University Of Leicester

Mathematics Department

Postgraduate Student

Methods For Large-Scale Data Analysis & Machine Learning For Intelligent Image Processing

Author:
Stephen L Green

Supervisor:
Prof Ivan Tyukin

December 2019

Acknowledgements

I first wish to thank Prof Ivan Tyukin and Prof Alexander Gorban for supervising me for the duration of this project. I also wish to thank Robert Sanders, Richard Burton and Skye Rosetti for their assistance with some of the programming and understanding of the MatConvNet package and other queries along with Apical/ARM Ltd. for providing me with the equipment and software for the practical aspects of my project. Finally I'd like to thank my parents Andrew and Christine Green for their continued support during this PhD project.

Contents

Part I

Abstract

This project originates from research on methods and techniques for real time analysis and handling of large dimensional data with respect to the presence of uncertainties and scarce rates of sampling. The main applicational focus is the development of methods of object detection in the basis of higher dimensional representation of objects in their relative feature spaces. During these studies, it became clear that due to internal uncertainties and biases in the small amount of data available for this task, a theory for improving performance of generic AI systems regarding the minimisation of misclassifications is required for this project. Recently discovered phenomenon in stochastic separation theorems [1] have offered a way to remedy issues with errors in generic AI systems, providing that decision variables in the AI systems are sufficiently high dimensional [2], [3], [4], [5]. At the time work on this thesis started, these results were limited to only a few sets of practically relevant distributions. The theoretical focus of this work was to develop existing methods by generalising the results to k-tuples in product measure distributions and to overcome the concentration limit found in a later work [6]. The main application areas of these theories include but are not limited to:

1. Computational complexity in real time processing of high dimensional data.
2. Insufficient richness or quality of data.
3. Computational complexity of training a detector in real time.
4. Ensuring invariance to different methods of detection.

This thesis assesses these problem and describes recent developments in theoretical and practical results over the last four years. Section II introduces machine learning and the main associated problems. An overview of classification techniques precedes formal definitions of the remaining issues as mathematical problems to be solved. Section III details examples of tactile language classification through hardware and software. Section IV presents the main theoretical body of the thesis, culminating in two methods of classifying sign language gestures through an automated error corrector appended to a core AI going from a 82.4% success to 100% success with enough error clusters and a general separability measure to distinguish higher dimensional representations of projected data through kernel feature maps. Finally, Section V concludes this thesis with future outlooks for research and possible applications for the acquired results.

Part II

Introduction

1 The Main Machine Learning Problems

"Machine Learning is the science of getting computers to learn and act like humans do, and improve their learning over time in autonomous fashion, by feeding them data and information in the form of observations and real-world interactions." The aggregation of five separate definitions of machine learning [7].
The real world applications of Machine Learning span the breadth of academic disciplines available. This desire is drawn from the merits that an automated framework for decision making can have to fields such as Psychology, Economics, Engineering etc. The foundations set by Mathematics have given a proven structure to the advancement of Machine Learning as an appropriate tool in this regard. The increasing requirements for Machine Learning are built on the foundations of linear and nonlinear classifiers and the further developments that recent advancements have provided. The problems these technologies face are numerous, however there are a number of common threads between approaches and fields that occur during development. This section will focus on three of these areas and the generalised version of the Machine Learning problem as shown in [8].

1.1 The Problem Of Pattern Recognition

The problem of pattern recognition stems from learning functions $f(x, \alpha)$, $\alpha \in \Lambda$ with parameter set Λ that match the output of an external supervisor obeying the conditional distribution function $P(y|x)$ for a set of random vectors x from unknown distribution $P(x)$. For the binary indicator functions with output $y = \{0, 1\}$, the loss functional is given as:

$$L(y, f(x, \alpha)) = \begin{cases} 0 & \text{if, } y = f(x, \alpha) \\ 1 & \text{if, } y \neq f(x, \alpha) \end{cases} . \tag{1}$$

The expected value of this loss is given by the risk function:

$$R(\alpha) = \int L(y, f(x, \alpha)) dP(x, y). \tag{2}$$

For a known set of l i.i.d observations $(x_1, y_1), \ldots, (x_l, y_l)$ created according to an unknown $P(x, y) = P(x)P(y|x)$. The goal of the problem is to minimise this loss function such that $y = f(x, \alpha)$ as often as possible.

This field concerns finding patterns in the input data and reducing misclassifications [9]. These can be derived through supervised learning, with pre-characterised classes for each component of the information [10]. Many popular methods such as Transfer Learning and Multi-Instance learning were formed to minimise this loss functional for areas including drug activity prediction, text classification and visual tracking [11]. This is in contrast to Unsupervised Learning, which draw inferences from datasets consisting of input data without labeled responses. Through iterative processes such as Deep Learning, these draw on commonalities in the data and react base on the presence or absence of features in each input to classify patterns. This technique has experienced use in fields where small distinctions drastically change what label the input is given such as recent results in Face Recognition [12].

1.2 The Problem Of Regression Estimation

For a real value y, $f(x, \alpha), \alpha \in \Lambda$ with parameter set Λ is the set of real functions with the regression function:

$$f(x, \alpha_0) = \int y dP(y|x), \tag{3}$$

where $f(x, \alpha_0)$ minimises the risk function $R(\alpha)$ over the class of functions $f(x, \alpha)$. If $f(x, \alpha) \in L_2$, then this regression function minimises the loss:

$$R(\alpha) = \int (y - f(x, \alpha))^2 dP(x, y), \tag{4}$$

where the problem minimises this function for some unknown probability measure $P(x, y)$ with a known dataset $(x_1, y_1), \ldots, (x_l, y_l)$. A $n + 1$ dimensional variable $z = (x, y) = (x_1, \ldots, x_n, y)$ (where $x_1, \ldots, x_n$ is i.i.d) defines the loss functional to be minimised:

$$R_{\text{emp}}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} (y_i - f(x, \alpha))^2. \tag{5}$$

This estimates the conditional expectation of the dependent variable y as a function of independent variables x. The desired approximation $E(y|x)$ for conditional probability $y|x$ (y when x is known), forecasts the output as required based on the known results. These techniques are often very efficient with little computational resources required [13]. These methods often require some form of pre-processing (often called *Feature Engineering*) where attributes with little contribution to the output y or those that are well correlated with other inputs are removed.

1.3 The Problem Of Density Estimation

For the set of densities $p(x, \alpha)$, $\alpha \in \Lambda$ with parameter set Λ, the optimal density estimation minimises the risk functional:

$$R(\alpha) = \int -\ln(p(x, \alpha))dP(x, y), \tag{6}$$

with some $P(x)$ for i.i.d data $x_1, \ldots, x_n$. Estimating the density functions from a series of functions $p(x, \alpha)$, the maximum likelihood function to approximate density as:

$$R_{emp}(\alpha) = -\frac{1}{l}\sum_{i=1}^{l} \ln p(x_i, \alpha). \tag{7}$$

Equation (7) finds the optimum probability density function $P(x)$ to map the inputs onto. Given some choice of features, a pattern can emerge from the existing outputs, creating a series of boundaries which all new points are expected to follow. Any points outside of these estimations can be treated as outliers, where if the loss function is small enough, optimum parameters can be established.

1.4 The Generalised Learning Problem And Empirical Risk Minimisation

Equations (2),(4) and (6) are applications of the general learning problem, common to all optimisation methods. This problem is the minimisation of the risk function:

$$R(\alpha) = \int Q(z, \alpha)dP(z). \tag{8}$$

For a set of functions $Q(z, \alpha), \alpha \in \Lambda$ with parameter set Λ, and a probability measure $P(z)$ with i.i.d components of empirical data $z_1, \ldots, z_l$ defined on the space Z for this expected risk functional. The empirical risk minimisation induction principle approximates the risk minimisation function $Q(z, \alpha_0)$ by the function $Q(z, \alpha_l)$, which minimises the empirical risk:

$$R_{emp}(\alpha) = \frac{1}{l}\sum_{i=1}^{l} Q(z, \alpha_i). \tag{9}$$

The end goal of the generalised learning problem then becomes how to best reduce this empirical risk minimisation induction principle (ERM principle). To achieve this, some classification methods will be considered.

2 Linear Classifiers

The basis of real time analysis for handling large dimensional data resides in the separation and precise labelling of a series of categories through both linear and nonlinear classification. This is the approach of Machine Learning. This problem is then primarily a question of the appropriate methods of classification, and it's application towards a group of classes along with the density estimation of the pre-processed data that needs to be assessed within the confines of the system [14].
The question then becomes how this approach is applied to higher dimensional spaces, and what adjustments need to be made if at all. This section discusses the main forms of Linear Classification and the common applications along with their strengths and weaknesses. Once Linear Classifiers have been established, this will be repeated for nonlinear forms and the results will then be examined in the context of higher dimensionality and what can be done to minimise the machine learning problem.

2.1 Fisher Discriminants

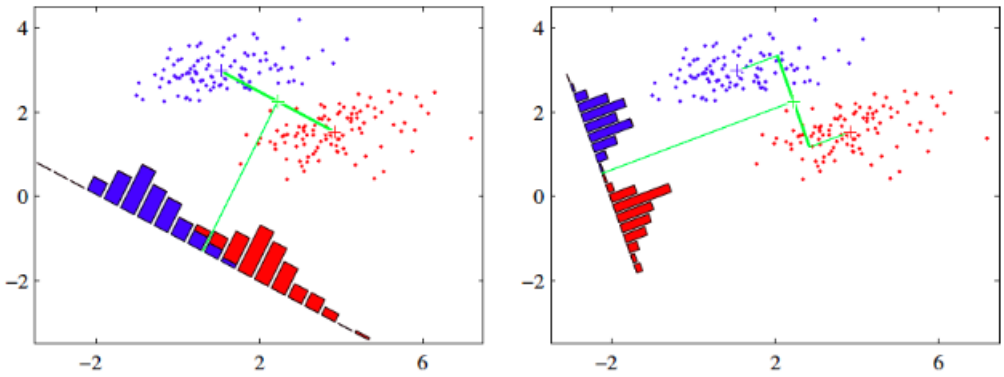


Figure 1: Regular histogram projection methods show a large class overlap from the line joining the respective class means on the left into the new projected space. Using fisher discrimination to find the point of maximum separation on the right shows a projection with greater class separation and a minimised class overlap for a 2D space best mapped to a 1D line. [15].

Fisher's linear discriminant (FLD) is a method of finding a linear set of features that categorises multiple classes of variables through a linear combination of features, discovered by Sir Ronald Fisher in 1936 [16] (This can also be generalised to Multiple Discriminant Analysis (MDA) for a non-binary set of classes). Traditional classification problems transform existing data with the aim of reducing dimensions which can be further projected into a new feature space and then be reclassified. This process has many implementations, and the optimal approach can vary depending on the original input's dimensionality. The

original dimensional space D has x objects with k classes. This is projected onto a new independent $K-1$ dimensional space consisting of weight transformations w_k on the original objects x forming the new feature set:

$$y_k = w_k^T x,\ k = 1, 2, \ldots, K-1. \tag{10}$$

The total matrix y of objects in the new $K-1$ dimensional space is achieved by a transformation through the total weight vector W:

$$y = W^T x, \tag{11}$$

Equation (11) finds the point of maximum separation that minimises class overlap, giving the largest ratio between the class scatter S_C between k classes and the intra-class scatter S_I of N objects in a given class k.
The class scatter S_C for a number of datums N_k in a class $\mathcal{C}_k$ for the original input space $x_n \in \mathbb{R}^D$ is given by:

$$S_C = \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} (x_n - m_k)(x_n - m_k)^T,\ m_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} x_n. \tag{12}$$

The intra-class scatter S_I for a total number of datums $N = \sum_k N_k$ for the original input space $x_n \in \mathbb{R}^D$ is given by:

$$S_I = \sum_{k=1}^{K} N_k (m_k - m)(m_k - m)^T,\ m = \frac{1}{N} \sum_{k=1}^{K} N_k m_k. \tag{13}$$

The data will be projected into the feature space J of rank $K-1$ such that each resulting matrix is independent. Using the weight vector W, for $y_n \in \mathbb{R}^{K-1}$, the projection function J of the weight matrix W can be inferred where the weight values for W are given by the eigenvalues of $\frac{S'_C}{S'_I}$:

$$J(W) = Tr\left(\frac{S'_C}{S'_I}\right). \tag{14}$$

The new class scatter $S_{C'}$ in the new input space $y_n \in \mathbb{R}^{K-1}$ is given by:

$$S_{C'} = \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} (y_n - \mu_k)(y_n - \mu_k)^T,\ \mu_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} y_n. \tag{15}$$

The intra-class scatter $S_{I'}$ in the new input space $y_n \in \mathbb{R}^{K-1}$ is given by:

$$S_{I'} = \sum_{k=1}^{K} N_k(\mu_k - \mu)(\mu_k - \mu)^T, \; \mu = \frac{1}{N}\sum_{k=1}^{K} N_k \mu_k. \tag{16}$$

As a result of measure concentration phenomena, simple FLD's can solve a subclass of higher dimensional classification problems even with large data sets with lots of dimensions through these techniques. This forms a part of the *Blessing Of Dimensionality* [17] [18]. These can also have the converse problem, where lower dimensional approaches that classify elements will perform poorly in higher dimensions where the sparsity of newly introduced dimensions lowers the performance rate of the classifier. This forms a part of the *Curse Of Dimensionality* [19],[20],[21], and navigating these two properties will be a core component of later sections.

2.2 Perceptrons

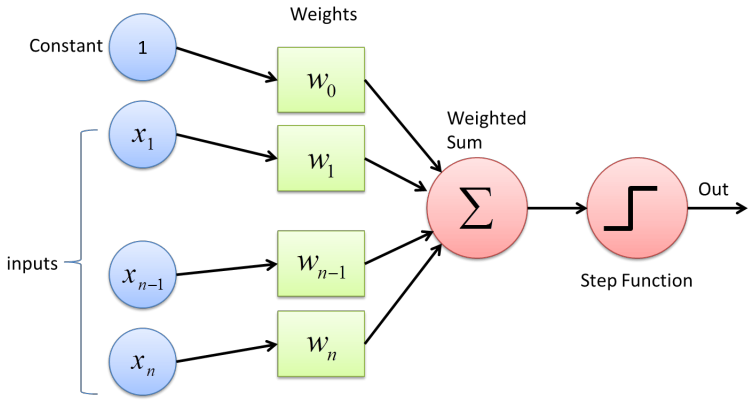


Figure 2: An illustration of a perceptron taking an input vector x and constant bias 1 and producing a binary output after the activation function.

The Perceptron was the first iterative algorithm proposed by Frank Rosenblatt in 1956 [22], and while the limited potential of the learning machines have been utilised to the fullest [23], it laid the ground work for successive algorithmic classifiers [24]. All perceptrons are formed with four major components: A set of input values x, a predefined weight vector w and bias b, a net sum calculation operation and a final activation function that output's the binary value y. The perceptron initially takes the input values $x_1, \ldots, x_n$ and performs a feature space mapping $\phi(x)$, then multiplies them by the weight vectors $w_1, \ldots, w_n$. This also includes a bias value of $\phi(x_0) = 1$ with a weight w_0. In these situations, the weights control the strength of a given node in the system, and the bias allows the activation function to be shifted up or down as appropriate. The resulting values

are then summed together and an activation function f classifies the result depending on whether the threshold is met:

$$y(x) = \begin{cases} +1, & w^T\phi(x) \geq 0 \\ -1 & w^T\phi(x) < 0 \end{cases}. \tag{17}$$

The target values are defined as $t = 1$ for class $\mathcal{C}_1$ and $t = -1$ for class $\mathcal{C}_2$. This gives the ideal function $w^T\phi(x_n) \geq 0$ for elements in class $\mathcal{C}_1$ and $w^T\phi(x_n) < 0$ for elements in class $\mathcal{C}_2$. For a label $t_n \in \{-1, +1\}$ assigned to each of the n elements, the goal of the perceptron is therefore to maximise the number of successful classifications where:

$$w^T\phi(x_n)t_n > 0, \tag{18}$$

and to minimise the number of elements in the set of misclassifications $\mathcal{M}$ given by the *perception criterion* function:

$$E_p(w) = -\sum_{n\in\mathcal{M}} w^T\phi_n t_n. \tag{19}$$

The algorithm runs iteratively. For each iteration, if the pattern x_n is properly classified the weights are unchanged. If not, then $\phi(x_n)$ is added for $\mathcal{C}_1$ classes and subtracted for $\mathcal{C}_2$ classes. For each iteration step i, the quantity of $E_p(w)$ will be reduced (since $||\phi_n t_n||^2 > 0$ [25]):

$$-(w^{(i+1)})^T\phi_n t_n = -(w^i)^T\phi_n t_n - (\phi_n t_n)^T\phi_n t_n < -(w^i)^T\phi_n t_n. \tag{20}$$

There is a common property that's explicitly present in all binary classifiers that defines the number of partitions a classifier can form for a given data set. This property is called the *Vapnik-Chervonenkis Dimension* (VC) [26] and is demonstrated in Figure 3.

Definition 1 *(Vapnik-Chervonenkis Dimension) A set C can be defined as part of a series of sets H such that:*

$$H \cap C = \{h \cap C | h \in H\}. \tag{21}$$

This set C is shattered by H if $H \cap C$ contains all subsets of C such that:

$$|H \cap C| = 2^{|C|}. \tag{22}$$

The VC dimension of H is the largest integer D where set C with cardinality D is shattered by H (such that $|C| = D$) [27].

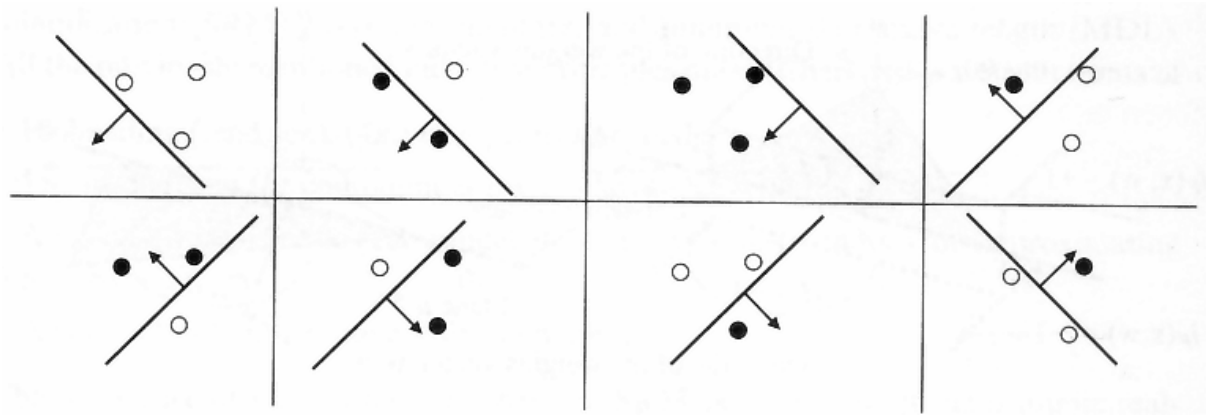


Figure 3: Linear classifiers have a VC dimension of 3 as the hyperplane shatters all 2^3 combinations of 3 points however some 4 point configurations can't be separated [28].

This enables an estimate for the upper bound on the test error for a classification model:

$$P\left(\text{test error} \leq \text{training error} + \sqrt{\frac{1}{N}\left[D\left(\log\left(\frac{2N}{D}\right)+1\right)-\log\left(\frac{\eta}{4}\right)\right]}\right) = 1-\eta. \quad (23)$$

$0 \leq \eta \leq 1$ is the learning rate and N is the size of the whole training set where $D \ll N$ [26], otherwise overfitting can occur such that the training error is lower than the testing error. These bounds can be used explicitly to justify the use of linear stochastic separation theorems, as the VC dimension explicitly defines the smallest dimensional space that the separating hyperplane can successfully partition data in.

2.3 Support Vector Machines

Given labelled training data, Support Vector Machines produce an optimal hyperplane that categorises new examples using supervised learning [29],[30],[31]. This extends the idea of perceptrons, which finds a separating hyperplane dependent of predefined weights and bias's initially chosen before iterating [32]. Support Vector Machines utilise the smallest boundary between the hyperplane and any of the elements (referred to as a *margin*). This margin is given by the perpendicular distance between the closest point x_n to the hyperplane [33]. By optimising the weights w and the bias b, the maximum margin is found by solving:

$$\underset{w,b}{\text{argmax}}\left\{\frac{1}{||w||}\min_n(t_n(w^T\phi(x_n)+b))\right\}. \quad (24)$$

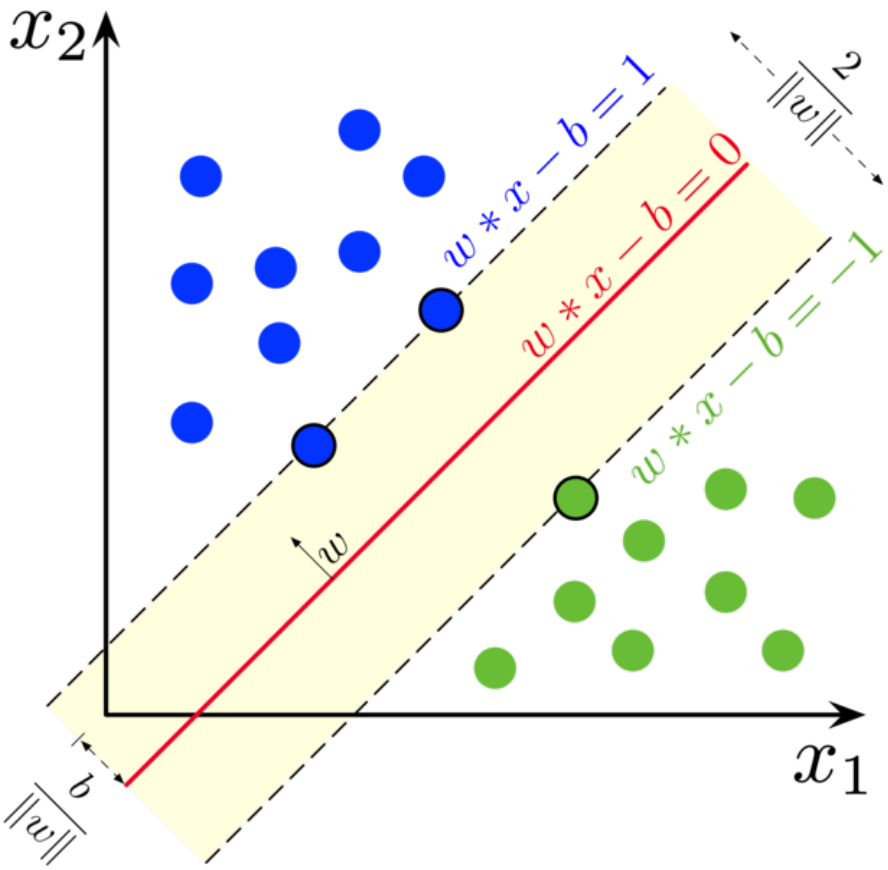


Figure 4: A hyperplane separates two classes while a maximum margin is defined for the points with the smallest perpendicular distance to the hyperplane. This distance is given by the value $||w||$ and minimising $||w||^{-1}$

Re-scaling the weights and bias's, all data points can be made to satisfy the constraint:

$$t_n(w^T\phi(x_n) + b) \geq 1, n = 1, \ldots, N, \tag{25}$$

where the closest point to the hyperplane always equals 1. For values of x_n where this holds, the constraint is *active*, otherwise the constraint is *inactive* [34]. Since at least one point has to equal 1, this will always be the minimum point such that the problem is now maximising $\frac{1}{||w||}$ [35] (See Figure 4 for a demonstration.)
Using Lagrangian multipliers $a_n \geq 0$ [36] for each constraint where $a = (a_1, \ldots, a_n)^T$, the primal form of the Lagrangian is formed:

$$L(w, b, a) = \frac{1}{2}||w||^2 - \sum_{n=1}^{N} a_n\{t_n(w^T\phi(x_n) + b) - 1\}. \tag{26}$$

Setting the initial weight and bias vectors to 0 each, the conditions for this become:

$$w = \sum_{n=1}^{N} a_n t_n \phi(x_n), \tag{27}$$

$$0 = \sum_{n=1}^{N} a_n t_n. \tag{28}$$

Taking the Lagrangian with respect to a gives the dual representation to be maximised:

$$\tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m \kappa(x_n, x_m), \tag{29}$$

subject to the constraints:

$$a_n \geq 0, \tag{30}$$

$$\sum_{n=1}^{N} a_n t_n = 0. \tag{31}$$

This applies for values $n = 1, \ldots, N$ and the kernel function $\kappa(x, x') = \phi(x)^T \phi(x')$ which will be covered in Section 3.1.

2.4 Reflective Summary

The relative simplicity of linear classifiers are appealing to problems that are inherently classifiable. The relationship between the predictive variables and the outcomes can be tracked and there's always some quantifiable contribution. This makes the approach compatible with lots of programming languages, with native packages for the aforementioned variants included in versions of R, Python, Java, Javascript and other languages, with little familiarity required. Linear classifiers are often utilised when the computational runtime needs to be minimised, with only a series of summations of weight transformations requiring calculations. This technique performs well even in higher dimensional space however the data should be regularised before processing to avoid overfitting [37]. The biggest limitation of these classifiers is that they cannot solve nonlinearly separable problems (for example, linear classifiers can separate OR & AND functions but not XOR functions). Nonlinear problems make up the majority of reasons why modern AI is in demand however, it is possible to transform the original input into a feature space allowing better representation. This is the basis for kernel functions.

3 Nonlinear Classifiers

While the limited potential of linear learning machines have been utilised to the fullest, as highlighted in the 1969's book Perceptrons by Marvin Minsky and Seymour Papert [23], real world applications often require more expressive hypothesis spaces than the ones provided by linear functions. In these cases, the main concept can't be simplified to a linear combination of parameters and instead uses more abstract features. While multi-layered functionals were suggested as a way of make existing classifiers more susceptible to new data, an approach was needed to form nonlinear classifiers without resorting to multilayered methods. These classifiers involve mapping the set of datums x to elements of a feature space $\phi(x)$ through some transition function ϕ. This approach is best seen in how Kernels are represented in Reproducing Kernel Hilbert Spaces as seen in this section.

3.1 Kernels

3.1.1 Reproducing Kernel Hilbert Spaces

$\mathcal{X}$ is defined as the input space along with a set of natural numbers $\mathbb{N}_n = \{1, 2, \ldots, n\}$. The function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{C}$ is a *kernel* on $\mathcal{X}$ if for any given finite $x = \{x_j : j \in \mathbb{N}_n\}$, there exists a Hermitian, positive, semi-definite matrix:

$$\kappa_{\mathcal{X}} = \kappa(x_i, x_j) : i, j \in \mathbb{N}_n. \tag{32}$$

Reproducing Kernel Hilbert Spaces (RKHS) have their point evaluations as continuous linear functionals [38]. This regularization behaviour is equivalent to the functional representation for $\{c_j : j \in \mathbb{N}\} \subseteq \mathbb{C}$:

$$f = \sum_{j \in \mathbb{N}_n} c_j \kappa(\cdot, x_j). \tag{33}$$

While for many applications kernels in Hilbert spaces are usually infinite dimensional [39], this can express most optimisation problems with RKHS regularizers as kernel expansions of the training data as their solution [40].

3.1.2 Nonparametric Representer Theorem

Theorem 1 *[40] For a non-empty set $\mathcal{X}$, a positive definite real-valued kernel κ on $\mathcal{X} \times \mathcal{X}$, a training sample $(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \mathbb{R}$, a strictly monotonically increasing real-valued function g on $[0, \infty[$, an arbitrary cost function $c : (\mathcal{X} \times \mathbb{R}^2)^m \to R \cup \{\infty\}$*

and a class of functions:

$$\mathcal{F} = \left\{ f \in \mathbb{R}^X \middle| f(\cdot) = \sum_{i=1}^{\infty} \beta_i \kappa(\cdot, z_i), \beta_i \in \mathbb{R}, z_i \in \mathcal{X}, ||f|| < \infty \right\}. \tag{34}$$

$|| \cdot ||$ *is the norm in the RKHS* H_κ *associated with* k*, i.e. for any* $\mathcal{X}, \beta_i \in R (i \in \mathbb{N})$*,*

$$\left|\left| \sum_{j \in \mathbb{N}_n} \beta_j \kappa(\cdot, x_j) \right|\right|^2 = \sum_{j \in \mathbb{N}_n} \sum_{i \in \mathbb{N}_n} \beta_i \beta_j \kappa(x_i, x_j). \tag{35}$$

Then any $f \in \mathcal{F}$ *minimising the regularised risk functional*

$$c((x_1, y_1, f(x_1)), \ldots, (x_m, y_m, f(x_m))) + g(||f||), \tag{36}$$

admits a representation of the form:

$$f(\cdot) = \sum_{i=1}^{m} a_i \kappa(\cdot, x_i). \tag{37}$$

A decision boundary from the class of functions with infinite sums will always minimise the regular risk function, plus the empirical risk function over finite sums of the original data points through the expansion of kernels. [40] reviews the traits exhibited by the Representer Theorem showing that for a class of functions κ, the respective class $\mathcal{F}$ can approximate any continuous function on a compact domain [41]. This leads into the Universal Approximation Properties.

3.1.3 Universal Approximation Properties

For a fixed, but arbitrary compact subset of $\mathcal{X}$ denoted by Z, $C(Z)$ is the space of all continuous complex-valued functions with maximum norm $|| \cdot ||$. The set $\kappa(z)$ of all functions of $C(Z)$ is defined as:

$$\kappa(Z) = \overline{span}\{\kappa_y : y \in Z\}. \tag{38}$$

For $\kappa_y : \mathcal{X} \to \mathbb{C}$ where $\kappa_y(x) = \kappa(x, y)$. This leads to the *universal approximating property* where $||f - g|| < \epsilon$ for $f \in C(Z)$ & $g \in \kappa(Z)$. This is the property of a *universal kernel* where $\kappa(Z)$ is *dense* in $C(Z)$ (such that $\kappa(Z) = C(Z)$). These properties apply to a broader class of nonlinear classifiers, most notably in feed forward neural networks [42].

3.2 Neural Networks

3.2.1 Network Architecture

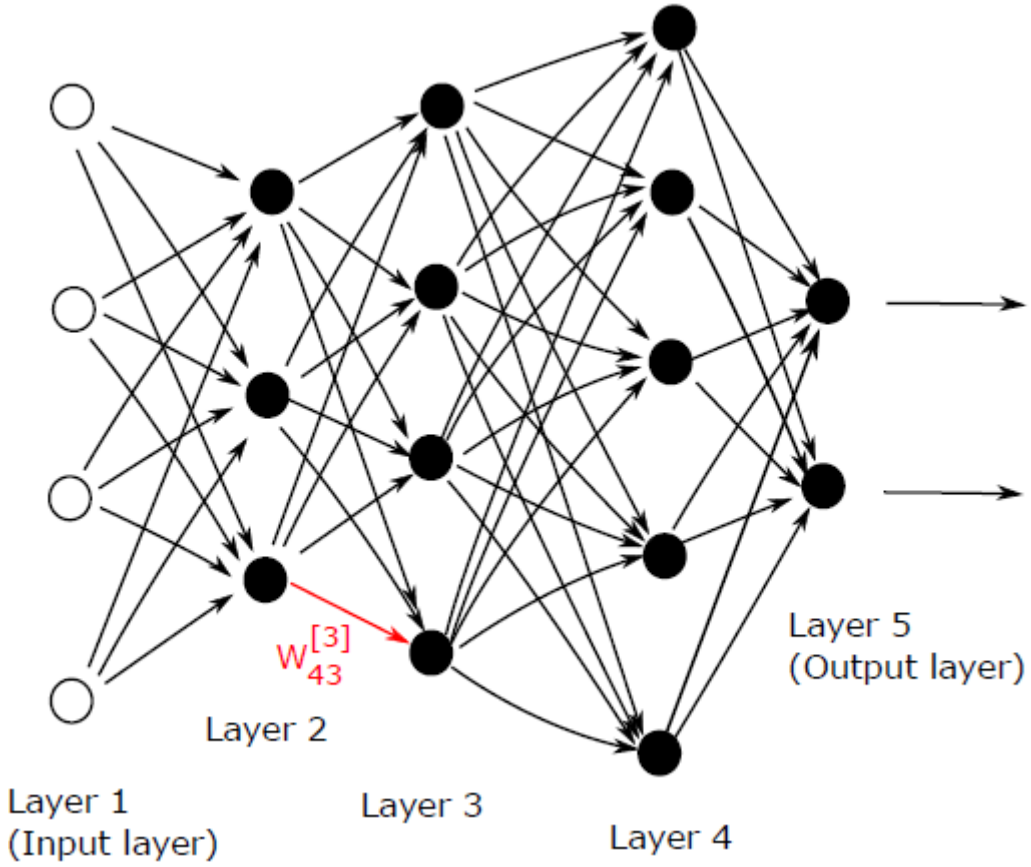


Figure 5: A network with three hidden layers. The output from the third neuron in layer 2 is weighted by factor $w_{43}^{[3]}$ when going into the fourth neuron in layer 3 [43].

Neural Networks extend the principles for creating perceptrons into a nonlinear scale. Rather than running iteratively, neural network consist of layers; starting from an input layer, going through a series of *hidden layers* before exiting as the product of a final output layer [44]. A network with L layers has n_l neurons for layer $l = 1, \ldots, L$. $w_{jk}^{[l]}$ denotes the weight that neuron j at layer l applies to neuron k at layer $l-1$, where $W^{[l]} \in \mathbb{R}^{n_l - n_{l-1}}$ denotes the matrix of weights at layer l ($b_j^{[l]}$ is the bias of neuron j in layer l which form the bias vector $b^{[l]} \in \mathbb{R}^{n_l}$). For an input $x \in \mathbb{R}^{n_l}$, the output is defined as $a_j^{[l]}$ for neuron j and layer l (this is called the *activation* in hidden layers). The *feed forward* process is defined as:

$$a^{[1]} = x \in \mathbb{R}^{n_1}, \tag{39}$$

$$a^{[l]} = \sigma(W^{[l]} a^{[l-1]} + b^{[l]}) \in \mathbb{R}^{n_l}, l = 2, 3, \ldots, L. \tag{40}$$

N training points are inputs $\{x^{\{i\}}\}_{i=1}^{N} \subset \mathbb{R}^{n_1}$ with target outputs $\{y(x^{\{i\}})\}_{i=1}^{N} \subset \mathbb{R}^{n_L}$.

Defining with the L_2 form:

$$C_{x^{\{i\}}} = \frac{1}{2}||y(x^{\{i\}}) - a^{[L]}(x^{\{i\}})||_2^2, \tag{41}$$

the goal is then to minimise the difference between the predicted values and actual values through the *quadratic cost function* of all weights and biases:

$$\text{Cost } = \frac{1}{N}\sum_{i=1}^{N} C_{x^{\{i\}}}. \tag{42}$$

3.2.2 Universal Approximation Bounds

Consider the space of functions f on a bounded set B (a unit ball for example). Let Γ_B be the set of functions f defined in B and let $\Gamma_{B,C}$ be a set of functions from Γ_B which satisfies (10) in [45]. For each $C > 0$, Theorem 2 captures both the universal approximating properties of neural networks and their advantages over kernel classifiers through a linear combination of functions such as the *sigmodial functions* $f(x) = \frac{e^x}{e^x+1}$. This allows a boundary to be defined for the integrated square error for approximation.

Theorem 2 *[45] For every function f in $\Gamma_{B,C}$ and every probability measure μ, there exists a linear combination of sigmoidal functions $f_n(x), n \geq 1$ which can be defined as:*

$$\int_B (f(x) - f_n(x))^2 \mu(dx) \leq \frac{(2C)^2}{n}. \tag{43}$$

This sets a boundary for the approximation error in functions by Artificial Neural Networks. This theorem assumes that these sigmoidal functions are not fixed but are chosen. However, for a fixed set of functions (the set of feature maps in kernel approximation for example), the convergence rate is not as good as the rate provided in Theorem 2 as this becomes dependent on the dimensionality of the domain. [45] specifies the lower bound on the approximation performance in this case.

For the uniform probability distribution μ of unit cube $B = [0,1]^n$:

$$d(f,g) = \left(\int_{[0,1]^d} (f(x) - g(x))^2 dx\right)^{\frac{1}{2}}, \tag{44}$$

represents the distance between two functions in $L_2(\mu, B)$. For a function f and a series of basis functions $h_1, h_2, \ldots, h_n$ (where $H_n = \text{span}\{h_1, h_2, \ldots, h_n\}$), the error in the approximation of f by the best linear combination of basis functions is given as:

$$d(f, H_n) = d(f, \text{span}\{h_1, h_2, \ldots, h_n\}). \tag{45}$$

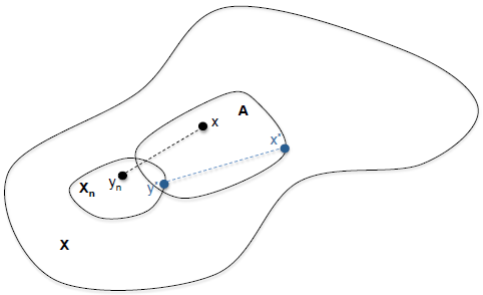


Figure 6: The approximation error defined in (46) of X_n from a set $A \subset X$ is defined in the figure as the distance between $x^* \in A$ & $y^* \in X_n$. This gives a measure of how well X_n approximates the worst point in $x \in A$. The Kolmogorov n-width is the smallest distance given by any subspace $X_n \in X$ [46].

The approximation error for the worst case functions in $\Gamma_C = \Gamma_{B,C}$ is given by:

$$\sup_{f \in \Gamma_c} d(f, H_n). \tag{46}$$

To characterise worst case approximations for all bases in Γ_C, the Kolmogorov n-width is a lower bound of all worst case distances (46) (Figure 6) over all sets of basis functions:

$$W_n = \inf_{dim(H_n)=n} \sup_{f \in \Gamma_c} d(f, H_n) = \inf_{dim(H_n)=n} \sup_{f \in \Gamma_c} \inf_{h \in H_n} \left(\int_{[0,1]^d} (f(x) - h(x))^2 dx \right)^{\frac{1}{2}}. \tag{47}$$

Theorem 3 *[45] For every choice of fixed based functions, a universal constant* $\kappa > \frac{1}{8\pi e^{\pi-1}}$ *and a fixed constant* $C > 0$ *with a domain dimensionality* d*, the Kolmogorov n-width of functions in* Γ_C *is bounded by:*

$$W_n \geq \kappa \frac{C}{d} \left(\frac{1}{n} \right)^d. \tag{48}$$

Theorem 3 reveals the worst case scenario for the basis. With m functions, there is some function not in the space of useful functions. This distance is bigger than the above boundary. With a fixed basis, the worst function will approximate the performance rate. If the parameters are changed, then a different rate of convergence emerges which is independent of the dimensionality of the domain (unlike back-propagation which aims to change the basis). A nonlinear approach to optimisation is required, as current problems with biased data, overfitting, and labelling noise can be reduced if these boundaries are taken into higher dimensional space. In these cases the error converges to 0 with a rate of $\frac{1}{n}$ in L_2, as the constant C will be dependent on the domain. Resolving the worst

case means further action can be taken to reduce the cost function as much as possible through gradient descent and back propagation in these later sections. It's unknown whether these kernels converge to a given minimum, as these decision boundaries are prone to errors (which will be the focus on the rest of this sections).

3.2.3 Stochastic Gradient Method

To reduce the quadratic cost function as much as possible, an iterative approach emerges using *gradient descent* to converge to the most optimal, cost efficient vector [47]. For the vector of all weights and biases $p \in \mathbb{R}^s$ where s is the total number of neurons in the neural network ($s = 23$ for Figure 5), the vector p updates with rule:

$$p \to p - \eta \nabla \text{Cost(p)}. \tag{49}$$

Cost(p) reparametrises the original function for the new dependency Cost : $\mathbb{R}^s \to \mathbb{R}$ with a small learning rule η and the gradient for the sum of partial derivatives (recalling (41)):

$$\nabla \text{Cost(p)} = \frac{1}{N} \sum_{i=1}^{N} \nabla C_{x\{i\}}(p). \tag{50}$$

Replacing the mean of individual gradient with a single random training point is less computationally expensive for large datasets where repeatedly calculating gradient vectors becomes inefficient [48]. This leads to a *stochastic gradient method*:

1. Choose a random i from $\{1, 2, \ldots, N\}$,
2. Update: $p \to p - \eta \nabla C_{x\{i\}}(p)$.

3.2.4 Back Propagation

The stochastic gradient method can train a neural network through calculating the partial derivatives of the cost function with respect to the weights and bias's of each neuron [49]. Initialising a single training point C gives:

$$C = \frac{1}{2} ||y - a^{[L]}||_2^2. \tag{51}$$

The weighted input $z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]} \in \mathbb{R}^l$ propagates information through the learning step $a^{[l]} = \sigma(z^{[l]})$ with an error for the jth neuron in layer l:

$$\delta_j^{[l]} = \frac{\partial C}{\partial z_j^{[l]}} \text{ for } 1 \leq j \leq n_l \ \& \ 2 \leq l \leq L. \tag{52}$$

The Hadamard product $(x \circ y)_i = x_i y_i$ derives a new learning rule:

$$\begin{aligned} \delta^{[L]} &= \sigma'(z^{[L]}) \circ (a^L - y), \\ \delta^{[l]} &= \sigma'(z^{[l]}) \circ \sigma'(W^{[l+1]})^T \delta^{[l+1]}, \\ \frac{\partial C}{\partial b_j^{[l]}} &= \delta_j^{[l]}, \\ \frac{\partial C}{\partial w_{jk}^{[l]}} &= \delta_j^{[l]} a_k^{[l-1]}. \end{aligned} \tag{53}$$

A forward pass calculates $a^{[1]}, a^{[2]}, \ldots, a^{[L]}$ sequentially then $a^{[L]}$ gives $\delta^{[L]}$ which derives $\delta^{[l-1]}, \delta^{[l-2]} \ldots, \delta^2$. This calculates partial derivatives through *back propagation* [50].

3.3 Deep Learning

Practical experience shows building multi layered networks can be advantageous over shallow networks. *Deep Learning* has determined parameters of multi layered structures since 1986 [51], while shallow networks require many more connections to replicate performance levels [52] (for example, ten times the number of connections are required in speech recognition [53]). This is computationally expensive, as floating point operations are performed with each made connection [54],[55]. Lower level computations are called *subroutines*, which push information forward until the final layer creates function ϕ [56].

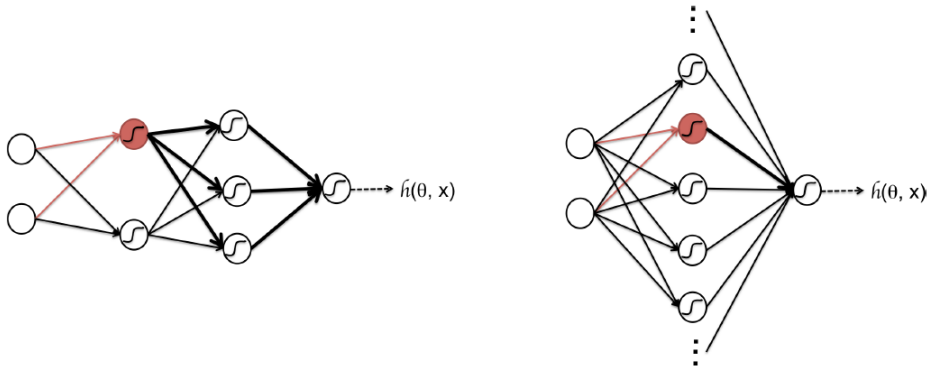


Figure 7: The *Deep Network* on the left reuses computation done by the red node 3 times to the second output layer, the *Shallow Network* on the right uses its output once.

Using a shallow network is equivalent to a program with no subroutines. With no subroutines, the code processing the shallow networks needs to be written explicitly, making the total length of code for shallow networks longer than deep networks giving a less efficient running time as computational costs are not reused [57].

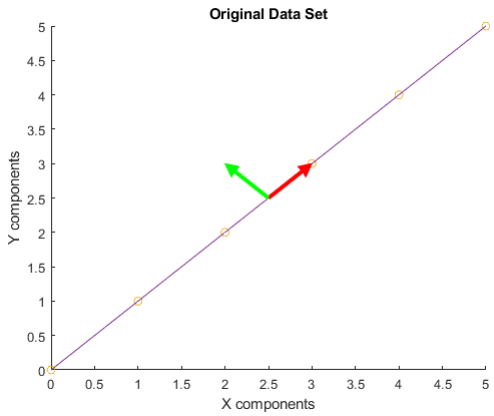
3.4 Pre-Processing and Dimensionality Reduction

3.4.1 Pre-Processing

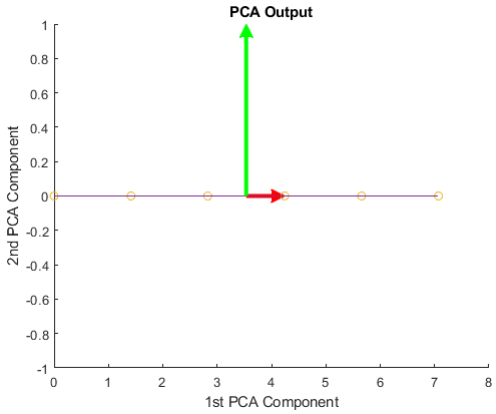
Before any data is sent into an algorithm, it often goes through some form of pre-processing, as the derived information directly affects the model's ability to learn [58]. These techniques can be as simple as accounting for missing values in an existing dataset, to partitioning the data into training sets, test sets and validation sets if required. Once the data is in some numeric form (relabelling categoric data if necessary), there can then be some form of feature scaling done before being transformed by the algorithm. This can include subtracting the mean from the whole set in a *centering* process. It can also then involve dividing by the standard deviation, *standardising* the data (such that the mean of the data is 0 and the standard deviation is 1 [59]). Alternatively, the original data can be *normalised* where the minimum value is taken from each datum and the whole set is divided by the range. Once the data has been cleaned, the data can go through a regularisation process such as PCA or ICA.

3.4.2 Principal Component Analysis

Developed by Karl Pearson in 1901 [60], Principal Component Analysis (PCA) is a orthogonal basis transformation that diagonalises a covariance matrix such that its eigenvectors are ranked by how large their respective variances are.
Orthogonal projections onto the eigenvectors are called *principal components* [61]. In PCA, the new set of dimensions are linearly independent and ranked according to the variance of data present in each component, decorrelating the original covariance matrix. As more components are retained, more internal information is kept, however this information is less necessary as more components are added [62]. For regularisation, it's common to pick a decision rule that underlines how many components are kept. Common examples include taking the first k dimensions, choosing k dimensions whose variance is greater than a set value, or by using the Kaiser-Guttman test, which keeps all components whose eigenvalues are above the mean eigenvalue for the full covariance matrix.
This geometric projection defines the goal of how to find the best summary of the original data by using a lower number of principal components through the aforementioned rules. This differs from the linear regression [63] approach, as PCA minimises the perpendicular distance from a data point and a principal component, while linear regression is the distance between the actual value of the output and what the line of best fit predicts the final value [64]. This approach gives the most accurate data representation in lower dimensional space in the directions of maximum variance.



(a) Before PCA, this data exists in both the X and Y dimensions



(b) After PCA all of the variance is contained within the first principal component, such that the second can be dropped entirely.

Figure 8: Using PCA can rank dimensions by how much variance each dimension has in comparison to the rest of the components. In this example, after PCA the second component can be dropped entirely as a two dimensional data set becomes one dimensional.

3.4.3 Independent Component Analysis

The problem with PCA is that decorrelation is not the only factor for ensuring independence. This is a desired outcome as for two independent variables, the product's expectation will be 0. By this metric, the task is to achieve full independence for each element x so the inner product for each kernel is 0: $\kappa(x, z) = \langle \phi(x) \cdot \phi(z) \rangle = 0 \forall x, z \in X$. To make sure all components are independent, it's possible to use Independent Component Analysis (ICA) to ensure statistical independence.

ICA is a statistical and computational technique that reveals the hidden factors that define sets of random variables [65]. The major differences between ICA and PCA is that all the components remaining after post-processing are of equal weight, with no hierarchy between vectors. The resulting components are also invariant to the sign of the sources where a white letter on a black background is the same as a white letter on a black background [66]. The theoretical foundations of ICA for a randomly observed vector $X = [X_1, X_2, \ldots, X_m]^T$ are given as the product of an $m \times m$ mixing matrix that randomises the position of elements and a random vector $S = [S_1, S_2, \ldots, S_m]^T$ given by $X = AS$. The goal of ICA is to find the unmixing matrix $W = A^{-1}$ such that the best approximation Y of S is given: $Y = WX \cong S$. For ICA to be successful, five conditions must be met [66]:

1. All sources creating a source matrix S must be statistically independent.

2. The mixing matrix A must be square and full rank.

3. The only probability distribution should be contained in S with no external noise.

4. Data must be centered before further processing where the mean of the data is zero.

5. Only one of the source signals can have a Gaussian probability density function.

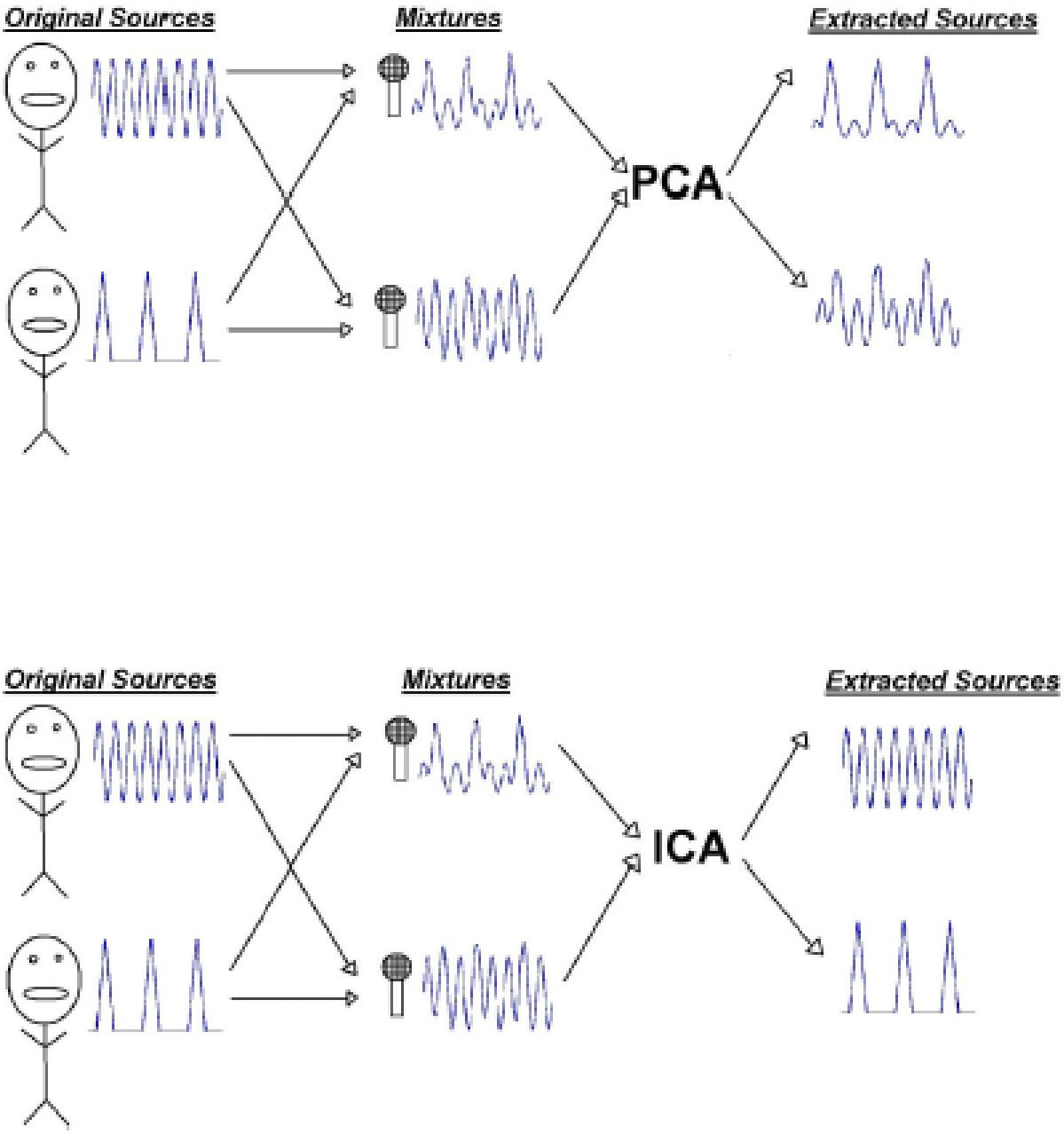


Figure 9: The diagram in the paper [66] refers to a 'cocktail party effect' where two people are speaking over each other producing mixed signals. The goal of PCA is to increase the clarity of the mixed signals so they can be audible to the rest of the party. The goal of the ICA is to retrieve the two original mutually independent sources from the vailable mixed signals so each person's voice can be heard clearly.

3.5 Reflective Summary

Considering kernels as shallow networks with huge hidden layers, comparisons between the two resemble the advantages and disadvantages of shallow and deep networks respectively. For example, large collections of neurons act like databases which resemble nonlinear functions. Kernel methods can avoid the large computations present in similarly large shallow neural networks through use of the *kernel trick*. This involves writing the optimisation process so that the hidden layers appear in the dot product of another example $< \phi(x), \phi(x') >$ which can be easily calculated.

In practice, kernel methods require support vectors surrounding the decision function (representing a separating hyperplane) that grows as more data is added, increasing the computational cost for ϕ for large datasets. Neural Networks don't have this issue as the computational cost is determined by how many connections the neural networks have and they do not rely on the size of the input data. This is especially true in networks where not every connection is necessary as applied in Convolutional Neural Networks.

ICA picks independent components without considering their importance unlike PCA. PCA maps onto projections with the highest variability such that when the first few components are picked, they are projected into the most relevant directions of the problem. ICA projects on dimensions which may not be relevant to the data as the actual dimensionality isn't featured in ICA.

While ICA can be used as an alternative to PCA, ICA doesn't do dimensionality reduction by default. In this respect, if just ICA is performed then because of no dimensionality reduction, the correctors would pick up noise in the data and the components will not generalise well. It's possible to run PCA and then ICA however in the majority of cases the computational complexity isn't practical unless the number of dimensions is well optimised.

4 Considering Issues In Machine Learning

4.1 Problems Overview

Artificial Intelligence (AI) systems have risen dramatically from being the subject of niche academic and practical interests to an accepted and widely-spread facet of modern life. Industrial giants such as Google, Amazon, and Microsoft offer a broad range of AI-based services, including intelligent image and sound processing and recognition.

Deep Learning and related computational technologies [67], [68] are perceived as state-of-the art tools for producing various AI systems. In visual recognition tasks, these systems deliver unprecedented accuracy [69] at reasonable computational costs [70]. Despite these advances, several fundamental challenges hinder further progress of these technologies.

All data-driven AI systems make mistakes, regardless of how well they are trained. Some examples of these have received global public attention [71], [72]. Mistakes may arise due to uncertainty in empirical data, data misrepresentation, and through imprecise or inaccurate training. Conventional approaches to reducing errors include altering training data and improving design procedures [73], [74], [75], [76], transfer learning [77], [78], [79] and privileged learning [80]. These approaches invoke extensive training procedures. The training itself, whilst eradicating errors, may introduce new errors by the nature of the steps involved (randomised sampling of mini-batches, randomised training sets etc).

A technology is proposed where errors of the original legacy AI are removed with high probability, through small cascading *neuronal ensembles* into the original AI's decision-making. Ensembles methods and classifiers' cascades [81], [82], [83], [84] constitute a well-known framework for improving classification performance. This shows geometry of high-dimensional spaces, in agreement with blessing of dimensionality [2], [17], [18] enables efficient construction of AI error correctors with guaranteed performance bounds.

This is similar to neurogenesis deep learning [85], classical cascade correlation [86], greedy approximation [45], and randomised methods for training neural networks [87], including deep stochastic configuration networks [88], [89]. This does not require extensive training or pre-conditioning, and can be set up as a non-iterative procedure. In this case, computational complexity scales linearly with the training set size in the best possible case.

This stems from concentration of measure phenomenon [90], [91], [92], [93], [94], and approaches AI learning from conventional probabilistic settings [95], [96], and stochastic separation theorems [1], [2], [6]. The building blocks of the algorithm are simple threshold, perceptron-type [97] classifiers. It can be shown, subject to mild assumptions on statistical properties of legacy AI signals, small cascading neuronal ensembles of linear classifiers are an efficient tool for learning away systematic errors.

4.2 Issue 1: Separation Of K-tuples For Product Measure Distribution

An AI system is an operator mapping elements of its input set, $\mathcal{U}$, to the set of outputs, $\mathcal{Q}$. Examples of inputs $\boldsymbol{u} \in \mathcal{U}$ are images, temporal or spatiotemporal signals, and the outputs $\boldsymbol{q} \in \mathcal{Q}$ correspond to labels, classes, or some quantitative characteristics of the inputs. Inputs $\boldsymbol{u}$, outputs $\boldsymbol{q}$, and internal variables $\boldsymbol{z} \in \mathcal{Z}$ of the system represent the system's state. The state itself may not be available for observation but some of its variables or relations may be accessed. Alternatively, it can be assumed there is a process which assigns an element of $\boldsymbol{x} \in \mathbb{R}^n$ to the triple $(\boldsymbol{u}, \boldsymbol{z}, \boldsymbol{q})$. A diagram illustrating the setup for a generic AI system is shown in Figure 10, which considers an add-on ensemble that maps samples $\boldsymbol{x}$ to auxiliary signals $\boldsymbol{s} \in \mathcal{S}$ (also known as *improvement signals*) in addition to the core AI. These improvement signals are used to improve the performance of the original AI. An example of such an improvement signal could be an indication that the current state of the core AI system represented by $\boldsymbol{x}$ corresponds to an erroneous decision. At the integration stage, this information will alter the overall response.
With regards to the operations f_i performed at the individual nodes (circles in the diagram on Figure 10), this will focus on the following relevant class:

$$f_i(\boldsymbol{x}) = f\left(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle - c_i\right), \tag{54}$$

where function $f : \mathbb{R} \to \mathbb{R}$ is piece-wise continuous, $\boldsymbol{w}_i \in \mathbb{R}^n$, and $c_i \in \mathbb{R}$. This class of functions is broad enough to enable the ensemble to function as a universal approximation device (e.g. choosing function f in the class satisfying conditions discussed in [45]) as well as to fit into a wide range of common AI architectures including decision trees, multilayer neural networks, perceptions, and deep learning convolutional neural networks.
Additional insight into "isolation" properties of elements (54) compared with more natural choices such as ellipsoids or balls (see [98]) is provided in Section 9.5. Over a relevant period of time, the AI system generates a large finite set of measurements $\boldsymbol{x}_i$. This is assessed by an external supervisor and partitioned into the union of sets $\mathcal{M}$ and $\mathcal{Y}$:

$$\mathcal{M} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\},\ \mathcal{Y} = \{\boldsymbol{x}_{M+1}, \ldots, \boldsymbol{x}_{M+k}\}. \tag{55}$$

The set $\mathcal{M}$ may contain measurements corresponding to expected operation of the AI, whereas elements from $\mathcal{Y}$ constitute core AI's performance singularities. These singularities may be both desired (related e.g. to "important" inputs $\boldsymbol{u}$) and undesired (related e.g. to errors of the AI). The function of the ensemble is to respond to these singularities selectively by producing improvement signals $\boldsymbol{s}$ in response to elements from the set $\mathcal{Y}$.

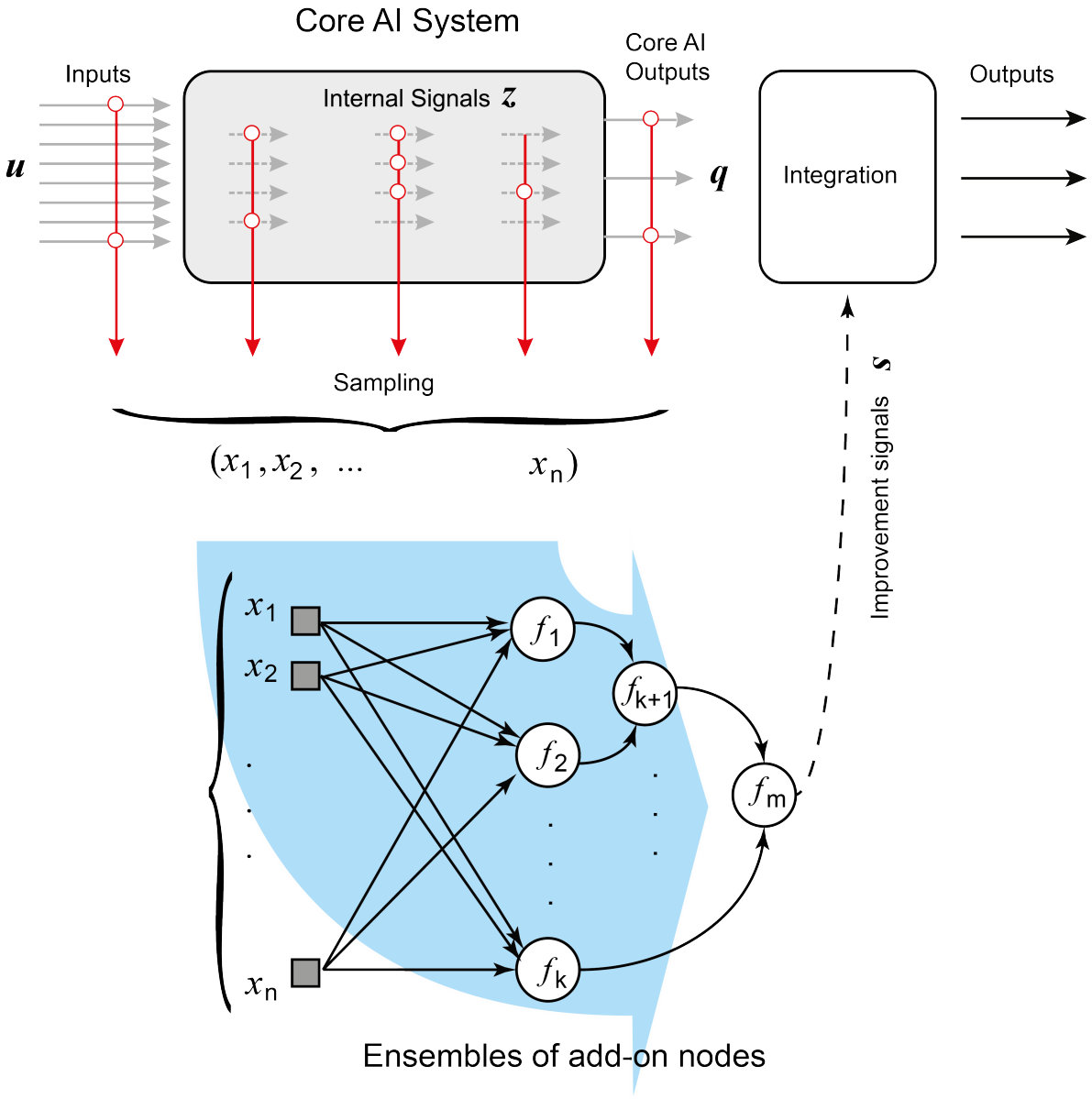


Figure 10: Functional diagram of correcting neuronal ensembles for a generic Artificial Intelligence system.

A straightforward approach to derive such ensembles is to use standard optimization facilities such as error back-propagation and its modifications, or other iterative procedures similar to greedy approximation [45], stochastic configuration networks [89] (see [85]). However, this may require excessive computational resources. Even in the simplest case of a single-element add-on system that separates the entire set $\mathcal{X}$ from $\mathcal{Y}$, determining the best possible solution, in the form of e.g. support vector machines [96], is non-trivial computationally. Theoretical worst-case estimates of computational complexity for determining parameters of the single-element system are of the order $O((M+k)^3)$ [100]. The question is if there exists a computationally efficient procedure for determining ensembles or add-on networks such that:

1) They "improve" performance of a generic AI system with guaranteed probability.

2) They consist of elements (54).

Preferably the computational complexity of the procedure is to be linear or even sublinear in $M+k$. Subject to some mild technical assumptions on the way the sets $\mathcal{M}$ and $\mathcal{Y}$ are generated, a family of simple algorithms with the required characteristics can be derived. These algorithms are motivated by Stochastic Separation Theorems [1], [2], [6]. For consistency, adapted version of these theorems are presented and discussed in Section 9.1.1. The algorithms themselves are presented and discussed in Section 9.1.2.

4.3 Issue 2: Nonlinear Stochastic Separation Theorems

Kernel classifiers have been recognised as a powerful tool for a range of classification problems [101], [102]. They offer an extension of linear classifiers to the nonlinear ones and the Representer Theorem [40] states that kernel classifiers minimise a range of risk functionals that can be expressed as kernel expansions over sample points. This allows an expansion of support vector machines [103] to the realm of kernel classifiers and offers a computationally efficient way to construct classifiers with nonlinear decision boundaries. Choosing a particular kernel for a given task is recognised as a hard theoretical and computational problem [104]. Several approaches try to address this, such as grid search algorithms [105], [106], automatic tuning of kernel parameters [107], genetic algorithms [108], and other heuristics [109]. These methods allow a selection of optimal feature spaces via explicit statistical evaluation of kernel classifiers over a family of kernel candidates. Instead of repeatedly solving a given classification problem with a given family of kernel classifiers directly, relevant statistical properties of kernels and their corresponding feature maps are investigated and assessed. This motivation stems from Cover's theorem [110], [111], suggesting that higher dimensionality of feature maps relative to that of the original

data may play a role in success of kernel classifiers, and other relevant bodies of work [1], [2], [5], [6], [18], [93], [112], [113], [114] on properties and geometry of high-dimensional spaces. Links between dimensionality and separability have been explored in the literature on statistical learning theory through e.g. the concept of the Vapnik-Chervonenkis (VC) dimension [115] measuring richness of classification rules which can be implemented by a classifier. By adapting stochastic separation theorems [4], [1], [3], [5], [6], [116] to kernel classifiers and providing their kernel generalizations, these results can derive computable separability measures for kernel classifiers, including given samples of empirical data (see [117] exploring the notion of the local Rademacher complexity). One of the outcomes of such generalization is an explicit characterization of kernel separability properties in terms of finite-dimensional volume integrals over domains determined by the kernel functions themselves. This suggests that even when kernel feature maps are infinite dimensional, separability properties of these maps can be expressed in terms of (finite) dimensionality of the space to which the original data belongs [99].

4.4 Issue 3: Overcoming The Concentration Limit

As more time and resources are dedicated to optimising current AI system technology, there's trepidation regarding it's full potential, with its failings always in the public eye. A survey polling 2473 AI-Incorporated companies reported that 25% of companies experience a failure rate of over 50% [118]. This is a widespread issue, contributing to a high demand for non-destructive corrective AI systems.

A cascading corrector for one shot methods is desirable. The approach in [2] flags new errors in each iteration preserving the legacy AI's structure without back propagation which can introduce new errors [4],[50]. This criteria requires binary classification that separates correct classifications from incorrect classifications [5]. A new decision rule is needed, which detects the current systems failings and forms a correct decision rule. The correction methods proposed in [6] have been applied to distributions of a reasonably broad class. This has to satisfy the constraint for probability distributions ρ in a ball $\mathbb{B}_n$:

$$\rho(y) < \frac{Cr^n}{V_n(\mathbb{B}_n)}, \tag{56}$$

for an arbitrary constant $C > 0$, the volume of the ball $V_n(\mathbb{B}_n)$ and a radius $r \in (0, 2)$. The probability of successful correction can be bounded by $1 - CM\left(\frac{r}{2}\right)^n$, where M is the number of points in the data sample. If $r > 2$, the boundary is impractical as $\left(\frac{r}{2}\right)^n$ grows exponentially such that this may become negative for a single point. The question is whether this boundary can extend further with probability distributions that satisfy this constraint with $r > 2$, and at what cost.

Part III

Object Recognition And Case Studies

5 Convolutional Neural Networks

Conventional network structures connect every neuron in the hidden layer to every neuron in the input layer. The computational complexity of conventional neural networks increases proportionally to the size of the starting image (For example, a 128x128x24 square with 24-bit colours will have 393,216 dimensions for each hidden neuron). Connecting each neuron to a portion of the available connections through a predefined structure of input neurons reduces this proportional relationship, such that computational complexity is more manageable at higher dimensions with this altered structure [119]. Forcing these connections in multiple layers creates a deep locally connected network.

Further parameter reduction is achieved through *weight sharing* where the parameters

(a) All neurons to hidden layer (b) Adjacent neurons to hidden layer

Figure 11: Examples of input neurons connecting to the first hidden layer of a network.

are made equivalent. This convolution process applies filters to many positions in the input signals, connecting to a max pooling layer that takes the maximum value of the output neurons and feeds the information forward to the next highest layer [120]. This approach means that these neurons are invariant to any shift in the input network.

This reduction creates the structure for the *Convolutional Neural Network* (CNN) with successful results in large scale image and video recognition challenges [121], [122], [123]. The application of translational invariance (See Figure 12) feeds forward to multiple filters attached to each input neuron, creating a structure of maps whose multi-channeled outputs are similar to images at the input neuron stage which concatenate colour channels to a single input. CNN's have gone through a series of improving iterations [124] with variations aiming to improve object recognition performance and developing increasingly computationally efficient technologies and these experiments will demonstrate how these processes can be implemented into standard image recognition practices [125].

(a) $x_1 = [0, 1, 0, 0, 0, 0, 0, \ldots]$ (b) $x_2 = [0, 0, 0, 1, 0, 0, 0, \ldots]$

Figure 12: 1D representations of a white dot. Figure 12a is transformed by w_2 and is two pixels away from Figure 12b, transformed by w_5. As the weights are equalised, the value after max pooling is unchanged. This property is called *Translational Invariance*.

6 Image Visualisation: An Example

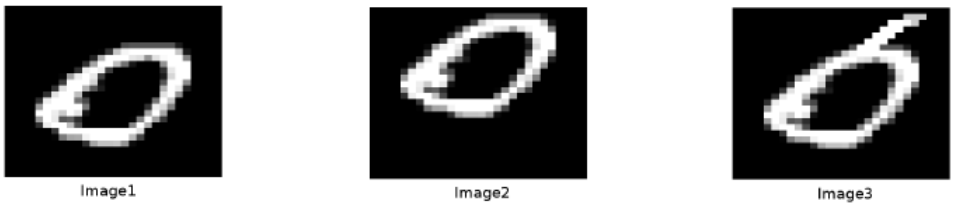


Figure 13: Three 28x28x1 images. The left and centre images represent the digit 0 and the right image is the digit 6

The aim is to correctly label the first two images as the image class "0" and the third image as the image class "6". All single channel grayscale images can be broken down into a grid of colour intensities between 0 (black) and 255 (white). This representation is concatenated into a single vector where small position changes to a non-zero digit greatly changes the composition of the new vector.

In this example, the difference between the two classes is an appendage protruding from the top of the shape forming the digit "6". The non-zero pixel intensities in this section are therefore much higher than the "0" counterpart. The euclidian distance between each image can then be taken using the formula:

$$d(a, b) = \sqrt{\sum_{i=1}^{28}\sum_{j=1}^{28}(a(i, j) - b(i, j))^2}. \tag{57}$$

The pixelwise distance d between the first and second 28x28x1 images is given to be about 1000 while the pixelwise distance d between the first and third images is about 3000. Each image is handwritten so intra-class variation must be taken into account when pixel boundaries are implemented. This can then be mapped with a non linear

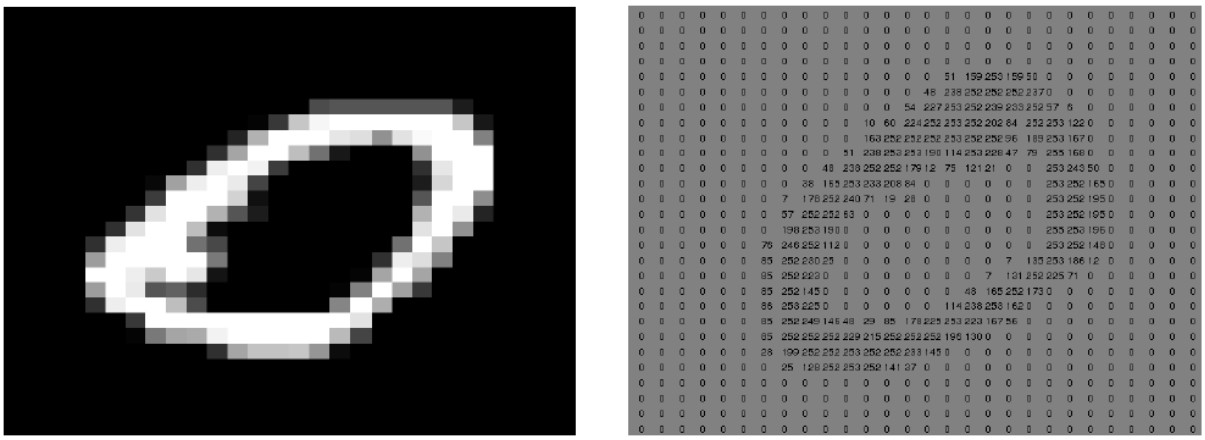


Figure 14: The representation of the number 0 in MNIST is translated into it's grayscale pixel value equivalent through the recognition software.

function to correctly identify the data [126]. Correct labelling is nearly impossible to achieve with linear decision functions however nonlinear functions can be employed for computationally complex problems such as dealing with natural data like speech and vision recognition [127].

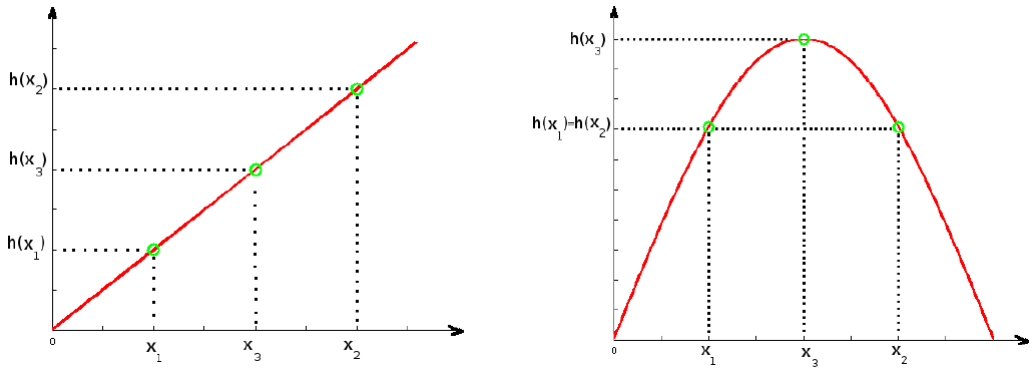
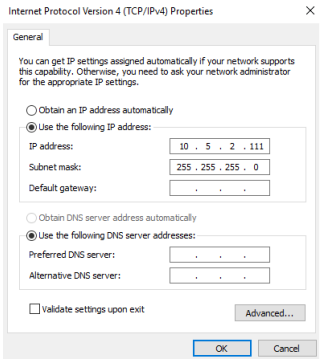


Figure 15: For a case where $x_1 = 2$, $x_2 = 6$, $x_3 = 4$ (so x_1 is closer to x_3 than x_2 the goal to have a decision that $h(x_1) = h(x_2) \neq h(x_3)$. The linear function of the left fails to capture the decision while the nonlinear function on the right does [124].

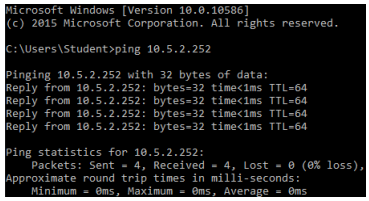
7 Case Study 1: Single Set Gesture Recognition

7.1 Overview

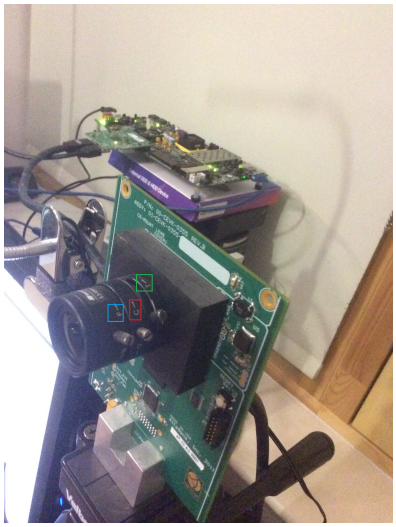
As the structure for vastly reducing the required parameters for images to be classified has been defined in Section 5, this case study provides a demonstration for how images can be categorised in real time. A hardware based approach to real time detection is explored. A dedicated ZYNQ-7000 signal processing board acts as an intermediary between a state of the art camera and an Operating System (OS). This section details the successful operation of the hardware and software necessary to recognise a single set of one handed gestures in contrast to a class of unrelated images based on research conducted by ARM LtD. [128]. Integrating techniques employed in Section ImageVisual, after tuning the lens the camera tracks gestures in real time. These images are captured and transformed into signals compared against a set of non-gesture images. A simple CNN detects the presence of gestures and the results are normalised for further distinction. A summary of the results is then considered. If the results are satisfactory for single gesture class recognition, more classes can be added for further detection. In the case that gesture recognition isn't possible, more techniques will be considered.



(a) Fixed properties for the Ethernet port to connect to the board (wifi can interfere with the process so it is turned off)



(b) The ping command sends data to the board and retrieves a result showing the board is active.



(c) The camera used for tracking data. The squares represent different tuning rings for picture quality.

Figure 16: How to set up and check that the board and camera are correctly connected to the computer.

7.2 Hardware Specifications And Required Software

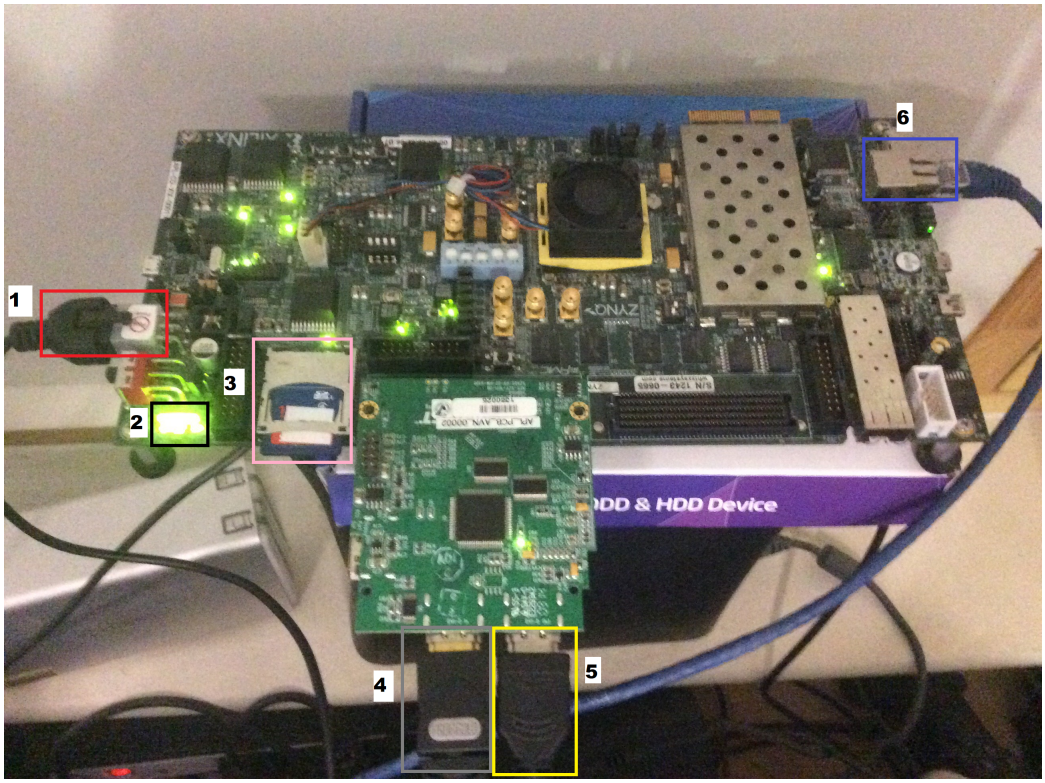
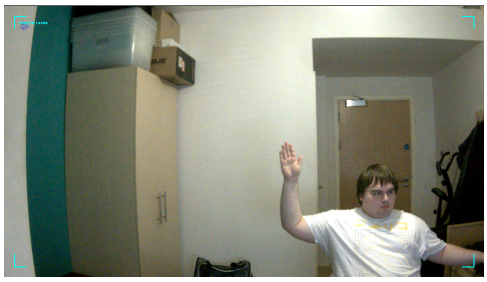


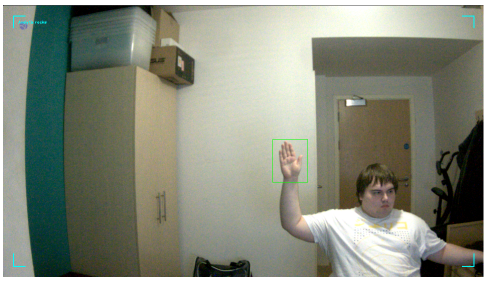
Figure 17: The red box (1) shows a 6 pin power supply through a transformer.
The black box (2) has a back row of 6 lights active when the board is and the front 4 lights show external connections. The centre left light flashes when the camera is active.
The pink box (3) holds a SD card which tells the camera to follow the procedural algorithm for tracking a given object in the camera's line of sight.
The gray box (4) shows the gold input cable from a HDMI port to a DVI port on the back of the camera which provides data to the SD card in the circuit board.
The gold box (5) houses the silver input cable which attaches to an Epiphan DVI2USB capture card that saves images and transfers them through a USB cable from the card.
The blue box (6) is a port housing an Ethernet cable that goes directly to the computer. This port flashes white when the camera is on.

After initial set up, the computer is configured to the board's IP address. In Windows 10, Control Panel ⇒ Network + Internet ⇒ View Network Connections brings up the Ethernet port. Going into Internet Protocol 4 and changing the IP to 10.5.2.111 (the same address as the board with subnet mask 255.255.255.0 and default gateway). This is checked by typing *ping 10.5.2.252* into command prompt. An additional circuit board has a HDMI cable that connects to a capture card via a DVI output to a USB 3.0 cable that captures images and saves them. VirtualDub is installed along with the capture card drivers, which checks what the camera records. By opening Veedub64, and clicking Capture AVI, the camera recording is visible for testing purposes.

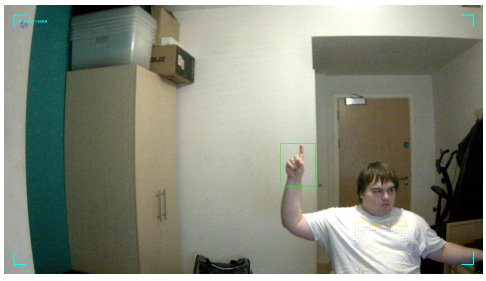
The camera has three variable settings to attain a clear image. The green rectangle in Figure 16c is the focus lens, which moves the focal plane closer to the camera's sensor. The red rectangle in Figure 16c controls the camera's aperture, dictating the light let into the lens and the blue rectangle in Figure 16c is the zoom lens which manipulates the angle of the lens, which can switch a panoramic view to a telephoto view.
The ThumbCollect folder contains the tracking demo (one for the internal webcam and one for the camera's connection to the circuit board). Participants sit about two metres away, towards the camera with their flat palm at arms length away from the body. The algorithm recognises the gesture and the software takes 250 images for one minute once the palm has been detected.
A green border will then appear around the subjects hand on screen. When this occurs, the participant's hand makes an index finger shape so the camera can record these new images so the neural network can be trained (this transition takes 10 frames so roughly 240 index finger gestures are in a single session). Multiple participants were used in these experiments (about 50 in total) for a different range of hands and altogether roughly 10,000 images were collected for the total set of gestures.



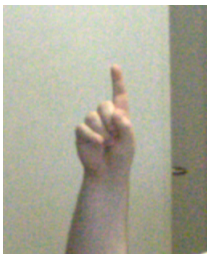
(a) The flat palm is shown to the camera as the camera searches for the gesture.



(b) The algorithm positively identifies the palm and begins tracking the gesture.



(c) The subject switches gestures and the camera follows the hand movement.



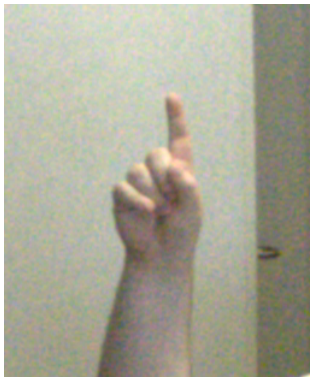
(d) An image of the captured index finger sent to the computer.

Figure 18: The camera captures a given gesture and how it sends it back to the OS.

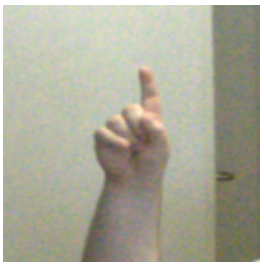
7.3 Image Pre-Processing

When all images are taken, they undergo pre-processing before they are sent to the neural network for training. The data capture process starts taking images the instance it recognises a given gesture in it's field of view. As the length of the gesture away from the camera isn't consistent, the size of the images varies between different sessions. Before the set of gestures is processed, the propriety image manipulation software "Xnview" is used to resize all images to a consistent height, width and format such that every image that goes into the CNN is a 128x128 24-bit bitmap square. When the image sizes are consistent, every image is copied and flipped so the set of 10,000 mostly right hands became a set of 20,000 hand gestures evenly split between both hands. A class of negative images was formed whose only common link was that they had no presence of gestures in each image. From over 300,000 images available, two sets were formed, with a 1:3 ratio of test to training images (a common ratio for Neural Networks) and a 1:9 ratio of gesture set to negative set ratio (mimicking later experiments with ten gestures):

1. A test set containing 5000 images of index fingers and 45,000 unrelated images.
2. A training set containing 15,000 images of index fingers and 135,000 unrelated images.



(a) The original unaltered picture attained from the camera hardware. This example is a 254x310 PNG image file however the data in the training and test sets are 24-bit bitmaps.



(b) The picture has now been resized to a 128x128 PNG. While the picture has been compressed and the ratio of height to width has been altered the key features remain so it can go through to processing.

Figure 19: An example of image transformation. This was done to all 20000 images collected for the neural network.

7.4 Gesture Structure

7.4.1 CNN_MNIST Architecture

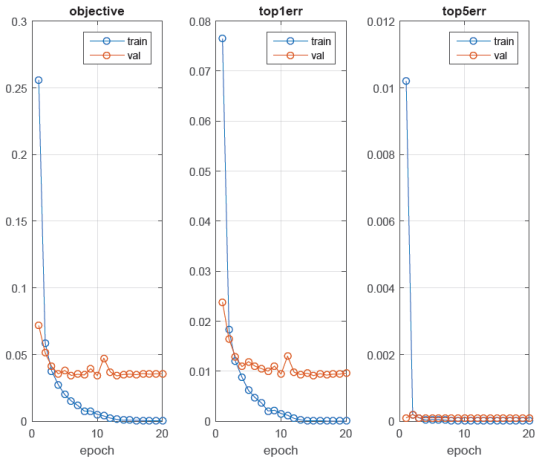
This algorithm is heavily based on the architecture in the *CNN_MNIST* algorithm from the MatConvNet MATLAB Toolbox [129],[130]. This is trained on the MNIST dataset [131], a series of handwritten numbers between 0 to 9, on 28x28 grayscale squares split evenly between a 10,000 image test set and a 60,000 image training set that's typically used as a benchmark that CNN's can be trained on, with the current record only mislabelling 21 images from the test set with an accuracy of 99.79% [132]. The convolutional neural network trains itself by running through twenty iterations (referred to in the system as *epochs*) on 100 randomly chosen digits labelled correctly. After training, a table with details on the eight layers of CNN_MNIST and the plot of epochs against the loss of the objective function and the chance of the correct label not being the highest prediction result or within the top five results for both the training and validation sets.

layer	0	1	2	3	4	5	6	7	8
type	input	conv	mpool	conv	mpool	conv	relu	conv	softmxl
name	n/a	layer1	layer2	layer3	layer4	layer5	layer6	layer7	layer8
support	n/a	5	2	5	2	4	1	1	1
filt dim	n/a	1	n/a	20	n/a	50	n/a	500	n/a
num filts	n/a	20	n/a	50	n/a	500	n/a	10	n/a
stride	n/a	1	2	1	2	1	1	1	1
pad	n/a	0	0	0	0	0	0	0	0
rf size	n/a	5	6	14	16	28	28	28	28
rf offset	n/a	3	3.5	7.5	8.5	14.5	14.5	14.5	14.5
rf stride	n/a	1	2	2	4	4	4	4	4
data size	28	24	12	8	4	1	1	1	1
data depth	1	20	20	50	50	500	500	10	1
data num	100	100	100	100	100	100	100	100	1
data mem	306KB	4MB	1MB	1MB	313KB	195KB	195KB	4KB	4B
param mem	n/a	2KB	0B	98KB	0B	2MB	0B	20KB	0B

parameter memory|2MB (4.3e+05 parameters)|
data memory|8MB (for batch size 100)|

cnn_train: resuming by loading epoch 20

(a) The output of the CNN_ MNIST function. The process is split into 8 sections consisting of convolution, max pooling and rectified linear units before the final softmax layer.



(b) After each epoch the error of the objective function is calculated along with the errors for the top 1 & top 5 prediction. With each epoch, the error rate exponentially decreases to a minimum.

Figure 20: The resulting outputs of the CNN_ MNIST MATLAB file. After completion the function can then analyse images and output their characterisation and the likelihood of matching with the result.

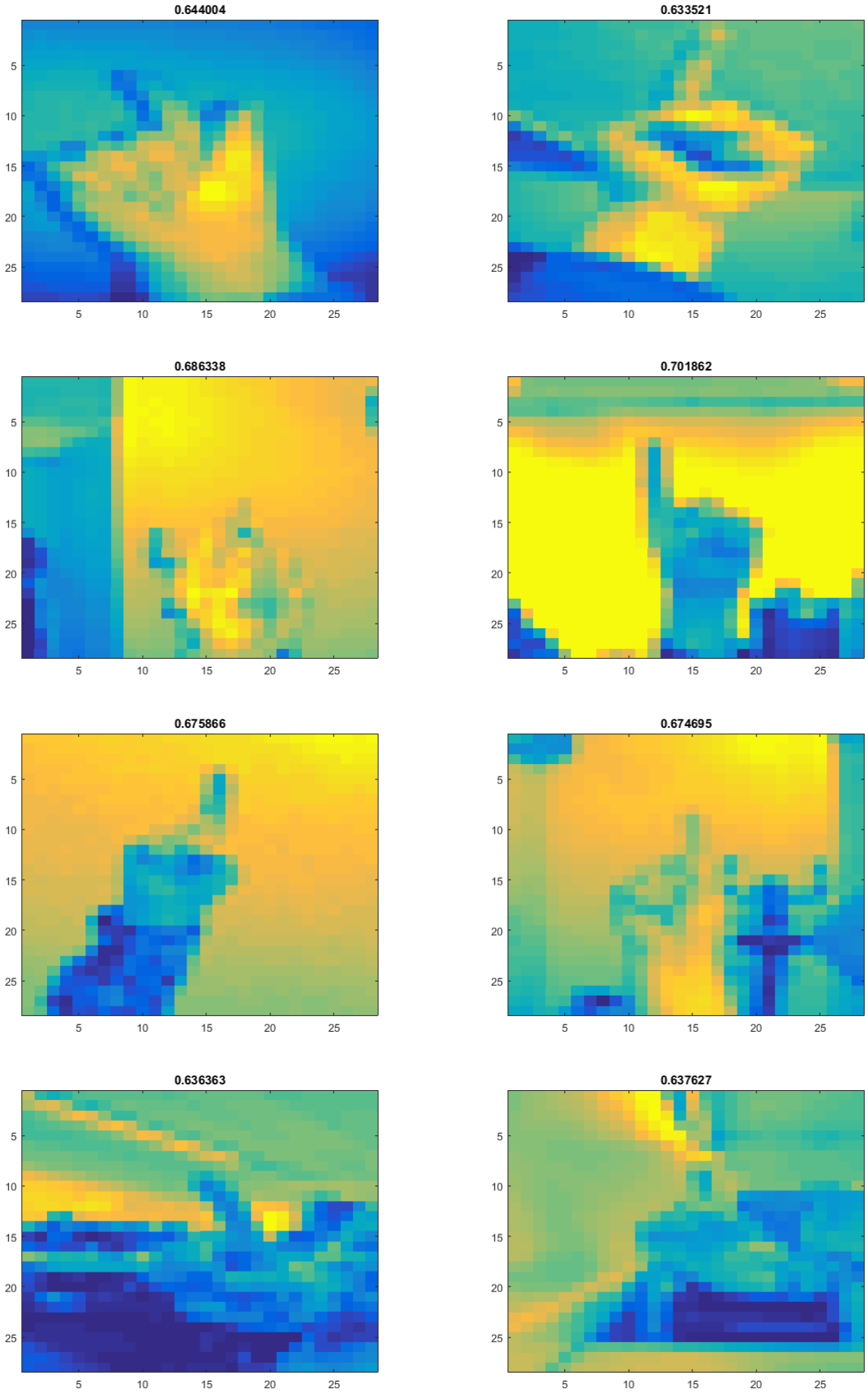


Figure 21: 8 index fingers filtered through MNIST before grayscaling (See Appendix). The score represents the classification result.

7.4.2 CNN Structure

Figure 20a gives the layer structure for the modified CNN_MNIST program. The algorithm first funnels the data into the input layer by resizing to 28x28 grayscale squares matching the format of the MNIST training set. The first convolutional layer connects the concatenated vector to a bank of 20 single precision $5 \times 5 \times 1$ filters to the second layer of hidden neurons, shrinking the square image further from 28x28 to 24x24 with relation:

$$f : \mathbb{R}^{M \times N \times K} \to \mathbb{R}^{M' \times N' \times K'}, \, x \mapsto y. \tag{58}$$

This convolutional layer precedes the first pooling layer of the algorithm. This halves the dimension sizes of the current layer through a selected feature operator. CNN_MNIST chooses the max pooling operator, moving a 2x2 square two pixels across the entire image (this translation is called the *stride*), taking the largest number to the next layer resulting in a 12x12x1 square (another common feature operator is Average Pooling, which takes all values captured in the stride and outputs the mean of them to the next layer). After repeating the process, the third convolutional layer is followed by the Rectified Linear Unit (ReLU) activation function:

$$f(x) = \max(0, \, x). \tag{59}$$

This is used to improve gradient propagation and gives more efficient computation [133]. After a final layer of convolution, the softmax layer classifies the image with the label that has the highest probability of matching the original image (Figure 21 gives the normalised images with their respective probability of being a gesture).

(a) Correct (b) Incorrect (c) Incorrect (d) Incorrect

Figure 22: CNN_MNIST will be trained to successful categorise the gesture as it's own class and the other objects as a non-gesture set. This non-gesture set include landscape scenery, random objects and objects mimicking gestures to train the program not to misclassify these false images.

7.5 Results

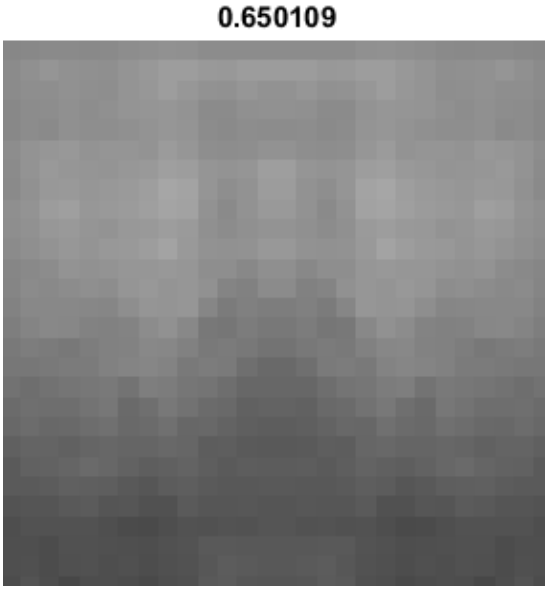
After training, a small set of fifty gestures and fifty negatives was formed, featuring fifty index finger gestures from fifty different participants and mixture of negatives as shown in Figure 22. The trained convolutional neural network predicted the presence of gestures with 65.01% certainty in the set of gestures and 63.92% certainty in the set of negatives (See Figure 23), with a 1.09% difference between classifications.
Following the preliminary results, the whole dataset of 20,000 gestures and 180,000 negative images was fed into the CNN_MNIST algorithm. This iteration gave 67.90% certainty for the gestures and 63.06% for the negatives with a difference of 3.84% between them, with the full range of results presented in the table below.

	Index	Negative	Total
Minimum	0.579207658767700	0.536803066730499	0.536803066730499
Maximum	0.729841709136963	0.733080506324768	0.733080506324768
Range	0.150634050369263	0.196277439594269	0.196277439594269
Mean	0.679025549340248	0.630671212495036	0.635506646179557

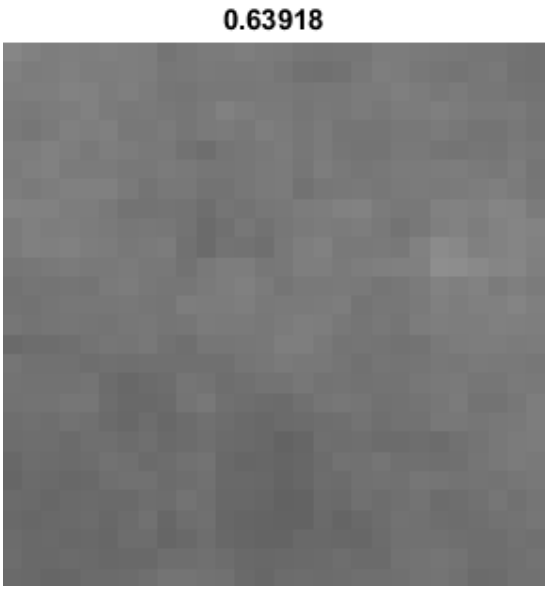
Table 1: Contrasting the classification scores of all 200,000 images

By taking 50% as the passing threshold, every image is classed as a gesture according to the smallest percentage of 53.68% in Table 1. Therefore based on this criteria, both results are very poor, with the small differences in intraclass gestures and large differences in intraclass negatives contributing little to distinguishing images. A pattern emerges in the gesture set (the average for each pixel value is plotted in Figures 23a and 23c) where both experiments have a notably darker section in the bottom third and centre of the image, with the other 5/9ths having a notably lighter tone. This occurrence is likely due to the central index finger gesture along with the participants shoulder being caught in the images (since each image is mirrored, there is a symmetry along the Y axis such that the darker section occurs regardless of whether the participant is left or right handed). In contrast, the negative set has no internal correlations and is an amalgamation of pixels that's more pronounced in the fullset version which is a nearly perfect gray square, with the range of pixel values being just 18 compared to the variation in the gestures, from 104 to 122.

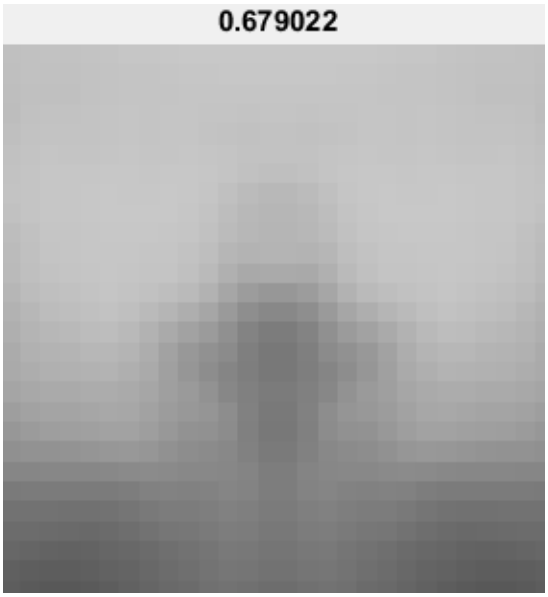
7.5.1 Average Results For Small And Large Datasets



(a) The average of 50 28x28 images constituting the Index Fingers in the small set as an image with the accompanying score.



(b) The average of 50 28x28 images constituting the Negatives in the small set as an image with the accompanying score.



(c) The average of 20,000 28x28 images constituting the Index Fingers in the full set as an image with the accompanying score.

(d) The average of 180,000 28x28 images constituting the Negatives in the full set as an image with the accompanying score.

Figure 23: The average values for each pixel in the 28x28 image sets were calculated. The result was divided by 255 and plotted in MATLAB where 0 is a white square and 1 is a black square. The gesture set has distinctive areas emerging in the bottom third and centre of the image represented by the central gesture and the camera catching the shoulders while the negative set is an indistinct gray shape.

7.5.2 Normalised Results

Total Populations	Predicted Condition Positive	Predicted Condition Negative
True Condition Positive	1808	18182 (Type II Error)
True Condition Negative	77393 (Type I Error)	102607

Table 2: Truth Table of normalised results.

To refine the results produced from the algorithm, the results are normalised. From the results in Table 2, subtracting the minimum and dividing by the range gives 200,000 values between 0 and 1. For the set of normalised results, if the new score is higher than 0.5 then the image is classed as a gesture and the image if classified as a Negative if the threshold is not met. With these results, a truth table is made showing the likelihood of Type I and Type II error and a *receiver operating characteristic* (ROC) curve is used to illustrate this. From the table below, the normalised accuracy is calculated as $\frac{\text{True Positive + True Negative}}{\text{Total Image Number}} = \frac{1808+102607}{200000} = 0.522125$, such that each image has a 52.21% chance of being classified correctly.

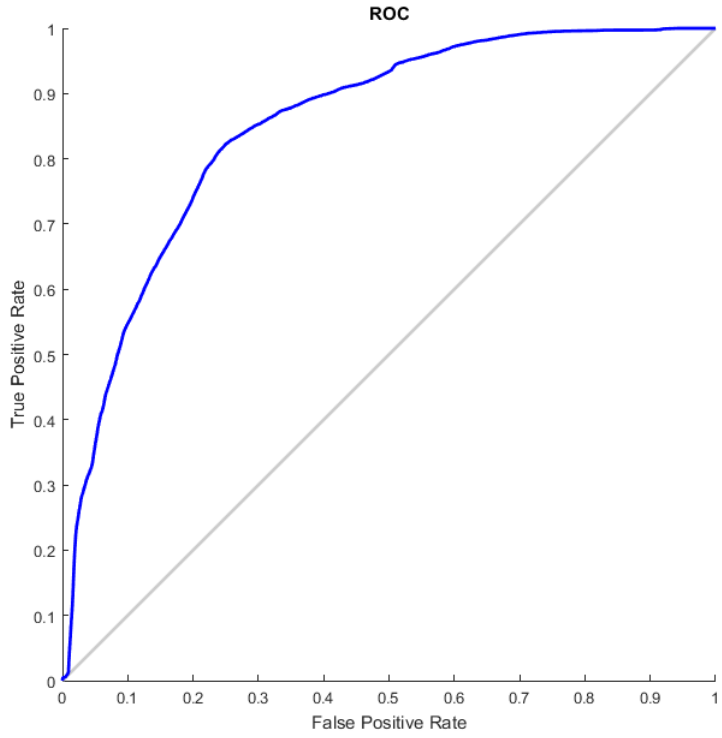


Figure 24: The graph occupies the upper left quadrant of the leading diagonal showing that there is a trend towards a positive classification when predicting data.

7.6 Single Gesture Summary

The normalised results show a system that categorises images better than generic random selection but not by a statistically significant measure. The 3.84% difference for pre-normalised scores is a large step down from the 99.2% success rate on MNIST. Most notably, this algorithm was designed exclusively to work on MNIST, and switching to a general images is not an optimal approach. Even cropped, there is background variation that CNN_MNIST accounts for which is magnified when shrunk to a small scale. This applies to colour channels as the algorithm grayscales images before further processing, so further detail is lost, contrasting with MNIST which only has white text on a black background with minimal intra-class overlap. Finally while the original intention sought the most fitting comparison of ten defined classes, these modifications only try and classify a set of unrelated images with an amalgamation covered by a single class.
Moving forward, a stronger CNN architecture will be chosen that's proven capable of handling multiple varied coloured classes at once, with proven results in image recognition challenges such as ILSVRC [134] and a larger set of related classes to give the recognition application a dedicated purpose. For this reason, a general purpose sign language detector was formed with one of the most popular algorithms for image recognition in an entirely software based process, moving away from the hardware restrictions of the image processor based properties in this experiment.

(a) F in Semaphore Sign Language

(b) F in American Sign Language

(c) F in British Sign Language

(d) F in Chinese Sign Language

Figure 25: Gestural applications that object detection can be trained to classify.

8 Case Study 2: Sign Language Classification

8.1 Overview

After the disappointing results obtained in Section 7, a stronger classifier is necessary. Rather than a dedicated processor which transfers images from a camera to an OS, this approach will solely focus on the classification of existing images. Rather than aiming to track motion in real time, which is the larger focus of the supplied hardware, the aim of this Section is to create a working CNN that's able to categorise multiple classes of images successfully in a computationally efficient manner.
For these tests, the recognition of Deafblind sign language will be tested. This is due to the language's high novelty and practical use if successful class partitions can be formed. The structures of popular CNN's will be contrasted with each other, with a focus on the variations of Inception and how advancements in AI architecture have evolved over time. Once the images have been taken, Inception will train on them, and a final accuracy score is produced. Patterns will be observed in any mislabelled images and some other applications will be considered, demonstrating the flexibility of the current state of CNN's.

8.2 Deafblindness

Deafblindness is a sensory disability resulting from hearing and vision loss, significantly affecting communication, socialisation, and daily living [135]. Studies report between 0.2% and 3.3% of people suffer from this including 36% of individuals over 85. Acquired Deafblindness happens over time [136], often caused by Usher's Syndrome [137], which develops in teenagers as a side effect of brain damage. Alternatively, Conditional Deafblindness is when a person is deafblind from birth, occuring through pregnancy complications or genetic conditions, often before communication is learned. [138]. Hearing aids, braille, professional interpreters, and recent developments such as PARLOMA have helped with direct communication [139] however these are cumbersome as professional translators are expensive and PARLOMA is still in trial stages.
Continuing with gesture recognition, further classification techniques are used on the Deaf Blind Sign Language alphabet, a variation on British, Austrian and New Zealand Sign Language (BANZSL) [135]. This has twenty-six unique classes for each letter performed by two people: The deaf-blind participant and an interpreter. The interpreter signs motions onto the other person's hand and these form complete words and sentences.
Translating sign language using neural networks has been around almost as long as CNN's [140], with much work done describing their functionalities and applications [141], [142], [143], with wearable technology that translates gestural movement into a series of letters

[144]. Through CNN's, each gesture is assigned a unique label for the network to correctly classify. Sign Language recognition often prioritises correct identification of singular letters in languages over whole words. This process called Fingerspelling is appealing as there's only 26 labels to classify over thousands of words in any given sign language dictionary such as the over 3000 word lexicon present in American Sign Language [145].
A technology that could convert BANZSL into text would be a huge leap forward for communication between deafblind people. While there have been strides advancing conventional sign language recognition, this form has very little research on it. This program uses a deep CNN that recognises respective gestures and produce it's textual equivalent.

Figure 26: Each deaf-blind gesture and their respective class labels from A to Z.

8.3 Methods

8.3.1 Pre-Processing

After the experimentation with the image processor, a new set of images is required. No database exists of cataloged image samples for the Deaf-Blind alphabet, so 26,000 colour images with multiple participants with varying age, gender and skin tone were taken with a Samsung Galaxy S10+'s front facing 10 megapixel camera (see Figure 26 for examples of each gesture). Each class contains 1000 images taken from all possible angles, distances and sizes including the signed gestures all at different locations to mitigate issues of background noise interfering with classification so the only consistent trait with inter-class images is the gesture itself.
These experiments will be conducted with a Convolutional Neural Network called Inception. Most naive CNN's have a common problem that the most relevant section of an image class have a large variation in size, sometime occupying a large portion of the image or in other cases being a minute feature of it. As an example, while other popular CNN's such as AlexNet and VGGNet have proved successful with repeating convolutional layers fixed to either large filters or grids of multiple smaller filters [146], they suffer from huge computational requirements which can compound when working with large datasets as will be necessary for this stage of the project. For example, VGGNet uses multiple 3x3 filters, at the pre-fully connected layer these require 512 as inputs and outputs giving 9x512x512 calculations in total, nearly two and a half million for each consecutive filter. This makes choosing the kernel sizes for convolutional operations in deep networks crucial, as larger kernels are recommended for globally distributed information and small kernels are recommended for locally distributed information [147], with these operations often being computationally expensive.

The first iteration of Inception (also knows as GoogLeNet) side-stepped this problem by having filters of multiple sizes ($1 \times 1, 3 \times 3, 5 \times 5$) operating on the same level, making a wider network as opposed to a deeper network. The results are the concatenated and sent to the next inception module until the final softmax layer classifies each resulting vector (this is the image's *bottleneck*). While deep neural networks are usually computationally expensive, Inception employs extra 1x1 convolutions that limit the amount of input channels and are less computationally expensive than the regular "naive" method. The second version of Inception aimed to reduce the number of *representational bottlenecks*. This contains a large loss of information through too many dimensions being reduced from the image's original state along with improving the factorisation method that deduced which filter path was the most optimal to decrease computational complexity [149]. As a 5×5 convolution is 2.78 times more computionally expensive than a

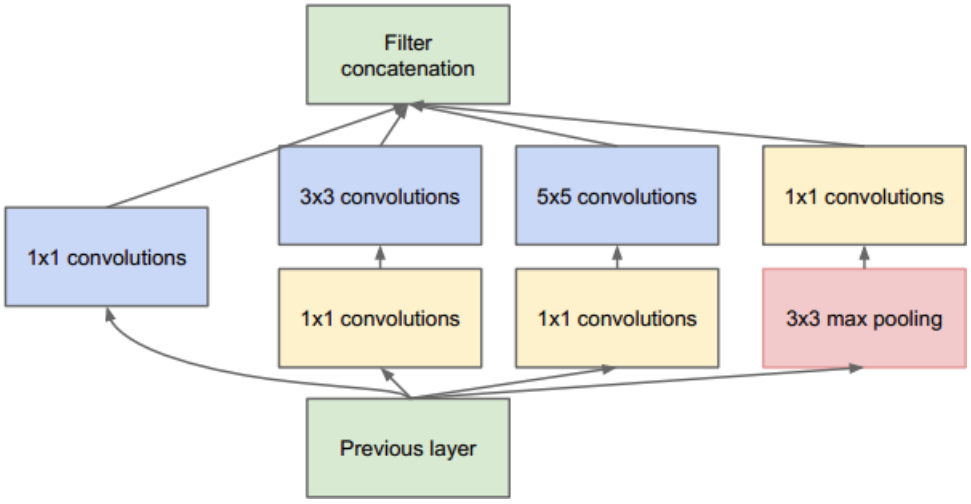


Figure 27: Cross section of the most common architectures of Inception [148].

3×3 variant, two 3×3 filters were stacked in place of the 5×5 section increasing the algorithms performance. This process was applied again dimension-wise, as factorising $n \times n$ filters to $1 \times n$ then $n \times 1$ also reduced complexity, with the 3×3 case being 33% cheaper than the process in Inception Version 1. Along with expanding the filter banks so that three variants of the regular inception module were utilised related to their place in the algorithm and this second version was faster computationally and has less errors.

Network	Top-5 Error	Top-1 Error
GoogLeNet (V1)	N/A	7.9%
BN-Inception (V2)	22.0%	5.8%
Inception-V3	18.8%	4.3%
Inception-ResNet-V1	18.8%	3.8%
Inception-V4	17.7%	3.8%
Inception-ResNet-V2	17.8%	3.7%

Table 3: Each variation of Inception along with the Top-1 and Top-5 from a single model with 144 cropped sections evaluated from the ILSVRC 2012 classification benchmarks (the variant used for future tests in bold).[149], [150].

The version used for this project is Inception Version 3. Released in 2015, this introduces minor changes from Inception Version 2 such as the introduction of 7x7 convolution layers in a $1 \times 7 \rightarrow 7 \times 1$ convolution pattern along with label smoothing which regularises losses that prevents overfitting. The results for Inception Version 3 are superior to similar networks of it's time. Since these experiments were performed, Inception Version 4 was released which streamlines the architecture with reduction blocks that are now explicitly defined, cleaning the structure leading to improved computational costs. There

are also two versions of Inception-ResNet, with similar costs to Inception Version 3 & 4 respectively however residual connections are introduced that add the output of the convolutional operation to the inception module for smaller error loss [150]. While these versions do work faster and produce better results. The results were given with the third version so there would be a better spread of label mis-matches to correct in Section IV.

8.3.2 Processing

Each image is resized to a 299x299x3 square regardless of shape where all images comprise of 268,203 values between 0 and 255 in this structure for each of the three RGB colour channels. Then the images go through five initial convolutional layers with interspersed max pooling layers before being filtered into groups based on correlation statistics of the previous layer [151], with the necessity for the larger filters increasing through each pass with a module variant with expanded filter banks in the last two blocks before the final concatenation and grid reductions taking place twice during this process. Inception runs through all 26,000 images with a learning rate of 0.01 and 15,000 training steps and a batch size of 100 images for the training, test and validation steps, such that 10% of this set is used for training and 10% is used for validation. This process creates a single 1x2048 vector for each image before being classification by the final softmax layer. This assigns a likelihood that the image corresponds to the given label and the highest likelihood is chosen as Inception's prediction. With 26 labels and N+2048N parameters where N is the number of labels, there are 53,274 parameters in total that Inception runs through in the softmax layer for a single image [152].

8.4 Results

Inception mislabelled 988 deaf-blind gestures, giving a final accuracy total of 96.2% with zero threshold after the final softmax layer. The error split presented in Figure 28 shows the ratio of the errors present, with most gestures falling within the baseline of three standard deviations excepting the gestures for M (and O marginally). Looking into the abnormal number of errors for M, of the 97 errors present in the 1000 pictures tested, 54 of these predicted N as the correct gesture, with an average difference of 0.17 between these predictions of N being the present gesture (incorrect) and M being the present gesture (correct). The difference between these two gestures is slight, as the M gesture is performed by putting the middle three fingers across the recipients palm and the N gesture is made by putting just the middle and index finger on the palm (See M & N in Figure 26). Especially from angles parallel to the participant's hand, some of the these can be very hard to distinguish from each other accounting for these large portion

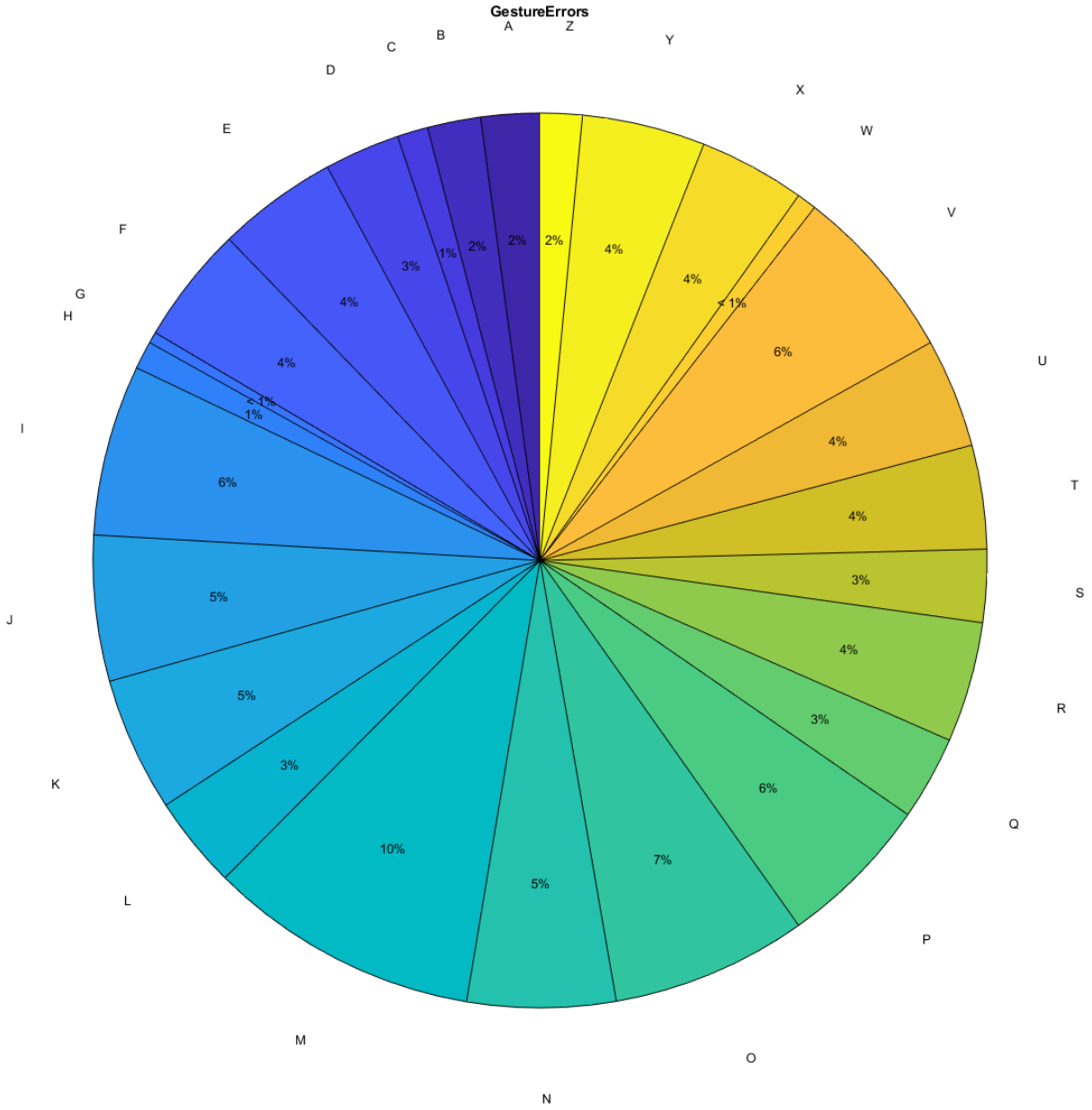
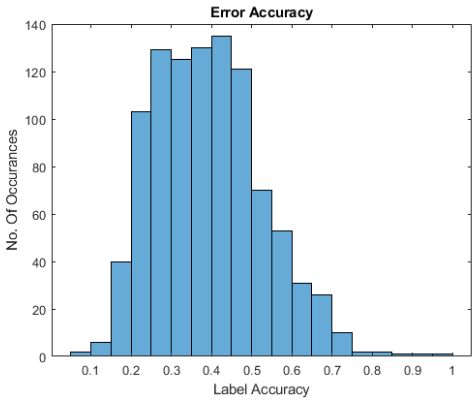
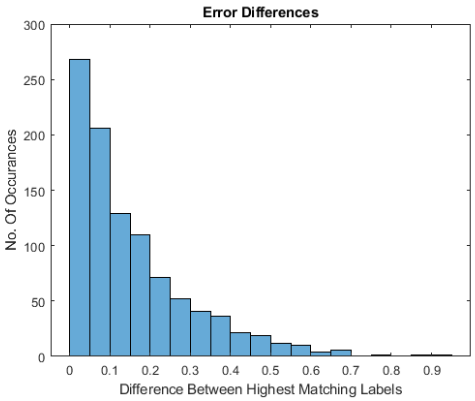


Figure 28: Percentage split of the 988 errors with a mean of 38 and a standard deviation of 21.6

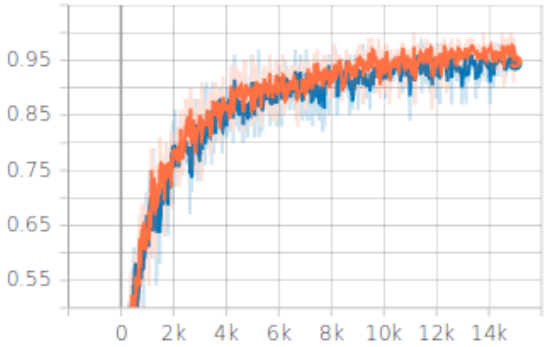
of errors. The accuracy values for the mislabelled images is comparatively high (see Figure 29a), with Inception being assured that the labels it did mismatch are correct, however looking at the differences between the top labels (see Figure 29b), there is an exponential curve formed where even having a threshold of 0.1 would cause 474 errors to be reconsidered (48% of the total amount) with even higher thresholds being worth considering depending on the data's sensitivity. This is derived from the normalised data points $\langle \frac{x_i}{||x_i||}, \frac{y_i}{||y_i||} \rangle$ with 209 principal components made in the regularisation portion of the algorithm, a 90% component reduction from the original 1x2048 bottleneck vector.



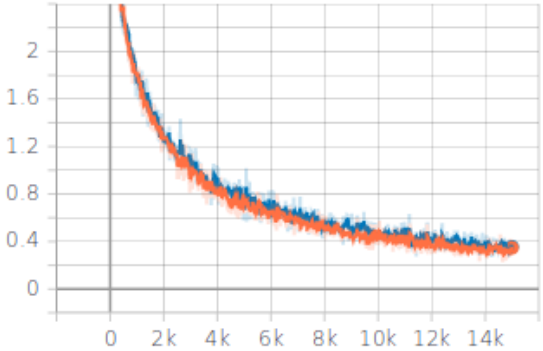
(a) The frequency of the highest accuracy values for each of the present errors.



(b) Accuracy difference between the two labels Inception predicted the highest.



(c) Plot of the accuracy Inception provides over the number of epochs.



(d) Plot of the cross entropy from Inception over the number of epochs

Figure 29: Results from Inception Version 3 tested on the Deaf-Blind dataset.

8.5 Summary

This presents an algorithm capable of detecting BANZSL Deaf-Blind sign language with a high rate of accuracy. This breaks down communication barriers present between the Deaf-Blind population and the rest of the world and with these results. Further development can expand the current deaf-blind vocabulary as new words can avoid repetitious dactylologies (BANZSL has over 5000 words in its lexicon and there is no difficulty adding these as extra classes based on the research provided). Other tactile sign languages can be added such as TADOMA which involves having your middle three fingers over the face of the deaf-blind participant to communicate [153], further increasing the field of what can be recognise by such a technology. Finally, this dataset can be made open source for other users to contribute their own images for further sign language classification.

With the new technology tested on the BANZSL deaf-blind alphabet, the successful re-

sults meant that other avenues of image recognition could be tested for further optimization analysis. Since the current research was in gesture recognition, this can be expanded into two further areas of interest with little to no prior research in their fields and both sets of classes were perfect for further experimentation into dactylological research.

8.6 Further Applications

8.6.1 Semaphore Recognition

Figure 30: Representations of all thirty classes performed with two red and orange flags. These are all static positions except for Error which places both flags upright and is unique to the gesture's motion without conflicting with other signals.

Semaphore is a long range communication system using the transference of visual signals that has been around since 1792 when Claude Chappe came up with a series of relay towers equipped with two moveable flagpoles that could be seen up to 20 miles away with the aid of a spyglass [154]. Today, Semaphore uses two arm length flags to perform full body gestures that correspond to individual letters and numbers with special positions for direct signals (See Figure 30). This practice is still used by maritime organisations and mountaineers where oral and electronic communications can fail with difficult weather conditions, showcasing a current need for further research in automated detection [155]. Existing Semaphore recognition literature alludes to body recognition systems that track the position of the body and it's relation to the gesture being performed [156]. This approach has the advantage of being able to track movements in real time and often with small error rates however as with PARLOMA in the deaf-blind situation, these methods are comparatively impractical to the approach presented here. Using Inception Version 3 for 30,000 images split evenly into 30 classes, the algorithm achieves a 99.82% success rate, with 53 errors between all classes (12 classes contained no errors while B, E & V contained more than six mislabels likely due to class overlap although this is relatively minor).

The technology can be taken even further down this branch, extending the current lexicon. Japanese Flag Semaphore uses a solid red and white flag compared to the diagonal

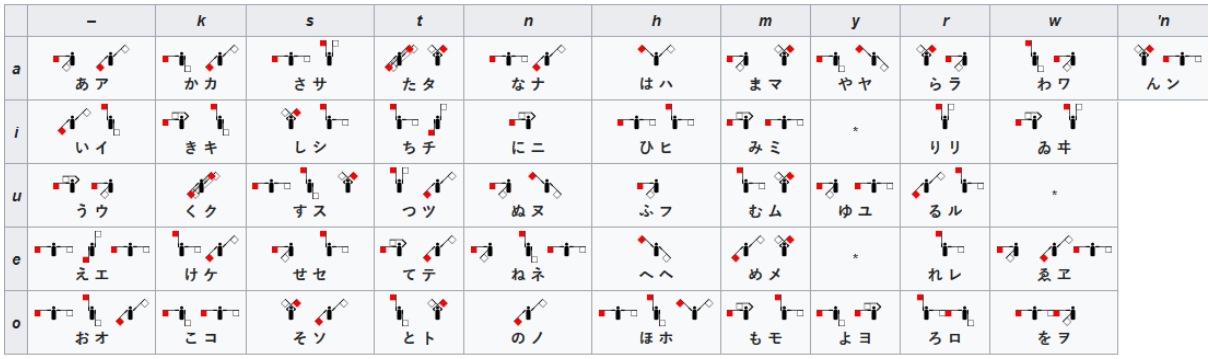


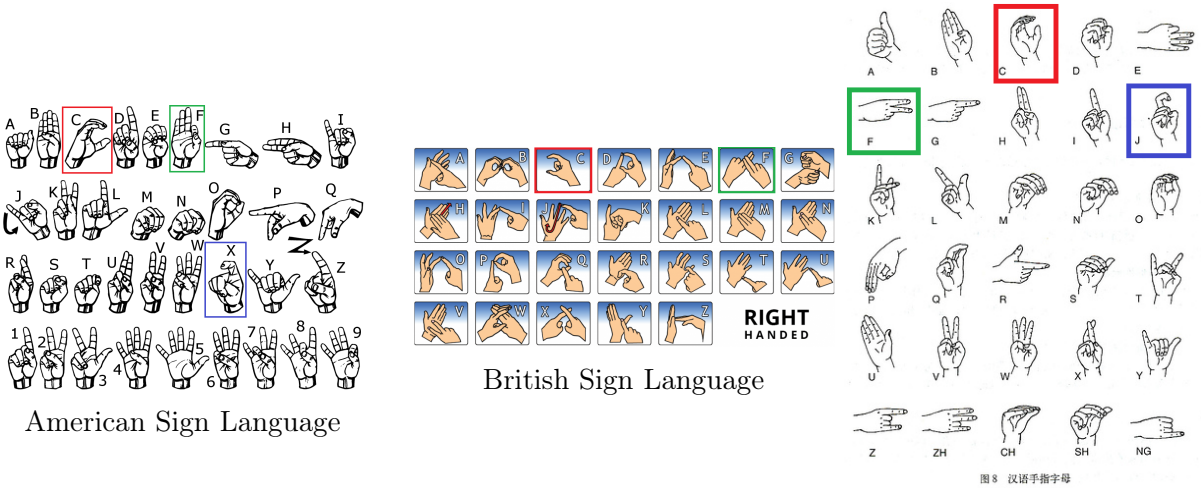
Figure 31: Each of the forty-eight classes in the Japanese Semaphore Alphabet, performed with a red flag in the left hand and a white flag in the right hand [157]. Characters in this format can use combinations of fifteen base signals to represent a larger repertoire of characters such that a theoretical detection would have to take into account the previous positions to deduce the currently signed character.

pattern of western semaphore. This language also consists of 48 different characters for each letter in the katakana alphabet (see Figure 31) and while there have been attempts to categorise the alphabet [158], further improvements can be made. As the flag colours have different patterns, no two positions in both languages are exactly alike, leaving a possible 78 positions that could be tried for an expanded system.

8.6.2 Multiple Sign Language Classification

From the successful results with deaf-blind sign language and the western semaphore alphabet, building on the suggested idea of a program that could classify two different semaphore alphabets with no existing overlap, this idea was carried over into translating multiple tactile sign languages into their textual equivalent. Unlike Semaphore which is only used to transmit long range signals as a secondary communication method, Sign Language is spoken by roughly 70 million people in the world. The exact figures of how many tactile variations there are differ, with some expressions being the visual equivalent of accents rather than unique lexicons however estimates show there are between 150 and 300 catalogued sign language variants [159].
This technology will focus on recognising the eighty-five unique gestures present in three of the most widely used sign languages:

1. American Sign Language (Known by 500,000 people in the United States [140]).
2. British Sign Language (Known by 150,000 people in the United Kingdom [160]).
3. Chinese (Pinyin) Sign Language (Estimated to be known by 2,500,000 people in China).



American Sign Language

British Sign Language

Chinese Sign Language

Figure 32: For multiple languages it's important to correctly categorise unique gestures. For example, the sign for C (Red Box) is consistent through all languages while the symbols for F (Green Box) only occur in those classes. A preference should also be made to the user on which language they think is being presented as the gestures presented in the Blue Box are the same but represent different letters in their respective languages.

While directed single language gesture recognition has produced very promising results, such as a test on a 5113 word lexicon for Chinese Sign Language giving a final accuracy rate of 91.9% [161] with motion gloves, class overlap becomes a very big problem when working with large amounts of gestures. With both intra-class and extra-class similarities being a problem, the work achieved previously [162] will be amended such that the same sign will be universal through each sign language lexicon, and a preference could be made to the end user. This technology is ideal for a user who understands that some tactile language is being performed however they do not understand which dialectical variation this could be. With one notable exception [163], this is an under-covered problem and one where the results could be life changing for the 466 million people in the world who have hearing loss (34 million of these people being children [164]).
InceptionV3 correctly labeled 90.66% of the images. The difference between chosen label value and correct label was small through most of the errors. All the errors were within 3 s.t.d's (43 errors) of the error mean (93 errors), such that there wasn't any particular class that consistently gave mislabelled images.
Despite 7941 errors, this is a strong enough result that further tests on gesture recognition can be warranted. The three datasets were taken by myself and while the most diverse approach was taken, adding more people of differing ages, genders, races and disabilities could improve conventional detection. Since a strong foundation has been established, entire words can be included just as the result for Chinese Sign Language [161] proved

was possible. More popular languages can be added such as Japanese Sign Language (JSL) and Spanish Sign Language (SSL) to increase the programs validity. Finally this research can be made open source, so other users can contribute their own findings or image classes so the algorithm doesn't become complacent picking up on details irrelevant to the gesture classes as a better system is developed within the community.

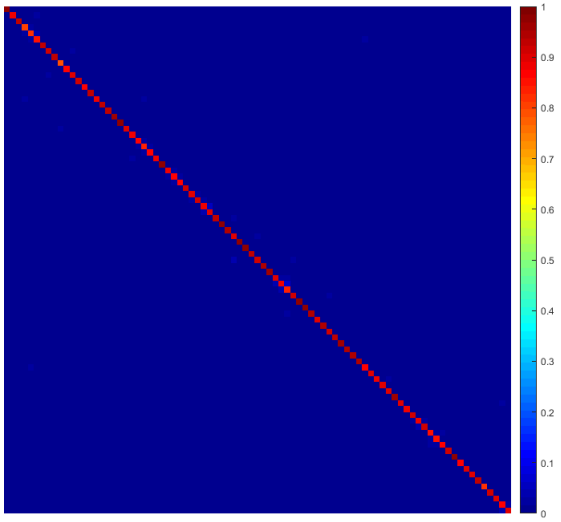


Figure 33: The confusion matrix shows the performance of the supervised learning algorithm. Each row shows the instance in the predicted class while the column represent the actual class. The colour bar shows the intensity of how many labels correspond to that class. The dark red line in the leading diagonal shows a strong rate of true positives while the dark blue squares surrounding it shows there is a very weak concentration of errors scattered elsewhere with no strong tendencies towards a particular incorrect label in any of the 85 cases.

.

Part IV

Extensions Of Stochastic Separation Theorems And Algorithms For Efficient Error Correction In Legacy AI Systems

9 Separation Of K-Tuples For Product Measure Distribution

9.1 Main Results

9.1.1 Mathematical Preliminaries

Following standard assumptions (see e.g. [95], [96]), suppose that all $\boldsymbol{x}$ are generated in accordance with some distribution, and the actual measurements $\boldsymbol{x}_i$ are samples from this distribution. For simplicity, a traditional setting in which all such samples are identically and independently distributed (i.i.d.) [95] is adopted. With regards to the elements of $\boldsymbol{x}_i$, the following technical condition is assumed:

Assumption 1 *Elements $\boldsymbol{x}_i$ are random i.i.d. vectors drawn from a product measure distribution:*

A1) Their x_{ij}-th components are independent and bounded random variables X_j: $-1 \leq X_j \leq 1$, $j = 1, \ldots, n$,

A2) $E[X_j] = 0$, and $E[X_j^2] = \sigma_j^2$.

The distribution itself, however, is supposed to be unknown. Let

$$R_0^2 = \sum_{i=1}^{n} \sigma_i^2 > 0. \tag{60}$$

Then the following result holds (see [1]).

Theorem 4 *Let $\boldsymbol{x}_i \in \mathcal{M} \cup \mathcal{Y}$ be i.i.d. random points from the product distribution satisfying Assumption 1, $0 < \delta < 1$, $0 < \varepsilon < 1$ and $R_0 > 0$. Then*

1) for any i,

$$\begin{aligned} &P\left(1-\varepsilon \le \frac{\|\boldsymbol{x}_i\|^2}{R_0^2} \le 1+\varepsilon\right) \ge 1-2\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right); \\ &P\left(\frac{\|\boldsymbol{x}_i\|^2}{R_0^2} \ge 1-\varepsilon\right) \ge 1-\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right); \end{aligned} \tag{61}$$

2) for any $i, j,\ i \ne j$,

$$\begin{aligned} &P\left(\left\langle \frac{\boldsymbol{x}_i}{R_0}, \frac{\boldsymbol{x}_j}{R_0}\right\rangle < \delta\right) \ge 1-\exp\left(-\frac{R_0^4\delta^2}{2n}\right); \\ &P\left(\left\langle \frac{\boldsymbol{x}_i}{R_0}, \frac{\boldsymbol{x}_j}{R_0}\right\rangle > -\delta\right) \ge 1-\exp\left(-\frac{R_0^4\delta^2}{2n}\right); \end{aligned} \tag{62}$$

3) for any given $\boldsymbol{y} \in [-1,1]^n$ and any i

$$P\left(\left\langle \frac{\boldsymbol{x}_i}{R_0}, \frac{\boldsymbol{y}}{R_0}\right\rangle < \delta\right) \ge 1-\exp\left(-\frac{R_0^4\delta^2}{2n}\right). \tag{63}$$

Proof of Theorem 4 as well as other technical statements are provided in Section 9.4.

Remark 1 Note that if $\sigma_i^2 > 0$, then $R_0^2 > n\min_i\{\sigma_i^2\}$. Hence the r.h.s. of (61)–(63) become exponentially close to 1 for n large enough.

The following Theorem is now immediate:

Theorem 5 (1-Element separation) *Let elements of the set $\mathcal{M} \cup \mathcal{Y}$ be i.i.d. random points from the product distribution satisfying Assumption 1, $0 < \varepsilon < 1$, and $R_0 > 0$. Consider $\boldsymbol{x}_{M+1} \in \mathcal{Y}$ and let*

$$\ell_1(\boldsymbol{x}) = \left\langle \frac{\boldsymbol{x}}{R_0}, \frac{\boldsymbol{x}_{M+1}}{R_0}\right\rangle,\ h_1(\boldsymbol{x}) = \ell_1(\boldsymbol{x}) - 1 + \varepsilon. \tag{64}$$

Then

$$\begin{aligned} &P\left(h_1(\boldsymbol{x}_{M+1}) \ge 0 \text{ and } h_1(\boldsymbol{x}_i) < 0 \text{ for all } \boldsymbol{x}_i \in \mathcal{M}\right) \\ &\ge 1-\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right) - M\exp\left(-\frac{R_0^4(1-\varepsilon)^2}{2n}\right). \end{aligned} \tag{65}$$

(This proof is provided in Section 9.4.)

Remark 2 Theorem 5 establishes the fact that the set $\mathcal{M}$ can be separated away from $\mathcal{Y}$ by a linear functional with reasonably high probability. It also specifies the separating hyperplane, (64), and provides an estimate from below of the probability of such an event,

(65). This estimate, as a function of n, approaches 1 exponentially fast. Note that the result is intrinsically related to the work [113] on quasiorthogonal dimension of Euclidian spaces.

Theorem 6 below summarises the case when the set $\mathcal{Y}$ contains more than one element.

Theorem 6 (k-Element separation. Case 1) *Let elements of the set $\mathcal{M} \cup \mathcal{Y}$ be i.i.d. random points from the product distribution satisfying Assumption 1, and $R_0 > 0$. Let, additionally,*

$$\left\langle \frac{\boldsymbol{x}_i}{R_0}, \frac{\boldsymbol{x}_j}{R_0} \right\rangle \geq \beta, \tag{66}$$

for all $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{Y}$, $\boldsymbol{x}_i \neq \boldsymbol{x}_j$. Pick $0 < \varepsilon < 1$. Observe that

$$1 - \varepsilon + \beta(k-1) > 0, \tag{67}$$

and consider

$$\begin{aligned} \ell_k(\boldsymbol{x}) =& \left\langle \frac{\boldsymbol{x}}{R_0}, \frac{\overline{\boldsymbol{x}}}{R_0} \right\rangle, \ \overline{\boldsymbol{x}} = \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{x}_{M+i}, \\ h_k(\boldsymbol{x}) =& \ell_k(\boldsymbol{x}) - \frac{1 - \varepsilon + \beta(k-1)}{k}. \end{aligned} \tag{68}$$

Then

$$\begin{aligned} & P\left(h_k(\boldsymbol{x}_j) \geq 0 \ \& \ h_k(\boldsymbol{x}_i) < 0 \text{ for all } \boldsymbol{x}_i \in \mathcal{M}, \ \boldsymbol{x}_j \in \mathcal{Y}\right) \\ & \geq 1 - k \exp\left(-\frac{2R_0^4 \varepsilon^2}{n}\right) - M \exp\left(-\frac{R_0^4 (1 - \varepsilon + \beta(k-1))^2}{2k^2 n}\right). \end{aligned} \tag{69}$$

(The proof is provided in Section 9.4.) The value of β in estimate (66) may not be available. If this is the case then the corollaries from Theorems 5 and 6 apply.

Corollary 1 (k-Element separation. Case 1) *Let elements of the set $\mathcal{M} \cup \mathcal{Y}$ be i.i.d. random points from the product distribution satisfying Assumption 1, and $R_0 > 0$. Pick $0 < \delta, \& 0 < \varepsilon < 1$. Observe that*

$$1 - \varepsilon - \delta(k-1) > 0, \tag{70}$$

and let

$$\begin{aligned} \ell_k(\boldsymbol{x}) =& \left\langle \frac{\boldsymbol{x}}{R_0}, \frac{\overline{\boldsymbol{x}}}{R_0} \right\rangle, \ \overline{\boldsymbol{x}} = \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{x}_{M+i}, \\ h_k(\boldsymbol{x}) =& \ell_k(\boldsymbol{x}) - \frac{1 - \varepsilon - \delta(k-1)}{k}. \end{aligned} \tag{71}$$

Then

$$\begin{aligned} &P\left(h_k(\boldsymbol{x}_j) \geq 0 \ \&\ h_k(\boldsymbol{x}_i) < 0 \text{ for all } \boldsymbol{x}_i \in \mathcal{M},\ \boldsymbol{x}_j \in \mathcal{Y}\right) \\ &\geq 1 - k\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right) - \frac{k(k-1)}{2}\exp\left(-\frac{R_0^4\delta^2}{2n}\right) \\ &- M\exp\left(-\frac{R_0^4(1-\varepsilon-\delta(k-1))^2}{2k^2 n}\right). \end{aligned} \tag{72}$$

Corollary 2 (k-Element separation. Case 2) *Let elements of the set $\mathcal{M}\cup\mathcal{Y}$ be i.i.d. random points from the product distribution satisfying Assumption 1, $0 < \varepsilon < 1$, $0 < \mu < 1-\varepsilon$ and $R_0 > 0$. Pick $\boldsymbol{x}_j \in \mathcal{Y}$ and consider*

$$\begin{aligned} \ell_k(\boldsymbol{x}) = \left\langle \frac{\boldsymbol{x}}{R_0}, \frac{\boldsymbol{x}_j}{R_0} \right\rangle,\ h_k(\boldsymbol{x}) = \ell_k(\boldsymbol{x}) - 1 + \varepsilon + \mu, \\ \Omega = \left\{ \boldsymbol{x} \in \mathbb{R}^n \ \middle|\ \left\langle \frac{\boldsymbol{x}_j}{R_0}, \frac{\boldsymbol{x}_j - \boldsymbol{x}}{R_0} \right\rangle \leq \mu \right\}. \end{aligned} \tag{73}$$

Then

$$\begin{aligned} &P\left(h_k(\boldsymbol{x}) \geq 0 \text{ and } h_k(\boldsymbol{x}_i) < 0 \text{ for all } \boldsymbol{x}_i \in \mathcal{M}, \boldsymbol{x} \in \Omega\right) \\ &\geq 1 - k\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right) - M\exp\left(-\frac{R_0^4(1-\varepsilon-\mu)^2}{2n}\right). \end{aligned} \tag{74}$$

Proofs of the corollaries are provided in Section 9.4. Theorems 4 – 6 and Corollaries 1, 2 suggest that simple elements (54) with f being mere threshold elements

$$f(s) = \mathrm{Step}(s) = \begin{cases} 1, & s \geq 0 \\ 0, & s < 0 \end{cases}, \tag{75}$$

possess remarkable selectivity. For example, according to Theorem 5, the element

$$f\left(\langle \boldsymbol{x}, \boldsymbol{w}\rangle - c\right),\ \boldsymbol{w} = \frac{\boldsymbol{x}_{M+1}}{R_0^2},\ c = \frac{\|\boldsymbol{x}_{M+1}\|^2}{R_0^2}, \tag{76}$$

assigns "1" to $\boldsymbol{x}_{M+1}$ and "0" to all $\boldsymbol{x}_i \in \mathcal{M}$ with probability that is exponentially (in n) close to 1. To illustrate this point, consider a simple test case with a set comprising of $10,000$ i.i.d. samples from the (uniform) product distribution in $[-1,1]^n$. For this distribution, $R_0 = \sqrt{n/3}$, $|\mathcal{M}| = M = 9999$, and estimate (65) becomes:

$$\begin{aligned} &P\left(h_1(\boldsymbol{x}_{M+1}) \geq 0 \text{ and } h_1(\boldsymbol{x}_i) < 0 \text{ for all } \boldsymbol{x}_i \in \mathcal{M}\right) \\ &\geq 1 - \exp\left(-\frac{2}{9}n\varepsilon^2\right) - M\exp\left(-\frac{1}{18}n(1-\varepsilon)^2\right). \end{aligned} \tag{77}$$

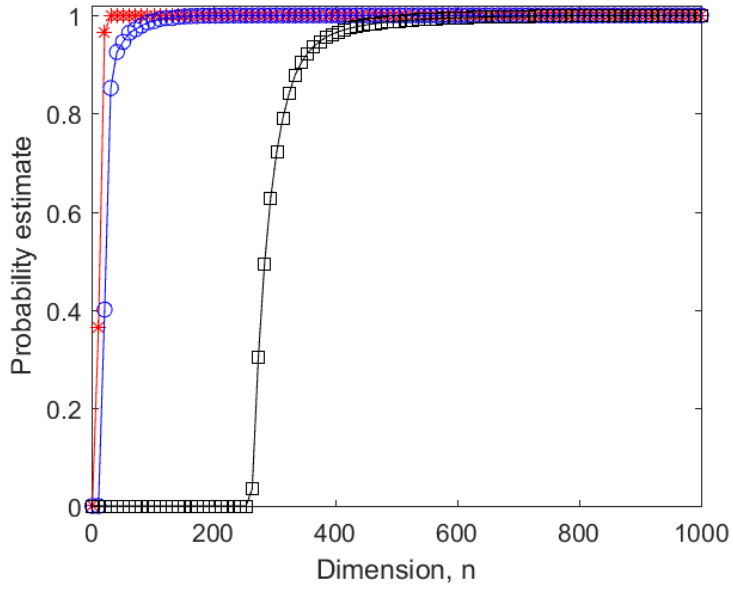


Figure 34: The probability (frequency) that an element of the set $\mathcal{M}$ (formed by random i.i.d. vectors drawn from the n-dimensional cube $[-1,1]^n$) is separable from the rest by hyperplanes (64) (and their variants) as a function of n; $M = |\mathcal{M}| = 10,000$. Red stars correspond to empirical frequencies of the events A_i: $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle < \|\boldsymbol{x}_i\|^2$ for all $\boldsymbol{x}_j \in \mathcal{M}$, $\boldsymbol{x}_i \neq \boldsymbol{x}_j$. Blue circles show frequencies of the events B_i: $\|\boldsymbol{x}_i\|^2/R_0^2 \geq 1-\varepsilon$ and $\langle \boldsymbol{x}_i/R_0, \boldsymbol{x}_j/R_0 \rangle < 1-\varepsilon$ for all $\boldsymbol{x}_j \in \mathcal{M}$, $\boldsymbol{x}_i \neq \boldsymbol{x}_j$, $\varepsilon = 0.2$. Black squares show the right-hand side of probability estimate (77) for the same value of $\varepsilon = 0.2$.

The right-hand side rapidly approaches 1 as n grows. The estimate, however, could be rather conservative as is illustrated with Figure 34. An alternative approach allows a small margin of error by determining a hyperplane separating the set $\mathcal{Y}$ from nearly all elements $\boldsymbol{x} \in \mathcal{M}$. An estimate of success for the latter case follows from Lemma 1 below

Lemma 1 *Let elements of the set $\mathcal{M}$ be i.i.d. random points, and let*

$$\ell(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{w} \rangle\,, \;\; h(\boldsymbol{x}) = \ell(\boldsymbol{x}) - c, \tag{78}$$

and $0 \leq p_* \leq 1$ be such that

$$P\left(h(\boldsymbol{x}_i) \geq 0\right) \leq p_*, \tag{79}$$

for an arbitrary element $\boldsymbol{x}_i \in \mathcal{M}$. Then

$$\begin{aligned} &P\left(h(\boldsymbol{x}) \geq 0 \text{ for at most } m \text{ elements } \boldsymbol{x} \in \mathcal{M}\right) \geq \\ &\quad (1-p_*)^M \exp\left(\frac{(M-m+1)p_*}{1-p_*}\right)\left(1 - \frac{1}{m!}\left(\frac{(M-m+1)p_*}{(1-p_*)}\right)^m\right). \end{aligned} \tag{80}$$

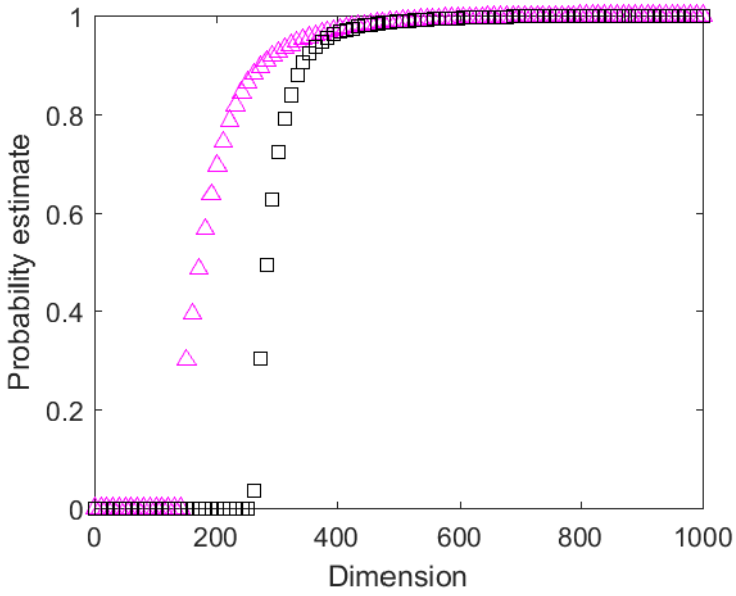


Figure 35: Probability estimates (77) (black squares) and (81) (magenta triangles) as functions of n for the set $\mathcal{M}$ formed by random i.i.d. vectors drawn from the n-dimensional cube $[-1,1]^n$ and $\varepsilon = 0.2$.

(The proof is provided in 9.4.) According to Lemma 1 and Theorem 4,

$$\begin{aligned} &P\left(h_1(\boldsymbol{x}_{M+1}) \geq 0 \text{ and } h_1(\boldsymbol{x}_i) \geq 0 \text{ for at most } n\ \boldsymbol{x}_i \in \mathcal{M}\right) \\ &\geq (1-p_*)^M \exp\left(\frac{(M-n+1)p_*}{1-p_*}\right)\left(1-\frac{1}{n!}\left(\frac{(M-n+1)p_*}{(1-p_*)}\right)^n\right) \\ &- \exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right),\ p_* = \exp\left(-\frac{R_0^4(1-\varepsilon)^2}{2n}\right). \end{aligned} \tag{81}$$

$n+1$ random points (in general position) are linearly separable with probability 1. With probability 1, spurious n points can be separated away from $\boldsymbol{x}_{M+1}$ by a second hyperplane. Figure 35 compares separation probability estimates for such a pair of hyperplanes with that of the hyperplane formed by h_1 in Theorem 5, (65).

9.1.2 Fast AI Up-Training Algorithms

Theorem 6 as well as Corollaries 1, 2 and Lemma 1 provide a simple computational framework for construction of networks and cascades of elements (54). Following the setup presented in Section 4.2, recall that the data is sampled from the core AI system and is partitioned into the sets $\mathcal{M}$ and $\mathcal{Y}$. The latter set $\mathcal{Y}$ corresponds to singular events which are picked up by the cascade. Their union, $\mathcal{S} = \mathcal{M} \cup \mathcal{Y}$, is the entire available dataset for cascade construction. The algorithms for fast construction of the correcting

ensembles can now be developed. The first algorithm is a recursion of which each iteration is a multi-step process. This algorithm is provided below.

Algorithm 1 (Correcting Ensembles) Initialization: set $\mathcal{M}^1 = \mathcal{M}$, $\mathcal{Y}^1 = \mathcal{Y}$, and $\mathcal{S}^1 = \mathcal{S}$, define a list or a model of *correcting actions* – formalised alternations of the core AI in response to an error/error type.

Input to the i-th iteration: Sets $\mathcal{M}^i$, $\mathcal{Y}^i$, and $\mathcal{S}^i = \mathcal{M}^i \cup \mathcal{Y}^i$; the number of clusters, p, and the value of filtering threshold, θ.

1. *Centering.* All current data available is centered. The centered sets $\mathcal{S}_c^i$ and $\mathcal{Y}_c^i$ are formulated by subtracting the mean $\overline{\boldsymbol{x}}(\mathcal{S}^i)$ from the elements of $\mathcal{S}^i$ and $\mathcal{Y}^i$, respectively:

$$\mathcal{S}_c^i = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x} = \xi - \overline{\boldsymbol{x}}(\mathcal{S}^i),\, \xi \in \mathcal{S}^i\}, \tag{82}$$

$$\mathcal{Y}_c^i = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x} = \xi - \overline{\boldsymbol{x}}(\mathcal{S}^i),\, \xi \in \mathcal{Y}^i\}, \tag{83}$$

2. *Regularization.* The covariance matrix, $\mathrm{Cov}(\mathcal{S}^i)$, of $\mathcal{S}^i$ is calculated with the corresponding eigenvalues and eigenvectors. New regularized sets $\mathcal{S}_r$, $\mathcal{Y}_r$ are produced as follows. All eigenvectors $h_1, h_2, \ldots, h_m$ with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$ that are above a given threshold combine into a single matrix $H = (h_1|h_2|\cdots|h_m)$. The threshold can be chosen on the basis of the Kaiser-Guttman test [165] or otherwise (e.g. by keeping the ratio of the maximal to the minimal in the set of retained eigenvalues within a given bound). The new sets are defined as:

$$\mathcal{S}_r^i = \{\boldsymbol{x} \in \mathbb{R}^m | \boldsymbol{x} = H^T\xi,\, \xi \in \mathcal{S}_c^i\}, \tag{84}$$

$$\mathcal{Y}_r^i = \{\boldsymbol{x} \in \mathbb{R}^m | \boldsymbol{x} = H^T\xi,\, \xi \in \mathcal{Y}_c^i\}, \tag{85}$$

3. *Whitening.* The two sets then undergo a whitening coordinate transformation ensuring that the covariance matrix of the transformed data is the identity matrix:

$$\mathcal{S}_w^i = \{\boldsymbol{x} \in \mathbb{R}^m | \boldsymbol{x} = W\xi,\, \xi \in \mathcal{S}_r^i,\; W = \mathrm{Cov}(\mathcal{S}_r^i)^{-\frac{1}{2}}\}, \tag{86}$$

$$\mathcal{Y}_w^i = \{\boldsymbol{x} \in \mathbb{R}^m | \boldsymbol{x} = W\xi,\, \xi \in \mathcal{Y}_r^i,\; W = \mathrm{Cov}(\mathcal{S}_r^i)^{-\frac{1}{2}}\}, \tag{87}$$

4. *Projection (optional).* Project elements of $\mathcal{S}_w^i$, $\mathcal{Y}_w^i$ onto the unit sphere by scaling them to the unit length: $\boldsymbol{x} \mapsto \varphi(\boldsymbol{x})$, $\varphi(\boldsymbol{x}) = \boldsymbol{x}/\|\boldsymbol{x}\|$.

5. *Training: Clustering.* The set $\mathcal{Y}_w^i$ (the set of errors) is partitioned into p clusters $\mathcal{Y}_{w,1}^i, \ldots, \mathcal{Y}_{w,p}^i$ that's elements are pairwise positively correlated.

6. *Training: Nodes creation and aggregation.* For each $\mathcal{Y}^i_{w,j}$, $j = 1, \ldots, p$ and its complement $\mathcal{S}^i_w \setminus \mathcal{Y}^i_{w,\,j}$ the following separating hyperplanes are constructed:

$$\begin{aligned} h_j(\boldsymbol{x}) =& \ell_j(\boldsymbol{x}) - c_j, \\ \ell_j(\boldsymbol{x}) =& \left\langle \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|}, \boldsymbol{x} \right\rangle, \; c_j = \min_{\xi \in \mathcal{Y}^i_{w,\,j}} \left\langle \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|}, \xi \right\rangle \\ \boldsymbol{w}_j =& (\mathrm{Cov}(\mathcal{S}^i_w \setminus \mathcal{Y}^i_{w,j}) + \mathrm{Cov}(\mathcal{Y}^i_{w,j}))^{-1} (\overline{\boldsymbol{x}}(\mathcal{Y}^i_{w,j}) - \overline{\boldsymbol{x}}(\mathcal{S}^i_w \setminus \mathcal{Y}^i_{w,j})). \end{aligned} \tag{88}$$

   Retain only hyperplanes where $c_j > \theta$. For each retained hyperplane, a corresponding element (54) is made:

$$f_j(\boldsymbol{x}) = f\left(\left\langle W H^T (\boldsymbol{x} - \overline{\boldsymbol{x}}(\mathcal{S}^i)), \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|} \right\rangle - c_j \right), \tag{89}$$

   with f being a function that satisfies: $f(s) > 0$ for all $s \geq 0$ and $f(s) \leq 0$ for all $s < 0$, and add it to the ensemble. If optional step 4 was used then the functional definition of f_j becomes:

$$f_j(\boldsymbol{x}) = f\left(\left\langle \varphi\left(W H^T (\boldsymbol{x} - \overline{\boldsymbol{x}}(\mathcal{S}^i)) \right), \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|} \right\rangle - c_j \right), \tag{90}$$

   where φ is the mapping implementing projection onto the unit sphere.

7. *Integration/Deployment.* Any $\boldsymbol{x}$ generated by the original core AI is put through the ensemble of $f_j(\cdot)$. If for some $\boldsymbol{x}$, any value of $f_j(\boldsymbol{x}) > 0$, then a *correcting action* is performed on the core AI. The action depends on the purpose of correction as well as on the problem at hand. This could include label swapping, signalling an alarm, ignoring/not reporting etc. The combined system becomes new core AI.

8. *Testing.* Assess performance of new core AI on a relevant data set. If needed, generate new sets $\mathcal{M}^{i+1}$, $\mathcal{Y}^{i+1}$, $\mathcal{S}^{i+1}$ (with possibly different error types and definitions), and repeat the procedure.

Steps 1–3 of the algorithm are standard pre-processing routines. In the context of Theorems 4–6, step 1 ensures that the first part of *A2)* in Assumption 1 holds, and step 2, along with regularization, guarantees that all components of $\boldsymbol{x}_i$ are uncorrelated (a necessary condition for part *A1)* of Assumption 1 to hold). The Whitening transformation in step 3, normalizes the data so that $\sigma_i^2 = 1$ for all relevant values of i. Step 4, whilst optional, is introduced to account for data irregularities and clustering that may negatively affect data separability. An illustration of potential utility of this step is shown in Figure 36, however individual components of the data vectors may no longer satisfy the

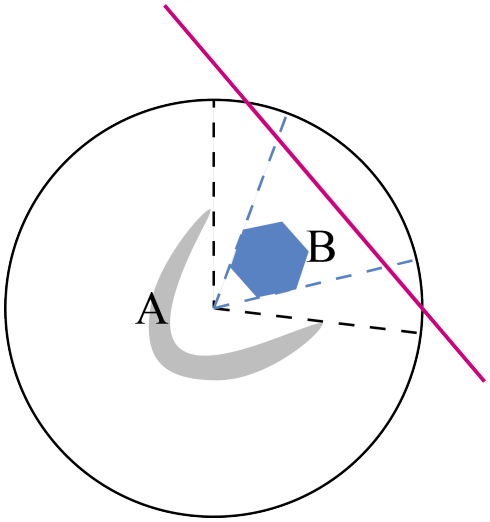


Figure 36: Linear separability via projection onto a sphere. Sets A (shown as a shaded gray domain) and B (depicted as a blue filled hexagon) are not linearly separable in the original coordinates. Their projections onto the sphere, however, are separable (the separating hyperplane is shown in red).

independence assumption. Nevertheless, if the data is reasonably distributed, different versions of separation theorems may apply to this case too [2], [6]. Section 9.2 illustrates the effect of this step in an example application. Step 5, clustering, is motivated by Theorem 6 and Corollaries 1, 2 suggesting that the rate of success in isolating multiple points from the background set $\mathcal{M}$ increases when these multiple points are positively correlated or are spatially close to each other. Step 6 is a version of (68), with a different normalization and a slight perturbation of weights. This particular normalization is motivated by the experiments shown in Figure 34. As for the choice of weights, these hyperplanes are standard Fisher discriminants, although they are not far from (68). In view of the previous steps, $\mathrm{Cov}(\mathcal{S}^i_w \setminus \mathcal{Y}^i_{w,j}) + \mathrm{Cov}(\mathcal{Y}^i_{w,j})$ is diagonally dominated and close to the identity matrix. When $|\mathcal{S}^i_w| \gg |\mathcal{Y}^i_{w,j}|$, term $\overline{\boldsymbol{x}}(\mathcal{S}^i_w \setminus \mathcal{Y}^i_{w,j})$ is nearly zero, and hence $\boldsymbol{w}_j$ is approximately at the centroid of the cluster. Node filtering is necessitated by the concentration of measure effects as exemplified by Theorem 4. Functions f in step 7 can be implemented using a range of ReLU, threshold, $\tanh(\cdot)$, sigmoidal functions etc. These are available in the majority of standard core AI systems. Computational complexity of each step in the recursion, except for step 5, is at most $M + k$. Complexity of the clustering step may generally be superpolynomial (worst case) as is e.g. the case for standard k-means clustering [166]. If sub-optimal solutions are accepted [167] then the complexity of this step scales linearly with $M + k$. Further computational optimisations are possible through randomised procedures such as in k-means++ [168]. Algorithm 1 is based on the intuition and rationale stemming from Theorem 5, 6 and their corollaries. This can be

modified to take advantage of the possibilities offered by Lemma 1 and the subsequent discussions. The modifications are that each node added in step 6 is assessed and "corrected" by an additional hyperplane as required. The modified algorithm is summarised as Algorithm 2 below.

Algorithm 2 (With cascaded pairs) Initialization: set $\mathcal{M}^1 = \mathcal{M}$, $\mathcal{Y}^1 = \mathcal{Y}$, and $\mathcal{S}^1 = \mathcal{S}$, define a list or a model of *correcting actions* – formalised alternations of the core AI in response to an error/error type.

Input to the i-th iteration: Sets $\mathcal{M}^i$, $\mathcal{Y}^i$, and $\mathcal{S}^i = \mathcal{M}^i \cup \mathcal{Y}^i$; the number of clusters, p, and the value of filtering threshold, θ.

1-5. As in Algorithm 1.

6. *Training: Nodes creation and aggregation.* For each $\mathcal{Y}^i_{w,j}$, $j = 1, \dots, p$ and its complement $\mathcal{S}^i_w \setminus \mathcal{Y}^i_{w,\,j}$ construct the following separating hyperplanes:

$$\begin{aligned} h_j(\boldsymbol{x}) =& \ell_j(\boldsymbol{x}) - c_j, \\ \ell_j(\boldsymbol{x}) =& \left\langle \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|}, \boldsymbol{x} \right\rangle, \; c_j = \min_{\xi \in \mathcal{Y}^i_{w,\,j}} \left\langle \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|}, \xi \right\rangle \\ \boldsymbol{w}_j =& (\mathrm{Cov}(\mathcal{S}^i_w \setminus \mathcal{Y}^i_{w,j}) + \mathrm{Cov}(\mathcal{Y}^i_{w,j}))^{-1} (\overline{\boldsymbol{x}}(\mathcal{Y}^i_{w,j}) - \overline{\boldsymbol{x}}(\mathcal{S}^i_w \setminus \mathcal{Y}^i_{w,j})). \end{aligned} \tag{91}$$

Only retain hyperplanes where $c_j > \theta$. For each retained hyperplane, create a corresponding element $f_j(\cdot)$ in accordance to element (89) (or (90) if step 4 was used). For each retained hyperplane,

- Determine the complementary set $\mathcal{C}_j$ comprised of elements $\boldsymbol{x} \in \mathcal{S}^i_w \setminus \mathcal{Y}^i_w$ for which $h_j(\boldsymbol{x}) \geq 0$ (the set of points that are accidentally picked up by the f_j "by mistake")
- project elements of the set $\mathcal{C}_j \cup \mathcal{Y}^i_{w,j}$ orthogonally onto the hyperplane $h_j(\boldsymbol{x}) = \ell_j(\boldsymbol{x}) - c_j$ as

$$\boldsymbol{x} \mapsto \left(I - \frac{1}{\|\boldsymbol{w}_j\|^2} \boldsymbol{w}_j \boldsymbol{w}_j^T \right) \boldsymbol{x} + c_j \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|} = P(\boldsymbol{w}_j)\boldsymbol{x} + b(\boldsymbol{w}_j, c_j); \tag{92}$$

- Determine a hyperplane

$$h_{j,2}(\boldsymbol{x}) = \ell_{j,2}(\boldsymbol{x}) - c_{j,2}, \; \ell_{j,2}(\boldsymbol{x}) = \langle \boldsymbol{w}_{j,2}, \boldsymbol{x} \rangle. \tag{93}$$

Separating projections of $\mathcal{C}_j$ from $\mathcal{Y}^i_{w,j}$ so that $h_{j,2}(\boldsymbol{x}) < 0$ for all projections from $\mathcal{C}_j$ and $h_{j,2}(\boldsymbol{x}) \geq 0$ for all projections from $\mathcal{Y}^i_{w,j}$. If no such planes exist, a fisher linear discriminant or another relevant procedure may be used.

- Create a node

$$f_j^{\perp}(\boldsymbol{x}) = f\left(\left\langle P(\boldsymbol{w}_j)WH^T(\boldsymbol{x} - \overline{\boldsymbol{x}}(\mathcal{S}^i)) + b(\boldsymbol{w}_j, c_j), \boldsymbol{w}_{j,2}\right\rangle - c_{j,2}\right), \tag{94}$$

  or, in case step 4 was used

$$f_j^{\perp}(\boldsymbol{x}) = f\left(\left\langle P(\boldsymbol{w}_j)\varphi(WH^T(\boldsymbol{x} - \overline{\boldsymbol{x}}(\mathcal{S}^i))) + b(\boldsymbol{w}_j, c_j), \boldsymbol{w}_{j,2}\right\rangle - c_{j,2}\right); \tag{95}$$

- Create the pair's response node, f_j^{c}, so that a non-negative response is generated only when both nodes produce a non-negative response

$$f_j^{\mathrm{c}}(\boldsymbol{x}) = f\big(\mathrm{Step}(f_j(\boldsymbol{x})) + \mathrm{Step}(f_j^{\perp}(\boldsymbol{x})) - 1\big); \tag{96}$$

- add the combined $f_j^{\mathrm{c}}(\cdot)$ to the ensemble.

7. *Integration/Deployment.* Any $\boldsymbol{x}$ that is generated by the original core AI is put through the ensemble of $f_j^{\mathrm{c}}(\cdot)$. If for some $\boldsymbol{x}$ any of the values of $f_j^{\mathrm{c}}(\boldsymbol{x}) > 0$ then a *correcting action* is performed on the core AI. The action is dependent on the purpose of correction as well as the current issue. This could include label swapping, signalling an alarm, ignoring/not reporting etc. The combined system becomes new core AI.

8. *Testing.* Assess performance of new core AI on a relevant data set. If needed, generate new sets $\mathcal{M}^{i+1}$, $\mathcal{Y}^{i+1}$, $\mathcal{S}^{i+1}$ (with possibly different error types and definitions), and repeat the procedure.

The next section illustrates how these algorithms work in a case study example involving a core AI in the form of a reasonably large convolutional network trained on a moderate-size dataset.

9.2 Distinguishing the Ten Digits in American Sign Language: a Case Study

9.2.1 Setup and Datasets

This case study investigates and tests the approach on a challenging problem of gesture recognition in the framework of distinguishing ten digits in American Sign Language. To apply the approach, a core AI had to be generated first. As the core AI, Inception Version 3 [170] (whose architecture is shown in Figure 37) is used. The model was trained on ten sets of images that correspond to the American Sign Language gestures for 0-9 (see Figure

38). Each set contained 1000 unique images consisting of profile shots of the person's

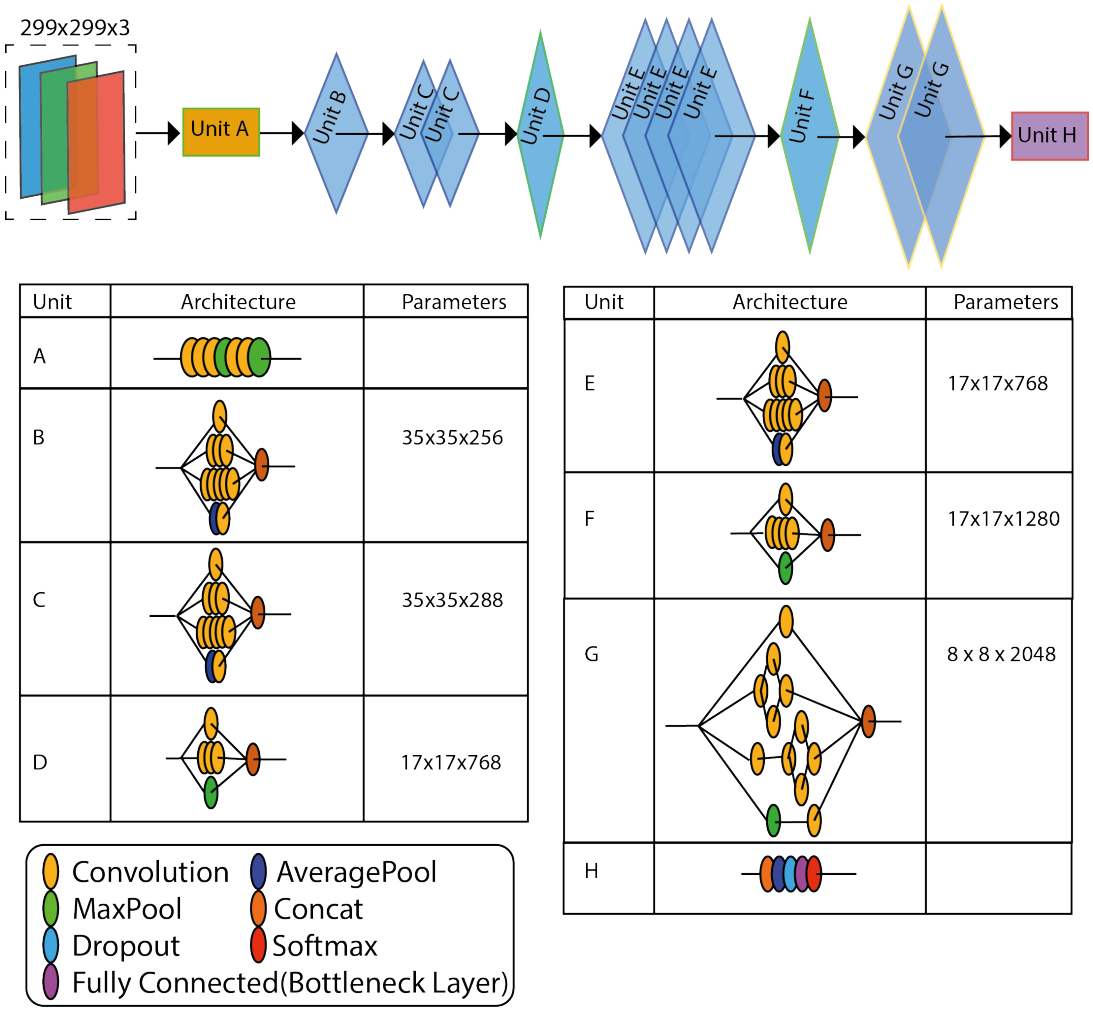


Figure 37: Architecture of the adopted Inception deep neural network model.

hand, along with 3/4 profiles and shots from above and below. The states $\boldsymbol{x}_i$ are the vectors containing the values of pre-softmax layer bottlenecks of size n for however many neurons are in the penultimate layer. Schematically the network's layer whose outputs are $\boldsymbol{x}_i$ is shown in the diagram in Figure 37 as the fully connected (bottleneck layer) in Unit H. Elements of this set that gave incorrect readings are noted and copied into the set $\mathcal{Y}$.

9.2.2 Experiments and results

Once the network was trained, additional 10,000 images of the same ratio were evaluated using the trained system. For these experiments the classification decision rule was to return a gesture number that corresponds to the network output with the highest score if the highest score exceeds a given threshold, $\gamma \in [0, 1]$. Ties are resolved by a arbitrarily

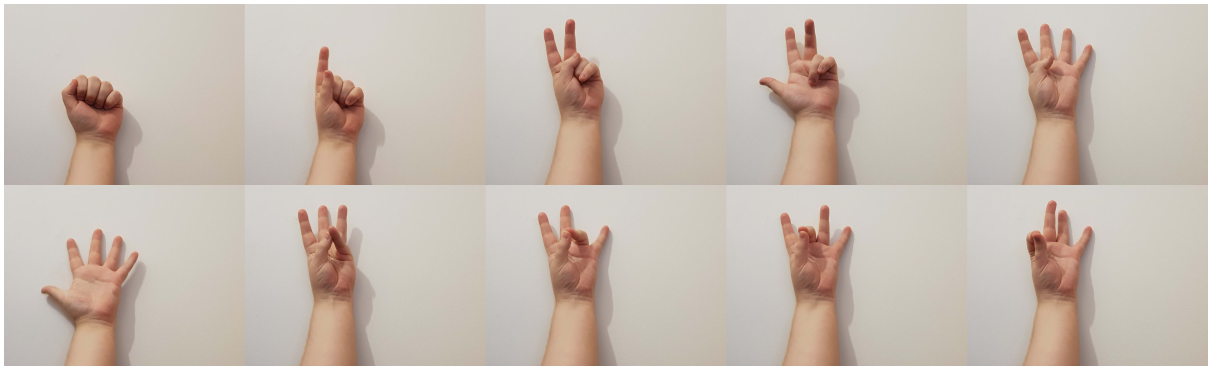


Figure 38: Examples of the sort of images that appear in the current model's data set of the American Sign Language single hand positions for 0 (top left) to 9 (bottom right)

Table 4: Error types in the system. Stars mark instances/events that are not accounted for.

Presence of a gesture	System's response	Error Type
Yes	Correctly classified	True Positive
	Incorrectly classified	False Positive
	Not reported	False Negative
No	Reported	False Positive*
	Not reported	True Negative*

random selection. If the highest score is smaller than or equal to the threshold then no responses are returned. To assess performance of the resulting classifier, the following indicators as functions of threshold γ hold:

$$\begin{aligned} \text{True Positive Rate } (\gamma) &= \frac{TP(\gamma)}{FN(\gamma) + TP(\gamma)}, \\ \text{Misclassification Rate } (\gamma) &= \frac{FP(\gamma)}{FP(0)}, \end{aligned} \tag{97}$$

where $TP(\gamma)$ is the number of true positives, $FN(\gamma)$ is the number of false negatives, and $FP(\gamma)$ is the number of false positives. The definitions of these variables are provided in Table 4. In these experiments no negatives were added to the test set, as the original focus is to illustrate how the approach may cope with misclassification errors. This implies that the number of True Negatives is 0 for any threshold $\gamma \in [0, 1]$ and, consequently, the rate of false positives is always 1. Therefore, instead of using traditional ROC curves showing the rate of true positives against the rate of false positives, a different family of curves in which the rate of false positives is replaced with the rate of misclassification as defined by (97) are utilised. A corresponding performance curve for this Core AI system is shown in Figure 39. At $\gamma = 0$ the result was an 82.4% success rate. At this end of the

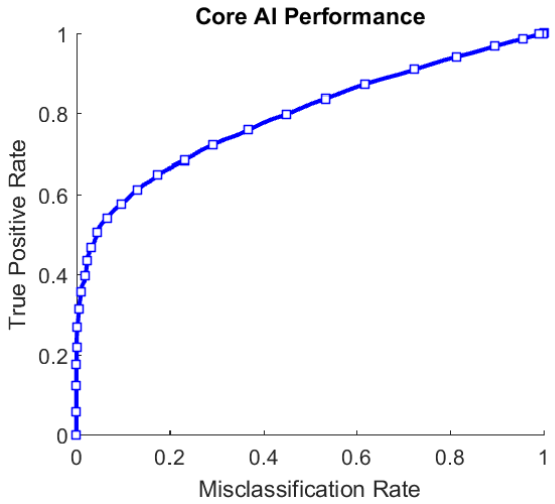


Figure 39: Performance curve for the Core AI system (Inception) on the testing set of 10,000 images. Threshold γ was varying from 1 (bottom left corner) to 0 (top right corner)

Table 5: Errors per gesture (misclassifications). Top row corresponds to the gesture number, the bottom row indicates the number of errors for each gesture.

0	1	2	3	4	5	6	7	8	9
10	52	235	62	410	80	269	327	207	108

curve, the observed performance was comparable/similar to that reported in e.g. [169] (see also references therein). The numbers of errors per each gesture in the trained system are shown in Table 5. The variance of errors is mostly consistent among the ten classes with very few errors for the "0" gesture, likely due to its unique shape among the classes. Once the errors were isolated, Algorithms 1 and 2 construct correcting ensembles to improve the original core AI. To train the ensembles, the testing data set of 10,000 images that was implemented to assess performance of Inception was split into two non-overlapping subsets. The first subset, comprised of 6592 records of data points corresponding to correct responses and 1408 records corresponding to errors, was used to train these correcting ensembles. This subset was the ensemble's training set, and it accounted for 80% of the data. The second subset, the ensemble's testing set, combined 1648 data points of correct responses and 352 elements labelled as errors to test the ensemble.
Both algorithms have been run on the first subset, the ensemble's training set. For simplicity of comparison and evaluation, this was only iterated once (i.e. did not build cascades of ensembles). In the regularization step, the Kaiser-Guttman test returned 174 principal components reducing the original dimensionality more than 10 times. After the whitening transformation, step 3, the values of $|\langle \boldsymbol{x}_i/\|\boldsymbol{x}_i\|, \boldsymbol{x}_j/\|\boldsymbol{x}_j\|\rangle|$ (shown in Figure 40)

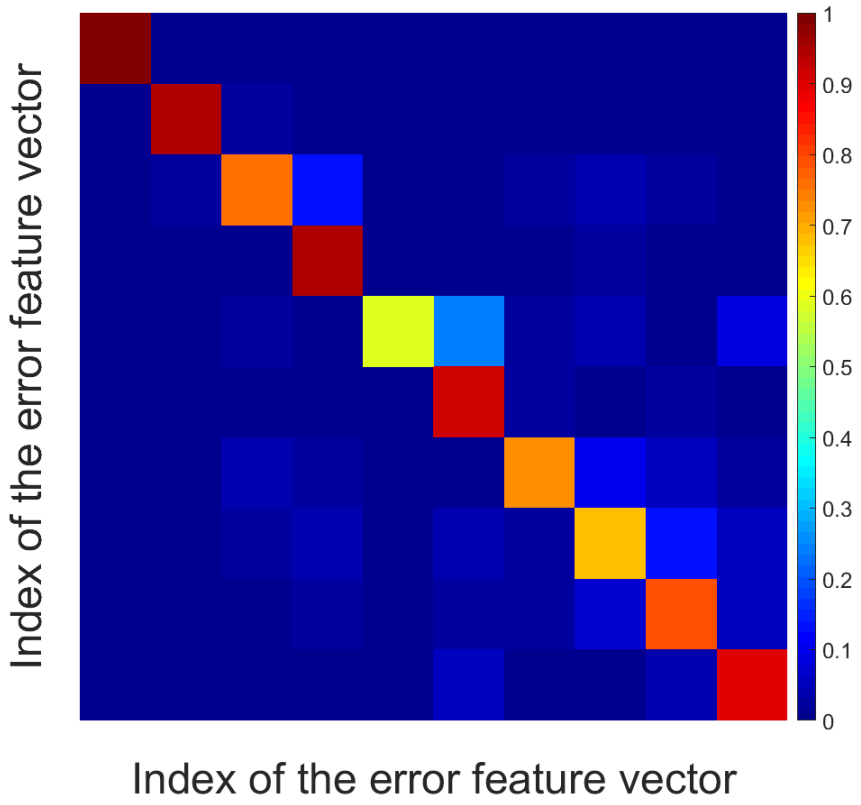


Figure 40: Values of $|\langle \boldsymbol{x}_i/\|\boldsymbol{x}_i\|, \boldsymbol{x}_j/\|\boldsymbol{x}_j\|\rangle|$ (color-coded) for data points labelled as errors.

were assessed. According to Figure 40, data points labelled as errors are largely orthogonal to each other apart from few modestly-sized groups.

The number of clusters, parameter p in step 5, was varied from 2 to 1408 in regular increments [171]. As a clustering algorithm, standard k-means routine (k-means ++) is supplied with MATLAB 2016a. For each value of p, k-means algorithm runs 10 times. For each clustering pass the corresponding nodes f_j (or their pairs for Algorithm 2) as prescribed in step 6 are constructed, and combined them into a single correcting ensemble in accordance with step 7.Before assessing performance of the ensemble, the filtering properties of the ensemble as a function of the number of clusters used were evaluated. For consistency with predictions provided in Theorems 4 – 6, Algorithm 1 was used in this exercise. Results of the test are shown in Figure 41. Note that as the number p of clusters increases, the True Positive Rate at $\gamma = 0$ as a function of p, approaches 1 regardless of the projection step 4. This is in agreement with theoretical predictions stemming from Theorem 5. It was also observed that performance drops rapidly with the average number of elements assigned to a cluster. In view of the earlier observation that vectors labelled as "errors" appear to be nearly orthogonal to each other, this drop is consistent with the bound provided in Corollary 1.

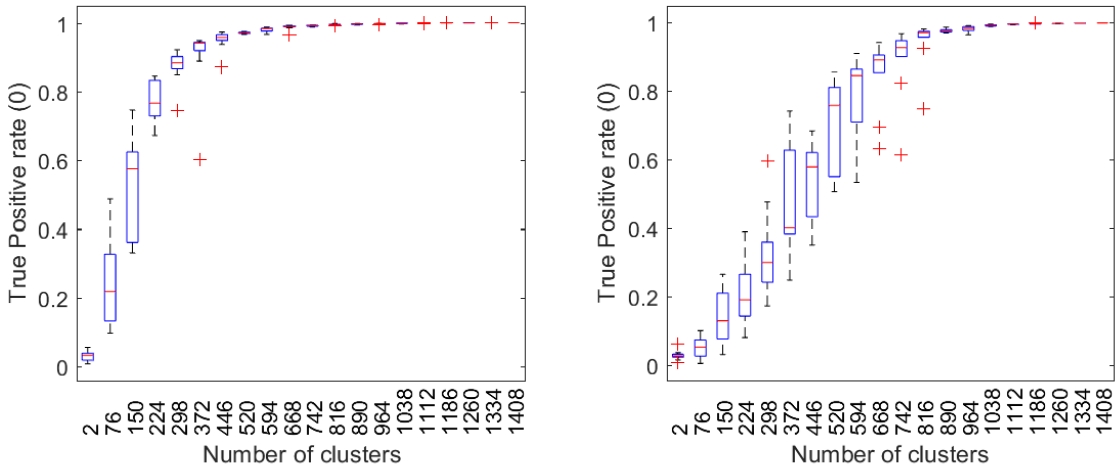


Figure 41: True Positive rate at $\gamma = 0$ for the combined system as a function of the number of clusters, p. Left panel corresponds to Algorithm 1 *with* the optional projection Step 4. Right panel corresponds to Algorithm 1 *without* the projection Step 4.

Next, the performance of Algorithms 1, 2 and resulting ensembles on the training and testing sets for $p = 76, 150, 224, 298$ are assessed. In both algorithms, optional projection step 4 was used. The value of threshold θ was set to 0.2. In general, the value of threshold θ can be selected arbitrarily in an interval of feasible values of c_j. When the optional projection step 4 is used, this interval is $(0, 1)$. Here the value of θ so that the hyperplanes $h_{j,2}$ in step 6 of Algorithm 2 produced by standard perceptron algorithm [97] that consistently yielded perfect separation are set. Results are shown in Figs. 42 and 43. According to Figure 42, performance on the training set grows with p, as expected. This is aligned with theoretical results presented in Section 9.1.1. The ensembles rapidly removes misclassification errors from the system, at some cost to True Positive Rate. The latter cost for Algorithm 2 appears to be negligible. On the testing set (see Figure 43), the picture changes. As the number p of clusters/nodes grows, performance of the ensemble saturates, with Algorithm 1 catching-up with Algorithm 2. This signals an over-fit and indicates a point where further increases in the number of nodes do not translate into expected improvements of the combined systems's performance. Lack of correlations between feature vectors corresponding to errors (illustrated with Figure 40) may also contribute to the observed performance saturation on the testing set.
For γ small both algorithms removed significant proportions of misclassification errors. This could be due to that training and testing sets for Algorithms 1 and 2 have been created from the performance analysis of the original core AI system at $\gamma = 0$.

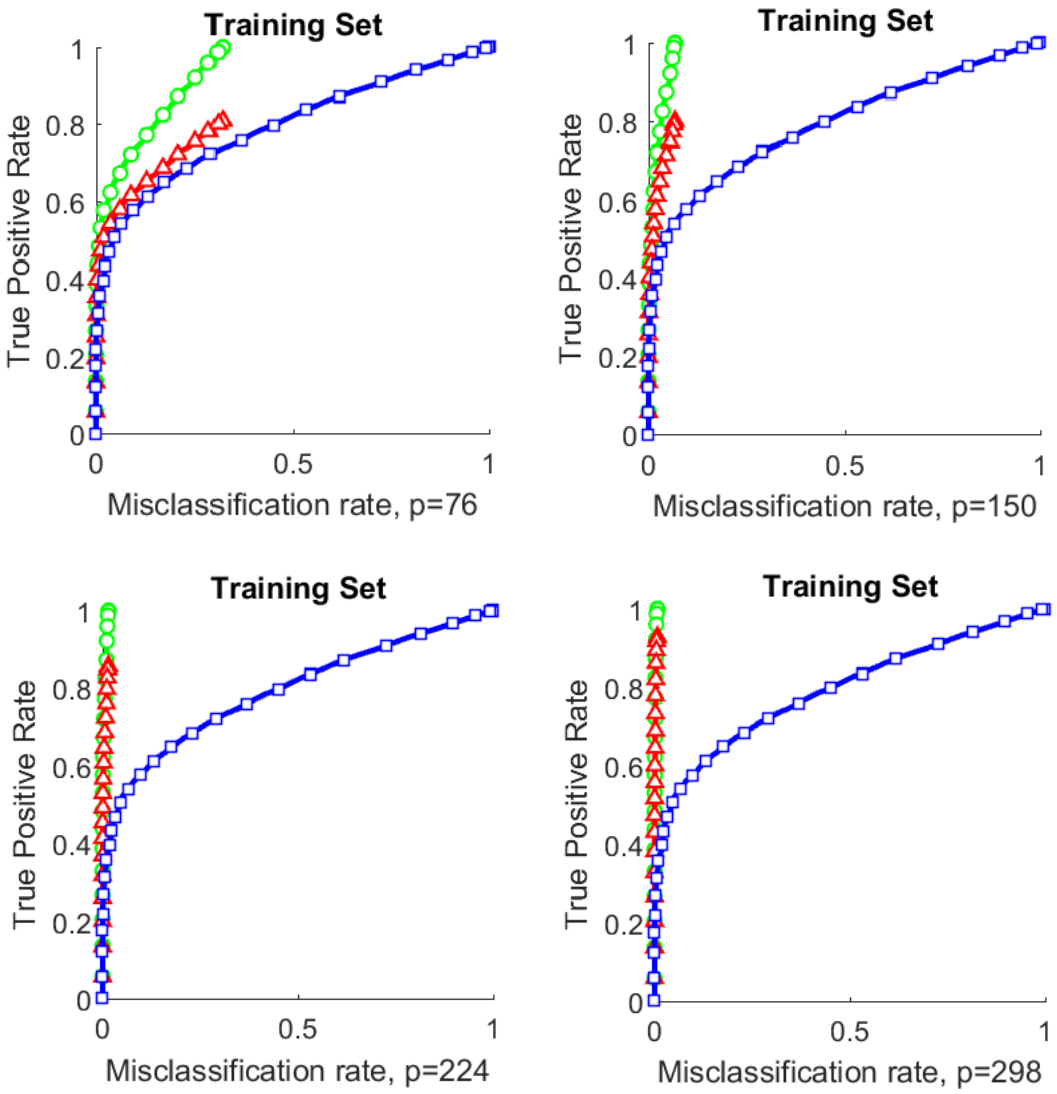


Figure 42: Combined system performance of the core AI (blue squares) after application of Algorithms 1 (red triangles) and 2 (green circles) on the training set for different numbers of clusters p in step 5. Elements in this set constructed the corrector.

9.3 Summary

This study presents a novel method for computationally efficient improvements of generic AI systems, including sophisticated Multilayer and Deep Learning neural networks. These improvements are easy-to-train ensembles of elementary nodes. After the ensembles are constructed, a possible next step could be to further optimise the system via error back-propagation. Mathematical operations at the nodes involve computations of inner products followed by standard nonlinear operations like ReLU, step functions, sigmoidal functions etc. The technology was illustrated with a simple case study confirming its viability. These proposed ensembles can also be employed for learning new tasks too.

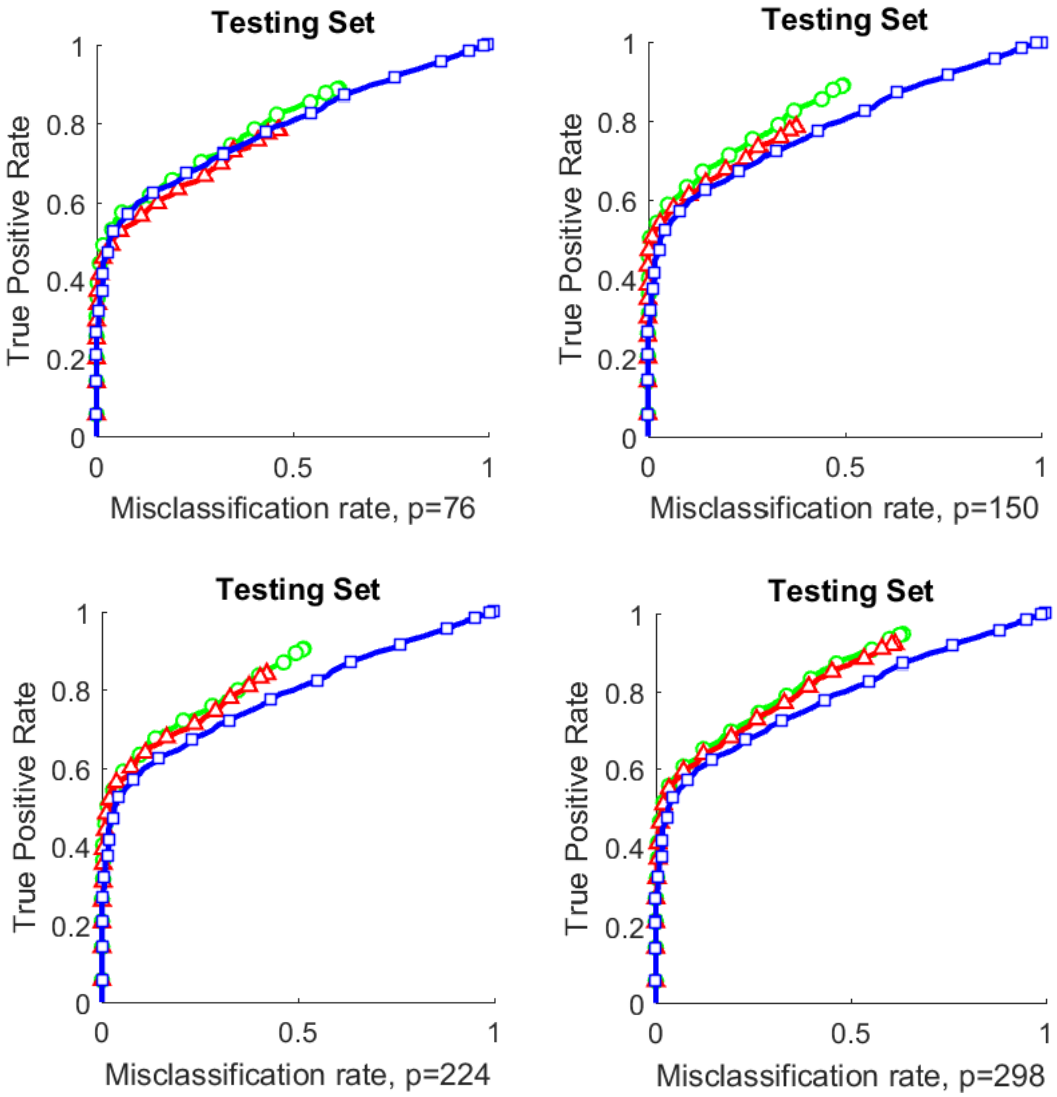


Figure 43: Combined system performance of the core AI (blue squares) after application of Algorithms 1 (red triangles) and 2 (green circles) on the testing set for different numbers of clusters p in step 5. No elements in this set constructed the corrector.

The proposed concept builds on previous work [3] and complements the results for equidistributions in a unit ball to product measure distributions. When the clustering structure is fixed, the method is inherently one-shot and non-iterative, and its computational complexity scales at most linearly with the size of the training set. Sub-linear computational complexity of the ensembles construction makes the technology particularly suitable for large AI systems that have already been deployed and operational.
An intriguing and interesting feature of the approach is that training of the ensembles is largely achieved via Fisher linear discriminants. This, combined with the ideas from [6], paves the way for a potential relaxation of the i.i.d. assuming sampled data.

9.4 Proofs Of Theorems And Other Technical Statements

Proof of Theorem 4

The proof follows immediately from Theorem 2 in [1] and Hoeffding inequality [172]. Indeed, if $t > 0$, X_i are independent bounded random variables, i.e. $a_i \leq X_i \leq b_i$, $i = 1, \ldots, n$, and $\overline{X} = 1/n \sum_{i=1}^{n} X_i$ then (Hoeffding inequality)

$$\begin{aligned} P\left(\overline{X} - E[\overline{X}] \geq t\right) &\leq \exp\left(-\frac{2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right), \\ P\left(|\overline{X} - E[\overline{X}]| \geq t\right) &\leq 2\exp\left(-\frac{2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right). \end{aligned} \tag{98}$$

Given that $x_{ij} = X_j$ where X_j are independent random variables with $-1 \leq X_j \leq 1$, $E[X_j^2] = \sigma_j^2$ (Assumption 1), it's observed that $\|\boldsymbol{x}_i\|^2 = \sum_{j=1}^{n} X_j^2$

$$\begin{aligned} &P\left(\left|\frac{\sum_{j=1}^{n} X_j^2}{n} - E\left[\frac{\sum_{j=1}^{n} X_j^2}{n}\right]\right| \geq t\right) = \\ &P\left(\left|\frac{\sum_{j=1}^{n} X_j^2}{n} - \frac{R_0^2}{n}\right| \geq t\right) = P\left(\left|\frac{\|\boldsymbol{x}_i\|^2}{R_0^2} - 1\right| \geq \frac{tn}{R_0^2}\right) \\ &\leq 2\exp\left(-2nt^2\right). \end{aligned} \tag{99}$$

Denoting $\varepsilon = tn/R_0^2$ and recalling that $0 \leq X_j^2 \leq 1$ this concludes that inequalities (61) holds true. Noticing that $E[x_{ik}x_{jk}] = E[x_{ik}]E[x_{jk}] = 0$, $E[y_k x_{ik}] = 0$, $-1 \leq x_{ik} \leq 1$, $-1 \leq x_{jk} \leq 1$, and $-1 \leq y_k x_{ik} \leq 1$, observe that estimates (62) and (63) follow. □

Proof of Theorem 5

Recall that for any events $A_1, \ldots, A_k$ the following estimate holds:

$$P(A_1 \& A_2 \& \cdots \& A_k) \geq 1 - \sum_{i=1}^{k}(1 - P(A_i)). \tag{100}$$

Let

$$\frac{\|\boldsymbol{x}_{M+1}\|^2}{R_0^2} \geq 1 - \varepsilon \ \ (\text{event } A_1), \tag{101}$$

and

$$\left\langle \frac{\boldsymbol{x}_{M+1}}{R_0}, \frac{\boldsymbol{x}_i}{R_0} \right\rangle < 1 - \varepsilon \ \ (\text{events } A_2, \ldots A_{M+1}). \tag{102}$$

The statement now follows from (61) and (62) with $\delta = 1 - \varepsilon$, and (100). □

Proof of Theorem 6

Suppose that $\|\boldsymbol{x}_i\|^2/R_0^2 \geq 1-\varepsilon$ for all $\boldsymbol{x}_i \in \mathcal{Y}$ (event A_1). Then

$$\ell_k(\boldsymbol{x}_i) = \frac{1}{k}\left(\frac{\|\boldsymbol{x}_i\|^2}{R_0^2} + \sum_{\boldsymbol{x}_j\in\mathcal{Y},\ \boldsymbol{x}_j\neq\boldsymbol{x}_i}\left\langle\frac{\boldsymbol{x}_i}{R_0},\frac{\boldsymbol{x}_j}{R_0}\right\rangle\right) \geq \frac{1-\varepsilon+\beta(k-1)}{k} \quad \text{for all } \boldsymbol{x}_i \in \mathcal{Y}. \tag{103}$$

Note that $\overline{\boldsymbol{x}} \in [-1,1]^n$ by construction. Consider events

$$\ell_k(\boldsymbol{x}_j) = \left\langle\frac{\boldsymbol{x}_j}{R_0},\frac{\overline{\boldsymbol{x}}}{R_0}\right\rangle < \frac{1-\varepsilon+\beta(k-1)}{k} \text{ (events } A_{1+j}), \tag{104}$$

$j = 1,\ldots,M$. According to Theorem 4 and (100)

$$\begin{aligned} P(A_1) &\geq 1 - k\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right), \\ P(A_{1+j}) &\geq 1 - \exp\left(-\frac{R_0^4(1-\varepsilon+\beta(k-1))^2}{2k^2n}\right). \end{aligned} \tag{105}$$

Hence

$$\begin{aligned} &P(A_1\&\cdots\&A_{1+M}) \geq \\ &1 - k\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right) - M\exp\left(-\frac{R_0^4(1-\varepsilon+\beta(k-1))^2}{2k^2n}\right). \end{aligned} \tag{106}$$

The result now follows $\square$.

Proof of Corollary 1

Let $\|\boldsymbol{x}_i\|^2/R_0^2 \geq 1-\varepsilon$ for all $\boldsymbol{x}_i \in \mathcal{Y}$ (event A_1), and

$$\left\langle\frac{\boldsymbol{x}_i}{R_0},\frac{\boldsymbol{x}_j}{R_0}\right\rangle \geq -\delta \text{ for all } \boldsymbol{x}_i,\boldsymbol{x}_j \in \mathcal{Y},\ \boldsymbol{x}_i \neq \boldsymbol{x}_j \text{ (event } A_2). \tag{107}$$

The corollary follows immediately from (100) and Theorem 4 (equations (61), (62)) $\square$.

Proof of Corollary 2

Let $\|\boldsymbol{x}_j\|^2 \geq 1-\varepsilon$. Then $h(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in \Omega$. Estimate (74) hence follows from Theorem 4 and (100). $\square$.

Proof of Lemma 1

The proof follows that of Theorem 3 in [3]. Let $\boldsymbol{x}$ be an element of $\mathcal{M}$ and p_h be the probability of the event $h(\boldsymbol{x}) \geq 0$. Then the probability of the event that $h(\boldsymbol{x}) \geq 0$ for at most m elements from $\mathcal{M}$ is

$$P(M,m) = \sum_{k=0}^{m-1} \left(\begin{array}{c} M \\ k \end{array} \right) (1-p_h)^{M-k} p_h^k. \tag{108}$$

Observe that $P(M,m)$, as a function of p_h, is monotone and non-increasing on the interval $[0,1]$, with $P(M,m) = 0$ at $p_h = 1$ and $P(M,n) = 1$ at $p_h = 0$. Hence

$$P(M,m) \geq \sum_{k=0}^{m-1} \left(\begin{array}{c} M \\ k \end{array} \right) (1-p_*)^{M-k} p_*^k = (1-p_*)^M \sum_{k=0}^{m-1} \left(\begin{array}{c} M \\ k \end{array} \right) \left(\frac{p_*}{1-p_*} \right)^k. \tag{109}$$

Given that $\frac{(M-m+1)^k}{k!} \leq \left(\begin{array}{c} M \\ k \end{array} \right) \leq \frac{M^k}{k!}$ for $0 \leq k \leq m-1$, this implies:

$$P(M,m) \geq (1-p_*)^M \sum_{k=0}^{m-1} \frac{1}{k!} \left(\frac{(M-m+1)p_*}{1-p_*} \right)^k. \tag{110}$$

Expanding e^x at $x = 0$ with the Lagrange remainder term:

$$e^x = \sum_{k=0}^{m-1} \frac{x^k}{k!} + \frac{x^m}{m!} e^{\xi}, \ \xi \in [0,x], \tag{111}$$

results in the estimate

$$\sum_{k=0}^{m-1} \frac{x^k}{k!} \geq e^x \left(1 - \frac{x^m}{m!} \right) \text{ for all } x \geq 0. \tag{112}$$

Hence

$$\begin{array}{l} P(M,m) \geq \\ (1-p_*)^M \exp\left(\frac{(M-m+1)p_*}{1-p_*} \right) \left(1 - \frac{1}{m!} \left(\frac{(M-m+1)p_*}{(1-p_*)} \right)^m \right). \end{array} \tag{113}$$

□

9.5 Hyperplanes Vs Ellipsoids For Error Isolation

Consider

$$C_n(\varepsilon) = B_n(1) \cap \left\{ \xi \in \mathbb{R}^n \;\middle|\; \left\langle \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}, \xi \right\rangle \geq 1 - \varepsilon \right\}. \tag{114}$$

Let

$$\rho(\varepsilon) = (1 - (1-\varepsilon)^2)^{\frac{1}{2}}. \tag{115}$$

Note that $\rho(\varepsilon)$ is the radius of the ball containing the spherical cap C_n. Lemma 2 estimates volumes of spherical caps $C_n(\varepsilon)$ relative to relevant n-balls of radius $\rho(\varepsilon)$.

Lemma 2 *Let $C_n(\varepsilon)$ be a spherical cap defined as in (114), $\varepsilon \in (0,1)$. Then*

$$\frac{\rho(\varepsilon)^{n+1}}{2}\left[\frac{1}{\pi^{\frac{1}{2}}}\frac{\Gamma\left(\frac{n}{2}+1\right)}{\Gamma\left(\frac{n}{2}+\frac{3}{2}\right)}\right] < \frac{\mathcal{V}(C_n(\varepsilon))}{\mathcal{V}(B_n(1))} \leq \frac{\rho(\varepsilon)^n}{2}. \tag{116}$$

Proof of Lemma 2

Note that [173] $\mathcal{V}(B_n(r)) = r^n \mathcal{V}(B_n(1))$ for all $n \in \mathbb{N}$ $r > 0$. Hence the estimate of $\mathcal{V}(C_n(\varepsilon))$ from above is:

$$\mathcal{V}(C_n(\varepsilon)) \leq \frac{1}{2}\mathcal{V}(B_n(1))\rho(\varepsilon)^n. \tag{117}$$

Calculate the estimate of $\mathcal{V}(C_n(\varepsilon))$ from below. It is clear that

$$\mathcal{V}(C_n(\varepsilon)) = V(B_{n-1}(1)) \int_{1-\varepsilon}^{1} (1-x^2)^{\frac{n-1}{2}} dx. \tag{118}$$

The integral in the right-hand side of (118) can be estimated from below as

$$\begin{aligned} &\int_{1-\varepsilon}^{1} (1-x^2)^{\frac{n-1}{2}} dx > \int_{1-\varepsilon}^{1} (1-x^2)^{\frac{n-1}{2}} x dx \\ &= \frac{1}{2} \cdot \frac{1}{\frac{n}{2}+\frac{1}{2}} \cdot (1-(1-\varepsilon)^2)^{\frac{n+1}{2}} = \frac{1}{2} \cdot \frac{1}{\frac{n}{2}+\frac{1}{2}} \cdot \rho(\varepsilon)^{n+1} \end{aligned} \tag{119}$$

Recall that $B_n(1) = \frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)}$, $\Gamma(x+1) = x\Gamma(x)$. Hence

$$B_{n-1}(1) \cdot \frac{1}{\frac{n}{2}+\frac{1}{2}} = \frac{\pi^{\frac{n-1}{2}}}{\Gamma\left(\frac{n}{2}+\frac{1}{2}\right)} \frac{1}{\frac{n}{2}+\frac{1}{2}} = \frac{\pi^{\frac{n-1}{2}}}{\Gamma\left(\frac{n+1}{2}+1\right)}, \tag{120}$$

and

$$\int_{1-\varepsilon}^{1} (1-x^2)^{\frac{n-1}{2}} dx > \frac{1}{2}\mathcal{V}(B_n(1))\rho(\varepsilon)^{n+1}\left[\frac{1}{\pi^{\frac{1}{2}}}\frac{\Gamma\left(\frac{n}{2}+1\right)}{\Gamma\left(\frac{n}{2}+\frac{3}{2}\right)}\right] \tag{121}$$

□

Corollary 3 *Let $C_n(\varepsilon)$ be a spherical cap defined as in (114), $\varepsilon \in (0,1)$, and $B_n(k\rho(\varepsilon))$ be an n-ball with radius $k\rho(\varepsilon)$, $k \in \mathbb{R}_{>0}$. Then*

$$\frac{\mathcal{V}(B_n(k\rho(\varepsilon)))}{\mathcal{V}(C_n(\varepsilon))} < k^n \frac{2\pi^{\frac{1}{2}}}{\rho(\varepsilon)} \left[\frac{\Gamma\left(\frac{n}{2}+1\right)}{\Gamma\left(\frac{n}{2}+\frac{3}{2}\right)}\right]^{-1}. \tag{122}$$

Remark 3 *Using Stirling's approximation:*

$$\frac{\Gamma\left(\frac{n}{2}+1\right)}{\Gamma\left(\frac{n}{2}+\frac{3}{2}\right)} = O\left(n^{-\frac{1}{2}}\right). \tag{123}$$

Thus $\frac{\mathcal{V}(B_n(\varepsilon))}{\mathcal{V}(C_n(\varepsilon))} < k^n H(n,\varepsilon)$ *where* $H(n,\varepsilon) = O\left(\frac{n^{1/2}}{\rho(\varepsilon)}\right)$.

According to Corollary 3 the volumes of $B_n(\varepsilon)$, $B_n(\kappa\rho(\varepsilon))$, $\kappa \in (0,1)$ decay exponentially with dimension n relative to that of $C_n(\varepsilon)$. This implies that distance-based detectors are extremely localised. In comparison with simple perceptrons, the proportion of points to which they respond positively is negligibly small. However, filtering properties of hyperplanes are extreme in high dimension (see Theorems 5, 6 in Section 9.1.1). This combination of properties makes perceptrons and their ensembles particularly attractive for fine-tuning of existing AI systems.

10 Nonlinear Stochastic Separation Theorems

10.1 Kernel Stochastic Separation Theorems

10.1.1 Preliminaries and Problem Formulation

Let $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ be a set of vectors in $\mathbb{R}^n$ sampled from some distribution with a corresponding probability density function p. Each element of the sample is subjected to a transformation

$$\Phi : \mathbb{R}^n \to \mathbb{H}, \tag{124}$$

called a *feature map*, mapping $\boldsymbol{x}_i \in \mathbb{R}^n$ into $\Phi(\boldsymbol{x}_i)$ in some Hilbert space $\mathbb{H}$. This assumes that the feature map Φ is known. The process induces a new random variable, $\Phi(\boldsymbol{x})$, and a corresponding distribution. This also assumes that the feature map Φ is such that

$$E[\Phi] = \int \Phi(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} \tag{125}$$

exists. For the moment $E[\Phi] = 0$ is assumed, and this technical assumption is explained later in Section 10.1.4. For the given feature map $\Phi(\boldsymbol{x})$, a kernel function κ is defined as

follows

$$\kappa : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R},\ \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j)), \tag{126}$$

and a kernel classifier is the function

$$f : \mathbb{R}^n \to \mathbb{R},\ f(\boldsymbol{x}) = \sum_{j=1}^{m} \alpha_j \kappa(\boldsymbol{y}_j, \boldsymbol{x}) - b, \tag{127}$$

where $\alpha_j, b \in \mathbb{R}$, and $\boldsymbol{y}_j \in \mathbb{R}^n$. In the simplest binary classification setting, the classifier assigns positive values to elements $\boldsymbol{x} \in \mathbb{R}^n$ from the set corresponding to Class 1 and negative values to elements from set to Class 2.

Definition 2 *A point $\boldsymbol{x} \in \mathbb{R}^n$ is kernel separable with the kernel function (126) from a set $\mathcal{Y} \subset \mathbb{R}^n$, if there exist $m \in \mathbb{N}$, α_j, $\boldsymbol{y}_j \in \mathbb{R}^n$, $j \in \{1, \ldots, m\}$ such that*

$$\sum_{j=1}^{m} \alpha_j \kappa(\boldsymbol{y}_j, \boldsymbol{x}) > \sum_{j=1}^{m} \alpha_j \kappa(\boldsymbol{y}_j, \boldsymbol{y}) \tag{128}$$

for all $\boldsymbol{y} \in \mathcal{Y}$.

Definition 3 *A set $\mathcal{X} \subset \mathbb{R}^n$ is kernel separable with the kernel function (126) from a set $\mathcal{Y} \subset \mathbb{R}^n$, if there exist $m \in \mathbb{N}$, α_j, $\boldsymbol{y}_j \in \mathbb{R}^n$, $j \in \{1, \ldots, m\}$ such that*

$$\sum_{j=1}^{m} \alpha_j \kappa(\boldsymbol{y}_j, \boldsymbol{x}) > \sum_{j=1}^{m} \alpha_j \kappa(\boldsymbol{y}_j, \boldsymbol{y}) \tag{129}$$

for all $\boldsymbol{y} \in \mathcal{Y}$ and $\boldsymbol{x} \in \mathcal{X}$.

Definition 4 *A set $S \subset \mathbb{R}^n$ is kernel separable with the kernel function (126) if for each $\boldsymbol{x} \in S$ there exist $m \in \mathbb{N}$, α_j, $\boldsymbol{y}_j \in \mathbb{R}^n$, $j \in \{1, \ldots, m\}$ such that*

$$\sum_{j=1}^{m} \alpha_j \kappa(\boldsymbol{y}_j, \boldsymbol{x}) > \sum_{j=1}^{m} \alpha_j \kappa(\boldsymbol{y}_j, \boldsymbol{y}) \tag{130}$$

for all $\boldsymbol{y} \in S$, $\boldsymbol{y} \neq \boldsymbol{x}$.

In addition to the notions of kernel separability, and similar to [6], the notion of Fisher separability to kernel classifiers is utilised.

Definition 5 *A point $\boldsymbol{x} \in \mathbb{R}^n$ is Fisher separable with the kernel function (126) from a set $\mathcal{Y} \subset \mathbb{R}^n$, if*

$$\kappa(\boldsymbol{x}, \boldsymbol{x}) > \kappa(\boldsymbol{x}, \boldsymbol{y}) \tag{131}$$

for all $\boldsymbol{y} \in \mathcal{Y}$. *A set* $S \subset \mathbb{R}^n$ *is Fisher separable with the kernel function (126) if, for each* $\boldsymbol{x} \in S$, *(131) holds for all* $\boldsymbol{y} \in S$, $\boldsymbol{y} \neq \boldsymbol{x}$.

It's clear that Fisher separability with a given kernel function automatically implies kernel separability and as such is a stronger property. Note also that the notion of Fisher separability can be further extended to the notion of Fisher separability with a threshold $\gamma \in [0, 1)$, see [6], by replacing (131) with

$$\gamma\kappa(\boldsymbol{x}, \boldsymbol{x}) > \kappa(\boldsymbol{x}, \boldsymbol{y}). \tag{132}$$

This generalization allows to meaningfully pose the separability problem for kernels like $\kappa(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}/\|\boldsymbol{x}\|, \boldsymbol{y}/\|\boldsymbol{y}\|)$ for which the first part of Definition 5 always holds true.

10.1.2 Kernel Separability Of Points

The first result is provided in Theorem 7

Theorem 7 *Let* $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M \in \mathbb{R}^n$ *in* $B_n(1)$ *be given, and let* $\boldsymbol{x}$ *be drawn from a distribution with the probability density function* $p(\boldsymbol{x}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M)$ *supported inside* $\kappa(x, x) \leq 1$. *Then* $\boldsymbol{x}$ *is Fisher separable with the kernel function (126) from the set* $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\}$ *with probability at least*

$$1 - \sum_{i=1}^{M} \int_{\gamma\kappa(\boldsymbol{x},\boldsymbol{x}) - \kappa(\boldsymbol{x},\boldsymbol{y}_i) \leq 0} p(\boldsymbol{x}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M) d\boldsymbol{x}. \tag{133}$$

Proof of Theorem 7. Consider events

$$A_i : \ \boldsymbol{x} \text{ is Fisher separable from } \boldsymbol{y}_i. \tag{134}$$

It is clear that

$$P(\text{not } A_i) = \int_{\gamma\kappa(\boldsymbol{x},\boldsymbol{x}) - \kappa(\boldsymbol{x},\boldsymbol{y}_i) \leq 0} p(\boldsymbol{x}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M) d\boldsymbol{x}. \tag{135}$$

Recall that

$$P(A_1 \& A_2 \& \cdots \& A_M) \geq 1 - \sum_{i=1}^{M} P(\text{not } A_i). \tag{136}$$

Combining the last two observations, the probability that $\boldsymbol{x}$ is separable from all $\boldsymbol{y}_i$ is bounded from below by the expression in (133). □

Corollary 4 *Suppose that assumptions of Theorem 7 hold. Let us further assume that*

there is a $\lambda \in (0,1)$*, an* $L \in \mathbb{R}_{>0}$*, and a function* $\alpha : \mathbb{N} \to \mathbb{R}$ *such that*

$$\int_{\gamma\kappa(\boldsymbol{x},\boldsymbol{x})-\kappa(\boldsymbol{x},\boldsymbol{y})\leq 0} p(\boldsymbol{x}|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_M)d\boldsymbol{x} \leq L\lambda^{\alpha(n)} \tag{137}$$

for all $\boldsymbol{y} \in \mathbb{R}^n$ *in* $B_n(1)$*. Then* $\boldsymbol{x}$ *is Fisher separable with the kernel function (126) from the set* $\mathcal{Y} = \{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_M\}$ *with probability at least*

$$1 - ML\lambda^{\alpha(n)}. \tag{138}$$

It has been shown in [6] that for the feature maps $\Phi(\boldsymbol{x}) = \boldsymbol{x}$ and $p(\cdot|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_M)$ defined on $\mathbb{B}_n$ and bounded from above by $L/V_n(\mathbb{B}_n)$, condition (137) holds with

$$\lambda = \frac{1}{2},\ \alpha(n) = n. \tag{139}$$

Corollary 5 *Consider the set* $S = \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_M\}$ *in which* $\boldsymbol{x}_i \in \mathbb{R}^n$*,* $i = 1,\ldots,M$ *are random i.i.d. vectors. Let* $p : \mathbb{R}^n \to \mathbb{R}$ *be the corresponding probability density function, and let there exist* $\lambda \in (0,1)$*,* $L \in \mathbb{R}_{>0}$*, and a function* $\alpha : \mathbb{N} \to \mathbb{R}$ *such that*

$$\int_{\gamma\kappa(\boldsymbol{x},\boldsymbol{x})-\kappa(\boldsymbol{x},\boldsymbol{y})\leq 0} p(\boldsymbol{x})d\boldsymbol{x} \leq L\lambda^{\alpha(n)} \tag{140}$$

for all $\boldsymbol{y} \in \mathbb{R}^n$ *in* $B_n(1)$*. Then the set* S *is Fisher separable with the kernel function (126) with probability at least*

$$1 - (M-1)ML\lambda^{\alpha(n)}. \tag{141}$$

Theorem 7 and Corollaries 4, 5 extend stochastic separation theorems to kernel classifiers. Note that dimensionality N of the space to which the feature map, $\Phi(\cdot)$, maps original data points $\boldsymbol{x}$ need not be finite. According to these results, the probabilities of kernel separability in these cases can still be estimated via integration in finite dimensional spaces using e.g. (137), (140). Since Theorem 7 and Corollaries 4 concern Fisher separability, these bounds apply to kernel separability too (see Definitions 2 – 4). However, all statements derived so far could be generalised to cases when the functions κ themselves are not kernels. An interesting question is if these results can be extended to characterise separability of two random sets. The above formalism can be generalised to answer this question too.

10.1.3 Kernel Separability Of Two Random Sets

Consider two random sets $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ and $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K\}$. Let there be a learning algorithm which, for the given $\mathcal{X}$, $\mathcal{Y}$ or their subsets, produces a function

$$f(\cdot) = \sum_{i=1}^{d} \alpha_i \kappa(\boldsymbol{z}_i, \cdot), \ \alpha_j \in \mathbb{R}. \tag{142}$$

The vectors $\boldsymbol{z}_i$, $i = 1, \ldots, d$ are supposed to be known. Furthermore, suppose that the function f is such that

$$f(\boldsymbol{y}_j) > \sum_{m,k=1}^{d} \alpha_m \alpha_k \kappa(\boldsymbol{z}_m, \boldsymbol{z}_k) \tag{143}$$

for all $\boldsymbol{y}_j \in \mathcal{Y}$. In other words, denoting $\boldsymbol{w} = \sum_{i=1}^{d} \alpha_i \Phi(\boldsymbol{z}_i)$, the following holds true:

$$(\boldsymbol{w}, \boldsymbol{w}) < (\boldsymbol{w}, \Phi(\boldsymbol{y}_i)) \text{ for all } i = 1, \ldots, K. \tag{144}$$

Note that since the $\mathcal{Y}, \mathcal{X}$ are random, it is natural to expect that the vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$ is also random. The following statement can now be formulated:

Theorem 8 *Consider sets $\mathcal{X}$ and $\mathcal{Y}$. Let $p_\alpha(\boldsymbol{\alpha})$ be the probability density function associated with the random vector $\boldsymbol{\alpha}$, and $\boldsymbol{\alpha}$ satisfies condition (143) with probability 1. Then the set $\mathcal{X}$ is kernel separable with the kernel function (126) from the set $\mathcal{Y}$ with probability at least*

$$1 - \sum_{i=1}^{M} \int_{H(\boldsymbol{\alpha}, \boldsymbol{x}_i) \leq 0} p_\alpha(\boldsymbol{\alpha}) d\boldsymbol{\alpha}, \tag{145}$$

where

$$H(\boldsymbol{\alpha}, \boldsymbol{x}_i) = \sum_{k,m=1}^{d} \alpha_k \alpha_m \kappa(\boldsymbol{z}_k, \boldsymbol{z}_m) - \sum_{m=1}^{d} \alpha_m \kappa(\boldsymbol{z}_m, \boldsymbol{x}_i). \tag{146}$$

Proof of Theorem 8. The proof of the theorem is similar to that of Theorem 7. Consider events

$$A_i : \ (\boldsymbol{w}, \boldsymbol{w}) > (\boldsymbol{w}, \Phi(\boldsymbol{x}_i)). \tag{147}$$

Events A_i are equivalent to that $H(\boldsymbol{\alpha}, \boldsymbol{x}_i) > 0$. Eq. (145) provides a lower bound for the probability that all these events hold. Recall that vectors $\boldsymbol{\alpha}$ satisfy (144), and hence:

$$\begin{aligned} \sum_{m=1}^{d} \alpha_m \kappa(\boldsymbol{z}_m, \boldsymbol{x}_i) =& (\boldsymbol{w}, \Phi(\boldsymbol{x}_i)) \\ &< (\boldsymbol{w}, \Phi(\boldsymbol{y}_j)) = \sum_{m=1}^{d} \alpha_m \kappa(\boldsymbol{z}_m, \boldsymbol{y}_j) \end{aligned} \tag{148}$$

for all $\boldsymbol{x}_i \in \mathcal{X}$ and $\boldsymbol{y}_j \in \mathcal{Y}$ with probability at least (145). The statement now follows immediately from Definition 3. $\square$

Theorem 8 generalises earlier k-tuple separation theorems [4], [116] in that it applies to a much broader class of classifiers and isn't limited to a particular set of distributions. Similar to Corollaries 4, 5, this establishes conditions linking dimensionality of the vector $\boldsymbol{\alpha}$ with the probability of separation. An example of such condition could be existence of $L > 0$, $\lambda \in (0, 1)$ and a function $\beta : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ such that

$$\int_{H(\boldsymbol{\alpha},\boldsymbol{y})\leq 0} p_{\alpha}(\boldsymbol{\alpha}) d\boldsymbol{\alpha} \leq L\lambda^{\beta(d,n)} \tag{149}$$

for any $\boldsymbol{y} \in \mathbb{R}^n$.

10.1.4 Kernels with $E[\Phi(\boldsymbol{x})] \neq 0$

Theorems 7, 8, their corollaries, and Fisher separability notions in Definition 5 have been produced under the simplifying assumption that $E[\Phi(\boldsymbol{x})] = 0$. These statements can be surmised to more general settings with

$$E[\Phi(\boldsymbol{x})] = \overline{\Phi}. \tag{150}$$

This generalisation can be achieved by replacing kernel functions $\kappa(\boldsymbol{x}, \boldsymbol{y}) = (\Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}))$ with

$$\begin{aligned} \tilde{\kappa}(\boldsymbol{x}, \boldsymbol{y}) = (\Phi(\boldsymbol{x}) - \overline{\Phi}, \Phi(\boldsymbol{y}) - \overline{\Phi}) = \kappa(\boldsymbol{x}, \boldsymbol{y}) \\ - \int p(\boldsymbol{x})\kappa(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} - \int p(\boldsymbol{y})\kappa(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} \\ + \int \int p(\boldsymbol{x})p(\boldsymbol{y})\kappa(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y} \end{aligned} \tag{151}$$

in relevant expressions. In practice, $E[\Phi(\boldsymbol{x})]$ can be replaced with the sample mean over a finite sample $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ leading to the following approximations of $\tilde{\kappa}(\boldsymbol{x}, \boldsymbol{y})$:

$$\begin{aligned} \tilde{\kappa}(\boldsymbol{x}, \boldsymbol{y}) = \kappa(\boldsymbol{x}, \boldsymbol{y}) \\ - \frac{1}{N} \sum_{i=1}^{M} (\kappa(\boldsymbol{x}_i, \boldsymbol{y}) + \kappa(\boldsymbol{x}, \boldsymbol{x}_i)) + \frac{1}{N^2} \sum_{i,j=1}^{M} \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j). \end{aligned} \tag{152}$$

10.1.5 Kernel Separability Measure

One of the outcomes of the theoretical results is the kernel separability characterisation expressed e.g. in terms of the upper bound $L\lambda^{\alpha(n)}$ on the integral

$$\int_{\tilde{\kappa}(\boldsymbol{x},\boldsymbol{x})-\tilde{\kappa}(\boldsymbol{x},\boldsymbol{y})\leq 0} p(\boldsymbol{x})d\boldsymbol{x}, \tag{153}$$

in the left-hand side of (140). The bound determines how well a kernel $\kappa(\cdot,\cdot)$ separates points in samples from a given distribution. The smaller the value of λ, and the faster the function $\alpha(\cdot)$ grows with n, the better the kernel's separability is (as per Definition 5). Direct derivation of λ, $\alpha(\cdot)$, requires knowledge or at least bounds on the probability density functions p, p_α. In practice, probability density functions are rarely known. However, it is not unrealistic to expect that some data sample $\mathcal{X}$ from the distribution might be available. In this case, then evaluation of the above integral may be replaced with:

$$\omega(\boldsymbol{y},n)=\frac{|\{\boldsymbol{x}\in\mathcal{X}\subset\mathbb{R}^n\ |\boldsymbol{x}\neq\boldsymbol{y},\ \tilde{\kappa}(\boldsymbol{x},\boldsymbol{x})-\tilde{\kappa}(\boldsymbol{x},\boldsymbol{y})\leq 0\}|}{|\mathcal{X}|}. \tag{154}$$

The empirical kernel separability measure, can then be defined as the average

$$\Omega_a(n)=\frac{\sum_{i=1}^{|\mathcal{X}|}\omega(\boldsymbol{x}_i,n)}{|\mathcal{X}|}, \tag{155}$$

or

$$\Omega_{\max}(n)=\max_{\boldsymbol{x}_i\in\mathcal{X}}\omega(\boldsymbol{x}_i,n). \tag{156}$$

The next section establishes how these measures can characterise and compare different kernel functions with respect to their ability to separate points in random sets.

10.2 Examples

10.2.1 Polynomial Kernels For An Equidistribution In The $[-1,1]^n$ Cube

In the first group of experiments, behavior of polynomial kernels in a synthetic test in which the original data samples are generated from equidistributions in the unit cubes $[-1,1]^n$ of varying dimension n is observed. The kernel functions were chosen as follows:

$$\kappa(\boldsymbol{x},\boldsymbol{y})=((\boldsymbol{x},\boldsymbol{y})+1)^p, \tag{157}$$

where parameter p took values in the set $\{1,2,3\}$. Observe that for $p=1$ the centralised kernel $\tilde{\kappa}$ reduces to standard inner product $(\boldsymbol{x},\boldsymbol{y})$, and for quadratic kernels, $p=2$, the

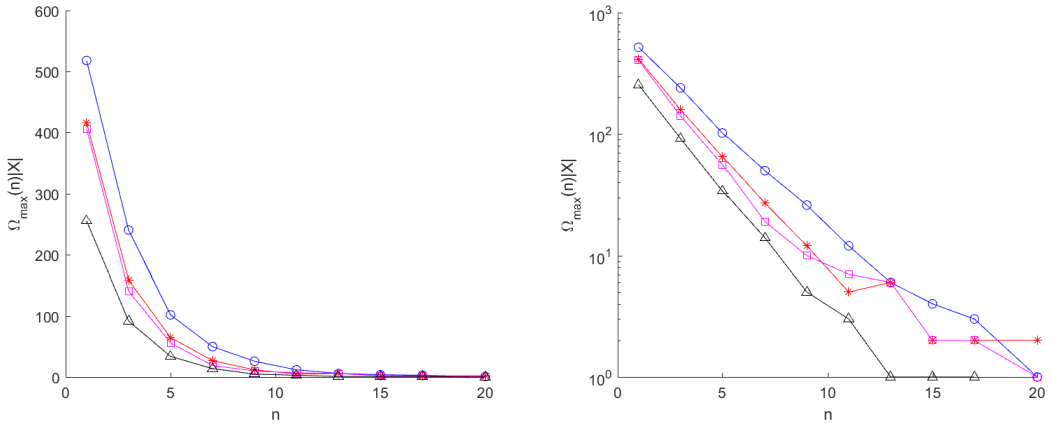


Figure 44: Separability characterisations for polynomial kernels with $p = 1, 2, 3$. Blue circled line corresponds to $p = 1$, red starred line corresponds to $p = 2$, and magenta squares show performance of the chosen polynomial kernel with $p = 3$. Black line with triangle markers corresponds to Fisher discriminants built over quadratic feature maps followed by centralisation and whitening. The left panel shows original data, and right panel shows the same data but in the logarithmic scale.

feature map is:

$$\Phi(\boldsymbol{x}) = (x_1^2, \ldots, x_n^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_{n-1}x_n, \sqrt{2}x_1, \ldots, \sqrt{2}x_n, 1). \tag{158}$$

For each given value of n a sample of $M = 1000$ vectors were generated and calculated the value of $\Omega_{\max}(n)$ (see Eq. (156)) from a sub-sample of 1000 points chosen randomly from this sample. The outcomes of this process for different values of p are summarised in Figure 44. According to Figure 44, separability of quadratic kernels is higher than the original Fisher discriminants. This isn't surprising given that the dimensionality of the quadratic feature map, $\frac{n(n+1)}{2} + n + 1$, is significantly higher than that of the original space, n. Unexpectedly, these experiments reveal that re-weighting of features, e.g. via the whitening transformation, may produce higher performance gains than choosing a kernel with a higher-dimensional feature map (black triangled lines in Figure 44 vs squared magenta lines). Such re-weighting generates a different inner product and hence corresponds to a kernel function that is different from the ones specified in (157).

10.2.2 Kernels with Polynomial Feature Maps For Inception Bottlenecks

This example investigated and tested kernel separability in the setting when the original features are bottlenecks of a pre-trained convolutional neural network. In particular, a network trained to distinguish ten digits in American Sign Language. The network was an Inception deep neural network model [170] whose architecture and training process

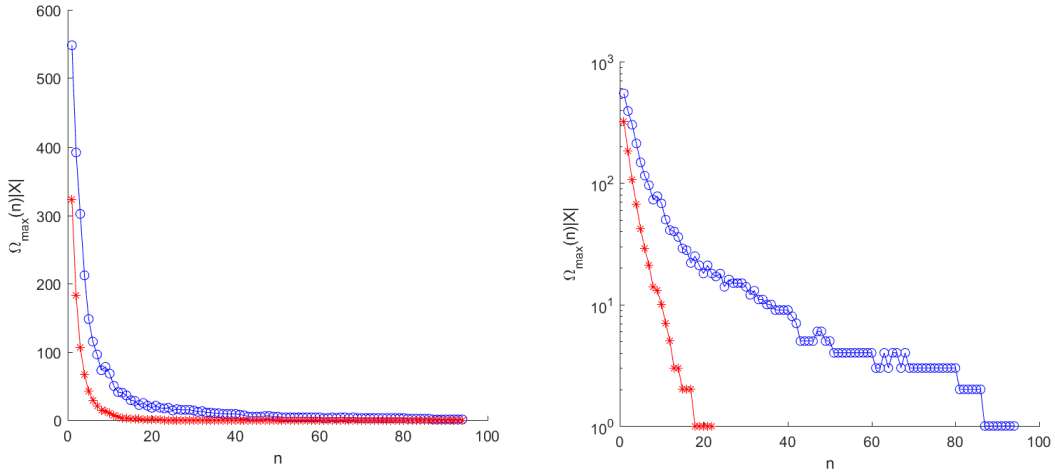


Figure 45: Separability of the feature vectors sampled from Inception bottlenecks. Blue circled lines show $\Omega_{\max}(n)|\mathcal{X}|$ as a function of the number n of the principal components retained. Red starred line shows $\Omega_{\max}(n)|\mathcal{X}|$ for kernels with quadratic, centered, and whitened feature map (158). Left panel shows original data, and right panel shows the same data but in the logarithmic scale.

has been described in detail in [116]. The model was trained [174] on ten sets of images corresponding to the American Sign Language pictures for 0-9. Each set contained 1000 unique images consisting of profile shots of the person's hand, along with 3/4 profiles and shots from above and below. The states $\boldsymbol{x}_i$ are the vectors containing the values of pre-softmax layer bottlenecks of size n for however many neurons are in the penultimate layer. It was shown in [116] that these bottlenecks can be used to construct error correcting cascades for such systems, and using stochastic separation theorems, such cascades can be derived using mere Fisher discriminants. The higher is the dimension of the bottlenecks $\boldsymbol{x}_i$ the larger is the probability that the error correcting cascades are successful. It is therefore interesting to see if employing kernels in place of the original linear classifiers could potentially improve error correction performance. To assess this, original bottle necks $\boldsymbol{x}_i$ were projected on n principal components, and constructed higher-dimensional representations of the projected data using quadratic kernel feature map (158). This was followed by centering and whitening transformation. For the new feature vectors defined in this way, $\Omega_{\max}(n)$ for n ranges from 1 to 94. These results are shown in Figure 45, which suggests that using kernels induced by quadratic feature maps may offer superior point separability properties as compared to the original feature vectors.

10.3 Summary

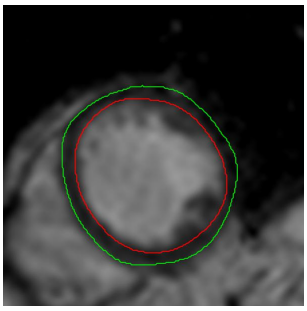
This study an extension of the framework of stochastic separation theorems to arbitrary kernel classifiers. Separability criteria emerges from these generalisations to reduce to finite-dimensional volume integrals, despite the fact that the feature maps corresponding to relevant kernels may be infinite-dimensional. Furthermore, a general separability result for two random sets was constructed. The latter result assumes some prior knowledge of the distribution of the weights of the classifier. These results produced empirical kernel separability characterisations for arbitrary kernel functions. The application of these new characterisations has been illustrated with two case study examples. These examples showed that if an additional whitening and re-weighting of features are allowed, then the point separability performance of the induced kernel may be drastically improved. Further investigations generalisation capabilities of such kernel classifiers and their derivatives for the problem of AI error correction [4], [116] can be the subject of future work.

Part V

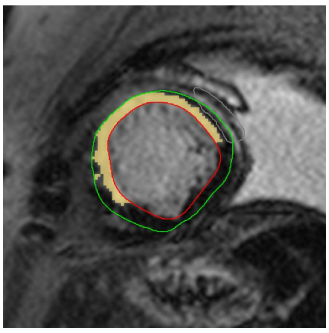
Further Applications Of Research And Conclusions

11 Further Applications

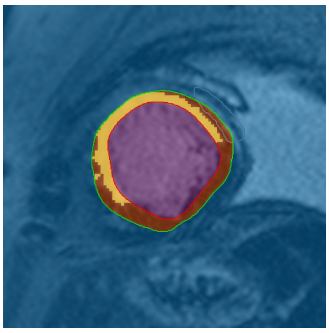
11.1 Myocardular Infarctions



(a) An image of a healthy heart segmented into chambers and muscle area marked by the red endocardial rings surrounded by a black muscular vessel indicated by the green epicardial ring.)



(b) The white objects inside the border between the endocardial and epicardial rings indicate the presense of a myocardial infarction. This area is typically contoured where the damaged areas are labelled manually.



(c) This shows a fully labelled image, with the purple blood flow surrounded by red muscle with yellow scarring and a blue background. With enough labelled images an automated process can be enacted [180].

Figure 46: Cross sections of a CMRI scan of the heart taken from the pericardium at the bottom to the top pulmonary artery and the human annotations made to each image.

In conjunction with Prof. Gerry McCann at the National Institute Of Health Research (NIHR) at Glenfield Hospital in Leicester and Dr Shuihua Wang at Leicester University, the scope of this project expanded it's applications into the field of medical images and the automated classification of a series of Cardiovascular Magnetic Resonance Imaging (CMRI) scans. These scans represent an image slice taken from the heart pointing downwards with the pulmonary artery feeding blood into the top to the pericardium at the bottom. This 3D object is split into between 10-14 horizontal slices depending on the patient within a few millimetres between each slice (the heart is a comparable size to a fist, often 12 cm by 8 cm wide by 6 cm thick [175]). In these images, the research team is looking for the presence of any scarring inside the muscle tissue which is used as a strong predictor of irregular changes in the left ventricular section of the heart ob-

struction, [177]. This scarring is often found in patients suffering from acute ST segment elevation myocardial infarctions (STEMI), where the size of these myocardial infarctions showed a positive correlation with mortality. [178], [179].
The myocardian CMRI images have three components: The central vessel is a grey circular object that shows the blood flowing directly into the heart. This is surrounded by an endocardial border while an outer black layer shows the outside muscle that convulses to pump the blood into the body. It's at this area where scarring can occur, shown as white fragments in this outer layer. The epicardial border then separates these two inner sections with the rest of the heart scan. Using pixel intensities, it's possible to build on the current literature [177], [180] to form the first fully automated process for quantifying myocardial infarcions through CMRI.

11.2 Protein Folding

Collaborating with the Leicester Institute of Structural and Chemical Biology, work was conducted on a program that could detect biological specimens present in cryo-electron microscopy images and provide a self correcting process that would find complete particles even with noise filters attached to them. In the current selection process outlined by Dimitry Tegunov and Patrick Cramar [181], a single MRC [182] image contains a combination of particles representing specimens and artifacts on an unclear background. Using their selection program Boxnet, the program evaluates, corrects and processes this data until distinctions can be made. The problem with this process is compared to unfiltered data, noise distortion makes separating particles and artifacts difficult.

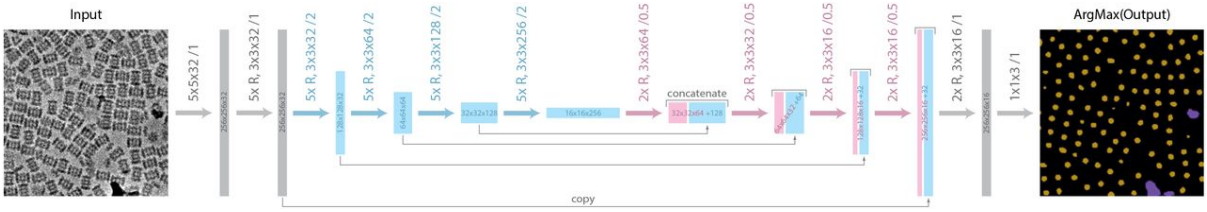
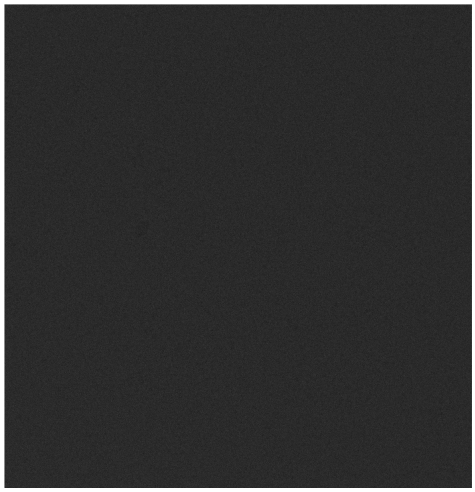
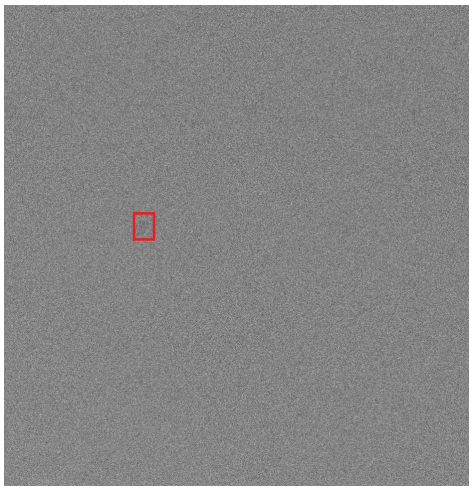


Figure 47: Boxnet is a convolutional ResNet architecture with 72 layers, a set of 35 ResNet blocks with 2 conventional CNN's. After initial convolution with 32 5x5 kernels the data is processed through 5 sets of 5 ResNet blocks with factor 2 downsampling and doubling the number of channels to determine higher order features. This then goes through 5 sets of 2 ResNet blocks, upsampled by factor 2. Each set is concatenated with it's mirrored counterpart, taking the new higher order features with higher spatial resolutions of old layers. The final output is a 256x256x16 layer that's projected onto 3 class channels. Similar to bottlenecks in Inception, further tests can be run to determine false positives, using the Argmax operation to choose the best label over Softmax [181].

The sample MRC image is a 3838x3710 rectangle of values between 0 and 255, giving 14,238,980 parameters in total. These represent grayscale pixel intensities that mask hidden particles in a noise filter as shown in Figure 48a. By dividing the square by the largest pixel value, each number is now between 0 to 1 and as Figure 48b shows, some patterns start to emerge to a casual observer such as the red square that shows a small area of comparatively high intensity values. Putting these through Boxnet, 109 particles are detected in the MRC image. These results are hard to verify through usual means however building on work discussed in Chapter 3, the final 256x256x16 matrix that ResNet produces goes through a further two ResNet blocks before being split by three 1x1 convolutional kernels where the pixelwise argmax gives the final judgement on whether that pixel contains the presence of a particle, an artifact or a background. Using this technology, it's possible to pre-assess if these elements have been detected correctly or not.



(a) Noisy filtered image of masked particle system



(b) Normalised variant with clearer particles

Figure 48: While these images appear incomprehensible to most humans and machines with the large amount of excess noise (even if some elements can be made out such as the red square in the normalised version in Figure 48b).

12 Conclusions

In this thesis, three issues with Machine Learning theory have been identified with regards to giving the best possible performance when time and resources have most likely been spent on developing the core AI system for a given situation. Given that these AI systems are driven by the requirement to manage large volumes of data, there is always scope for errors in this system, as the raw data is always biased and uncertain.
This degree of uncertainty varies from known cases which can be modelled, to radical cases, where the uncertainty presence is to an unknown (This is the case for most machine learning applications as the probability distribution is usually unknown).
With the identification of these issues, solutions have been proposed in the forms of Algorithms 1, 2 and Theorems 7, 8. These theoretical results have been tested in real life scenarios and have provide further areas where these results can be applied to in the future. In addition to these investigations, defining tighter bounds on error correction performance is known to be a very hard mathematical challenge and the findings established here will help further studies in this area of research.
Through these studies, error correction procedures have been reviewed and implemented successfully into a number of tactile sign languages resulting in a series of novelties. This work has also proposed the first ever system capable of recognising multiple types of sign language in one algorithm through a pre-defined legacy AI as shown in Section 8.6.2 along with a novel semaphore recognition system which can be supplemented with the aforementioned error correction system.
The methods described in this work can be tuned to a degree of success. All parameters these algorithms are customizable and can be adjusted to suit the corrector's sensitivity. For example, Sections 9.2 and 10.2.2 detail transformations made to a 10000x2048 matrix to single out element clusters relative to their principal dimension.
The first technique clusters singularities out of the core AI. For a training set, this example increases the probability of a successful classification from 82.4% to 100% knowing just 16% of the errors in this set. While this falters at the test set, more clusters would push these principles forward, and further work can be constructed remedying this.
Section 10.2.2 extends these principals to separating points in random sets. The comparison between random elements drawn from the unit cube and the bottlenecks shows a higher separability for common elements. This is pushed further with dimensionality, as while 174 principal components are attained for PCA via the Kaiser-Guttman test, linear and quadratic kernels use 18 and 86 dimensions respectively for complete separation. With future work, a system that utilises both of these theories with adequate prepossessing techniques could create a refined separation system.

13 Bibliography

References

[1] A. N. Gorban and I. Y. Tyukin. Stochastic separation theorems. Neural Networks, 94: 255–259, 2017.

[2] A. N. Gorban and I. Y. Tyukin. Blessing of dimensionality: mathematical foundations of the statistical physics of data. Philosophical Transactions of the Royal Society A, 376, 2018.

[3] One-Trial Correction of Legacy AI Systems and Stochastic Separation Theorems, Gorban, A.N. and Burton, R. and Romanenko, I. and Tyukin, I., Information Sciences, 484, 237-254, 2019

[4] Tyukin, I.Y., Gorban, A.N., Sofeykov, K.I. and Romanenko, I., 2018. Knowledge transfer between artificial intelligence systems. Frontiers in neurorobotics, 12.

[5] A. Gorban, I. Tyukin, and I. Romanenko, "The blessing of dimensionality: Separation theorems in the thermodynamic limit," TFMST 2016, `https://arxiv.org/abs/1610.00494v1`

[6] A. N. Gorban, A. Golubkov, B. Grechuk, E. M. Mirkes, and I. Y. Tyukin. Correction of ai systems by linear discriminants: Probabilistic foundations. Information Sciences, 466: 303–322, 2018.

[7] `https://emerj.com/ai-glossary-terms/what-is-machine-learning/`

[8] Vapnik, V.N., 1999. An overview of statistical learning theory. IEEE transactions on neural networks, 10(5), pp.988-999.

[9] Bhamare, D. and Suryawanshi, P., 2018. Review on Reliable Pattern Recognition with Machine Learning Techniques. Fuzzy Information and Engineering, pp.1-16.

[10] Boixader Ibáñez D. Special issue on pattern recognition techniques in data mining. Pattern Recognit Lett. 2017;93:1–2.

[11] Wu J, Yinan Y, Chang H, et al. Deep multiple instance learning for image classification and autoannotation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2015. p. 3460–3469.

[12] Aslan MS, Hailat Z, Alafif TK, et al. Multi-channel multi-model feature learning for face recognition. Pattern Recognit Lett. 2017;85:79–83.

[13] https://bit.ly/2kEYyaF

[14] Cristianini, N., Shawe-Taylor, J. (2000). An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press.

[15] Li, Cheng Yen. "Fisher Linear Discriminant Analysis." (2014).

[16] Fisher, R.A., 1936. The use of multiple measurements in taxonomic problems. Annals of eugenics, 7(2), pp.179-188.

[17] Donoho, David L. "High-dimensional data analysis: The curses and blessings of dimensionality." AMS math challenges lecture 1.2000 (2000): 32.

[18] P.C. Kainen, Utilizing geometric anomalies of high dimension: when complexity makes computation easier. In Computer-Intensive Methods in Control and Signal Processing: The Curse of Dimensionality, Springer, 1997, pp. 283–294.

[19] D. G. Luenberger, Optimization by Vector Space Methods, NY: Willey, 1969

[20] David G. Luenberger and Yinyu Ye (1973). Linear And Nonlinear Programming

[21] R. O. Duda and P. E. Hart. Pattern Classification and Scene Analysis. Wiley, 1973.

[22] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review, 65:386-408, 1959.

[23] M.L. Minsky and S.A. Papert. Perceptrons. MIT Press, 1969. Expanded Edition 1990.

[24] Joseph, D.B. and Bart, D.M. and K, S.J.A. (2002). Least Squares Support Vector Machines, World Scientific Publishing Company.

[25] Christopher M. Bishop (2006). Pattern Recognition And Machine Learning

[26] Vapnik, Vladimir (2000). The nature of statistical learning theory. Springer.

[27] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability and its Applications, 16(2):264-280, 1971.

[28] https://bit.ly/36BjHVr

[29] Jonathan Robinson (2008). Support Vector Machine Learning

[30] Wang, L. (2005), Support Vector Machines: Theory and Applications

[31] Boyle, B. (2011), Support Vector Machines: Data Analysis, Machine Learning and Applications

[32] Murty, M.N., Raghava, Rashm. (2016), Support Vector Machines And Perceptrons: Learning, Optimization, Classification And Applications To Social Networks.

[33] Vapnik, V. Cortes, C. (1995), Support Vector Networks

[34] Hulkoti, H. (2014), Combining Support Vector Machine Learning with the Discrete Wavelet Trannsform and Contourlet Transform in Image Compression

[35] Deng, L. and Yu, D., 2014. Deep learning: methods and applications. Foundations and Trends® in Signal Processing, 7(3–4), pp.197-387.

[36] https://www.oxfordreference.com/view/10.1093/oi/authority.20110803100047903

[37] Guo-Xun Yuan; Chia-Hua Ho; Chih-Jen Lin (2012). "Recent Advances of Large-Scale Linear Classification" (PDF). Proc. IEEE. 100 (9).

[38] Micchelli, C.A., Xu, Y. and Zhang, H., 2006. Universal kernels. Journal of Machine Learning Research, 7(Dec), pp.2651-2667.

[39] G. Wahba. Spline Models for Observational Data, volume 59 of CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia, 1990.

[40] Schölkopf, B., Herbrich, R. and Smola, A.J., 2001, July. A generalised representer theorem. In International conference on computational learning theory (pp. 416-426). Springer, Berlin, Heidelberg.

[41] Shawe-Taylor, J. and Cristianini, N., 2004. Kernel methods for pattern analysis. Cambridge university press.

[42] Schölkopf, B., Smola, A.J. and Bach, F., 2002. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press.

[43] arXiv:1801.05894

[44] Kecman, V., 2001. Learning and soft computing: support vector machines, neural networks, and fuzzy logic models. MIT press.

[45] Barron, A.R., 1993. Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information theory, 39(3), pp.930-945.

[46] Evans, J.A., Bazilevs, Y., Babuška, I. and Hughes, T.J., 2009. n-Widths, sup–infs, and optimality ratios for the k-version of the isogeometric finite element method. Computer Methods in Applied Mechanics and Engineering, 198(21-26), pp.1726-1741.

[47] Bottou L. (2012) Stochastic Gradient Descent Tricks. In: Montavon G., Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg

[48] Bottou, L., 2010. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010 (pp. 177-186). Physica-Verlag HD.

[49] Rojas, R., 2009. Neural Networks: a systematic introduction.

[50] Hecht-Nielsen, R., 1992. Theory of the backpropagation neural network. In Neural networks for perception (pp. 65-93). Academic Press.

[51] Dechter, Rina. (1986). Learning While Searching in Constraint-Satisfaction-Problems.. AAAI. 178-185.

[52] Bengio, Y., Goodfellow, I. and Courville, A., 2017. Deep learning (Vol. 1). MIT press.

[53] L. Ba and R. Caurana. Do deep nets really need to be deep? arXiv preprint arXiv:1312.6184, 2013.

[54] Le, Q.V., 2015. A tutorial on deep learning part 1: Nonlinear classifiers and the backpropagation algorithm & A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. Google Inc., Mountain View, CA,

[55] Boser, B.E., Guyon, I.M. and Vapnik, V.N., 1992, July. A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory (pp. 144-152). ACM.

[56] Bengio, Y., 2009. Learning deep architectures for AI. Foundations and trends® in Machine Learning, 2(1), pp.1-127.

[57] Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural networks, 61, pp.85-117.

[58] https://bit.ly/2LLAtvE

[59] https://bit.ly/2mMYPsL

[60] Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space". Philosophical Magazine. 2 (11): 559–572. doi:10.1080/14786440109462720.

[61] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, Artificial Neural Networks – ICANN'97. pages 583-588. Springer Lecture Notes in Computer Science, Volume 1327, 1997.

[62] D. Jackson, Stopping Rules in Principal Components Analysis: A Comparison of Heuristical and Statistical Approaches, Ecology 74 (8) (1993) 2204-2214.

[63] Molnar, Christoph. "Interpretable machine learning. A Guide for Making Black Box Models Explainable", 2019. https://christophm.github.io/interpretable-ml-book/.

[64] Altman, N. and Krzywinski, M., (2015). Points of significance: simple linear regression.

[65] https://www.cs.helsinki.fi/u/ahyvarin/whatisica.shtml

[66] Langlois, D., Chartier, S. and Gosselin, D., 2010. An introduction to independent component analysis: InfoMax and FastICA algorithms. Tutorials in Quantitative Methods for Psychology, 6(1), pp.31-38.

[67] P. P. Brahma, D. Wu, and Y. She. Why deep learning works: a manifold disentanglement perspective. IEEE Transactions On Neural Networks And Learning Systems, 27 (10): 1997–2008, 2016.

[68] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[69] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. Int. J. Comput. Vis., pages 1–42, 2014. DOI:10.1007/s11263-015-0816-y.

[70] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5mb model size. arXiv preprint, arXiv:1602.07360, 2016.

[71] R. Beene, A. Levin, and E. Newcomer. Uber self-driving test car in crash wasn't programmed to brake. https://www.bloomberg.com/news/articles/2018-05-24/uber-self-driving-system-saw-pedestrian-killed-but-didn-t-stop, 2018.

[72] C. Foxx. Face recognition police tools "staggeringly inaccurate". http://www.bbc.co.uk/news/technology-44089161, 2018.

[73] A. Kuznetsova, S. Hwang, B. Rosenhahn, and L. Sigal. Expanding object detector's horizon: Incremental learning framework for object detection in videos. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 28–36, 2015.

[74] I. Misra, A. Shrivastava, and M. Hebert. Semi-supervised learning for object detectors from video. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3594–3602, 2015.

[75] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3282–3289, 2012.

[76] S. Zheng, Y. Song, T. Leung, and I. Goodfellow. Improving the robustness of deep neural networks via stability training. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4480–4488, 2016. https://arxiv.org/abs/1604.04326.

[77] T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. ICLR 2016, 2015.

[78] L. Pratt. Discriminability-based transfer between neural networks. Advances in Neural Information Processing, (5): 204–211, 1992.

[79] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Advances in neural information processing systems, pages 3320–3328, 2014.

[80] V. Vapnik and R. Izmailov. Knowledge transfer in svm and neural networks. Annals of Mathematics and Artificial Intelligence, pages 1–17, 2017.

[81] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55 (1): 119–139, 1997.

[82] J. Gama and P. Brazdil. Cascade generalization. Machine learning, 41 (3): 315–343, 2000.

[83] L. Rokach. Ensemble-based classifiers. Artificial Intelligence Review, 33 (1-2): 1–39, 2010.

[84] D. Wang and C. Cui. Stochastic configuration networks ensemble with heterogeneous features for large-scale data analytics. Information Sciences, 417: 55–71, 2017.

[85] T. J. Draelos, N. E. Miner, C. C. Lamb, C. M. Vineyard, K. D. Carlson, C. D. James, and J. B. Aimone. Neurogenesis deep learning. In IEEE International Joint Conference on Neural Networks (IJCNN), pages 526–533. IEEE, 2017.

[86] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In Advances in neural information processing systems, pages 524–532, 1990.

[87] S. Scardapane and D. Wang. Randomness in neural networks: an overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 7 (2), 2017.

[88] D. Wang and M. Li. Deep stochastic configuration networks with universal approximation property. arXiv preprint arXiv: 1702.05639. 2017, 2017a.(An updated version of this work has been published in the Proceedings of 2018 IJCNN).

[89] D. Wang and M. Li. Stochastic configuration networks: Fundamentals and algorithms. IEEE transactions on cybernetics, 47 (10): 3466–3479, 2017b.

[90] J. Gibbs. Elementary Principles in Statistical Mechanics, developed with especial reference to the rational foundation of thermodynamics. Dover Publications, New York, 1960 (1902).

[91] A. N. Gorban. Order-disorder separation: Geometric revision. Physica A, 374: 85–102, 2007.

[92] M. Gromov. Metric Structures for Riemannian and non-Riemannian Spaces. With appendices by M. Katz, P. Pansu, S. Semmes. Translated from the French by Sean Muchael Bates. Birkhauser, Boston, MA, 1999.

[93] M. Gromov. Isoperimetry of waists and concentration of maps. GAFA, Geomteric and Functional Analysis, 13: 178–215, 2003.

[94] P. Lévy. Problèmes concrets d'analyse fonctionnelle. Gauthier-Villars, Paris, second edition, 1951.

[95] F. Cucker and S. Smale. On the mathematical foundations of learning. Bulletin of the American mathematical society, 39 (1): 1–49, 2002.

[96] V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. Neural Computation, 12 (9): 2013–2036, 2000.

[97] F. Rosenblatt. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, 1962.

[98] J. Anderson, M. Belkin, N. Goyal, L. Rademacher, and J. Voss. The more, the merrier: the blessing of dimensionality for learning large Gaussian mixtures. Journal of Machine Learning Research: Workshop and Conference Proceedings, 35: 1–30, 2014.

[99] Tyukin, I.Y., Gorban, A.N., Grechuk, B. and Green, S., 2019, July. Kernel Stochastic Separation Theorems and Separability Characterizations of Kernel Classifiers. In 2019 International Joint Conference on Neural Networks (IJCNN) (pp. 1-6). IEEE.

[100] O. Chapelle. Training a support vector machine in the primal. Neural computation, 19 (5): 1155–1178, 2007.

[101] T. Hofmann, B. Sch¨olkopf, and A. J. Smola, "Kernel methods in machine learning," The annals of statistics, pp. 1171–1220, 2008.

[102] R. Herbrich, Learning kernel classifiers: theory and algorithms, 2001.

[103] V. Vapnik, The Nature of Statistical Learning Theory. Springer-Verlag, 2000.

[104] K.-P. Wu and S.-D. Wang, "Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space," Pattern Recognition, vol. 42, no. 5, pp. 710–717, 2009.

[105] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," IEEE transactions on Neural Networks, vol. 13, no. 2, pp. 415–425, 2002.

[106] N. Ancona, G. Cicirelli, E. Stella, and A. Distante, "Object detection in images: run-time complexity and parameter selection of support vector machines," in Pattern Recognition, 2002. Proceedings. 16th International Conference on, vol. 2. IEEE, 2002, pp. 426–429.

[107] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," Machine learning, vol. 46, no. 1-3, pp. 131–159, 2002.

[108] S. Lessmann, R. Stahlbock, and S. F. Crone, "Genetic algorithms for support vector machine model selection." 2006.

[109] M. Varewyck and J.-P. Martens, "A practical approach to model selection for support vector machines with a gaussian kernel," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 41, no. 2, pp. 330–340, 2011.

[110] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," IEEE transactions on electronic computers, no. 3, pp. 326–334, 1965.

[111] S. S. Haykin, Neural networks and learning machines. Pearson Upper Saddle River, 2009, vol. 3.

[112] Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing, Donoho, D. and Tanner, J., Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 367, 1906, 4273–4293, 2009, The Royal Society Publishing

[113] P. C. Kainen and V. Kurkova. Quasiorthogonal dimension of euclidian spaces. Appl. Math. Lett., 6 (3): 7–10, 1993.

[114] A. Gorban, I. Tyukin, D. Prokhorov, and K. Sofeikov, "Approximation with random bases: Pro et contra," Information Sciences, vol. 364–365, pp. 129–145, 2016.

[115] On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities, Vapnik, V.N. and Chervonenkis, A. Ya., Theory of Probability & Its Applications, 16, 2, 264–280, 1971, SIAM

[116] Tyukin, I.Y., Gorban, A.N., Green, S. and Prokhorov, D., 2019. Fast construction of correcting ensembles for legacy Artificial Intelligence systems: Algorithms and a case study. Information Sciences, 485, pp.230-247.

[117] Learning kernels using local Rademacher complexity, Cortes, C. and Kloft, M. and Mohri, M., Advances in neural information processing systems, 2760–2768, 2013

[118] Artificial Intelligence Global Adoption Trends and Strategies (IDC US45120919) `https://www.idc.com/getdoc.jsp?containerId=US45120919`

[119] Combining Support Vector Machine Learning With The Discrete Cosine Transform In Image Compression by Vojislav Kecman and Jonathan Robinson

[120] `https://cs231n.github.io/`

[121] Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. CoRR, abs/1311.2901, 2013. Published in Proc. ECCV, 2014.

[122] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In Proc. ICLR, 2014.

[123] Simonyan, K. and Zisserman, A. Two-stream convolutional networks for action recognition in videos. CoRR, abs/1406.2199, 2014. Published in Proc. NIPS, 2014.

[124] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, 2012.

[125] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[126] Histograms of Oriented Gradients for Human Detection by Navneet Dalal and Bill Triggs.

[127] ImageNet Classification with Deep Convolutional Neural Networks by Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton

[128] Remote Visual/Optical Interface for 3-D Surface Visualisation by Richard Burton.

[129] MatConvNet: Convolutional Neural Networks For MATLAB by Andrea Vedaldi and Karel Lenc.

[130] MatConvNet by Andrea Vedaldi, Karel Lenc and Ankush Gupta

[131] `http://yann.lecun.com/exdb/mnist/`

[132] Romanuke, V.V., 2016. Training data expansion and boosting of convolutional neural networks for reducing the MNIST dataset error rate.

[133] Glorot, X., Bordes, A. and Bengio, Y., 2011, June. Deep sparse rectifier neural networks. In Proceedings of the fourteenth international conference on artificial intelligence and statistics (pp. 315-323).

[134] `http://www.image-net.org/challenges/LSVRC/`

[135] C. Valli, editor.The Gallaudet Dictionary of American Sign Language. Gallaudet U. Press, Washington, DC,2006.

[136] https://nationaldb.org/library/page/1934

[137] https://ghr.nlm.nih.gov/condition/usher-syndrome

[138] Australian Deafblind Council (ADBC) 2004 https://www.deafblindinformation.org.au/about-deafblindness/

[139] http://www.dbglove.com/pages/en/home/

[140] Mitchell, Ross E., et al. "How many people use ASL in the United States? Why estimates need updating." Sign Language Studies 6.3 (2006): 306-335.

[141] Starner, T. and Pentland, A., 1997. Real-time american sign language recognition from video using hidden markov models. In Motion-Based Recognition (pp. 227-243). Springer, Dordrecht.

[142] Liang, R.H. and Ouhyoung, M., 1998, April. A real-time continuous gesture recognition system for sign language. In Proceedings third IEEE international conference on automatic face and gesture recognition (pp. 558-567). IEEE.

[143] Imagawa, K., Lu, S. and Igi, S., 1998, April. Color-based hands tracking system for sign language recognition. In Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition (pp. 462-467). IEEE.

[144] Starner, Thad, Joshua Weaver, and Alex Pentland. "Real-time american sign language recognition using desk and wearable computer based video." IEEE Transactions on pattern analysis and machine intelligence 20.12 (1998): 1371-1375.

[145] Johnston, Trevor. "The lexical database of auslan (australian sign language)." Sign Language & Linguistics 4.1 (2001): 145-169.

[146] https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

[147] https://bit.ly/2kBZtoF

[148] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[149] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[150] Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." Thirty-First AAAI Conference on Artificial Intelligence. 2017.

[151] Tyukin, Ivan Y., et al. "Efficiency of Shallow Cascades for Improving Deep Learning AI Systems." 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018.

[152] Rathi, D., 2018. Optimization of Transfer Learning for Sign Language Recognition Targeting Mobile Platform. arXiv preprint arXiv:1805.06618.

[153] Reed, Charlotte M., et al. "Research on the Tadoma method of speech communication." The Journal of the Acoustical society of America 77.1 (1985): 247-257.

[154] ISBN 978-0-86341-327-8 Communications: an international history of the formative years R. W. Burns, 2004

[155] L. Deng and 1. Ma, "Spontaneous speech recognition Using a statistical articulatory model for the vocal tract resonance dynamics," 1. Acoust.Soc.Am. vol. 108, pp. 3036-3048, 2000.

[156] Hung, I.-C., Hsu, H.-H., Chen, N.-S., & Kinshuk. (2015). Communicating through body: a situated embodiment-based strategy with flag semaphore for procedural knowledge construction. Educational Technology Research and Development, 63(5), 749–769. doi:10.1007/s11423-015-9386-5

[157] "Flag semaphore", `https://en.wikipedia.org/wiki/Flag_semaphore#Japanese_semaphore`

[158] Noriyuki Iwane, "Arm movement recognition for flag signaling with Kinect sensor", In the IEEE International Conference on Virtual EnvironmentsHuman-ComputerInterfaces and Measurement Systems (VECIMS) Proceedings, 2012, p.86-90.

[159] Different Types Of Sign Language Around The World `https://bit.ly/2k6ueFO`

[160] British Deaf Association `https://bda.org.uk/help-resources/`

[161] Fang, Gaolin & Gao, Wen & Zhao, Debin. (2007). Large-Vocabulary Continuous Sign Language Recognition Based on Transition-Movement Models. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on. 37. 1 - 9. 10.1109/TSMCA.2006.886347.

[162] Green S., Tyukin I., Gorban A. (2020) Using Convolutional Neural Networks to Distinguish Different Sign Language Alphanumerics. In: Oneto L., Navarin N., Sperduti A., Anguita D. (eds) Recent Advances in Big Data and Deep Learning. INNSBDDL 2019. Proceedings of the International Neural Networks Society, vol 1. Springer, Cham

[163] K. Kumar, Vinay & Goudar, R.H. & Desai, V.T.. (2015). Sign Language Unification: The Need for Next Generation Deaf Education. Procedia Computer Science. 48. 673-678. 10.1016/j.procs.2015.04.151.

[164] `http://www.who.int/mediacentre/factsheets/fs300/en/`

[165] D. Jackson. Stopping rules in principal components analysis: A comparison of heuristical and statistical approaches. Ecology, 74 (8): 2204–2214, 1993.

[166] D. Arthur and S. Vassilvitskii. How slow is the k-means method? Proceedings of the twenty-second annual symposium on Computational geometry, pages 144–153. ACM, 2006.

[167] J. A. Hartigan and M. A. Wong. A K-Means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28 (1): 100–108, 1979.

[168] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[169] V. Bheda and D. Radpour. Using deep convolutional networks for gesture recognition in american sign language. arXiv preprint arXiv:1710.06836, 2017.

[170] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.

[171] V. Pestov, Is the k-NN classifier in high dimensions affected by the curse of dimensionality? Comput. Math. Appl. 65 (2013) 1427–1437.

[172] W. Hoeffding. Probability inequalities for sums of bounded random variables. Journal of the American statistical association, 58 (301): 13–30, 1963.

[173] K. Ball. An elementary introduction to modern convex geometry. Flavors of geometry, 31: 1–58, 1997.

[174] `https://www.tensorflow.org/tutorials/image_retraining`

[175] Betts, J. Gordon (2013). Anatomy & physiology. pp. 787–846. ISBN 978-1-938168-13-0.

[176] Bulluck, Heerajnarain, et al. "Impact of microvascular obstruction on semiautomated techniques for quantifying acute and chronic myocardial infarction by cardiovascular magnetic resonance." Open Heart 3.2 (2016): e000535.

[177] Klem I, Shah DJ, White RD, et al. Prognostic value of routine cardiac magnetic resonance assessment of left ventricular ejection fraction and myocardial damage: an international, multicenter study. Circ Cardiovasc Imaging 2011;4:610–19.

[178] Wagner A, Mahrholdt H, Holly TA, et al. Contrast-enhanced MRI and routine single photon emission computed tomography (SPECT) perfusion imaging for detection of subendocardial myocardial infarcts: an imaging study. Lancet 2003;361:374–9.

[179] McAlindon, Elisa, et al. "Quantification of infarct size and myocardium at risk: evaluation of different techniques and its implications." European Heart Journal-Cardiovascular Imaging 16.7 (2015): 738-746.

[180] Khan, Jamal N., et al. "Comparison of semi-automated methods to quantify infarct size and area at risk by cardiovascular magnetic resonance imaging at 1.5 T and 3.0 T field strengths." BMC research notes 8.1 (2015): 52.

[181] Tegunov, Dimitry, and Patrick Cramer. "Real-time cryo-EM data pre-processing with Warp." BioRxiv (2018): 338558.

[182] `http://www.ccpem.ac.uk/mrc_format/mrc2014.php`

14 Appendix: Publications

This work has been published in the following papers:

1. Green S., Tyukin I., Gorban A. (2020) Using Convolutional Neural Networks to Distinguish Different Sign Language Alphanumerics.

2. Tyukin, I.Y., Gorban, A.N., Green, S. and Prokhorov, D., 2019. Fast construction of correcting ensembles for legacy Artificial Intelligence systems: Algorithms and a case study. Information Sciences, 485, pp.230-247.

3. Tyukin, I.Y., Gorban, A.N., Grechuk, B. and Green, S., 2019, July. Kernel Stochastic Separation Theorems and Separability Characterizations of Kernel Classifiers. In 2019 International Joint Conference on Neural Networks (IJCNN) (pp.1-6) IEEE.

15 Appendix: Conferences

This work has been presented at the following conferences:

1. International Joint Conference on Neural Networks (IJCNN) 2018: Rio De Janiero.
2. Artificial Intelligence International Conference (A2IC) 2018: Barcelona.
3. International Joint Conference on Neural Networks (IJCNN) 2019: Budapest.
4. International Neural Network Society Conference (INNS) 2019: Genoa.