution will be.

The architecture is similar to the first model used above, however it is much deeper in that it has more layers. Because of this, it takes a long time to train on

⁶taken from http://file.scirp.org/Html/4-7800353_65406.htm

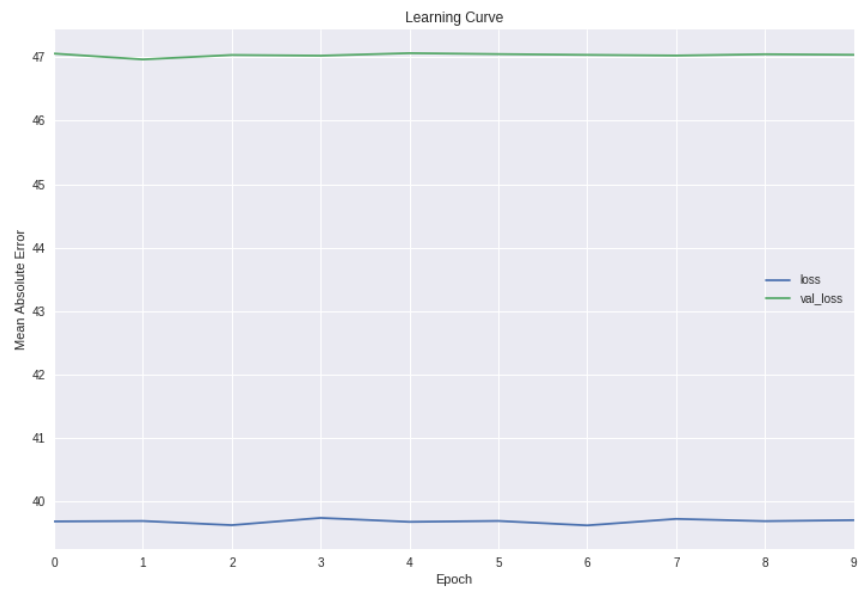


Figure 5.16: Shallow Neural Network Learning Curve

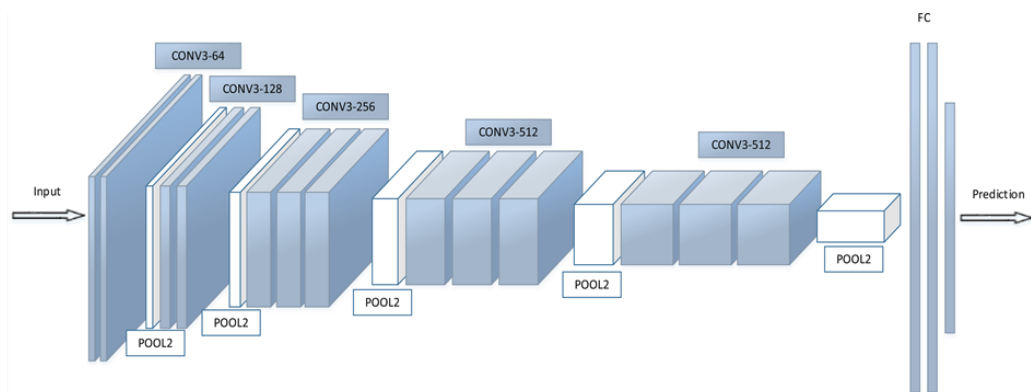


Figure 5.17: VGG architecture

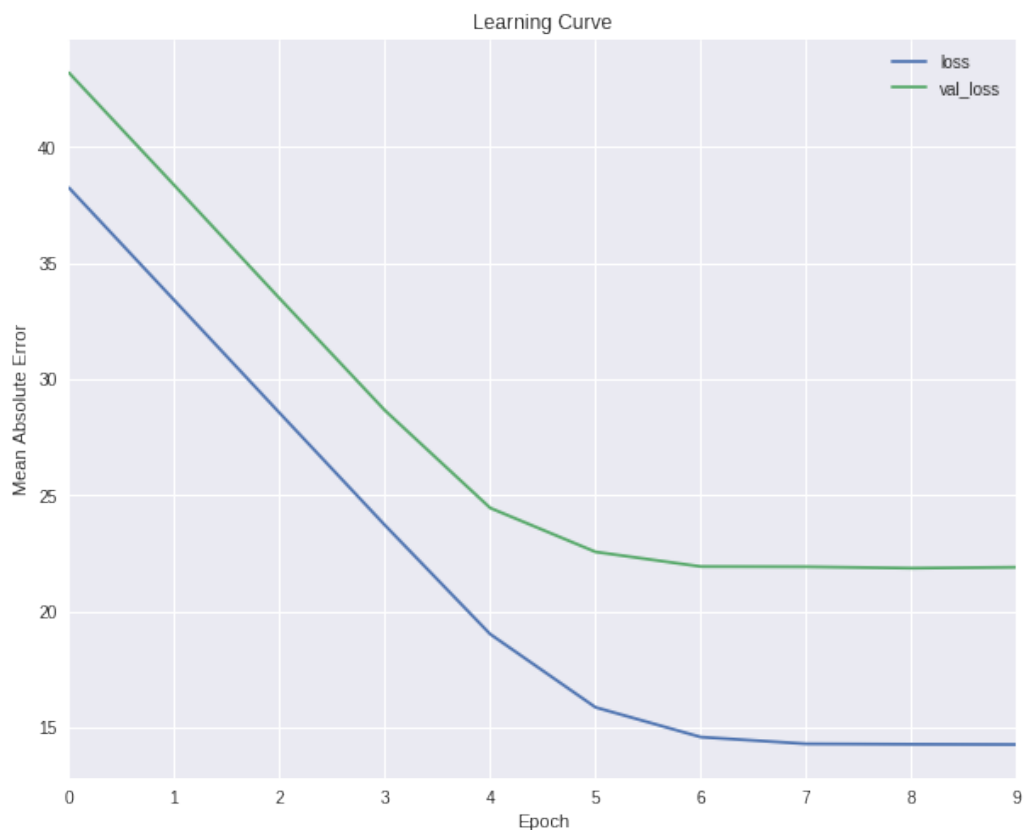


Figure 5.18: VGG with All Layers Fixed

our dataset. The weights for this model were obtained from an online repository of common image detection models⁷. This is effectively first detecting low level features and then concepts in the image, and then uses these concepts to estimate the air pollution level.

In Figure 5.18 we can see the Learning curve of this set up. We can see that that for the first 7 epochs it is learning useful features, but after that it plateaus. Note that this is for one of the 10 test train split iterations, and so the end result for this example is lower than the average of all 10 iterations.

In this section, we establish a baseline for the effectiveness of this model. We simply use the weights of the standard model, and only modify the final layer. As a result of doing this, we obtain test error Mean Absolute Error of **15.02**, which is lower than the shallow baseline of **15.22** established above. This shows that using images alone can be used to train a model to detect air pollution levels better than GPS.

⁷<https://keras.io/applications/>

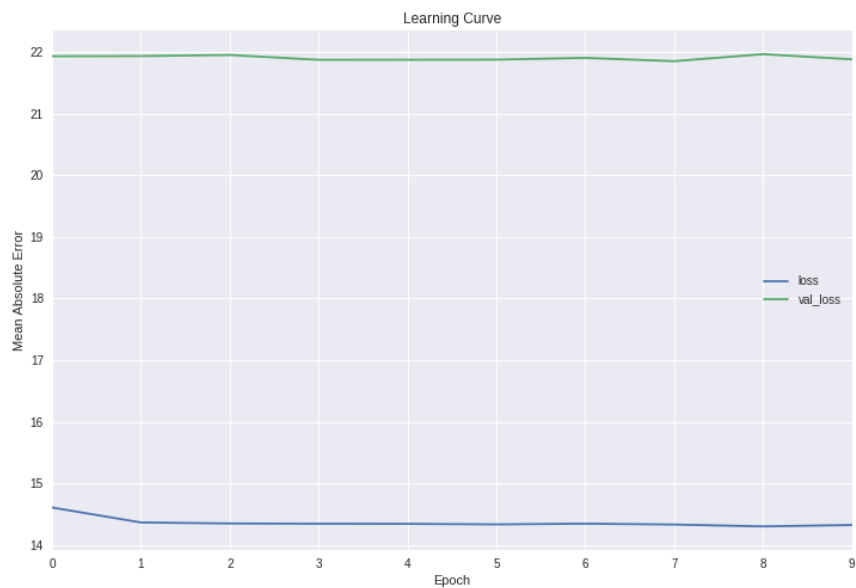


Figure 5.19: VGG Fixed Convolutions Learning Curve

In Section 5.4.3 and Section 5.4.4 we will examine modifications and different configurations of this architecture, where various amounts of the network are learned. We will see that increasing the entropic capacity of the model reduces performance.

5.4.3 Regression using a VGG model with the convolutional layers fixed

In this experiment we increase the number of trainable layers thus making the network more complex. The weights learned for convolutional layer remain fixed, but the 3 dense layers at the end are updated as shown in Figure 5.17. While more complex, this also increases the capacity of the model to learn.

The learning curve observed from this model can be seen in Figure 5.19. We can see that it has already reached the lowest test error by the second epoch, and that further epochs are unnecessary. This shows that the regressor is learning useful features initially, although it is possibly underfitting.

The results obtained with this model are a Mean Absolute Error of **15.19**. This is slightly better than our shallow baseline established above.

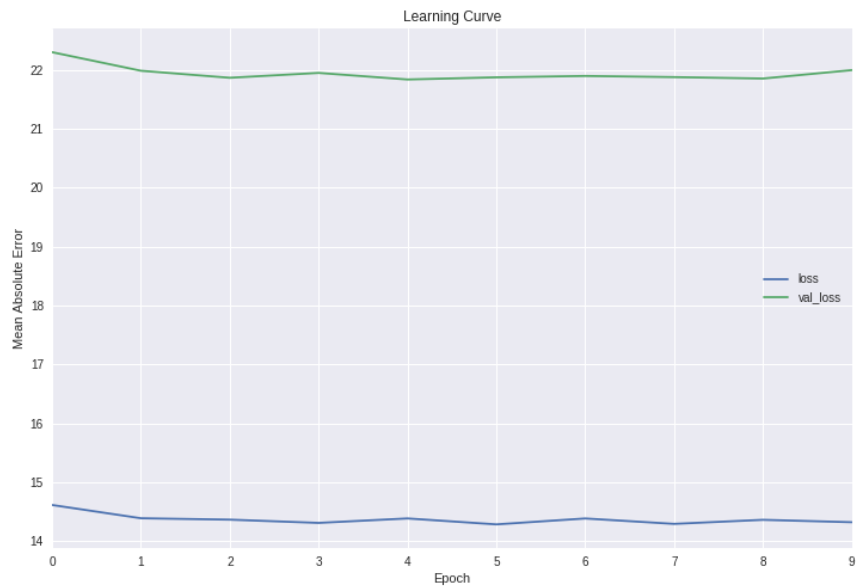


Figure 5.20: VGG Adaptable Model Learning Curve

5.4.4 Regression using an end-to-end trained VGG model

Once again we use the VGG pre-trained network from (Simonyan and Zisserman, 2014) to analyse the images because of its performance and availability. This time we will use a fully end-to-end model, where we update all the weights in the model, only retaining the architecture.

From Figure 5.20 we can see that the training error falls slightly for the first 3 epochs. Hence this shows that it has a higher capacity to learn than the first two configurations of the VGG network we have already seen.

The test performance of this classifier was a Mean Absolute Error of **15.08**, which is lower than the score of **15.22** achieved using just GPS, temperature and vil-lageID features. This is worse however than that achieved in Section 5.4.2.

5.4.5 Regression using ResNet

The previous experiments have shown that in general, deeper networks perform the best in terms of estimating air pollution levels from images. An alternative to the VGG network is ResNet, which is a model that is extremely deep, and repre-

sents the current state of the art in image recognition projects (He et al., 2016). The main difference between ResNet and VGG is that ResNet has many more layers, up to 152 of them, and it is believed that it operates as an ensemble of many relatively shallow networks. A diagram of the ResNet's architecture is seen in Figure 5.21⁸

We use the ResNet architecture which is adapted to output an air pollution level estimates instead of scores for individual classes, in a similar way to how we modified the VGG architecture output to estimate pollution levels. The test set performance of this classifier is **15.45**. This is worse than the VGG model, and shows that VGG outperforms ResNet on this task. We can see from Figure 5.22 that the model overfits, because after 10 iterations, even though training error continues to decrease, the validation error remains constant.

5.4.6 Effectiveness of Data Augmentation

To evaluate the performance of data augmentation, we ran experiments with and without data augmentation. As the most effective strategy for learning the pollution level from images was the fixed weights VGG method seen above, we compare the performance when data augmentation is not used. The score obtained from not using the augmentation was a Mean Absolute Error of **15.11**. This is only slightly worse than when we used data augmentation, at **15.02**. So data augmentation does help learn generalisable features, but in this case it does not help very much.

5.4.7 Combined Image and Location Model

We have established that predicting the level of air pollution is possible from both images and from GPS. We now consider a combination of GPS and image features together. To do this we use a multi-input neural network. We use a two "branches" with a single output to achieve this goal.

The first branch takes the image as an input. Again, we use a fully adaptable VGG model. This time, we simply remove the final layer, which would have been the softmax function to output the probabilities of the image containing one of 1,000 classes. The previous layer contains 4,096 outputs. We will see below how this

⁸(taken from http://book.paddlepaddle.org/03.image_classification/)

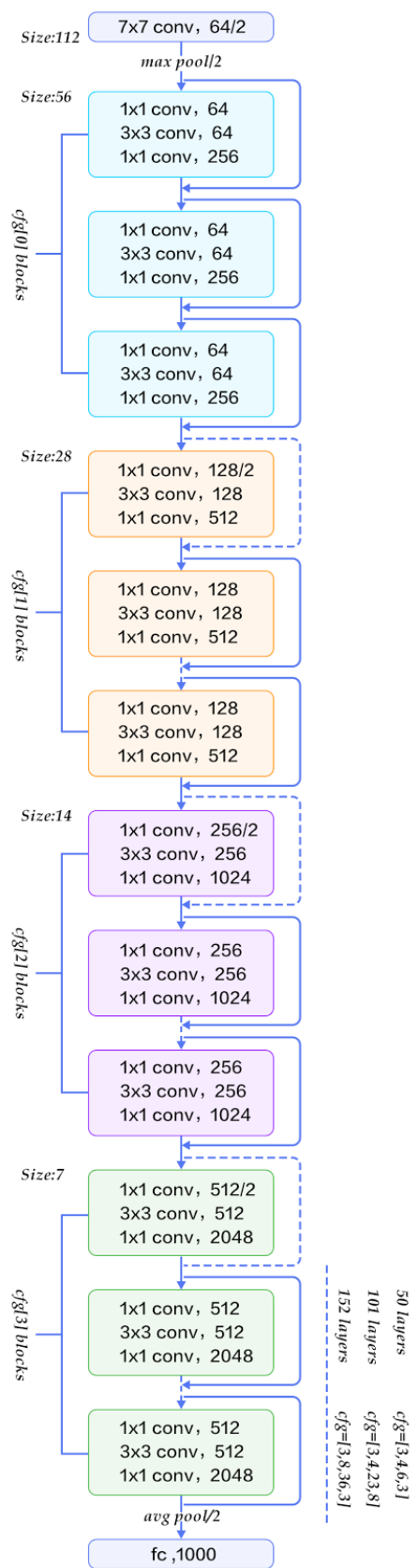


Figure 5.21: Resnet Architecture

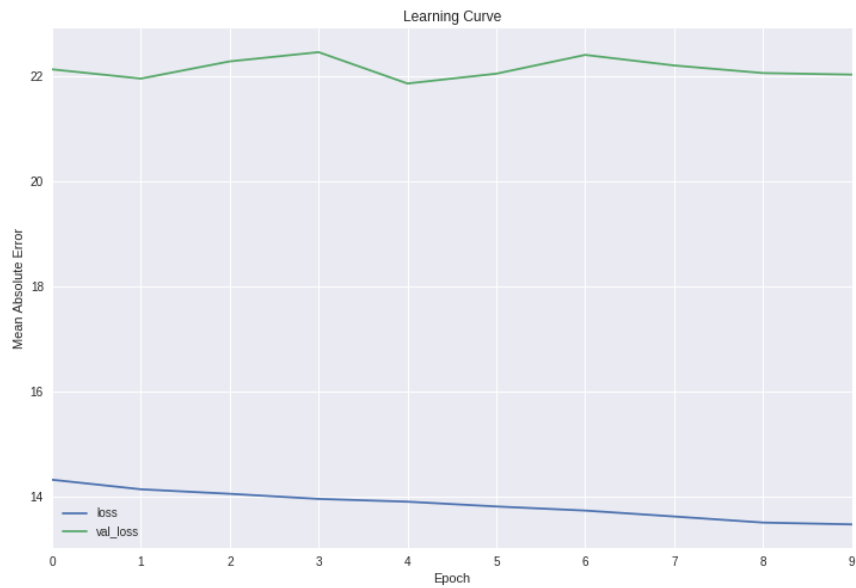


Figure 5.22: ResNet Model Learning Curve

attaches to the entire model.

The second branch takes the GPS, villageID and temperature as an input. These inputs are concatenated with the 4,096 features extracted from the image to form a new input vector. These serve as inputs to a densely connected layer with a single output for the pollution level. This model is illustrated in Figure 5.23

The resulting score is a Mean Absolute Error of **14.82**, which is lower than both the individual scores for either when we use images or GPS as inputs.

5.4.8 Evaluation

Table 5.5 shows a summary of the results of all of the deep learning methods used. We can see that of the image models, the VGG method with all layers fixed except the final one is the best performing method, and that it performs better than our baseline model using GPS and temperature features. We can also see that all of the deep models outperform the dummy baseline, with the exception of the shallow network. We can also see that the combined image and location model together have the best score overall.

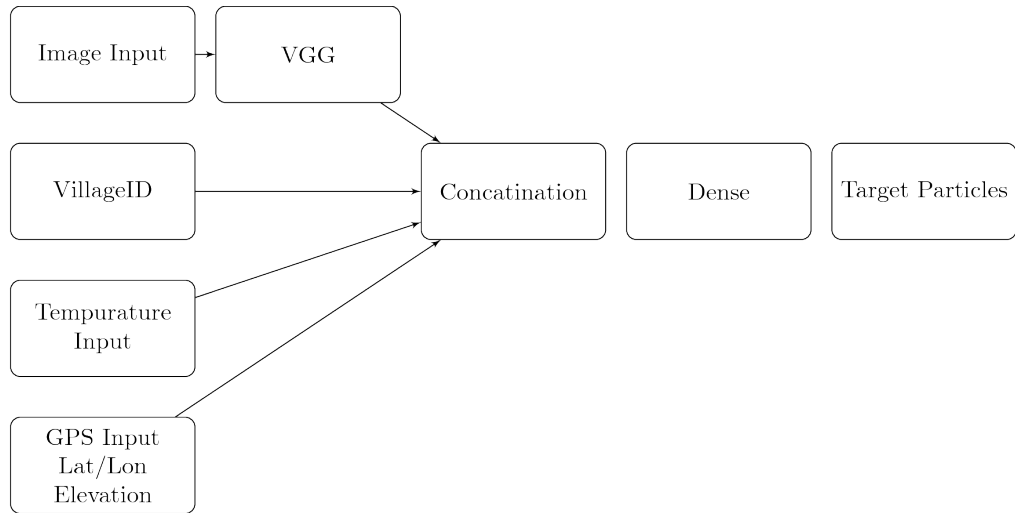


Figure 5.23: Combined Network Architecture

Table 5.5: Image Pollution Prediction Results

Architecture	Mean Absolute Error
VGG Model & GPS & Temperature	14.820587
VGG all layers fixed	15.026548
VGG all layers	15.082521
VGG last layers	15.190522
GPS & Temperature baseline	15.229376
Resnet	15.458944
Dummy baseline	18.299901
Shallow Network	40.474887

5.5 What is the Neural Network Learning?

We now ask the question what is the neural network learning. Neural Networks can be seen as black boxes in some sense, where it is hard to determine what the network is doing, other than by evaluating its performance on some data set. There is a risk in this part of our work, for example, that the model may be over-fitting by identifying objects which are local to this particular area which would not generalise well to the task. For example, these objects might simply be a random object that has no correlation with pollution levels in general, for example a post box which is easily identifiable and which many participants see. If this post box is situated in an area with a high level of pollution, the classifier may spuriously learn to associate the concept of 'post boxes' with pollution. Based on our analysis of the locations of the participants at different times, and we designed the cross validation set to reduce this possibility as much as possible. And so this scenario seems unlikely, and yet we cannot rule it out. In Section 6.4 on future work, we discuss some strategies that could be applied to figuring out what is actually being learned by the classifier.

5.6 Conclusion

In this chapter we have shown that it is possible to predict pollution levels using both static features such as GPS and temperature, and using image features from images taken from a wearable camera. We have investigated the data and pre-processed it so that it is suitable to include in a Machine Learning pipeline. We also evaluated the technique when using data augmentation to increase the number of images used in training.

We have also shown evaluations of both methods of predicting air pollution levels, and shown that deep learning based image processing is the best individual method of predicting air pollution levels. We also developed a combined architecture that takes both of these features into account to give the best results.

Potential applications of this research could be to warn people of high pollution exposure using smartphone images. This contributes to the thesis by showing another application of Machine Learning for real world use cases. In the next chapter we will compare the three applications together.

Chapter 6

Conclusions

6.1 Contributions to Case Study Domains

In Chapter 3, Chapter 4 and Chapter 5 we introduced the three case studies which form the core of this thesis. We will now state the contributions that we have made in each of the domains from which the case studies are drawn.

In Chapter 3 we developed a solution to a case study in the educational analytics problem domain. The aim of this case study was to predict the examination grades of students based on their VLE logs. We first demonstrated how to extract features from the log data, including some 8 features we found to be useful, listed in Section 4.3.3. Of these features, 3 were new features which we had not found in the literature, and found to be useful for modelling student grades. We demonstrated how we built an intervention system based on this classifier, and crucially how we used the outputs of the model to intervene to prevent students from dropping out of their module or from failing the end-of-module examination. We showed that this intervention had a positive impact on student dropout rates.

After this, we demonstrated a new method of estimating student success in passing examinations using a Recurrent Neural Network. As far as we were able find, this was the first example of applying a deep neural network to VLE log data. We demonstrated the advantages of this system by applying it to data which was a combination of all modules, which has the potential to solve the ‘cold-start problem’ associated with making predictions for new courses, or those which have changed a lot since previous deliveries, in both cases because there is insufficient

log data available. The results of this model were promising, despite only using a single feature (log counts), compared to the 4 features per week that the non-deep learning method used. This is also easier to implement than our previous method, as it does not involve saving and loading the weights of up to 12 models per course, one for each week.

In Chapter 4 we developed a case study in the agricultural domain. The aim of this case study was to predict whether a calf was suckling or not from its mother, to allow a farmer to intervene and wean the calf allowing the mother to produce milk for commercial purposes. We examined the difficulties of developing a classifier when there is a constraint that the inference must be done in real time. We also examined a completely different set of features, where the data is an incoming stream of accelerometer and gyroscope data.

We developed a non-deep learning solution which performed very well in this task, achieving an AUC ROC score of 0.9835 on a cross validated data set on a large number of examples. We then examined this performance against a number of real world constraints which we were under. For example, we examined reducing the number of sensors we had available to use, and we found that we could still achieve good performance if we removed one of the two accelerometers which we were using. In fact almost any combinations of sensors which we had available to us performed well enough, with the exception of using a gyroscope by itself. We also examined the performance of the classifier reducing the frequency at which the devices sampled data. We found that we could reduce sampling frequency from 128Hz to 16Hz without degrading performance. This had a useful effect of the battery life of the device, and the processing power required to do computations. Finally, we developed a method of reducing false positives by taking an average over a time window. We examined the effect of changing the size of this window, and hence were able to make a case for a trade off between false-positives and time to detection.

We did not examine a deep learning solution to this problem, as we had achieved such a high accuracy using simpler solutions. We also did not use it as it would be difficult to implement on the limited hardware that we could deploy on a consumer device. This is one contribution to the thesis, that currently deep learning is not applicable to real time, low cost inference problems. However we expect this to change in the near future, and this left as a problem for future work, which we will discuss more in Section 6.4.

In our final case study in Chapter 5 we developed a solution to a case study in the

environmental science problem domain. The aim of this case study was to predict the level of pollution, measured in PM2.5, based on an image from a wearable camera.

We first examined the data in depth to determine if there are insights we could use which would be of use when building the classifier. One of the most important observations that we made was that the skewed nature of the pollution data would make it ineffective to use the normal ‘Mean Squared Error’ metric for training or validation. We examined a number of solutions to this point in Section 5.3.1, and determine that the Mean Absolute Error is an appropriate metric to use. Another alternative which may have been useful would be to detect the log of the pollution data, depending on how important it was to detect outliers.

We examined a number of solutions, examining both GPS and demographic features by themselves, Image features by themselves, and a combination of image, GPS together. We established that the best solution involved a mixture of all features together. To do this we created a hybrid Convolutional Neural Network which concatenated the output of the second last layer of VGG-16 classifier with GPS and Demographic features. One interesting aspect of this case study was that non-neural approaches performed very poorly, in comparison to the other two case studies where non-neural methods performed well enough to at least serve as a baseline against which we could benchmark our performance. Here the signal is so weak that non-neural methods struggle to outperform even a dummy model.

6.2 Comparison of Case Studies

In Chapter 1 we identified a number of axes under which we could classify different problem domains. We will now examine each case study under these axes.

6.2.1 Target Variable

In the first case study we examined the possibility of both using a categorical target variable (pass or fail) and using a numerical target (the grade achieved). Using a numerical target would be more useful, as we could get a finer-grained prediction of how well each student would perform. However, using a non-neural

network approach we could only get reasonable performance on the easier categorical target variable. We were able to infer a ranking using the probability of pass or fail, however this was less direct than simply predicting their expected grade. When we used the neural network approach using RNNs, we were able to predict a numerical target with much greater performance.

In the second case study, we used only had a categorical target variable - whether it was suckling or not suckling. This was because this action was “discrete”, it is only possible to be suckling or not suckling. The most useful output for our application was a probability of suckling. This allowed us to make a judgement about whether to make an intervention or not, based on whether this probability has crossed some threshold.

In the third case study we only had a numerical target variable – at what the level of pollution was at a given time. This had added challenges, as the target variable in this case had a small number of very large outliers. If this were a categorical variable we could re-balance the data set, training on an equal number of each class. However, a simpler solution in this case was to use a different training error which would not penalise outliers by a large amount. Hence we used the Mean Absolute Error.

6.2.2 Importance of Precision vs. Recall

In our first case study, both precision and recall were important. A false positive would in this example be a case of telling a student that they will fail, when in fact they are likely to pass. This could cause a student to feel unnecessary stress. A true negative in this case study would be a case of telling a student they will pass, when they are in fact likely to fail. This could cause a student who could pass with some extra level of effort to fail because they relax when they get this notification. It is difficult to say which of these two cases are worse, however it shows that we should not prioritise recall at the expense of precision or vice-versa. Because of this, we implemented a system where if the overall performance of a classifier for an individual course (measured using ROC AUC) was not good enough, we would not make interventions based on this classifier.

In our second case study precision was arguably more important than recall. A false positive in this case would mean that a calf would receive an intervention for suckling, when they did not actually suckle. Psychology would tell us that this might mean that the calf would be confused about why it is receiving an inter-

vention, and respond to the intervention in undesirable ways (Skinner, 1992). In comparison, a true negative would mean that the calf performed the suckling action without being penalised for it. This might reduce the effectiveness of the intervention if the true negative rate is too low, however we would expect in the long term that the calf would eventually associate the suckling action with the intervention.

Precision and recall do not as easily apply to our third case study, since the target variable is strictly numeric. However we can imagine a scenario in which we change this problem into a classification problem, for example by building a classifier which detects whether the target variable exceeds some critical threshold. In this case which would be more detrimental: False Positives or True Negatives? A false positive would correspond to a false alert of a critically high pollution level, whereas a true negative would correspond to not alerting an actual high level of pollution. In this case, the costs of a false alarm are not very high, as a user will typically not take expensive evasive action to avoid one instant of pollution. Hence, we can imagine this case study as one in which precision is more important than recall. This aligns with our choice in this case study to use the mean absolute error metric, instead of mean square error.

6.2.3 Real Time vs. Analysis

In our first case study the interventions were made once per week. This gave us plenty of time to do analysis and double check the outputs of the classifiers. Considering that we were intervening with human beings, and all of the ethical considerations that brings (see Appendix 2), this was useful as it gave us time to check that everything was functioning as it should do. If the interventions were to be given in a matter of seconds for example, the

In our second case study, on the other hand, interventions had to be made within a number of seconds of detecting that a calf was suckling. This was therefore an example of a real time application. We dealt with this in Section 4.3.7 by developing a method of trading off time to detection with precision.

In our third case study, we were doing an analysis only, with no intervention required. Hence we could take even more time to do analysis than we could for our first case study. This was a clear example of an analysis problem. However, in future we could extend this to a real time application for detecting excessive levels of pollution on a smartphone for example.

6.2.4 Deployment

In our first case study our deployment solution was centralised. Our models were stored on a central server and used to send emails and generate the contents of a web application which students could read. This gave us a high degree of control over our solution, as we could inspect and modify it as we felt was necessary.

In our second case study our deployed solution was decentralised. Our models would have to be trained, and then loaded onto a device to be used in the wild. This would give us no opportunity to update the models later. This made it very important to gather as much data as possible prior to this and to make sure the data was as varied as possible. It also made the process very conservative, as we had to start small and scale up our solution, checking at every step of the way that the models were behaving as expected.

6.2.5 Case Study Summary

We can see from the above the classifications of problem domains that the three case studies we have chosen are extremely varied. This diverseness drove many decisions in the process of applying machine learning to them. We will now examine this in the context of our hypothesis.

6.3 Hypothesis Evaluation

In this section we will examine each of the resulting models in the context of our original research hypothesis. To restate our hypothesis, we have claimed that ***“That problem-solving in diverse problem domains using Machine Learning applied to time series data requires diverse models in order to achieve the best performance”***. In Section 1.8 we introduced our reasoning for using the case study research methodology.

In Chapter 2 we examined the relevant literature for each of these three case studies. We found that there was much previous research on the first topic on the application of Machine Learning to education than there was on the the other two topics. For each of these sections we summarised the metrics, scores and algorithms used. We then found that some algorithms have come into and out of common usage over the previous 20 years, and in all cases neural network deep

learning methods seem to be the most successful option currently.

We can see from Section 6.2 that the set of case studies gave us a diverse set of problems. These differences were mostly driven from the fact that they were selected from different problem domains. From their respective chapters we can see that their solutions were varied. The first case study was optimally solved using a recurrent neural network. The second case study was optimally solved using a random forest (with the caveat that large models would not have been appropriate given the hardware and real-time constraints). The third case study was optimally solved using a hybrid neural network. Each of these case studies has provided evidence that the hypothesis is true, as they have each been drawn from a different problem domain, and each was optimally solved by a different model. Therefore our conclusion is that the hypothesis is verified: that each of the three case studies required a different model to be solved effectively.

6.4 Future Work

In Section 3.9 we developed a method of using a deep neural network with VLE log data to predict the end of semester performance of a student. This compared favourably to our method without using deep learning methods, even though we were restricted to a single feature - the weekly VLE usage of that student. We would like to extend this work by including more of the features developed in Section 3.3 in our deep learning model.

In Chapter 4 we investigated the problem of determining whether a calf was suckling or not. In this chapter a deep learning approach was not taken because the results of the non-deep learning approaches performed very well, and the constrained hardware of the sensors would not have enough capacity for a neural network approach. In future work we would like to use an RNN to take into account the time series nature of the data, rather than the smooting approach which we have taken. We believe that this has the potential to improve the performance of the classifier even more. We would also like to put this neural network on a chip, given the recent advances on embedded devices designed for neural network inference (Ionica and Gregg, 2015).

In Section 5.5 we asked the question - “what is the neural network learning?”. We would like to explore this in detail in future work. In particular, we would like to determine what are the visual features being learned. We suspect that it is

learning concepts such as 'traffic' and 'greenery'. However, as discussed earlier, it could also be learning features which are specific to that data set, and potentially overfitting by learning specific objects which are in the same picture from two different people. If this is the case, the learned object may not have real connection to the level of pollution in the environment. One approach to this problem is to inspect what is actually learnt by the neural network. There is much prior work on visualising what Convolutional Neural Networks learn (Girshick et al., 2014; Zeiler and Fergus, 2014; Maaten and Hinton, 2008).

Another approach that we would like to take in future work is to combine the Convolutional Neural Network with a Recurrent Neural Network so that the model can take into account the context of the estimated pollution levels before the prediction was made. A precedent for combining a Recurrent Neural Network and a Convolutional Neural Network can be found in Pinheiro and Collobert (2014), where the authors use this model to do scene labelling.

Appendix 1: Publications

David Azcona, Owen Corrigan, Philip Scanlon, and Alan F Smeaton. Innovative learning analytics research at a data-driven hei. *3rd. International Conference on Higher Education Advances*, 2017.

Owen Corrigan and Alan F Smeaton. A course agnostic approach to predicting student success from vle log data using recurrent neural networks. In *European Conference on Technology Enhanced Learning*, pages 545–548. Springer, 2017.

Owen Corrigan, Mark Glynn, Aisling McKenna, Alan Smeaton, and Sinead Smyth. Student data: data is knowledge—putting the knowledge back in the students’ hands. In *ECEL2015-14th European Conference on e-Learning: ECEL2015*, page 165. Academic Conferences and publishing limited, 2015a.

Owen Corrigan, Mark Glynn, Alan F Smeaton, and Sinead Smyth. Mining access logs with predictive analytics to improve student performance. *EdTech 2015, Irish Learning Technology Association Conference*, 2015b.

Owen Corrigan, Alan F Smeaton, Mark Glynn, and Sinéad Smyth. Using educational analytics to improve test performance. In *Design for Teaching and Learning in a Networked World*, pages 42–55. Springer, 2015c.

Alan F Smeaton, Sinead Smyth, Owen Corrigan, John Brennan, and Ely Egan. Predicting “at risk” students from log data. *Insight Student Conference 2014*, 2014.

Appendix 2: Ethical Considerations for the PredictED Project

In this appendix we will describe the steps we took to get clearance from Dublin City University to use the moodle data, as well as ethical and privacy issues that we encountered in managing and using student data. In this project we used two main sources of student data: logs from the Moodle LMS, and students' exam results from previous years. Note that much of the work in the appendix was not solely performed by myself, but instead by a team of people involved in this project. Hence, I am including this as an appendix, as I am not claiming that this as entirely my own personal work.

6.5 Research Ethics Committee

Before we began collecting and analysing the student data, we first submitted an application to the Research Ethics Committee of the University. This is required for any research activity involving human participation and is a standard procedure for research of this nature and involved justifying our use of the data, and defining the scope of the study. Following formal approval, we approached the University's Information Services & Systems (ISS) department to get access to the Moodle log data. We also approached the university registry to collect the exam results data. Together, and along with the University's Data Protection Officer, we agreed on a set of protocols for joining the two data sets together, anonymising the data, and the scope of the research. We were also helped in our analysis by the student supports services department.

One important point to make was that students who were not involved in our project (i.e. the ones who had been in college before this program ran and whose exam results we would be using) were not included in the scope of the data agreement. They did allow their data to be analysed as part of accepting the rules and regulations of studying in DCU and this was deemed acceptable by the University's Data Protection Officer.

We first used all the data to determine which courses/modules would be appropriate to run the intervention on. After this, we were only provided with data for the 10 courses which we identified and where the performance of the classifier was good enough that it would be worth running an intervention program on. At the end of the semester the program was evaluating our predicted exam performance for current students against the actual results achieved by the students.

6.6 Anonymisation

While doing this analysis, at all times the student data was anonymised except where we needed to send email alerts to students. The staff at ISS developed a special code which would allow me to link the results of the students together with their Moodle ID number. However, this did not correlate with an ID number which we could use to identify individual students, such as their student number.

6.7 Student Awareness of Monitoring and Opting into Project

In addition to formal University approval via the Research Ethics Committee, we asked individual module co-ordinators and Deans of Faculty if they would agree to the project. We also met with the Registrar, Deputy Registrar, Head of Information and Systems Services, Head of Student Support Services and a full sitting of the University's Teaching Committee in order to appraise them of what we were doing, to allay any fears or misconceptions they may of had.

At the beginning of the course/module, in the first year of the project, we visited each class in their lectures during the first week of the first semester to inform them directly of what we were doing. For the second year the students were shown a video describing the project on their first connection to moodle

so when the student first logged into their moodle page, they were shown a dialogue containing a plain language statement describing the experiment. As part of this dialogue, they could choose to opt into or out of receiving email alerts. They were also provided with a link, which was included in each email sent, which allowed them to opt out of the emails. These are part of the standard protocol for involving human subjects in research like this.

6.8 Data Protection

As part of our data protection obligations, we anonymised all the data, so that none of the results could be traced back to an individual student. We kept the data on a secure computer, locked behind a card swipe access door, and as part of our agreement with the University we ensured that this data would only remain on this computer and not stored on a cloud service.

6.9 Plain Language Statement

Below is the text of the plain language statement shown to students before they are asked to opt into or out of the study.

Introduction to the Research Study The Insight Centre for Data Analytics along with the School of Nursing and Human Sciences in DCU are conducting a study on predicting student success in a module based on Moodle interaction. This study is part funded by the Learning Innovation Unit and is supported by Student Support Services. You will probably have heard of this as part of your first year orientation and your module XXXX has been chosen to take part.

What does the study involve? Moodle is a Virtual Learning Environment that lecturers and students use in order to communicate, upload and download material. Throughout this study patterns of Moodle use will be used to predict success in a module. These predictions are also based on patterns of engagement from previous years. Insight data analytics are working to create as accurate a prediction of success as possible. Each week, you will receive an email reminder to check a personalised online leaderboard which will dis-

play the scores of all participants ranked based on levels of engagement with Moodle. This leaderboard will highlight whether or not your Moodle engagement is in line with what our data predicts to lead to successful completion of the module.

Is this a requirement for this course? No. We are gathering information about how students complete everyday module exercises and you need to do these tasks as part of the module. However, we are currently asking you to consider taking part in this study and letting us use your data to inform teaching practices and also for research purposes, and perhaps publication in the future. This is above the requirements for the course and you are under no obligation to do so. If you do not wish to allow us to use your data for research purposes then please indicate this on the consent form that you will be asked to complete following reading this Plain Language Statement.

Are there any risks anticipated It is not anticipated that taking part in the study will involve any risks greater than those encountered in everyday life. All that is being measured is the completion of academic work and attendance and interaction in class.

Are there any benefits to me in taking part? An indirect benefit to you is that it is hoped that this study will increase our knowledge of how best to help students complete their modules successfully in the future.

What will happen to my data? The data on Moodle activity is collected by ISS and analysed by the Data Analytics Team in Insight. This analysis allows identification of at-risk students by Moodle engagement levels. This data will be put on a leaderboard on Moodle for your fellow classmates to see. Being a user of the Moodle system, you have agreed to allow ISS the rights to your Moodle usage data.

Do I have to take part? No. You are under no obligation to sign the consent form indicating that you want to take part in this research study. Simply return a consent form indicating you do not wish to participate and we will not include you in the project. You do not have to give a reason.

What happens if I don't take part? There will be no penalties if you decide not to take part. This will not affect your on-going university life.

Confidentiality Participant confidentiality will be protected to the extent permitted by laws & regulations. While it is not anticipated that any concerns could arise from participation in the study the procedures set out by the university will be followed under any such instance. All the information collected in this study will be treated in the strictest of confidence. Participant data will be fully anonymised in any scientific publication. We ensure proper safeguards so that participation is confidential and data are securely stored and protected. This study will be run with the approval of DCU Research Ethics Committee (XXX insert Decision Number here XXX). In accordance with standard research data management practices data belong to you and your child will be securely retained for 5 years after the study is completed.

What do I do if I have any questions about this? Please contact one of the Investigators, Prof Alan Smeaton or Dr Sinead Smyth. Their contact details are below.

Contact details. If you have any queries please contact: *Details omitted.*

If participants have concerns about this study and wish to contact an independent person, please contact: The Secretary, Dublin City University Research Ethics Committee, c/o Office of the Vice-President for Research, Dublin City University, Dublin 9. Tel 01-7008000

Bibliography

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
- Lilly Suriani Affendey, Ikmal Hisyam Mohd Paris, Norwati Mustapha, Md. Nasir Sulaiman, and Zaiton Muda. Ranking of influencing factors in predicting students' academic performance, June 2010. URL <http://psasir.upm.edu.my/16423/>.
- Lalitha Agnihotri and Alexander Ott. Building a student at-risk model : An end-to-end perspective. In *Proceedings of the 7th International Conference on Educational Data Mining (EDM)*, pages 209–212, 2014.
- Kimberly E. Arnold and Matthew D. Pistilli. Course signals at purdue: Using learning analytics to increase student success. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge - LAK '12*, page 267, New York, New York, USA, April 2012. ACM Press. ISBN 9781450311113. doi: 10.1145/2330601.2330666. URL <http://dl.acm.org/citation.cfm?id=2330601.2330666>.
- J. D. Arthington and R. S. Kalmbacher. Effect of early weaning on the performance of three-year-old, first-calf beef heifers and calves reared in the subtropics. *Journal of Animal Science*, 2003. ISSN 00218812.
- Bk Baradwaj and Saurabh Pal. Mining educational data to analyze student's performance. *Internation Journal of Advanced Computer Science and Applications*, 2(6):63–69, 2012. URL <http://arxiv.org/abs/1201.3417>.
- Michael Barnes, Tom Duckett, Grzegorz Cielniak, Graeme Stroud, and Glyn Harper. Visual detection of blemishes in potatoes using minimalist boosted classifiers. *Journal of Food Engineering*, 98(3):339–346, 2010.

- Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.
- Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.
- Ellen Block. The comprehension strategies of second language readers. *Tesol quarterly*, 20(3):463–494, 1986.
- Angela Bovo, Stephane Sanchez, Olivier Heiguy, and Yves Duthen. Analysis of students clustering results based on Moodle log data. In *Proceedings of the 6th International Conference on Educational Data Mining*, pages 306–307, 2013. URL http://www.educationaldatamining.org/EDM2013/papers/rn_paper_54.pdf.
- E. Oran Brigham. *The Fast Fourier Transform and Its Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988. ISBN 0-13-307505-2.
- M. Delgado Calvo-Flores, E. Gibaja Galindo, M. C. Pegalajar Jiménez, and O. Pérez. Predicting students’ marks from moodle logs using neural network models. In *Proceedings of the IV international conference on multimedia and information and communication technologies in education (M-ICTEE2006)*, volume 1, pages 586–590, 2006. URL http://www.researchgate.net/publication/252761512_Predicting_students_marks_from_Moodle_logs_using_neural_network_models.
- Wagner Cambuzzi, Sandro José Rigo, and Jorge L V Barbosa. Dropout prediction and reduction in distance education courses with the learning analytics multi-trail approach. *Journal of Universal Computer Science*, 21(1):23–47, 2015.
- Donald T Campbell. Iii.“degrees of freedom” and the case study. *Comparative political studies*, 8(2):178–193, 1975.
- Kevin Casey and J Paul Gibson. (m)oodles of data: Mining moodle to understand student behaviour. In *Proceedings of the 10th International Conference on Engaging Pedagogy (ICEP10)*, pages 61–71, Maynooth, Ireland, December 2010. Griffith College Dublin. ISBN 978-1-906878-02-3.
- Sheng Chen, Colin FN Cowan, and Peter M Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on neural networks*, 2(2):302–309, 1991.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase represen-

- tations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- François Chollet et al. Keras. <https://github.com/fchollet/keras>, 2015.
- François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.
- Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.
- William W Cohen. Fast effective rule induction. In *Proceedings of the twelfth international conference on machine learning*, pages 115–123, 1995.
- Owen Corrigan and Alan F Smeaton. A course agnostic approach to predicting student success from VLE log data using recurrent neural networks. In *European Conference on Technology Enhanced Learning*, pages 545–548. Springer, 2017.
- Owen Corrigan, Alan F Smeaton, Mark Glynn, and Sinéad Smyth. Using educational analytics to improve test performance. In *Design for Teaching and Learning in a Networked World*, pages 42–55. Springer, 2015.
- Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.
- Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016. URL <http://arxiv.org/abs/1602.02830>.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- Norman K Denzin. *The research act: A theoretical introduction to sociological methods*. Transaction publishers, 1973.
- Savvas Dimitriadis and Christos Goumopoulos. Applying machine learning to extract new knowledge in precision agriculture applications. In *Informatics, 2008. PCI'08. Panhellenic Conference on*, pages 100–104. IEEE, 2008.
- Bernie Dodge, John Whitmer, and James P Frazee. Improving undergraduate student achievement in large blended courses through data-driven interventions. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, pages 412–413. ACM, 2015.

- Bing Dong, Burton Andrews, Khee Poh Lam, Michael Höynck, Rui Zhang, Yun-Shang Chiou, and Diego Benitez. An information technology enabled sustainability test-bed (itest) for occupancy detection through an environmental sensing network. *Energy and Buildings*, 42(7):1038–1046, 2010.
- Bruno S Faíçal, Fausto G Costa, Gustavo Pessin, Jó Ueyama, Heitor Freitas, Alexandre Colombo, Pedro H Fini, Leandro Villas, Fernando S Osório, Patrícia A Vargas, et al. The use of unmanned aerial vehicles and wireless sensor networks for spraying pesticides. *Journal of Systems Architecture*, 60(4):393–404, 2014.
- Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156, 1996.
- Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.
- Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.
- Bill Gillham. *Case study research methods*. Bloomsbury Publishing, 2000.
- Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- Barney Glaser. *Discovery of grounded theory: Strategies for qualitative research*. Routledge, 2017.
- Anat Goldstein, Lior Fink, Amit Meitin, Shiran Bohadana, Oscar Lutenberg, and Gilad Ravid. Applying machine learning on sensor data for irrigation recommendations: revealing the agronomist’s tacit knowledge. *Precision Agriculture*, pages 1–24, 2017.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- E Grassel and B Schirmer. The use of volunteers to support family careers of dementia patients: results of a prospective longitudinal study investigating expectations towards and experience with training and professional support. *Zeitschrift Fur Gerontologie Und Geriatrie*, 39(3):217–226, 2006.

- Sevinc Gulsecen and Arif Kubat. Teaching ict to teacher candidates using pbl: A qualitative and quantitative evaluation. *Educational Technology & Society*, 9(2): 96–106, 2006.
- Salvador Gutiérrez, Javier Tardaguila, Juan Fernández-Novales, and Maria P Diago. Data mining and nir spectroscopy in viticulture: Applications for plant phenotyping under field conditions. *Sensors*, 16(2):236, 2016.
- Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.
- Shigehiko Hayashi, Kenta Shigematsu, Satoshi Yamamoto, Ken Kobayashi, Yasushi Kohno, Junzo Kamata, and Mitsutaka Kurita. Evaluation of a strawberry-harvesting robot in a field test. *Biosystems engineering*, 105(2):160–171, 2010.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.
- Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
- Geoffrey Holmes, Andrew Donkin, and Ian H Witten. Weka: A machine learning workbench. In *Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, pages 357–361. IEEE, 1994.
- Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. Spam base dataset. *Hewlett-Packard Labs*, 1999.
- Carol Hosenfeld. Case studies of ninth grade readers. *Reading in a foreign language*, pages 231–249, 1984.
- Tiansi Hu and Yunsi Fei. Qelar: a machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks. *IEEE Transactions on Mobile Computing*, 9(6):796–809, 2010.

- Xueqiao Huang and John R Jensen. A machine-learning approach to automated knowledge-base building for remote sensing image analysis with gis data. *Photogrammetric engineering and remote sensing*, 63(10):1185–1193, 1997.
- Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.
- DH Hubel and TN Wiesel. Shape and arrangement of columns in cat’s striate cortex. *The Journal of physiology*, 165(3):559–568, 1963.
- Alfredo Huete, Kamel Didan, Tomoaki Miura, E Patricia Rodriguez, Xiang Gao, and Laerte G Ferreira. Overview of the radiometric and biophysical performance of the modis vegetation indices. *Remote sensing of environment*, 83(1):195–213, 2002.
- Mircea Horea Ionica and David Gregg. The movidius myriad architecture’s potential for scientific computing. *IEEE Micro*, 35(1):6–14, 2015.
- Sandeep M Jayaprakash, Erik W Moody, Eitel JM Lauría, James R Regan, and Joshua D Baron. Early alert of academically at-risk students: An open source analytics initiative. *Journal of Learning Analytics*, 1(1):6–47, 2014.
- R Johansson. Case study methodology. a key note speech at the international conference “methodologies in housing research” organised by the royal institute of technology in cooperation with the international association of people–environment studies, stockholm, 22-24 september, 2003.
- Michael P Johnson. Decision models for the location of community corrections centers. *Environment and Planning B: Planning and Design*, 33(3):393–412, 2006.
- M Kanevski, Roman Parkin, Aleksey Pozdnukhov, Vadim Timonin, Michel Maignan, V Demyanov, and Stéphane Canu. Environmental data mining and modeling based on machine learning algorithms and geostatistics. *Environmental Modelling & Software*, 19(9):845–855, 2004.
- Rachel Heiden Keeler. *A machine learning model of Manhattan air pollution at high spatial resolution*. PhD thesis, Massachusetts Institute of Technology, 2014.
- John D Kelleher, Brian Mac Namee, and Aoife D’Arcy. *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT Press, 2015.

- Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *arXiv preprint arXiv:1706.02515*, 2017.
- Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86, 2005.
- Miroslav Kubat, Robert C Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2-3):195–215, 1998.
- Siegfried Lamnek. Qualitative sozialforschung. lehrbuch. 4., vollständig überarbeitete auflage. *Beltz Verlag, Weinheim, Basel. Lyon-Callo, Vincent (2000), in: Medical Anthropology Quarterly*, 14(3):328–345, 2005.
- Yann LeCun, LD Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, UA Muller, Eduard Sackinger, Patrice Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261:276, 1995.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, pages 143–155, 1989.
- Yuntao Li, Chengzhen Fu, and Yan Zhang. When and who at risk? call back at these critical points. In *Educational Data Mining 2013*, 2017.
- Konstantinos Liakos, Serafeim Moustakidis, Georgia Tsiotra, Thomas Bartzanas, Dionysis Bochtis, and Constantinos Parisses. Machine learning based computational analysis method for cattle lameness prediction. In *HAICTA*, pages 129–140, 2017.
- Min-Sheng Liao, Cheng-Long Chuang, Tzu-Shiang Lin, Chia-Pang Chen, Xiang-Yao Zheng, Po-Tang Chen, Kuo-Chi Liao, and Joe-Air Jiang. Development of an autonomous early warning system for *bactrocera dorsalis* (hendel) outbreaks in remote fruit orchards. *Computers and electronics in agriculture*, 88:1–12, 2012.
- M. Lichman. UCI machine learning repository, 2013. URL <http://archive.ics.uci.edu/ml>.

- Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AWM van der Laak, Bram van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *arXiv preprint arXiv:1702.05747*, 2017.
- Xiaoyang Liu, Zheng Song, Edith Ngai, Jian Ma, and Wendong Wang. Pm2: 5 monitoring using images from smartphones in participatory sensing. In *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, pages 630–635. IEEE, 2015.
- George I Lovell. Justice excused: The deployment of law in everyday political encounters. *Law & Society Review*, 40(2):283–324, 2006.
- Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
- Leah P Macfadyen and Shane Dawson. Mining lms data to develop an “early warning system” for educators: A proof of concept. *Computers & education*, 54(2): 588–599, 2010.
- Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014.
- Paula Martiskainen, Mikko Järvinen, Jukka-Pekka Skön, Jarkko Tiirikainen, Mikko Kolehmainen, and Jaakko Mononen. Cow behaviour pattern recognition using a three-dimensional accelerometer and support vector machines. *Applied animal behaviour science*, 119(1):32–38, 2009.
- Rob McCarney, James Warner, Steve Iliffe, Robbert van Haselen, Mark Griffin, and Peter Fisher. The Hawthorne Effect: a randomised, controlled trial. *BMC Medical Research Methodology*, 7(1):30, 2007. ISSN 14712288. doi: 10 . 1186 / 1471-2288-7-30. URL <http://www.biomedcentral.com/1471-2288/7/30>.
- Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

- Robert J McQueen, Stephen R Garner, Craig G Nevill-Manning, and Ian H Witten. Applying machine learning to agricultural data. *Computers and electronics in agriculture*, 12(4):275–293, 1995.
- Sharan B Merriam. *Case study research in education: A qualitative approach*. Jossey-Bass, 1988.
- Ryszard S Michalski and James B Larson. Selection of most representative training examples and incremental generation of vl1 hypotheses: The underlying methodology and the description of programs esel and aq11. Technical report, University of Illinois at Urbana–Champaign, 1978.
- Matthew B Miles, A Michael Huberman, and Johnny Saldana. *Qualitative data analysis*. Sage, 2013.
- MM Molina, JM Luna, C Romero, and S Ventura. Meta-learning approach for automatic parameter tuning: A case study with educational datasets. *International Educational Data Mining Society*, 2012.
- Nitin Muttill and Kwok-Wing Chau. Machine-learning paradigms for selecting ecologically significant input variables. *Engineering Applications of Artificial Intelligence*, 20(6):735–744, 2007.
- John J Nay, Emily Burchfield, and Jonathan Gilligan. A machine learning approach to forecasting remotely sensed vegetation health. *arXiv preprint arXiv:1602.06335*, 2016.
- Caroline Ochieng, Sotiris Vardoulakis, and Cathryn Tonne. Household air pollution following replacement of traditional open fire with an improved rocket type cookstove. *Science of The Total Environment*, 580:440 – 447, 2017. ISSN 0048-9697. doi: <http://dx.doi.org/10.1016/j.scitotenv.2016.10.233>. URL <http://www.sciencedirect.com/science/article/pii/S0048969716322628>.
- Fumiya Okubo, Takayoshi Yamashita, Atsushi Shimada, and Hiroaki Ogata. A neural network approach for students’ performance prediction. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, pages 598–599. ACM, 2017.
- Margaret A Oliver and Richard Webster. Kriging: a method of interpolation for geographical information systems. *International Journal of Geographical Information System*, 4(3):313–332, 1990.

- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.
- Michael Quinn Patton. *Qualitative evaluation and research methods*. SAGE Publications, inc, 1990.
- Pedro Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In *International Conference on Machine Learning*, pages 82–90, 2014.
- John Platt et al. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods: support vector learning*, 3, 1999.
- J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
- Ole Raaschou-Nielsen, Zorana J Andersen, Rob Beelen, Evangelia Samoli, Massimo Stafoggia, Gudrun Weinmayr, Barbara Hoffmann, Paul Fischer, Mark J Nieuwenhuijsen, Bert Brunekreef, et al. Air pollution and lung cancer incidence in 17 european cohorts: prospective analyses from the european study of cohorts for air pollution effects (escape). *The lancet oncology*, 14(9):813–822, 2013.
- John Rogan, Janet Franklin, Doug Stow, Jennifer Miller, Curtis Woodcock, and Dar Roberts. Mapping land-cover modifications over large areas: A comparison of machine learning algorithms. *Remote Sensing of Environment*, 112(5):2272–2283, 2008.
- Cristóbal Romero, Sebastián Ventura, Pedro G Espejo, and César Hervás. Data mining algorithms to classify students. In *Educational Data Mining 2008*, 2008a.
- Cristóbal Romero, Sebastián Ventura, and Enrique García. Data mining in course management systems: Moodle case study and tutorial. *Computers & Education*, 51(1):368–384, August 2008b. ISSN 03601315. doi: 10 . 1016 / j . compedu . 2007 . 05 . 016. URL <http://www.sciencedirect.com/science/article/pii/S0360131507000590>.
- Cristóbal Romero, Juan Luis Olmo, and Sebastián Ventura. A meta-learning approach for recommending a subset of white-box classification algorithms for moodle datasets. In *Educational Data Mining 2013*, 2013.

José A. Ruipérez-Valiente, Pedro J. Muñoz-Merino, Carlos Delgado Kloos, Katja Niemann, and Maren Scheffel. Do Optional Activities Matter in Virtual Learning Environments? In *Proceedings of the 9th European Conference on Technology Enhanced Learning*, volume 8719 of *Lecture Notes in Computer Science*, pages 331–344. Springer International Publishing, 2014. ISBN 978-3-319-11199-5. doi: 10 . 1007 / 978-3-319-11200-8 _ 25. URL [http : / / dx . doi . org / 10 . 1007 / 978-3-319-11200-8_25](http://dx.doi.org/10.1007/978-3-319-11200-8_25).

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Toby Segaran. *Programming collective intelligence: building smart web 2.0 applications*. " O'Reilly Media, Inc.", 2007.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Larry D. Singell and Glen R. Waddell. Modeling retention at a large public university: Can at-risk students be identified early enough to treat? *Research in Higher Education*, 51(6):546–572, February 2010. ISSN 0361-0365. doi: 10 . 1007 / s11162-010-9170-7. URL [http : / / link . springer . com / 10 . 1007/s11162-010-9170-7](http://link.springer.com/10.1007/s11162-010-9170-7).

CSJ Sister Mary Edward Healy. Le play's contribution to sociology: His method. *The American Catholic Sociological Review*, pages 97–110, 1947.

Burrhus Frederic Skinner. " superstition" in the pigeon. *Journal of Experimental Psychology: General*, 121(3):273, 1992.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

Robert E Stake. *The art of case study research*. Sage, 1995.

Jeanette A Stingone, Om P Pandey, Luz Claudio, and Gaurav Pandey. Using machine learning to identify air pollution exposure profiles associated with early cognitive skills among us children. *Environmental Pollution*, 230:730–740, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Suzanne Taylor and Virginia Berridge. Medicinal plants and malaria: an historical case study of research at the london school of hygiene and tropical medicine in the twentieth century. *Transactions Of The Royal Society Of Tropical Medicine And Hygiene*, 100(8):707–714, 2006.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Cathryn Tonne, Maëlle Salmon, Margaux Sanchez, V Sreekanth, Santhi Bhogadi, Sankar Sambandam, Kalpana Balakrishnan, Sanjay Kinra, and Julian D Marshall. Integrated assessment of exposure to pm2. 5 in south india and its relation with cardiovascular risk: Design of the chai observational cohort study. *International journal of hygiene and environmental health*, 220(6):1081–1088, 2017.

Elias Vansteenkiste. Data science bowl 2017, predicting lung cancer: Solution write-up, team deep breath. [http : / / blog . kaggle . com / 2017 / 05 / 16 / data-science-bowl-2017-predicting-lung-cancer-solution-write-up-team-deep-breath/](http://blog.kaggle.com/2017/05/16/data-science-bowl-2017-predicting-lung-cancer-solution-write-up-team-deep-breath/), 2017.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics, 1996.

Edward O Wiley, Kristina M McNyset, A Townsend Peterson, C Richard Robins, and Aimee M Stewart. Niche modeling perspective on geographic range predictions in the marine environment using a machine-learning algorithm. *Oceanography*, 2003.

ML Williams, N Mac Parthaláin, P Brewer, WPJ James, and MT Rose. A novel behavioral model of the pasture-based dairy cow from gps data using data mining and machine learning techniques. *Journal of dairy science*, 99(3):2063–2075, 2016.

Tellis Winston. Introduction to case study. *The Qualitative Report*, 1997.

- Annika Wolff, Zdenek Zdrahal, Andriy Nikolov, and Michal Pantucek. Improving retention: Predicting at-risk students by analysing clicking behaviour in a virtual learning environment. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge - LAK '13*, page 145, New York, New York, USA, April 2013. ACM Press. ISBN 9781450317856. doi: 10.1145/2460296.2460324. URL <http://dl.acm.org/citation.cfm?id=2460296.2460324>.
- Annika Wolff, Zdenek Zdrahal, Drahomira Herrmannova, Jakub Kuzilek, and Martin Hlosta. Developing predictive models for early detection of at-risk students on distance learning modules. *Proceedings of the Fourth International Conference on Learning Analytics and Knowledge - LAK '14*, 2014.
- David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.
- Robert K Yin. *Case study research: Design and methods*. Sage publications, 2013.
- Jiaxuan You, Xiaocheng Li, Melvin Low, David Lobell, and Stefano Ermon. Deep gaussian process for crop yield prediction based on remote sensing data. In *AAAI*, pages 4559–4566, 2017.
- Zaidah Zainal. Case study as a research method. *Jurnal Kemanusiaan*, 5(1), 2017.
- Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
- Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.
- Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.