Random Projections for Scaling Machine Learning on FPGAs

Sean Fox, Stephen Tridgell, Craig Jin and Philip H.W. Leong
{sean.fox, stephen.tridgell, craig.jin, philip.leong}sydney.edu.au
School of Electrical and Information Engineering, Building J03
The University Of Sydney, 2006, Australia

***Abstract*—Random projections have recently emerged as a powerful technique for large scale dimensionality reduction in machine learning applications. Crucially, the projection can be obtained from sparse probability distributions, enabling hardware implementations with little overhead. In this paper, we describe a Field-Programmable Gate Array (FPGA) implementation alongside a kernel adaptive filter (KAF) that is capable of reducing computational resources by introducing a controlled error term, achieving higher modelling capacity for given hardware resources. Empirical results involving classification, regression and novelty detection show that a 40% net increase in available resources and improvements in prediction accuracy is achievable for projections which halve the input vector length, enabling us to scale-up hardware implementations of KAF learning algorithms by at least a factor of 2. An implementation on a FPGA-based network card allows novelty detection of an 8× 24-bit input vector with latency of 404 ns, this being a 26-fold reduction compared to an Intel Core i5-2400 processor.**

I. Introduction

For many machine learning applications, the original input variables are transformed into a new space in which the problem is easier to solve. This may occur because the discriminative characteristics are more prominent in the new space or the application itself is constrained by computational necessity. Over the last decade, examples of the latter case have become more frequent as data is generated and acquired at increasing rates. With this trend likely to continue, interest has grown in matrix algorithms for scalable factorisation and dimensionality reduction. Most methods are based on the mathematical properties of principle components or eigenvectors, which are relatively expensive to implement [1]. This is a deterrent for real-time machine learning systems, and can limit field programmable gate array (FPGA) implementations to problems with only a modest number of features. To address this issue and scale to higher dimensional inputs, improved techniques are desirable. In this paper we propose a hardware architecture for random projections, which has several advantages over analytical methods, such as principal component analysis (PCA) [2], for dimensionality reduction:

- Randomness can obtain high quality approximations while avoiding unnecessary computational complexity.
- Hardware implementations can be very efficient since they only involve static data structures; are highly parallelisable; only require multiplication by -1, 0, and +1; have minimal logic and memory requirements; and can be integrated more compactly on the same FPGA as the downstream learning algorithms.

Kernel adaptive filters (KAF) are a class of online learning algorithms which are highly suitable to FPGA implementations and can be adapted to perform different tasks, such as regression, classification and anomaly detection [3] [4]. However, the main limitation of KAFs is the evaluation of an inner product in the input space which consumes most of the FPGA's computational resources and limits applications to small input sizes. This can be addressed using conventional techniques to compress the input, but such an approach can impact performance in latency critical applications, especially if the data needs to be compressed on a central processing unit (CPU). Our random projection architecture overcomes this problem because it allows the pre-processing stage to be moved onto the FPGA alongside the KAF and occupies minimal resources. This is advantageous for a range of applications which require high data rates and low power consumption such as machine prognostics, network monitoring and remote hyperspectral image recognition.

While the primary focus in this paper is online machine learning, where practical applications are constrained by execution time or resource requirements, random projections should be equally effective for batch learning problems.

The key contributions of this paper are:

- To the best of our knowledge, this is the first known study in which random projections are used to scale up hardware implementations of machine learning algorithms. Without loss of generality, we only consider the online case as opposed to batch learning.
- The benefits and trade-offs of an FPGA implementation are demonstrated for examples involving classification, regression and novelty detection. We find that dimensionality reduction via random projections can better leverage the available hardware resources for increased modelling capacity and achieve similar performance.
- We describe a system-level implementation of the Naive Online regularised Risk Minimisation Algorithm (NORMA) on a low-latency PCIe network card that reduces latency by a factor of 26 over a single core CPU implementation.

978-1-5090-5602-6/16/$31.00

This paper is organised as follows: Section II describes random projections and summarises previous work in the area. A brief summary is also given for a selected KAF algorithm; Section III describes the proposed architecture; Section IV discusses the results and describes a PCIe bus-based system implementation; lastly, Section V draws conclusions from our work.

II. Background

A. Random Projections

In this section, the required theory of random projections for dimensionality reduction is summarised. Particular attention is given to sparse random projections for which highly efficient hardware implementations are achievable.

Dimensionality reduction is a process for reducing the number of input variables (or features) from m to k, where $m > k$, yielding a problem which requires less time and/or resources to compute. Care must be taken to ensure that features extracted in the lower dimension preserve the discriminatory information contained in higher dimensions. The principal component analysis (PCA) approach which finds the optimal squared error, low-rank approximation of the original data is widely used for achieving both these goals [2]. However, PCA's $\mathcal{O}(nm^2 + m^3)$ complexity [5] limits its utility in real-time applications. Techniques which find sparse solutions are more suitable, and include sparse PCA (SPCA) and orthogonal matching pursuit (OMP), the latter having $\mathcal{O}(nm)$ time complexity [6].

In a random projection, the original m-dimensional data are projected to a lower k-dimensional space via a single multiplication of a randomly generated matrix $R_{m\times k}$.

$$X_{n\times k}{}^{new} = X_{n\times m}R_{m\times k}$$

Surprisingly, random mappings are capable of maintaining the pair-wise distances of points in the original high dimensional space. This finding is a key result of the Johnson-Lindenstrauss lemma [7] and forms the basis of the present work, and that of others [8] [9] [10], [11]. A proof of the Johnson-Lindenstrauss lemma is given in reference [12].

Generating the random matrix is itself a straightforward process. The elements r_{ij} of R are often chosen as Gaussian distributed, but for streaming applications or massively high-dimensional problems, simpler distributions are preferred such as those described by Achlioptas [8] and Li et al. [11]. These introduce sparsity and significantly reduce computational requirements. In fact, the requirement for preserving pair-wise distances is that the entries of R are independent and identically distributed (i.i.d) with zero mean [13]. We followed Li et al. [11] in choosing r_{ij} according to the following distribution:

$$r_{ij} = \sqrt{\frac{s}{k}} \times \begin{cases} 1 & \text{with prob.} \quad 1/(2s) \\ 0 & \text{with prob.} \quad (1-1/s) \\ -1 & \text{with prob.} \quad 1/(2s) \end{cases} \qquad (1)$$

where s controls the sparsity of the projection, and k is the reduced number of dimensions. This distribution can be implemented efficiently in hardware because for large values of s, matrix R is mostly sparse, and multiplications with $\sqrt{\frac{s}{k}}$ can be delayed or avoided. In Li et al. [11], the authors showed that random projections were robust for $s = \sqrt{m}$ but for increasing s, variances for sparse random projections increased and larger errors could be expected.

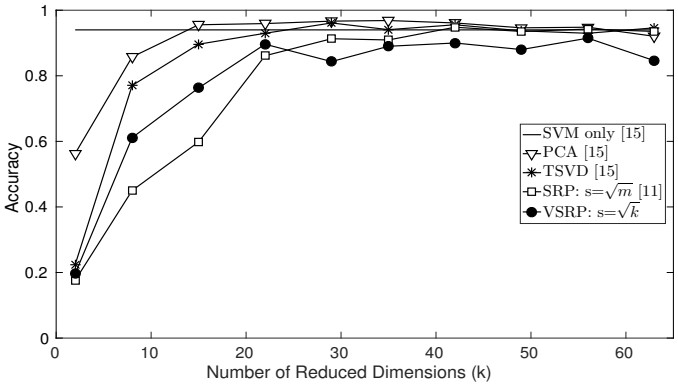


Fig. 1: Accuracy of VSRP compared with other techniques for dimensionality reduction

In this work, we choose $s = k$, which makes it easy to compute $\sqrt{\frac{s}{k}}$ but introduces more sparsity than suggested by Li et al. [11]. In subsequent sections we refer to this technique as VSRP, or *very sparse random projection*.

To assess the accuracy of VSRP, its performance is compared with sparse random projection (SRP) [11], PCA and truncated singular value decomposition (TSVD) [14] on a small subset of the MNIST dataset for image classification. The *sklearn.decomposition* module, from the *Scikit-learn* Python package [15], was used for the PCA and TSVD implementations. The input matrix, A, consisted of 1257 images containing 64 features, with $rank(A) = 61$. This means that every data point can be expressed as a linear combination of 61 independent vectors. A linear Support Vector Machine (SVM), from the *sklearn.svm* module [15], was trained following dimensionality reduction, and the accuracy plotted as a function of k in Figure 1. The VSRP technique was found to be competitive with other methods for dimensionality reduction.

B. NORMA

In online applications, one does not have a priori access to the data so models must be updated incrementally. Many training algorithms, including SVM, Gaussian processes and neural networks, require multiple passes over the entire dataset. In contrast, KAFs can be incrementally updated and are highly amenable to online implementations. KAFs employ the *kernel trick* to enable non-linear problems to be solved efficiently using linear techniques, via an implicit mapping of the input vector into a high-dimensional feature space [16], [4]. An example of a KAF is the Naive Online regularised Risk Minimisation Algorithm (NORMA) [17], which is used in this paper to learn a function, $f(x)$, which maps some input vector $x \in \mathbb{R}^m$ to a scalar output y. The learning algorithm is a stochastic gradient descent technique in which, for each new

x, the instantaneous predictive error is minimised [16]:

$$f_{t+1} = (1 - \eta\lambda) f_t - \eta_t l'(f_t(x_{t+1}), y_t)\kappa(x_{t+1}, \cdot) \quad . \tag{2}$$

where η is the learning rate (or step size), $(1-\eta\lambda)$ is the decay factor, $\kappa(x_{t+1}, \cdot)$ is some kernel function and $l'(f_t(x_{t+1}), y_t)$ represents the partial derivative of the loss function, or the instantaneous error function. By defining an appropriate loss function, NORMA can be used to solve classification, novelty detection and regression problems using equations 3, 4 and 5 respectively:

$$l(f(x) + b, y) = \max(0, \rho - y(f(x) + b)) - \nu\rho \tag{3}$$

$$l(f(x), y) = \max(0, \rho - f(x)) - \nu\rho \tag{4}$$

$$l(f(x), y) = \max(0, |y - f(x)| - \epsilon) + \nu\epsilon \tag{5}$$

where η, λ and ν are hyperparameters, b is an offset for the classification loss function, and ρ and ϵ are the threshold and tolerance values for training $f(x)$.

NORMA's capacity to model arbitrary functions comes from a suitable representation of $f(x)$. The kernel trick is employed to evaluate $f(x)$ as follows:

$$f(x) =< w, \phi(x) >= \sum_{i=1}^{|D|} \alpha_i \kappa(x, d_i) \tag{6}$$

where α_i are the weights, κ is the kernel function, $\mathcal{D}$ is the dictionary which is a small subset of the input data, $d_i \in \mathcal{D}$ and $|D|$ is the total number of entries. The update steps are obtained by differentiating the loss functions and substituting them into equation 2. The update equations for classification (7), novelty detection (8) and regression (9) are as follows:

$$(\alpha_i, \alpha_t, b, \rho) = \begin{cases} (\Omega\alpha_i, 0, b, \rho + \eta\nu) \text{ if } y(f(x) + b) \geq \rho \\ (\Omega\alpha_i, \eta y, b + \eta y, \rho - \eta(1 - \nu)) \text{ otherwise} \end{cases} \tag{7}$$

$$(\alpha_i, \alpha_t, \rho) = \begin{cases} (\Omega\alpha_i, 0, \rho + \eta\nu) \text{ if } f(x) \geq \rho \\ (\Omega\alpha_i, \eta, \rho - \eta(1 - \nu)) \text{ otherwise} \end{cases} \tag{8}$$

$$(\alpha_i, \alpha_t, \epsilon) = \begin{cases} (\Omega\alpha_i, 0, \epsilon - \eta\nu) \text{ if } |y - f(x)| \leq \epsilon \\ (\Omega\alpha_i, \delta\eta, \epsilon + \eta(1 - \nu)) \text{ otherwise} \end{cases} \tag{9}$$

where $\Omega = (1 - \eta\lambda) \leq 1$ is the decay factor, and $\delta = sign(y - f(x))$.

NORMA limits the size of the dictionary to the last $|D|$ dictionary entries. A new input, x_t, is added to the dictionary and the oldest one dropped, if it does not satisfy the condition of the appropriate update equations (Eq. 7, 8, or 9). The update of $\mathcal{D}$ is therefore:

$$[d_1, \cdots d_{|D|}] \rightarrow [x_t, d_1, \cdots d_{|D|-1}] \tag{10}$$

A sufficiently large $|D|$, and appropriate choice of hyperparameters gives the best prediction results. However, hardware implementations are often be limited to small $|D|$. We show that a random projection can be used to leverage greater model capacity and increase $|D|$ for high dimensional problems.

C. Literature Review

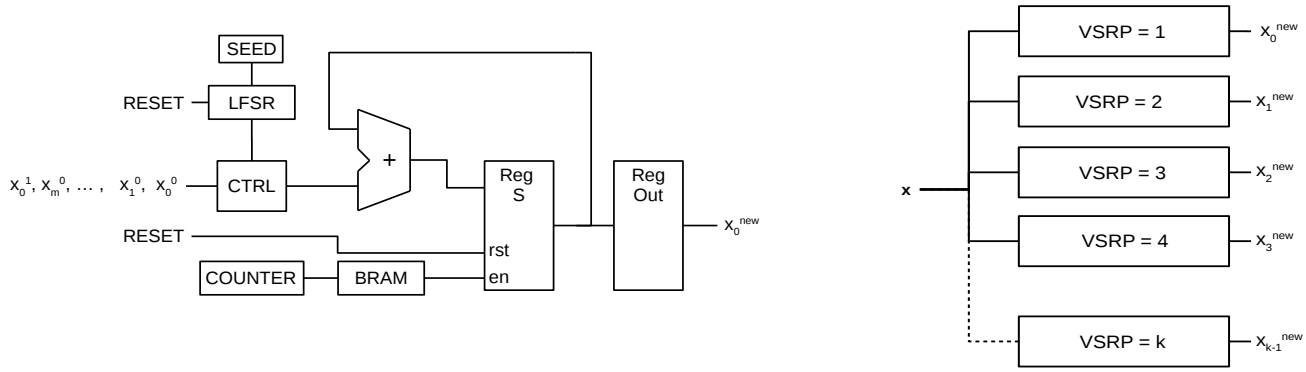
Efficient hardware implementations of dimensionality reduction is a well studied problem and several implementations for real-time applications and large-scale problems have been reported. Das et al. [18] describe an implementation of the PCA algorithm for network intrusion and anomaly detection. Rouhani et al. [6] proposed SSketch, a parallel implementation of the OMP algorithm for streaming hyperspectral images. OMP takes a dictionary and a signal as inputs and iteratively approximates the sparse representation of the signal by adding the best fitting element in every iteration. The relatively high complexity of the algorithm means that significant FPGA resources are required for its implementation. As a result, an implementation alongside NORMA (or another regression/classification algorithm) would be difficult to achieve.

In Bouganis et al. [19] a probabilistic framework is described for optimising algorithmic design choices for the Karhunen-Loeve Transform (KLT), which is a dimensionality reduction algorithm similar to PCA in performance and complexity. In particular, their work addresses the trade-off between prediction accuracy and resource utilisation. In our work, we make similar arguments but base our conclusions on empirical results for the random projection algorithm.

Other work in FPGA research have used random projections to improve modelling capacity by increasing the dimensionality. For example, Wang et al. [20] use a projection into a high-dimensional hidden layer to generate a large number of spatial-temporal patterns and train spiking neural networks. Despite this work, there is little evidence to suggest that random projections have been identified for hardware-friendly dimensionality reduction. Our work aims to fill this void.

Aside from FPGA research, the use of randomisation as a technique is gaining momentum in data science and machine learning communities [1], and recent work, such as the random encoding scheme described by Coates and Ng [21] and Rahimi and Recht [22], show surprisingly good results for training neural networks and large-scale kernel machines respectively. Earlier work involves improving performance of matrix factorisations specifically. Halko et al. [23] show how randomness can be used to find approximate solutions to computationally demanding problems in linear algebra, such as the Singular Value Decomposition (SVD) and the Least Squares (LS) problem, while Clarkson and Woodruff [24] show how these algorithms can be applied in a streaming model. This work describes strict bounds in terms of error-performance and computational cost, and is the primary motivation for our work in random projections.

Bingham et al. [9] and Dasgupta [10] have documented results for random projections on various real-world datasets, while sparse and very sparse implementations in Achlioptas et al. [8], Kane and Nelson [25] and Li et al. [11] have been shown to further reduce computational space and time. We build on this work and show that very sparse projections can be implemented with low overhead on an FPGA.



(a) VSRP module for one dimension

(b) k reduced dimensions

Fig. 2: Architecture for VSRP Module

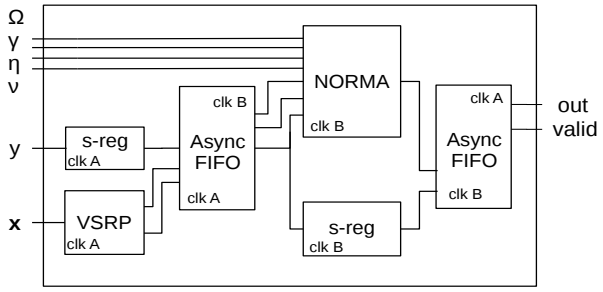


Fig. 3: A high level diagram of the random projection and NORMA processor

III. Architecture

In this section, our random projection architecture is described. We explore areas which are suitable for optimisations and discuss the scalability of our design. We also give a brief summary of the NORMA architecture, described in Tridgell et al. [4], to highlight constraints of the entire design.

A. Random Projection Module

A block diagram illustrating the data path for our "very sparse random projection (VSRP)" module is given in Figure 2. This corresponds to a dimensionality reduction $\mathbb{R}^m \rightarrow \mathbb{R}^k$ via a projection by a sparse random, trinary matrix, e.g.

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{m-1} \\ x_m \end{bmatrix}^T_{1\times m} \begin{bmatrix} 0 & -1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \cdots & -1 \\ -1 & 0 & \cdots & 0 \end{bmatrix}_{m\times k} = \begin{bmatrix} x_0^{new} \\ x_1^{new} \\ \vdots \\ x_k^{new} \end{bmatrix}^T_{1\times k}$$

We take a data stream as input, which is compatible with most sampled data and PCIe bus-based interfaces, and compute the dot product of an m-length input vector and a random matrix distributed according to Equation 1. Since the matrix can be generated independently from the input, the indices of the non-zero entries for each column are computed off-line, loaded to memory, and streamed for each input variable. This means that only m bits are required for each Block RAM ($BRAM$), and $m \times k$ bits are required for the entire matrix, where k is the number of reduced dimensions or columns. Each non-zero entry is either +1 or -1 and indicates whether the input is accumulated via addition or subtraction. We generate this randomly using a single bit from the output of a Linear Feedback Shift Register (LFSR) which determines whether the $CTRL$ block performs a two's complement on the input. We use 16-bit LFSR's, which are reinitialised after the last variable of each input has been streamed. This ensures the random matrix is the same for each new input vector. The state of the projection for a single dimension is given by $RegS$, which is accumulated for each feature by the adder block and the output of the $CTRL$ block. The enable signal, en, which is streamed from a $BRAM$, either writes the result to the new state or disables any change in state. When the first variable of a new input is observed, the state and LFSR reset, and a valid output is retrieved in 1 clock cycle. This means that dimensionality reduction is achieved in $m + 1$ clock cycles. As shown in Figure 2, this simple configuration is replicated for each reduced dimension. Therefore, VSRP area scales as $O(k)$.

B. NORMA Module

Figure 3 shows a block diagram of our integrated VSRP and NORMA design. NORMA is treated as a black box module with the following characteristics:

- NORMA accepts a vector, x, and a scalar, y, as input and returns a scalar, out, in a clock cycles, where a is determined by the number of pipeline stages.
- NORMA is parameterised by 4 hyperparameters, γ, Ω, η and ν, which are loaded once at run-time.
- NORMA requires memory for the dictionary, $\mathcal{D}$, and learning weights, α. The number of dictionary entries and weights are equivalent, and this is chosen at compile-time.

TABLE I: Classification results for VSRP+NORMA for projections of varying size in Fixed vs Floating point

Dataset	Artificial Classification ($i_w = 8$)					
k =	30	25	20	15	10	5
Floating $NORMA$	0.973	0.894	0.903	0.839	0.828	0.625
Fixed 18 bits	0.862	0.865	0.860	0.727	0.712	0.593
Fixed 24 bits	0.992	0.958	0.946	0.903	0.870	0.617
Fixed 30 bits	0.991	0.951	0.948	0.909	0.880	0.639
Fixed 36 bits	0.993	0.951	0.945	0.901	0.880	0.639

The resources required for the evaluation of the kernel function scales as $O(k|D|)$, available DSPs being the main constraint for large $|D|$ or k [4].

IV. Results and Discussion

This section describes the resource utilisation, performance and accuracy of our random projection implementation by testing a pre-existing design of the NORMA algorithm on three machine learning problems. In Subsection IV-A, we discuss the utility of random projections using a classification learning example. The benefits and typical design trade-offs are explained using a regression problem in Subsection IV-B, and in Subsection IV-C we describe our system-level implementation by considering the application of novelty detection in high-speed networks.

Our implementations were written in CHISEL [26], a hardware description language embedded in Scala. The NORMA implementation was derived from a design by Tridgell et al. [4], available from their Github repository. The designs were synthesised and implemented using Xilinx Vivado 2014.4. The target platform for IV-A and IV-B was a Xilinx Virtex-7 XC7VX690TFFG1930-2 FPGA. For the system-level implementation in IV-B, our target platform was an Exablaze ExaNIC X4 network card which uses a Xilinx Kintex-7 XC7K160TFBG676-2 [27] containing 600 DSPs, 25350 Slices and 325 Block RAMs. We benchmark performance against a single core C implementation running on an Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz and compiled with the '-O3' flag. Our code is freely available at https://github.com/sfox14/chisel-rp.

A. Performance and Resource Utilisation: Classification

We benchmark performance and utilisation on an artificial dataset which was generated using the *Scikit-learn* Python package [15]. The dataset contained 2000 samples, binary classes, and a high-dimensional input space with 30 features. A floating point implementation of VSRP+NORMA was written in Python and 1000 iterations of a particle swarm optimisation was run to obtain the hyperparameters. This was used as a reference for our fixed point CHISEL implementation, the results of which are presented in Table I. We used the area under the receiver operating characteristic (ROC) curve (AUC) [28] to measure classification accuracy. AUC scores are often preferred over percentage accuracy because they compensate for skewed datasets and biased models. An AUC which is close to 1 is optimal while scores near 0.5 indicate a random decision boundary. As discussed earlier, $|D|$, is the key parameter controlling accuracy and resource utilisation. In Table I, we do not consider any particular system-level constraints, but rather, as a proof of principle, use a very large dictionary size, $|D| = 200$, to discover the empirical bounds on classification performance for this example. As expected, prediction accuracy is reduced when projecting to increasingly lower dimensional spaces. It is interesting to note that halving the feature space (i.e. from 30 to 15) yields less than a 10% error for 24-bit fixed point (denoted as $i_w.i_f$ where $i_w = 8$ is the integer width and $i_f = 16$ is the fractional width). Even though the results seem to show fixed point outperforming floating point for more than 24 bits of precision, this is a noise effect, likely due to the small number of samples used for training and testing (train=1600 and test=400). Despite this observation, the errors introduced by the VSRP module remains relatively consistent, varying between 8-13% for a projection from $k = 30$ to $k = 15$. While a 10% error may be too large for some applications, random projection enables a massive savings in computational resources which can be re-allocated for improved accuracy (Section IV-B), and it also provides a potential solution for processing otherwise intractable high-dimensional datasets in real-time.

TABLE II: Performance of VSRP+NORMA for 8.16 Fixed, m=30 and $|D|$=45 on a Virtex 7

k =	30	25	20	15	10	5
NORMA						
DSPs (/3600)	3600	3239	2669	2069	1509	949
Freq (MHz)	90.1	90.9	94.3	95.2	97.1	98.0
Latency (Clocks)	12	12	12	11	11	11
Slices (/108300)	56484	35844	29844	21613	16346	9939
VSRP						
DSPs (/3600)	*	0	0	0	0	0
Freq (MHz)	*	344.8	347.2	350.9	363.6	392.1
Latency (Clocks)	*	31	31	31	31	31
Slices (/108300)	*	1025	795	571	392	182
VSRP + NORMA						
DSPs (/3600)	*	3239	2669	2069	1509	949
Freq-N (MHz)	*	83.3	84.7	87.7	88.7	89.3
Freq-RP (MHz)	*	333.3	339.0	350.9	354.6	357.1
Slices (/108300)	*	35887	30076	21841	16485	10080

In Table II, we show resource usage and clock frequency statistics for an 8.16-bit fixed point design with $|D| = 45$. The table shows decreasing usage of both NORMA and VSRP modules for projections to increasingly lower dimensions. Column 1 only contains results for NORMA because a projection into the same dimensional space gives no practical benefit. Here, NORMA's modelling capacity is constrained because all DSPs are used. We use this as a reference point while the rest of the table shows the extent to which reductions in the feature space can provide additional computational resources. This is possible because the number of multiplications in NORMA scales linearly with both $|D|$ and k.

To illustrate this point, consider the projection from 30 to 15 features (Table II): A reduction of nearly 43% in DSPs (3600 to 2069) and 62% in slices (56484 to 21841) is achievable for only 571 additional logic slices (or 1%). How these results are

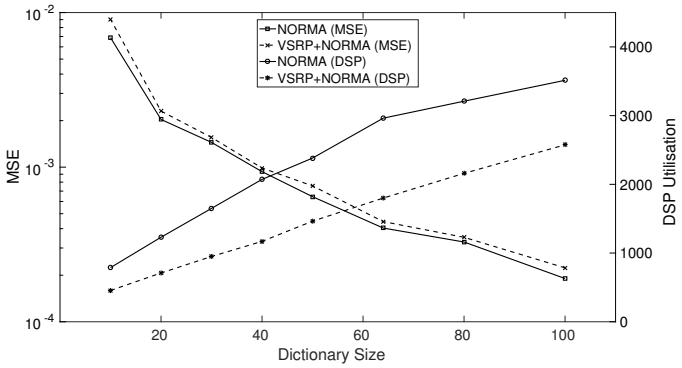


Fig. 4: Mean Square Error (MSE) and DSP Utilisation Trade-Off for NORMA and VSRP+NORMA, with m=16 and k=8

used depends on the practitioner and problem. In Subsection IV-B we give an example where we can achieve both improved accuracy and lower utilisation. The small overhead in logic also means the random projection module can operate at a significantly higher frequency than NORMA. This broadens the scope of possible applications to domains which are latency sensitive, such as network monitoring. However, since VSRP takes 30+1 clock cycles to generate a valid output and NORMA's operating frequency is about a quarter of VSRP, an additional 8 NORMA cycles are required. This outcome is mainly because our architecture only streams one feature per cycle, i.e. 24-bits, which doesn't optimise the input bandwidth of most systems. Double data-paths are an easy solution and involve adjusting each block in Figure 2 to handle two features in parallel.

B. Trade-Off and Benefits: Regression

Our second experiment involves the Mackey-Glass benchmark, which involves predicting the time series generated by the chaotic differential equation $dx(t)/dt = ax(t) + bx(t\tau)/(1 + x(t\tau)^{10})$ with ($a = 0.1$, $b = 0.2$ and $\tau = 30$). Our dataset contains 5000 examples from the KAFBOX [29] implementation, and we set up our model to predict the next time step given the previous 16 inputs. We use Mean Square Error (MSE) to measure performance and our results are given in Figure 4. Twin y-axes are preferred so that we can analyse the key trade-off between MSE and resources, given by DSP utilisation. We give plots for NORMA (continuous) and NORMA+VSRP (dashed), where our VSRP module reduces the input from 16 to 8 features. The MSE line indicates there is only a small additional predictive error using VSRP, whereas approximately 30-45% of DSPs can be saved at any point. This can be crucial for hardware platforms constrained by resources and creates an opportunity to take advantage of trade-offs between resources for increased dictionary items and reduced error. In fact, for 2000 DSPs, NORMA has capacity for only 40 dictionary elements on our hardware, whereas VSRP+NORMA can operate with twice as many elements (approximately 80) and achieve $5\times$ better accuracy.

C. System-Level Implementation: Novelty Detection

The VSRP+NORMA core was implemented on an ExaNIC X4 low-latency network card. The platform has four 10 GbE interfaces on a PCIe based host and can connect to four SFP (Small Form-Factor Pluggable) ports. We used a Chisel interface that is different from most hardware interfaces to integrate with the ExaNIC X4. The use of Scala allows very complex and parameterised structures to be expressed easily by taking advantage of the power of modern programming such as data structures and inheritance. Our interface exploits Scala inheritance by allowing any user module to inherit its IO structure from the CHISEL ExaNIC interface which is simply a wrapper for the ExaNIC FPGA Development Kit. This provides a clean way to import and use the interface instead of modifying an example project. It is our view that hardware with inheritance has the potential to improve code reusability and simplify user projects. As a proof of concept, our CHISEL interface is available at https://github.com/da-steve101/chisel-utils.git.

Our target application is novelty detection. We base our results on an artificial dataset which was generated using the *Scikit-learn* Python package [15]. The dataset contains 635 examples and only 8 features. We could only experiment with 8 features, $m = 8$, and 14 dictionary elements, $|D| = 14$, due to resource constraints. As a consequence of having few dictionary entries, NORMA only achieves an AUC score of 0.58, which indicates that the predictions are only slightly better than chance. Using the VSRP module to reduce the dimensionality to $k = 4$, we can increase the number of dictionary entries by $1.8\times$, to $|D| = 25$, for the same resources, which improves AUC to 0.65. If random projections is combined with a larger FPGA then significantly better predictive performance is expected.

The latency of our novelty detector is tested using the configuration given in Figure 5, where network traffic is received by a host on port 1. To simulate this setup, we use a single CPU as both the host and user, and send a single m length input vector, containing 24-bit words, from the user at Tx0 to the host at Rx1. The ExaNIC development kit automatically pads the data so that a minimum of 64 bytes is sent, which is a standard for Ethernet frames. The data path either loops back via the CPU host, or novelty detection is performed on the FPGA. In both cases, our system is designed to send the output back to the user immediately after completion. This is achieved on the FPGA by sending an initial 8 byte word which asserts the start of frame sequence. When the output is valid, we can immediately assert the end of frame and transfer a second word containing the 1-bit result from our novelty detector. The ExaNIC transmission logic appends an extra 4 bytes for error correcting capabilities. Therefore, 20 byte frames are sent from Tx1 in total. We use hardware timestamps in the ExaNIC to capture the time taken from when the first byte of data arrives at Rx1 to when the first byte is received at Rx0. We refer to this as the latency for one input vector.

TABLE III: Comparison of FPGA and CPU based novelty detectors on an ExaNIC X4 for one input vector

Impl.	m	k	$\|D\|$	Slices (/25350)	DSPs (/600)	Total Latency (us)	Reduction (×)	Core (us)
NORMA								
CPU (C)	8	-	14	-	-	10.20	-	0.502
FPGA	8	-	14	14058	526	0.376	27.13	0.176
VSRP+NORMA								
CPU (C)	8	4	25	-	-	10.51	-	0.805
FPGA	8	4	25	14115	558	0.404	26.01	0.204

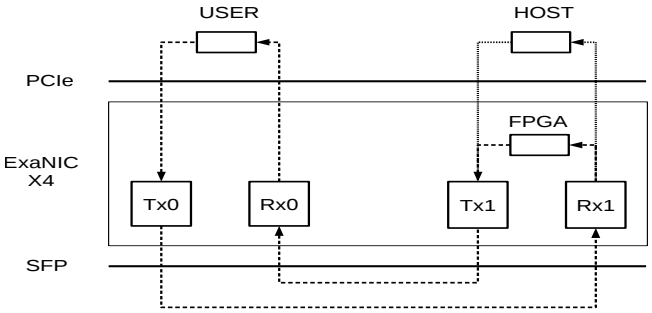


Fig. 5: Block diagram of the system implementation on a ExaNIC X4 network card

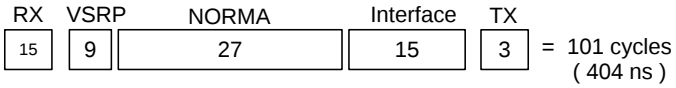


Fig. 6: Latency of the VSRP+NORMA system-level implementation in cycles

A summary of our latency results is given in Table III, and a breakdown of the latency of the VSRP+NORMA core is illustrated in Figure 6. The number of cycles is based on the ExaNIC's main clock frequency of 250 MHz. As described in Section III, the VSRP module completes in $m + 1$ cycles, where $m = 8$ is the input vector length. For $k = 4$ and $|D| = 25$, the NORMA module has 11 cycles of latency on a slower 104.2 MHz clock. This translates to approximately 27 cycles on the 250 MHz clock. The remaining latency is attributed to several pipeline registers for meeting place and route timing requirements as well as two asynchronous FIFO blocks for crossing clock domains. In the latter case, we used the Xilinx FIFO36E1 primitive. This added 3 and 2 cycles on the slower clock, for stepping down and up respectively. The Rx and Tx interfaces contribute an additional 60 ns and 140 ns, or 15 and 35 cycles, respectively. The CPU implementation has a total latency of $10.51\ \mu s$, this being $26\times$ higher than the FPGA implementation.

The bottleneck to system throughput in this implementation is the random projection module since it is not fully pipelined. This consumes a single 24-bit word each clock cycle at 250 MHz, and is equivalent to a throughput of 6.0 Gb/s which is approximately 60% of the data rate on the I/O ports of the ExaNIC. In contrast, the NORMA module can process 192-bit input vectors each cycle on a 104.2 MHz clock (for $m = 8$ and 24-bit words), corresponding to a throughput of 20 Gb/s. Even though NORMA can easily keep up with 10GbE line rates, our integrated design must wait for the VSRP module to finish computing the random projection.

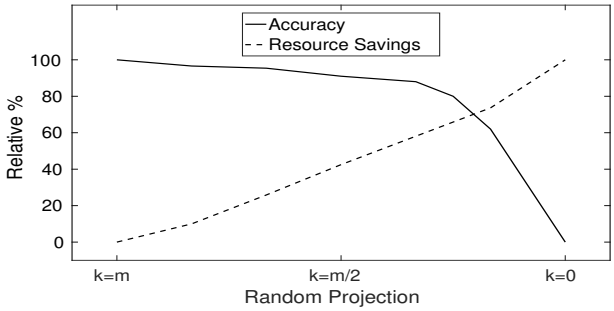


Fig. 7: Summary of results based on relative percentages for varying sized random projections

The throughput of VSRP+NORMA could be further improved in two ways. Firstly, only a single feature is transferred per clock cycle (i.e. 24 bits). This does not optimise the bandwidth of the ExaNIC X4 which can support 64 bit transfers with a 250 MHz clock. For best efficiency, our VSRP architecture could utilise triple data paths to saturate the bus. This would produce an output in fewer cycles, having the potential to increase throughput by $2.67\times$. Also, FPGA implementations of NORMA and VSRP are far better suited to high dimensional problems due to increased parallelism. While our system-level implementation provides a useful proof of concept, more significant performance gains can be obtained on larger FPGAs that can tackle bigger problems.

D. Summary

Figure 7 gives a summary of the VSRP+NORMA results (not including the system interface). We aggregated the outcomes for each experiment and extracted an average trend for the two key performance indicators. The accuracy curve of VSRP+NORMA refers to a percentage of NORMA's accuracy alone ($k = m$), and resource savings are given as a percentage increase in DSP blocks. We can see that a random projection which halves the input dimension, $k = \frac{m}{2}$, can generate an additional $\pm 40\%$ of computational resources for a small, $\pm 10\%$, error term. The savings in resources can be re-applied for increased model capacity and improved performance, as in Section IV-B. In the general case, random projections offer

an effective dimensionality reduction technique with efficient hardware implementations. Our experiments are based on datasets which have highly correlated variables and are hence well suited to this technique. Of course, the size, sparsity and error tolerance of the projection must be chosen in a problem-specific manner.

V. Conclusion

This paper demonstrated the utility of employing random projections for dimensionality reduction. Such an approach can be used to improve as well as trade off model capacity, processing time and hardware requirements for kernel adaptive filters. A very sparse random projection module was described which only involved multiplications by $+1, 0$ and -1. This module has a very efficient FPGA implementation involving only adders, shift registers and bit operations. Results involving classification, regression and novelty detection show that it is possible to simultaneously improve performance and reduce resource utilisation with little reduction in accuracy. Our baseline system level novelty detector achieved latency of $376\ ns$ without random projection. A design having similar resource utilisation and operating frequency but employing random projections allowed the dictionary size to be increased from 14 to 25, and the area under the receiver operating characteristic curve (AUC) to be improved from 0.58 to 0.65, with a degradation in latency to $404\ ns$.

Acknowledgments

This work was supported under the Australian Research Councils Linkage Projects funding scheme (project number LP130101034), and the Faculty of Engineering and Information Technologies, The University of Sydney by the Faculty Research Cluster Program.

References

[1] P. Drineas and M. W. Mahoney, "RandNLA: randomized numerical linear algebra," *Communications of the ACM*, vol. 59, no. 6, pp. 80–90, 2016.

[2] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[3] N. J. Fraser, D. J. Moss, J. Lee, S. Tridgell, C. T. Jin, and P. H. Leong, "A fully pipelined kernel normalised least mean squares processor for accelerated parameter optimisation," in *2015 25th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–6, IEEE, 2015.

[4] S. Tridgell, D. J. Moss, N. J. Fraser, and P. H. Leong, "Braiding: A scheme for resolving hazards in kernel adaptive filters," in *Field Programmable Technology (FPT), 2015 International Conference on*, pp. 136–143, IEEE, 2015.

[5] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3. JHU Press, 2012.

[6] B. D. Rouhani, E. M. Songhori, A. Mirhoseini, and F. Koushanfar, "Ssketch: An automated framework for streaming sketch-based analysis of big data on FPGA," in *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on*, pp. 187–194, IEEE, 2015.

[7] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a hilbert space," *Contemporary mathematics*, vol. 26, no. 189-206, p. 1, 1984.

[8] D. Achlioptas, "Database-friendly random projections," in *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 274–281, ACM, 2001.

[9] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 245–250, ACM, 2001.

[10] S. Dasgupta, "Experiments with random projection," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pp. 143–151, Morgan Kaufmann Publishers Inc., 2000.

[11] P. Li, T. J. Hastie, and K. W. Church, "Very sparse random projections," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 287–296, ACM, 2006.

[12] S. Dasgupta and A. Gupta, "An elementary proof of the Johnson-Lindenstrauss lemma," *International Computer Science Institute, Technical Report*, pp. 99–006, 1999.

[13] R. I. Arriaga and S. Vempala, "An algorithmic theory of learning: Robust concepts and random projection," in *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pp. 616–623, IEEE, 1999.

[14] P. C. Hansen, "Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank," *SIAM Journal on Scientific and Statistical Computing*, vol. 11, no. 3, pp. 503–518, 1990.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[16] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. Wiley Publishing, 1st ed., 2010.

[17] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE transactions on signal processing*, vol. 52, no. 8, pp. 2165–2176, 2004.

[18] A. Das, S. Misra, S. Joshi, J. Zambreno, G. Memik, and A. Choudhary, "An efficient fpga implementation of principle component analysis based network intrusion detection system," in *Proceedings of the conference on Design, automation and test in Europe*, pp. 1160–1165, ACM, 2008.

[19] C.-S. Bouganis, I. Pournara, and P. Y. Cheung, "Efficient mapping of dimensionality reduction designs onto heterogeneous FPGAs," in *15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2007)*, pp. 141–150, IEEE, 2007.

[20] R. M. Wang, G. Cohen, K. M. Stiefel, T. J. Hamilton, J. C. Tapson, and A. van Schaik, "An FPGA implementation of a polychronous spiking neural network with delay adaptation," *Frontiers in neuroscience*, vol. 7, p. 14, 2013.

[21] A. Coates and A. Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 921–928, 2011.

[22] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, pp. 1177–1184, 2007.

[23] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[24] K. L. Clarkson and D. P. Woodruff, "Numerical linear algebra in the streaming model," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 205–214, ACM, 2009.

[25] D. M. Kane and J. Nelson, "Sparser Johnson-Lindenstrauss transforms," *Journal of the ACM (JACM)*, vol. 61, no. 1, p. 4, 2014.

[26] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović, "Chisel: constructing hardware in a scala embedded language," in *Proceedings of the 49th Annual Design Automation Conference*, pp. 1216–1225, ACM, 2012.

[27] Exablaze, "ExaNIC X4 website." https://exablaze.com/exanic-x4, 2016. [Online; accessed 14-July-2016].

[28] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[29] S. Van Vaerenbergh, "Kernel methods toolbox KAFBOX: a Matlab benchmarking toolbox for kernel adaptive filtering," *Grupo de Tratamiento Avanzado de Senal, Departamento de Ingenieria de Comunicaciones, Universidad de Cantabria, Spain*, 2012.