VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

ENGR489 Preliminary Report

Megan Liang

Supervisor: Will Browne

Submitted in partial fulfilment of the requirements for
Software Engineering with Honours.

Abstract

The project will explore Artificial Intelligence applied to data mining with the use of LCSs to extract information from the huge amounts of data being collected every day. Drawbacks of LCS include being easily overwhelmed with too much data and being unable to extract information from low-level features. Therefore, feature selection and feature construction techniques will be used in conjunction with the LCS. The first half of the project evaluated that features extracted from a CNN can be reused as abstract features in a LCS. The second half of the project will validate whether LCS can apply learnt knowledge onto similar but unseen domains.

Contents

1 Introduction

Living in the petabyte era, we are drowning in information but starved for knowledge [20]. Given the explosive growth of rich data from information sources, challenging difficulties of information extraction have surfaced [1]. Data mining is the process of discovering interesting and useful patterns and relationships in large volumes of data [8]. The goal of data mining is to extract information from a data set and transform it into something functional to solve a defined problem. While data volumes proliferate, the knowledge it creates has not kept pace [1]. Conventional data analysis methods cannot be applied effectively to data mining as it involves tremendous amounts of data with high dimensionality and complexity. Artificial Intelligence methods such as Neural Networks (NN) applied to data mining has become an increasingly important field as it has demonstrated its effectiveness over traditional data analysis techniques.

1.1 Motivation

Since the Learning Classifier System (LCS) was developed by John Holland in the 1970s [12], their capabilities for rule induction in data mining have ignited renewed interest in this area. There have been a number of investigations on the architecture, where XCS, a popular implementation of LCS, have been the first to report both accurate and maximally generalised classifications [4]. The project will further explore Artificial Intelligence applied to data mining with the use of LCSs to extract information from the huge amounts of data being collected every day.

LCSs have been chosen as the Machine Learning (ML) technique to apply to data mining as they output human-readable rules which can be interpreted. Machine Learning techniques such as Deep Neural Networks (DNNs) have proven to be state-of-the-art in performance on problems that significantly outperform other solutions in multiple domains [17]. However, NNs contain human cognitive bias and are heuristic. This may lead to errors made in the processes within the system. With deep learning, there are many configurations which are prone to human bias such as the filters applied to each convolution and the assignment of weights on the parameters and nodes [13].

DNNs are difficult to interpret where learned high-level features cannot be extracted and transferred into different domains. Discriminative DNN models have proven to be easily fooled where they classify many unrecognisable images with near certainty as belonging to a recognised class [21]. This demonstrates that there are differences between the way DNNs and humans recognise objects and raises questions about the generalisation capacities of DNNs. If we regard two handwritten digits in Figure 1 as vectors of pixel values and compare these values, they are very different. This is the process and the level at which a Convolutional Neural Network (CNN) would operate (Refer to Section 1.5.1). However, regarded as shapes, the two images appear similar to a human observer [3]. As an input of a LCS are abstract features rather than pixels, it is less sensitive to image variations, where the angle, lighting and pose of objects in the images make training a CNN difficult.

Therefore, LCSs can analyse the components of an object, see the "bigger picture" and deduce knowledge which we can transfer to similar domains. For example, a LCS will observe that four circular wheels are a component of a car. We can then use the learnt knowledge of the representation of a car and apply it to another object such as trucks and buses. LCSs have the ability to construct features which will simplify the search space and therefore improve performance which is beneficial as redundancy, irrelevancy and complexity are all problems in data mining currently. There are drawbacks in LCSs where the solution space can be overwhelmed by too much data. The LCS solution is unhelpful with low-level

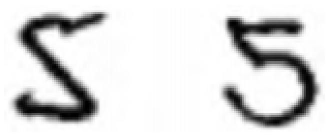


Figure 1: Examples of two handwritten digit extracted from [9]. In terms of pixel-to-pixel comparisons, these two images are different. As a human, the shapes appear similar where humans can recognise the images as fives.

inputs. Deep learning networks such as CNNs have an advantage of being able to handle large input data which contain low-level data such as pixels. Therefore, feature selection and feature construction methods will be utilised to reduce the number of features, rid of redundant and irrelevant features and find epistatic relationships between features [28](See Section 1.5.6).

1.2 Aim

The purpose of the project is to apply the concept of LCSs to Data Mining. This is to determine whether high-level representations of learnt features can be transferred to improve existing systems in order to solve new problems. Currently, an overwhelming amount of information is generated and applied to data mining systems. The system explored by this project aims to reduce the number of features by using feature selection methods, improve decision boundaries, the region of a problem space in which the output label of a classifier is ambiguous [ref] by utilising feature construction methods and provide a more compact solution than baseline solutions. Features gathered with feature selection and feature construction methods will be used in conjunction with an LCS. The produced compacted code fragments (CFs), representations of rules linking high-order "abstracted" information about input variables [28], can then be used to solve computer vision classification problems. In addition to this, CFs can be used as building blocks of knowledge from small problems to improve learning in problems of larger scale or similar domains [28].

1.3 Objectives

Given the aim and motivation, these objectives are in place to fulfil the aims and motivations mentioned.

1. The goal is to create methods for extracting and integrating features from layers in a DNN as input features to a LCS to determine if this will enhance the performance on solving classification problems compared to raw features selected by the LCS. The performance will be evaluated on standard data mining benchmark datasets and will be measured on criteria such as accuracy, fitness and the solution's generality (see Section 4 for definitions). This is to be achieved by mid-June 2018 and will be achieved when these sub-objectives are completed:

   (a) Evaluate the performance of a LCS on benchmark datasets such as Breast Cancer Wisconsin [25] and Zoo Data Set [26]. Refer to Section 2.3 for the explanations of why these datasets are used. Evaluation will be measured on criteria such as accuracy.

(b) Create a bespoke image shape dataset. The dataset will contain simple shapes without confounding variables which introduce ambiguity and distraction. Using existing image datasets such as imagenet [23] will increase the complexity of analysing whether the features extracted will convey a rich meaning. The dataset will be evaluated on a CNN for shape classification where evaluation is measured on the accuracy of the training and test sets.

(c) Evaluate whether features extracted from the CNN can be reused by evaluating on our benchmark datasets as well as external benchmarks such as MNIST to determine whether the features extracted convey a rich meaning.

2. The goal is to apply methods for feature selection with feature construction on LCSs where features are selected by a LCS and then constructed manually. The performance will be evaluated on standard data mining benchmark datasets and will be measured on criteria such as accuracy, fitness and the solution's generality. This is to be achieved by mid-August 2018 and will be achieved when these sub-objectives are completed:

   (a) Create bespoke image datasets with more complex combinations of shapes such as cups where the data set is building on the dataset from the first objective.

   (b) Construct embedded, filter or wrapper feature selection methods. These feature selection methods are defined as methods which embed the selection within the basic induction algorithm, methods which use feature selection to filter features passed to induction, and methods that treat feature selection as a wrapper around the induction process [6].

   (c) Evaluate these methods against the raw feature benchmarks. Evaluation will be measured on accuracy, fitness and the solution's generality.

3. The goal is to implement methods for feature construction based on code fragments constructed by the LCS applied to autonomous computer vision classification problems, where the performance will be evaluated on standard data mining benchmark datasets and will be measured on criteria such as accuracy, fitness and the solution's generality. This is to be achieved by the end of September 2018 and will be achieved when these sub-objectives are completed:

   (a) Construct feature construction methods which will further improve the accuracy, fitness or the solution's generality. Feature construction methods can be used to learn more precise decision boundaries.

   (b) Source image datasets relating to real world images and situations such as imagenet [23].

   (c) Evaluate the classifier system with feature selection and feature construction using measurement criteria such as accuracy, fitness and the solution's generality.

1.4 Related Work

Computer vision is a field which has emerged from mathematics and computer science coupled with the psychology of perception and the neurosciences [10]. Many computer vision techniques have drawn inspirations from biological patterns to form algorithms such as CNNs. From the beginning of cognitive Artificial Intelligence where Deep CNNs were popularised by Hinton, arguably the godfather of Deep Learning [16], deep learning techniques have made huge progress in the field of computer vision and image recognition. The Conference and Workshop on Neural Information Processing Systems (NIPS) is a Machine

Learning and computational neuroscience conference held every year which have involved notable publishers such LeCun [17] who have made huge advances in DNNs. However, the performance of DNNs have rapidly outpaced our understanding of the nature of their solutions where more interpretable systems have gained interest. Biological vision still contains largely unknown factors. Unreliable naive introspection is where our visual perceptions are prone to error without the complete understanding of how vision works [10]. Both Deconvnets [31], a system which visualises CNNs in an attempt to understand and improve the system, and "Shape Matching and Object Recognition Using Shape Context" [3] demonstrate a need to understand the systems which we create. This project attempts to continue this work by using a LCS where code fragments can be used to convey richer knowledge which can be transferable to other domains.

1.5 Background Work

1.5.1 Convolutional Neural Networks

Deep-learning methods are representation learning methods with multiple levels of representation [17]. The representations are obtained by composing simple but non-linear layers where each layer transforms the representation from the first level (starting with the raw input) into higher and more abstract representations in consecutive levels. With the composition of enough transformations, complex functions can be learned.

A Neural Network (NN) [11] is a network composed of neurons that have learnable weights and biases. A neuron can be in the input layer, hidden layers, or output layer. Input layer neurons reshape all input data uniformly. Hidden neurons receive some input and perform a non-linear transformation. An activation function is applied to all neurons in the output layer. A back-propagation algorithm is used to train the network.

CNN are a type of multi-layer neural network [17]. One of their main differences to neural networks is their structure. CNNs are designed to typically recognise visual patterns from the pixel images. The convolution layer of a CNN is where most of the computation is done. At a high level, a convolution is a sliding window function applied to a matrix representing image pixel values [27]. A fixed sized filter matrix is applied to the image matrix where each value is multiplied and summed together. A full convolution is completed when the filter is applied to the whole matrix. Using these filters during the convolutional step allows image processing results such as edge detection when the difference between a pixel and its neighbours is taken. Multiple layers of convolutions allow higher-level features to be extracted. For example, a CNN may learn to detect edges in the first layer, detect basic shapes in the second layer and then use the learnt shapes to understand facial features. Low-level pixel detection is useful for the project as a drawback of the LCS is that the induced code fragments will contain the most knowledge if the input data is represented with abstract high-level features. For example, the pixels of a car would not be useful as an input instance. However, the abstract features of four cars, windscreen and number plate would be useful. CNNs have the ability to extract abstract features from pixels by deriving information from the value and position of pixels by sliding filters in the convolutional layer. Therefore, in the first objective, we explore the idea of whether the filters produced in the convolutional layers contain abstract features similar to the perception of what humans see.

1.5.2 Inceptionism

Inceptionism [24], a concept discovered by Google engineers, is used to understand and visualise how deep networks are able to classify difficult problems. Humans classify objects by looking at features. For example, when observing a dog, we can classify it using

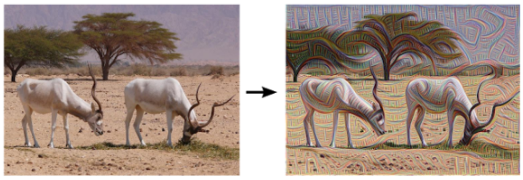


Figure 2: Lower convolutional layers tend to produce strokes or simple ornament-like patterns, extracted from [19]

identifiable features such the object has 4 legs, a tail and paws. Similar to the process of humans, CNNs have demonstrated the ability to observe low-level features such as edges and curves and transform them into more abstract features through convolutional layers. "Each layer progressively extracts higher and higher-level features of the image, until the final layer makes the decision" [19]. See section 3.3 on how inceptionism was used to evaluate whether feature reuse is possible for the project.

1.5.3 Learning Classifier System

LCSs combine a discovery component, usually driven by Evolutionary Computation (EC) methods such as a Genetic Algorithm (GA), with a learning component [28]. In the case of this project, the learning component is performed using supervised learning in order to produce a set of rules which describe the sample space. A rule comprises a condition (specified feature states) and an action (depicting the class) which can then be interpreted as an "IF condition, THEN action" expression [28].

The architecture of a LCS is very flexible where it consists of components interacting with each other. Components can be added, removed and modified to suit the problem in the given domain. As a result, LCSs can be applied to many problem domains which use Machine Learning. There are two main architectures, the Michigan-style architecture and the Pittsburgh-style architecture. Michigan-style accuracy and niche based LCSs have been chosen as the focal point of the project as it has a flexible and well-understood structure [28]. See the design choice in Section 2.2 for more information.

1.5.4 Learning Cycle

Figure 3 shows the learning cycle of one training instance where the main steps are described below. The structure presented is of a simple Michigan-style LCS.

1. The first step in an LCS learning cycle takes a single training instance from the environment which is the input to the population [*P*].

3. Matching is then performed where every rule in [*P*] is now compared to the training instance to gather a set of rules which match [*M*]. A matched rule occurs if all values in the rule condition is equal to the corresponding feature value in the training instance.

4. [*M*] is then split into a correct set [*C*] (rule proposes the correct action from training instance class) and an incorrect set [*I*] (rule proposes the incorrect action).

5. Covering, typically activated in the initial learning cycle when no rules are added into [*C*] or [*M*], acts as a smart population initialisation to ensure that there is at least a correct matching rule in [*C*] as well as ensuring that any rule in the [*P*] will match at least one training instance. Therefore, the LCS will always explore the search space in a progressive manner.

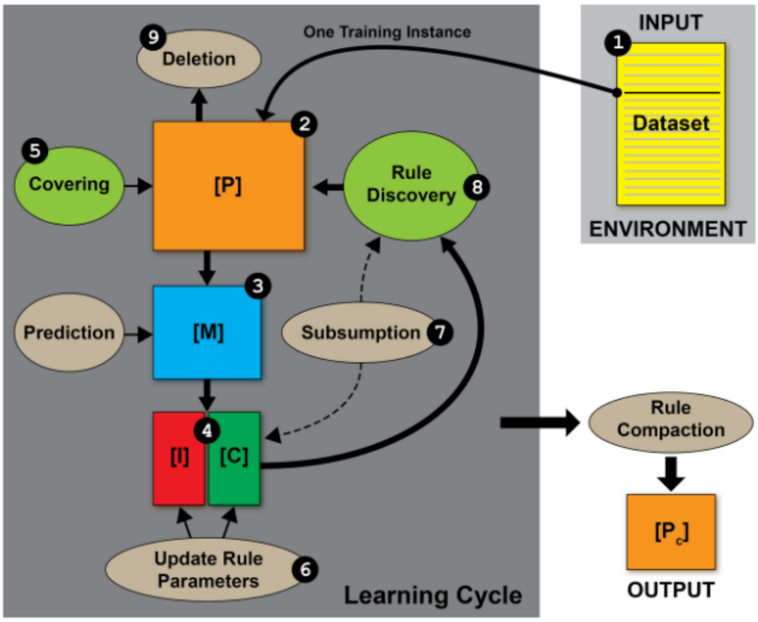


Figure 3: LCS learning cycle, extracted from [28]

7. Subsumption is used to remove rules which are over specific to reduce complexity and improve the efficiency of the cycle.

8. Rule discovery, typically using genetic algorithms selects two parents from [C] based on their fitness. This is done using GA selection techniques such as roulette wheel selection or tournament selection. Crossover or mutation techniques are now applied to generate two offsprings.

1.5.5 LCS generality

LCSs contain a number of components which need to be balanced in order to have a successful system. Accuracy-based systems such as XCS and UCS reward the classification based on the correctness of the predictions. Therefore, a classifier which contains rules which match all the instances is fully specific and fully accurate. However, along with accuracy and fitness, the solution's generality is a criteria that is used in this project. Generality occurs when a classifier can cover more than one problem instance with a single correct action. Problems which can be classified by a single rule is called a niche [28]. The solution obtained from the LCS should be generalised from the pressures demonstrated in Figure 4 which produces an efficient, human-readable and compact solution.

Set pressure is inherent to the structure of the LCS where it is based on the principle that with generality, there is more opportunity to breed. To prevent over generality, set pressure is balanced by fitness pressure. Fitness pressure pushes the rule population towards higher specificity and increased accuracy. Rules that are not fit will not be considered for breeding and may be deleted. Rules evolve towards accurate, maximally general rules ant the generality is supported by subsumption further. However, subsumption differs from set pressure as it focusses on maximally generalising the population representation by deleting over specific rules. Mutation pressure allows the population the opportunity to get out of local maxima by providing diversification.

DNNs do not have generality pressures inherent in the structure of LCS. Introducing dropout [22] and weight penalties such as L1 and L2 regularisation [22] in DNN structures

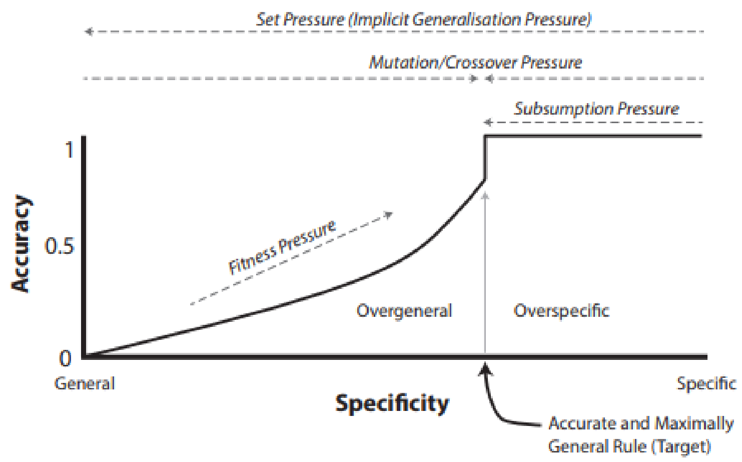


Figure 4: Pressures in an LCS which lead to an accurate and maximally general solution, extracted from [28]

prevent overfitting. However, these techniques require parameters to be set and include a certain degree of human bias. Furthermore, LCSs pressures look at the population generality based on the feature conditions and actions of a rule rather than low-level representations such as weights.

1.5.6 Feature Selection and Feature Construction

Feature selection is a common technique used to obtain a smaller set of more relevant features in order to improve the performance [30]. This is done with some degree of cardinality reduction where the number of features are reduced to improve the quality of the feature set. Maintaining a large amount of features increases the dimensionality of the problem which can be costly in terms of complexity in time and memory as well as decreasing the quality of information. Noisy and redundant data make it more difficult to discover meaningful patterns. Other benefits of feature selection include producing a simpler, easier to interpret model where generalisations are used to reduce overfitting.

Feature construction involves transforming a given set of input features to generate a new set of more meaningful features [18]. Newly generated features take into account the relationships in the previous feature space. Therefore, they are more meaningful and produce more accurate classifiers and decision boundaries.

2 Design

This project uses supervised learning to train a Michigan-style LCS and a CNN. The size and ratio of the datasets used for training has been carefully thought through. External datasets have been chosen to evaluate the classifiers.

2.1 Supervised Training

Supervised ML is the use of ML algorithms that utilise supplied instances to produce a model which then make predictions about future instances [15]. The resulting classifier can then be used to assign class labels to the testing instances where instance features are known, but the value of the class label is unknown.

In contrast, unsupervised learning is where the supplied instances are unlabelled and are generally applied to unsupervised clustering algorithms where the purpose is to discover useful classes of items [14]. Reinforcement learning involves the input information to be provided to the learning system by the environment [2]. The system must discover which actions yield the best reward by trying each action rather than being told which actions to take.

Due to the purpose of the project being to solve classification problems, the training data will contain labelled classes making unsupervised learning technique not applicable. Reinforcement learning usually involves a more difficult process as it must explicitly explore its environment in order to determine how to act. It is less suitable than supervised learning for the problem defined for this project as detailed training instances can be provided.

Supervised Learning is a form of inductive ML, a process where a system tries to induce general rules from a set of observed instances. The image, Figure 5 [15], describes the steps of supervised learning applied to a real-world problem. After the collection of data and separation of individual features and class labels, preprocessing is needed in order to remove noisy or missing feature values. There are many imputation methods which will be explored more if necessary to preprocess a specific dataset. Currently, datasets created for the purpose of validating ideas are used without the need to consider imputation. Feature selection and feature construction methods can then be used, before or during the training of the model.

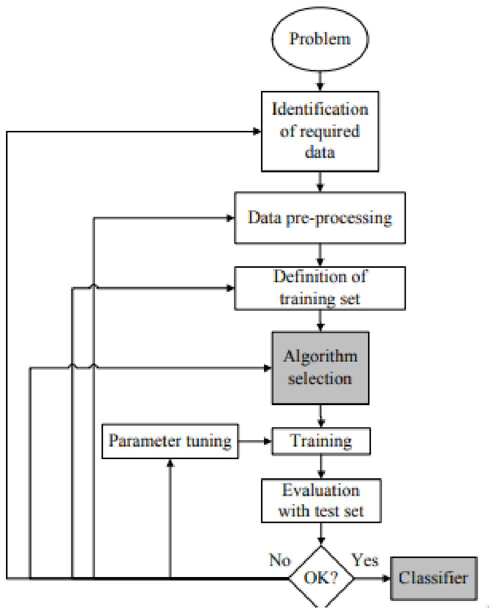


Figure 5: The process of supervised ML, extracted from [15]

2.2 Michigan-style LCS

The LCS paradigm encompasses many different problem types and learning components. There are many types of LCS which target different problem solutions. The two main variants are the Michigan-Style LCS and the Pittsburgh-Style LCS. Pittsburgh-style LCSs are closer in comparison to EC techniques as it evolves a population of solutions [7]. Michigan-style LCS involves the evolution and evaluation of rules individually [7]. Pittsburgh-style LCS suits supervised learning problem domains typical of data mining knowledge discovery from a dataset. Therefore, it runs on an offline learning environment that learns iteratively from sets of problem instances. The rule size generated is typically small and so it would be appropriate for a problem with a known solution size where the number of rules in the optimal solution is known or is small.

In comparison, Michigan-style LCS are often run on an online learning system suitable for reinforcement learning that learns iteratively from single problem instances but can be modified for supervised offline learning systems. Michigan-style LCS is typically more flexible if the solution size is unknown or large. Therefore, the project involves exploring a Michigan approach LCS due to its flexibility. XCS is the most well-known implementation of a LCS. However, it bases its fitness on reinforcement learning methods. UCS [28], an adaptation of XCS for supervised learning, is accuracy and niche based and will be used in this project.

2.3 Data sets

The Wisconsin Breast Cancer [25] and Zoo [26] data sets are used in the evaluation of the baseline LCS. The Wisconsin Breast Cancer dataset has been picked in order to demonstrate that LCSs have the ability to perform when there is feature information missing in the dataset. The Zoo dataset contains imbalanced classes where the mammals class is the majority containing 41 instances compared with the amphibians class which contains 4 classes. The configured LCS was able to classify these datasets to a high degree. See Section 3.1 for results.

A bespoke shapes dataset was created in order to investigate whether CNNs could produce features that can be reused by the LCS. As mentioned in my objective, an existing dataset was not used because the invariance in the images will increase the complexity of analysis unnecessarily.

As this project currently explores the concept of supervised learning, using a training set to train instances on the learning algorithm and a test set for performance measurement is necessary. A training set is a collection of instances from which a classifier is trained. A test set is a collection of instances which is unseen by the classifier but is used for measuring the performance of the learnt classifier. By using a test set, we can validate that the classifier is generalised where overfitting has not occurred. In order of having enough instances for training whilst having a hidden set for evaluation, the well-known ratio 70:30 will be applied on the training and test sets respectively on all learning of the classifiers where this allows enough instances for the system to train on and provides enough hidden instances for validation.

Instead of using a validation set, a collection of instances used in the training to minimise overfitting, other techniques have been used during the training of the CNN. Using a dropout rate, a rate set which refers to ignoring random neurons during the training phase can also reduce overfitting. With the LCS, subsumption and the systems pressure on generality will also reduce overfitting without the need of a validation set.

3 Implementation Accomplished

The first objective mentioned in Section 1.3 has been implemented. The goal was to create methods for extracting and integrating features from layers in a DNN as input features to the LCS. This will determine whether the higher-level features extracted could enhance the performance on solving classification problems compared to raw features selected by the LCS.

3.1 LCS Benchmark Evaluation

- Evaluate the performance of a LCS on benchmark datasets such as Breast Cancer Wisconsin [25] and Zoo Data Set [26]. Refer to Section 2.3 for the explanations of why these datasets are used. Evaluation will be measured on criteria such as accuracy.

For the project to succeed, there must be a good foundational understanding of how LCSs work in order to understand the specificities for the configuration of the system. The training and test set for each benchmark dataset has been split using a 70:30 split. This ratio ensures that the model has sufficient instances to learning from but also to provide a number of hidden instances to evaluate on. The dataset is also manually stratified in order to ensure a balance of all classes both the training and test set. For example, the Wisconsin Breast Cancer dataset has a 65:35 distribution of benign:malignant data. This is to ensure that the accuracy of the training will reflect in the accuracy of the test set. There are several imputation methods which can be applied to LCS. However, for easy and quick results, instances have been removed in the Breast Cancer Dataset. This is sufficient due to the distribution of the 2 classes and the number of instances being of adequate size (699 Instances). Machine Learning classifiers are often sensitive to imbalanced training datasets as a result to the proportions of the different classes which can lead to the system favoring majority classes. However, the LCS was able to classify the Zoo data set with a reasonable outcome. Results are run 30 times and averages are presented below to ensure consistency. Evaluation is done on accuracy. See Section 4 for the definition.

Results in the table below show that the trained LCS is sufficient in solving classification problems as well as having the capabilities of extracting knowledge from imbalanced domains such as the problem presented by the zoo dataset.

Dataset	Training Accuracy	Test Accuracy
Zoo dataset	100%	97.76%
Breast Cancer Wisconsin Dataset	98.46%	97.96%

3.2 Create a data set

- Create a bespoke image shape dataset. The dataset will contain simple shapes without confounding variables which introduce ambiguity and distraction. Using existing image datasets such as imagenet [23] will increase the complexity of analysing whether the features extracted will convey a rich meaning. The dataset will be evaluated on a CNN for shape classification where evaluation is measured on the accuracy of the training and test sets.

A bespoke dataset of shapes was made in order to determine whether CNNs produced information-rich features. The dataset consists of randomly generated, black and white, class balanced, 28 X 28 basic images such as lines, circles, ellipses, squares, rectangles and triangles as seen in Figure 6. It is hypothesised that shapes are the building blocks for more complex objects. Therefore, extracting the features produced by the convolutional layers may produce features which can then be used by the LCS.

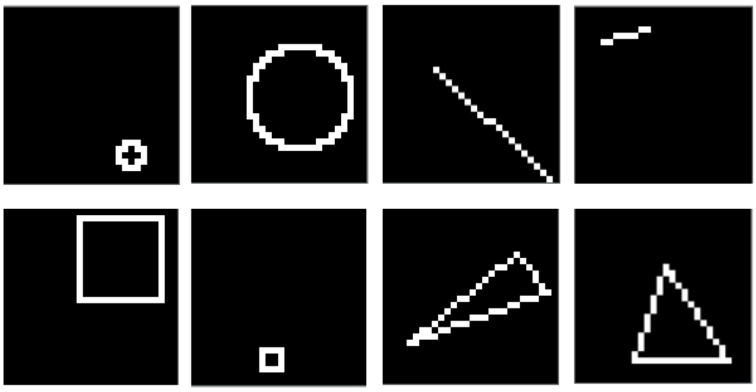


Figure 6: Input images to CNN

The structure of the CNN is shown in Figure 7 where there are two convolutional layers. The model extracted had been trained on 10000 steps with 400 of each shape for training. The test set consists on 120 of each shape. Accuracy of both training and test set could reach 100%.

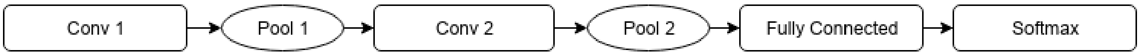


Figure 7: Structure of CNN used

3.3 Reusing features

- Evaluate whether features extracted from the CNN can be reused by evaluating on our benchmark datasets as well as external benchmarks such as MNIST to determine whether the features extracted convey a rich meaning.

The technique of inceptionism can be used to validate that the essence of an image has been extracted. The filters produced by the training of the CNN from each convolutional layer was used as inputs by the trained CNN where predictions were outputted. The image of squares has consistently determined that straight parallel or perpendicular lines are an essential feature of squares. Using the straight parallel lines image as an input has typically resulted in the prediction of lines. This supports the idea the what humans identify as a feature of a line is similar to the CNN.

The classifier also predicted lines significantly more than other classes. With 1280 images produced by the first convolutional layer, 509 predictions were lines (39.9%), 279 were triangles (21.9%), 269 (21.0%) were squares and 223 were circles (17.4%). This affirms the belief that shapes like triangles and squares are made up of lower-level features such as lines. Figure 8 shows the composition of each class.

Figure 9 shows examples of the images produced by the first convolution along with the predictions made by the classifier. This visualisation gives us more of an idea of the reasons behind the classification. For example, the third image shows 2 parallel lines, where the classifier classified the image as 76% line.

Furthermore, the MNIST dataset, an image set of hand-drawn digits, was used to determine whether the features produced by the convolutional layers were able to identify the shapes produced. A sample of images from the MNIST test set was used to evaluate

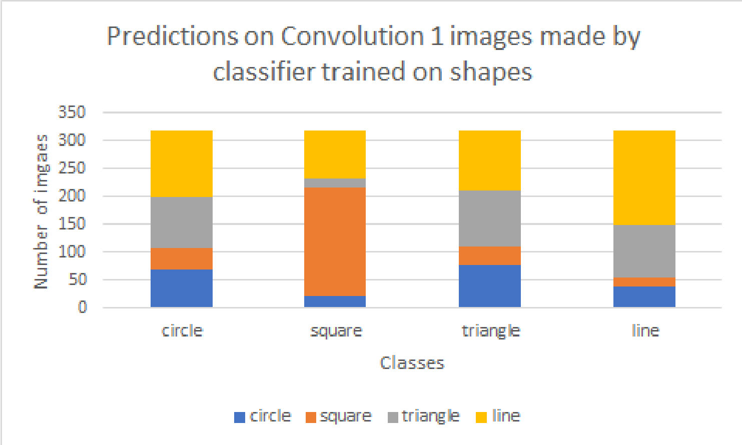


Figure 8: Prediction made in first convolutional layer

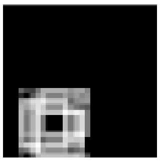
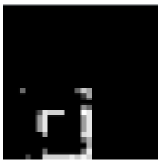
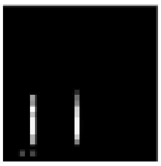
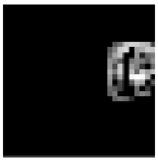
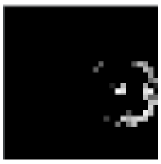
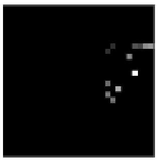
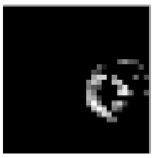
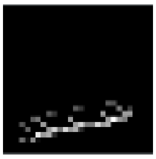
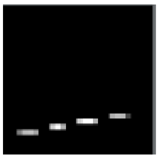
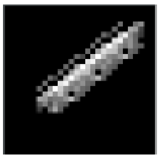
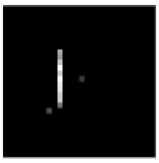
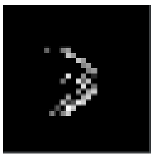
				
Square	Predict: 75% square	Predict: 78% circle	Predict: 76% line	
				
circle	Predict: 71% square	Predict: 73% circle	Predict: 81% line	Predict: 74% triangle
				
line		Predict: 77% circle	Predict: 91% line	Predict: 98% triangle
				
triangle			Predict: 71% line	Predict: 98% triangle

Figure 9: Example filters from the first convolutional layer

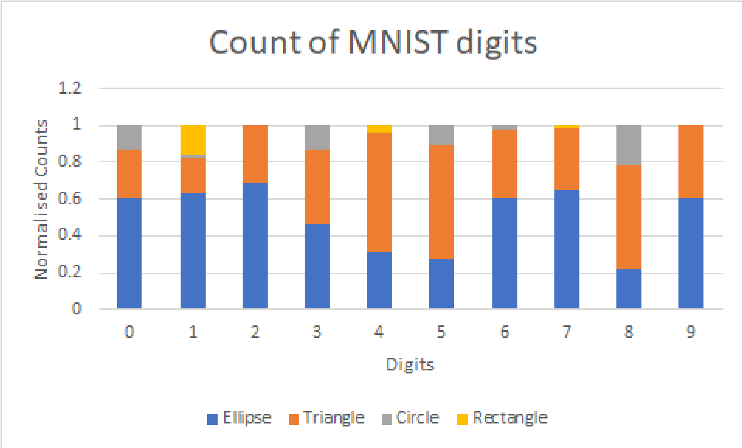


Figure 10: Count of predictions made from MNIST dataset

whether the features determined by the CNN could be interpreted by humans. For example, a 0 would be represented by a hollow circle and an 8 would be represented by two hollowed circles stacked vertically.

The original shapes dataset was adapted to have a white infill in order to represent the mnist data better. As even-distanced shapes such as circles and squares are less likely to occur in handwritten numbers, the dataset has further been modified to include ellipses and rectangles. During the creation of the shapes dataset, even-sided rectangles would be discarded (as they would classify as squares) and vice versa with circles and ellipses. Looking at hand drawn 0s, we would expect that a CNN which had been training on the shape dataset would classify the number as an ellipse or a circle due the roundness of edges. Results below show the count of predicted classes [square, rectangle, circle, ellipse, line, triangle] for each MNIST class taken from a sample of 500 instances. The most popular predicted class was the ellipse. This is due to the handwritten images containing unsymmetrical curve-like edges. Unsurprisingly, lines, a pixel in width, was not predicted. Squares were also not predicted where it is unlikely that the mnist dataset would contain any images with shape orthogonal edges.

Figure 11 is an example of the classifications performed by the CNN trained on white-filled shapes. All images show handwritten sevens. The first image, classified as a rectangle, shows the 7 with very straight perpendicular lines. The second image, classified as a triangle, represents the 7 with an acute angle similar to triangles. The third image, classified as an ellipse, represents the 7 with an angle which looks more curved than the second image.

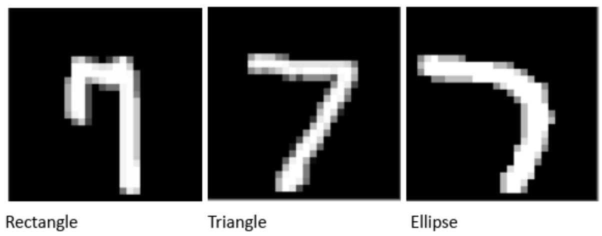


Figure 11: Prediction of handwritten sevens from mnist dataset

4 Evaluation

To prevent the failures relating to a waterfall model approach on the project, evaluation methods are provided for each objective mentioned in Section 1.3. This methodology will validate that the findings and ideas are meaningful at earlier stages of the project. Accuracy, fitness and the solution's generality are the criteria chosen for evaluation.

For classification evaluation for LCS, accuracy is used. Accuracy is the proportion of correct classifications among all classifications [28]. UCS is an accuracy-based LCS where accuracy is used to measure and evaluate the model whilst it is training as well as during evaluation against a test set.

Fitness (a representation of value or worth to a given rule)[28] is another measurement to evaluate whether a specific rule is fit or not. A fitness measure relevant to supervised learning is the number of correct data classifications (action of the rule is equal to the known action from the input data) divided by the total number of times the rule has matched the input data. Fitness can be seen as the long-term accuracy of the *i*th classifier.

The solution's generality is also a criteria of interest as LCS can output many solution spaces. Simple solutions are prefered, containing a small set of rules which cover a large range of niches, groups of instances accurately covered by an optimal rule [28].

Due to many ML techniques being stochastic, including LCSs, further tests may be beneficial in order to affirm that the results found in the project. A Student's T Test compares two averages and conveys the differences from each other. This evaluation conveys whether the probability of whether the results could have occurred through chance or whether the findings convey some meaning [5]. Therefore, the results presented will convey meaning with more certainty and confidence. An assumption of using the Student's T Test is that the results will present itself as a Gaussian distribution. The classifier results will need to conducted at least 30 times in order to for the sample size to be large enough to represent a Gaussian distribution [5]. However, if the assumption does not hold, another statistic analysis such as the Wilcoxon's Ran-Sum test [29] may be needed.

5 Future Plan

The plans for the project involve completing the objectives mentioned in section 1.3. Considering that the project is now at its halfway point, there is still a fair amount of work where the first objective needs to be wrapped up and work on the second and third objectives need to be started. The wrap up of the first objective will involve exploring using the second convolutional layer to see if higher level features constructed by the CNN will be more helpful than the images produced by the first convolutional layer. Later objectives will provide insight in whether the reuse of features will improve results by integrating the CNN and LCS systems. The timeline of each objective has been scheduled in Figure 12:

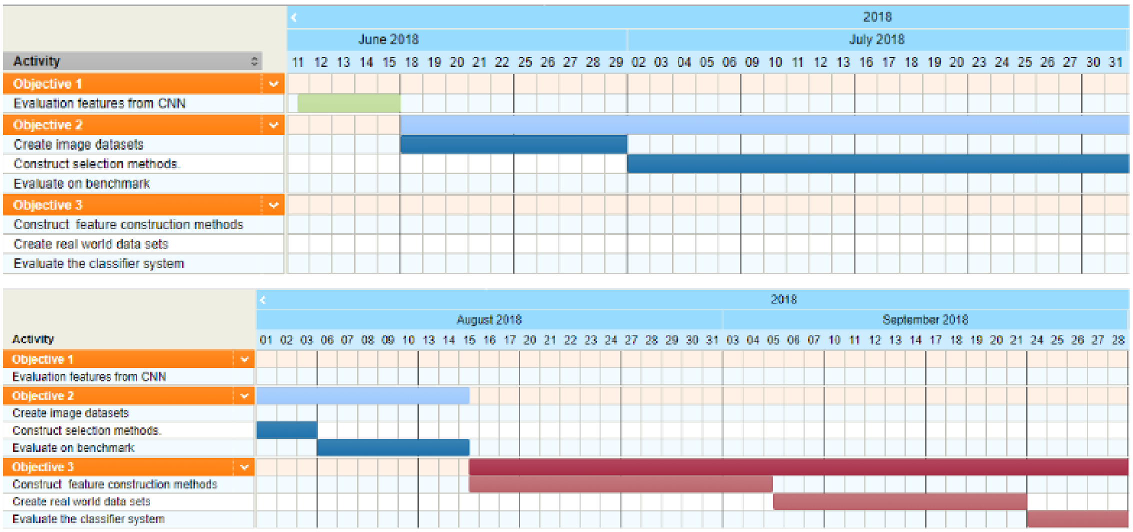


Figure 12: Gantt chart

References

[1] ALBALA, M. Making sense of big data in the petabyte age, June 2011.

[2] BARTO, A. G., AND SUTTON, R. S. Reinforcement learning in artificial intelligence. In *Advances in Psychology*, vol. 121. Elsevier, 1997, pp. 358–386.

[3] BELONGIE, S., MALIK, J., AND PUZICHA, J. Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence 24*, 4 (2002), 509–522.

[4] BERNADÓ-MANSILLA, E., AND GARRELL-GUIU, J. M. Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evolutionary computation 11*, 3 (2003), 209–238.

[5] BLAIR, R. C., AND HIGGINS, J. J. A comparison of the power of wilcoxon's rank-sum statistic to that of student's t statistic under various nonnormal distributions. *Journal of Educational Statistics 5*, 4 (1980), 309–335.

[6] BLUM, A. L., AND LANGLEY, P. Selection of relevant features and examples in machine learning. *Artificial intelligence 97*, 1-2 (1997), 245–271.

[7] BUTZ, M. V. Learning classifier systems. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation* (2010), ACM, pp. 2331–2352.

[8] CLIFTON, C. Defintion of data mining. `https://www.britannica.com/technology/data-mining`.

[9] DENG, L. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine 29*, 6 (2012), 141–142.

[10] HARTLEY, R., AND ZISSERMAN, A. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[11] HAYKIN, S., AND NETWORK, N. A comprehensive foundation. *Neural networks 2*, 2004 (2004), 41.

[12] HOLLAND, J. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan, USA, 1975.

[13] ILKKA KARANTA, T. H., AND KOISTINEN, P. Cognitive biases in neural networks - a study in reasoning under uncertainty. *Publications of the Finnish Artificial Intelligence Society (FAIS)* (1992).

[14] JAIN, A. K., MURTY, M. N., AND FLYNN, P. J. Data clustering: a review. *ACM computing surveys (CSUR) 31*, 3 (1999), 264–323.

[15] KOTSIANTIS, S. B., ZAHARAKIS, I., AND PINTELAS, P. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering 160* (2007), 3–24.

[16] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[17] LECUN, Y., BENGIO, Y., AND HINTON, G. Deep learning. *nature 521*, 7553 (2015), 436.

[18] MARKOVITCH, S., AND ROSENSTEIN, D. Feature generation using general constructor functions. *Machine Learning 49*, 1 (2002), 59–98.

[19] MORDVINTSEV, A., OLAH, C., AND TYKA, M. Inceptionism: Going deeper into neural networks. *Google Research Blog. Retrieved June 20*, 14 (2015), 5.

[20] NAISBITT, J., AND CRACKNELL, J. Megatrends: Ten new directions transforming our lives. Tech. rep., Warner Books New York, 1984.

[21] NGUYEN, A., YOSINSKI, J., AND CLUNE, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 427–436.

[22] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research 15*, 1 (2014), 1929–1958.

[23] STANFORD VISION LAB, STANFORD UNIVERSITY, P. U. imagenet.

[24] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V., RABINOVICH, A., ET AL. Going deeper with convolutions. Cvpr.

[25] UCI. Breast cancer wisconsin (diagnostic) data set. Tech. rep., University of California, Irvine.

[26] UCI. Zoo data set. Tech. rep., University of California, Irvine.

[27] UFLDL. Feature extraction using convolution. Tech. rep., April 2013. (Accessed on 06/11/2018).

[28] URBANOWICZ, R. J., AND BROWNE, W. N. *Introduction to Learning Classifier Systems*. Springer, 2017.

[29] WILCOXON, F. Some rapid approximate statistical procedures. *Annals of the New York Academy of Sciences 52*, 1 (1950), 808–814.

[30] YU, L., AND LIU, H. Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research 5*, Oct (2004), 1205–1224.

[31] ZEILER, M. D., AND FERGUS, R. Visualizing and understanding convolutional networks. In *European conference on computer vision* (2014), Springer, pp. 818–833.