Deep Learning Networks for Target Recognition

by Roan Song

prepared for A. K. Mishra

Department of Electrical Engineering

University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Electrical and Computer Engineering.

September 2016

Blank page

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature: ....
ROAN SONG

Date: ....

Acknowledgements

Abstract

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 2

Literature Review

2.1 What is a Literature Review (Temp Section)

A lit review...

- surveys related research
- summarizes related research
- links to related research

Gives an indication that I know what is out there, show a sufficient depth of knowledge in the field, and provide context for my research within the broader fields surrounding it.

i.e. the purpose is to...

- define terms / describe theories
- **justify decisions made**
- what am I building on
- view of alternate approaches / related work

Organise sections by theme. Compartmentalise and synthesise, bringing views and insights together into what is used in the project (good to indicate where in the report it is used).

Short-ish literature study (20-30% of manuscript). One to two references for each issue. I must choose well-accredited references. Peer review journal papers and conference papers are preferable. Web pages, forums, tutorials and datasheets are *okay* but not ideal. Use reference spreadsheet? Key points, authors, date, title/citation, notes, abstract/relevant snippets.

Topics to cover in this literature review:

- Classification
- Nearest Neighbour (naïve benchmark method)
- Multilayer Perceptron (step towards deep learning)
- The concept of 'deep' neural networks - usefulness is using perceptron + convolutional stuff
- Optimization (backpropagation really)
- the MSTAR data set
- image processing (should I be rescaling up or down?)
- direction for expansion (going deeper)

Also need to write a progress report

2.2 The MSTAR Dataset

2.2.1 Description

The MSTAR public mixed targets dataset contains X-band synthetic aperture radar (SAR) image chips of multiple targets. Each image has a resolution of 1 foot, and is captured in spotlight mode. Information pertaining to each target, including its elevation, depression angle, and target type is contained in a header section of each file. The header is followed by magnitude and phase data of the SAR imagery. Only the magnitude data shall be considered, to reduce the complexity of this study to that of image-based target recognition. [1]. The MSTAR dataset has been referenced in multiple publications, and is a popular choice of SAR image chips for machine learning endeavours.

2.2.2 Relevance

The MSTAR dataset was suggested for this study by A. Mishra. It provides a generic SAR image chip dataset on which any classification methods can conceivably be run, with minimal setup and hassle. The number of files available is enough to test any classifier sufficiently .

2.2.3 Separability Index

Thornton's Separability Index (SI) is used as a metric to show how separable the data is into its different classes, i.e. the confidence level with which distinct classes can be identified with no overlap. Perfectly separable data would have an SI of 1, or 100

2.3 Classification

2.3.1 Nearest-Neighbour Classification

Description

The nearest neighbour classifier operates as follows: When given an input, the classifier compares this input to the training data set, and finds the one that is closest to the input. For example, if men and women were to be classified by their heights, a given input would be classified as either male or female based on the data point in the training data with the height closest to that of the input. This can be expanded to multiple features/dimensions by taking the Euclidean distance between the input and each instance in the training data set. For images, this amounts to comparing, pixel by pixel, each pixel value, and finding the L2 distance between them. Note that there is no need to apply the square root to the distance; it is a monotonic operation, so it will not affect the ordering of the values, and will introduce additional computational complexity. The equation for calculating the L2 distance is:

$$d_2(a, b) = (a - b)^2$$

Each pixel usually has some relation to the pixels near to it, so there is the possibility for a better definition of 'distance' between images to be made [2, 3].

The nearest neighbour classifier has been shown to provide high classification rates (85+%) through sufficient image processing and classifier development. A caveat of the success, however, is the inclusion of image chip 'clutter' present in each image affecting the success of clarification. Images in the training dataset have similar clutter, and there is a non-negligible contribution to the success of the classification. [1]

Relevance

The nearest-neighbour classifier is used in this study as an example of a naïve classifier. Results of this classifier provide a good benchmark against which subsequent classifier performance can easily be measured. The success of other parties in classifying the MSTAR targets using nearest neighbour methods lays a convincing foundation for future development. There is undoubtedly room for improvement, beginning with the elimination of clutter's effect on classification.

2.4 Deep Learning

The objective of this study is to test the performance of deep learning-based classifiers on SAR image chip data. The success of naïve methods has already been proven [1], but lack the predictive power of a truly intelligent classifier. Neural networks and the application of deep learning are key to extracting features from the data to further improve classification rates.

2.4.1 Neural Networks

Description

A neural network is a system inspired by the perceived workings of the human brain; a system of neurons combine to perform tasks that exceed their individual capabilities. An input is passed through a series of neuron layers, each of which is tuned to identify characteristic features of the input between each layer, allowing for feature extraction and identification.

(From the hartford.edu site) A neural network consists of four main parts [4]:

1. Processing units{uj}, where each uj has a certain activation level aj(t) at any point in time.
2. Weighted interconnections between the various processing units which determine how the activation of one unit leads to input for another unit.
3. An activation rule which acts on the set of input signals at a unit to produce a new output signal, or activation.
4. Optionally, a learning rule that specifies how to adjust the weights for a given input/output pair.

Relevance

The motivation behind using neural networks is simple; instead of specifying basic characteristics for a system to detect, the system is given an input and a matching output and is left to develop its own perceptions of what important feature link the two. Through optimisation and iteration this can become a very successful form of classification.

2.4.2 Multilayer Perceptron

Description

A multilayer perceptron is an extension of the simple perceptron network. The initial design was an input layer, an output layer, and a hidden layer of neurons between. The perceptron was deemed unsuitable for complex classification tasks, as its input-output relationship could be reduced (via back-propagation) to a set of linear combinations. This means that is can only correctly classify data that is perfectly linearly separable, which limits its practical applications significantly.

The multilayer perceptron used two or more hidden neuron layers, with non-linear activation functions at each neuron. This makes a multilayer perceptron an example of a deep neural network. The depth allows for more complex feature detection, and ultimately better performance in complex classification tasks. The multilayer perceptron becomes difficult to optimise as the number of hidden layers grows, because the effect of each neuron on the output, and the effects of previous neurons on **that** neuron become progressively and recursively difficult to compute.

Relevance

Implementing a multilayer perceptron is a valuable exercise, as it provides an example of a deep neural network with a fairly straightforward implementation. Training time becomes a significant factor when using a deep neural network due to the time taken to complete back-propagation optimisation, so using a multilayer perceptron will force the development of more efficient methods of data pre-processing to speed up the training as much as possible.

2.5 Optimization and Training

A classifier is only operating efficiently when it is tuned to the data it is attempting to classify. Deep neural networks are initialised with random weights between their neurons, and at first use will perform worse on average than naïve classification methods. Through optimisation of these inter-neuron weights, however, the potential of deep neural networks can be reached, and classification results are expected to significantly improve. Tuning the classifier to the dataset is crucial, but optimising too heavily may result in *overfitting* of the data, leaving the classifier with no predictive power.

2.5.1 Back-propagation

Description

Back-propagation is a system by which the effects of weights between neurons is adjusted through an iterative process. The base case is that of a single input, single output system. Varying the weight on the input directly effects the output. This change can be easily recognised, and the weight can be changed to more suitably link the input to the desired output. This involves developing a method of changing weights in a sensible manner. The most common form of this is through *gradient descent*, whereby the weights are adjusted corresponding to their perceived effect on the output state, and their rate of change. Back-propagation is not guaranteed to find a global minimum, and can settle on a local minimum instead, which can be somewhat alleviated through the use of random weights and multiple training rounds, before choosing the best version of the classifier that has been discovered.

One of the key issues with back-propagation is its computational complexity. With deep neural networks, the sheer number of weights and their possible combinations make discerning their impact on the output very difficult, and computationally infeasible to perfectly optimise.

Relevance

Back-propagation is a popular and successful technique, well-suited to neural networks with only a few layers. With enough time, it can help to optimise much larger networks, and potentially improve classification accuracy by a large margin. It alleviates the concern of trying to find the perfect network from the offset; it allows any network to be tuned to be better than it currently is.

2.6 Performance Evaluation

2.6.1 Cross Validation

2.7 Dimensionality Reduction

2.7.1 PCA

Chapter 3

Methodology

3.1 Nearest Neighbour Classification

The steps necessary to implement the NN on the MSTAR dataset are as follows:

- Convert the raw + header MSTAR files into .tiff images
- figure out how to read and display these images in matlab
- write a pixel-by-pixel comparison method
- test the method by comparing one image to itself and others
- collate a set of mixed radar targets for testing
- extract data from each image/filename to help with seeing if the classification is correct
- TEST
- collect results!

After testing the Nearest Neighbour algorithm on a set of single-target data, it is necessary to see how well the algorithm performs when other targets are added to the mix. A subset of 10 images of each target is selected, and added to the testing pool.

A system needs to be made that automates the rotation of images and testing, in order to test the Leave-One-Out cross-verification accuracy.

After rescaling the images, predictive accuracy dropped to just 6/70. This is because scaling the images without accounting for the effects of their size leads to smaller images having a much smaller cumulative distance. Weighting the distance result by the number of pixels in an image rectified this immediately, bringing the predictive accuracy up to 56/70!

3.2 Data Processing

The first step is to get to grips with the data - processing the data set in a way that makes sense to use, and allows different classification methods to be implemented easily on it.

The MSTAR dataset contains eight different targets. The dataset is sorted by depression angle, and by 'scene'. All of the targets have data corresponding to a 15° depression angle. Some targets have additional angles available, but the 15° set is the one that will be used.

For supervised learning, we want to have each target in the dataset labelled with corresponding information, most notably its class. This is often done using a one-hot array (i.e. [1 0 0], [0 1 0] for 1 and 2, respectively) relating to each target. The MSTAR dataset stores the information of each target in a header section of each file. This is inconvenient when reading in image files directly, so I will most likely process the data and create the identifying arrays from the header data. Since I will have defined the format in a way that is convenient for me to use, it will make my life a lot easier.

3.3 Dimensionality Reduction

An image can be viewed as a collection of pixels, comprising a feature vector. If the size of this vector can be reduced while retaining enough information for classification, we can greatly increase our training and classification speeds. Simply selecting every other pixel (or one in five) would greatly reduce the dimensionality of the data but may remove features that are key to classification, thus having a negative impact on the results.

Chapter 4

Design

This chapter covers the design process of the study, including the

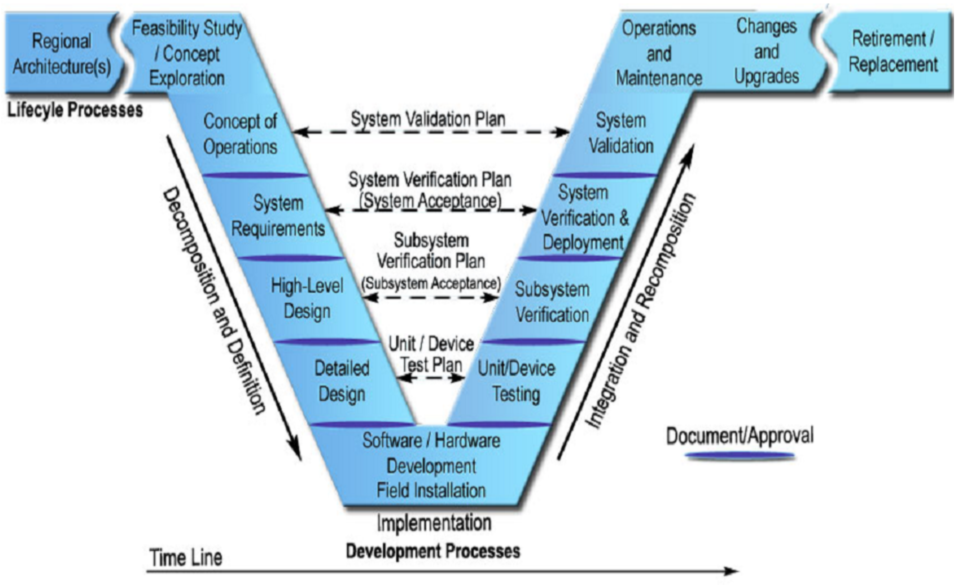

Figure 4.1: Vee Diagram

4.1 Design Context

This study is within the scope of an undergraduate level approach to both radar and machine learning, with an emphasis on machine learning; the techniques applied herewith are not limited to radar imagery, but attempts should be made to tailor the design to radar applications. With development this study should be adaptable to commercial use, and be helpful to people in the radar department who need assistance with radar target classification. Thus the study should develop an easily extensible framework for radar image classification, or at least guidelines to allow others to integrate with the work covered herein.

4.2 Feasibility Study / Concept Exploration

4.3 Decomposition and Definition

This section is devoted to describing the study in terms of its requirements, operation, and implementation. An accompaniment explaining the verification and validity of each subsection will be in the next section.

4.3.1 Concept of Operations

Radar target classification is an inexact science; interpreting a radar image and comparing it to a known case is not as straightforward in all cases as one might expect. Weather conditions, environmental clutter, and image resolution all obscure the target to varying degrees, making intuitive classification ineffective. Computer-based classification through analysis of multiple targets and the application of deep learning techniques should in theory allow distorted images to be classified after the computer is trained to recognise features pertaining to each class. Naïve methods of classification lack predictive power - the ability to 'guess' effectively if the target is obscured or unrecognised. Deep learning methods are the solution that this study proposes.

4.3.2 System Requirements

4.3.3 High-Level Design

For this study, these areas of design need to be focused on:

- Image Processing/Preparation
- Further Dimensionality Reduction
- Naïve Classification (Nearest Neighbour as a benchmark)
- Deep Learning Classification (Multilayer Perceptron)

4.3.4 Detailed Design

Image Processing/Preparation

The MSTAR dataset is a compilation of image chips, all of which contain a header, as well as magnitude and phase data. The images are between 54x54 and 192x193 pixels in size, which suggests that some form of image processing should be performed to make sure that all images are the same size. The targets in each image chip are centred, suggesting that cropping each image to a size where the target (and its shadow - useful in classification) are left whole, and as much of the surrounding clutter as possible is removed.

An alternate approach is to retain the data inherent in the environmental clutter and pad the smaller images with zeros, keeping all images in the set at the size of the largest image in the set. While this preserves all of the image chip data, processing larger images leads to infeasibly long training and classification times.

To accurately preserve the data while making the image as small as possible, the images will be cropped in a rectangular box, containing the target and its shadow. The crop dimensions will be chosen to consider all of the image chips. Further processing of the images may be done, to allow for quicker classification of a larger number of images.

Dimensionality Reduction

Each pixel in an image is taken as an feature, forming a feature vector with a length equal to the total number of pixels in the image. The image cropping mentioned in section 4.3.4 is very effective at reducing the size of this feature vector. The image size is reduced to ———- (a total of —– features). This can be reduced further through the application of dimensionality reduction techniques, such as PCA, LLE, SOM and non-linear methods.

The first foray into dimensionality reduction will be conducted using PCA techniques 2.7.1(cover in the literature review)

Nearest Neighbour Design

The high-level design is fairly straightforward, and is shown in figure 4.3.4. Its principles of operation are covered in section 2.3.1. To implement this classifier, the following is needed:

- Access to the dataset
- A choice of input image
- A method to calculate and sum the pixel-wise distances
- A variable storing the smallest distance and tentative classification
- A method displaying the chosen class and whether or not it is correct

Multilayer Perceptron Design

Implementation of a multilayer perceptron in software can be divided into discrete sections as follows:

-

4.3.5 Software Development and Implementation

—– The software present in this study has been made available on Github

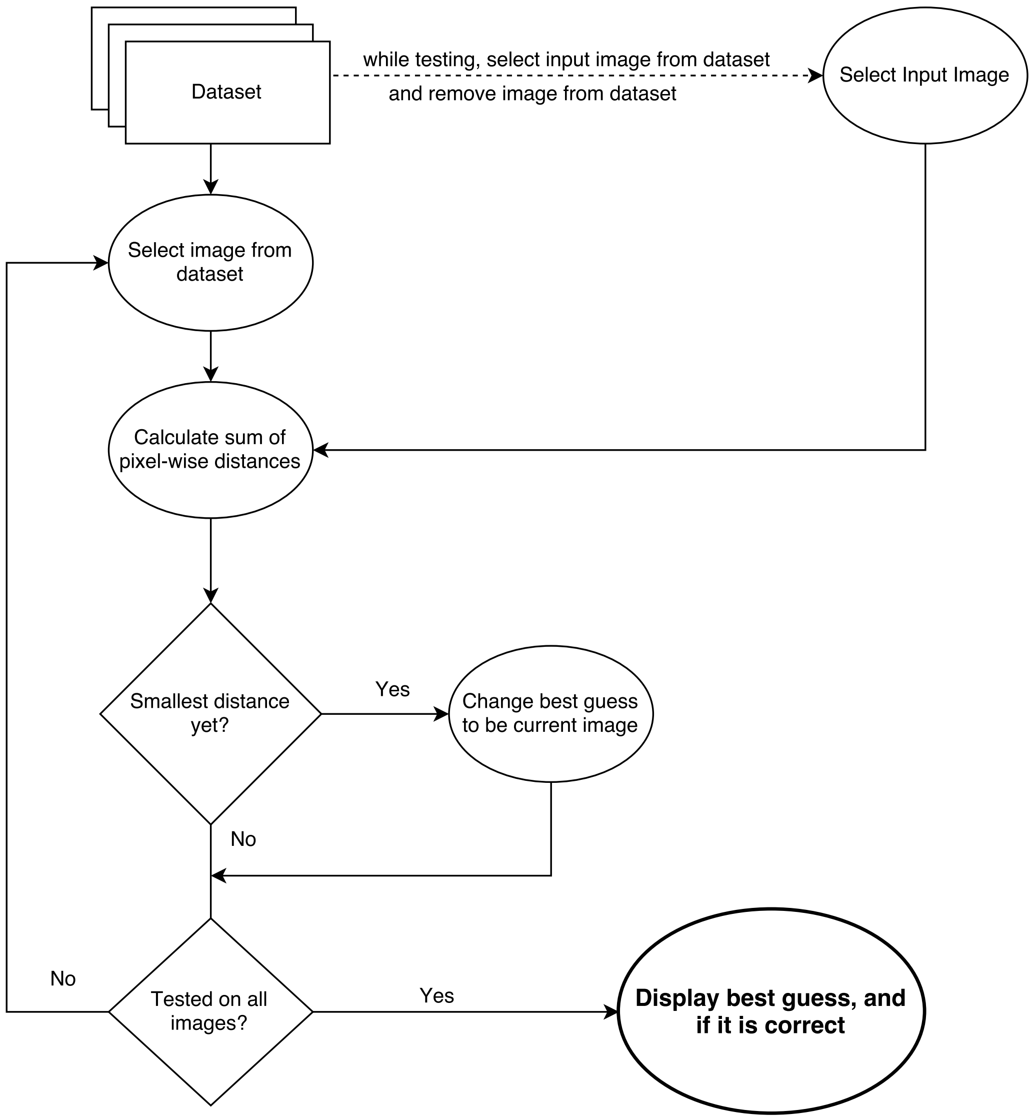
Dataset
while testing, select input image from dataset
and remove image from dataset
Select Input Image
Select image from dataset
Calculate sum of pixel-wise distances
Smallest distance yet?
Yes
Change best guess to be current image
No
No
Tested on all images?
Yes
Display best guess, and if it is correct

Figure 4.2: Nearest Neighbour Classification Flow Diagram

4.4 Integration and Recomposition

4.4.1 Software Development and Implementation

Algorithm Testing

For each classifier, their effectiveness on the MSTAR dataset must be tested and catalogued, beginning with the Nearest Neighbour implementation. Doing the Nearest Neighbour classification first establishes a benchmark against which further methods can be tested. The naïve nature of the Nearest Neighbour classifier means that it is not optimised for any particular dataset. It shows the efficacy of a generic algorithm applied to the MSTAR dataset.

For the effectiveness of any algorithm to be tested, correct and incorrect outputs must be defined. The MSTAR dataset includes target labels in a header section of each file, but since the operations are conducted on files with their header and phase data stripped away it merely adds computational complexity to find the corresponding header for each file and then parse it to extract data about each target. To simplify classification, a variant of a one-hot vector denoting the classes is attached to each target.The vector consists of a series of numbers, equal in length to the number of classes in the dataset. A '1' denotes that the instance is a member of the class corresponding to that entry in the vector, and the rest of the numbers are -1, showing that the instance is not in those classes. A file is created listing all of the filenames to be tested during the run of the algorithm. An example file with ten entries and two classes would look as follows:

```
HB03333.003.tiff 1 -1
HB03334.003.tiff 1 -1
HB03335.003.tiff 1 -1
HB03337.003.tiff 1 -1
HB03338.003.tiff 1 -1
HB14931.025.tiff -1 1
HB14932.025.tiff -1 1
HB14933.025.tiff -1 1
HB14934.025.tiff -1 1
HB14935.025.tiff -1 1
```

Testing During Development

The simplest way to find a classifier's efficacy is to test it on a wide variety of classes and on as many test instances as possible. To provide interim results, during the iterative phase of classifier development, only a subset of the dataset's images are used. This compromises the final accuracy of the classifier (it may perform differently on the full dataset), but bring with it the ability to test and train classifiers more quickly, due to the lower computational overheads. During the development this testing method allows for simple decisions regarding the direction of classifier implementation or optimisation to be made. In process of verifying the classifiers, the full dataset must be used to give an accurate picture of the classifier's performance and real-world implementation.

4.4.2 Subsystem Verification

The chosen method for verifying a classifier's efficacy once it is considered to be sufficiently optimised is simple; The classifier is tested on the dataset with a number of test instances; if time allows, leave-one-out cross-correlation (LOOCV) covered in section 2.6.1 will be used. This entails removing one instance from the dataset, training the classifier on the remaining points, and using the removed instance as a test case. Once this has been done, the instance is replaced, another is taken, and the process is repeated until every instance in the dataset has been tested. The system's performance is based on the number of correct classifications made during the process, and also gives a good estimate of the Separability Index of the data (section 2.2.3.

4.4.3 System Verification and Deployment

4.4.4 System Validation

4.4.5 Operations and Maintenance

4.4.6 Changes and Upgrades

4.4.7 Retirement / Replacement

Chapter 5

Results

Chapter 6

Conclusion

References

[1] R. Schumacher and K. Rosenbach, “Atr of battlefield targets by sar classification results using the public mstar dataset compared with a dataset by qinetiq, uk summary.”

[2] L. Wang, Y. Zhang, and J. Feng, “On the euclidean distance of images,” Ph.D. dissertation, School of Electronics Engineering and Computer Science, Peking University, 2005. [Online]. Available: http://www.cis.pku.edu.cn/faculty/vision/wangliwei/pdf/IMED.pdf

[3] D. Michie, D. J. Spiegelhalter, and C. Taylor, “Machine learning, neural and statistical classification,” 1994.

[4] I. Russel, “Definition of a neural network,” 1996. [Online]. Available: http://uhaweb.hartford.edu/compsci/neural-networks-definition.html

Appendix A

Progress Report

As of the 16th of September, certain progress has been made in three areas of the report, namely the Literature Review, Methodology, and Implementation.

A.1 Literature Review

The Literature Review has been somewhat written; the core topics of most relevance to the study have been covered, but the level of detail for each topic varies, and I feel it may be insufficient. This is a section that I feel needs further refinement and content, that will be added as the development of this study touches on more topics.

A.2 Implementation

I have implemented a working nearest neighbour classification system, but it is quite slow. I have implemented the first few stages of a multilayer perceptron with two hidden layers, and am currently busy with the implementation of a back-propagation algorithm to optimise the network.

A.3 Areas of Focus

The next avenue of development is to process the MSTAR image chip data to facilitate quicker classification. Until now I have been using only a subset of the MSTAR data (70 image chips), but this is not feasible in the final product, as it does not include sufficient training data. The time taken to process this subset is non-negligible, and something must be done to improve the training time. The idea is to process the images in such a way that the resolution of each image, and therefore the number of pixels to process, can be reduced while retaining all the crucial data contained in each image. This will include removing the clutter surrounding each target, and may involve clustering algorithms to preserve important areas of data.

I also need to finish my implementation of the multilayer perceptron, specifically the application of the back-propagation algorithm to optimise the network.